Hyper-Logger User Guide

Version 1

October 21, 2024

Developed by Transportation Laboratories

Version History

Version	Date	Revision Description
1	10/21/2024	Initial publication

Document Conventions

This document uses the following typographic and syntax conventions.

- Commands, command options, file names or any user-entered input appear in Courier type. Variables appear in Courier italic type.

  Example: Select the `cmdapp-`*`relVersion-buildVersion`*`.zip` file....

- User interface elements, such as field names, button names, menus, menu commands, and items in clickable dropdown lists, appear in Arial bold type.

  Example: **Type**: Click **Select Type** to display drop-down menu options.

- Cross-references are designated in Arial italics.

  Example: Refer to *Figure 1*…

- Click intra-document cross-references and page references to display the stated destination.

  Example: Refer to *Section 1* Overview *on page 1.*

  The clickable cross-references in the preceding example are *1*, *Overview*, and on page 1.

CyFlex Documentation

CyFlex documentation is available at https://cyflex.com/. View **Help & Docs** topics or use the **Search** facility to find topics of interest.

Table of Contents

1 OVERVIEW **1**

1.1 HYPER-LOGGER COMMANDS 1

1.2 ADVANTAGES AND DISADVANTAGES 1

1.2.1 Advantages 1

1.2.2 Disadvantages 1

2 USING `HOGGER` **2**

2.1 STARTING THE APPLICATION 2

2.2 SPECIFICATION FILE 2

2.3 COMPUTED EXPRESSION FORMAT 5

2.4 `HOGGER` APPLICATION BEHAVIOR TIPS 5

2.4.1 Database Operation 5

2.4.2 File Naming Structure 5

2.4.3 Triggering Rotations 6

3 USING `HOGGER_OUT` **7**

3.1 STARTING THE APPLICATION 7

3.2 SPECIFICATION FILE 7

3.3 `HOGGER_OUT` APPLICATION BEHAVIOR TIPS 12

3.3.1 CyFlex Libraries 12

3.3.2 Output Format 12

3.3.3 Data Options 12

1 Overview

The Hyper-Logger application is a data logging tool that records Cyflex variables into an SQLite3 database. Its intent is to provide logging capability independent of CyFlex timers. This enables logging performance at rates higher than any existing CyFlex timer is capable of without limitation by the number of samples and RAM on the system like the existing `floger` application.

Note that this function is available with CyFlex 7.1 and subsequent releases.

1.1 Hyper-Logger Commands

The `hogger` command logs and records Cyflex variables into an SQLite3 database. Refer to *Section 2 Using* `hogger` on page 2. The SQLite3 database itself is not viewable.

The `hogger_out` command converts the SQLite3 database to a viewable and usable format. Refer to Section 3 Using `hogger_out` on page 7.

1.2 Advantages and Disadvantages

The following are some advantages and disadvantages compared to `floger`.

1.2.1 Advantages

- When using buffered variables, the `hogger` command can run slower than the variable update rate and still capture all the changes.
- Use of buffered variables eliminates jitter in value capture.
- Data storage is limited by hard drive space, not memory.
- Each variable also contains the timestamp of its last update when captured.
- Variables that do not update are not stored to save space.
- Database files are rotated to prevent a single large database that is unmanageable.

1.2.2 Disadvantages

- Viewing a running database file is not possible.
- Large database file are split so viewing all data needs to have files concatenated together.
- No run time statistics are performed.

2 Using `hogger`

Use the `hogger` application to log and record Cyflex variables into an SQLite3 database.

2.1 Starting the Application

Start the `hogger` application in the `go.scp` or on the command line interface.
Example:

```
$ hogger /specs/hogr_spec.1 &
```

Refer to cyflex.com usage help for `hogger` for related information.

2.2 Specification File

The specification file normally contains the required options for `hogger`. Logging is controlled through the specification of events. The list of variables to log is supplied in the specification file. A sample specification and keyword use file is located in `/cyflex/specs.def/hogr_spec.def` and is also provided below.

```
#-----------------------------------------------------------------------------
#
#            NOTE:
#               !! Keywords that are marked with !! will accept
#                  any valid computed expression
#                  See COMPUTED EXPRESSION FORMAT at end of document
#
#               ** Keywords that are marked with ** are required
#                  All other keywords are optional
#-----------------------------------------------------------------------------
#
#      @DESCRIPTION   !! **
#         My_Test           <A title to be written to the output file>
#
#      @FILENAME      !! **
#         file_name         <the contents of the CYFLEX variable
#                            will be used as the name of file that
#                            will contain the logged data. This is
#                            saved as an archived database with a
#                            datetime stamp appended after the given
#                            filename.>
#      @START_EVENT
#         start_it          <the name of the event that will start
#                            the data acquisition. The event will be
#                            created if it doesn't exist. If not>
#                            specified the sampling will start
#                            immediately
#      @STOP_EVENT
#         stop_it           <the name of the event that will stop
#                            the data acquisition. The event will be
#                            created if it doesn't exist>
#       @RELEASE_EVENT
#          all_done          <the name of the event that will cause
#                             the loger task to exit. The event will be
#                             created if it doesn't exist>
#       @DONE_EVENT
#          all_done          <the name of the event that will be set
#                             when the data acquisition is complete.
#                             The event will be created if it doesn't
#                             exist>
#       @DB_ROTATE_EVENT
#          rotate_it         <the name of the event that will cause
#                             the hogger task to rotate the database.
#                             The event will be created if it doesn't
#                             exist>
#       @DB_ROTATE_COUNT
#          100000            <the number of inserts to acquire before
#                             rotating the database. If left blank
#                             the database will use the default of
#                             3,600,000 inserts before rotating. The
#                             minimum is 10000.>
#       @SCAN_INTERVAL !! **
#          10[sec]           < the time between hogger sampling which
#                              may be faster or slower than actual
#                              acquired data.>
#       @ENABLE        !!
#          loggin_ok         <the ASSET logical variable that is used
#                             to enable/disable logging data.>
#       @LOGGING_ACTIVE_LABEL
#          loggin_active     <the ASSET logical variable that will be set
#                             TRUE when data is being logged.>
#       @SYNC_EVENT
#         log_it_now         <the name of the event that will cause
#                             data to be logged. The event will be
#                             created if it doesn't exist>
#       @MAX_SCANS     !!
#          1000              <The maximum number of times to sample
#                             the channels and write their values to
#                             the output file. If left blank the the
#                             number of scans will be infinite and
#                             logging is limited by disk space.>
#       @SCAN_LIST        **
#         label_1            < the ASSET variable to be logged
#         label_2  %10.4f    < An optional format may be specified>
#                              NOTE : this won't do anything
#         label_2[units]     < Optional units may be specified for
#                              each label>
#         label_2 .MX        < Statistical member may be specified>
#                              when either @LOG_STATISTICS or
#                              @RUNNING_AVERAGE keywords are specified>
#
#         label_3 LOG_DIGITAL_DESCP < 'label_3 is a LOGICAL_VARIABLE and
#                                     description should be logged
#                                     instead of 0 or 1.
#                              NOTE : this won't do anything
#                            < The optional format, statistical member
#                              and LOG_DIGITAL_DESCP may be specified
#                              in any order.  If units are specified
#                              they must immediately follow the label
#                              name and be enclosed in []>
#      @REG_NAME
#        mylogr              < the name that should be used to register
#                              this instance of floger with the operating
#                              system. If this entry is not found the
#                              name of the spec file being read will be
#                              used as the registered name.
#      @OUTPUT_PATH
#         /data/mydata       < the directory path specifies where to
#                              write the output file. This entry may
#                              be a STRING_VARIABLE that contains the
#                              directory path for the output file.
#                              Default: /data/PC_format
#-----------------------------------------------------------------------------
##############################################################################
# Example of a valid spec file
##############################################################################

@DESCRIPTION
'test hogger'

@FILENAME
'test_hogger_data'

@SCAN_INTERVAL
10[msec]

@START_EVENT
e_start_hogging

@STOP_EVENT
e_stop_hogging

@RELEASE_EVENT
e_release_hogging

@LOGGING_ACTIVE_LABEL
Test_Hogr_Sts

@DB_ROTATE_COUNT
50000

@DB_ROTATE_EVENT
e_rotate_hogger

@WRITE_EXTERNAL_HEADER
/specs/rc_external_header
```

```
@SCAN_LIST
# ASSET variable label[optional units] [optional *format or statistical
member]
rc_volts
rc_logi1
rc_sine
rc_sine_c
rc_100[m]
rc_daq

$
```

2.3 Computed Expression Format

Computed expressions must be enclosed with double quotes ("). A literal string must be enclosed with a single quote ('). Strings maybe concatenated by using the plus (+) sign.

For example, assume the following computed expression was entered for the test description:

```
"'Engine model = ' + model + ' S/N = ' + serial"
```

Given that the CyFlex string variable <*model*> had a value of `<Enforcer 02>` and the string variable <*serial*> had a value <`14014957`>, the test description for the loger file would be equal to:

```
Engine model = Enforcer 02 S/N = 14014957
```

2.4 `hogger` Application Behavior Tips

2.4.1 Database Operation

The database uses a rotation technique to expedite file rotation time. When doing this, several auxiliary files are created at run time with similar names to the user-specified filename. Removing any of these at run time creates an exposure for catastrophic failure.

2.4.2 File Naming Structure

To prevent file collisions and loss of data, the supplied filename is appended with `DyyyymmddThhmmssPnnnnnnnn-n.db.xz` after rotation or application completion.

Before this the `.xz` will not be on the end. In addition:

- `D` specifies the date.
- `T` is time to the second.
- `P` is the pid/tid of the program.
- The final `-n` is a rotation number.
- `db` is for the SQL database.
- `xz` is the archive type after it is archived.

This file naming structure prevents collisions from multiple applications being started at the same time using the same file name (i.e., the same spec file was used).

2.4.3 Triggering Rotations

Do not trigger rotations faster than one second. The application will try to prevent rotation more than once every minute. Rapid rotation (like clock manipulation) creates exposure to adverse conditions.

3 Using `hogger_out`

Use the `hogger_out` application to convert the SQLite3 database created by the `hogger` application to a viewable and usable format.

3.1 Starting the Application

Start the `hogger_out` application on the command line interface. For example:

```
$ hogger_out -s /specs/hogr_out_spec.1 -i
/data/PC_format/database.db
```

Refer to cyflex.com usage help for `hogger_out`.

3.2 Specification File

A default or customized specification file is not required. Specify a `hogger`-created database on the command line to create a generic `floger` style output file by using default parameters.

A sample specification and keyword use file is located in `/cyflex/specs.def/hogr_out_spec.def` and is also provided below.

```
#----------------------------------------------------------------------------
#
#   NOTE:
#
#       **  Keywords that are marked with ** are required
#           All other keywords are optional
#
#
#----------------------------------------------------------------------------
#
#   @DESCRIPTION
#       my added description
#                          < A title to be written to the database file.
#                            This is given in hogger, but is only written
#                            when the keyword is present and a value for
#                            the description exists in the database.  If
#                            a description is also given here it is added
#                            to the description from the database. >
#
#   @INPUT_FILENAME       **
#       file_name          < The database file name created by hogger.
#                            This will contain the logged data. It is
#                            required either here or as a command line
#                            argument. >
#
#   @OUTPUT_FILENAME
#       file_name          < The name of the output file. This can be
#                            a relative or absolute path. If it is
#                            relative and a path is given in the
#                            @OUTPUT_PATH entry, then the path in the
#                            @OUTPUT_PATH entry will be prepended to
#                            the file name. If the file name is not
#                            given, the name of the input file will
#                            be used as the name of file that will
#                           contain the logged data.
#                           Default order:
#                               /data/PC_format/hogger_put.dat
#                               /tmp/hogger_out.dat when no spec
#                               ./hogger_out.dat >
#
#
#   @OUTPUT_PATH
#       /data/mydata      < The directory path specifies where to
#                           write the output file.
#                           Default order:
#                               /data/PC_format/
#                               /tmp/ >
#
#   @LOG_DIGITAL_DESCRIPTION
#       yes               < When logging a LOGICAL_VARIABLE log the
#                           TRUE or FALSE description instead of 1 or
#                           0. The entry 'yes' is optional. If no entry
#                           is found, 'yes' is assumed.
#
#   @PACKED
#       Yes               < Should the output file be a packed comma
#                           separated format? The entry
#                           'yes' is optional. If only the keyword is
#                           entered, then the output is packed. Entry
#                           is case insensitive>
#
#   @OVERWRITE_OUTPUT
#       Yes               < Should an existing output file be over
#                           written on startup? The entry
#                           'yes' is optional. If only the keyword is
#                           entered, then the output is overwritten.
#                           'no' is optional ans the default when omitting
#                           the keyword. Entry is case insensitive>
#
#   @WRITE_EXTERNAL_HEADER
#       filename          < the name of a file that will be copied to the
#                          output file when the output file is written to.
#                          This prepends the header to the output file. >
#
#   @TIME_FORMAT
#
#       DEFAULT_FORMAT    < the format for the date/time will be
#                           MM/DD/YYYY hh:mm:ss. For example,
#                           11/11/2004 11:10:11
#
#                           NOTE: If no entry follows the keyword the format
#                           will be DEFAULT_FORMAT.
#
#       HI_RES            < the format for the date/time will be
#                           MM/DD/YYYY hh:mm:ss.XXXXXX, where XXXXXX is
#                           microseconds. For example,
#                           11/11/2004 11:10:11.161
#
```

```
#       OLD_FORMAT        < the format of the time in the output file
#                           should be "MMM DD hh:mm:ss". For example,
#                           Jan 04 11:01:03
#
#       EURO              < the format of the time in the output file
#                           should be "YYYYMMDD hhmmss". For example,
#                           20050123 110103
#
#       EURO2             < the format of the time in the output file
#                           should be "DD/MON/YYYY hh:mm:ss". For example,
#                           03/Jan/2005 11:01:03
#
#       EURO3             < the format of the time in the output file
#                           should be "hhmmss". For example,
#                           110103
#
#       RELATIVE          < the format of the time in the output file
#                           will be a floating-point number relative to
#                           the start time of the test. >
#
#   @WRITE_HEADER
#       Always Never None_on_Append Enable_Transition_True
#
#                         < Always - will write headers on startup
#                                    This is the default
#
#                           Never  - Don't write headers on startup
#
#                           None_on_Append - Don't write headers when
#                                    appending to file
#
#                           Enable_Transition_True - Write headers
#                                    when the enable variable goes true
#
#                           All entries must be on the same line.
#                           They can be in any order and are case
#                           insensitive>
#
#   @DATA_OPTIONS
#       hogger fastest    < floger - one time reference data for all
#                           variables. This is the default.
#
#                           hogger - each variable has its own time data
#                           as well as the variable value column. Reference
#                           time data is still supplied.
#
#                           fastest - [optional] this uses the variable
logged
#                           with the most data points as the reference time
#                           data.
#
#                           slowest [optional] - this uses the variable
logged
#                           with the least data points as the reference time
#                           data. >
#
```

```
#    @DATA_FORMAT
#        CSV               < CSV - The data format will be comma separated
#                            values. This is the default.
#
#                            COMMA - same as CSV
#
#                            TSV - The data format will be tab separated
#                            values.
#
#                            TAB - same as TSV
#
#                            COLON - The data format will be colon separated
#                            values.
#
#                            SEMI - The data format will be semicolon
#                            separated
#                            values.
#
#                            SPACE - The data format will be space separated
#                            values.
#
#                            PIPE - The data format will be pipe separated
#                            values.
#
#                            JSON - The data format will be JSON.
#                                   * NOTE: This is not implemented yet.
#
#                            XML - The data format will be XML.
#                                  * NOTE: This is not implemented yet. >
#
#    @INCLUDE_FILTER
#        variable_name_1
#        variable_name_2
#        variable_name_3   < The names of variables that will be
#                            included in the report. If this list is
#                            empty, all variables will be included. >
#
#    @EXCLUDE_FILTER
#        variable_name_7
#        variable_name_8
#        variable_name_9   < The names of variables that will not be
#                            included in the report. This takes precedence
#                            over @INCLUDE_FILTER list. >
#   @CSAR_LABEL
#      csar              <Use the CSAR variable label in the header
#                         of the data output file.>
#
#      prefix            <Use the CSAR variable label including the ECM
#                         prefix on the label in the header of the data
#                         output file.>
#
#      ext               <Use the CSAR external label in the header
#                         of the data output file.>#
```

```
###############################################################################
#
#------------------------------------------------------------------------------
#   NOTE : no FTP stuff is implemented yet, or maybe ever.
#
#   @FTP_HOST
#       hostname          < the name of the remote destination that
#                           receive a copy of the logged data
#                           file. Default value = ctcxssetdl1.>
#
#   @FTP_ACCOUNT
#       account_name      < the account name to be logged into on
#                           hostname. Default value = datasend.>
#
#   @FTP_PASSWORD
#       password          < the password for account 'account_name'
#                           Default value = whatever is the default
#                           value for account datasend.>
#
#   @FTP_DIRECTORY
#       remote_dir        < the directory path be appended to the
#                           @FTP_PATH entry. NOTE: This entry is
#                           required to activate FTP transfers.
#                           There is no default>
#
#   @FTP_PATH
#       root_dir          < the directory on the remote destination
#                           that will receive the data file.
#                           Default value = /group/mech_dev/. This
#                           is prepended to the @FTP_DIRECTORY entry.>
#
#   @FTP_EVENT
#       ftp_it            < the name of the event that will be set
#                           to cause the logged data file to be
#                           ftp'd to '@FTP_HOST'. The event will be
#                           created if it doesn't exist.
#                           Default value = FTP_write.>
#------------------------------------------------------------------------------
#
###############################################################################
# Example of a spec file
###############################################################################

@DESCRIPTION
     My appended description

@INPUT_FILENAME
    /data/PC_format/test_hogger_data-D20230811T072824.db

@OUTPUT_FILENAME
    /data/PC_format/test.csv

@WRITE_HEADER
    none_on_append

@LOG_DIGITAL_DESCRIPTION
```

```
    yes

@OUTPUT_PATH
    /data/PC_format

@PACKED
    Yes

@OVERWRITE_OUTPUT
    no

@WRITE_EXTERNAL_HEADER

@TIME_FORMAT
    HI_RES

@DATA_OPTIONS
    floger fastest

@DATA_FORMAT
    CSV

@INCLUDE_FILTER
    rc_volts
    rc_daq
```

3.3 `hogger_out` Application Behavior Tips

3.3.1 CyFlex Libraries

The `hogger_out` application requires the use of Cyflex libraries to run and thus must be used on a system with Cyflex installed.

3.3.2 Output Format

The output format is a char delimited file. Though there are standard delimiters that can be chosen, any non-white space UTF-8 character can be selected on the command line.

3.3.3 Data Options

Two data options dictate the output format style.

3.3.3.1 *`floger`* *Style Data Format*

If no specification file is given, a `floger` style is produced with a one-time array that is associated with all the variable values.

3.3.3.2 `hogger` *Style Data Format*

The `hogger` style has a time array for all values, but each value will also have its own time array. This allows data to be accurately compared to the time reference. The time data that appears with each variable is the update time of that variable that is closest to the time reference without passing the time reference plus half the time to the next time reference.

By default, the time reference is the scan rate of the `hogger` application. By using the *fastest* and *slowest* modifiers, the time reference can be changed to use the variable with the highest count or lowest count in the database.