ALOE Session 6: Creating Waveform Components from Simulink

Ismael Gomez, Vuk Marojevic, Antoni Gelonch
Universitat Politècnica de Catalunya

April 2011

1. Objective

This session presents the ALOE Simulink Target for creating waveform components. The focus is on the generation of the C code implementation for ALOE from a Simulink model, rather than on the design and verification of Simulink models.

2. Overview

- System preparation
- Exporting components
- Compiling for ALOE
- Invoking the component

3. Requirements

- PC running Linux, kernel 2.6.21 or above
- ALOE version 1.3 downloaded and installed on your PC (see ALOE Session 1)
- Basic Linux user skills
- MATLAB and Simulink, Release 2008 or later
- A Simulink model of your waveform components

4. ALOE versions

We continuously evolve the ALOE framework and tools. This session is compatible with ALOE version 1.3. Consult http://flexnets.upc.edu/trac/wiki/ALOEedu for updates.

5. Procedure

5.1. Preparing the system

Digital signal processing modules can be automatically generated with MATLAB/Simulink. This high-level simulation tool allows testing the functionality of your module and synthesizing C code compatible with ALOE. In addition, an ALOE target for the Simulink Real-Time Workshop [The MathWorks, "Real-Time Workshop 7", 9400v06 09/07, available at www.mathworks.com] is available and automatically generates the interface to the *ALOE SWAPI*.

This document explains by means of a hands-on example how to create signal processing modules from Simulink. The ALOE Real-Time Workshop target, which is available with the ALOE package distribution, is used to generate C code for ALOE from a Simulink model. The model can be created from a set of Simulink library components or from user programmed Matlab functions (M-models).

MATLAB/Simulink and ALOE may run on two operating systems. You need to access the MATLAB and ALOE paths (not necessarily concurrently) and move data from one path to the other. This is explained in continuation.

5.1.1. Copying files to the MATLAB directory

The ALOE Simulink target features the following files:

- *lnx_callback.m*
- *lnx_install_dir.m*
- *aloe_main.c*
- *lnx_unix.tmf*
- *make_lnx.m*
- *aloe.tlc*
- *rtwmakecfg.m*

These files are included in under the *$ALOE/matlab/* directory. (*$ALOE/* indicates the directory where you extracted and installed ALOE on your PC.)

Create the following directory:

```
%MATLAB%/rtw/c/aloe/
```

%MATLAB% indicates the directory, where Matlab is installed on your PC. Now copy the ALOE Simulink target files from *$ALOE/matlab/* to *%MATLAB%/rtw/c/aloe/*.

Start Matlab and add *%MATLAB%/rtw/c/aloe/*to the Matlab path (**File->Set path...**).

5.1.2. Copying MATLAB directories

You also need to copy the Matlab libraries and headers to a Linux directory. Create the directory

```
matlab-files/
```

under your home directory, for example, and copy the following directories and their subdirectories from the MATLAB installation path to *matlab-files/*:

```
%MATLAB%/extern
%MATLAB%/rtw
%MATLAB%/simulink
%MATLAB%/toolbox
```

Note that the ALOE Simulink target directory *rtw/c/aloe* also needs to be copied.

The amount of files to copy may be large. Although it would suffice to copy only those toolboxes that will be used, we recommend copying all.

5.2. Generating C code for ALOE

Start Matlab and select **File->New->Model** to open the Simulink environment (or open an already created Simulink model). Add your Simulink blocks to create a model as usual. (For more help on creating models please refer to the Simulink documentation).

Generating C code for ALOE from a Simulink model requires configuring several options in the Simulink environment. You should first indicate the Simulink signals that will be used as inputs and outputs. A Simulink signal will not be accessible from ALOE unless you select the signal option *Test point*. Second, you need to explicitly specify whether a external signal is an input or an output. Therefore, add the "in_" (for an input signal) or "out_" (for an output signal) prefix to the signal name. The names that ALOE will use to access these signals will later be specified in the *Application Description File* as

- *in_itfName_re* and *out_itfName_re* for the real part and
- *in_itfName_im* and *out_itfName_im* for the imaginary part.

Finally, you need to select the method for accessing signal samples.

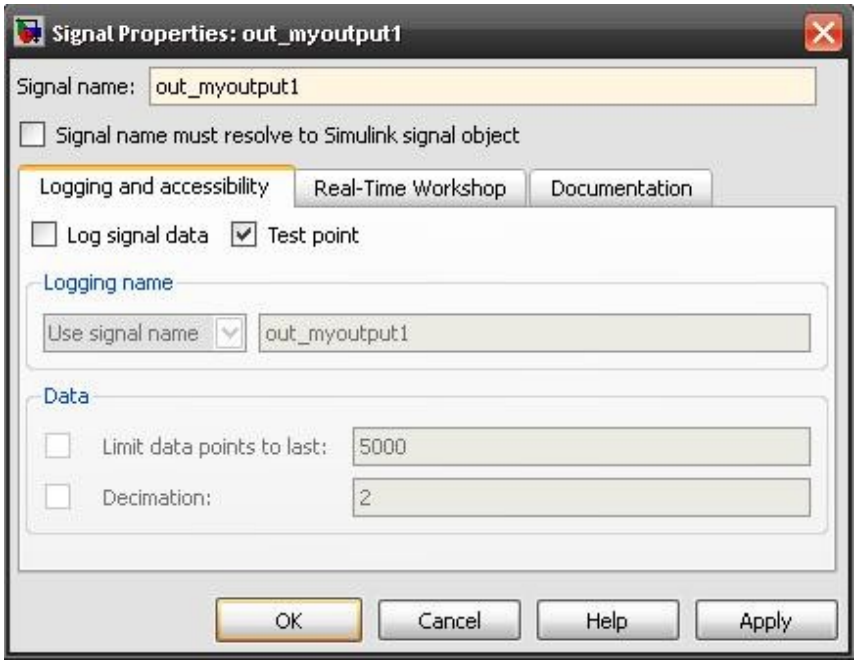

Figure 1 - Signal configuration.

The above options can be configured in the Signal Properties window. Right-click on the signal and select **Signal Properties**. Activate the *Test point* option and specify the *Signal name* **out_myoutput1**, for example (Figure 1).

Under the *Real-Time Workshop* tab select the storage class "ExportedGlobal" (Figure 2). This defines how the output signals **out_myoutput1** is accessed from ALOE, by means of a global variable.

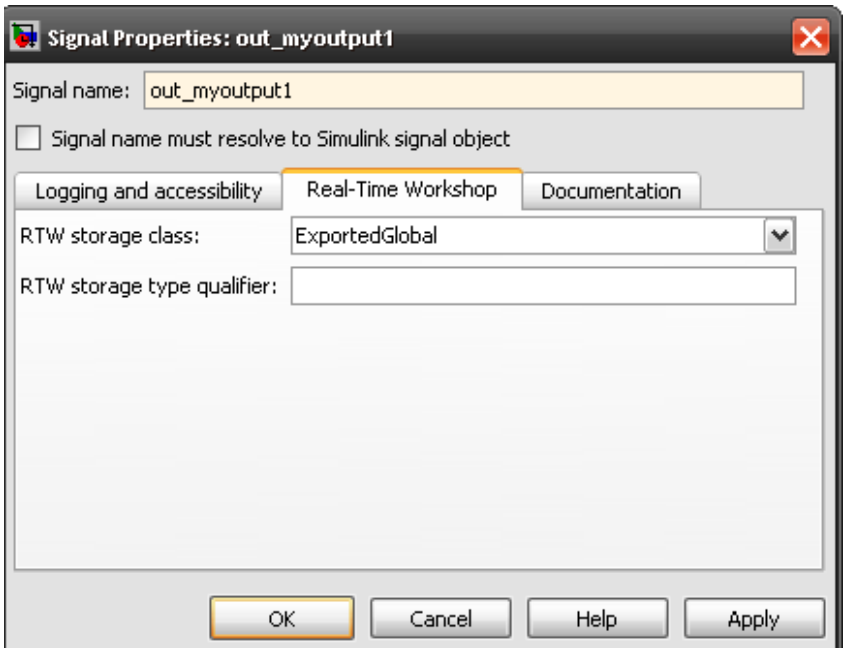

Figure 2 - Signal storage class.

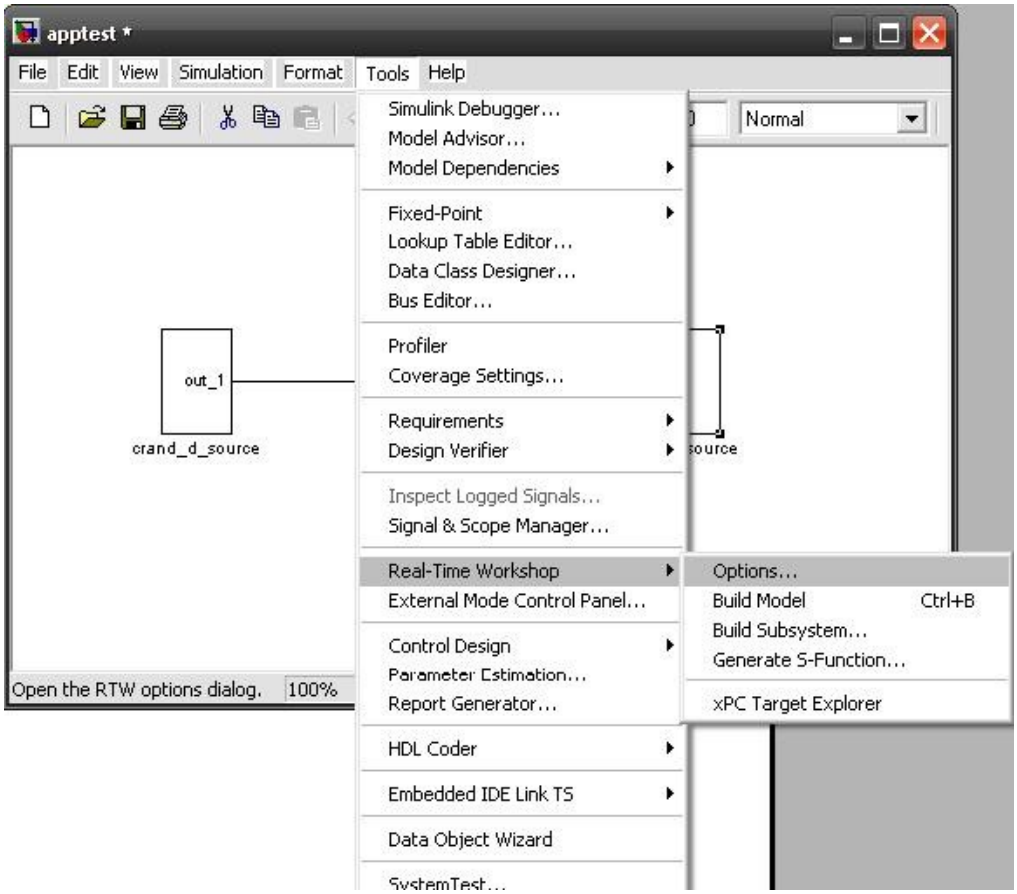

Figure 3 - Real-Time Workshop options.

The following steps configure the Simulink model to generate an implementation for ALOE. In the Simulink model window, choose **Tools->Real Time Workshop->Options...** form the main toolbar (Figure 3).

In the *Configuration Parameters* window (*Real-Time Workshop* tag) browse for the *System target file* **aloe.tlc** (Figure 4).

Select the Interface tag in under the *Real-Time Workshop* menu in the left subwindow and choose the **C API** *Interface* (Figure 5).

Make sure that target files are correctly specified for the Linux environment. Therefore, configure the *ALOE Target code generation* options as shown in Figure 6.

You finally need to select the model for converting continuous signals to a discrete representation. Therefore, select the *Solver* tag and choose **Fixed-step** *Type* and **Discrete (no continuous states)** *Solver* (Figure 7). Set the simulation time to infinite: **inf** as the *Stop time* option. Make sure that all other parameters are set according to Figure 7.

You can now generate the C code implementation for ALOE. Click on **Tools->Real Time Workshop->Build Model** and the code will be generated. (See the Matlab's main output screen in case of errors.)

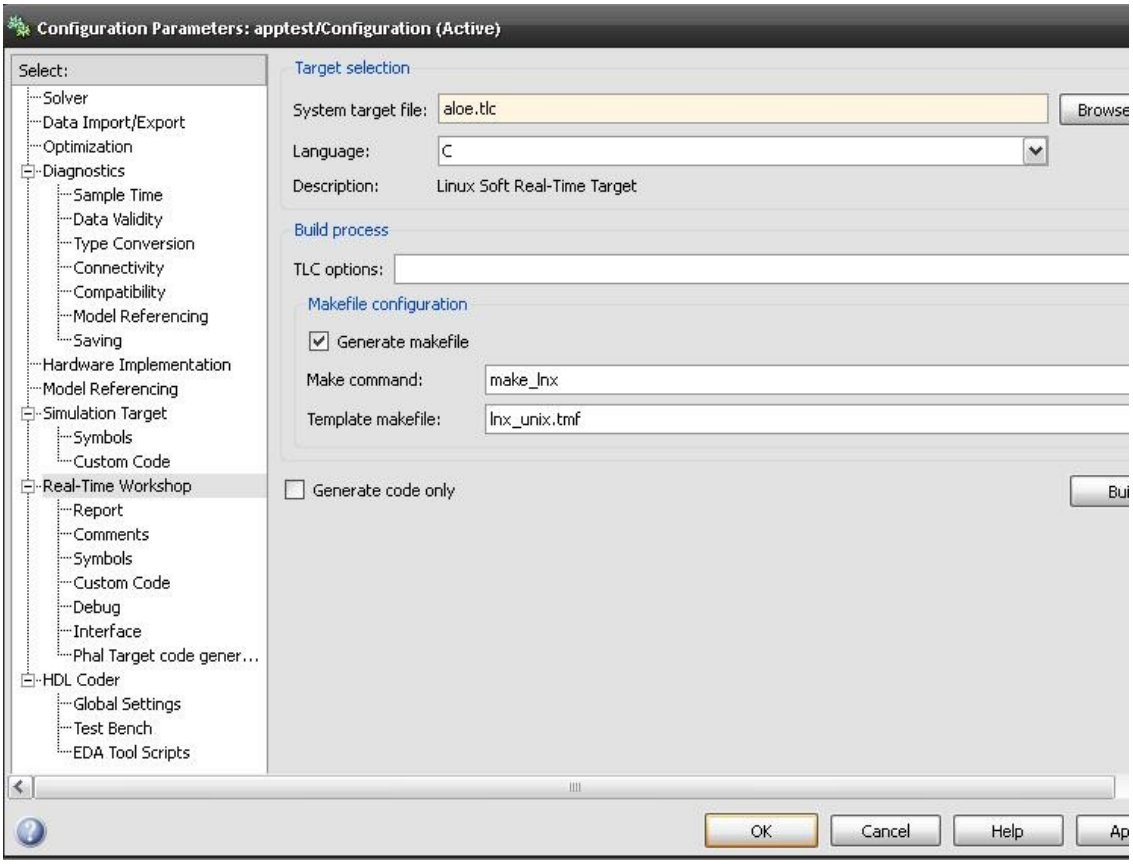

Figure 4 - System target file selection.

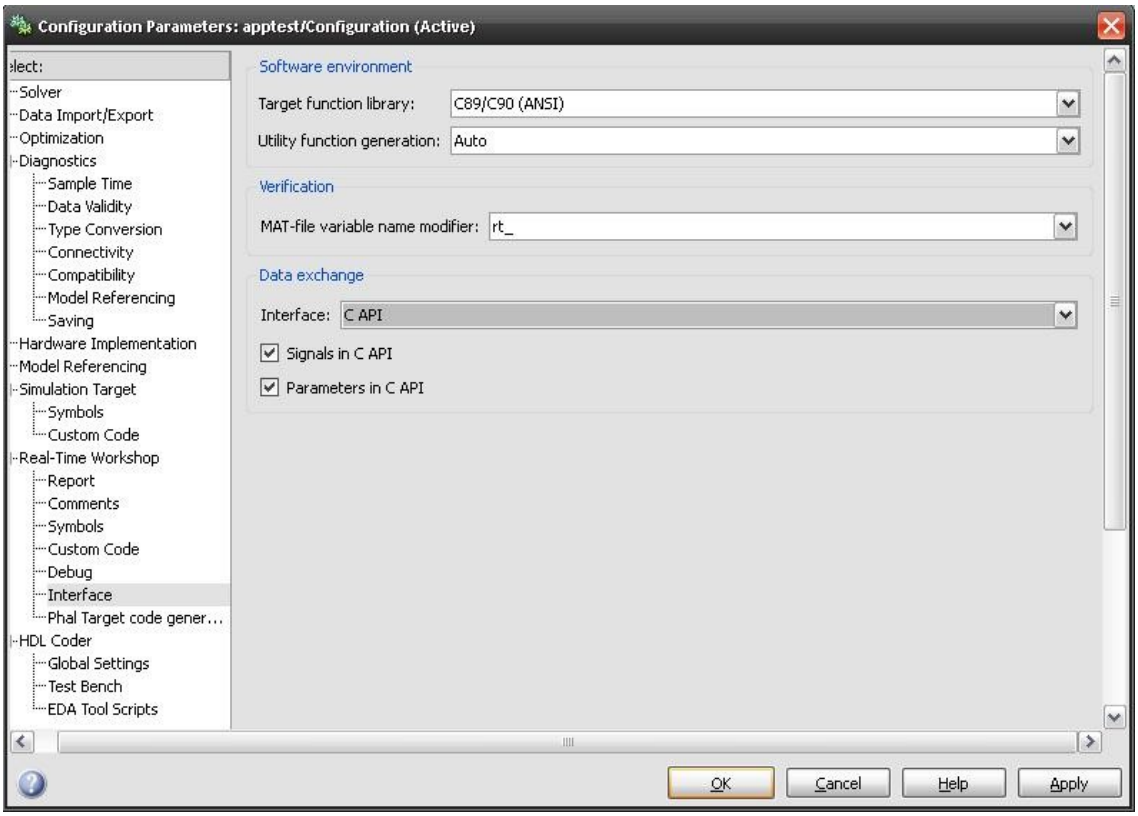

Figure 5 - System target signal interface configuration.

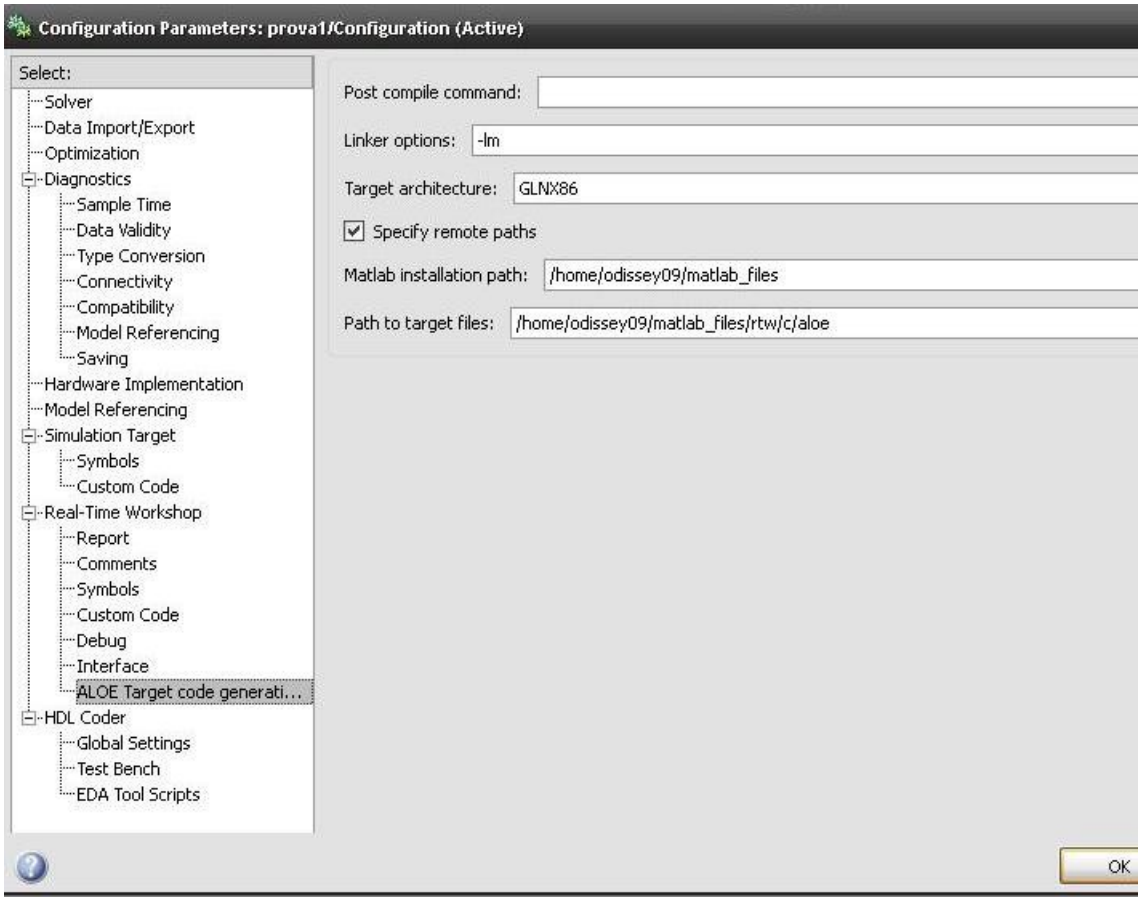

Figure 6 - Matlab paths configuration.

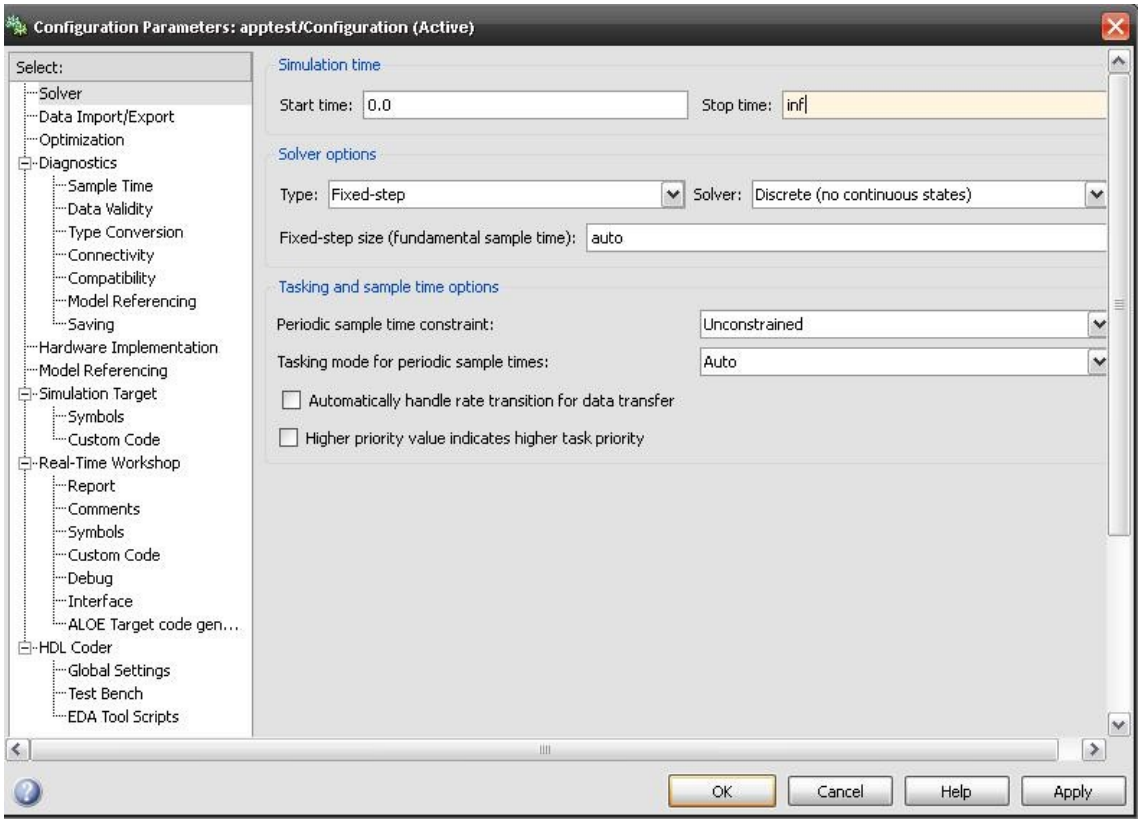

Figure 7 - Solver configuration.

5.3. Compiling for ALOE

The files generated by the Simulink model generator are available under *%MATALB%/modelname_aloe_rtw/*. Copy these files to a Linux directory: Create a new directory, *$ALOE/modules/matlab_modelname/*, for example, and copy the files from *%MATALB%/modelname_aloe_rtw/*.

Switch to the Linux directory that now contains the Simulink output files *($ALOE/modules/matlab_modelname/*) and compile the component:

```
make -f modelname.mk
```

This will create the executable file *modelname*.

5.4. Deploying the new components

Now copy the executables of all waveform components (generated from Simulink) to the ALOE SWMAN executable repository. This repository is located at *$ALOE/example-repository/swman_execs/linux/*. It is necessary to place the executables there for the ALOE framework being able to access them. Note that this directory typically links to */usr/local/bin*, which requires root privileges.

You can now create the *Application Description File* for specifying the components' interconnections. Pay attention to the interface name convention for input and output interfaces and real and imaginary parts. Figure 8 shows a draft of the *Application Description File* for the ALOE component *crand_d_source* connecting to *prova1*. If you are not familiar with *Application Description Files*, see ALOE Session 2 for more details.

Figure 8 - Application Description File draft.

```
object {
        obj_name=crand_d_source
        exe_name=crand_d_source
        proc=1000
        outputs {
                name=out_1_re
                remote_itf=in_1_re
                remote_obj=prova1
                bw=1000
        }
        outputs {
                name=out_1_im
                remote_itf=in_1_im
                remote_obj=prova1
                bw=1000
        }
}
```

This finishes ALOE session 6. Please send your feedback to flexnets.pmt@upc.edu.