vcdMaker

A User's Guide to vcdMaker
Edition 3
November, 2018

Table of Contents

1 Licensing

2 Introduction

Quite often while debugging a computer system huge log files are created and they need to be analyzed afterwards. That could be achieved by parsing the trace and extracting the relevant information but initially one needs to know what the information is. The *vcdMaker* tool set has been created to serve this purpose and help software and firmware engineers to analyze their applications and systems. It translates a human readable log files into the Variable Change Dump format files allowing for their further visualization which can be done using any of the VCD viewers. As a single picture is said to be worth thousand words the approach makes the debugging process easier and more efficient.

The tool set is provided as a freeware. There are no charges for using it but the development team would appreciate your feedback. All of the comments, requests or screen captures presenting the way you benefited from using it are appreciated. Contact us via *vcdmaker@mail.com*.

3 Installation

3.1 Windows

3.1.1 Installing

Start the installation by executing the *vcdMakerSetup.exe*. Then just follow the installer. During the process you will be able to adjust the installation path. The installer will copy all the necessary files, create the shortcut to the *vcdMaker* manual in your Programs Menu and update the user's environmental variable PATH. Therefore once installed a user will be able to invoke the application from any location.

The setup log could be created by invoking the installer in the following way:

```
vcdMakerSetup.exe /install /log vcdMakerInstallation.log
```

3.1.2 Uninstalling

The *vcdMaker* package can be easily and entirely removed by the Programs and Features Control Panel tool. Alternatively one could use the following command:

```
vcdMakerSetup.exe /uninstall
```

3.2 Linux

There are binary packages available for the Linux distributions using the RPM and the DPKG package managers. The packages contain the applications themselves, the manual pages, the user documentation as well as sample files. By default, they will be installed in the '/usr/share/doc/vcdMaker' folder.

3.2.1 Installing

RPM

```
rpm -i package_name.rpm
```

DPKG

```
dpkg -i package_name.deb
```

3.2.2 Uninstalling

RPM

```
rpm -e vcdmaker
```

DPKG

```
dpkg -r vcdmaker
```

4 Getting Started with vcdMaker

Running *vcdMaker* is easy but to make the learning curve really steep there have been provided sample files one could use to get the knack of the tool.

In the target installation folder there is a *sample* sub-folder containing the *example.txt* file (for Windows, for Linux see 3.2). It composes of an arbitrary set of signals and their values changing over time. The format of the input file has been described in 5. Copy the file to any folder you wish to experiment in.

Then, use the following command to create the output VCD file:

```
vcdMaker -t us -o output.vcd example.txt
```

The *vcdMaker* will parse the *example.txt* file and create the *output.vcd* file. That's it!

So as to get the feeling of how you could use the VCD file it is recommended to read the Chapter 8 first.

Using *vcdMerge* is equally easy! There are sample files provided as well. Let's merge *system1_time1.txt* and *system2_time2.txt*. The logs represent systems working in different time domains (they are not synchronized).

The synchronization points in both logs have been marked with the 'Sync' events. Browse the logs and search for them. Their time stamps are the parameters of the signals sources.

The syntax is:

```
vcdMerge.exe -o merge_t1_t2.vcd
   T,71050601,us,System1,Counter,system1_time1.txt
   T,234256037,us,System2,Counter,system2_time2.txt
```

For the synchronized sources (*system1_time1.txt* and *system2_time1.txt*) the time stamps shall be set to 0:

```
vcdMerge.exe -o merge_t1_t1.vcd
   T,0,us,System1,Counter,system1_time1.txt
   T,0,us,System2,Counter,system2_time1.txt
```

5 Input log files

vcdMaker tools can parse any user provided log. The syntax of the log can be specified by the XML file which has been described in 5.1. Alternatively the logs to be processed might be created in the inherent *vcdMaker* format presented in 5.2.

5.1 Specifying the user log format

5.1.1 The XML structure

The user log format can be specified by the XML file with the following structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE signals SYSTEM "vcdMaker.dtd">
<signals>

<vector>
    <line> Regular expression describing the line including group matching </line>
    <timestamp> Timestamp expression </timestamp>
    <name> Name expression </name>
    <value> Integer value expression </value>
    <size> Size expression </size>
</vector>

<real>
    <line> Regular expression describing the line including group matching </line>
    <timestamp> Timestamp expression </timestamp>
    <name> Name expression </name>
    <value> Float value expression </value>
</real>

<event>
    <line> Regular expression describing the line including group matching </line>
    <timestamp> Timestamp expression </timestamp>
    <name> Name expression </name>
</event>

…
</signals>

</log_configuration >
```

There might exist several vector, real or event nodes. According to the user needs. The syntax of the XML file can be verified with the provided DTD file (samples/vcdMaker.dtd in the target installation directory):

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT signals   (vector | real | event)*>

<!ELEMENT vector    (   line,
                        timestamp,
                        name,
                        value,
                        size)
>
<!ELEMENT real      (   line,
                        timestamp,
                        name,
                        value)
>
<!ELEMENT event     (   line,
                        timestamp,
                        name)
>
<!ELEMENT line (#PCDATA)>
<!ELEMENT timestamp (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT value (#PCDATA)>
<!ELEMENT size (#PCDATA)>
```

For example, to parse the inherent *vcdMaker* format one could use the configuration file below:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE signals SYSTEM "vcdMaker.dtd">
<signals>

<vector>
    <line>#([[:d:]]+) ([[:graph:]]+) ([[:d:]]+) ([[:d:]]+)( +.*)?</line>
    <timestamp>dec(1)</timestamp>
    <name>txt(2)</name>
    <value>dec(3)</value>
    <size>dec(4)</size>
</vector>

<real>
    <line>#([[:d:]]+) ([[:graph:]]+) ([[:d:][:punct:]]+) f( +.*)?</line>
    <timestamp>dec(1)</timestamp>
    <name>txt(2)</name>
    <value>flt(3)</value>
</real>

<event>
    <line>#([[:d:]]+) ([[:graph:]]+) e( +.*)?</line>
    <timestamp>dec(1)</timestamp>
    <name>txt(2)</name>
</event>

</signals >
```

5.1.2 Signal line specification

To construct the signal correctly one needs some information describing it. The <*line*> element of the XML helps defining the data useful to create it from the parsed log line. It is a regular expression containing groups which can be referred to later. See the example above.

```
Info! At least, a log line shall contain its name and value.
```

5.1.3 Expression functions

There are several functions which can be used to construct signal properties and they have been listed below:

- dec(group_number)

  It returns the integer representation of the string provided by the regex capture group. It must be a valid decimal string, e.g. '153'.

- hex(group_number)

  It returns the integer representation of the string provided by the regex capture group. It must be a valid hexadecimal string, e.g. 'A78'.

- line()

  It returns the number of the parsed log line. Hint: it allows for parsing logs not containing timestamps at all. It could be useful in some circumstances.

- flt(group_number)

  It returns the float representation of the string provided by the regex capture group. It must be a valid float string, e.g. '3.14159'.

- txt(group_number)

It returns the string captured by the regex group. It must be a valid string, e.g. 'Sensor_1'.

group_number – decimal integer in the range from 1 to the last index of the given regex group

5.1.4 Integer expressions

The integer expression is a combination of arithmetic operators and expression functions. The precedence of operators is traditional. Available operands: '*', '/', '+', '-', '()'.

Example

10 * (dec(1) + dec(3)) + hex(2)

The value of the expression can be easily calculated applying simple arithmetics.

If dec(1) equals 2, hex(2) equals 0xB and dec(3) equals 5 then the calculated output value is 81.

5.1.5 Float expressions

The float expression is a combination of arithmetic operators and the flt() expression function which is the only accepted function in this case. The precedence of operators is traditional. Available operands: '*', '/', '+', '-', '()'.

Example

2.5 * (flt(1) + flt(2)) – 0.9

Again, the value of the expression is based on a simple arithmetic evaluation.

If flt(1) equals 2.5, flt(2) equals 1.1 then the calculated output value is 8.1.

5.1.6 String expressions

It is a combination of strings (in double quotation marks) and regex groups. Available operands: '+'.

Example

"SomeName." + txt(1) + "OtherName" + txt(2) + "1"

So, in the final string 'txt(1)' will be replaced by the first regex capture group while 'txt(2)' will be replaced by the second regex capture group.

If txt(1) is "XYZ" while txt(2) is ".ABC" the output string will be:

"SomeName.XYZOtherName.ABC1"

5.1.7 Timestamp expression

This must be a valid integer expression. The timestamps do not have appear in the ascending order. Signals will be properly sorted by a tool. The line() function returns the processed log line number.

5.1.8 Name expression

This must be a valid string expression. The name of the signal shall follow the guideline presented in the chapter 5.1.11.

5.1.9 Value expression

For a vector signal this must be a valid integer expression. The line() function returns always 0.

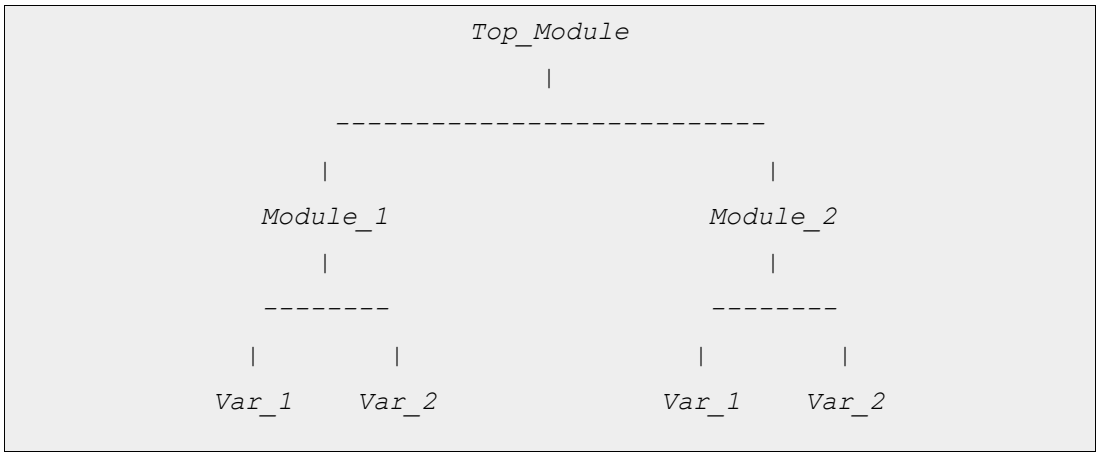
For a real signal this must be a valid float expression.

5.1.10 Size expression

This must be a valid integer expression. The line() function returns always 0.

5.1.11 Signal naming

This is a structure describing the hierarchy for accessing the signal or the variable. Each level of the structure must be separated by the '.' character. The hierarchy could be illustrated by the tree. For instance:



In this example there are four variables in the system above and they shall be defined as follows:

```
Top_Module.Module_1.Var_1
Top_Module.Module_1.Var_2
Top_Module.Module_2.Var_1
Top_Module.Module_2.Var_2
```

The nesting of the signal name is not limited to three levels. A user can use whatever nesting level is required.

5.1.12 Example

The provided sample files (user1.xml and user2.xml) can be used as a reference.

```
vcdMaker -t us -o output.vcd -u user1.xml example_user.txt
```

```
vcdMerge.exe -o merge_t1_t2.vcd
  U{user2.xml},71050601,us,System1,Counter,system1_time1_use
  r.txt T,234256037,us,System2,Counter,system2_time2.txt
```

5.2 The inherent *vcdMaker* log format

Let's present the inherent format of the input log file by an example below.

```
...
#77655698 FRDM.Sensors.Magnetometer.MagY 873 32
#77655817 FRDM.Sensors.Magnetometer.MagZ 3126 32
#77756251 FRDM.Buttons.SW1 1 1 Key pressed
#77805674 FRDM.Sensors.Magnetometer.MagX 64955 32
#77805797 FRDM.Sensors.Magnetometer.MagY 853 32
#77805916 FRDM.Sensors.Magnetometer.MagZ 3191 32
#77806061 FRDM.Sensors.Slider 99 7
#77806070 FRDM.Sync e
#77808473 FRDM.Sensors.Accelerometer.AccX 0.538574 f
#77808473 FRDM.Sensors.Accelerometer.AccY 0.172852 f
#77808473 FRDM.Sensors.Accelerometer.AccZ 0.706543 f
...
```

In general a log signal syntax is defined as follows:

```
#Timestamp Signal_name Value <Size | f | e> Optional_comment
```

There are common properties of the log signals:

- **Timestamp** – The system time stamp. The time unit is provided later during the creation of the VCD file. **The entries in the log file do not have to be ordered. In other words, the entries with lower time stamp values might follow the entries with higher time stamp values.** The events will be reordered properly during the VCD file creation. This might be useful when there are a few loggers in your system accessing the same logging device (e.g. UART). They all might log at their own pace and flush their buffers randomly.
- **Signal name**

  The naming convention has been described in the chapter 5.1.11.

There are also properties specific to different signal types:

- integer values

  ```
  #Timestamp TopModule...SubModule.VarName Value Size Comment
  E.g.
  #77655698 FRDM.Sensors.Magnetometer.MagY 873 32
  ```

  Value – The value of the variable.

  Size - The *size* parameter represents the number of bits needed to represent the value. Currently the maximum size shall not exceed 64 (any integer between larger than 0 and

lesser than 65).

- real values

```
#Timestamp TopModule...SubModule.VarName Value f Comment
E.g.
#77808473 FRDM.Sensors.Accelerometer.AccZ 0.706543 f
```

Value - Real values shall be expressed in the '`%f printf()`' format.

- events

```
#Timestamp TopModule...SubModule.EventName e Comment
E.g.
#77806070 FRDM.Sync e
```

6 Creating VCD files

To produce the VCD file from a log file use the following command:

```
vcdMaker -t timeUnit [-c LineCounterName] [-u UserLogFormat] [-
v] -o filename.vcd logFilename
```

filename – The target VCD file name.

timeUnit – The time unit in which the samples have been recorded. Available time units are: *s*, *ms*, *us*, *ns*, *ps*, *fs*.

-c LineCounterName – the switch is optional and it allows for adding a signal containing the log line number referencing to any signal change occurring at a given time. By following the line number one can easily identify the interesting place in the log. The LineCounterName name shall follow the standard signal format. If no top module name is provided *Top* will be assumed. For example, if you name it just 'Counter' it will appear in the output VCD file as 'Top.Counter' as every variable needs its rooting.

-u UserLogFormat – the switch is optional and it allows for providing the XML file describing the format of the log provided. See details in 5.1. If not provided, the syntax presented in 5.2 is expected.

v – the switch is optional while it enables the verbose mode in which the application will print out all of the log lines not matching the expected format. It is recommended to use this mode after the initial *vcdMaker* execution returns a positive number of invalid lines.

logFilename – The name of the log file to be processed.

Examples:

Basic syntax:

vcdMaker -t us -o out.vcd example.txt

User defined log format:

vcdMaker -t us -o out.vcd -u vcdMaker.xml example.txt

Adding the line counter:

vcdMaker -t us -o out.vcd -c Counter example.txt

7 Merging log files

There is also a possibility to create a single output VCD file out of several input log files. The feature might be useful while analyzing behaviors within distributed systems. The syntax of the *vcdMerge* command is as follows:

```
vcdMerge [-t timeUnit] [-v] -o filename sources
```

filename – The target VCD file name.

timeUnit – The time unit in which the output VCD file will be produced. Available time units are: *s*, *ms*, *us*, *ns*, *ps*, *fs*. If the parameter is not provided then the finest time base among the sources is selected. E.g. if one of the sources provides a log in 'ms' while another one provides a log in 'us' the output file will be in 'us'.

```
Warning! When the time unit is forced and it is less accurate
that the finest unit within the provided sources the time stamps
of some signals in the output VCD file might contain a rounding
error (+/- 1 [time unit]). Usually, it is better to use the
automatically chosen time unit.
```

-v – the switch is optional. It enables the verbose mode in which the application will print out all of the log lines not matching the expected format.

sources – The descriptions of the signals sources.

A user might merge as many sources as needed.

Each source is characterized be a set of parameters separated with ',':

format,*timestamp*,*unit*,*prefix*,*counter*,*filename*

format	The format of the input file. Use 'T' for the inherent *vcdMaker* log format described in 5.2. Use 'U{log_format.xml}' to specify the user log format. See details in 5.1.
timestamp	The synchronization time for the input log. Set to 0 for synchronized logs.
unit	The unit of the time stamp.
prefix	The prefix to be added to all the signals within the unit. It can be left empty. Then, no prefix is added.
counter	The line counter signal name to be added to the output file. It can be left empty. Then, no line counter signal is added.
filename	The name of the file to be merged.

Examples:

All source options used:

```
vcdMerge -o out.vcd -v T,71050601,us,Prefix1,Counter1,log1.txt
                       U{vcdMaker.xml},234256037,us,Prefix2,Counter2,log2.txt
```

No signal counters:

```
vcdMerge -o out.vcd -v T,71050601,us,Prefix1,,log1.txt T,234256037,us,Prefix2,,log2.txt
```

No prefixes nor signal counters:

```
vcdMerge -o out.vcd -v T,71050601,us,,,log1.txt T,234256037,us,,,log2.txt
```

Those two logs contain data from two different targets fitted with accelerometers. The boards were bundled together. When the logs are properly aligned and displayed in the analog form it can be easily spotted that the signals registered are correlated. One can also notice that the Z-axes were flipped.

8 Viewing VCD files

There are many applications on the market allowing for viewing Variable Change Dumb format files and they are especially popular among RTL designers. Although very often such tools are quite expensive there exist a few free alternatives and one of them has been presented later in this manual.

8.1 GtkWave

GtkWave has been chosen for various reasons. It's free, light, easy to use and it has got rich features.

8.1.1 Downloading

The tool can be downloaded from:

http://gtkwave.sourceforge.net/

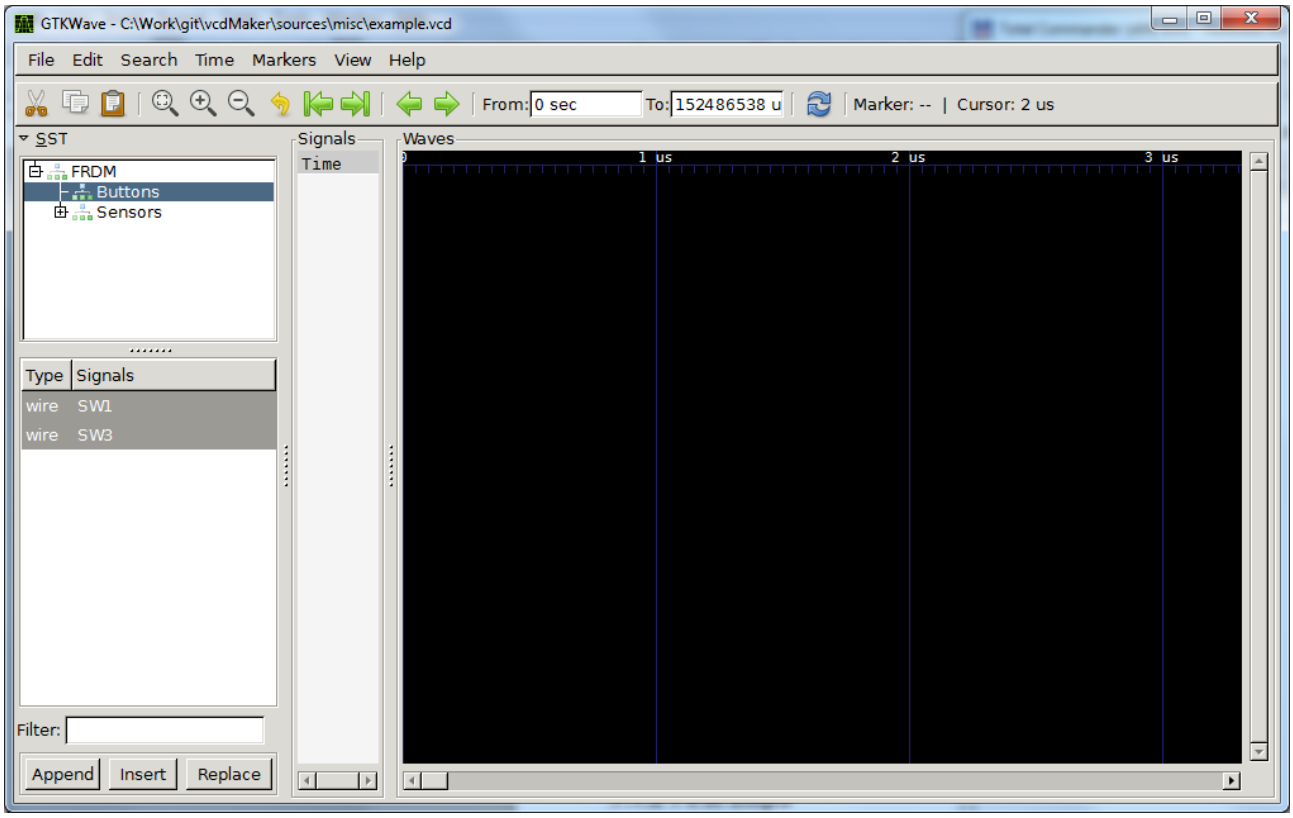
8.1.2 First steps

For the users convenience there has been provided the introduction to the GtkWave on the following pages. The further details related to the tool can be found in the GtkWave documentation itself.

Step 1

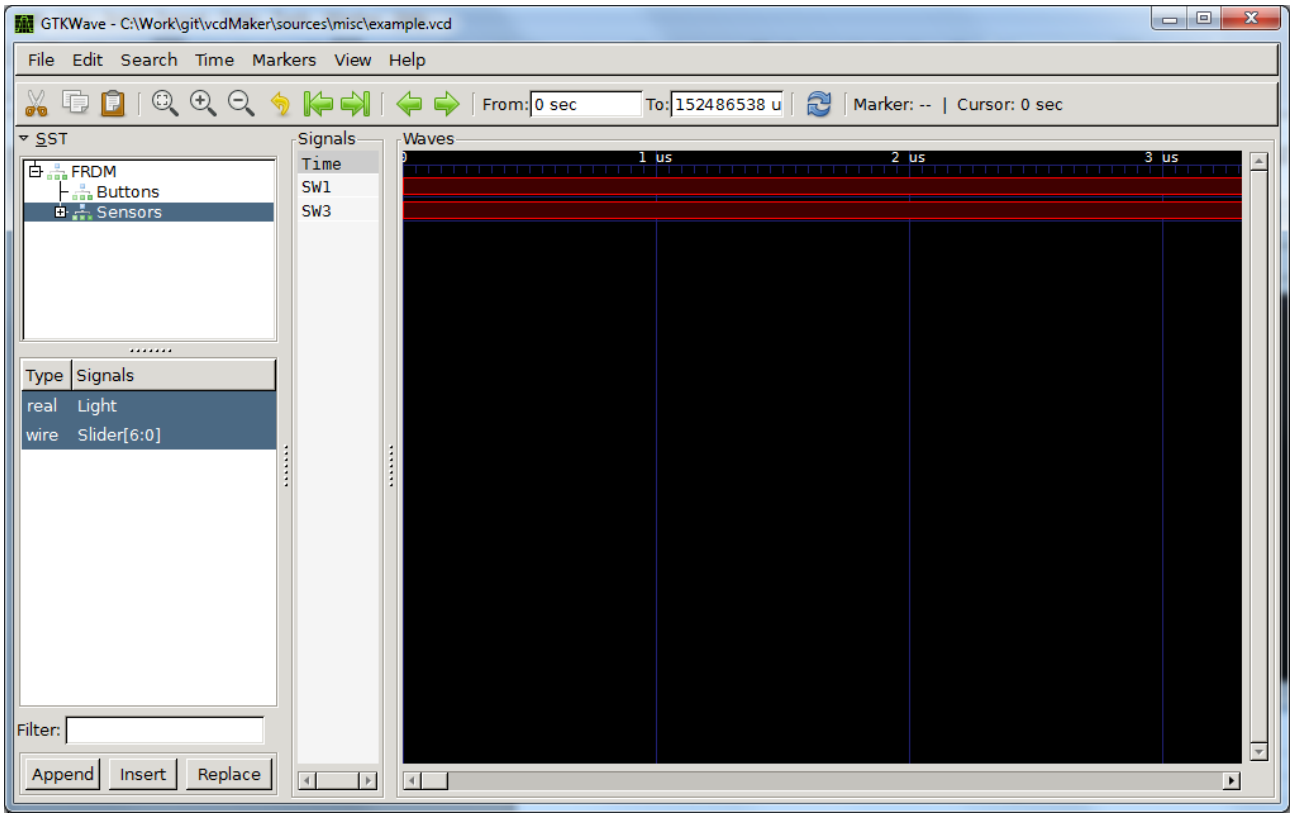
Run the tool. Drag and drop the *examples.vcd* file to the GtkWave.

In the top SST pane expand the FRDM and select Buttons. Keep 'Ctrl' pressed and in the bottom SST pane select SW1 and SW3. Click the *Append* button.



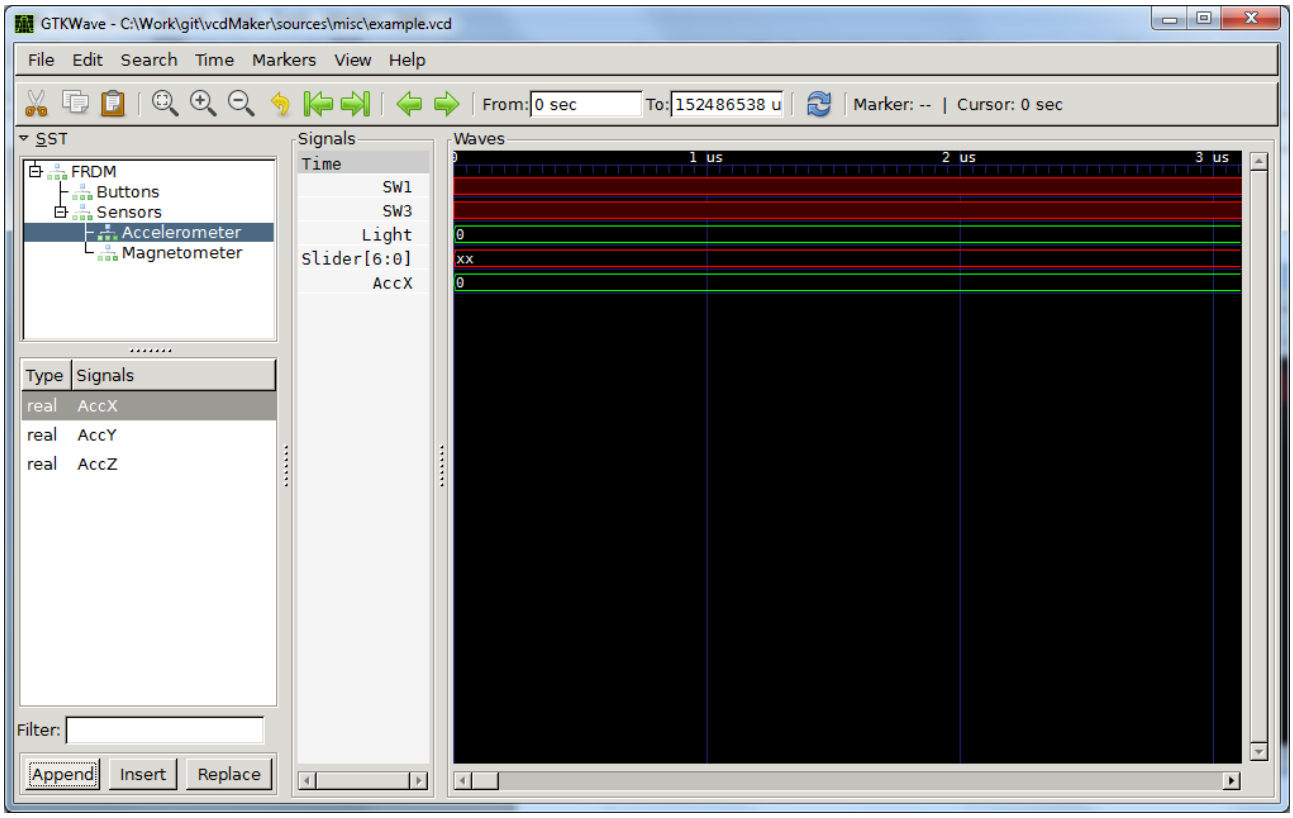
Step 2

In the top SST pane set the focus on Sensors. Keeping 'Ctrl' pressed select Light and Slider from the bottom SST pane. Click the *Append* button.



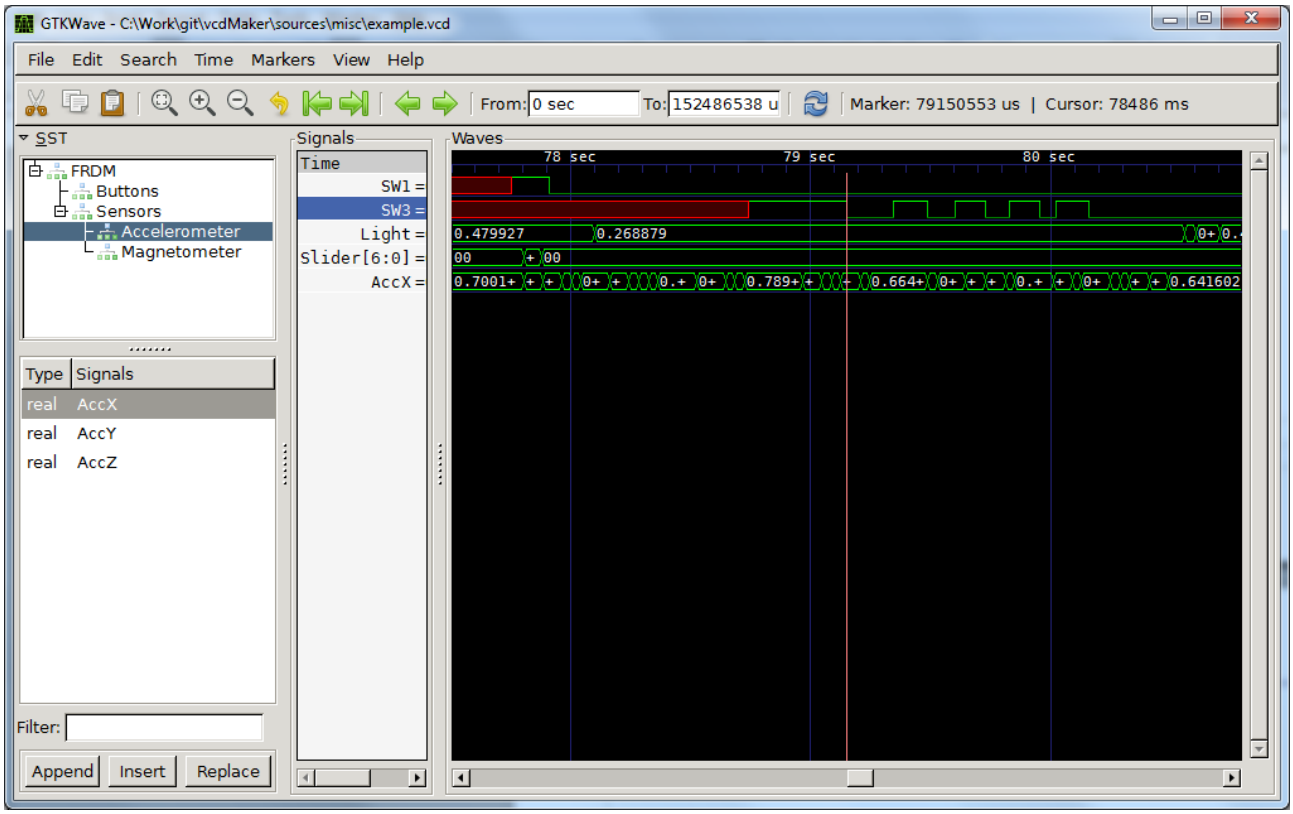
Step 3

In the top SST pane expand Sensors and select Accelerometer. In the bottom SST pane select AccX. Click *Append* again.



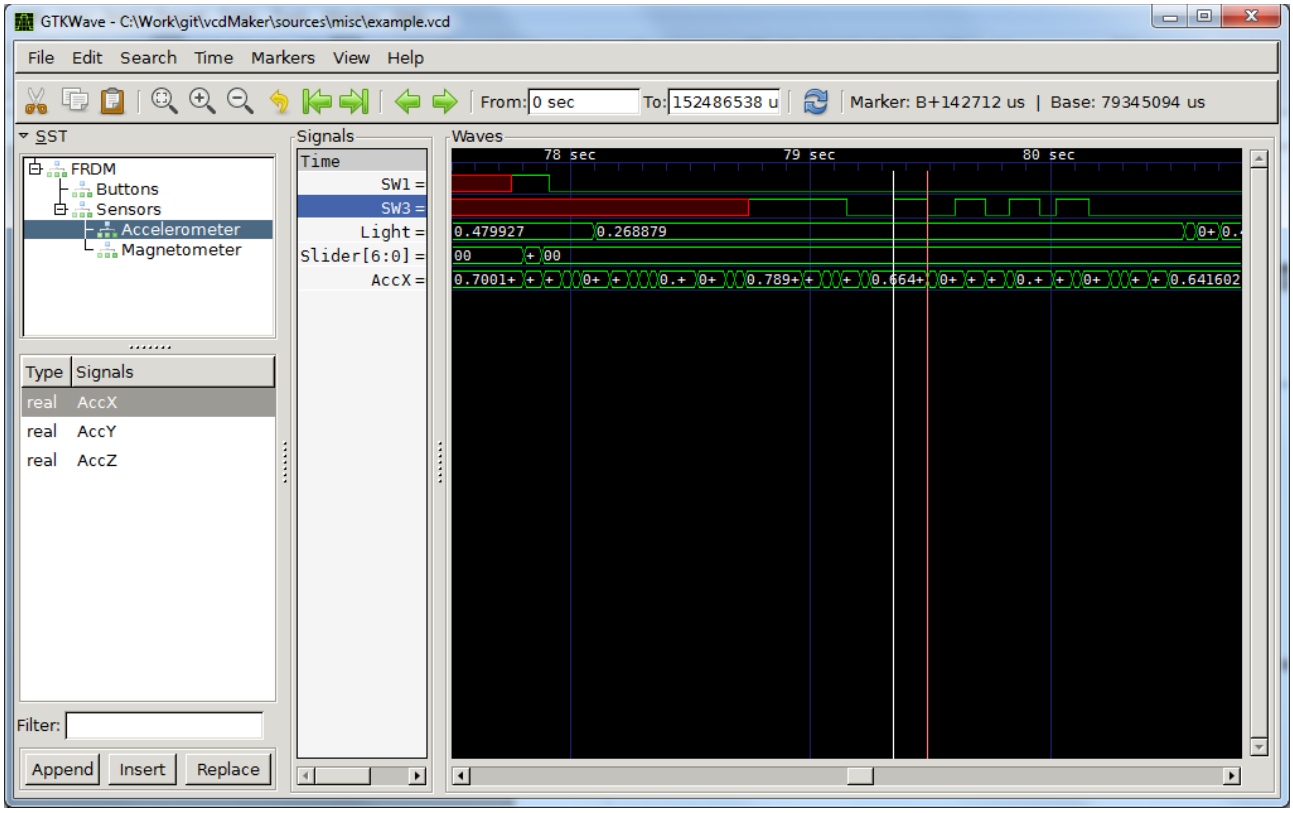
Step 4

In the Signals pane select SW3. Press *Find Next Edge* button (green right arrow) three times. Then, press *Zoom Out* button (zoom minus) several times so as you get a similar view scale as below.



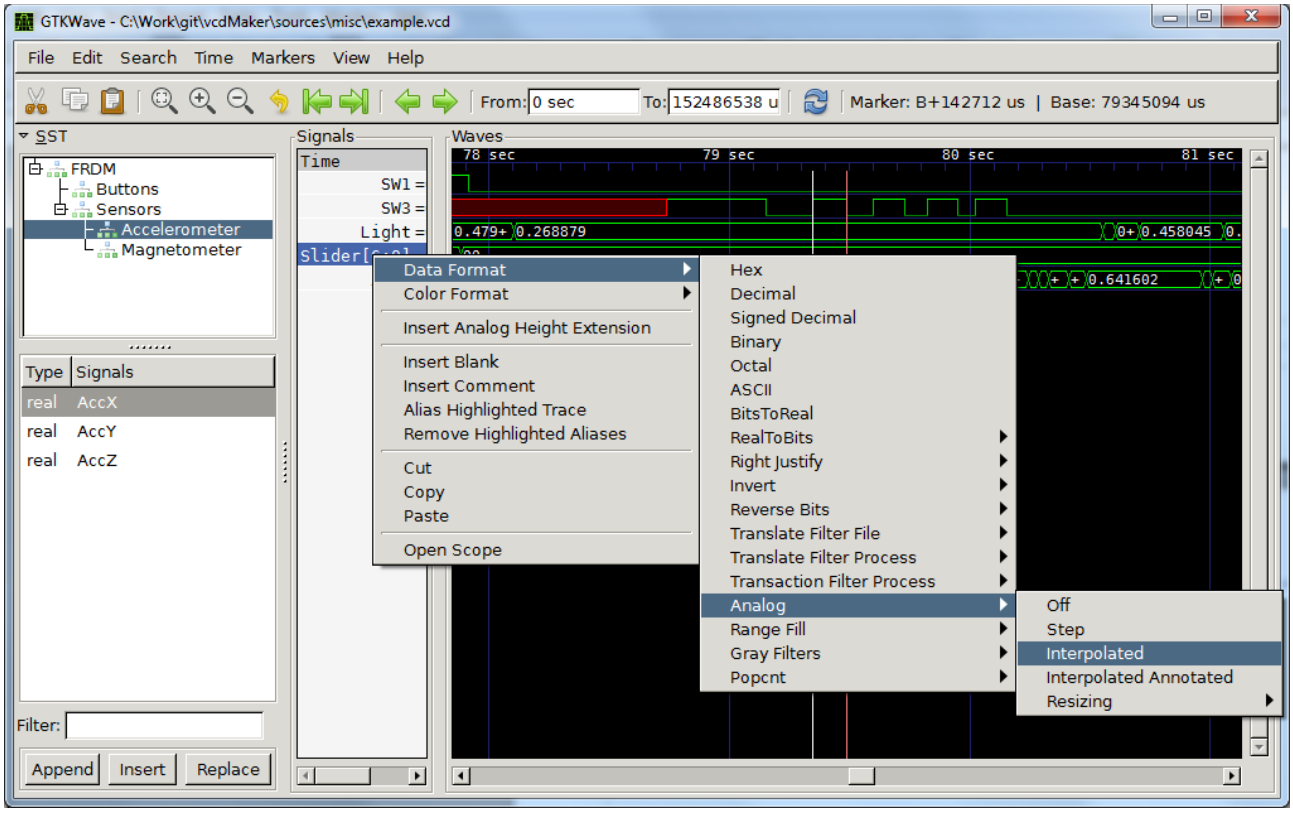
Step 5

Now, click close to the next rising edge of the SW3. Press 'b' key. Then click on the nearest falling SW3 edge. As measured by GtkWave the pulse is 142.712 [ms] width. Congratulations! You just learned how to measure time between two registered events.



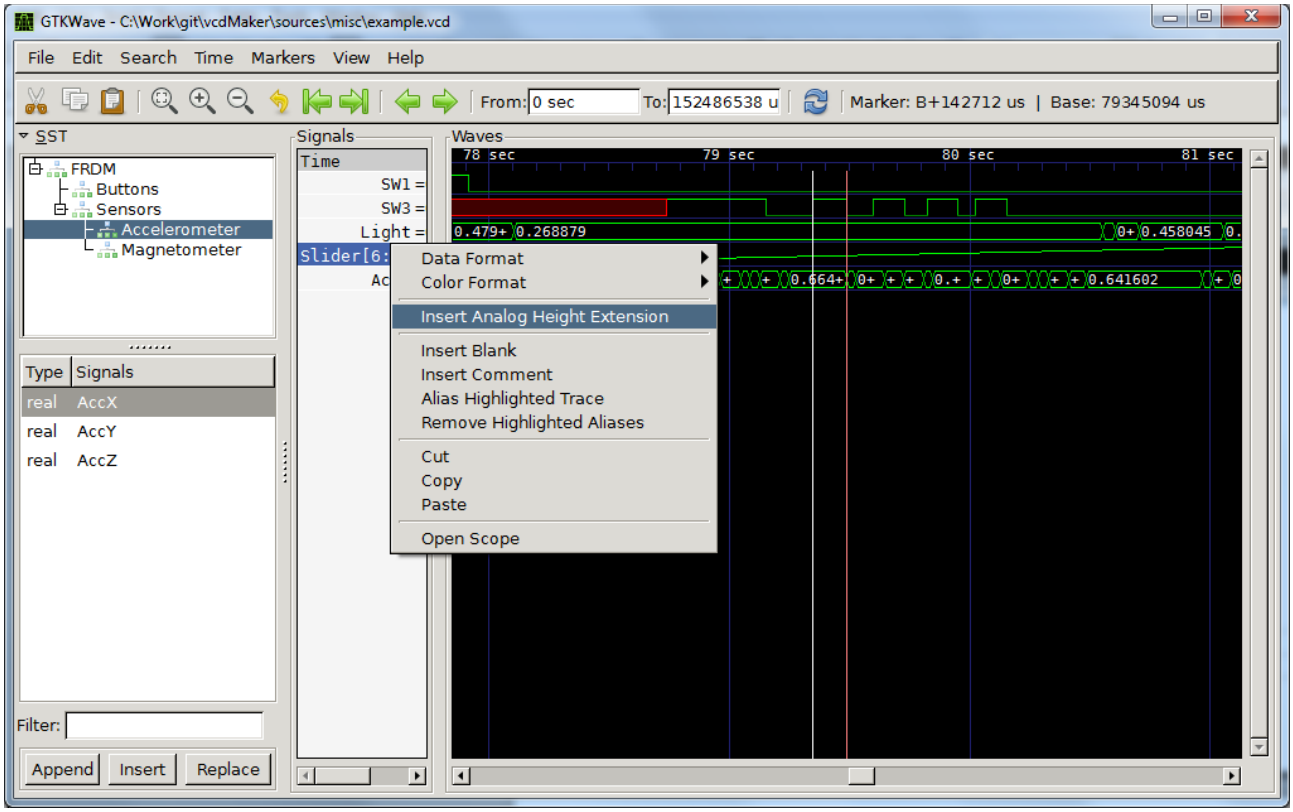
Step 6

In the Signals pane select Slider. Right click on Slider. Select *Data Format -> Analog -> Interpolated*. Now, you can scroll left and right the Waves window to see the changes of the signal over time. Of course, you can zoom it in or out to get the better results.



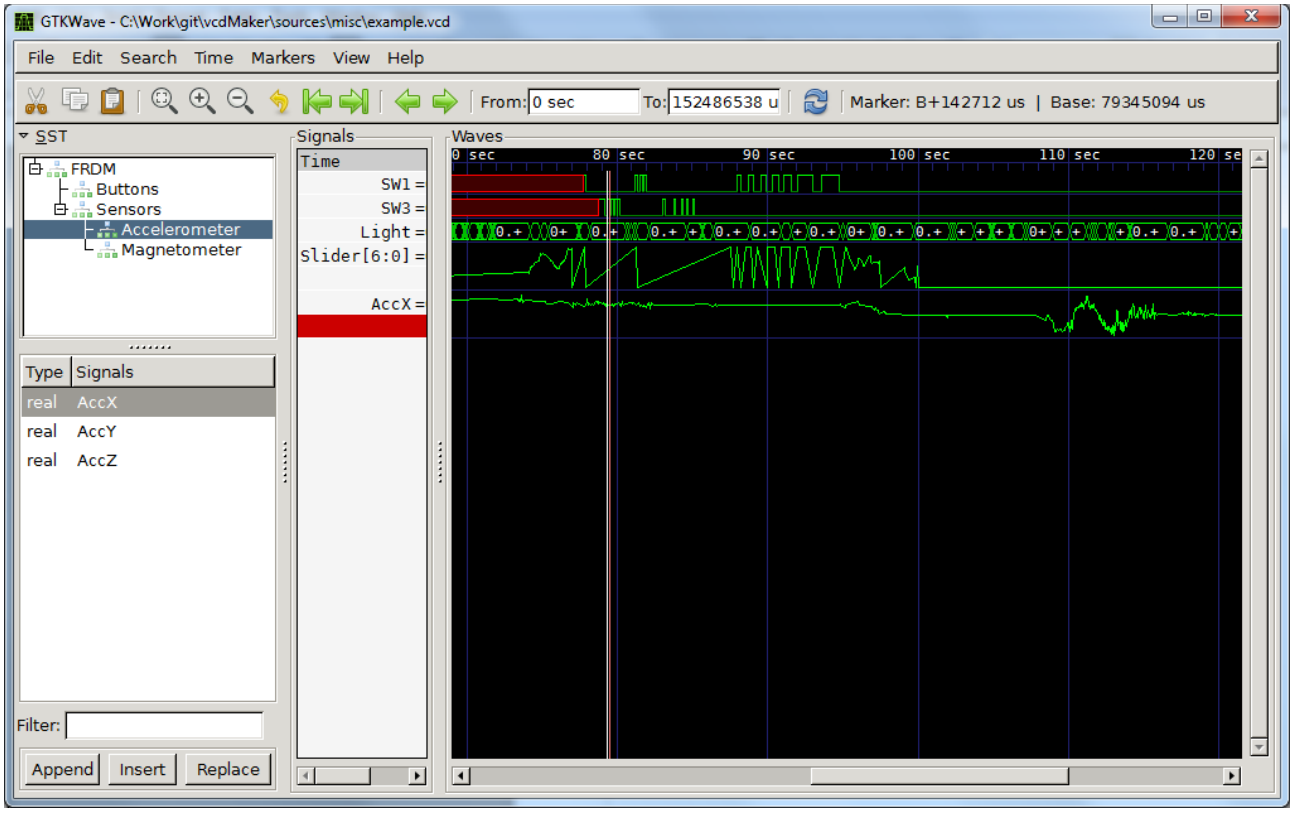
Step 7

In the Signals pane select Slider and right click on it. Select *Insert Analog Height Extension*. It will make the row higher and scaled better for the analog view. You can repeat steps 6 and 7 for AccX as well.



Step 8

Well done! Hopefully, these simple steps let you know the most basic operations of the GtkWave. There are also the most commonly used ones as well. For further information about the tool please refer to its documentation.



9 Warnings and errors

Legend:

Common
vcdMaker
vcdMerge

The following table presents a list of possible warnings.

Warning	String	Description
0001	Synchronization time out of bounds. Cannot merge {source}.	The entire signal source cannot be merged as its new synchronization point timestamp falls out of 64-bit value. Consider changing the output time unit.
0002	Timestamp out of bounds. Cannot merge {signalName} at {timestamp} {timeUnit}.	The signal cannot be merged as its new timestamp falls out of 64-bit value. Consider shortening the log or changing the output time unit.
0003	Evaluating {file_name}. Line {line_number}: {log_line} Expression: {problematic expression} Value {value} exceeds the {size}-bit size of the vector. Dropping the signal.	The log line matching the vector signal type has been encountered but the value of the signal to be created exceeds its declared size. The signal won't be created. The size of the vector shall be increased.

The errors likely to occur has been listed below.

Error	String	Description
0001	Opening file {fileName} failed, it either doesn't exist or is inaccessible.	Missing file. Check if the path and the file name are correct.
0002	Inconsistent signal: {signalName}. Types: {signal1Type} / {signal2Type}. Sizes: {signal1Size} / {signal2Size}. Sources: {signal1Source} and {signal2Source}.	There are inconsistent signals (from the same or different sources). vcdMaker provides also the log line numer at which the inconsistent signal appeared. Make sure you have no two similar signals differing only by their types or sizes. For vcdMerge, consider adding or changing the source specific prefix.
0003	Invalid time unit: {timeUnit}.	The time unit out of spec. Make sure the provided time unit is valid: [s, ms, us, ns, ps, fs].
0004	Synchronization point value out of bounds: {timeStamp}.	The synchronization point falls out of the 64-bit value.

		Consider changing the output time unit.
0005	Invalid log file format: {format}.	The file format out of spec. Currently only 'T' format is suppoerted.
0006	Invalid synchronization point value: {synchronizationPoint}.	The synchronization point value wasn't provided in the valid decimal format. Correct the value of the synchronization point.
0007	Invalid number of source parameters: {sourceDescription}.	Wrong number of source parameters, separated by ','. Correct the source description.
0008	Leading time out of bounds.	Leading source time out of bounds. Consider changing the output time unit.
0009	There are at least two signal sources required.	Too few signal sources. At least two signal sources are required.
0010	XML - No parsing regular expression.	The regular expression describing the user log syntax is missing.
0011	XML - No timestamp expression.	The expression used to create the signal timestamp is missing.
0012	XML - No name expression.	The expression used to create the signal name is missing.
0013	XML - No value expression.	The expression used to create the signal value is missing.
0014	XML - No size expression.	The expression used to create the signal size is missing.
0015	XML - Unexpected tag: {tag_name}	Unexpected tag in the user provided XML file.
0016	Opening XML file {file_name} failed. The XML file might be incorrect.	The XML file could not be read. Its syntax might be wrong.
0017	No signals creators. Hint: Verify the correctness of the XML file specifying the user log format.	Log files could not be parsed as there is no valid signal creatos registered. Most likely the syntax of the provided XML specifying the user log format is wrong.
0018	Parsing error in {file_name}: {Erroenous expression}	User expression parsing error. It might be a wrong expression syntax, an invalid function name or an erroenous function parameter.
0019	Evaluation error in {file_name}. Line {line_number}: {log_line} Expression: {problematic expression} {Error reason}	The error occured while evaluating one of the user expressions. The possible error reasons are: • Division by zero The evaluation of the expression leads to the division by zero. • Cannot convert to decimal The argument of the dec() function is incorrect. • Cannot convert to hex The argument of the hex() function is incorrect. • Cannot convert to double

		The argument of the flt() function is incorrect. • '<empty string>' is returned when one of the conversion functions is fed with an empty string. • Out of range decimal value The value captured by regex is too big to be converted successfully. • Regex group index out of range The referenced group index does not exist. • Overflow while adding The addition operation resulted in the value above the upper limit of the type. • Overflow while multiplying The multiplication operation resulted in the value above the upper limit of the type. • Underflow while substracting The substraction operation resulted in the value below the lower limit.
0020	{Regex library error} Regex: {regular expression}	The regex error. The original message from the library is returned. It might indicate too complicated regular expression. Most likely the regular expression could be simplified and still do its job. It is recommended to test the correctness of the expression before using it. Hint! Use https://regex101.com/.
0021	Vectors sizes greater than 64 bits are not allowed. Requested {size}-bit size.	Currently vector signal size is limitted to 64 bits.

10 Reference projects

There are a few projects which have been created using popular development platforms and which demonstrate the way *vcdMaker* tools could be used.

The demo for the Freescale FRDM-KL46Z kit:

https://developer.mbed.org/users/ketjow/code/vcdMaker_Demo/

The demo for the STM Discovery L476 kit:

https://developer.mbed.org/users/ketjow/code/vcdMaker_Demo_DISCO_L476/

The demo for the Microchip Xpress kit:

https://mplabxpress.microchip.com/mplabcloud/example/details/261

11 Release notes

11.1 Version 1.0.1

- **vcdMaker**
  - The initial version of the application
  - Supported platforms:
    - *Win32*

11.2 Version 2.0.1

- **vcdMaker**
  - New features:
    - *Updated command line interface – removed ' -f ' option*
    - *Line counter*
    - *Events logging*
  - Supported platforms:
    - *Win32*
    - *Linux*
  - Resolved issues:
    - *#1 - Floats in vcd not rounded.*
- **vcdMerge**
  - The initial version of the application
  - Supported platforms:
    - *Win32*
    - *Linux*

11.3 Version 3.0.1

- **vcdMaker/ vcdMerge**
  - New features:
    - *Parsing user logs*
    - *Error/Warning reporting*
  - Resolved issues:
    - *#22 – Linux target shall be able to cope with logs containing Windows EOLs*

- Supported platforms:
  - *Win32*
  - *Linux*