vcdMaker

A User's Guide for vcdMaker
Edition 1
February, 2016

Table of Contents

1 Licensing

2 Introduction

Quite often while debugging a computer system there are created huge log files which 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 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 which allows for their further visualization. As a single picture is said to be worth thousand words the approach makes the debugging process easier and more efficient.

The tool is provided as a freeware. There are no charges for using it but the development team would appreciate some 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 Installing

Start the installation by executing the *vcdMakerInstaller.msi*. 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:

```
msiexec /i vcdMakerInstaller.msi /l*v vcdMakerInstallation.log
```

3.2 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:

```
msiexec /x vcdMakerInstaller.msi
```

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

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 7 first.

5 Creating log files

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

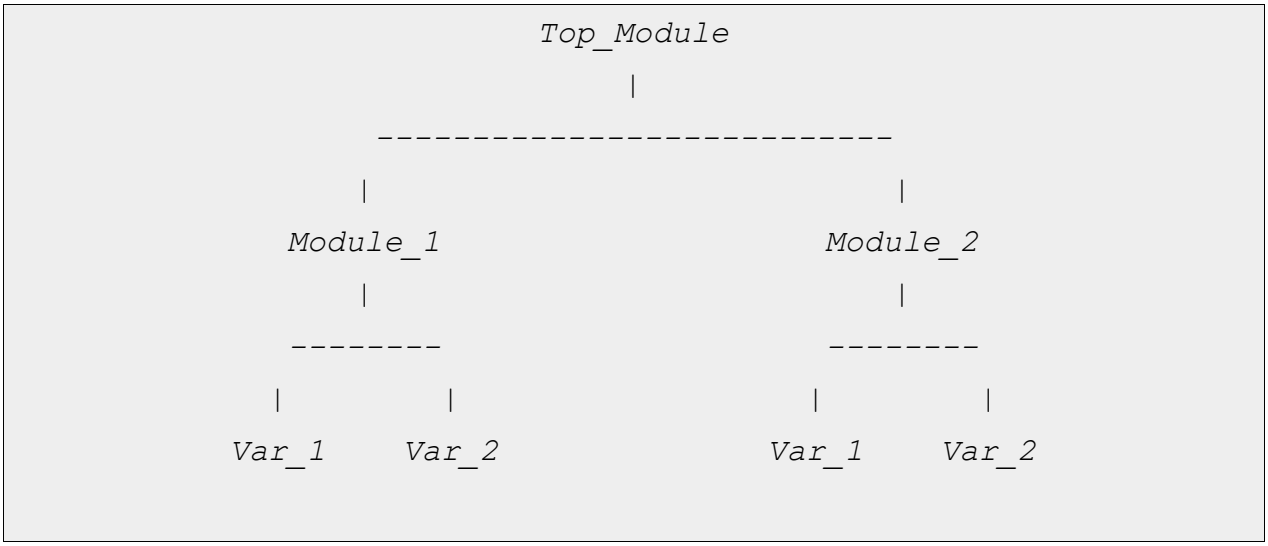
```
...
#77655698 FRDM.Sensors.Magnetometer.MagY 873 32
#77655817 FRDM.Sensors.Magnetometer.MagZ 3126 32
#77756251 FRDM.Buttons.SW1 1 1 Key pressed
#77805674 FRDM.Sensors.Magnetometer.MagX 64955 32
#77805797 FRDM.Sensors.Magnetometer.MagY 853 32
#77805916 FRDM.Sensors.Magnetometer.MagZ 3191 32
#77806061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

The structure of the log line is:

```
#TS Level1...LevelX.Signal_Name Signal_Value [size | f] Comment
```

TS – 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.** It might be that 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 the buffers randomly. The events will be reordered properly during the VCD file creation.

Level and **Signal_Name** – The structure describing the hierarchy for accessing the signal or the variable. The hierarchy can be illustrated by the tree. For instance:



In total there are four variables in the system above. They could be described 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
```

Signal_Value – The signal value can be an integer value or a real value (represented by the `%f` `printf()` format).

[size | f] – The signal hierarchy is followed by the *size* parameter or *f*. The *size* parameter shall follow integer signals and represent the number of bits needed to represent the value. The *f* character shall follow a real value.

Comment – The comment is optional. It might be any user defined string.

6 Creating VCD files

Producing the VCD file from the log file saved in the format presented in 5 is not difficult. The syntax of the command is:

```
vcdMaker -f log_filename -o vcd_filename.vcd -t time_unit [-v]
```

log_filename – The name of the log file created in the format presented in 5.

vcd_filename – The target VCD file name.

time_unit – The time unit in which the samples have been recorded. Available time units are: *s*, *ms*,

us, *ns*, *ps*, *fs*.

-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.

7 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.

7.1 GtkWave

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

7.1.1 Downloading

The tool can be downloaded from:

http://gtkwave.sourceforge.net/

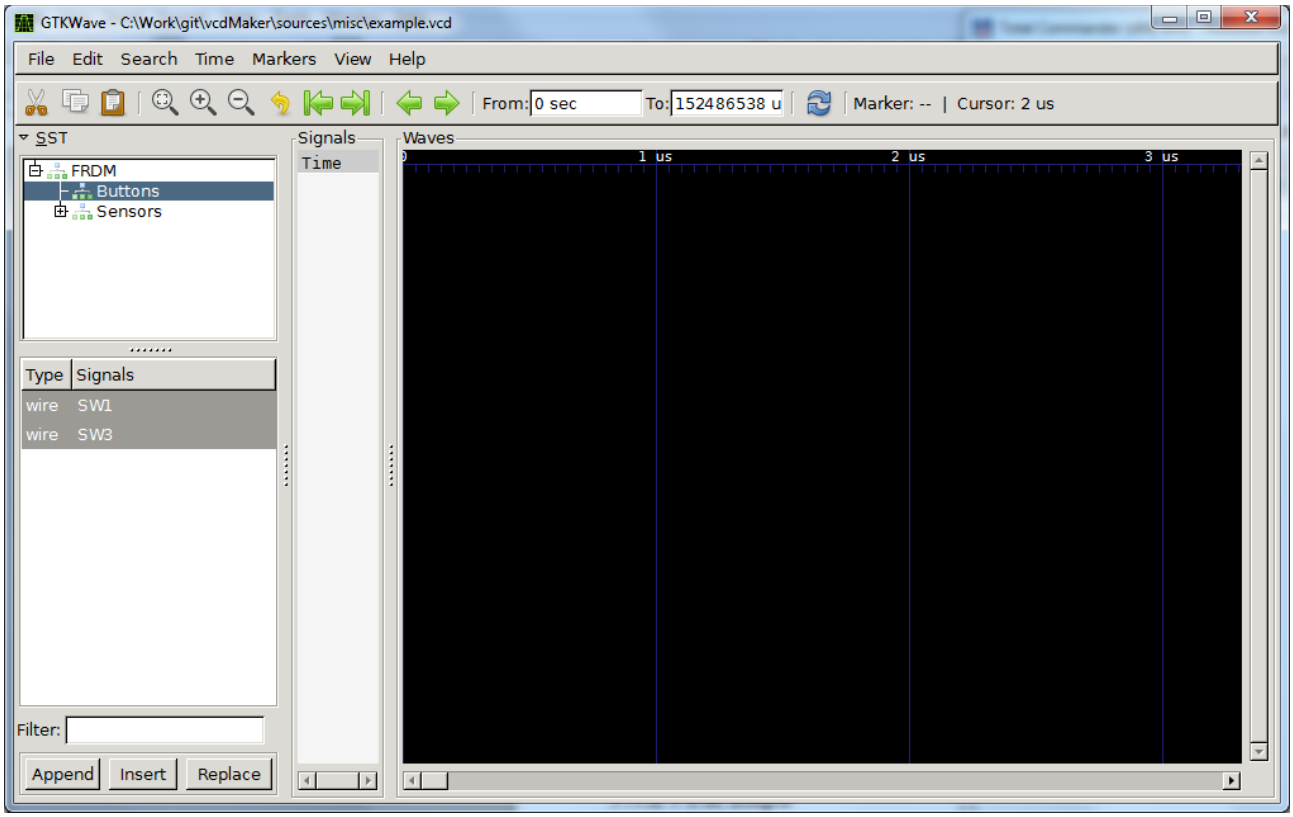
7.1.2 First steps

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

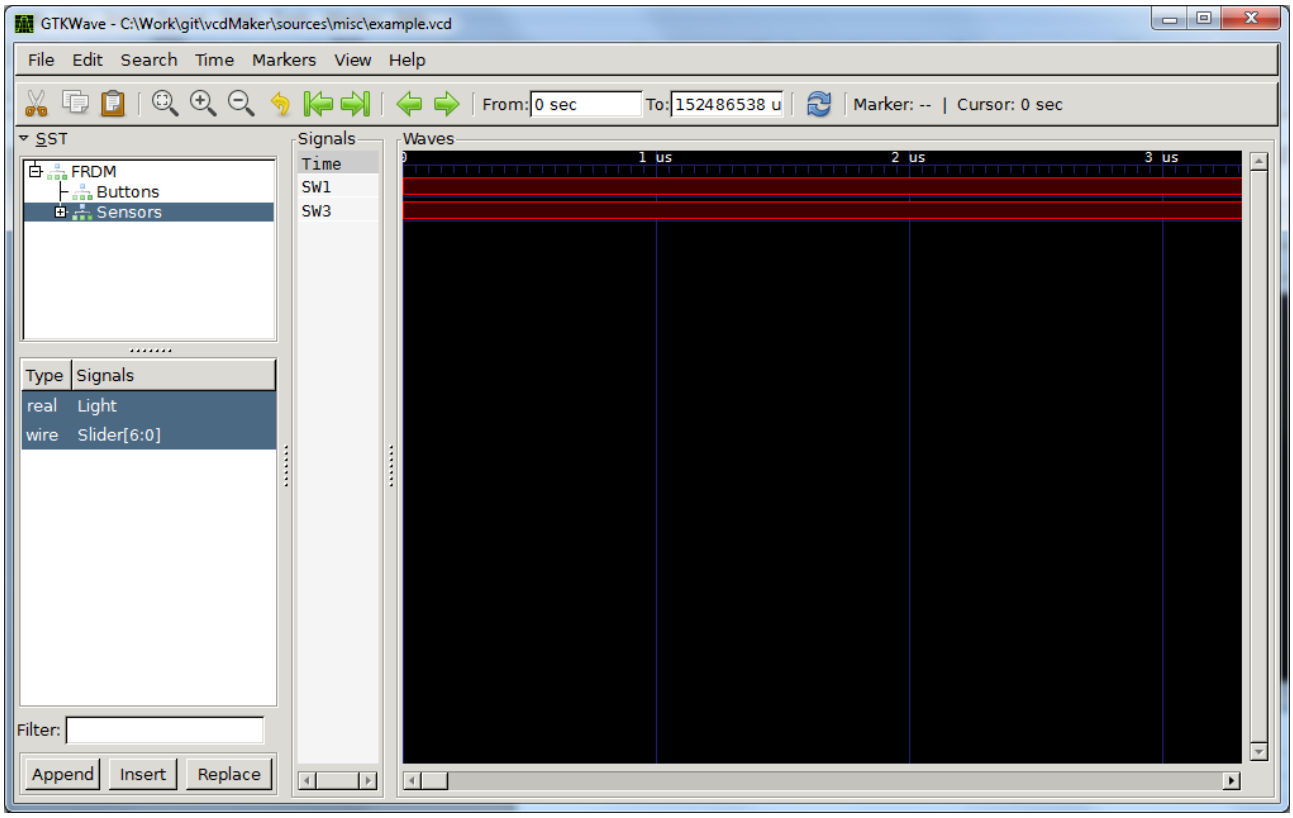
Run the tool. Drag and drop the *examples.vcd* file to the GtkWave.

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.



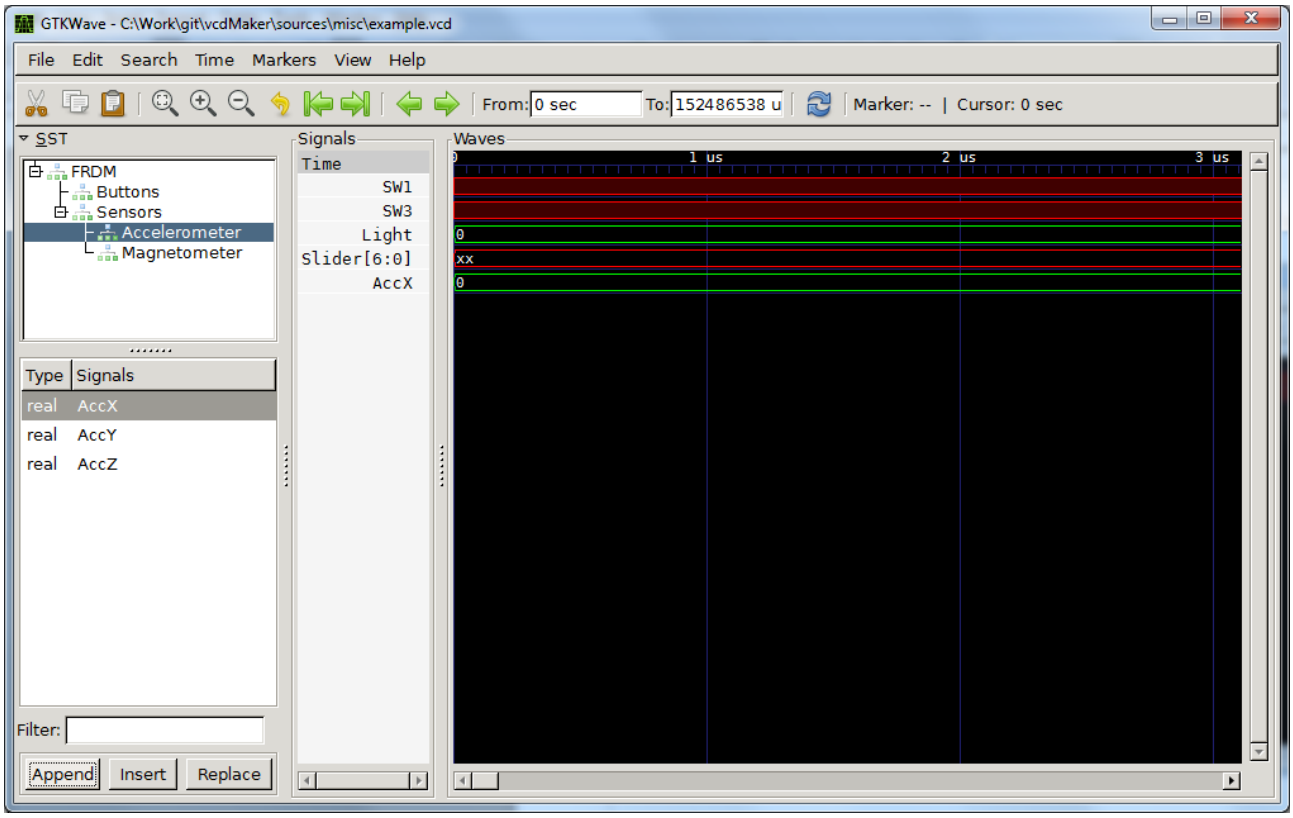
Step 2

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.



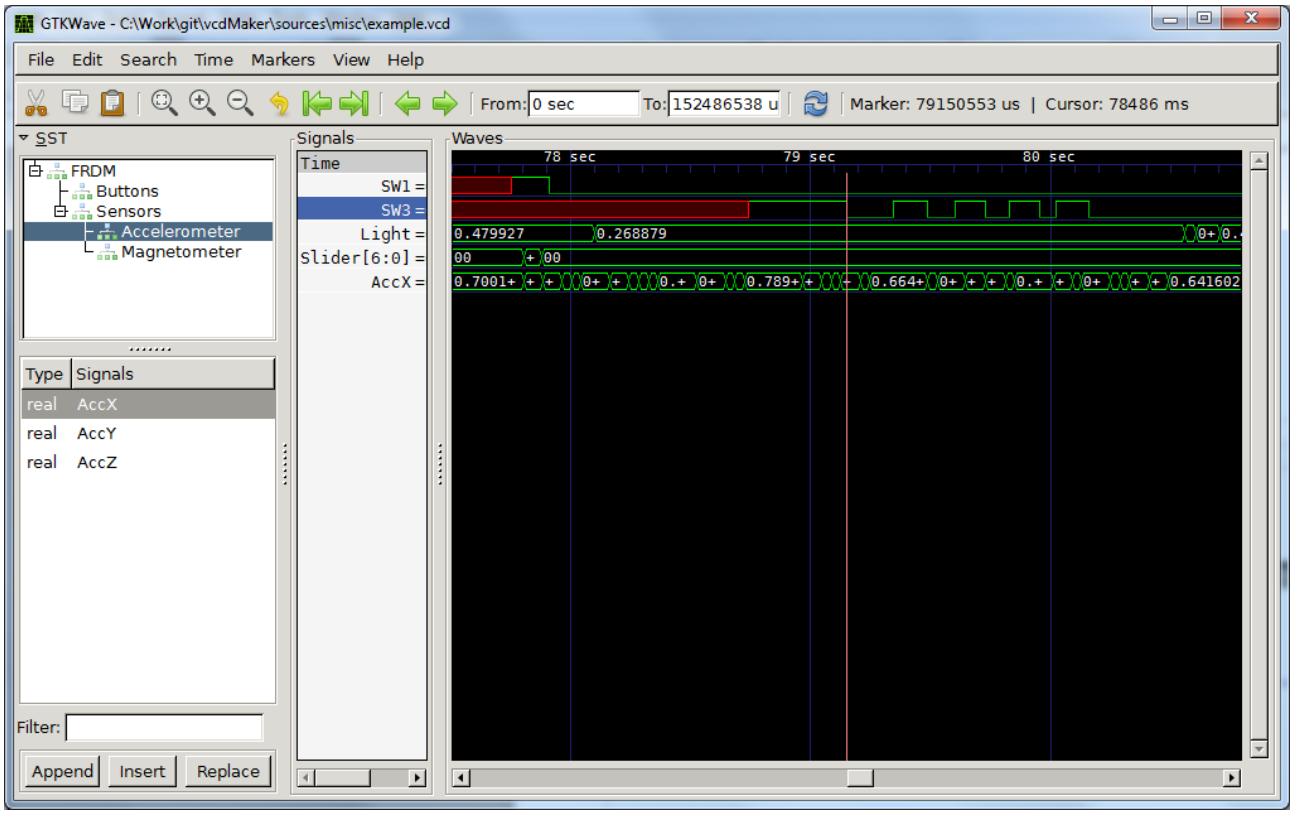
Step 3

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



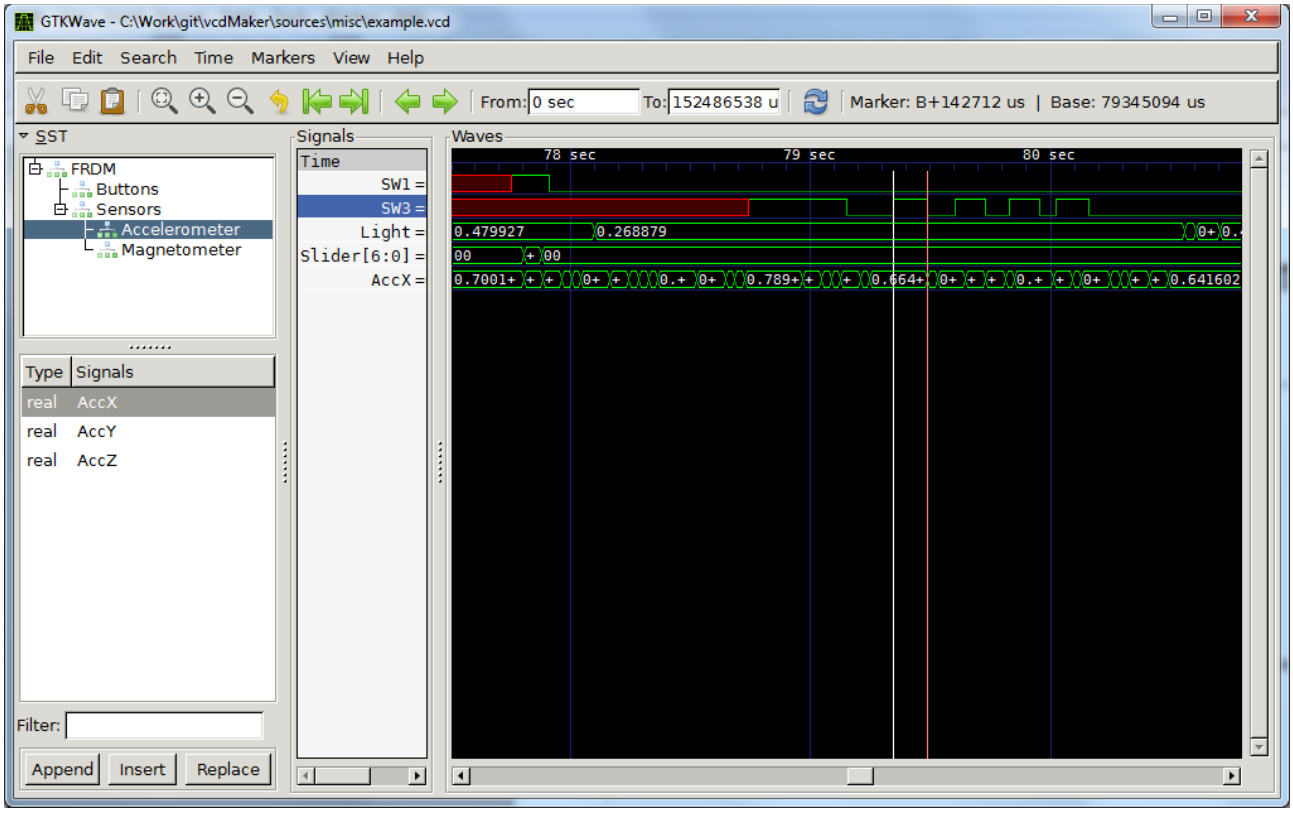
Step 4

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.



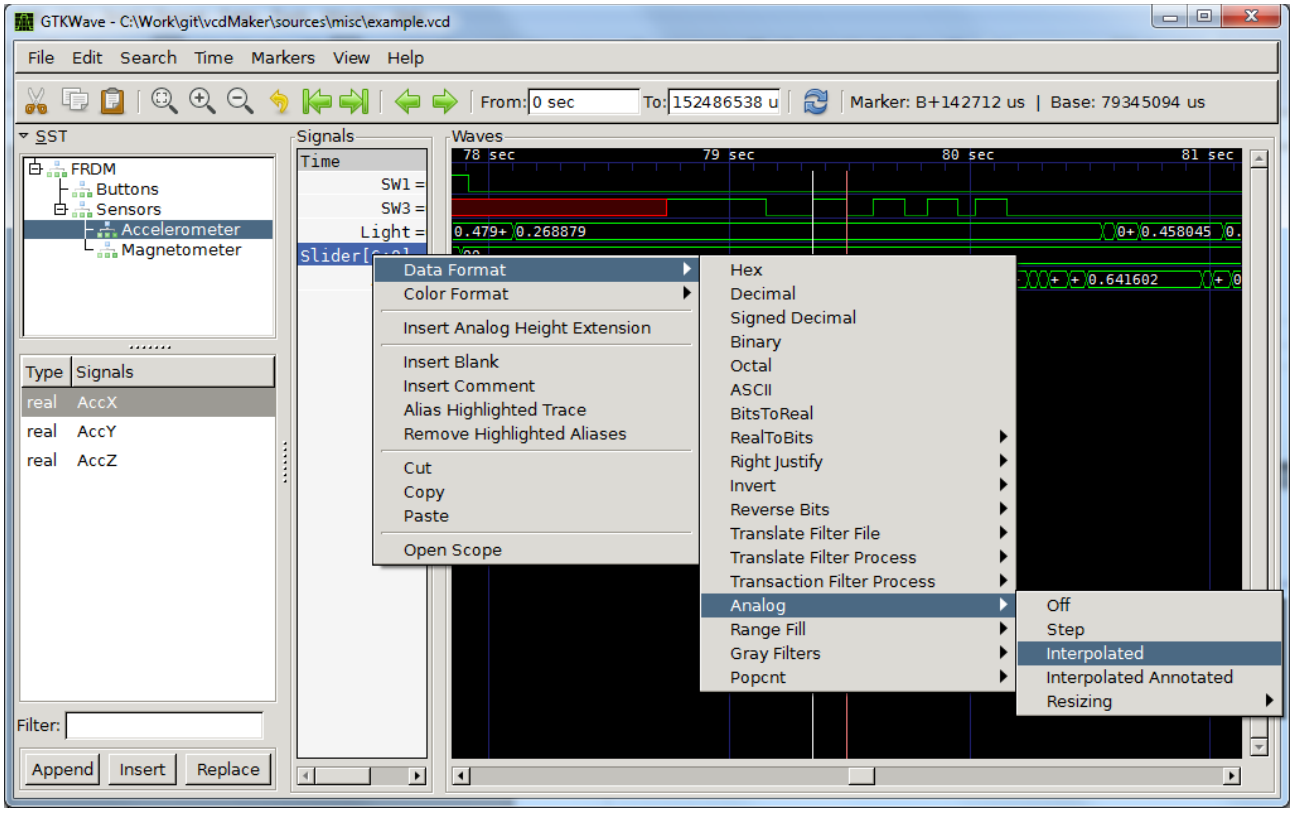
Step 5

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.



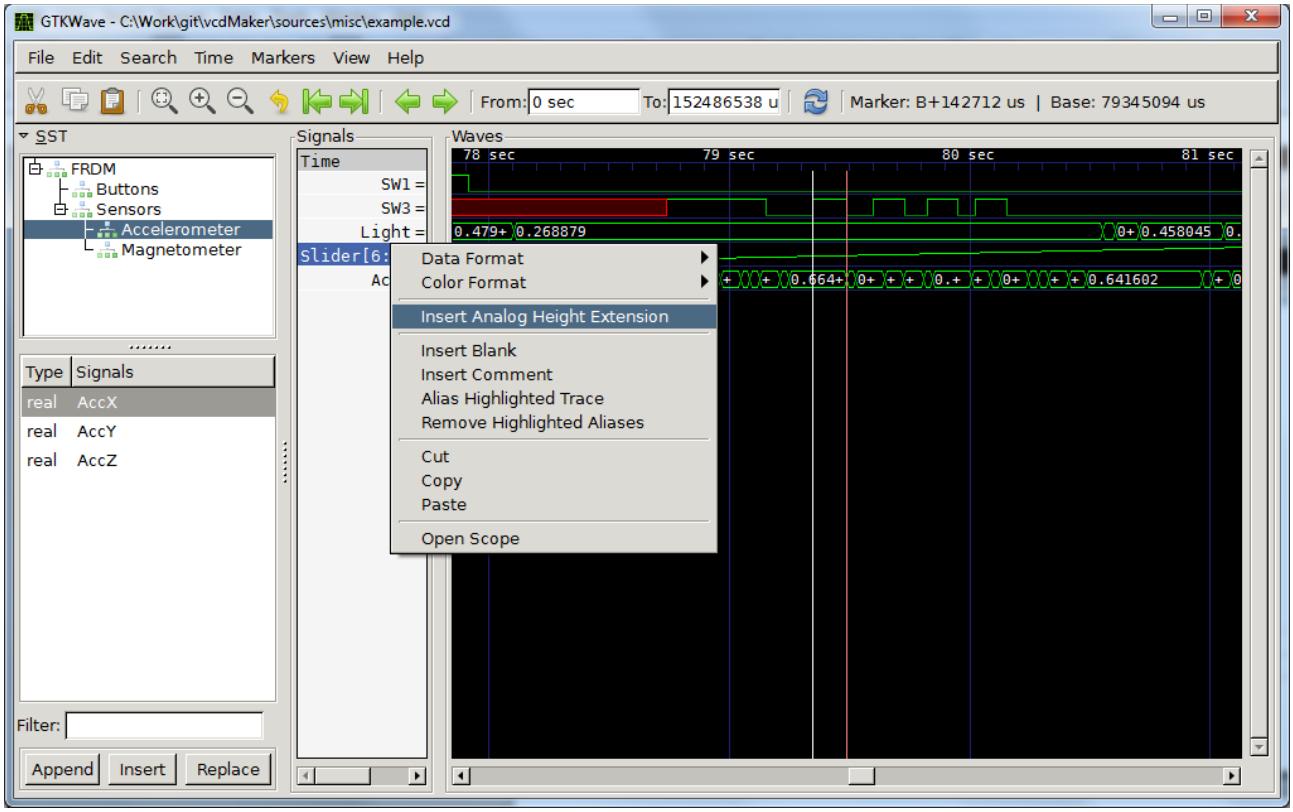
Step 6

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.