TMW_logo_blue

Development of a Simple RTOS Target

Authors: Roger Theyyunni

Created On: Jan 2012

Document Version: **1.0**

Table of Contents

[1 Introduction 3](#_Toc315855734)

[1.1 Supported Features 3](#_Toc315855735)

[1.2 Non supported features 3](#_Toc315855736)

[2 Steps for Building a Simple Target with OS support 3](#_Toc315855737)

[2.1 Supporting ExternAL Mode 5](#_Toc315855738)

[3 Target ARCHITECTURE 6](#_Toc315855739)

[4 Workflow 7](#_Toc315855740)

# Introduction

This document describes how to develop an Embedded Coder Target executing under a Real Time Operating System. An example target has been developed for a QNX RTOS target for PowerPC processors. This example is for QNX, but only supports PowerPC because the compiler tools used are for PowerPC.

Most of the files listed in this document are derived from some file in the shipping release. To see the exact changes made, you can do a diff of the target file and the shipping file. For example, qnxertmainlib.tlc is a modification of matlabroot\rtw\c\tcl\mw\ ertmainlib.tlc in the shipping project.

## Supported Features

* Automatic build of QNX executable from a Simulink model using “tmf” based approach
* Supports single tasking and multi-tasking modes of operation. Multi rate models are supported
* The generated executable is downloaded onto the QNX Target using “Momentics IDE”
* External mode support is provided so that once the executable from the model has been loaded and started by Momentics IDE, parameters can be tuned from the Simulink model and data monitored on scopes
* The target supports referenced models, but external mode has not been tested on referenced models

## UNSupported Features

* No I/O blocks are supported
* Re-runs are not supported. After each run, the model has to be reloaded and restarted
* A simple Posix timer has been used for timing, a more deterministic timer might be more meaningful for a more robust QNX Target

# Supported MATLAB Releases

R2011a

# Steps for Building a Simple Target with OS support

The steps used to build a simple target based on an RTOS are the following. Note that there are several methods to build a target; the method described here is one simple approach.

* Create a target directory for example:

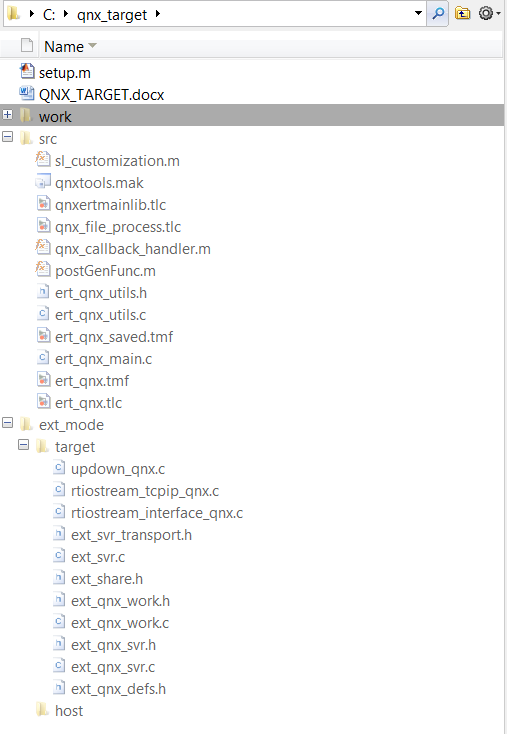


Figure 1, Target Directory strucutre

* Create a system target file - <ert>\_<tgtos>.tlc file. Add any configuration dialog or call back handler. **Example**: ert\_qnx.tlc
* Create a <tgt>tools.mak file with tools specification. Example erttools.mak
* Create an ert\_<tgtos>.tmf file by modifying ert\_lcc.tmf file. Add any additional tokens that you might need. Include the <tgtos>.mak file in the tmf file instead of the default file - $(MATLAB\_ROOT)\rtw\c\tools\lcctools.mak. **Example**: ert\_qnx.tlc
* In the ert\_<tgtos>.tmf add a line to compile files in the target specific files – for example:
  + %.o: $(QNX\_MW\_ROOT)/ext\_mode/src/%.c
    - $(CC) -c -o $(@F) $(CFLAGS) $<
* In the ert\_<tgtos>.tmf, modify EXT\_SRC macro to include files to support external mode. For example:
  + ext\_qnx\_svr.c, updown\_qnx\_svr.c, rtiostream\_tcpip\_qnx.c, rtiostream\_interface\_qnx.c
* In the ert\_<tgtos>.tmf file, add a line to compile external mode files. For example:
  + %.o : $(QNX\_MW\_ROOT)/ext\_mode/target/%.c
    - $(CC) -c -o $(@F) $(CFLAGS) $<

Customize ert\_main creation by modifying ertmainlib.tlc. Rename ertmainlib.tlc to <tgt>ertmainlib.tlc. Modify TLC functions FcnGenerateMultitaskingOSCode() and FcnGenerateSingletaskingOSCode(). Modify these functions to:

* Create a semaphore for each rate.
* Create thread for each rate with appropriate priority – higher priority for faster rate. Each thread waits on the semaphore for that rate. The fastest thread waits on a semaphore or message from a timer interrupt
* Install timer interrupt, and associate it with a semaphore

**Example**: qnxertmainlib.tlc

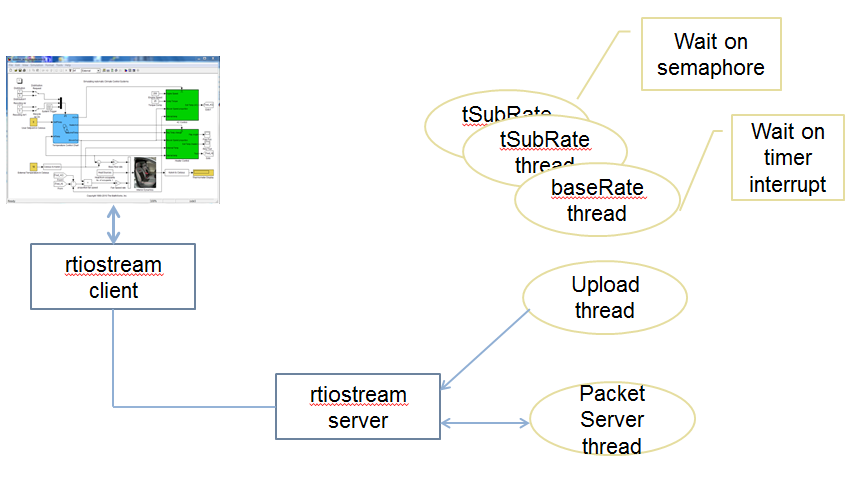
* Create <tgt>\_file\_process.tlc derived from example\_file\_process.tlc. In the file uncomment the statement to set ERTCustomFileTest to TLC\_TRUE. Include <tgt>ertmainlib.tlc, as shown in qnx\_file\_process.tlc

## Supporting ExternAL Mode

To support external mode, create the files shown in qnx\_target\ext\_mode\target, derived from the files in matlabroot\rtw\c\src\ext\_mode\common. Modify these files to include your RTOS files. In ext\_<tgt>\_work.c create the thread for upload server and the packet server as shown in the example file ext\_qnx\_work.c. Note that this file is derived from ext\_work.c and the call to create the tasks, taskspawn, has been replaced by calls to QNX threads. Also, modify the calls to the RTOS calls for semaphore creation – use your OS’s API.

To support external mode, two threads are created – upload thread and packet server thread. The upload thread is responsible for transferring data from the Simulink executable in the target to the scopes in the Simulink model running on the host. The packet server thread is used to receive control commands from the Simulink on the host. One of the bugs detected in the shipping software was that the packet server should not be preempted by the upload server, because that might result in the scope data being sent as a response to a request to the packet server. To overcome this bug, the priority of the packet server has been made one higher than the priority of the upload server thread.

# Target ARCHITECTURE



# WorkFLOW

