GBB CAN Monitor

This device monitors the GBB CAN Bus and collects CAN traffic. It is designed to be run without user intervention. It does provide a web based interface through run time options are setting, the current status of the capture is monitored and collected data downloaded.

Hardware

1. Connect the USB power as shown in Figure 1

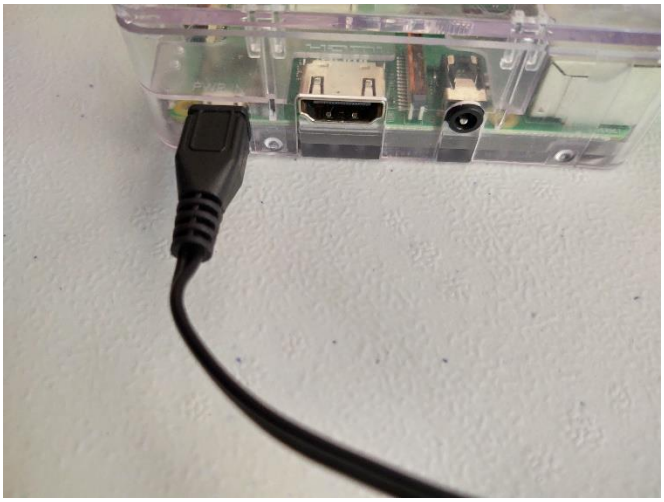

Figure 1

2. Connect the CAN USB Device cable as shown in Figure 2. Any of the available USB ports may be used. Note that the Ethernet cable is optional

Figure 2

3. The fully assembled unit appears as shown in Figure 3. Again, the Ethernet cable is optional. For the unit to run properly, the assembly must be completed to the point show below. The unit does not need to have the wiring to the CAN bus to work correctly but it must have the CAN adapter device (the blue device) connected to the CAN Monitor device using the supplied USB cable.

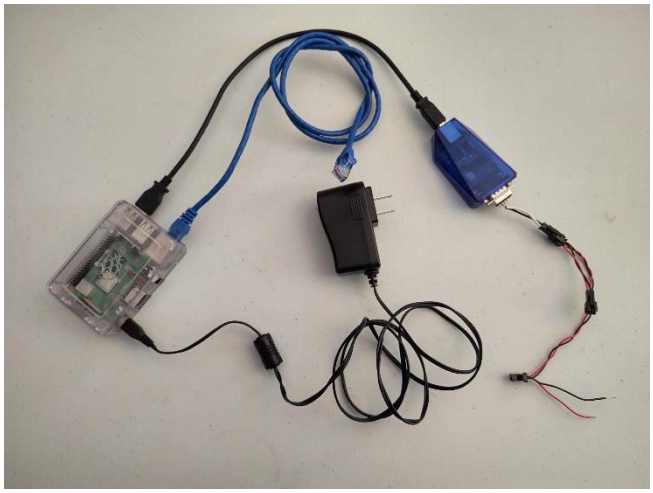

Figure 3

4. Attach the connectors on the CANUSB device to the GBB CAN Bus.

Once the unit is powered up and attached to the CAN Bus, it will begin to collect and store CAN data.

GBB CAN USB Monitor WWW Interface

The CAN Monitor has a www site which provides the user a simple means of communicating with the unit.

The unit has a fixed IP address of 192.168.100.111. To access this site

1. Connect an Ethernet cable to the CAN Monitor as shown in Figure 2.
2. Connect the Ethernet cable to the Ethernet port on a PC.
3. Set the fixed IP address of the port into which the cable is connected to an address that is on the same network as the unit. For example 192.168.100.112 may be used.
4. Open a web browser on the PC and connect to the unit using the URL

   192.168.100.111:8001

5. The web page shown in Figure 4 will be displayed.

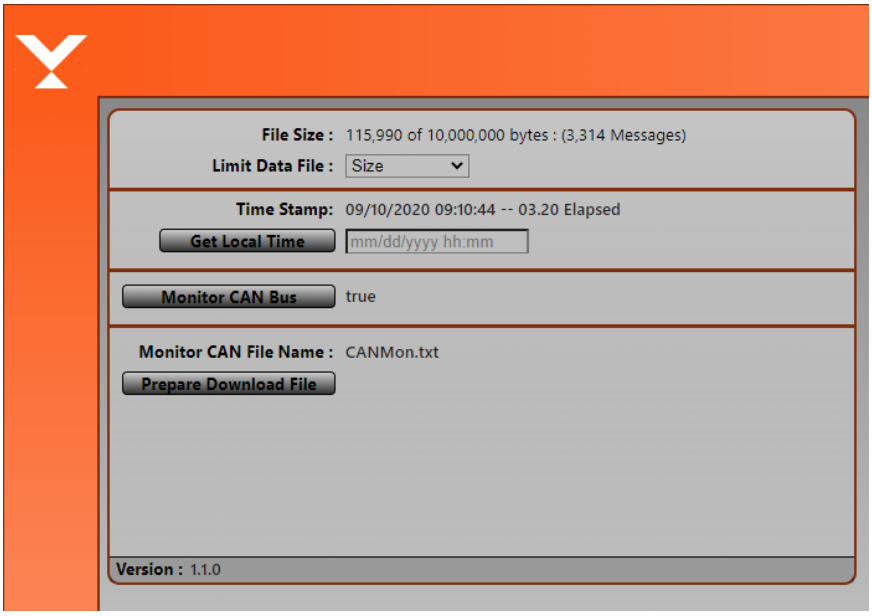

Figure 4

While messages are being collected, the top line field is updated to reflect the current amount of data monitored and stored. The amount values displayed are dependent upon the file limits settings.

Setting File Limits

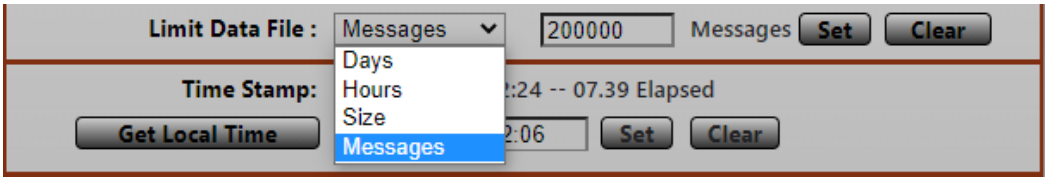

Figure 5

Limits are places on the size of the data files to prevent file sizes from exceed the amount of disk space available on the unit. The imposed limits can be terms of time (how long in hours or days the unit runs), number of messages (the maximum number of messages the unit collects in a single session) or maximum size (in increments of 10 megabytes) to which a file grows before the unit stops collecting data.

The first line of the section displays the size of the collected data in terms of the type of limit (time, message count, file size) and the maximum size of the limit. Regardless of the type of limit selected, the actual number of messages collected is always displayed.

The size limit default values and minimum/maximum values are shown in Table 1.

LIMIT	DEFAULT	MINIMUM/MAXIMUM
Days	7	1/7
Hours	96	1/96
Messages	2,000,000	1/10,000,000
Size	10,000,000	10,000,000/500,000,000

Table 1

The CAN Monitor powers up with a default limit of 10,000,000 MBytes.

The monitor starts collecting CAN data about 30-45 seconds after power up and will continue to do so until powered down or the **Monitor CAN Bus** button is pressed and set to **false**.

Time Stamping

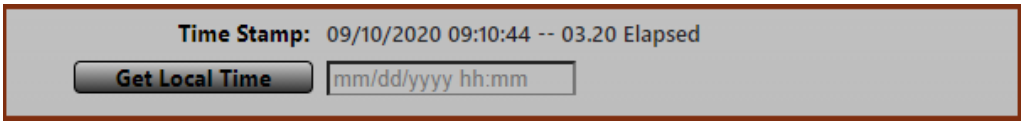

Figure 6

The CAN Monitor hardware has no battery backed real time clock and since in most operating environment will not have public internet access. Therefore, the unit's NTP client has been turned off and the time zone has been set to GMT+0. The unit does keep track of time and when powered up starts its clock from the time it recorded when it was last powered down.

In the absence of user shell access to the unit and the complexities of dealing with time on a non-system managed, ad-hoc basis, a time stamp feature has been added.

This feature allows the user to get the local time from the browser and begin time stamping the transactions as they are read in from the CAN network starting from this point. The first line in Figure 6 shows the current time stamp starting time followed by the elapsed hours:minutes.seconds from the time the unit was powered up or the time stamp was last set.

To set the Time Stamp, click on **Get Local Time.** The www application reads the local time from the PC running the users web browser. This time/date will appear as shown in Figure 7

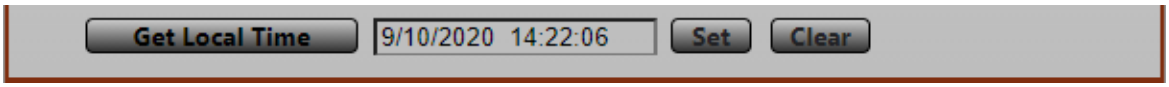

Figure 7

Once the time/date has been retrieved the user may edit it. Click **Set** to set the time stamp. Click **Clear** to cancel the operation and retain the currently displayed time stamp. Once the time stamp has been set, the **Time Stamp** field displays the time/date and the Elapsed time display is reset to show the time elapsed from the displayed **Time Stamp.**

To insure the transactions captured in the data file contain the desired time stamps, it is recommended the time stamp be set before any data is capture by the CAN Monitor.

Monitor CAN Bus option

This option allows for the temporary suspension of data collection. If the option is set to true, the CAN Monitor is actively collecting CAN data. If the option is set to false, the CAN Monitor is not actively collecting data although the session is still active. That is CAN data is still being read it is just not being stored. The time stamp time is still updated and when a session is paused and restarted, the data time stamp will reflect this pause. The session can be resumed by clicking the **Monitor CAN Bus** to set the option to **true**. All data collected before a session is paused is retained

Downloading Files

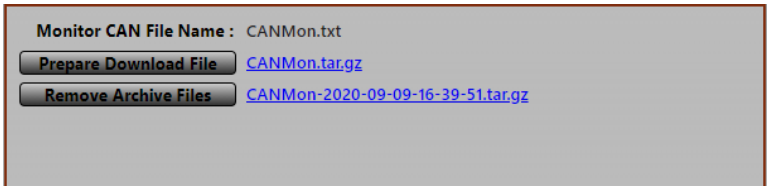

Figure 8

To store the data from one session before starting another the CAN Data collected during a session may be downloaded by pressing the **Prepare Download File** button. Once this button is pressed a link appears to the right of the button as shown in Figure 8.

Click this link to download the current session data. The unit continues to collect data after this file is prepared. Subsequent downloads of the data during a session will include any data collected during the session including any data previous downloaded during the session.

The monitor unit also maintains data files from the most previous 5 monitor sessions. When the **Prepare Download File** button is pressed, download links for the last sessions are presented. The files are dated with the time stamp set when the monitoring session was started, or the time stamp set if it was changed during the session. As with the **CANMon.txt** file, clicking on the archived files download link will download the files. The actual data file is compressed using the Linux tar command and the data file must be extracted from the downloaded file with a tool such as 7-Zip.

If there are archived files available, the **Remove Archive Files** button is displayed. This allows the users to remove all archived files from the unit.

Normal Use

Once started the CAN Monitor can run unattended. It will continue to collect data until either powered, a collection limit has been reached, or the **Monitor CAN Bus** has been set to **false**.

For the best results, it is recommended a monitor session begin as follows:

1. Power up the CAN Monitor before any unit on the CAN bus begins to transmit data or the CAN Monitor has been physically connected to the GBB CAN bus. This is not necessary but insures all data is collected from all units and helps to insure time stamping of data is correct.
2. Connect to the CAN Monitor's www site.

3. Check that the collection limits set are the desired limits.
4. Set the time stamp.
5. Make sure the **Monitor CAN Bus** option is set to true.
6. Power up the devices to be monitored or connect the CAN Monitor to the GBB CAN Bus.

Output File Format

The file created by the CAN Monitor is a simple ascii file where each CAN packet is on a separate line and each line contains 3 space separated files. All fields are in HEX format.

1. 29 bit Frame ID (packed into 4 bytes)
2. 64 bit Data (packed into 8 bytes)
3. The time in seconds when the CAN packet arrived

The data is collected in the raw protocol format and no attempt at any translation is made. See **GBB CAN Monitor Data Viewer** for data translation.

GBB CAN Monitor Data Viewer

As noted above the data collected is stored in a compact HEX format. The Data Viewer utility reads a data file and creates a somewhat more human readable version of the data.

The viewer attempts to parse the data file into the fields described in the GBB CAN protocol specification documents and store the information in a simple text file

The data viewer is a command line program and contains no GUI. It must be run from within a Windows CMD window.

Installation

The data viewer installation file is a simple zip file containing all necessary files required to run this utility. It contains

- canmonview.exe
- DeviceDefs.json
- libwinpthread-1.dll
- GBB CAN Monitor Read Me.pdf

The dll file is required to run the utility. DeviceDefs.json is a file containing the protocol definitions used by the utility to parse the CAN fields in the input file.

The zip file should be extracted to a single directory.

Usage

The data viewer is run from command line and accepts several command line options.

```
C:> canmonview

Usage: canmonview options

      -d, --deffile filename  : Specify a protocol definition file (default DeviceDefs.json)

      -f, --file filename     : Specify a data file to parse (default CANMon.txt)
```

```
        -h, --help              : Display this message
```

To run canmonview with default settings, the file CANMon.txt must be in the same directory as DeviceDefs.txt file and both must exist in the same directory in which the canmonview command is executed. These restrictions can be overridden via the -f and -d options. The -f option is needed if the file to be parsed is not CANMon.txt such as a download archive file.

For simplicities sake and since the application is small, it is recommended the installation and runtime operation be done in the same directory.

Running the data viewer will produce an output file whose name is based on the original input file name. For example, using the default input file name CANMon.txt the output file will be named CANMonParsed.txt. Archived files download and processed are named similarly.

Below is an example of the output of the data viewer.

	DATE	PROTOCOL	SRC	DST	PTP	ERR	MSG	DESCRIPTION	VALUE
4	09/09/2020 12:39:57	REC	NCU	BRD	0	F0	3	Start Interval[002a]	0.000
5	09/09/2020 12:39:57	REC	NCU	BRD	0	F0	3	Restart Time[0028]	300.000
6	09/09/2020 12:39:57	REC	NCU	BRD	0	F0	3	Walk-in Time[0029]	800.000
7	09/09/2020 12:39:57	REC	NCU	BRD	0	F0	3	Walk-in[0032]	0.000
8	09/09/2020 12:39:57	REC	NCU	BRD	0	F0	3	[002f]	00010000

Where the fields are：

DATE	The date of the transaction based upon the time stamp set on the www page.
PROTOCOL	The specific device protocol used by this packet. If the protocol is not one of those defined in DeviceDefs.json, the numeric protocol is displayed in HEX.
SRC	The Source destination of the packet. If the packet is from the NCU the label **NCU** is displayed, otherwise the address of the device is displayed in HEX.
DST	The destination address to which the packet is addressed. If the destination is the NCU the label reads **NCU**. If this is a broadcast message the label reads **BRD** otherwise the address is displayed in HEX.
ERR	The packet error code.
MSG	The packet message type
DESCRIPTION	A text version of the message. If the message is not recognized the description number is displayed in HEX
VALUE	The packet's value. If defined in the DeviceDefs.json file the data is displayed in it's native format such as a floating point number or a ASCII text field otherwise the value is displayed in HEX format.

See the GBB CAN Protocol documents for more complete descriptions of these fields

Fields for which a protocol definition is not found in DeviceDefs.json and cannot be translated are displayed in HEX format.

DeviceDefs.json may be updated to include missing protocols. However, it is strongly recommended that the casual user not do this. While the file structure is a simple JSON format the definitions can get lengthy and as such errors are easy to make which will cause the file to be unable to be read by the data viewer. This will cause the program to fail and no output will be produced. However, if you

understand the GBB CAN protocols, can understand the JSON format which is not difficult and are feeling somewhat brave or foolhardy, go for it.