RPIt Quick Start

Preparing the Target

A RPIt target is a Linux system running a Debian-like distribution. It can be an Intel or ARM platform, a Debian, Ubuntu or Raspbian distribution. Follow the distribution installation procedure to install a minimalistic system. There is no need for a graphical UI, packages could be limited to textual interactions. The essential development packages should be installed. Additional RPIt-specific packages are installed automatically when running the “setup.m” script. When running this script, the target should be connected to the internet in order to be able to download the packages.

The target should have a “pi” account. This account should be able to gain root access without a password as it is the case on the Raspberry Pi. This means that a command like “sudo bash” does not ask for a password.

All interactions between Simulink and the target are done with a password-less ssh communication. A private/public key pair is generated by “setup.m” and the passwords public key is uploaded to the target so that Simulink script using PuTTY can run distant script through ssh without asking a password to the user.

Running “setup.m” should yield the following trailing outputs in the Matlab console:

```
> Answer 'y' if asked for storing the key in the cache in the cmd window.
> Uploading public key to the target.
> Verifying the authorized keys list.
> Public key already known by the target.
> Checking the passworless connection [CRTL-C if hanging]...
> Passwordless configuration successfully operating.
> Distant target is a RPI.
> Copying MATLAB files (may take a while).
> Uploading S-Function files.
> Creating a working directory 'RTW' in user 'pi' home dir.
> Uploading Polaris rom file.
> Installing the 'screen' utility on the target.
> Installing the 'libusb-1.0-0-dev' package on the target.
> Enabling user serial port access by adding pi to the dialout group.
> Enabling user access to the EV3 USB port.
> Enabling user access to the NXT USB port.
> Reduce latency of FTDI chipsets.
> Installing the i2c utilities on the RPI.
> Verifying /etc/modules.
> /etc/modules already up to date.
> Configuring i2c at 400kHz.
> Verifying /boot/config.txt.
> /boot/config.txt already up to date.
> Adding user pi to the i2c group.
> Adding user pi to the gpio group.
> Checking cpu governor configuration.
```

```
> rc.local already set cpu governor to performance mode.
> Configuration of RPIt successfully completed.
> NOTE: please reboot your target for changes to take effect.
```

A typical HIL experiment in Simulink external mode consists in the following fully automated steps:

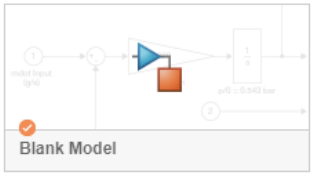
1. The Simulink model is converted into C code.
2. This code is uploaded to the target into the “/home/pi/RTW” directory.
3. The code is compiled directly onto the target (the Raspberry Pi is powerful enough to sustain this step).
4. The resulting executable is renamed “rpi” and is distantly launched within a “screen” context.
5. A TCP/IP socket communication is established by Simulink with the target executable.
6. Simulink remotely starts the experiment.
7. Bidirectional quasi-real-time communication allows for downloading measurement data from the target (for example to update Scopes plots) and to send to the target updated block parameters on-the-fly.
8. Pressing the “Stop” button tells the target executable to run the “Terminate” method and to quit. Socket communication is finally shut down.

Note

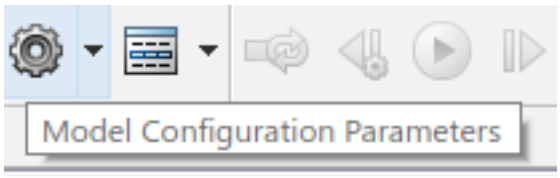
A good practice is to have only one model per directory. Indeed, the whole content of the directory is uploaded to the target, so if the directory holds too many models, the upload time may be very long.

Creating a Model from Scratch

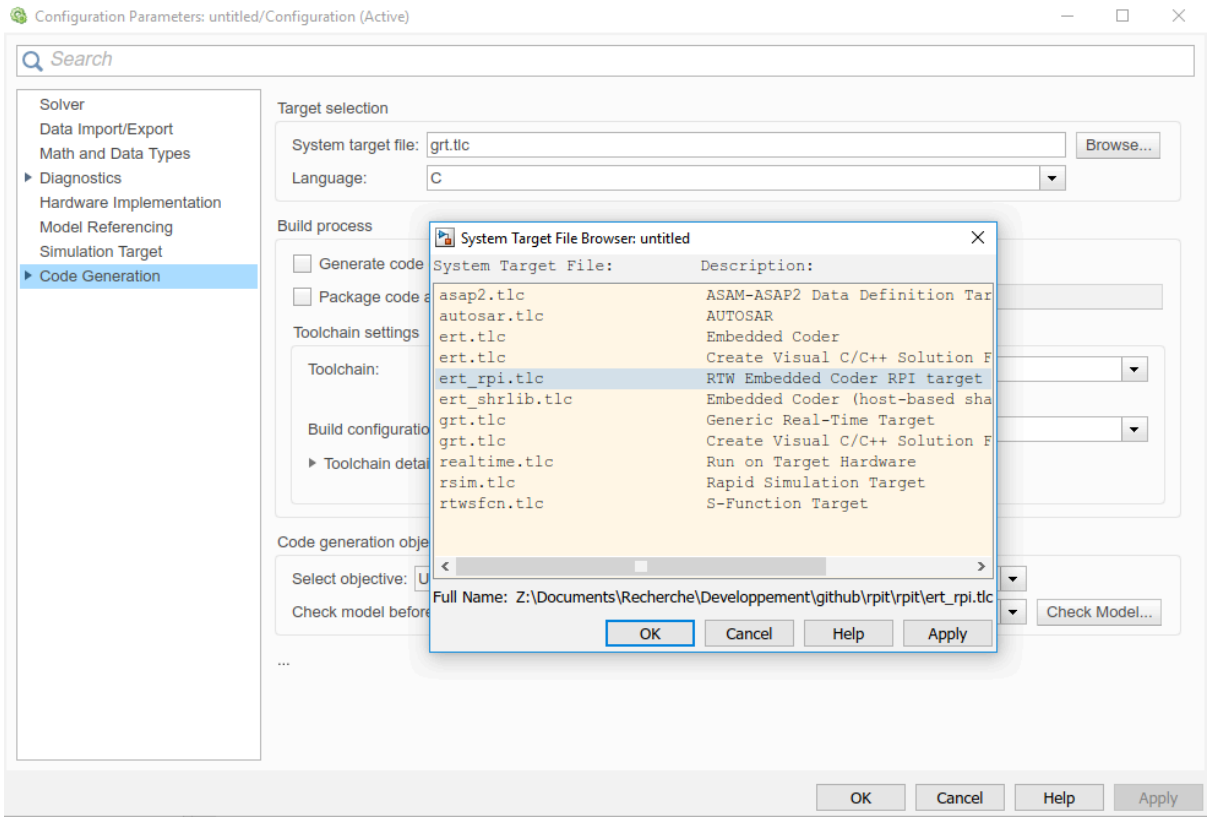
1. Create a blank Simulink model.



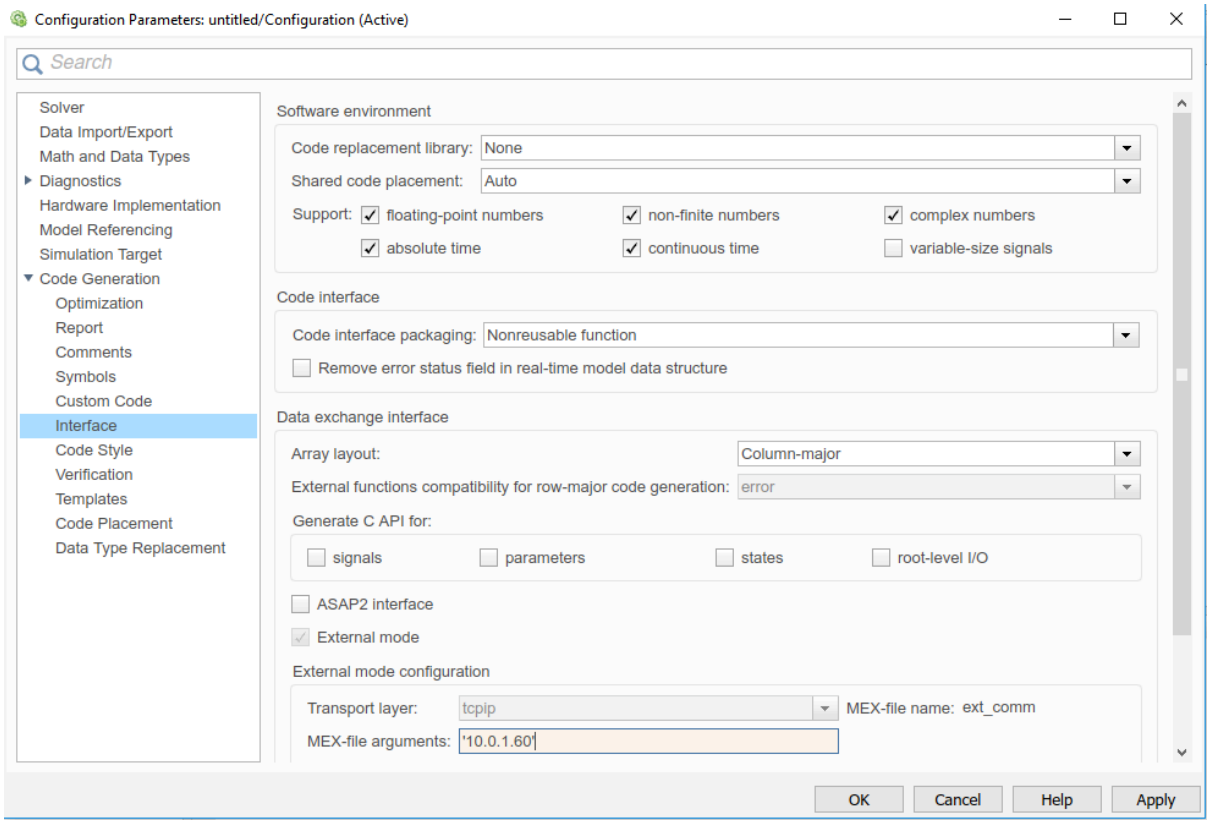
2. Open the model configuration parameters.



3. Select “Code Generation”, press the “Browse” button and select “ert_rpi.tlc”, then “OK”.



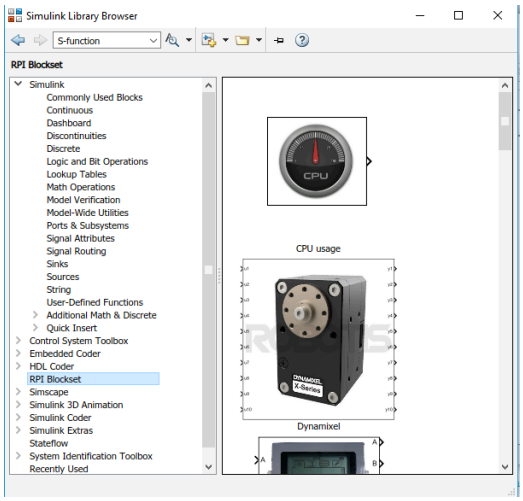
4. Expand “Code Generation” and select “Interface”.



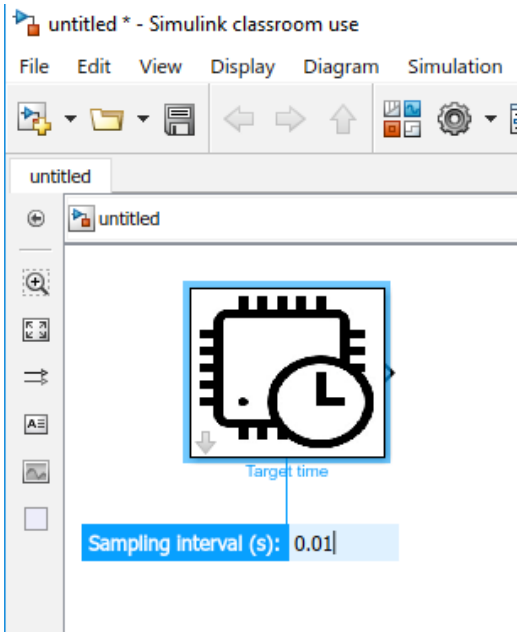
5. In the “MEX-file arguments:” field enter the IP address of the target surrounded by quotes like ‘xxx.yyy.zzz.aaa’. Then click “OK”. This procedure auto-configures the model for using it with RPIt.

6. Open the library browser.

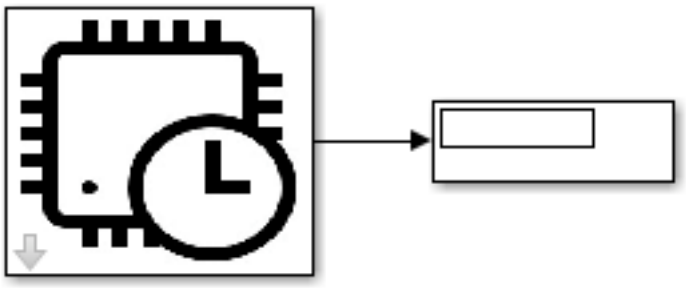
7. Select the RPI blockset.



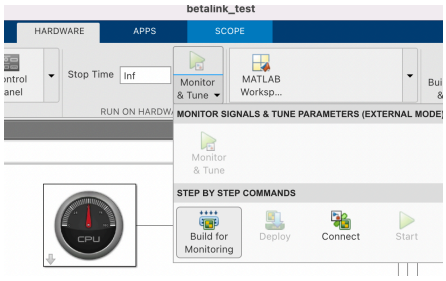
8. Drag and drop the “Target time” bloc and enter 0.01 as Sampling interval.



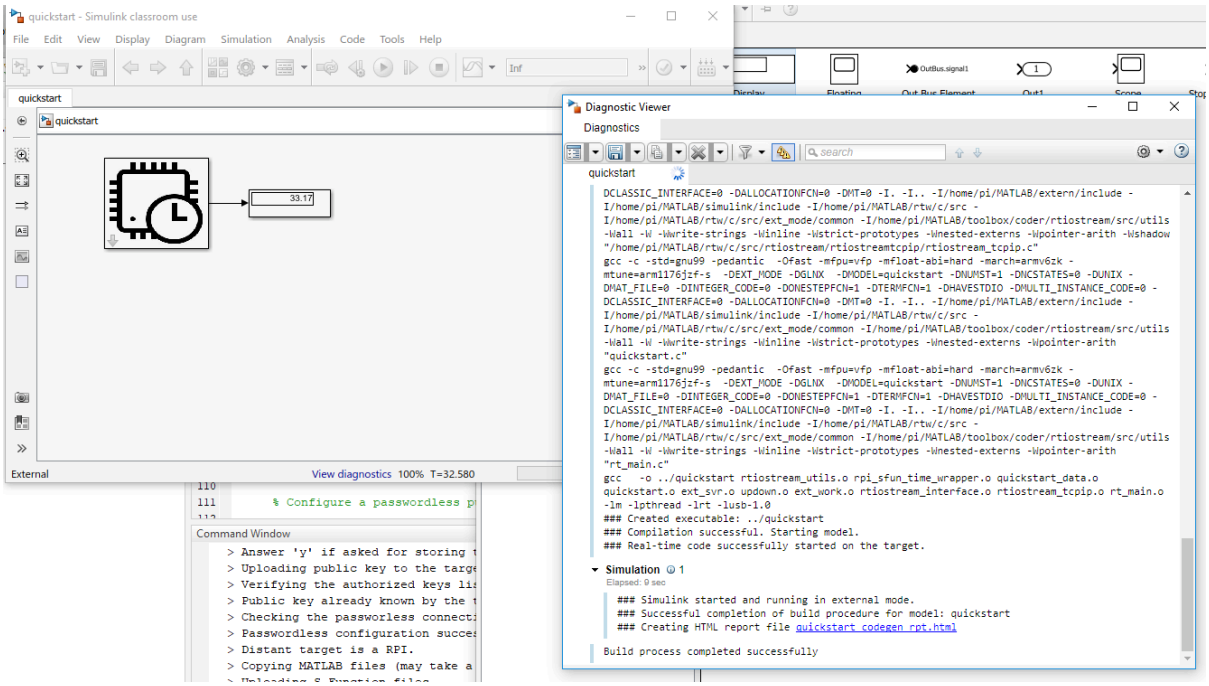
9. In the “Sinks” section, drag and drop the “Display” block and connect its input to the “Target time” output.



10. Save the model and push the “Build model” button.



11. A approx. 1 minute procedure starts yielding the following output. At the end, the HIL experiment is running and the “Display” block prints the target time which should be approximately equal to the simulation time.



12. Pressing the “Stop” button stops the experiment.

13. The full output of the “Build” action can be obtained by clicking on “Diagnostic viewer”:

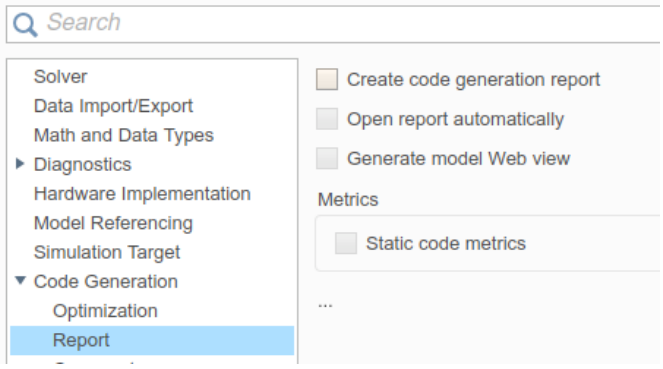
```
### Starting build procedure for model: quickstart
### Generating code and artifacts to 'Model specific' folder structure
 Code Generation  3
Elapsed: 1:04 min
### Generating code into build folder: C:
\Users\jacques\Documents\MATLAB\Quickstart\quickstart_ert_rtw

### Invoking Target Language Compiler on quickstart.rtw
### Using System Target File: Z:
\Documents\Recherche\Developpement\github\rpit\rpit\ert_rpi.tlc

### Loading TLC function libraries
### Initial pass through model to cache user defined code
### Caching model source code
### Writing header file quickstart_types.h
### Writing header file quickstart.h
### Writing header file rtwtypes.h
.
### Writing header file multiword_types.h
### Writing source file quickstart.c
### Writing header file quickstart_private.h
### Writing source file quickstart_data.c
### Writing header file rtmodel.h
.
```

```
### Writing source file rt_main.c

### TLC code generation complete.

### Generating TLC interface API.

.

### Creating data type transition file quickstart_dt.h

.### Evaluating PostCodeGenCommand specified in the model

.

### Processing Template Makefile: Z:
\Documents\Recherche\Developpement\github\rpit\rpit\ert_rpi.tmf

### Wrapping unrecognized make command (angle brackets added)

###    <make>

### in default batch file

### Creating quickstart.mk from Z:
\Documents\Recherche\Developpement\github\rpit\rpit\ert_rpi.tmf

### Make will not be invoked - BUILD macro in template makefile is set to 'no'

### Target IP address: 10.0.1.60

### Target is a Raspberry Pi.

### Synchronizing host and target clocks.

### Kill any pending rtw process.

### Uploading quickstart to the target.

### Compiling quickstart on the target (may take awhile).

gcc -c -std=gnu99 -pedantic  -Ofast -mfpu=vfp -mfloat-abi=hard -march=armv6zk
-mtune=arm1176jzf-s  -DEXT_MODE -DGLNX  -DMODEL=quickstart -DNUMST=1 -DNCSTATES=0 -DUNIX
-DMAT_FILE=0 -DINTEGER_CODE=0 -DONESTEPFCN=1 -DTERMFCN=1 -DHAVESTDIO -DMULTI_INSTANCE_CODE=0
-DCLASSIC_INTERFACE=0 -DALLOCATIONFCN=0 -DMT=0 -I. -I.. -I/home/pi/MATLAB/extern/include -I/
home/pi/MATLAB/simulink/include -I/home/pi/MATLAB/rtw/c/src -I/home/pi/MATLAB/rtw/c/src/
ext_mode/common -I/home/pi/MATLAB/toolbox/coder/rtiostream/src/utils     -Wall -W -Wwrite-
strings -Winline -Wstrict-prototypes -Wnested-externs -Wpointer-arith -Wshadow "/home/pi/
MATLAB/toolbox/coder/rtiostream/src/utils/rtiostream_utils.c"


gcc -c -std=gnu99 -pedantic  -Ofast -mfpu=vfp -mfloat-abi=hard -march=armv6zk
-mtune=arm1176jzf-s  -DEXT_MODE -DGLNX  -DMODEL=quickstart -DNUMST=1 -DNCSTATES=0 -DUNIX
-DMAT_FILE=0 -DINTEGER_CODE=0 -DONESTEPFCN=1 -DTERMFCN=1 -DHAVESTDIO -DMULTI_INSTANCE_CODE=0
-DCLASSIC_INTERFACE=0 -DALLOCATIONFCN=0 -DMT=0 -I. -I.. -I/home/pi/MATLAB/extern/include -I/
home/pi/MATLAB/simulink/include -I/home/pi/MATLAB/rtw/c/src -I/home/pi/MATLAB/rtw/c/src/
ext_mode/common -I/home/pi/MATLAB/toolbox/coder/rtiostream/src/utils     -Wall -W -Wwrite-
strings -Winline -Wstrict-prototypes -Wnested-externs -Wpointer-arith -Wshadow "/home/pi/
MATLAB/blocks/rpi_sfun_time_wrapper.c"


gcc -c -std=gnu99 -pedantic  -Ofast -mfpu=vfp -mfloat-abi=hard -march=armv6zk
-mtune=arm1176jzf-s  -DEXT_MODE -DGLNX  -DMODEL=quickstart -DNUMST=1 -DNCSTATES=0 -DUNIX
-DMAT_FILE=0 -DINTEGER_CODE=0 -DONESTEPFCN=1 -DTERMFCN=1 -DHAVESTDIO -DMULTI_INSTANCE_CODE=0
-DCLASSIC_INTERFACE=0 -DALLOCATIONFCN=0 -DMT=0 -I. -I.. -I/home/pi/MATLAB/extern/include -I/
home/pi/MATLAB/simulink/include -I/home/pi/MATLAB/rtw/c/src -I/home/pi/MATLAB/rtw/c/src/
ext_mode/common -I/home/pi/MATLAB/toolbox/coder/rtiostream/src/utils     -Wall -W -Wwrite-
strings -Winline -Wstrict-prototypes -Wnested-externs -Wpointer-arith -Wshadow "/home/pi/
MATLAB/rtw/c/src/ext_mode/common/ext_svr.c"


gcc -c -std=gnu99 -pedantic  -Ofast -mfpu=vfp -mfloat-abi=hard -march=armv6zk
-mtune=arm1176jzf-s  -DEXT_MODE -DGLNX  -DMODEL=quickstart -DNUMST=1 -DNCSTATES=0 -DUNIX
-DMAT_FILE=0 -DINTEGER_CODE=0 -DONESTEPFCN=1 -DTERMFCN=1 -DHAVESTDIO -DMULTI_INSTANCE_CODE=0
-DCLASSIC_INTERFACE=0 -DALLOCATIONFCN=0 -DMT=0 -I. -I.. -I/home/pi/MATLAB/extern/include -I/
home/pi/MATLAB/simulink/include -I/home/pi/MATLAB/rtw/c/src -I/home/pi/MATLAB/rtw/c/src/
ext_mode/common -I/home/pi/MATLAB/toolbox/coder/rtiostream/src/utils     -Wall -W -Wwrite-
strings -Winline -Wstrict-prototypes -Wnested-externs -Wpointer-arith "quickstart_data.c"


gcc -c -std=gnu99 -pedantic  -Ofast -mfpu=vfp -mfloat-abi=hard -march=armv6zk
-mtune=arm1176jzf-s  -DEXT_MODE -DGLNX  -DMODEL=quickstart -DNUMST=1 -DNCSTATES=0 -DUNIX
-DMAT_FILE=0 -DINTEGER_CODE=0 -DONESTEPFCN=1 -DTERMFCN=1 -DHAVESTDIO -DMULTI_INSTANCE_CODE=0
-DCLASSIC_INTERFACE=0 -DALLOCATIONFCN=0 -DMT=0 -I. -I.. -I/home/pi/MATLAB/extern/include -I/
home/pi/MATLAB/simulink/include -I/home/pi/MATLAB/rtw/c/src -I/home/pi/MATLAB/rtw/c/src/
ext_mode/common -I/home/pi/MATLAB/toolbox/coder/rtiostream/src/utils     -Wall -W -Wwrite-
strings -Winline -Wstrict-prototypes -Wnested-externs -Wpointer-arith -Wshadow "/home/pi/
MATLAB/rtw/c/src/ext_mode/common/updown.c"


gcc -c -std=gnu99 -pedantic  -Ofast -mfpu=vfp -mfloat-abi=hard -march=armv6zk
-mtune=arm1176jzf-s  -DEXT_MODE -DGLNX  -DMODEL=quickstart -DNUMST=1 -DNCSTATES=0 -DUNIX
-DMAT_FILE=0 -DINTEGER_CODE=0 -DONESTEPFCN=1 -DTERMFCN=1 -DHAVESTDIO -DMULTI_INSTANCE_CODE=0
-DCLASSIC_INTERFACE=0 -DALLOCATIONFCN=0 -DMT=0 -I. -I.. -I/home/pi/MATLAB/extern/include -I/
home/pi/MATLAB/simulink/include -I/home/pi/MATLAB/rtw/c/src -I/home/pi/MATLAB/rtw/c/src/
ext_mode/common -I/home/pi/MATLAB/toolbox/coder/rtiostream/src/utils     -Wall -W -Wwrite-
strings -Winline -Wstrict-prototypes -Wnested-externs -Wpointer-arith -Wshadow "/home/pi/
MATLAB/rtw/c/src/ext_mode/common/ext_work.c"


gcc -c -std=gnu99 -pedantic  -Ofast -mfpu=vfp -mfloat-abi=hard -march=armv6zk
-mtune=arm1176jzf-s  -DEXT_MODE -DGLNX  -DMODEL=quickstart -DNUMST=1 -DNCSTATES=0 -DUNIX
```

```
-DMAT_FILE=0 -DINTEGER_CODE=0 -DONESTEPFCN=1 -DTERMFCN=1 -DHAVESTDIO -DMULTI_INSTANCE_CODE=0
-DCLASSIC_INTERFACE=0 -DALLOCATIONFCN=0 -DMT=0 -I. -I.. -I/home/pi/MATLAB/extern/include -I/
home/pi/MATLAB/simulink/include -I/home/pi/MATLAB/rtw/c/src -I/home/pi/MATLAB/rtw/c/src/
ext_mode/common -I/home/pi/MATLAB/toolbox/coder/rtiostream/src/utils     -Wall -W -Wwrite-
strings -Winline -Wstrict-prototypes -Wnested-externs -Wpointer-arith -Wshadow "/home/pi/
MATLAB/rtw/c/src/ext_mode/common/rtiostream_interface.c"

gcc -c -std=gnu99 -pedantic  -Ofast -mfpu=vfp -mfloat-abi=hard -march=armv6zk
-mtune=arm1176jzf-s  -DEXT_MODE -DGLNX  -DMODEL=quickstart -DNUMST=1 -DNCSTATES=0 -DUNIX
-DMAT_FILE=0 -DINTEGER_CODE=0 -DONESTEPFCN=1 -DTERMFCN=1 -DHAVESTDIO -DMULTI_INSTANCE_CODE=0
-DCLASSIC_INTERFACE=0 -DALLOCATIONFCN=0 -DMT=0 -I. -I.. -I/home/pi/MATLAB/extern/include -I/
home/pi/MATLAB/simulink/include -I/home/pi/MATLAB/rtw/c/src -I/home/pi/MATLAB/rtw/c/src/
ext_mode/common -I/home/pi/MATLAB/toolbox/coder/rtiostream/src/utils     -Wall -W -Wwrite-
strings -Winline -Wstrict-prototypes -Wnested-externs -Wpointer-arith -Wshadow "/home/pi/
MATLAB/rtw/c/src/rtiostream/rtiostreamtcpip/rtiostream_tcpip.c"

gcc -c -std=gnu99 -pedantic  -Ofast -mfpu=vfp -mfloat-abi=hard -march=armv6zk
-mtune=arm1176jzf-s  -DEXT_MODE -DGLNX  -DMODEL=quickstart -DNUMST=1 -DNCSTATES=0 -DUNIX
-DMAT_FILE=0 -DINTEGER_CODE=0 -DONESTEPFCN=1 -DTERMFCN=1 -DHAVESTDIO -DMULTI_INSTANCE_CODE=0
-DCLASSIC_INTERFACE=0 -DALLOCATIONFCN=0 -DMT=0 -I. -I.. -I/home/pi/MATLAB/extern/include -I/
home/pi/MATLAB/simulink/include -I/home/pi/MATLAB/rtw/c/src -I/home/pi/MATLAB/rtw/c/src/
ext_mode/common -I/home/pi/MATLAB/toolbox/coder/rtiostream/src/utils     -Wall -W -Wwrite-
strings -Winline -Wstrict-prototypes -Wnested-externs -Wpointer-arith "quickstart.c"

gcc -c -std=gnu99 -pedantic  -Ofast -mfpu=vfp -mfloat-abi=hard -march=armv6zk
-mtune=arm1176jzf-s  -DEXT_MODE -DGLNX  -DMODEL=quickstart -DNUMST=1 -DNCSTATES=0 -DUNIX
-DMAT_FILE=0 -DINTEGER_CODE=0 -DONESTEPFCN=1 -DTERMFCN=1 -DHAVESTDIO -DMULTI_INSTANCE_CODE=0
-DCLASSIC_INTERFACE=0 -DALLOCATIONFCN=0 -DMT=0 -I. -I.. -I/home/pi/MATLAB/extern/include -I/
home/pi/MATLAB/simulink/include -I/home/pi/MATLAB/rtw/c/src -I/home/pi/MATLAB/rtw/c/src/
ext_mode/common -I/home/pi/MATLAB/toolbox/coder/rtiostream/src/utils     -Wall -W -Wwrite-
strings -Winline -Wstrict-prototypes -Wnested-externs -Wpointer-arith "rt_main.c"

gcc   -o ../quickstart rtiostream_utils.o rpi_sfun_time_wrapper.o quickstart_data.o
quickstart.o ext_svr.o updown.o ext_work.o rtiostream_interface.o rtiostream_tcpip.o
rt_main.o     -lm -lpthread -lrt -lusb-1.0

### Created executable: ../quickstart
### Compilation successful. Starting model.
### Real-time code successfully started on the target.
 Simulation  1
Elapsed: 9 sec
Build process completed successfully
```

Note

You may want to disable the pop-up of the code generation report by unchecking these checkboxes:



Tips for Model Debugging

If there is an error on the target side, there is no easy means to be alerted within Simulink. When something goes wrong, it is possible to manually launch the target executable and see the debug messages in the Linux console. It is even possible to manually recompile the target in order to troubleshoot bugs.

Manual Launch of the Target Executable

1. Start a ssh session with the target (here the IP address of the target is 10.0.1.60):

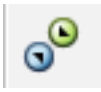
```
Connection to 10.0.1.60 closed.
MacBook-Pro-de-Jacques-6:host jacques$ ssh pi@10.0.1.60
pi@10.0.1.60's password:
Linux dextair 4.19.42-v7+ #1219 SMP Tue May 14 21:20:58 BST 2019 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Jun 28 15:57:34 2019 from 10.0.1.49
pi@dextair:~ $ cd RTW
pi@dextair:~/RTW $ ls
polaris.rom         quickstart.slx  slprj                    teensyshot_test_1.slx    teensyshot_test.slx
quickstart_ert_rtw  rpi             teensyshot_test_1_ert_rtw  teensyshot_test_ert_rtw  teensyshot_test.slx.autosave
pi@dextair:~/RTW $
```

2. Go into the “RTW” directory. There is a “rpi” executable. This is the last executable built by RPIt when pressing the “Build” button in the Simulink model window. Usually this executable in launched automatically by the “Build” procedure, but you can also launch it manually. To do that, enter the following command:

3. The target executable is now waiting for a valid socket connection coming from

```
pi@dextair:~/RTW $ ./rpi -tf inf -w
```

   Simulink. In Simulink, press the “Connect To Target” button.



4. If the connection was successful, the appearance of the button should change.

5. You can now press the “Run” button, and the experiment should start.

6. In the target ssh terminal, you should see at least the following message:

```
pi@dextair:~/RTW $ ./rpi -tf inf -w

** starting the model **
```

7. If there are additional meaningful errors, they should appear here. You can, of course, add your own debug message directly in the target code. This is explained in the next section.

Compiling the Target Program Manually

You can modify the target code and recompile it manually. In the ssh terminal (see previous section), go into "~/RTW/myname_ert_rtw" where "myname" is your model name. It the following screenshot, you can see all the files generated by the "quickstart" model described in the beginning of this tutorial.

```
pi@dextair:~/RTW $ ls
polaris.rom          quickstart.slx  slprj                      teensyshot_test_1.slx    teensyshot_test.slx
quickstart_ert_rtw   rpi             teensyshot_test_1_ert_rtw  teensyshot_test_ert_rtw  teensyshot_test.slx.autosave
pi@dextair:~/RTW $ cd quickstart_ert_rtw/
pi@dextair:~/RTW/quickstart_ert_rtw $ ls
buildInfo.mat        ext_svr.o          quickstart.bat     quickstart.h               quickstart_types.h       rt_main.c           rtwtypes.h
codedescriptor.dmr   ext_work.o         quickstart.c       quickstart.mk              rpi_sfun_time_wrapper.o  rt_main.o           updown.o
codeInfo.mat         html               quickstart_data.c  quickstart.o               rtiostream_interface.o   rtmodel.h
compileInfo.mat      modelsources.txt   quickstart_data.o  quickstart_private.h       rtiostream_tcpip.o       rtw_proj.tmw
defines.txt          multiword_types.h  quickstart_dt.h    quickstart_targ_data_map.m rtiostream_utils.o       rtwtypeschksum.mat
pi@dextair:~/RTW/quickstart_ert_rtw $
```

Simply recompile the target executable by entering the following command (here myname=quickstart):

```
pi@dextair:~/RTW/quickstart_ert_rtw $ make -f quickstart.mk
gcc   -o ../quickstart rtiostream_utils.o rpi_sfun_time_wrapper.o quickstart_data.o quickstart.o ext_svr.o updown.o ext_work.o rtiostream_interface.o rtiostream
_tcpip.o rt_main.o     -lm -lpthread -lrt -lusb-1.0
### Created executable: ../quickstart
pi@dextair:~/RTW/quickstart_ert_rtw $
```

The executable name is "myname" (here "quickstart"). You can test this program with Simulink using the same procedure described in the previous section:

```
pi@dextair:~/RTW/quickstart_ert_rtw $ ../quickstart -tf inf -w
```