Building a realtime model at the TAMA Linux machine

Kiwamu Izumi
Dept. of Astronomy, Univ. of Tokyo
Oct. 9th 2012

1 Scope of this paper

This document attempts to summarize how to build and run a realtime model, which runs on a realtime linux machine at the TAMA basement floor. This document has been written along the course of building a model for simulating the KAGRA frequency stabilisation system in a realtime model. Therefore most of the topics described here maybe related to the frequency stabilization model more or less.

2 Overview

The realtime model is a model which creates an associated realtime process (module). The realtime model may also refer to the resultant executable module too. The realtime module is built from a set of of C codes, some of which are prepared by the LIGO CDS group as a library and template and some are newly built from the simulink model. Figure 1 shows a cartoon of the realtime model illustrating the relation of the realtime model and its associated codes.

At the beginning of the building process, users have to edit a simulink model in which desired control sequence and so forth can be specified as a drawing. Then the simulink diagram will be interpreted to C codes by a Perl script. Since the simulink model is merely a text file, what the Perl script does is to parse the contents to appropriate C code lines. Then the C codes are complied and it results in a kernel module which plays the role of the realtime processor. At this point users can run the process by insmod which is usually wrapped in a auto-generated shell script.

3 Setup

The realtime digital system at TAMA consists of a realtime Gentoo linux machine and a standard Ubuntu linux machine. The Gentoo machine is

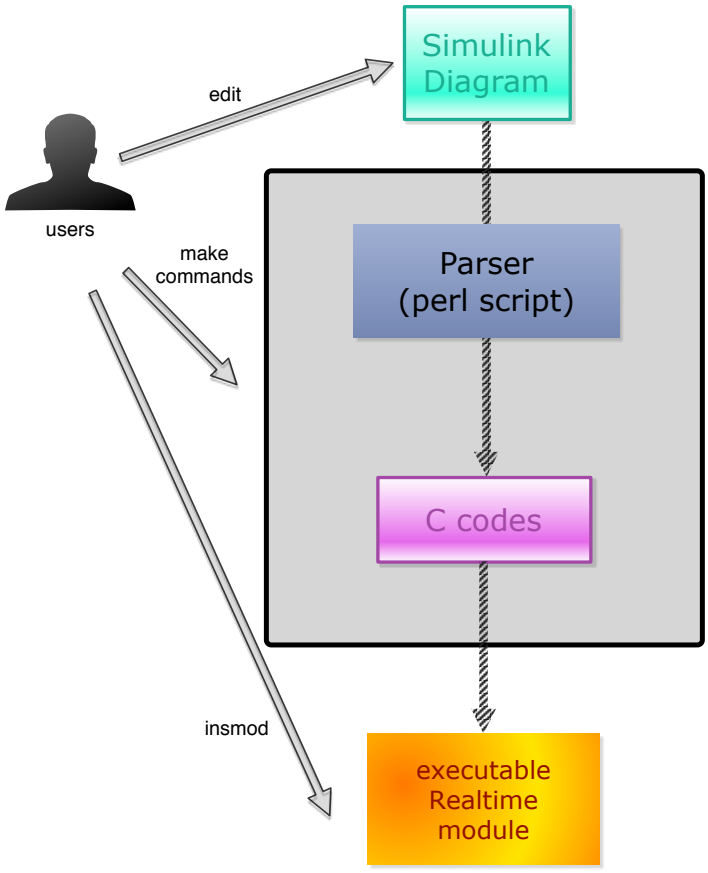

Figure 1: Simulink model and its associated codes and modules.

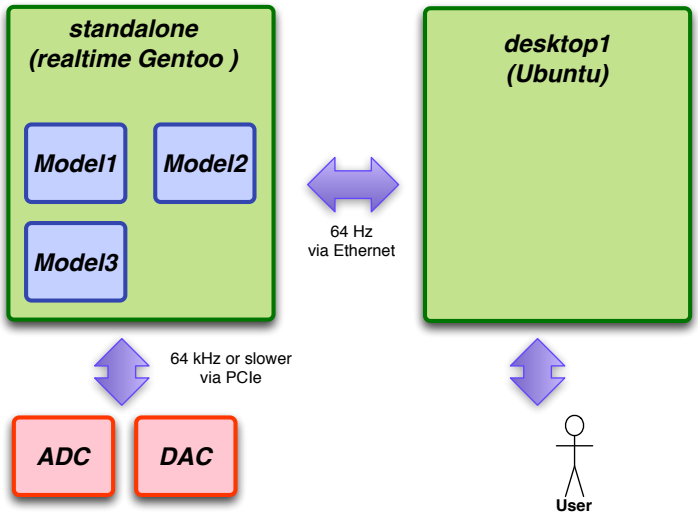

Figure 2: A schematic of the machine setup at TAMA. There are two Linux machines at the moment; a realtime Gentoo machine for running realtime models and a standard Ubuntu machine. They are called *standalone* and *desktop1* respectively.

dedicated for running the realtime processes and the Ubuntu machine provides an interface of the realtime process to users by monitoring the digital values and controlling the EPICS values. Figure 2 illustrates a schematic of the setup.

4 How to make and build a realtime model

This section describes procedure to make and build a realtime model. All the description here is meant to be specific to the TAMA digital system. Therefore the descriptions below may not be directly applied to another digital system. Note that the TAMA digital is at a RTS version of 2.1.4 at the moment while most of the LIGO lab have a version of 2.4 or later.

4.1 Drawing a simulink diagram

First of all it requires a simulink diagram, which is merely used as a "drawing". Go to a directory which contains all the simulink diagrams.

```
cd /opt/rtcds/tst/x1/core/advLigoRTS/src/epics/simulink/
```

Call matlab R2010b (which had been already supplied with the standalone machine from the beginning),

```
/opt/apps/linux64/matlab/bin/matlab &
```

In the matlab window, include the pre-defined LIGO module blocks by typing

```
addpath lib
```

This allows you to use the LIGO module blocks such as IIR filters and so forth. Additionally one can open a window which contains all the available blocks by typing

```
open CDS_PARTS
```

This simulink diagram shows a correction of the available blocks for this purpose. Most of the blocks shown in this window are usually accompanied with a text box in which descriptions are described by the LIGO CDS developers.

Then open a model file which you want to edit (or you can create a new model)

```
open x1AAA
```

where x1 denotes the name of the interferometer (e.g. c for Caltech, h for Hanford and x for a test) and AAA can be substituted by a model name, for example, x14, x15 and fsp. With the x1AAA simulink diagram one can edit the model as desired.

An important thing here is the 'model description'. This usually shows up in the upper left corner of the simulink diagram. One needs to properly assign 'dcuid', 'specific cpu' and so forth. If another model uses the same dcuid or specific cpu, the two models will conflict and thus won't correctly run. So be aware of it.

In the same text field, one can choose the sampling frequency of the model, which corresponds to the update cycle of the realtime processor. Currently x1fsp, which is for the simulated frequency stabilisation system, has a sampling frequency of 64kHz. The name of the host computer must be *standalone* in the case of the TAMA digital system.

4.2 Make and Install

Make and install of the model is fairly simple ; one just type a couple of make commands. Behind the make commands there is a perl script which actually interpret and parse the simulink diagram into several C codes as explained in section 2. The procedure is explained in the following sentences. All the following commands need to be executed in standalone in order to appropriately run the make commands which generally depends on the file and library locations.

First of all one must go to the place where the makefile resides,

```
cd /opt/rtcds/tst/x1/
```

Then make the model

```
make x1AAA
```

where AAA denotes the name of the model which you want to compile.

Then create associated daq files, MEDM screens and kernel object (realtime model) by typing,

```
make install-x1AAA
```

This command will create an executable kernel object – that is the realtime process.

Note that install-x1AAA actually calls three different make commands that are 'make install-daq-x1AAA', 'make install-screens-x1AAA' and 'make install-x1epics' (?). Therefore one can separately make each command if needed. In the past we needed to type all these commands sequentially, but recently it has been changed such that one can make the model just by 'make x1AAA'.

4.3 Run the realtime model

After the make commands, the realtime model is created, but it will not run spontaneously. Therefore one needs to manually run the realtime module by executing a script.

First, you log into the realtime OS and go to the script directory,

```
cd /opt/rtcds/tst/x1/scripts/
```

Then start the model by typing,

```
./startx1AAA
```

This is a script automatically generated during the make process. What it does is to 'insmod' (install-module) the kernel module of x1AAA.ko where ko stands for kernel object. Note that this startup command includes a kill command such that it automatically kills the previously running realtime process. In addition to it there is a dedicated command to kill the process, called killx1AAA in the same directory.

The log messages will be written in a ring buffer of the OS such that the messages can be read by just typing dmesg. This message was previously written in a local log file, but it has been changed. One can check the startup time by checking the date of the log file which resides in /opt/rtcds/tst/x1/target/logs/

4.4 Preparation of IIR filters

Use foton. Go to a directory in which filter coefficients are saved,

```
cd /opt/rtcds/tst/x1/chans/
```

Then call foton by typing

```
foton &
```

Foton is an graphical interface to design the IIR filter. Once IIR filters are designed and assigned by foton, one always needs to load the coefficients. The loading can be done by pressing the dedicated button on the MEDM screens.

5 MEDM screens

Once the realtime model is successfully build and installed, MEDM can give us an easy access to the EPICS values (e.g. filter gains, offsets and etc.). The MEDM screens are currently in

```
/opt/rtcds/tst/x1/screens/
```

Some of the screen files in this directory are automatically generated through the make chain. For these filters one needs to be careful because the screen file can be automatically updated by the make chain. Therefore it is always better to make a different directory to store user-custom-made screens so that so as not to be updated by the make chain.