GRF Tools Suite

A Collection of Tools for the Routine Processing of GRF Data

Revision 1.2.0

May 2003

Banfill Software Engineering

Valdez, Alaska

(907) 835-4122

http://www.banfill.net/

Table of Contents

Table of Figures

Chapter 1

Introduction

1.1 Overview

The GRF Tools Suite is a collection of software tools that handle waveform data in the Generic Recording Format or GRF. These tools provide basic processing functionality across various platforms that support TCP/IP networking and are the reference implementation of the GRF.

Unless otherwise noted, all of the tools are currently available on the following platforms:

- 32-bit Windows® (98, ME, NT, 2000, XP) on Intel® hardware
- Linux® 2.2.x or later on Intel hardware

The suite currently consist of the following tools:

- **GRFHub** – This is the GRF hub server, a client/server application acts as a data concentrator. GRFHub connects to other GRF servers such as digitizers running GRFd, brings these data together, and serves them to other clients. Additionally, these data may be stored to a local repository.

  GRFHub also has the ability to monitor USGS DST streams, convert these data to GRF, and incorporate these data in to the flow of GRF data.
- **GRFTrig** – This is the GRF Trigger. This is a hub server like GRFHub except that triggered event data are served to clients and/or stored in a repository. A fully general network trigger is implemented which uses modular channel triggers of various types.
- **GRFLog** – This is the GRF logger client application, which displays the contents (logs) of GRF packets in various ways. Various statistical analyses of the packets can also be performed and reported.
- **GRFConvert** – This is the GRF data converter, a client application that converts GRF data to various standard analysis formats. Among the currently supported formats are GSE2.0, MATLAB, SAC, SEG-Y, Mini-SEED, and PC-SUDS.
- **GRF2EW** – This is the GRF to Earthworm version 6.1 data source module. This GRF client application converts GRF waveform data to Earthworm TraceBuf packets and then places them on an Earthworm Ring.
- **GRFCheck and DSTGen** – GRFCheck is the GRF data checker, an application that performs integrity checks on GRF packet image files. DSTGen is a simple DST function generator for generating synthetic DST stream data.

A chapter is dedicated to each tool and that chapter is the reference documentation for that particular tool.

1.2 The GRF port number

As of version 1.2.0, the Internet Assigned Numbers Authority (IANA) has assigned both the TCP and the UDP registered port number 3757 for use by GRF applications. Note that the old ephemeral port number (49642) is no longer used as the default GRF port number by the tools in this suite.

All GRF servers in the GRF Tools Suite now listen on port 3757 for client connections by default and all client applications with the suite will attempt to connect to this port by default. For all applications in the suite, the port number is still user configurable and may be changed as required.

1.3 ISO8601 Representation of Date and Time

As of version 1.0.8, all of the GRF Tools applications use ISO8601 date and time representations exclusively. The ISO8601 standard provides a common and unambiguous way to represent dates and times. All of the details about this standard are available in the ISO8601:2000(E) standards document which can be obtained directly from the International Organization for Standardization (ISO) at http://www.iso.org.

The GRF Tools applications support the *calendar*, or month/day, representation and the *ordinal*, or day-of-year, representation. The *week* representation is not supported at this time. The user specifies which representation will be used by setting the `DATE_FORMAT` environment variable, `DATE_FORMAT=Calendar`, or `DATE_FORMAT=Ordinal`. If this variable is set to some other value, or if it is not set at all, the representation defaults to ordinal.

Both formats use the *extended* representation, meaning that various delimiter and designator characters are present within the representation.

Ordinal representations are of the form:

```
yyyy-dddThh:mm:ss.ssssssZ
```

Calendar representations are of the form:

```
yyyy-MM-DDThh:mm:ss.ssssssZ
```

Where `yyyy` is the four digit year, `ddd` is the three digit ordinal day number (day-of-year), `MM` is the two digit month, `DD` is the two digit calendar day (day-of-month), the '`T`' character is the time field designator, `hh` is the two digit hour of day, `mm` is the two digit minute of hour, `ss.ssssss` is the two digit second of minute with up to six digits to the right of the decimal point showing fractional seconds, and the trailing '`Z`' character indicates that times are UTC.

1.4 Installation

This section describes how to install the GRF Tools Suite onto the host computer. Regardless of platform, the GRF tools reside in a series of subdirectories under a *home* directory. The GRF home directory is the directory where you will install the GRF Tools Suite below. Several subdirectories are created during the install procedure:

- **bin** – This directory is where the executable files for the tools are stored.
- **etc** – This is where ancillary files are stored. This includes the various configuration files, release notes, and scripts that are used by the tools.
- **doc** – This is where documentation is stored. Look here for the various manuals and other documents pertaining to the system. Generally, all documentation is in the Adobe® portable document format and is viewable using the Adobe Acrobat® Reader® software that is provided on the distribution CD-ROM and is available free of charge at Adobe's website.

There may be other subdirectories in the GRF home directory depending what other GRF software is installed.

To install the GRF Tools Suite on your system, follow the appropriate procedure for your platform.

Windows® Systems

The 32-bit Windows (Win32®) versions of the various GRF tools can run on Windows 98 1st and 2nd editions, Windows ME, Windows NT® 4.0 SP4 or later, Windows 2000, and Windows XP. Although the tools may run on older and/or newer versions of Windows, they have not been tested in those environments. Please check the GRF home page (see below) for the latest information about running the GRF tools on newer versions of Windows.

The GRF Tools for Windows systems are provided in a self-extracting archive file named `grftools-win32-1_2_0.exe` that is located in the `\install` directory on the distribution CD-ROM. To extract the contents of this archive to your hard disk, simply double click on the `grftools-win32-1_2_0.exe` file.

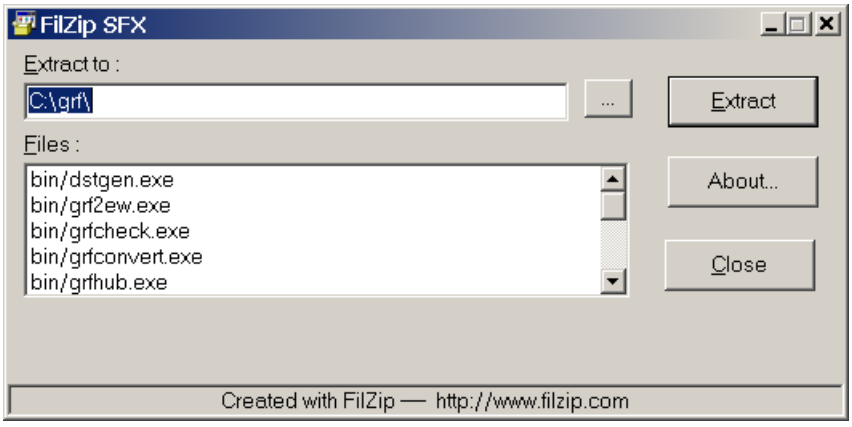

Figure 1 GRFTools self-extracting archive dialog box.

The archive extraction dialog box appears and the 'Extract to' or GRF home directory is set to `C:\grf` by default. It is highly recommended that you install the GRF tools in this directory unless you have specific reason not to. If you choose to install the tools there,

simply click 'Extract' or press return. If you will install the tools elsewhere, type the destination pathname and then click 'Extract' or press return.

A batch program, `etc\environ.bat`, is provided in the distribution that will set up the environment for the GRF tools and the Java™ 2 Runtime Environment (J2RE) if it is present. This batch program needs to be executed at the command prompt before running the GRF tools. `etc\environ.bat` sets several important environment variables and if you installed the GRF tools somewhere other than the default location, `c:\grf`, you will need to edit this file to reflect the actual GRF home directory. You can also set the `DATE_FORMAT` environment variable to select the ISO8601 date and time representation that will be used by the tools.

environ.bat

```
@echo off

echo Setting up GRF tools environment...

rem - Adjust the home directory as appropriate for your system...
set GRF_HOME=c:\grf

rem – Set the ISO8601 date and time representation...
rem set DATE_FORMAT=Ordinal
set DATE_FORMAT=Calendar

rem - These should not be changed...
set GRF_BIN=%GRF_HOME%\bin
set GRF_ETC=%GRF_HOME%\etc

set PATH=%GRF_BIN%;%PATH%

rem - Setup Java 2 Runtime Environment if present...
if exist %GRF_ETC%\j2re_env.bat call %GRF_ETC%\j2re_env.bat
```

environ.bat

Figure 2 Contents of the `etc\environ.bat` file.

It is highly recommended that you modify the shortcut that you use to start a command prompt to automatically execute the `etc\environ.bat` batch program automatically.

Windows NT and 2000 systems

1. Right click on the 'Command Prompt' icon on your desktop and select 'Properties' from the pop up menu.
2. On the 'Shortcut' tab, in the 'Target' box, append `/k c:\grf\etc\environ.bat` to the command there. If you installed somewhere other than `c:\grf`, substitute the appropriate path for `c:\grf`. The 'Target' string should look something like this:

   `%SystemRoot%\system32\cmd.exe` **`/k c:\grf\etc\environ.bat`**
3. The bold text is the text that you should type. Click the 'OK' button to close the shortcut properties dialog box.
4. Double click the 'Command Prompt' icon on the desktop. You should see the following messages just before the command prompt is displayed:

   ```
   Setting up GRF tools environment...
   ```
5. Type **`grflog -h`** at the command prompt. You should see the grflog help screen.

Windows 98 and ME systems

1. Right click on the 'MS-DOS Prompt' shortcut that you will use when running the GRF tools and Earthworm. Select 'Properties' from the pop up menu.
2. On the 'Program' tab, enter **`c:\grf\etc\environ.bat`** in the 'Batch file:' text box. If you installed somewhere other than **`c:\grf`**, substitute the appropriate path for **`c:\grf`**.
3. On the 'Memory' tab, select '2048' in the 'Initial environment' drop down box.

4. Click 'OK' to close the properties dialog box.
5. Double click the 'Command Prompt' icon on the desktop. You should see the following messages just before the command prompt is displayed:
   ```
   Setting up GRF tools environment...
   ```
6. Type **`grflog -h`** at the command prompt. You should see the GRFLog help screen.

Linux Systems

While there are many different methods that one can use to set up the GRF tools on a Linux system, we recommend a setup that is limited to a single user to reduce the impact on the system as a whole and which generally does not require that user to work as the superuser.

The Linux versions of the tools can run on most Linux distribution with kernels later than 2.2 but the software has been developed and tested on Red Hat® Linux versions 6.1 through 7.3. The GRF Tools Linux executables are distributed in a compressed (gzip) tape archive (tar) file named `grftools-linux-x86-1_2_0.tgz`. To install these tools in your home directory:

1. Login as the user that will run the GRF tools. Make sure that your home directory is the current working directory. Note that the text in bold type is the command that you should type.
   ```
   linux % cd
   ```
2. Insert the CD-ROM into the drive and mount it. Typically, the command will be:
   ```
   linux % mount /mnt/cdrom
   ```
3. If the CD-ROM is mounted somewhere other than /mnt/cdrom, substitute that mount point in the command below. Unpack the archive:
   ```
   linux % tar -zxvf /mnt/cdrom/install/grftools-linux-x86-1_1_0.tgz
   ```
4. Edit your shell startup script to call one of the environment setup scripts provided. If you use bash, edit your `~/bashrc` to source `~/etc/environ.bash`. If you use the C shell, edit `~/cshrc` to source `~/etc/environ.csh`. You will need to edit these files if you installed to a directory other than you home directory. The `GRF_HOME` environment variable should point to your home directory.
5. Open a new terminal and type **`grflog -h`**. You should now see the GRFLog help screen.

1.5 The GRF Home Page

Banfill Software Engineering maintains the GRF home page at http://www.banfill.net/grf.html. Go to this page to:

- Get the latest news and information about the GRF.
- Get the latest GRF Tools Suite and other software and/or services.
- Make suggestions, submit software, and/or otherwise participate in the development of the GRF.
- Get GRF unit identifier number assignments.

Contacting Us

Our offices are located in Valdez Alaska. We can be reached at:

Banfill Software Engineering

P.O. Box 462
Valdez, AK 99686-0462 USA
(907) 835-4122 Voice
http://www.banfill.net
mailto:info@banfill.net

Chapter 2

GRFHub

2.1 Overview

GRFHub is a client/server application that concentrates data from many sources, optionally stores these data to a repository, and then serves these data clients. Data sources are other GRF server applications such as GRFd, other GRFHub's, and/or USGS-DST style digital streams. Client applications might include GRFConvert, GRFTrig, and/or other GRFHub's.

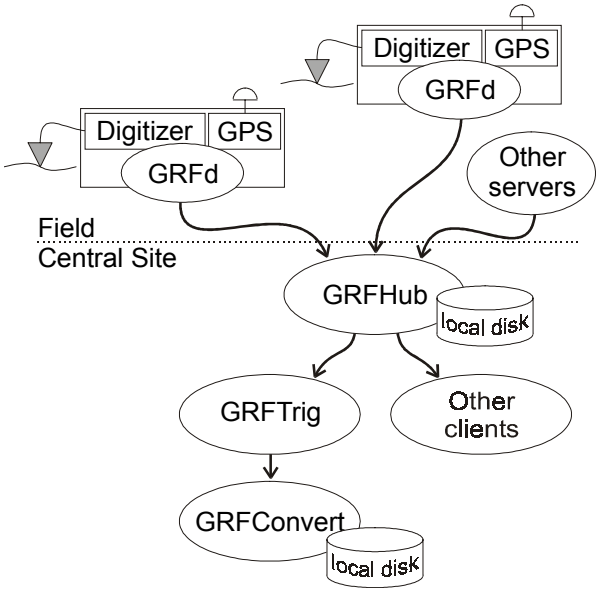

Figure 3 Schematic view of a simple network illustrating the role of GRFHub.

The figure shows a simple network consisting of several field digitizers, such as NetDAS[1], and a simple central recording site where the data are concentrated and stored to disk, network triggering is performed, and events are convert to a standard analysis format and stored to disk.

In the figure, everything above GRFHub is a source of incoming data. GRFHub is connected to each of these servers as client and is receiving waveform data from each. Each

[1] NetDAS is a powerful GRF based field digitizer unit available from DAQ Systems, http://www.daqsystems.com, (406) 586-8028.

line in the figure represents a TCP socket connection. The 'other servers' shown in the figure might even include other GRFHub's.

Everything below GRFHub is a client. The figure shows GRFTrig, the GRF trigger connected to GRFHub and it will receive waveform data from all three of the upstream servers through the hub, trigger on those data, and then serve event data to GRFConvert, which will then convert these event records into a standard analysis format and store them on disk.

Many other clients might be connected simultaneously and receive the same data from the hub. For example, GRF2EW may be connected and feeding the wave ring on an Earthworm system, or GRFConvert may be connected directory to the hub and convert continuous records to a standard format and store them on disk. The figure shows only one of many possible configurations.

2.2 Running GRFHub

GRFHub can be run from the command line as a console application or in the background as a dæmon. The program is controlled through an ASCII configuration file. The program also accepts several command line arguments and a simple help screen is available by typing **`grfhub -h`** at the command line.

As there are some differences in how you start and stop GRFHub on different platforms, we'll cover each separately below.

Windows Systems

On most Windows systems, you have two choices for how you will run GRFHub. You can simply run it from the command line and press Ctrl-C or Ctrl-Break to stop it.

If your system is running Windows NT or 2000, you can install GRFHub as a *service*. Services are not available on Windows 9*x* or ME. A service is a program that runs in the background much like a dæmon on UNIX-like systems. Services are running on the system even when no user is logged in. Services can be controlled from the Services applet in the Control Panel. GRFHub allows you to control its service directly from the command line.

Below is the help screen from the Win32 version of GRFHub:

```
C:\grf>grfhub -h
GRFHub version 1.2.0 (May  1 2003 16:04:08), pid:696

Usage: grfhub [[-hvqdb] [-l logfile] [configuration file]] |
              [-install | -remove | -start | -stop | -status]

   -h           Help display.
   -v           Verbose logging.
   -q           Quiet logging.
   -d           Debug logging.
   -b           Run in background. (No)
   -l logfile   Log to logfile. (Off)

   -install     Install as a Win32 service.
   -remove      Uninstall Win32 service.
   -start       Start Win32 service.
   -stop        Stop Win32 service.
   -status      Report Win32 service state.

The configuration file defaults to ./grfhub.conf if not specified.

[] = optional, () = default, | = mutually exclusive.
```

Before GRFHub can be run as a service, it must be installed as a service. This is accomplished using the `-install` command line option.

```
C:\grf>grfhub -install
Successfully installed grfhub version 1.2.0 as a service.
```

Once installed, the GRFHub service will start automatically every time that the system is started and will remain installed until it is removed. To remove the service, use the `-remove` command line option.

To start the service from the command line, use the `-start` option:

```
C:\grf>grfhub -start
grfhub service start is pending...
grfhub version 1.2.0 service is running.
```

If there was an error or if the service is not yet installed an error message will be reported here. To stop the service from the command line, use the `-stop` option. The service can also be started and stopped by using the Services applet from the Control Panel and/or the `net start grfhub` and `net stop grfhub` commands at the command prompt.

The status of the service will be reported by using the `-status` option. For example, if the service is installed but not currently running:

```
C:\grf >grfhub -status
grfhub version 1.2.0 service is currently stopped.
```

If the service has not been installed or has been removed:

```
C:\grf >grfhub -status
ERROR: OpenService(grfhub): The specified service does not exist as an
installed service.
```

Linux Systems

GRFHub can be executed in the foreground from the command line as a console application or run as a dæmon in the background. When running in the foreground, press Ctrl-C to stop the program.

Run the program in the background by specifying the `-b` option on the command line. This causes the program to detach from the terminal and run as a dæmon process. The help screen for the Linux version of the software follows:

```
linux % grfhub -h
GRFHub version 1.1.0 (Oct 22 2002 11:20:09), pid:22388

Usage: grfhub [-hvqdb] [-l logfile] [-s facility] [configuration file]

   -h           Help display.
   -v           Verbose logging.
   -q           Quiet logging.
   -d           Debug logging.
   -b           Run in background. (No)
   -l logfile   Log to logfile. (Off)
   -s facility  Log to syslog facility, `LOCAL0'-`LOCAL7'. (Off)

[] = optional, () = default, | = mutually exclusive.
```

When the program is running as a dæmon, it must be stopped by sending the TERM signal. This is easily accomplished using either the `kill` *`pid`* command where *`pid`* is the process identifier or by using the `killall grfhub` command.

GRFHub can use the system logger on your system for its log output. This is particularly useful when running the program as a dæmon as the log output can be managed automatically by the system. The eight local syslog facilities are supported, `LOCAL0` through `LOCAL7`. One of these facilities should be configured for use by the GRF tools software. For

example, to have syslogd maintain the file `/var/log/grf.log` as facility `LOCAL3`, you would add the following lines to syslogd's configuration file (typically `/etc/syslog.conf`):

```
# GRF Tools log output
local3.*   /var/log/grf.log
```

After restarting syslogd, the `LOCAL3` facility will be available for use. For more information about the system logger on your system, consult `/etc/syslog.conf` and/or the 'syslogd' manual pages.

2.3 Command Line Options

In this section, we cover the command line options that are common between GRFHub on all supported platforms. These options have to do with control of log output and specifying the name of a configuration file. The configuration file is covered in the next section.

As shown in the previous sections, the `-h` option displays a small help screen on the console.

Logging Options

These options can be specified within the configuration file. If they appear on the command line, those settings override the settings within the configuration file.

The `-l` *`filename`* option is used to specify the name of the file to write log output to. When running as a console application, all log output is written to the console. If this option specifies a filename, all output is written to the console and to the specified file.

The `-q`, `-v`, and `-d` options are used to specify *quiet*, *verbose*, and *debug* logging levels respectively. The default logging level is *normal*. Quiet logging specifies that only errors and other critical information be output. Normal logging includes some additional informational messages. Verbose logging includes much more detailed informational messages. Debug logging includes extremely detailed messages about the internal workings of the program.

2.4 GRFHub Configuration

GRFHub is controlled primarily through its configuration file. This configuration file allows you to configure and control all aspects of the programs behavior. This simple ASCII file can be edited using your favorite text editor.

This file can be specified on the command line at program startup or the program will search for a default configuration file named `grfhub.conf`. If the configuration file was not specified on the command line, the program will look for `grfhub.conf` in the directory pointed to by the `GRF_ETC` environment variable. If that variable is not set, Windows systems will then look for `grfhub.conf` in the current working directory, UNIX-like systems will look for `/etc/grfhub.conf`.

The configuration file is made up of keywords followed by an entry. Some keywords define groups of entries and curly braces {} are used to delimit these groups. White space characters or commas separate keywords and entries. Comments may appear anywhere in the file and always begin with a pound sign '#'.

The file is made of four basic groups of entries:

- **Logging** –This group configures and controls the programs logging behavior.

- **Connections** – This group defines upstream client connections that the hub will maintain.
- **GPS** – This group configures a GPS receiver for use as source of timing information for DST streams.
- **Server** – This group configures the GRF server.

Throughout the example configuration file, the default values for each entry are shown as end-of-line comments inside parentheses.

Top-level entries

These entries are called 'top-level' entries because they appear outside of any configuration group and generally appear near the top of the configuration file.

grfhub.conf – top level entries

```
# $Id: grfhub.conf,v 1.13 2003/03/27 19:41:47 cvs Exp $ */
# Copyright (C) 2000-2003 - Robert Banfill - All rights reserved.
# Configuration file for grfhub version 1.2.0 or later.
# Defaults for each entry are shown in ()

Daemon              No          # Run in background? (No)
PIDFile             /var/run/grfhub.pid # Process lock file (None)
DateFormat          Ordinal     # Calendar or (Ordinal), overridden by the
                                # DATE_FORMAT environment variable.
```

grfhub.conf – top level entries

Figure 4 grfhub.conf top-level entries.

Note that the 'Dæmon' and 'PIDFile' entries only have meaning on UNIX-like operating systems. See §2.1, Running GRFHub, above for a discussion about running GRFHub as a service on 32-bit Windows systems.

The 'Dæmon' entry is used to specify whether the program will run in the background as a *dæmon* and is equivalent to the `-b` command line option. By setting this entry to 'Yes', this becomes the default behavior of the program. Note the end-of-line comments starting with the pound sign (#). The 'Dæmon' entry specifies that the program should run in the background. The 'No' in parenthesis at the end of the comment indicates that if this entry is set to 'No' or the entry is not found in the configuration file, the program will not run as a dæmon.

The 'PIDFile' entry specifies the name of a lock file used by the program to ensure that only one instance of the server is allowed to run at a time. The file is created at startup and the process identifier (PID) is written into the file as ASCII text. If the file cannot be created, the program will exit indicating an error. If this entry does not appear in the configuration file, the program will not perform this process locking operation. Note that the user executing the program must have create, read, and write permission for this lock file and that the filename should be specified as an absolute path, i.e., it should begin with a solidus (/) character, to avoid an dependency on the current working directory at startup.

The 'DateFormat' entry is used to specify the default ISO8601 date and time representation (see §1.1) that will be used by the program. Valid entries include 'Ordinal' and 'Calendar' with 'Ordinal' being the default value. If the `DATE_FORMAT` environment variable is set, its setting overrides this setting in the configuration file.

The 'Logging' group

This group contains entries that configure how log output is handled by the program. These entries are equivalent to their command line counterparts discussed in the previous section.

The following example sets the logging level to *verbose* and directs log messages to a file named grfhub.log in the /grf/log directory.

grfhub.conf – logging group

```
# This group defines how logging is handled...
Logging {
#    Syslog          local3      # Use syslog facility, LOCAL0 through LOCAL7 (Off)
    File            /grf/log/grfhub.log # Log to file (Off)
#    Level           Quiet       # Be vewy, vewy quiet...
    Level           Verbose     # Output lots of info...
#    Level           Debug       # Debugging message spray...
}
```

grfhub.conf – logging group

Figure 5 grfhub.conf 'Logging' group entries.

The syslog entry is used to specify the syslog facility that will be used for log output. See the discussion about syslog in the previous section for more about using this feature.

The file entry is used to specify a log file specification. In addition to writing log messages to the console, log messages will also be written to this file. This setting can be overridden by the `-l` command line option.

There are four logging levels; `quiet`, `normal`, `verbose`, and `debug`, with normal being the default. Quiet logging restricts log output to warnings and error conditions only, normal, includes important informational messages, verbose includes much more detailed informational messages, and debug includes extremely detailed informational messages.

The 'Connections' group

The connections group defines a list of upstream connections to maintain and some attributes of those connections. These are client GRF socket connections and/or USGS-DST style digital stream connections through asynchronous serial interfaces.

Below is the connections group from the example configuration file provided in the distribution.

grfhub.conf – connections group

```
# Upstream client connections to maintain...
Connections {
    QueueDepth      128         # Depth of connection queue as packets (32)
    Timeout         30          # Socket I/O timeout in seconds (30)

    # GRF server endpoints to connect to...
    Endpoints {
        192.168.2.1
#        netdas.banfill.net:3757
    }

    # USGS-DST style digital streams to acquire...
#    DigitalStreams {
#        DataType        CM8     # Data type in GRF data packets (INT32), INT24, or CM8
#        SamplesPerPacket 500    # Maximum number of samples/packet (500)
#        Port {
#            Device      /dev/ttyS0
#            BaudRate    9600    # Port speed (9600)
#            Parity      None    # (None), Even, or Odd
#            DataBits    8       # 7 or (8)
#            StopBits    1       # (1) or 2
#            GRFUnitID   10000   # GRF unit ID number (Must be unique!)
#            SamplingRate 100.0  # Any supported by device (100.0)
```

```
#            ByteOrder   MSB/LSB # (MSB/LSB) or LSB/MSB
#            SyncByte    0x0d    # Sync byte value (0x0d)
#            AuxByte     Timing  # Aux byte value or 'Timing' (0x0a)
#            Channel 1 {
#                Name GRF:Dst1:SHZ # GRF channel name: network:station:component
#                CountsPerVolt 13107 # Digitizer constant (13107)
#                DCOffset 0      # DC offset in counts (0)
#            }
#            Channel 2 { Name GRF:Dst1:SHN, CountsPerVolt 13107, DCOffset 0 }
#            Channel 3 { Name GRF:Dst1:SHE, CountsPerVolt 13107, DCOffset 0 }
#        }
#        Port {
#            Device /dev/ttyS1, BaudRate 9600, GRFUnitID 10001, SamplingRate 100.0
#            Channel 1 { Name GRF:Dst2:SHZ }
#            Channel 2 { Name GRF:Dst2:SHN }
#            Channel 3 { Name GRF:Dst2:SHE }
#        }
#    }
}
```

grfhub.conf – connections group

Figure 6 grfhub.conf 'Connections' group entries.

The connections group consists of four keywords, one of which defines a list, and one that defines a sub-group.

The 'QueueDepth' entry

This entry is used to specify the number of GRF packets that can be stored in the queue between the upstream client threads and the server thread. This entry should not need to be changed under normal operating conditions.

The 'Timeout' entry

This entry is used to specify the timeout value in seconds for upstream client socket connections. If the network connections to upstream clients are very slow or have a very high latency, you may need to increase this value. Under normal operating conditions, the default value of 30 seconds should be appropriate.

The 'Endpoints' list

This list is used to specify a list of upstream server endpoints that will be connected to as clients. The program will create a client thread to service each of these connections and maintain them for the life of the program.

Endpoints may be specified as IP numbers in dotted decimal form or as a domain name. The port number may be specified by appending a colon (:) followed by the decimal port number to the address. If no port is specified, the default port (3757) is used. For example, '192.168.1.1' is equivalent to '192.168.1.1:3757'.

Any number of endpoints may be specified. There is no hard limit on the number of endpoints. It is only limited be available memory.

The 'DigitalStreams' sub-group

This sub-group is used to define one or more USGS DST style digital streams. These streams will be read, GRF data packets created, and these data packets will be incorporated with the data from all other upstream sources.

Please refer to Appendix B for information about the DSTGen program. This program can be used to generate synthetic function data simulating a single three channel DST station and can be very useful while configuring and testing DST streams.

The 'DataType' entry

This entry is used to specify the data type that the GRF packets generated from digital streams will contain. DST streams always consist of 16 bit signed data and these data will be promoted to 32 bit signed integers. It is recommended that you always set this entry to 'CM8' data compression so that the data are stored efficiently as possible.

The 'SamplesPerPacket' entry

Use this entry to specify the maximum number of samples that a single GRF data packet will contain. A GRF data packet is typically limited to 1024 bytes in length and when data is stored in compressed form this packet may contain as many as 780 or so samples. This can cause high latencies at lower sampling rates and can cause problems for some downstream client applications that are feeding other systems.

This value defaults to 500.

The 'Port' sub-groups

This sub-group defines a single DST stream. A DST stream consists of an asynchronous serial interface and its associated parameters, DST format parameters, and GRF channel names and unit ID number.

Currently, there is a limit of four on the number of DST ports that may be active simultaneously, however, this can be easily increased if needed by rebuilding the software. Please contact us if you need this limit raised.

The 'Device' entry

Use this entry to specify the device name of the asynchronous serial interface from which to read DST data. On UNIX-like systems this typically has the form, '/dev/ttyS*n*', where *n* is the zero based port number. On Win32 system, this has the form, 'com*n*', where *n* is the one based port number.

The 'BaudRate', 'Parity', 'DataBits' and 'StopBits' entries

Use these entries to specify the operating parameters of the asynchronous serial interface. These entries default to 9600 baud, no parity, eight data bits, and one stop bit, respectively.

The 'GRFUnitID' entry

Use this entry to specify the GRF unit ID number that will identify the digitizer that is the source of the data on this port. This number must be unique network wide. The GRF unit ID number is a 32 bit unsigned integer and it is recommended that you use numbers in the range of 20000 to 29999 inclusive for DST streams. For a current list of GRF assigned numbers please go to the GRF home page.

The 'SamplingRate' entry

Use this entry to specify the sampling rate at which the digitizer sourcing these data is operating. It is critical that this value is correct or the incoming data will be corrupted.

The 'ByteOrder', 'SyncByte' and 'AuxByte' entries

Use these entries to specify the DST parameters for the port. The ByteOrder entry specifies the order of the two bytes in each 16-bit sample. 'MSB/LSB' specifies that the most

significant byte comes first and is the default setting. This is commonly referred to as *big endian* or *network* byte order. 'LSB/MSB' specifies that the least significant byte comes first and is only supported on a few DST capable digitizers. This is commonly referred to as *little endian* or Intel byte order.

The SyncByte entry specifies the value of the DST synchronization byte. By default, this is 0x0D and should not need to be changed.

The AuxByte entry specifies the value of the DST auxiliary byte. By default, this is 0x0A. Some digitizers can encode a one pulse-per-second (PPS) signal in the most significant bit (bit 7) of the auxiliary byte. GRFHub can use this information along with an attached NMEA 0183 capable GPS receiver to correct the timing on incoming DST data. To activate these features, set the AuxByte entry to 'Timing' and see the GPS group below.

The 'Channel' sub-groups

These groups contain entries that define the GRF name and calibration data for each channel on the DST digitizer. Naming a channel activates that channel. Unnamed channels are not active and their data are not processed. DST compatible instruments generally output either one or three channels of 16-bit data.

Each channel sub-group contains one mandatory entry and two optional entries. The 'Channel' keyword followed by that channel number defines a channel sub-group. Note that all channel numbers are one based, that is, the first channel on the digitizer is channel 1, not channel 0.

The 'Name' entry

Each channel sub-group must contain a name entry followed by a valid GRF name. This entry activates the channel and assigns the GRF name to that channel. A GRF name consists of three colon delimited identifiers of the form, *network*:*station*:*component*, where:

network = Up to 7 character network identifier.
station = Up to 15 character station identifier.
component = Up to 7 character component identifier.

The 'CountsPerVolt' entry

This optional entry is used to specify the integer number of digital counts per volt for this channel. This value is stored in the GRF data packet header so that downstream software may convert the digital waveform data to volts.

This value defaults to $2^{16}/5.0$ (+/-2.5) or 13107.

The 'DCOffset' entry

This optional entry is used to specify the DC offset value in integer digital counts for this channel. This value is subtracted from each sample before encoding into GRF data packets.

This value defaults to 0.

The 'GPS' group

Entries in this group define the parameters needed to communicate with a NMEA 0183 compatible GPS receiver. The timing information gathered from the GPS receiver is used only to correct timing on data from DST streams (see above) and is not otherwise needed for operation. The GPS group from the example configuration file is given below.

grfhub.conf – gps group

```
# NMEA 0183 GPS device settings...
GPS {
    Port           /dev/ttyS1  # Port where GPS device is attached (/dev/ttyS0)
    Baudrate       4800        # 300 through 115200 (4800)
    Parity         None        # (None), odd, or even
    Databits       8           # 7 or (8)
    Stopbits       1           # (1) or 2
    Latency        240000      # Microseconds added to decoded GPS time (0)
}
```

grfhub.conf – gps group

Figure 7 grfhub.conf 'GPS' group entries.

The 'Port' entry

This entry specifies the device name of the asynchronous serial interface from which NMEA 0183 messages will be read. On UNIX-like systems this typically has the form, '/dev/ttyS*n*', where *n* is the zero based port number. On Win32 systems, the device name will be of the form, 'com*n*', where *n* is a port number between 1 and 128, inclusive.

The 'Baudrate', 'Parity', 'Databits' and 'Stopbits' entries

These entries specify the various communications parameters for the asynchronous serial interface used to communicate with the GPS receiver. The NMEA 0183 standard specifies these settings as 4800-baud with no parity; eight data bits, and one stop bit. These are the defaults values.

The 'Latency' entry

This entry is used to specify, in microseconds, the average relationship between PPS signal and the arrival of the corresponding NMEA RMC message. There is typically about a one quarter second (250000 microseconds) delay between the rising edge of the PPS signal and decode of the corresponding RMC message from the receiver.

The 'Server' group

The server group contains entries and sub-groups of entries that configure the behavior of the server in general. Below is the server group from the example configuration file provided in the distribution:

grfhub.conf – server group

```
# GRF Server settings...
Server {
    FriendlyName    GRFHub      # Name used to describe this server (Grfhub)
    GRFUnitID       500         # GRF unit ID for this server (Must be unique!)
    Endpoint        *:3757      # Listen at this endpoint, * = any interface (*:3757)
    MaxConnections  8           # Maximum allowed simultaneous client connections (8)
    QueueDepth      128         # Depth of client queues as packets (32)

    # Lists of hosts allowed or denied connection as data clients (Allow all)
#    AllowHosts {
#        localhost
#        192.168.1.0/24
#        192.168.2.0/24
#        192.168.3.0/24
#    }
#    DenyHosts {
#        192.168.1.2
#        192.168.2.2
#    }

    # Store waveform data to local repository (off)
#    Repository {
#        Path            /mnt/repository
       # File naming format specifier:
```

```
        #  %Y  = four digit year, %y = two digit year,
        #  %M  = two digit month, %D = two digit day of month,
        #  %d  = three digit day of year,
        #  %h  = two digit hour of day,
        #  %m  = two digit minute of hour,
        #  %s  = two digit integer second of minute,
        #  %ms = three digit integer millisecond of second,
        #  %us = six digit integer microsecond of second,
        #  %l  = record length as integer seconds (variable length),
        #  %lm = three digit integer millisecond of record length,
        #  %lu = six digit integer microsecond of record length,
        #  %u  = GRF unit number (variable length), %% = literal '%'.
        #  (ISO9660 compatable: %Y%d%h%m%s_%l_%u.grf)
#        NameFormat      "%Y-%d-%h-%m-%s.%ms_%l.%lm_%u.grf"
#        RecordLength    3600    # Record length in seconds
#        QueueDepth      128     # Depth of queue as packets (64)
#    }
}
```

grfhub.conf – server group

Figure 8 grfhub.conf 'Server' group entries.

The server group contains eight entries, two of which define lists and one that defines a sub-group. First, we examine the five non-group entries.

The 'FriendlyName' entry

This entry is used to assign a simple name to the server that will be used in informational messages. For example, when a client connects to this server, this name will be used in the connection message.

This name can be a maximum of sixteen characters in length and defaults to 'GRFHub' if this entry is not found in the configuration file.

The 'GRFUnitID' entry

Use this entry to assign a GRF unit ID number to this server. This will be the unit ID used in informational packets originate at this server. This is not the unit ID that contained in data packets that come through this server. Those data packets contain the unit ID of the digitizer that created the packets.

The unit ID defaults to zero and it is recommended that settings for this entry be in the range from 10000 to 19999, inclusive, which is reserved for GRF hubs and triggers. For a list of currently assigned GRF numbers please visit the GRF home page.

The 'Endpoint' entry

This entry is used to specify the TCP endpoint on which the server will listen for client connections. A TCP endpoint consists of an IP number and a TCP port number. This is specified as a fully qualified domain name (FQDN) or IP address in dotted decimal form followed by a colon followed by the decimal TCP port number.

Note that a particular host may have several IP numbers, as an IP number is associated with a particular interface on the host. If you specify a particular IP number in this endpoint, the server will listen for connections only on that particular interface. If you specify an asterisk '*' for the IP number, the server will listen for connections on all interfaces on the host.

The default server endpoint is '*:3757'. This specifies that the server will listen for connections at TCP port number 3757 and accept connections from any interface on the host. This is the registered port number assigned to the GRF by the Internet Assigned Numbers Authority (IANA). If additional port numbers are needed, they should be randomly chosen

from the dynamic or private port range (ports 49152 through 65535) to avoid conflict with registered port numbers.

The 'MaxConnections' entry

This entry specifies the maximum number of simultaneous data client connections that the server will accommodate. If more than this number of clients attempts to connect, the connection will be denied with the error 'Too many connections'.

The default value for this entry is eight.

The 'QueueDepth' entry

This entry specifies the maximum number of packets that may be stored in the queue between a client thread and the digitizer thread. The default value is 64. This value should only need to be modified if the network connections from the server to its clients are very slow due to very heavy traffic.

The 'AllowHosts' and 'DenyHosts' sub-groups

These two sub-groups define lists of hosts allow or denied access to the server. If neither list is defined, all clients will be allowed to connect. If the AllowHosts list is defined, only those hosts will be allowed to connect. If the DenyHosts list is defined, hosts in that list will be denied access.

Host entries in both lists may be specified as:

- Fully qualified domain names such as 'host.banfill.net'
- IP numbers in dotted decimal notation such as '192.168.1.1'
- IP numbers in *classless* or CIDR notation like '192.168.1.0/24' which specifies all hosts on the class C network 192.168.1.0 (24 bits of network mask or 255.255.255.0).

These sub-groups allow administrators to control access to the server in a very flexible way. For example, if all hosts except for two in the 192.168.1.0 network should have access, the allow list would contain the entry '192.168.1.0/24' granting all hosts on that network access. The deny list might then contain the entries '192.168.1.43, 192.168.1.152' denying access to only those two particular hosts.

The 'Repository' sub-group

This sub-group defines how the server should store GRF packet images to files with a repository directory. To disable the repository, simply comment out or delete the 'Repository' sub-group entries. The repository, when enabled, receives exactly the same data as any active clients.

The 'Path' entry

This entry is used to specify the path to the directory where GRF packet image files are to be stored. Files will be named as specified with the 'NameFormat' entry below.

The 'NameFormat' entry

Use this entry to specify the naming convention used to name files created in the repository. The name format string is made up of ASCII characters and the following format specifiers that always begin with a percent sign (%):

Specifier	Meaning
%Y	The four digit year, '2002'
%y	The two digit year, '02'
%M	The two digit month, '01'
%D	The two digit day of month, '01'
%d	The three digit day of year, '001'
%h	The two digit hour of day, '00'
%m	The two digit minute of hour, '00'
%s	The two digit integer second of minute, '00'
%ms	The three digit integer millisecond of second, '000'
%us	The six digit integer microsecond of second, '000000'
%l	The record length as integer seconds (variable length), '0'
%lm	The three digit integer millisecond of record length, '001'
%lu	The six digit integer microsecond of record length, '000001'
%u	The GRF unit number (variable length), '0'
%%	A literal percent sign, '%'

Figure 9 Repository file naming format specifiers.

The default name format specifier string is '%Y%d%h%m%s_%l_%u.grf' and this is compatible with the ISO9660 filesystem commonly used on CD-ROM's as it has only as single dot, a three character extension, and is less the 32 characters long. This name specifier yields names like '2002001120000_120_1000.grf' for a two-minute record from January 1 2002 and 12:00:00.

The 'RecordLength' entry

This entry is used to specify the length of GRF packet image files stored in the repository directory in seconds. Note that the GRF packet image files will be broken on the nearest packet boundary after the data length of the file has exceeded this length. This means that the data length of the packet image files in the repository will rarely be exactly the length specified.

The 'QueueDepth' entry

This entry specifies the maximum number of packets that may be stored in the queue between repository thread and the digitizer thread. The default value is 128. This value should only need to be modified on very slow systems or when writing to very slow storage media.

Chapter 3

GRFTrig

3.1 Overview

GRFTrig acts as a GRF hub with an internal modular trigger, which serves only triggered data to downstream clients and/or to a repository. GRFTrig implements a general network-triggering scheme. Individual incoming channels are triggered using either an STA/LTA event trigger, or an amplitude trigger. A network of these channels are defined and then triggered based on channel trigger coincidence. Only these triggered data are served to clients.

GRFTrig generates and responds to GRF informational messages. This functionality allows it to cooperate with other programs to implement complex triggering schemes. For example, the digital input/output capabilities of GRFd can be used to instruct GRFTrig to trigger on digital input events and/or to cause digital output events when a GRFTrig network or channel triggers.

The triggering algorithms implemented in GRFTrig were designed to be relatively simple and understandable. Our design goal was to make network triggering easy to configure and deterministic in its behavior.

The example network shown in figure 2 shows GRFTrig connected as a client to a GRFHub as a source of data. GRFConvert is connected to GRFTrig as a client, which will create event records in one of its output formats. Note that GRFTrig is itself a GRF hub and could simply replace the GRFHub in the example.

3.2 Running GRFTrig

GRFTrig can be run from the command line as a console application or in the background as a dæmon. The program is controlled through an ASCII configuration file. The program also accepts several command line arguments and a simple help screen is available by typing `grftrig -h` at the command line.

As there are some differences in how you start and stop GRFTrig on different platforms, we'll cover each separately below.

Windows Systems

On most Windows systems, you have two choices for how you will run GRFTrig. You can simply run it from the command line and press Ctrl-C or Ctrl-Break to stop it.

If your system is running Windows NT or 2000, you can install GRFTrig as a *service*. Services are not available on Windows 9*x* or ME. A service is a program that runs in the background much like a dæmon on UNIX-like systems. Services are running on the system even when no user is logged in. Services can be controlled from the Services applet in the Control Panel. GRFTrig allows you to control its service directly from the command line.

Below is the help screen from the Win32 version of GRFTrig:

```
C:\grf>grftrig -h
GRFTrig version 1.2.0 (May  1 2003 16:05:11), pid:1752

Usage: grftrig [[-hvqdb] [-l logfile] [configuration file]] |
               [-install | -remove | -start | -stop | -status]

   -h           Help display.
   -v           Verbose logging.
   -q           Quiet logging.
   -d           Debug logging.
   -b           Run in background. (No)
   -l logfile   Log to logfile. (Off)

   -install     Install as a Win32 service.
   -remove      Uninstall Win32 service.
   -start       Start Win32 service.
   -stop        Stop Win32 service.
   -status      Report Win32 service state.

The configuration file defaults to ./grftrig.conf if not specified.

 [] = optional, () = default, | = mutually exclusive.
```

Before GRFTrig can be run as a service, it must be installed as a service. This is accomplished using the `-install` command line option.

```
C:\grf>grftrig -install
Successfully installed grftrig version 1.2.0 as a service.
```

Once installed, the GRFTrig service will start automatically every time that the system is started and will remain installed until it is removed. To remove the service, use the `-remove` command line option.

To start the service from the command line, use the `-start` option:

```
C:\grf>grftrig -start
grftrig service start is pending...
grftrig version 1.2.0 service is running.
```

If there was an error or if the service is not yet installed an error message will be reported here. To stop the service from the command line, use the `-stop` option. The service can also be started and stopped by using the Services applet from the Control Panel and/or by using the `net start grftrig` and `net stop grftrig` commands at the command prompt.

The status of the service will be reported by using the `-status` option. For example, if the service is installed but not currently running:

```
C:\grf >grftrig -status
grftrig version 1.2.0 service is currently stopped.
```

If the service has not been installed or has been removed:

```
C:\grf >grftrig -status
ERROR: OpenService(grftrig): The specified service does not exist as an
installed service.
```

Linux Systems

GRFTrig can be executed in the foreground from the command line as a console application or run as a dæmon in the background. When running in the foreground, press Ctrl-C to stop the program.

Run the program in the background by specifying the `-b` option on the command line. This causes the program to detach from the terminal and run as a dæmon process. The help screen for the Linux version of the software follows:

```
linux % grftrig -h
GRFTrig version 1.2.0 (May  1 2003 11:21:46), pid:23467

Usage: grftrig [-hvqdb] [-l logfile] [-s facility] [configuration file]

   -h           Help display.
   -v           Verbose logging.
   -q           Quiet logging.
   -d           Debug logging.
   -b           Run in background. (No)
   -l logfile   Log to logfile. (Off)
   -s facility  Log to syslog facility, `LOCAL0'-`LOCAL7'. (Off)

[] = optional, () = default, | = mutually exclusive.
```

When the program is running as a dæmon, it must be stopped by sending the TERM signal. This is easily accomplished using either the **`kill` *`pid`*** command where ***`pid`*** is the process identifier or by using the **`killall grftrig`** command.

GRFTrig can use the system logger on your system for its log output. This is particularly useful when running the program as a dæmon as the log output can be managed automatically by the system. The eight local syslog facilities are supported, `LOCAL0` through `LOCAL7`. One of these facilities should be configured for use by the GRF tools software. For example, to have syslogd maintain the file `/var/log/grf.log` as facility `LOCAL3`, you would add the following lines to syslogd's configuration file (typically `/etc/syslog.conf`):

```
# GRF Tools log output
local3.*   /var/log/grf.log
```

After restarting syslogd, the `LOCAL3` facility will be available for use. For more information about the system logger on your system, consult `/etc/syslog.conf` and/or the 'syslogd' manual pages.

3.3 Command Line Options

In this section, we cover the command line options that are common to GRFTrig on all supported platforms. These options have to do with control of log output and specifying the name of a configuration file. The configuration file is covered in the next section.

As shown in the previous sections, the `-h` option displays a small help screen on the console.

Logging Options

These options can be specified within the configuration file. If they appear on the command line, those settings override the settings within the configuration file.

The `-l` *`filename`* option is used to specify the name of the file to write log output to. When running as a console application, all log output is written to the console. If this option specifies a filename, all output is written to the console and to the specified file.

The `-q`, `-v`, and `-d` options are used to specify *quiet*, *verbose*, and *debug* logging levels respectively. The default logging level is *normal*. Quiet logging specifies that only errors and other critical information be output. Normal logging includes some additional informational messages. Verbose logging includes much more detailed informational messages. Debug logging includes extremely detailed messages about the internal workings of the program.

3.4 GRFTrig Configuration

GRFTrig is controlled primarily through its configuration file. This configuration file allows you to configure and control all aspects of the programs behavior. This simple ASCII file can be edited using your favorite text editor.

This file can be specified on the command line at program startup or the program will search for a default configuration file named `grftrig.conf`. If the configuration file was not specified on the command line, the program will look for `grftrig.conf` in the directory pointed to by the `GRF_ETC` environment variable. If that variable is not set, Windows systems will then look for `grftrig.conf` in the current working directory, UNIX-like systems will look for `/etc/grftrig.conf`.

The configuration file is made up of keywords followed by an entry. Some keywords define groups of entries and curly braces {} are used to delimit these groups. White space characters or commas separate keywords and entries. Comments may appear anywhere in the file and always begin with a pound sign '#'.

The file is made of four basic groups of entries:

- **Logging** – This group configures and controls the programs logging behavior.
- **Connections** – This group defines upstream client connections that the hub will maintain.
- **Server** – This group configures the GRF server.
- **Network** – This group defines a network trigger.

Throughout the example configuration file, the default values for each entry are shown as end-of-line comments inside parentheses.

Top-level entries

These entries are called 'top-level' entries because they appear outside of any configuration group and generally appear near the top of the configuration file.

grftrig.conf – top level entries

```
$Id: grftrig.conf,v 1.10 2003/03/27 19:41:47 cvs Exp $ */
# Copyright (C) 2000-2003 - Robert Banfill - All rights reserved.
# Configuration file for grftrig version 1.2.0 or later
# Defaults for each entry are shown in ()

Daemon              No          # Run in background? (No)
PIDFile             /var/run/grftrig.pid # Process lock file (None)
DateFormat          Ordinal     # Calendar or (Ordinal), overridden by the
                                # DATE_FORMAT environment variable.
```

grftrig.conf – top level entries

Figure 10 grftrig.conf top-level entries.

Note that the 'Dæmon' and 'PIDFile' entries only have meaning on UNIX-like operating systems. See §3.1, Running GRFTrig, above for a discussion about running GRFTrig as a service on 32-bit Windows systems.

The 'Dæmon' entry is used to specify whether the program will run in the background as a *dæmon* and is equivalent to the `-b` command line option. By setting this entry to 'Yes', this becomes the default behavior of the program. Note the end-of-line comments starting with the pound sign (#). The 'Dæmon' entry specifies that the program should run in the background. The 'No' in parenthesis at the end of the comment indicates that if this entry is set to 'No' or the entry is not found in the configuration file, the program will not run as a dæmon.

The 'PIDFile' entry specifies the name of a lock file used by the program to ensure that only one instance of the server is allowed to run at a time. The file is created at startup and the process identifier (PID) is written into the file as ASCII text. If the file cannot be created, the program will exit indicating an error. If this entry does not appear in the configuration file, the program will not perform this process locking operation. Note that the user executing the program must have create, read, and write permission for this lock file and that the filename should be specified as an absolute path, i.e., it should begin with a solidus character, to avoid any dependency on the current working directory at startup.

The 'DateFormat' entry is used to specify the default ISO8601 date and time representation (see §1.1) that will be used by the program. Valid entries include 'Ordinal' and 'Calendar' with 'Ordinal' being the default value. If the `DATE_FORMAT` environment variable is set, its setting overrides this setting in the configuration file.

The 'Logging' group

This group contains entries that configure how log output is handled by the program. These entries are equivalent to their command line counterparts discussed in the previous section.

The following example sets the logging level to *verbose* and directs log messages to a file named `grftrig.log` in the `/grf/log` directory.

grftrig.conf – logging group

```
# This group defines how logging is handled...
Logging {
#    Syslog          local3      # Use syslog facility, LOCAL0 through LOCAL7 (Off)
    File            /grf/log/grftrig.log # Log to file (Off)
#    Level           Quiet       # Be vewy, vewy quiet...
    Level           Verbose     # Output lots of info...
#    Level           Debug       # Debugging message spray...
}
```

grftrig.conf – logging group

Figure 11 grftrig.conf 'Logging' group entries.

The syslog entry is used to specify the syslog facility that will be used for log output. See the discussion about syslog in the previous section for more about using this feature.

The file entry is used to specify a log file specification. In addition to writing log messages to the console, log messages will also be written to this file. This setting can be overridden by the `-l` command line option.

There are four logging levels; `Quiet`, `Normal`, `Verbose`, and `Debug`, with normal being the default. Quiet logging restricts log output to warnings and error conditions only, normal, includes important informational messages, verbose includes much more detailed informational messages, and debug includes extremely detailed informational messages.

The ‘Connections’ group

The connections group defines a list of upstream connections to maintain and some attributes of those connections. Below is the connections group from the example configuration.

grftrig.conf – connections group

```
# Upstream connections to maintain...
Connections {
    QueueDepth      128         # Depth of connection queue as packets (64)
    Timeout         30          # Socket I/O timeout in seconds (5)
    # Server endpoints to connect to...
    Endpoints {
#         192.168.2.1
        localhost:49642
    }
}
```

grftrig.conf – connections group

Figure 12 grftrig.conf ‘Connections’ group entries.

The connections group consists of three keywords, one of which defines a list.

The ‘QueueDepth’ entry

This entry is used to specify the number of GRF packets that can be stored in the queue between the upstream client threads and the server thread. This entry should not need to be changed under normal operating conditions.

The ‘Timeout’ entry

This entry is used to specify the timeout value in seconds for upstream client socket connections. If the network connections to upstream clients are very slow or high latency, you may need to increase this value. Under normal operating conditions, the default value of 30 seconds should be appropriate.

The ‘Endpoints’ list

This list is used to specify a list of upstream server endpoints that will be connected to as clients. The program will create a client thread to service each of these connections and maintain them for the life of the program.

Endpoints may be specified as IP numbers in dotted decimal form or as a domain name. The port number may be specified by appending a colon (:) followed by the decimal port number to the address. If no port is specified, the default port (3757) is used. For example, ‘192.168.1.1’ is equivalent to ‘192.168.1.1:3757’.

Any number of endpoints may be specified. There is no hard limit on the number of endpoints. It is only limited be available memory.

The ‘Server’ group

The server group contains entries and sub-groups of entries that configure the behavior of the server in general. Below is the server group from the example configuration file provided in the distribution:

grftrig.conf – server group

```
# Server settings...
Server {
    FriendlyName    GRFTrig     # Name used to describe this server (GRFTrig)
    GRFUnitID       600         # GRF unit ID used for this server (Must be unique!)
    Endpoint        *:3757      # Listen at this endpoint, * = any interface (*:3757)
    MaxConnections  8           # Maximum allowed number of simultaneous connections (8)
    QueueDepth      128         # Depth of client queues as packets (64)
```

```
    # Lists of hosts allowed or not allowed to connect as data clients (Allow all)
#    AllowHosts {
#        localhost
#        192.168.1.0/24
#        192.168.2.0/24
#    }
#    DenyHosts {
#        192.168.1.2
#        192.168.2.2
#    }

    # Store waveform data to local repository (off)
#    Repository {
#        Path /mnt/repository/events
       # File naming format specifier:
       #  %Y  = four digit year, %y = two digit year,
       #  %M  = two digit month, %D = two digit day of month,
       #  %d  = three digit day of year,
       #  %h  = two digit hour of day,
       #  %m  = two digit minute of hour,
       #  %s  = two digit integer second of minute,
       #  %ms = three digit integer millisecond of second,
       #  %us = six digit integer microsecond of second,
       #  %l  = record length as integer seconds (variable length),
       #  %lm = three digit integer millisecond of record length,
       #  %lu = six digit integer microsecond of record length,
       #  %u  = GRF unit number (variable length), %% = literal '%'.
       #  (ISO9660 compatable: %Y%d%h%m%s_%l_%u.grf)
#        NameFormat      "%Y-%d-%h-%m-%s.%ms_%l.%lm_%u.grf"
#        RecordLength    3600    # Record length in seconds
#        QueueDepth      128     # Depth of queue as packets (64)
#    }
}
```

grftrig.conf – server group

Figure 13 grftrig.conf 'Server' group entries.

The 'Server' group contains eight entries, two of which define lists and one that defines a sub-group. First, we examine the five non-group entries.

The 'FriendlyName' entry

This entry is used to assign a simple name to the server that will be used in informational messages. For example, when a client connects to this server, this name will be used in the connection message.

This name can be a maximum of sixteen characters in length and defaults to 'GRFTrig' if this entry is not found in the configuration file.

The 'GRFUnitID' entry

Use this entry to assign a GRF unit ID number to this server. This will be the unit ID used in informational packets originate at this server. This is not the unit ID that contained in data packets that come through this server. Those data packets contain the unit ID of the digitizer that created the packets.

The unit ID defaults to zero and it is recommended that settings for this entry be in the range from 10000 to 19999, inclusive, which is reserved for GRF hubs and triggers. For a list of currently assigned GRF numbers please visit the GRF home page.

The 'Endpoint' entry

This entry is used to specify the TCP endpoint on which the server will listen for client connections. A TCP endpoint consists of an IP number and a TCP port number. This is specified as a fully qualified domain name or IP address in dotted decimal form followed by a colon followed by the decimal TCP port number.

Note that a particular host may have several IP numbers, as an IP number is associated with a particular interface on the host. If you specify a particular IP number in this endpoint, the server will listen for connections only on that particular interface. If you specify an asterisk '*' for the IP number, the server will listen for connections on all interfaces on the host.

The default server endpoint is '*:3757'. This specifies that the server will listen for connections at TCP port number 3757 and accept connections from any interface on the host. This is the registered port number assigned to the GRF by the Internet Assigned Numbers Authority (IANA). If additional port numbers are needed, they should be randomly chosen from the dynamic or private port range (ports 49152 through 65535) to avoid conflict with registered port numbers.

The 'MaxConnections' entry

This entry specifies the maximum number of simultaneous data client connections that the server will accommodate. If more than this number of clients attempts to connect, the connection will be denied with the error 'Too many connections'.

The default value for this entry is eight.

The 'QueueDepth' entry

This entry specifies the maximum number of packets that may be stored in the queue between a client thread and the digitizer thread. The default value is 64. This value should only need to be modified if the network connections from the server to its clients are very slow due to very heavy traffic.

The 'AllowHosts' and 'DenyHosts' sub-groups

These two sub-groups define lists of hosts allow or denied access to the server. If neither list is defined, all clients will be allowed to connect. If the AllowHosts list is defined, only those hosts will be allowed to connect. If the DenyHosts list is defined, hosts in that list will be denied access.

Host entries in both lists may be specified as:

- Fully qualified domain names such as 'host.banfill.net'
- IP numbers in dotted decimal notation such as '192.168.1.1'
- IP numbers using *classless* or CIDR notation like '192.168.1.0/24', which specifies all hosts on the class C network 192.168.1.0 (24 bits of network mask or 255.255.255.0).

These sub-groups allow administrators to control access to the server in a very flexible way. For example, if all hosts except for two in the 192.168.1.0 network should have access, the allow list would contain the entry '192.168.1.0/24' granting all hosts on that network access. The deny list might then contain the entries '192.168.1.43, 192.168.1.152' denying access to only those two particular hosts.

The 'Repository' sub-group

This sub-group defines how the server should store GRF packet images to files with a repository directory. To disable the repository, simply comment out or delete the 'Repository' sub-group entries. The repository, when enabled, receives exactly the same data as any active clients. For GRFTrig, this means that for each network event, a GRF packet image file will be created in the repository directory.

The 'Path' entry

This entry is used to specify the path to the directory where GRF packet image files are to be stored. Files will be named as specified with the 'NameFormat' entry below.

The 'NameFormat' entry

Use this entry to specify the naming convention used to name files created in the repository. The name format string is made up of ASCII characters and the following format specifiers that always begin with a percent sign (%):

Specifier	Meaning
%Y	The four digit year, '2002'
%y	The two digit year, '02'
%M	The two digit month, '01'
%D	The two digit day of month, '01'
%d	The three digit day of year, '001'
%h	The two digit hour of day, '00'
%m	The two digit minute of hour, '00'
%s	The two digit integer second of minute, '00'
%ms	The three digit integer millisecond of second, '000'
%us	The six digit integer microsecond of second, '000000'
%l	The record length as integer seconds (variable length), '0'
%lm	The three digit integer millisecond of record length, '001'
%lu	The six digit integer microsecond of record length, '000001'
%u	The GRF unit number (variable length), '0'
%%	A literal percent sign, '%'

Figure 14 Repository file naming format specifiers.

The default name format specifier string is '%Y%d%h%m%s_%l_%u.grf' and this is compatible with the ISO9660 filesystem commonly used on CD-ROM's as it has only as single dot, a three character extension, and is less the 32 characters long. This name specifier yields names like '2002001120000_120_1000.grf' for a two-minute record from January 1 2002 and 12:00:00.

The 'RecordLength' entry

This entry is used to specify the length of GRF packet image files stored in the repository directory in seconds. GRFTrig will create a new packet image file for each network trigger event and so this entry is only meaningful as the maximum length that a record can reach.

Note that the GRF packet image files will be broken on the nearest packet boundary after the data length of the file has exceeded this length. This means that the data length of the packet image files in the repository will rarely be exactly the length specified.

The 'QueueDepth' entry

This entry specifies the maximum number of packets that may be stored in the queue between repository thread and the digitizer thread. The default value is 128. This value should only need to be modified on very slow systems or when writing to very slow storage media.

'Network' groups

Network groups are used to define networks of stations. A network consists of a list of voting or trigger channels, their trigger attributes, network trigger criteria, a list of channels to be recorded, and the recording attributes.

A network group may contain nine entries, two of which specify sub-groups.

Below is an example network definition:

grftrig.conf – network groups

```
# Define one or more networks...
Network "XX" {                      # Network name ("Net'n'")
        Trigger STA/LTA {
            Channel         4CH:EHZ # Source channel name
            MeanSeconds        30.0 # Time constant for running mean removal (30.0)
            StaSeconds          0.2 # Short-term average length (0.1)
            LtaSeconds         10.0 # Long-term average length (10.0)
            TriggerRatio       10.0 # STA/LTA ratio causing trigger (8.0)
            DetriggerRatio      2.0 # STA/LTA ratio causing detrigger (2.0)
            LtaHold             Yes # Hold LTA during trigger? (Yes)
        }
        Trigger STA/LTA {
            Channel 4CH:EHN, MeanSeconds 30.0, StaSeconds 0.2,
            LtaSeconds 10.0, TriggerRatio 10.0, DetriggerRatio 2.0, LtaHold Yes
        }
        Trigger STA/LTA {
            Channel 4CH:EHE, MeanSeconds 30.0, StaSeconds 0.2,
            LtaSeconds 10.0, TriggerRatio 10.0, DetriggerRatio 2.0, LtaHold Yes
        }
#        Trigger Amplitude {         # Amplitude or threshold trigger channel
#            Channel         4CH:EHZ # Source GRF channel name
#            MeanSeconds        30.0 # Time constant for running mean removal (30.0)
#            TriggerAmplitude  50000 # Absolute amplitude threshold in counts (50000)
#            TriggerSeconds      0.1 # Must stay over threshold for this long (0.1)
#            DetriggerAmplitude 5000 # Absolute amplitude threshold in counts (5000)
#            DetriggerSeconds    0.1 # Must stay under threshold for this long (0.1)
#        }
    }
    TriggerVotes                  3 # Trigger network when this many voters are
                                    #   triggered within the coincidence window (1)
    CoincidenceWindow           5.0 # Triggering coincidence window in seconds (1.0)
    DetriggerVotes                0 # Detrigger network when triggered voters
                                    #   fall below this number(0)

    PreEventSeconds            10.0 # Seconds of pre-event data to include (5.0)
    PostEventSeconds           20.0 # Seconds of post-event data to include (5.0)

    MinEventSeconds            30.0 # Output data will be at least this long (1.0)
    MaxEventSeconds           180.0 # Output data will not be longer than this (120.0)

    Channels {                      # Data for these channels are output
        4CH:EHZ, 4CH:EHN, 4CH:EHE
    }
}
```

grftrig.conf – network groups

Figure 15 grftrig.conf 'Network' group entries.

The network entry is used to define and name a network group. The name follows the network keyword and must be 7 or fewer characters in length.

This group defines a series of voting channels that will trigger as a group based on a coincidence algorithm. There is no hard limit on the number of networks that may be defined. Networks may also overlap, that is, channels may appear in multiple networks.

The 'TriggerVotes' entry

This entry is used to specify the number of voting channel triggers that must occur within the coincidence window in order to declare a network event. When a network event is

declared, the earliest trigger time of all the voting channels is taken as the network trigger time.

The 'CoincidenceWindow' entry

This entry is used to specify the length of the triggering coincidence window in seconds. When the number of voting channel triggers specified in the TriggerVotes entry occurs within the number of seconds specified here relative to each other, a network trigger is declared.

When a network trigger is declared, the earliest trigger time from all voting channels is taken as the network trigger time.

The 'DetriggerVotes' entry

Once a network event is declared, data is recorded until the number of voting channel triggers remaining in the triggered state is greater than the value specified in this entry. At that point, network detrigger is declared.

When a network detrigger is declared, the latest detrigger time of all of the voting channels is taken as the network detrigger time.

The 'PreEventSeconds' entry

This entry specifies the number of seconds to subtract from the network trigger time to get the time of the earliest data to serve to clients.

The 'PostEventSeconds' entry

This entry specifies the number seconds to add to the network detrigger time to get the time of the latest data to serve to clients. This time of latest data will then be constrained by the MinEventSeconds and MaxEventSeconds entries.

The 'MinEventSeconds' entry

This entry specifies the minimum length of the data that will be served to the clients. The time of the earliest data is determined based on the network trigger time minus the PreEventSeconds entry. The time of latest data will be either; the network detrigger time plus the PostEventSeconds entry, or the time of the earliest data plus the time of this entry, whichever is greater.

The 'MaxEventSeconds' entry

This entry specifies the maximum allowed length of any event. The value of this entry is used both to limit the length of voting channel triggers and to limit the length of any data served to clients.

The 'Voters' sub-group

The voters sub-group is used to define triggering channels that will vote for network triggers. This sub-group may contain many instances of a single sub-group keyword, 'Trigger', which defines a single triggering channel. There is no hard limit on the maximum number of voting triggers in a particular network. It is only limited by available memory.

'Trigger' sub-groups

The trigger keyword declares and defines a channel trigger and may only occur inside of the voters sub-group. The trigger type is named following the trigger keyword. Currently, there are two types of triggers available:

- **STA/LTA** – A simple short-term average/long-term average event trigger
- **Amplitude** – A simple amplitude or threshold trigger

The contents of the trigger group are the settings for the trigger depending on the type. Each trigger sub-group must contain a 'Channel' entry that specifies the GRF name of the channel associated with the trigger. All of the other trigger group entries depend on the trigger type.

'STA/LTA' trigger parameters

Below we present an example trigger sub-group that specifies an STA/LTA event trigger:

grftrig.conf – sta/lta trigger group

```
. . .

Trigger STA/LTA {           # STA/LTA trigger channel
    Channel         4CH:EHZ # Source GRF channel name
    MeanSeconds        30.0 # Time constant for running mean removal (30.0)
    StaSeconds          0.1 # Short-term average length (0.1)
    LtaSeconds         10.0 # Long-term average length (10.0)
    TriggerRatio        8.0 # STA/LTA ratio causing trigger (8.0)
    DetriggerRatio      2.0 # STA/LTA ratio causing detrigger (2.0)
    LtaHold             Yes # Hold LTA during trigger? (Yes)
}

. . .
```

grftrig.conf – sta/lta trigger group

Figure 16 grftrig.conf STA/LTA event trigger sub-group.

The channel entry is used to specify the GRF name of the channel whose data will be analyzed for this trigger. The network identifier portion of this name is optional and will be replaced with the name of this network in the output data. In this example, the network portion of the name is 'GRF', this will be replaced with 'EVN' in the output data as that is the name of the network that this trigger is in. You could equivalently specify 'Stn2:BHZ', effectively wildcarding the network identifier, as long as there is no other channel by that name in incoming data.

The trigger algorithm is quite simple: a running mean is computed from the incoming sample data and subtracted out to remove any DC offset from the data. The short-term (STA) and long-term (LTA) averages are then computed. The ratio of these (STA/LTA) is then computed and checked against the trigger ratio setting. If it exceeds this setting, a trigger is declared. If LTA hold is specified, the LTA is held at the value that it had at trigger time and not computed for the duration of the event. While triggered, the ratio is checked against the detrigger ratio. When it drops below this value, the channel is detriggered.

Note that the LTA hold option can create situations where a channel can never detrigger. In all cases, the channel will be forced to detrigger after the length specified in the MaxEventSeconds entry specified for the network group.

'Amplitude' trigger parameters

Below we present an example trigger sub-group that specifies an amplitude trigger:

grftrig.conf – amplitude trigger group

```
. . .

Trigger Amplitude {         # Amplitude or threshold trigger channel
    Channel         4CH:EHZ # Source GRF channel name
    MeanSeconds        30.0 # Time constant for running mean removal (30.0)
    TriggerAmplitude  50000 # Absolute amplitude threshold in counts (50000)
    TriggerSeconds      0.1 # Must stay over threshold for this long (0.1)
    DetriggerAmplitude 5000 # Absolute amplitude threshold in counts (5000)
    DetriggerSeconds    0.1 # Must stay under threshold for this long (0.1)
```

```
        }
        . . .
```

grftrig.conf – amplitude trigger group

Figure 17 grftrig.conf amplitude trigger sub-group.

As in the STA/LTA trigger, the channel entry is used to specify the GRF name of the channel whose data will be analyzed for this trigger. The network identifier portion of this name is optional and will be replaced with the name of this network in the output data.

The trigger algorithm is a simple amplitude, or threshold, trigger. As with the event trigger above, the running mean is computed and subtracted to remove any DC offset present in the incoming data. The data is then rectified (the absolute value is taken) and compared to the value specified in the TriggerAmplitude setting. The amplitude must stay above the specified trigger amplitude value for the time specified in the TriggerSeconds entry for a trigger to be declared.

Detrigger is exactly the opposite: the amplitude must stay below the value specified in the DetriggerAmplitude entry for the time specified in the DetriggerSeconds entry. Again, in no case will the trigger last longer than the time specified in the MaxEventSeconds entry for the network group.

The 'Channels' list

This sub-group defines the channels that will be recorded when a network trigger is declared. The clients will see data from these channels during a network event. Note that voting channels are not included automatically and must be included in this list if they are to be seen in the output data.

The network identifier portion of the GRF channel name will be replaced with the name of this network on output. You may omit the network identifier in the GRF names in this list unless they will be needed to uniquely identify channels in the incoming data.

There is not hard limit on the number of channels in a network. This is only limited by available memory.

3.5 GRF informational messages

As channel and network triggers occur, GRFTrig generates GRF informational messages indicating that these events have occurred. These messages are sent both upstream, that is, toward the source of the data, and downstream to clients.

These messages are used by GRFConvert to detect event boundaries and can be used by GRFd in to cause digital output events.

Channel trigger/detrigger messages

For each channel trigger that occurs, a message containing the following text is generated:

`Channel trigger:` *`network:station:component timestamp`*

Where, *`network:station:component`* uniquely identifies the particular channel that has triggered and *`timestamp`* is the time at which this channel triggered.

As channels detrigger, a corresponding message is generated indicating that the channel has detriggered:

`Channel detrigger:` *`network:station:component timestamp`*

Here, *`timestamp`* indicates the channel detrigger time.

Network trigger/detrigger messages

As network trigger criteria are met, as message containing the following text is generated:

```
Network trigger: network(n) timestamp
```

Where, *network* is the name of the triggered network, *n* is the number of votes for trigger, and the *timestamp* is the network trigger time. The network trigger time is the earliest channel trigger time of the voting channels.

When the network detriggers, a message containing the following text is generated:

```
Network detrigger: network timestamp
```

Here, *timestamp* indicates the network detrigger time. The network detrigger time is the latest channel detrigger time of the voting channels.

Special network trigger messages

When GRFTrig receives a GRF informational message containing the following text:

```
GRFTrig trigger network timestamp
```

It will trigger the network with the name *network* and set the network trigger time to the time indicated by *timestamp*. If the indicated network does not exist, the message is ignored.

When GRFTrig receives a message containing the following text:

```
GRFTrig detrigger network timestamp
```

It will detrigger the specified network and set the detrigger time to the time indicated in *timestamp*.

Note that these messages are case sensitive including the network name.

These messages are typically generated by digital input events within GRFd. Please refer to the GRFd documentation for more about digital input/output events.

Chapter 4

GRFLog

4.1 Overview

GRFLog is a simple GRF client that displays the contents of GRF packets received from a GRF server in various forms and can perform various analyses on GRF packet streams.

It can be used to:

- Log the entire contents of all GRF packets in verbose detail
- Log only brief header information for all GRF packets
- Log only GRF informational packets
- Perform timing analysis on a selected data channel
- Analyze data packets and display statistics about their contents
- Capture GRF packets into a packet image file

GRFLog output is displayed on the console or can be stored to a log file on disk. In addition to acting as a client to any GRF server, GRFLog can read GRF packet images from a file (for example, files created by the repository thread in GRFd, GRFHub, or GRFTrig).

4.2 Running grflog

GRFLog is controlled entirely from the command line. The following help screen is displayed when the `-h` option is specified on the command line:

```
C:\grf>grflog -h
GRFLog version 1.2.0 (May  1 2003 16:04:40), pid:1636

Usage: grflog [-htvina] [-p [station:]component] [-r seconds]
              [-o output_file] [-c capture_file]
              -s ip_address[:port] | [-f] input_file

   -h          Help display.
   -t          Terse output, single line headers.
   -v          Verbose output, fully annotated headers.
   -i          Log GRF information messages only.
   -n          No sample data, headers only.
   -a          Analyze data packets and log statistics.
   -p channel  Analyze timing on specified station:component.
   -r seconds  Set socket read timeout. (30 seconds)
   -o file     Output file. (stdout)
   -c file     GRF packet capture file. (none)
   -s socket   Socket to connect to for input data,
                port number defaults to 3757 if not specified.
   [-f] file   GRF packet image file to read for input data.
```

```
[] = optional, () = default, | = mutually exclusive.
```

The options, or switches, are used to control various behaviors of the software. Note that because this software runs on a variety of platforms, options always begin with a hyphen (-), never with a solidus (/). Note also that switches, that is, options not followed by arguments, can be combined. For example, to specify both `-v` and `-n`, you may specify `-vn`. The space between an option and a required argument is also optional. For example, to specify an output file with the `-o` option, you could specify either `-o grflog.log` or `-ogrflog.log`.

To stop the program press 'Ctrl-C', 'Ctrl-break', or send the TERM signal using the `kill` command if available.

4.3 Specifying the source of GRF data

GRF packets may be read from a GRF server via a socket connection or from a GRF packet image file. Use either the `-s` *`endpoint`* or `-f` *`filename`* option to specify the source of GRF data. Note that the program can read from only one data source at a time. These two options are mutually exclusive.

As an example, to connect to a GRF server running on the local machine at the default port number, you might use the following command:

```
linux % grflog -s localhost
```

To specify a port number other than the default (3757), append a colon (:) followed by the decimal port number to the address. For example, to connect to a GRF server on localhost at TCP port number 49643, you might use the following command:

```
linux % grflog -s localhost:49643
```

To read GRF packets from an image file named `data.grf`, you might use the following command:

```
linux % grflog -f data.grf
```

4.4 Data logging options

The various data logging options below can be used individually, or together, to obtain various views of the data contained in GRF data packets.

The default behavior is to log the GRF packet information in terse, one line per packet, format and dump data for data packets. The follow log fragment shows this output:

```
linux % grflog -s netdas4ch
GRFLog version 1.2.0 (May  1 2003 16:04:40), pid:436
Connected to unit 1000 (NetDAS-4CH) at 192.168.1.5:3757
XX:4CH:EHN 1000:3:56341 2003-05-03T04:29:38.242Z/04:29:43.222Z (Good) CM8 498 2.14:1
     -969     +860    +2560    +3219    +2668    +1379     -264    -1429
    -1681     -652    +1211    +2523    +2464    +1396     +356      -85
      +71     +591    +1143    +1444    +1023     +375      +99     +380
  lines deleted…
     -281    -1177     -765     +616    +2016    +2636    +1635     +380
     -425     -349     -309     +332    +1227    +2120    +2315    +1403
     +219     -676
XX:4CH:EHE 1000:4:56342 2003-05-03T04:29:39.502Z/04:29:44.502Z (Good) CM8 500 2.16:1
    +2375    +2172    +1280     +388      -33     -137     -132      -37
       +4     +151     +188     +171      -17     -301     -224     +188
     +824    +1659    +2331    +2487    +2179    +1264     -237    -1597
  lines deleted…
     +460     -173      -69     +616     +792     +527     +355     +400
     +407     +247     +452     +515     +759     +504     +291     +271
     +447     +483     +355     +528
XX:4CH:EHZ 1000:2:56343 2003-05-03T04:29:40.762Z/04:29:45.762Z (Good) CM8 500 2.18:1
```

```
     -752     -373     +240     +760     +824     +451      -40     -281
      -84     +328     +623     +527     +103     -420     -749     -769
     -520     -101     +207     +351     +339     +212     +155     +136
  lines deleted…
     -284     -277     -281     -284     -284     -284     -281     -281
     -280     -284     -277     -277     -281     -281     -284     -284
     -289     -285     -280     -277
Unit 1000: GPS unlocked: 2003-05-03T04:43:55Z
BSE:4CH:EHZ 1000:2:01282 2002-10-23T00:43:54.364Z/00:43:59.164Z (Good) CM8 480 2.06:1
     -232     -229     -345     -304     +100     +147      -32       -4
     -188     -405     -313      -61      +88      +88      -61     -144
     -188     -404     -500     -304      +11      +52     -125     -217
  lines deleted…
     +404     +104     -241     -372     -204      +40     +387     +488
     +339      +16     -237     -336     -228      -49      -13      +79
     +171     +132      -48      -49
Unit 1000 packets: 3 @ 1.18/sec, sequence breaks: 0
Total packets: 3 @ 1.18/sec, sequence breaks: 0
```

Each line beginning with 'Unit *nnnn*' is a GRF informational message, where *nnnn* is a GRF unit identification number. Various GRF tools generate these messages. The particular message in the example above was generated by GRFd notifying us that the GPS receiver on unit number 1000 unlocked at 2003-05-03T04:43:55Z.

Lines beginning with the GRF name 'XX:4CH:…' are GRF data packets. The fields from right to left are:

- The GRF name of this channel (network:station:component)
- The unit ID, channel number, and packet sequence number (unit:channel:sequence)
- The time-interval for the waveform data in this packet. This time-interval consists of the initial sample UTC date and time (IST) followed by the solidus character (/) followed by the last sample UTC time (LST) plus one sampling period (i.e., the IST to expect on the next data packet from this channel). Note that for this example, ISO8601 'Calendar' date and time representations are being used (`DATE_FORMAT=Calendar`).
- The GRF time quality (see below for a description of these values)
- The data type and number of samples in this packet
- If the type is CM8, the compression ratio achieved in this packet

The sample data follows and is printed in eight columns in row major order. If you do not wish to see the sample data, use the `-n` option to see packet header information only with no data:

```
linux % grflog -ns netdas4ch
GRFLog version 1.2.0 (May  1 2003 16:04:40), pid:1756
Connected to unit 1000 (NetDAS-4CH) at 192.168.1.5:3757
XX:4CH:EHN 1000:3:56528 2003-123T04:34:45.456Z/04:34:50.456Z (Good) CM8 500 2.16:1
XX:4CH:EHE 1000:4:56529 2003-123T04:34:46.746Z/04:34:51.726Z (Good) CM8 498 2.14:1
XX:4CH:EHZ 1000:2:56530 2003-123T04:34:48.606Z/04:34:53.526Z (Good) CM8 492 2.12:1
Unit 1000: GPS unlocked: 2003-123T04:34:55Z
XX:4CH:EHN 1000:3:56532 2003-123T04:34:50.456Z/04:34:55.346Z (Good) CM8 489 2.10:1
XX:4CH:EHE 1000:4:56533 2003-123T04:34:51.726Z/04:34:56.676Z (Good) CM8 495 2.13:1
XX:4CH:EHZ 1000:2:56534 2003-123T04:34:53.526Z/04:34:58.526Z (Good) CM8 500 2.22:1
XX:4CH:EHN 1000:3:56535 2003-123T04:34:55.346Z/04:35:00.346Z (Good) CM8 500 2.21:1
Unit 1000: GPS locked: 2003-123T04:35:02Z
XX:4CH:EHE 1000:4:56537 2003-123T04:34:56.676Z/04:35:01.676Z (Poor) CM8 500 2.17:1
XX:4CH:EHZ 1000:2:56538 2003-123T04:34:58.526Z/04:35:03.506Z (Poor) CM8 498 2.14:1
XX:4CH:EHN 1000:3:56539 2003-123T04:35:00.346Z/04:35:05.256Z (Poor) CM8 491 2.11:1
Unit 1000 packets: 12 @ 0.72/sec, sequence breaks: 0
Total packets: 12 @ 0.72/sec, sequence breaks: 0
```

Note that for this example, ISO8601 'Ordinal' date and time representations are used (`DATE_FORMAT=Ordinal`).

Analyzing data and logging statistics (-a)

Various statistics about the data contained in data packets can be displayed by using the `-a` option. Note that these statistics are computed on a per packet basis only and are computed in the same manner as those reported by GRFConvert when using its `-f stats` option. Please refer to the GRFConvert chapter in this document for more details about the methods used to compute these values.

In the following log fragment, the `-a` option is combined with the `-n` option to summarize the data in a relatively compact form:

```
C:\grf>grflog -nas netdas4ch
GRFLog version 1.2.0 (May  1 2003 16:04:40), pid:1808
Connected to unit 1000 (NetDAS-4CH) at 192.168.1.5:3757
XX:4CH:EHE 1000:4:56772 2003-05-03T04:41:21.803Z/04:41:26.803Z (Good) CM8 500 3.98:1
 Min: -72, Max: -45, Range: 28, Mean: -59.65, Sigma: 4.80
XX:4CH:EHZ 1000:2:56773 2003-05-03T04:41:25.203Z/04:41:30.203Z (Good) CM8 500 3.98:1
 Min: -476, Max: -444, Range: 33, Mean: -458.75, Sigma: 5.36
XX:4CH:EHN 1000:3:56774 2003-05-03T04:41:25.363Z/04:41:30.363Z (Good) CM8 500 3.98:1
 Min: 220, Max: 252, Range: 33, Mean: 239.68, Sigma: 4.54
XX:4CH:EHE 1000:4:56775 2003-05-03T04:41:26.803Z/04:41:31.803Z (Good) CM8 500 4.00:1
 Min: -69, Max: -49, Range: 21, Mean: -59.94, Sigma: 4.26
XX:4CH:EHZ 1000:2:56776 2003-05-03T04:41:30.203Z/04:41:35.203Z (Good) CM8 500 3.98:1
 Min: -477, Max: -444, Range: 34, Mean: -459.38, Sigma: 5.70
XX:4CH:EHN 1000:3:56777 2003-05-03T04:41:30.363Z/04:41:35.363Z (Good) CM8 500 3.98:1
 Min: 224, Max: 251, Range: 28, Mean: 238.97, Sigma: 4.27
XX:4CH:EHE 1000:4:56778 2003-05-03T04:41:31.803Z/04:41:36.803Z (Good) CM8 500 3.98:1
 Min: -73, Max: -45, Range: 29, Mean: -59.21, Sigma: 4.47
XX:4CH:EHZ 1000:2:56779 2003-05-03T04:41:35.203Z/04:41:40.203Z (Good) CM8 500 3.98:1
 Min: -477, Max: -440, Range: 38, Mean: -459.20, Sigma: 5.76
Unit 1000 packets: 8 @ 0.55/sec, sequence breaks: 0
Total packets: 8 @ 0.55/sec, sequence breaks: 0
```

Note that for this example, ISO8601 'Calendar' date and time representations are used (`DATE_FORMAT=Calendar`).

Verbose logging of GRF packet header (-v)

By using the verbose output option, `-v`, you can view the entire annotated contents of each GRF packet header. The following log fragment shows two packets as displayed when using both the `-n` and `-v` options:

```
C:\grf>grflog -vns 192.168.2.1
GRFLog version 1.2.0 (May  1 2003 16:04:40), pid:1808
Connected to unit 1000 (NetDAS-4CH) at 192.168.1.5:3757
GRF common header:
 Signature:       GRF
 Version:         1
 Length:          590
 Sequence number: 56814
 Unit ID:         1000
 Type:            1 - Data
 GRF data header:
  Channel:        4
  Network:        XX
  Station:        4CH
  Component:      EHE
  Time:           2003-05-03T04:42:31.717Z
  Corrected time: 2003-05-03T04:42:31.804Z quality: Good
  Rate:           100.000000 sps
  Corrected rate: 99.998444 quality: Poor
  Counts/volt:    400000
  Data type:      2 - CM8 compression
  Data samples:   500
GRF common header:
 Signature:       GRF
 Version:         1
 Length:          592
 Sequence number: 56815
 Unit ID:         1000
```

```
 Type:            1 - Data
 GRF data header:
  Channel:        2
  Network:        XX
  Station:        4CH
  Component:      EHZ
  Time:           2003-05-03T04:42:35.137Z
  Corrected time: 2003-05-03T04:42:35.204Z quality: Good
  Rate:           100.000000 sps
  Corrected rate: 99.998444 quality: Poor
  Counts/volt:    400000
  Data type:      2 - CM8 compression
  Data samples:   500
Unit 1000 packets: 2 @ 0.38/sec, sequence breaks: 0
Total packets: 2 @ 0.38/sec, sequence breaks: 0
```

4.5 Viewing only GRF informational messages

Various GRF tools generate GRF informational messages that are carried in GRF information packets as data. GRFd creates informational messages about the state of the GPS receiver and digital I/O events among others. GRFTrig also generates various information messages about channel and network triggers.

GRFLog can be used to display only these messages and ignore everything else by using the `-i` option. The following log fragment shows GPS information messages coming from GRFd:

```
linux % grflog -is netdas4ch
GRFLog version 1.2.0 (May  1 2003 16:04:40), pid:1116
Connected to unit 10000 (GRFHub) at 127.0.0.1:3757
Unit 1000: GPS position at 2003-123T04:20:00Z, lat:+61.136433, long:-146.351950
Unit 1000: GPS position at 2003-123T04:30:00Z, lat:+61.136400, long:-146.352083
Unit 1000: GPS unlocked: 2003-123T04:34:55Z
Unit 1000: GPS locked: 2003-123T04:35:02Z
Unit 1000: GPS position at 2003-123T04:40:00Z, lat:+61.136383, long:-146.352083
Unit 1000 packets: 2445 @ 0.60/sec, sequence breaks: 0
Total packets: 2445 @ 0.60/sec, sequence breaks: 0
```

The following log output shows informational messages from GRFTrig:

```
linux % grflog -is localhost:49643
GRFLog version 1.2.0 (May  1 2003 16:04:40), pid:1724
Connected to unit 10001 (GRFTrig) at 127.0.0.1:49643
Unit 10001: Channel trigger: XX:4CH:EHN 2003-05-03T04:51:43.163Z
Unit 10001: Channel detrigger: XX:4CH:EHN 2003-05-03T04:51:44.333Z
Unit 10001: Channel trigger: XX:4CH:EHZ 2003-05-03T04:51:43.173Z
Unit 10001: Channel detrigger: XX:4CH:EHZ 2003-05-03T04:51:44.323Z
Unit 10001: Channel trigger: XX:4CH:EHE 2003-05-03T04:51:43.163Z
Unit 10001: Channel detrigger: XX:4CH:EHE 2003-05-03T04:51:44.353Z
Unit 10001: Network trigger: XX(3) 2003-05-03T04:51:43.163Z
Unit 10001: Network detrigger: XX 2003-05-03T04:51:44.353Z
Unit 1000 packets: 15 @ 0.20/sec, sequence breaks: 0
Unit 10001 packets: 8 @ 0.11/sec, sequence breaks: 0
Total packets: 23 @ 0.31/sec, sequence breaks: 0
```

Note that for this example, ISO8601 'Calendar' date and time representations are used (`DATE_FORMAT=Calendar`).

4.6 Channel timing analysis option

GRFLog can be used to analyze timing information on a specified channel to observe timing system behavior. Use the `-p` *`station:component`* option to perform this analysis. The station and/or the component identifier may be wild-carded, that is, they may be specified as '*' meaning all. As it is only meaningful to examine timing on a single channel for each available station, if you specify `-p *`, the first channel from each available station will be analyzed. If you specify `-p EHZ`, all channels named EHZ from all available stations will be

analyzed. This is an implied wildcard of the station identifier. If you specify `-p 4CH:EHZ`, only channel EHZ from station 4CH will be analyzed.

The following log fragment shows grflog output using this option:

```
C:\grf>grflog -s 192.168.2.1 -p 4CH:EHZ
GRFLog version 1.2.0 (May  1 2003 16:04:40), pid:1808
Connected to unit 1000 (NetDAS-4CH) at 192.168.1.5:3757
XX:4CH:EHZ 1000:2:57264 2003-123T04:54:59.535183Z (Good) 99.998444 (Poor) -0.000018
XX:4CH:EHZ 1000:2:57267 2003-123T04:55:04.535235Z (Good) 99.998442 (Good) -0.000025
XX:4CH:EHZ 1000:2:57270 2003-123T04:55:09.535298Z (Good) 99.998442 (Good) -0.000014
XX:4CH:EHZ 1000:2:57273 2003-123T04:55:14.535355Z (Good) 99.998442 (Good) -0.000020
XX:4CH:EHZ 1000:2:57276 2003-123T04:55:19.535409Z (Good) 99.998442 (Good) -0.000023
XX:4CH:EHZ 1000:2:57279 2003-123T04:55:24.535471Z (Good) 99.998442 (Good) -0.000015
XX:4CH:EHZ 1000:2:57282 2003-123T04:55:29.535530Z (Good) 99.998442 (Good) -0.000018
XX:4CH:EHZ 1000:2:57285 2003-123T04:55:34.535585Z (Good) 99.998442 (Good) -0.000022
XX:4CH:EHZ 1000:2:57288 2003-123T04:55:39.535649Z (Good) 99.998442 (Good) -0.000013
Unit 1000 packets: 52 @ 0.60/sec, sequence breaks: 0, time tears: 0
Total packets: 52 @ 0.60/sec, sequence breaks: 0, time tears: 0
```

Information about each analyzed data packet is output on a single line. The fields from the left to the right are:

- The GRF channel name as network:station:component
- The GRF unit identifier, channel number, and sequence number
- The corrected initial sample time (IST) for the packet. Note that for this example, ISO8601 'Ordinal' date and time representations are used (`DATE_FORMAT=Ordinal`).
- The IST quality (see below for a description of these values)
- The corrected sampling rate as samples/second
- The sampling rate quality (see below for a description of these values)
- The difference between the expected IST and the actual IST for this packet

The expected time of the next packet is computed based on the IST, sampling rate, and number of samples from the current packet. The difference between the expected and actual IST is a direct observation of timing continuity.

When the program exits, a summary line for each GRF unit that was seen is displayed followed by totals. The fields in these summary lines from left to right are as follows:

- The GRF unit identifier of the station
- The number of packets received from that station. This is the total number of packets, not just for the channel analyzed for timing
- The average rate at which packets from this station were received in packets per second
- The total number of sequence breaks seen in the data from this station
- The total number of time tears seen from this station. A time tear is defined as a difference between the actual and expected time of packet that is greater than plus or minus one half of the sampling period

The last line is simply the total number of packets, sequence errors, and time tears seen on all stations analyzed. The packet rate is the overall packet rate.

4.7 GRF time quality indicators

GRF time quality indicators are used to indicate the quality of both the IST and the sampling rate. The quality indicator for the IST has the following meaning:

Quality	Description
Unknown	The quality of the time is completely unknown.
Bad	The time is known to be invalid.
Poor	The time quality has been good but the timing system is currently free running. This typically indicates that a GPS is currently unlocked.
Good	The timing system is actively correcting time relative to its reference. This typically means that a GPS is locked and that time is actively being corrected to the best of the system's ability.
Very Good	The time is known to be within +/- 500 microseconds of UTC.

Figure 18 GRF time qualities.

In the example above, the 'Poor' quality for the sampling rate means that the digitizer has not run long enough for GRFd to compute a precise correction for the sampling rate based on the PPS signal analysis. Instead, the nominal sampling rate is being used. Once corrections are being made, the sampling rate quality will be promoted to 'Good'.

Chapter 5

GRFConvert

5.1 Overview

GRFConvert is a GRF data conversion client that converts GRF data into various standard analysis formats. These formats currently include:

- GSE2.0 waveform segments
- SAC binary records in big or little endian byte order
- Mini-SEED (data-only SEED) volumes
- PASSCAL modified SEG-Y in big or little endian byte order
- MATLAB version 4 MAT-file format variable files
- PC-SUDS version 1.51 data files
- ASCII data files

Support for additional formats is being added all the time. Please contact us if the format that you require is not listed here. We intend to support all "standard" analysis formats, meaning all analysis formats that are generally used within the seismic community.

In addition to converting data to analysis formats, GRFConvert can also be used to compute and display various statistics about the waveform data it is processing. This information can be very useful in testing and/or quality monitoring.

Like all GRF clients, GRFConvert can connect to a GRF server via a socket connection or read GRF packet images from a file.

5.2 Running GRFConvert

GRFConvert is controlled entirely from the command line. The following help screen is displayed when the `-h` option is specified on the command line:

```
C:\grf>grfconvert -h
GRFConvert version 1.2.0 (May  1 2003 16:00:58), pid:1868

Usage: grfconvert [-hqdblNSvx] [-n name_format] [-f format] [-r rec_len]
                  [-t seconds] [-L list_file] [-o path] -s socket|image_file

   -h           Help display.
   -q           Quiet logging.
   -d           Debug logging.
   -b           Big endian output, if appropriate. (native)
   -l           Little endian output, if appropriate. (native)
   -N           If format is ASCII, don't include header info.
   -S           If format is ASCII, use simple single column format.
   -v           If format is ASCII or STATS, output samples in volts.
   -x           Drop data packets with 'unknown' or 'bad' time quality.
   -n format    Output file naming format string.
   -f format    Output data format:
                   ASCII, (GSE2.0), SAC, SEG-Y, SEED, SUDS, MATLAB, or STATS.
   -r rec_len   Output record length in seconds. (3600 seconds)
   -t seconds   Set socket read timeout. (30 seconds)
   -L list_file ASCII list file to append output filenames to. (none)
   -o path      Output directory. (./)
   -s socket    Socket endpoint to read GRF packets from.
                port number defaults to 3757 if not specified.
   image_file   GRF packet image file to read packets from.

[] = optional, () = default, | = mutually exclusive.
```

The options, or switches, are used to control various behaviors of the software. Note that because this software runs on a variety of platforms and operating systems, options always begin with a hyphen (–), never a solidus (/). Note also that switches, that is, options that are not followed by an argument, can be combined. For example, to specify both `-v` and `-b`, you may specify `-vb`. The space between an option and a required argument is optional. For example, to specify an output format with the `-f` option, you could specify either `-f ascii` or `-fascii`.

To stop the program press 'Ctrl-C', 'Ctrl-break', or send the TERM signal using the `kill` command if it is available.

5.3 Specifying the source of GRF data

GRF packets may be read from a GRF server via a socket connection or from a GRF packet image file. Use the `-s` *`endpoint`* option or the `filename` of a packet image file to specify the source of GRF data. Note that the program can read from only one data source at a time.

As an example, to connect to a GRF server on the local machine at the default port number, you might use the following command:

```
linux % grfconvert -s localhost
```

To specify a port number other than the default (3757), append a colon (:) followed by the decimal port number to the address. For example, to connect to a GRF server on localhost at port 49643, you could use the following command:

```
linux % grfconvert -s localhost:49643
```

To read GRF packets from an image file named `data.grf`, you might use the following command:

```
linux % grfconvert data.grf
```

5.4 Output path, data format, and record length

By default, output data files will be created in the current working directory. Use the `-o` *`pathname`* option to specify a different output directory.

The default output data format is GSE2.0. Use the `-f` *`format`* option, where *`format`* is `ascii`, `sac`, `seed`, `seg-y`, `suds`, or `matlab`, to specify other formats. Binary formats use the host native byte order by default. This means that SAC records created on Linux running on Intel hardware, for example, will be in little endian byte order. To create these records in big endian byte order, use the `-b` option.

Use the `-r` *`seconds`* option to specify the maximum length of output records. By default, records up to 3600 seconds (1 hour) will be created. GRFConvert will simply end the current record and start a new one once this length is reached. Note that records are always broken at the nearest packet boundary, not at the sample on the time when the record reaches the specified length.

As an example, to connect to a server on localhost and generate continuous fifteen minute GSE2.0 records in the /data directory, you might use the following command:

```
linux % grfconvert -r 900 -o /data -s localhost
```

To create ten minute long, big endian, SAC records, you might use this command:

```
linux % grfconvert -bf sac -r 600 -s localhost
```

5.5 Statistical analysis of waveform data

There is a special output format specifier named `stats`. When the `-f stats` option is specified, GRFConvert does not generate any output data files. Instead, the waveform data records are analyzed and per channel data statistics are logged.

These statistics include the mean, the range, and the standard deviation (sigma) of the waveform sample data. Some additional information is included in the log output including the GRF unit ID and channel number, the initial sample time, sampling rate, time and rate qualities, and the length of the record.

Use the `-r` option to control the length of the record segments that are to be analyzed. The following example is analyzing one minute records and logging the results:

```
C:\grf>grfconvert -f stats -s netdas4ch -r 60
GRFConvert version 1.2.0 (May  1 2003 16:00:58), pid:1248

GRF source:
 Type:         Socket
 Name:         192.168.1.5
Output format: Computing per channel data statistics
Output path:   ./
Record length: 60 seconds

Connected to unit 1000 (NetDAS-4CH) at 192.168.1.5:3757
Channel name: XX:4CH:EHZ (1000:2)
 Time: 2003-123T05:03:24.093Z (Good), rate: 99.998 sps (Good)
 Number of samples: 6000, 60.001 seconds
 Min: -481, Max -437, Range: 45, Mean: -458.80, Sigma: 5.77
Channel name: XX:4CH:EHE (1000:4)
 Time: 2003-123T05:03:27.273Z (Good), rate: 99.998 sps (Good)
 Number of samples: 6000, 60.001 seconds
```

```
 Min: -73, Max -45, Range: 29, Mean: -59.57, Sigma: 4.45
Channel name: XX:4CH:EHN (1000:3)
 Time: 2003-123T05:03:27.553Z (Good), rate: 99.998 sps (Good)
 Number of samples: 6000, 60.001 seconds
 Min: 223, Max 256, Range: 34, Mean: 238.20, Sigma: 4.53
```

`Min` and `Max` are the minimum and maximum sample value found in x.

`Range` is the peak-to-peak amplitude in x.

$$range = \max(x) - \min(x) + 1$$

`Mean` is the arithmetic mean amplitude, x-bar:

$$\bar{x} = \frac{1}{N} \cdot \sum_{i=1}^{N} x_i$$

`Sigma` is the standard deviation:

$$\sigma = \sqrt{\frac{1}{N-1} \cdot \sum_{i=1}^{N} (x_i - \bar{x})^2}$$

where: x is the waveform sample data in digital counts or volts (`-v` option).
N is the number of samples in the record segment.

5.6 Triggered vs. continuous data

GRFTrig inserts GRF informational messages into the packet stream containing channel and network trigger and detrigger times. When GRFConvert encounters network detrigger messages, it closes the current output record. This means that if GRFConvert is connected to GRFTrig, each network trigger will generate one output data record.

If GRFConvert is connected to GRFd or GRFHub, continuous data is converted. The records are segmented based on the record length value as specified using the `-r` option. See the previous section for more information about `-r` option.

5.7 Output file naming format specifiers

Each output format has a default file naming format specification that is given below. This default specification can be overridden by using the `-n` *`format`* option. *`format`* is specified as quoted string made up of ASCII characters and the following format specifiers:

Specifier	Description
%Y	four digit year, e.g., '2001'
%y	two digit year, .e.g., '01'
%M	two digit month, e.g., '01'
%D	two digit day of month, '01'
%d	three digit day of year, e.g., '001'
%h	two digit hour of day, e.g., '00'
%m	two digit minute of hour, e.g., '00'
%s	two digit integer second of minute, e.g., '00'
%ms	three digit integer millisecond of second, e.g., '000'
%us	six digit integer microsecond of second, e.g., '000000'
%l	record length as integer seconds (variable length)
%lm	three digit integer millisecond of record length, e.g., '000'
%lu	six digit integer microsecond of record length, e.g., '000000'

%u	GRF unit number (variable length)
%c	GRF channel number (variable length)
%N	GRF network identifier (<= 7 characters)
%S	GRF station identifier (<= 15 characters)
%C	GRF component identifier (<= 7 characters)
%%	Literal percent sign (%)

Figure 19 GRFConvert output filename format specifiers.

The time referenced by the first ten specifiers in the list is the initial sample time of the output record.

Note that some output formats generate one file per channel per record while others generate one file per record. It is important that output filename be unique to avoid file overwriting. See output format specific information below for more information about this issue.

As an example, the command:

```
linux % grfconvert -f ascii -n "%Y%d%h%m%s%ms_%N_%S_%C.ascii" -s localhost
```

Would create one ASCII data file per channel with names like:

```
2000362022332496_XX_STN2_EHZ.ascii
```

5.8 Format: GSE2.0

This is the default output data format. One output data file is generated per record containing a waveform segment for each channel. These files are named using the following default filename format:

```
"%Y%d%h%m%s%ms_%N.gse"
```

Note that because only one file is generated per record, only the network identifier is used in the default filename.

Sample data are encoded using the CM6 encoding[2]. This is an ASCII based encoding scheme using only the ASCII characters +,-,0 through 9, A through Z, and a through z. This format is non-binary and platform independent.

For each channel, a WID2 record is output that contains the station and component identifiers taken from the GRF channel name, a DAT2 record is output containing the sample data, and a CHK2 record is output containing the data checksum. Note that the WID2 record imposes a maximum length for the station identifier of five characters and a maximum length for the component identifier of three characters. If you will be working with GSE data, you should name your GRF channels so that characters will not be truncated in the conversion process.

The following fragment of a GSE2.0 data file named `2000347181114731_XX.gse` shows the records output for each channel:

2000347181114731_XX.gse

```
WID2 2000/12/12 18:11:14.731 Stn2  BHZ      CM6     1912  100.158691  1.00E+000  1.000         -1.0
0.0
DAT2
apAqvIcKUOV0uFl6UlAkm6Ur8kpDX8xBUw6r3XKn3nQkzPUu0h9gTyKsCoOrGUs1nEz9V8fGbHzAcFko
70Uo-i1xQV8kpTeDnTUwCv7tRY5zRZNXPUtBkqPt3ePh4knPWEku+VW7iTWJlV-pFUl6uLIUz+dTlVGk
```

lines deleted…

[2] All of the details about GSE2.0 are available in GSE Conference Room Paper 243 (GSE/CRP/243), Annex 3 - Formats and Protocols for Data Exchange, Section 4.5 - GSE2.0 Waveform Segments. This document is available on-line at http://www.pidc.org/web-gsett3/CRP-243.

```
7v+Vj+UtGw9kwIvJpFURgMtFfCUlMkoQpIkxBVq8vIVNvOxGkEfKUqQUJXCknCm6lX9UwIv1UxOi+n1k
vBp+eJW-d0u-klC7UnOcHkz9rCViTW2kwCsHs7VcDcN
CHK2     12903498
```

2000347181114731_XX.gse

Note that the WID2 record has wrapped to the next line, this is due to the formatting of this document.

5.9 Format: ASCII

When the ASCII output format is specified using the `-f ascii` option, output data files will be named using the following default filename format:

```
"%Y%d%h%m%s%ms_%N_%S_%C.ascii"
```

One ASCII data file per channel per record will be created. If you specify a file naming format specifier, by sure that the filenames created will be unique across all channels.

The following ASCII data file fragment show the header information that is included in output data files by default. Each line of the header begins with a pound sign (#). You may omit this header information by using the `-N` option on the command line.

counts.ascii

```
# GRF unit number:     1000
# GRF channel number:  4
# Network ID:          XX
# Station ID:          4CH
# Component ID:        EHE
# Initial sample time: 2002-10-23T19:10:49.859230Z Good
# Sampling rate:       100.000000 Poor
# Counts per volt:     398000
# Number of samples:   500
      -88      -88       -84      -96       -93      -84      -92      -89
      -85      -93       -89      -93      -100      -92      -89      -89
      -89      -89       -97      -96       -89      -92      -92      -97
lines deleted...
       -92      -89       -93     -100       -93      -89      -93     -100
      -100      -96       -93       -93      -88      -88      -88       -85
      -101      -96       -89       -88      -93      -96      -89       -89
```

counts.ascii

Sample data in digital counts are written in eight columns in row major order.

Sample data may be converted to volts by using the `-v` option. The following data file fragment shows data in volts:

volts.ascii

```
# GRF unit number:     1000
# GRF channel number:  4
# Network ID:          XX
# Station ID:          4CH
# Component ID:        EHE
# Initial sample time: 2002-10-23T19:11:44.860486Z Good
# Sampling rate:       100.000000 Poor
# Counts per volt:     398000
# Number of samples:   1500
-2.110553e-004 -2.135678e-004 -2.336683e-004 -2.336683e-004 -2.336683e-004
-2.236181e-004 -2.336683e-004 -2.336683e-004 -2.311558e-004 -2.336683e-004
-2.336683e-004 -2.236181e-004 -2.412060e-004 -2.336683e-004 -2.311558e-004
lines deleted...
-2.336683e-004 -2.336683e-004 -2.211055e-004 -2.311558e-004 -2.236181e-004
-2.311558e-004 -2.512563e-004 -2.437186e-004 -2.412060e-004 -2.211055e-004
-2.412060e-004 -2.412060e-004 -2.336683e-004 -2.512563e-004 -2.336683e-004
```

volts.ascii

Sample data in volts are written in five columns in row major order.

Simple single column output is available by specifying the `-S` (capital S) option. When the `-S` option is specified without the `-N` option, eleven useful header values appear before

the sample data in the output. Note that the italicized comments describing each of these fields below were added here and do not appear in actual output.

simple.ascii

```
1001          Integer unit ID number
3             Integer channel number on unit
XX:4CH:EHN    Network:station:component (NSC) identifier string or GRF name
2001          4 digit integer IST year
355           3 digit integer IST day of year
20            2 digit integer IST hour of day
18            2 digit integer IST minute of hour
55.793560     Floating point second of IST minute (microsecond resolution)
0.010000      Floating point sampling period or delta (seconds per sample, microsecond resolution)
419430        Integer counts per volt
30167         Integer number of samples that follow
+1279         Samples...
+283
+56
-809
-2320
-2993
. . .
```

simple.ascii

If the `-N` option is specified along with `-S`, these eleven header fields are omitted and the output will contain a single column of sample data. Note also that the `-v` option, when used along with `-S`, will cause the sample values to be expressed as volts instead of digital counts.

5.10 Format: SAC

When the SAC[3] output format is specified using the `-f sac` option, output data files will be named using the following default filename format:

```
"%Y%d%h%m%s%ms_%N_%S_%C.sac"
```

One SAC data file per channel per record will be created. If you specify a file naming format, please by sure that the filenames created will be unique across all channels.

Output SAC data files are binary and by default, are created in the host native byte order. Use the `-l` or `-b` options to specify little or big endian byte order as required.

The network, station, and component identifiers in SAC are limited to seven characters. As the GRF network and component identifiers are limited to seven characters, they pose no problem. However, the GRF station identifier can be as long as fifteen characters and it may be truncated during the conversion process.

5.11 Format: Mini-SEED

When the data-only SEED[4] or Mini-SEED volume output is specified using the `-f seed` option, output data files will be named using the following default filename format:

```
"%Y%d%h%m%s%ms_%N_%S_%C.mseed"
```

One Mini-SEED volume file per channel per record will be created. If you specify a file naming format, please by sure that the filenames created will be unique across all channels.

By definition, binary data within SEED volumes is in network or big endian byte order so the `-l` and `-b` options will have no effect.

---

[3] Additional information about SAC and its file formats is available on the SAC home page at http://www.llnl.gov/sac/.

[4] Additional information about SEED is available from the IRIS Consortium at http://www.iris.washington.edu/.

Only the first two characters of the GRF network name, the first five characters of the GRF station name, and the first three characters of the GRF component name will make it into the Mini-SEED volume. Care should be taken to use a naming convention that is compatible with these limits with this output format will be used.

5.12 Format: PASSCAL modified SEG-Y

When the PASSCAL modified SEG-Y[5] output format is specified using the `-f seg-y` option, output data files will be named using the following default filename format:

```
"%Y%d%h%m%s%ms_%N_%S_%C.segy"
```

One SEG-Y data file per channel per record will be created. If you specify a file naming format, please by sure that the filenames created will be unique across all channels.

Output SEGY data files are binary and by default, are created in the host native byte order. Use the `-l` or `-b` options to specify little or big endian byte order as required.

Only the first four characters of the GRF station identifier and the first two characters of the GRF component identifier are used in these records so careful GRF naming conventions should be used if data will be converted to this format.

5.13 Format: SUDS

When the SUDS[6] version 1.51 output data format is specified using the `-f suds` option, one file per record is created in the output directory. The data file contains all channels for the record. The file is named using the following filename format by default:

```
"%Y%d%h%m%s%ms_%N.suds"
```

Note that because only one file is generated per record, only the network identifier is used in the default filename.

By definition, the SUDS format is a binary, little endian (Intel) byte order format.

SUDS version 1.51 data files are created that contain the new SUDS_LONGIDENT structure that allows the use of long station/component identifiers. The long identifier structure will contain the complete GRF network:station:component identifier. Note that for backward compatibility, the short SUDS_STATIDENT fields are also generated using the first four characters of the GRF station identifier and the first five characters of the GRF component identifier and so careful GRF naming conventions should still be used when data are to be converted to this format.

5.14 Format: MATLAB variable files

When MATLAB version 4 MAT-file[7] format variable file output is specified using the `-f matlab` option, MATLAB variable files will be created in the output directory. These files will be named using the following default filename format:

```
"%Y%d%h%m%s%ms_%N_%S_%C.mat"
```

---

[5] Additional information about PASSCAL modified SEG-Y is available at http://www.passcal.nmt.edu/.

[6] Additional information about SUDS version 1.51 and the Win-SUDS Utilities is available from Banfill Software Engineering at http://www.banfill.net/suds.html.

[7] Additional information about this format is available from MathWorks at http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/matfile_format.pdf/.

One MATLAB variable file per channel per record will be created. If you specify a file naming format, please by sure that the filenames created will be unique across all channels.

These are binary files and by default, are created in the host native byte order. Use the `-l` or `-b` options to specify little or big endian byte order as required.

Each variable file contains three variables and a vector. A variable named '*network_station_component*_time' contains the initial sample time of the record as the number of seconds since the epoch (1970-01-01T00:00:00Z). A variable named '*network_station_component*_delta' contains the sampling period in seconds per sample. A variable named '*network_station_component*_volt' contains the counts per volt value.

The sample data are stored in a single column vector named '*network_station_component*'. *n* = rows(*network_station_component*) = the number of samples in the record.

Chapter 6

GRF2EW

6.1 Overview

GRF2EW is a simple GRF client application that converts GRF data packets to Earthworm TraceBuf messages and writes them to an Earthworm transport ring. The program also generates heartbeat messages at a user specified interval. The program is controlled primarily through an ASCII configuration file but it also accepts various command line options.

GRF2EW requires Earthworm version 6.1 or later and is currently only available on Win32 platforms.

6.2 Running GRF2EW

GRF2EW is invoked from the command line by typing `grf2ew` and pressing return. However, most Earthworm users will likely use the `startstop` program to control operation of the program along with other Earthworm modules. We will cover the configuration of the `startstop` program below.

A simple help screen is available by specifying the `-h` option:

```
C:\grf>grf2ew -h
Usage: grf2ew [-hdc] [-s ip_addr[:port] | -f input_file]
           [-t seconds] [-r output_ring] [-H seconds]
           [configuration file]

   -h  Help display.
   -d  Debug logging.
   -c  Apply sampling rate corrections (No).
   -s  IP address and port number to connect to for input data.
          port number defaults to 3757 if not otherwise specified.
   -f  File to read for input data.
   -t  Socket read timeout in seconds. (30)
   -r  Output ring for TRACE_BUF messages. (WAVE_RING)
   -H  Heartbeat interval in seconds. (10).

The configuration file argument defaults to './grf2ew.d' if not otherwise
specified.

[] = optional, () = default, | = mutually exclusive.
```

All of these command line options are equivalent to their corresponding entries in the configuration file.

Driving GRF2EW with startstop

In order to use the `startstop` program to drive GRF2EW you will need to create a configuration file (see below) in your Earthworm parameters directory and copy the `grf2ew.exe` executable to the Earthworm bin directory. Typically, the configuration file will be named:

```
\Earthworm\v6.1\params\grf2ew.d
```

and the executable:

```
\Earthworm\v6.1\bin\grf2ew.exe
```

Once these are in place, edit the `startstop_nt.d` file and add the `grf2ew` module. Below is an example fragment of this file that runs only `wave_serverV` and `grf2ew`:

startstop_nt.d

```
# startstop.d
#
nRing               1
Ring   WAVE_RING 1024
#
MyModuleId        MOD_STARTSTOP  # Module Id for this program
HeartbeatInt      15             # Heartbeat interval in seconds
MyPriorityClass   Normal         # For startstop
LogFile           1              # 1=write a log file to disk, 0=don't
KillDelay         10             # number of seconds to wait on shutdown
                                 #  for a child process to self-terminate
                                 #  before killing it
#
Process          "wave_serverV wave_serverV.d"
PriorityClass     Normal
ThreadPriority    Normal
Display           NoNewConsole

Process          "grf2ew grf2ew.d"
PriorityClass     Normal
ThreadPriority    Normal
Display           NoNewConsole
```

startstop_nt.d

Figure 20 Example startstop_nt.d file.

Once things are properly configured, `startstop` will drive `grf2ew` along with all other Earthworm modules.

6.3 Configuring GRF2EW

The configuration file allows you to configure and control all aspects of the programs behavior. This simple ASCII file can be edited using your favorite editor.

The name of the configuration is passed as an argument on the command line at program startup. It is an Earthworm convention to give module configuration files an extension of `.d` and therefore the default configuration filename is `grf2ew.d`.

The configuration file is made up of keywords followed by an entry. Some keywords define groups of entries and curly braces {} are used to delimit these groups. White space characters or commas separate keywords and entries. Comments may appear anywhere in the file and always begin with a pound sign '#'.

The file is made of two basic groups of entries:

- Server – This group specifies the server that GRF2EW will connect to as a client.
- Earthworm – This group defines various Earthworm parameters.

Throughout the example configuration file, the default values for each entry are show in end-of-line comments inside parentheses.

The 'Server' group

The server group defines the source server that the program will connect to. This group contains two keywords:

grf2ew.d – server group

```
# $Id: grf2ew.d,v 1.2 2002/07/19 21:12:44 cvs Exp $ */
# Copyright (C) 2000-2002 - Robert Banfill - All rights reserved.
# Configuration file for grf2ew version 1.0.5 or later
# Defaults for each entry are shown in ()

# The logging group is now deprecated as logging in now performed
# using the Earthworm logging facility.

# GRF server settings...
Server {
    Endpoint          localhost         # Server endpoint to connect to
    ReadTimeout       30                # Socket read timeout in seconds (5)
}
```

grf2ew.d – server group

Figure 21 grf2ew.d 'Server' group entries.

The 'Endpoint' entry

This entry is used to specify an upstream server endpoint that will be connected to as a client. The program will maintain this connection for the life of the program.

An endpoint may be specified as an IP number in dotted decimal form or as a domain name. The port number may be specified by appending a colon (:) followed by the decimal port number to the address. If no port is specified, the default port (3757) is used. For example, '192.168.1.1' is equivalent to '192.168.1.1:3757'.

The 'ReadTimeout' entry

This entry is used to specify the timeout value in seconds for upstream client socket connections. If the network connection to the server is very slow or high latency, you may need to increase this value. Under normal operating conditions, the default value of 30 seconds should be appropriate.

The 'Earthworm' group

This group is used to specify the parameters of the Earthworm system. It has three keyword entries:

grf2ew.d – earthworm group

```
# Earthworm settings...
Earthworm {
    Ring              WAVE_RING  # Destination ring for TRACEBUF messages (WAVE_RING)
    ModuleName        MOD_GRF2EW # Our module name (MOD_GRF2EW)
#    InstallationID    INST_WILD  # Installation identifier (INST_WILD).  May be entered
                                 #   as a lookup string or decimal value 0-255.  If the
                                 #   environment variable EW_INSTALLATION is defined,
                                 #   its contents are used.  This configuration entry
                                 #   overrides the environment variable setting.
    Heartbeat         15         # Heartbeat interval in seconds (15)
}
```

grf2ew.d – earthworm group

Figure 22 grf2ew.d 'Earthworm' group entries.

The 'Ring' entry

This entry is used to specify the Earthworm ring that GRF2EW will attach to and write TraceBuf messages. The startstop program will create this ring as specified in your `startstop.d` configuration file.

GRF names are copied to Earthworm SCN names in the output TraceBuf messages. Note that the station name is limited to seven characters in TraceBuf messages so you should carefully choose GRF station names to avoid truncation in the conversion process.

The 'InstallationID' entry

The installation identifier may be specified as either a string that will be looked up in the `earthworm_global.d` file, or an unsigned integer value in the range 0-255. This entry defaults to '`INST_WILD`', which resolves to 0.

At program startup, installation identifier is set to the default value. The environment is then searched for a variable named EW_INSTALLATION and if found, the installation identifier is set to its value. If the InstallationID entry is then found in the configuration file, the installation identifier is set to its value overriding the environment variable setting.

The 'ModuleName' entry

This the Earthworm module name used to identify the GRF2EW program. This module name must be defined in your `earthworm.d` file.

The 'Heartbeat' entry

This entry is used to specify the interval in seconds at which the GRF2EW program will generate heartbeat messages to the output ring.

Appendix A

GRFCheck

A.1 Overview

GRFCheck is a simple command line tool that checks the integrity of GRF packet image files. The program will process a GRF packet image file and:

- Detect any missing packets by testing the continuity of the GRF packet sequence number on each packet. If a sequence break is detected, the current output file is closed and a new output packet image file is created.
- Detect timing discontinuities (time tears) within continuous waveform data stored in GRF data packets. When a time tear is detected, the current output file is closed and a new output packet image file is created.
- Break up large packet image files into smaller files based on the time length of waveforms contained in the data or the number of packets.

The program reads packets from its input file, processes each of these packets, and then writes them to an output file. These output files can be named according to a file name format specifier that you provide.

A.2 Running GRFCheck

GRFCheck is controlled entirely from the command line. A help screen that summarizes the various program options is available by specifying the `-h` option on the command line.

```
C:\grf>grfcheck -h
GRFCheck version 1.2.0 (May  1 2003 16:00:36), pid:1868

Usage: grfcheck [-hv] [-l length] [-f filename_format]
       [-o output_path] input_file

   -h  Help display.
   -v  Verbose logging. (Off)
   -f  Output filename format. ("%Y%d%h%m%s%ms_%l%lm_%u.grf")
   -l  Max output record length in seconds. (3600)
   -n  Max output record length in packets. (Unlimited)
   -o  Output path. (./)

[] = optional, () = default, | = mutually exclusive.
```

The program expects one mandatory argument on the command line, the name of the input file. This is the name of the GRF packet image file that will be processed.

The `-v` option simply causes the program output detailed information about its activities to the console. This is off by default and console output is greatly reduced.

The `-f format` option allows you to specify the file naming format string that will be used when the program generates output files. The format string is made up of format specifiers, which begin with a percent sign (%) and ASCII characters.

Specifier	Meaning
%Y	The four digit year, '2002'
%y	The two digit year, '02'
%M	The two digit month, '01'
%D	The two digit day of month, '01'
%d	The three digit day of year, '001'
%h	The two digit hour of day, '00'
%m	The two digit minute of hour, '00'
%s	The two digit integer second of minute, '00'
%ms	The three digit integer millisecond of second, '000'
%us	The six digit integer microsecond of second, '000000'
%l	The record length as integer seconds (variable length), '0'
%lm	The three digit integer millisecond of record length, '001'
%lu	The six digit integer microsecond of record length, '000001'
%u	The GRF unit number (variable length), '0'
%%	A literal percent sign, '%'

Figure 23 GRFCheck output file name format specifiers.

The default output file name format is "%Y%d%h%m%s%ms_%l%lm_%u.grf" which generates names like:

```
2002037120000000_120000_1001.grf
```

This time and length are represented to milliseconds and the GRF unit ID number is at the end of the file name.

Use the `-l` option to limit the time length of waveform data in output files. By default, output files will be limited in length to one hour (3600 seconds).

Use the `-n` options to limit the number of packets that an output file can contain. By default, there is no limit.

Use the `-o pathname` option to specify an alternate output directory. By default, output files are created in the current working directory.

Appendix B

DSTGen

B.1 Overview

The DSTGen program is used to simulate a single digitizer generating three channels of 16-bit waveform data as an USGS-DST style digital stream. This program writes these data to a standard asynchronous serial interface and allows the user to control the sampling rate, frequency, and amplitude of the generated function data.

B.2 Running DSTGen

The DSTGen program is controlled entirely from the command line. A help screen that summarizes the various program options is available by specifying the –h option on the command line.

```
linux % dstgen -h
dstgen version 1.2.0 (May  1 2003 16:00:01), pid:1456

Usage: dstgen [-a amplitude] [-f frequency] [-r rate]
              device[,speed,parity,databits,stopbits]

   -h           Help display.
   -a           Function amplitude in counts. (10000)
   -f           Function frequency in hertz. (0.10)
   -r           Sampling rate in hertz. (100.00)

The port parameters default to '9600,n,8,1'

[] = optional, () = default, | = mutually exclusive.
```

The program generates three channels of data using the following functions:

- **Channel 1** – A sine wave
- **Channel 2** – A triangle wave
- **Channel 3** – A square wave

The frequency and amplitude of these functions are always the same and are controlled by specifying the `-f` and/or `-a` options. All three functions pass through zero at the same point. The default frequency is 0.1 Hz or 10 seconds and the default amplitude is 10000 counts.

The sampling rate is controlled by specifying the `-r` option. Note that at 9600 baud, 100 samples/second (sps), the default, is the maximum rate. At 19200 baud, 200 sps is the maximum.

The device name and line settings are specified as a string argument. For example, to specify the device as com2 at 19.2k baud on a Windows system, you might use the following command:

```
C:\grf>dstgen com2,19200
```

Note that there are no spaces between `com2` and the comma and `19200`. On a Linux system, you might use:

```
linux % dstgen /dev/ttyS0,19200
```

The following figure shows waveforms from a single DST station viewed using the GRFViewer program.

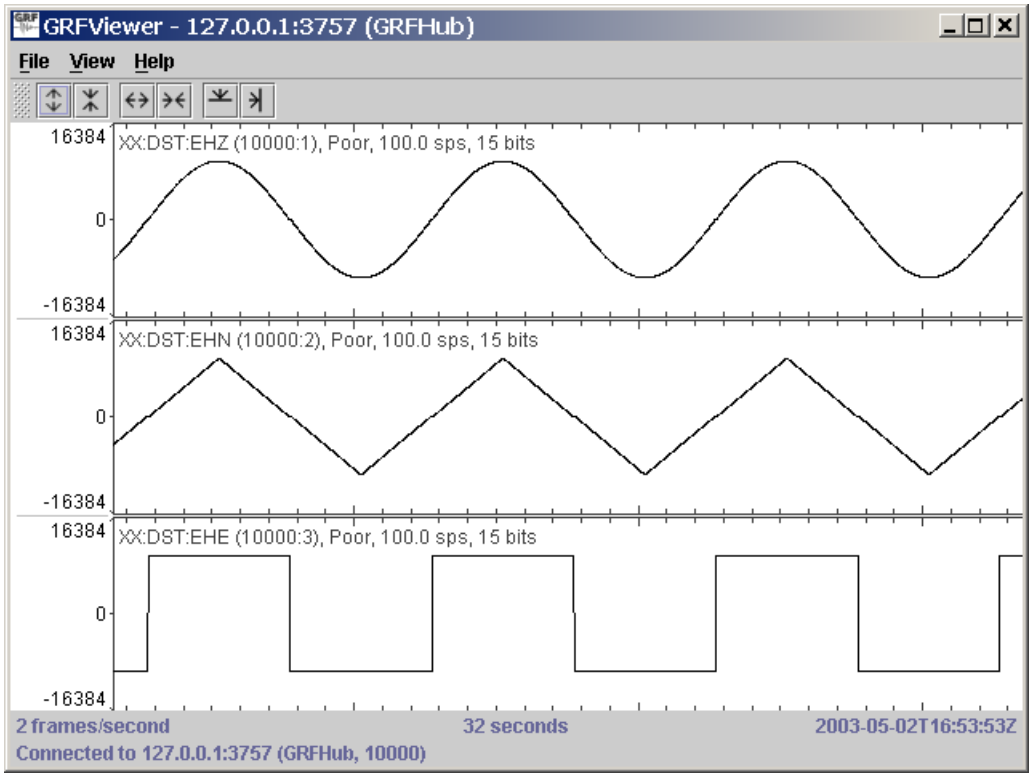

Figure 24 DST waveform data generated by DSTGen in GRFViewer.

The DSTGen program generated these signals using its default settings.

Appendix C

GNU General Public License

Version 2, June 1991

59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

C.1 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by

someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

C.2 Terms and conditions for copying, distribution and modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print

such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this

License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by

the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Index

Banfill Software Engineering
http://www.banfill.net/
(907) 835-4122