Embedded Target for Motorola MPC555 Release Notes

See the “Embedded Target for Motorola MPC555 1.0.1 Release Notes” on page 1-1 for an overview of this new product and a list of known software and documentation problems.The following topics are discussed in these Release Notes:

Printing the Release Notes

If you would like to print the Release Notes, you can link to a PDF version.

Contents

1 Embedded Target for Motorola MPC555 1.0.1 Release Notes

1

Embedded Target for Motorola MPC555 1.0.1 Release Notes

Introduction to the Embedded Target for Motorola MPC555

Note The Embedded Target for Motorola MPC555 1.0.1 is the first release of this product as part of a MathWorks release CD. Version 1.0 of this product was initially released in Web-downloadable form after Release 12.1 was released, but before Release 13.

The Embedded Target for Motorola MPC555 is an add-on product for use with the Real-Time Workshop Embedded Coder. It provides a complete and unified set of tools for developing embedded applications for the Motorola MPC555 processor.

Used in conjunction with Simulink, Stateflow, and the Real-Time Workshop Embedded Coder, the Embedded Target for Motorola MPC555 lets you

- Design and model your system and algorithms.
- Compile, download, run and debug generated code on the target hardware, seamlessly integrating with industry-standard compilers and development tools for the MPC555.
- Use co-simulation and rapid prototyping techniques to evaluate performance and validate results obtained from generated code running on the target hardware.
- Deploy production code on the target hardware.

Feature Summary

Simulation and Co-Simulation

- Automatic S-function generation lets you validate your generated code in software-in-the-loop (SIL) simulation.
- Processor-in-the-loop (PIL) co-simulation lets you integrate generated code, running on target processor, into your simulation.
- SIL and PIL code components are generated by the Real-Time Workshop Embedded Coder. These simulation components are in the same compact and efficient format as the production code generated for final deployment.

Production Code Generation

- Generates production code for use on the target MPC555 microcontroller.
- Generates project or make files for popular cross-development systems:
  - Diab C cross-compiler
  - Metrowerks CodeWarrior
- Debugger support:
  - Wind River Systems Single Step debugger
  - Metrowerks CodeWarrior debugger

Code and Performance Analysis

Web-viewable code generation report includes:

- Analysis of RAM/ROM usage and other variables.
- Analysis of code generation options used, with optimization suggestions.
- Hyperlinks to all generated code files.
- Hyperlinks from generated code to source model in Simulink.

Applications Development and Rapid Prototyping

- Generation of real-time, standalone code for MPC555
- Scheduler and time functions for single- rate or multirate real-time operation
- CAN-based loader for download of generated code to RAM or flash memory
- CAN-based host-target communications for non-real-time retrieval of data on host computer

Device Driver Support

- The Embedded Target for Motorola MPC555 Library provides device driver blocks that let your applications access on-chip resources. The I/O blocks support the following features of the MPC555:
  - Pulse width modulation (PWM) generation via the Modular Input/Output Subsystem (MIOS) PWM unit.
  - Analog input via the Queued Analog-to-Digital Converter (QADC64).
  - Digital input and output via the MIOS.

  - Digital input via the QADC64.
  - Frequency and pulse width measurement via the MIOS Double Action Submodule (MDASM).
  - Transmit or receive Controller Area Network (CAN) messages via the MPC555 TouCAN modules.
  - Utility blocks such as a watchdog timer.
- Device driver blocks support a *pass - through* option. The pass-through option lets you leave your device driver blocks in your model during simulation. You can then provide a Simulink signal to use in place of the actual device driver signal.

CAN Support

- Transmit or receive CAN messages via the MPC555 TouCAN modules.
- CAN Drivers (Vector) library provides blocks for configuring and connecting to Vector-Informatik CAN hardware and drivers.
- The CAN Message Blocks library includes blocks for transmitting, receiving, decoding, and formatting CAN messages. The CAN Message Blocks library also supports message specification via the Vector-Informatik CANdb standard.

Code Validation and Performance Analysis

Code Validation. Since signal data is available to Simulink during each sample interval in a PIL simulation, you can observe signal data on Scopes or other Simulink signal viewing blocks. You can also store signal data to MAT-files via To File blocks. To validate the results obtained by the generated code running on the target processor, you can compare these files to results obtained using a normal Simulink plant/controller simulation.

Determining Code Size. In control design it is critical to ensure that the size of the generated code does not exceed physical limitations of RAM and ROM. The Embedded Target for Motorola MPC555 automatically produces a code generation report that displays the RAM usage and ROM size of the generated code.

This capability is useful when selecting which code generation optimizations will be used. After determining the size of the required RAM and ROM, you can

consider which code generation optimizations to use, and consider modifications to the model design.

Known Software and Documentation Problems

This section describes known software and documentation problems in Version 1.0.1.

Accelerator Mode

Models that contain CAN blocks from the Embedded Target for Motorola MPC555 libraries do not operate properly in Simulink Accelerator mode.

CAN Blocks and CAN Download Control Panel

- The blocks in the CAN Drivers (Vector) blockset, and the `candnload` utility, require correct installation of a CAN card and drivers from Vector-Informatik.

  In addition, after installing the drivers from Vector-Informatik, you must locate the `vcand32.dll` library, and add the path to the location where `vcand32.dll` was installed to the Windows system path.

  If the system path is not updated correctly, errors in the use of the CAN Drivers (Vector) blockset and the `candnload` utility occur.

- This note describes a limitation in the Vector CAN Receive block that is caused by a problem in the Vector CAN drivers. Please contact Vector-Informatik technical support regarding the availability of a solution.

  When a Vector CAN Receive block is configured for a virtual channel, the block accepts *any* Extended Identifier, regardless of the specified CAN message identifier and CAN message type. This limitation does not apply to hardware channels, which function correctly.

- Due to a limitation of the Vector Informatik CAN hardware/software package, a maximum of 30 Vector CAN Receive Blocks, on a given channel, can be included in a model.

  Note that this limitation does not apply to Vector CAN Transmit Blocks. An unlimited number of Vector CAN Transmit Blocks can be included in a model.

- During a download using the CAN Download Control Panel with the Embedded Target for Motorola MPC555 real-time target, you may see the following message in the MATLAB command window:

```
Message not sent. Tx queue is full. Attach a device to the network
to flush queue.
```

The likely cause of this is that the TouCAN module on the target device you are connected to is offline. If you are using a PCI host PC CAN card then the solution is to reset the target device, which will then execute the boot code. The PC CAN transmit queue is then flushed and the download proceeds.

If however, you are using a CANCardX in a laptop, there may be some differences in the way the CAN transmit queue is flushed. In this case the download may fail. To work around this limitation, make sure there is always at least one enabled CAN device on the CAN bus. One way to do this is to use a Y-cable to connect both CAN ports on your PC together with the CAN port on the target.

If you are using the second CAN port on the PC in this way, you must make sure that it is initialized. To do this, use the `btest32` utility supplied with the Vector Informatik driver software. You can invoke the `btest32` utility from the PC command prompt. The following example runs `btest32` with a baud rate of 500000:

```
btest32 500000
```

CAN Calibration Protocol Block Incompatibility with lcc C Compiler

Due to a current Stateflow limitation, it is not possible to use the CAN Calibration Protocol Block if you use the lcc C Compiler as the default compiler (as designated by the `mex -setup` command). "Undefined reference" errors will occur when the model is updated if you attempt to use this block with lcc.

The CAN Calibration Protocol Block is fully compatible with the Microsoft Visual C/C++ compiler. To use the CAN Calibration Protocol Block, you must install Microsoft Visual C/C++ and associated environment variables as described in the “Installing Real-Time Workshop” section of the Real-Time Workshop documentation) and set Microsoft Visual C/C++ to be the default compiler via the `mex -setup` command.

This limitation will be removed in a future Stateflow release.

Compiler Optimizations

In some very rare instances, due to compiler defects, compiler optimizations applied to Embedded Target for Motorola MPC555 generated code may cause the executable program to produce incorrect results, even though the code itself is correct.

When using the RT target, you can set the **Optimize compiler for** option to **none** to work around problems caused by compiler optimizations.

For the PIL and AE targets, please refer to your compiler's documentation for information on how to lower the optimization level of the compiler or turn off optimizations. Then, having found the optimization switches required, you can enter the options directly into the **Make command** field in the **Target configuration** category of the Real-Time Workshop pane of the **Simulation Parameters** dialog.

By default, the **Make command** field contains the command

```
make_rtw
```

To add your optimization switches, append the following to the make_rtw command:

```
MPC555_OPTIM_FLAGS=<Your optimization switches>
```

For example, if the correct switch for your compiler is `-O`, the **Make command** field would contain

```
make_rtw MPC555_OPTIM_FLAGS=-O
```

Configuration for Non-Default Hardware

The Embedded Target for Motorola MPC555 has been developed and fully tested using Phytec phyCORE-MPC555 development board. We strongly recommend the use of this board for getting started with the Embedded Target for Motorola MPC555.

If you are using different MPC555 hardware, it may be necessary to perform some additional manual configuration. The section “Hardware Clock Configuration” on page 1-12 describes how to configure the Embedded Target for Motorola MPC555 real-time target for use on hardware with external oscillator frequencies other than 20 MHz.

The following sections give additional examples of hardware-specific configuration changes that may be required. This information is not intended to be exhaustive.

Using the PIL Target With Non-Default Hardware

If you are using the Metrowerks CodeWarrior development environment, the relevant hardware configuration settings are contained in

```
matlabroot/toolbox/rtw/targets/mpc555dk/pil/BSPs/phyCORE-555/
src/phyCORE-MPC555_BDM_relocate_init.cfg
```

If you are using the Diab and SingleStep development environment, the configuration settings are contained in

```
matlabroot/toolbox/rtw/targets/mpc555dk/pil/
singlestep_script_template.tlc.
```

The necessary changes to these files depend on the hardware that you are using. As an example, the following settings should be made for an Axiom CME-555 development board:

- Change the value for `BR1` from `0xFFF00001` to `0xFFF00003` (CodeWarrior)
- Change the value for `BR1` from `0x00000001` to `0x00000003` (SingleStep)
- Change the value for `OR1` from `0xFFE00000` to `0xFFF80002`
- Change the value for `SCCR` from `0x00000000` to `0x00010000`
- Change the value for `PLPRCR` from `0x00000000` to `0x00400000`

Note that these values are given only as an example.

Depending on your hardware, you may also need to configure switches and jumper settings. Consult the documentation for your development board.

Using the RT Target With Non-Default Hardware

As with the PIL target, the exact changes required for the real-time target will depend on your hardware. You should make the changes to the values for `BR1`, `OR1`, `SCCR` and `PLPRCR` by modifying each of the files

```
matlabroot/toolbox/rtw/targets/mpc555dk/common/drivers/
app_startup/mw_ram.wsp
```

matlabroot/toolbox/rtw/targets/mpc555dk/common/drivers/app_startup/mw_flash.wsp

matlabroot/toolbox/rtw/targets/mpc555dk/common/drivers/bootcode/bootloader.h

Note that after making any changes to `bootloader.h`, you must re-compile the boot code as described in "Boot Code Parameters for CAN Download" in the *Embedded Target for Motorola MPC555 User's Guide*.

Default Changed for CCP Block Total Number of Object Descriptor Tables

The default value for the **Total Number of Object Descriptor Tables (ODTs)** parameter of the CCP block has been changed to `8`. Formerly, the default value was `254`.

Device Driver Blocks Support Limitations

The Embedded Target for Motorola MPC555 real-time (RT) target supports all blocks in the Embedded Target for Motorola MPC555 libraries.

The Algorithm Export (AE) and PIL targets and Simulink Accelerator also provide limited support for blocks in the Embedded Target for Motorola MPC555 libraries. These limitations will be removed in a future release.

Table 1-1 and accompanying notes summarize the blocks that are supported by each of these targets.

Table 1-1: Device Driver Blocks Support

Block or Block Group	Supported by RT Target	Supported by PIL Target	Supported by AE Target	Supported by Simulink Accelerator
MPC555 Resource Configuration	Yes	Yes	Yes	Yes
Watchdog	Yes	Yes	Yes	Yes

Table 1-1: Device Driver Blocks Support (Continued)

Block or Block Group	Supported by RT Target	Supported by PIL Target	Supported by AE Target	Supported by Simulink Accelerator
Modular Input/ Output System (MIOS1) block group	Yes	Yes	Yes	Yes
Queued Analog-Digital Converter Module-64 block group	Yes (see Note 1)	Yes	Yes (see Note 1)	No
CAN 2.0B Controller Module (TouCAN) block group	Yes	Yes (see Note 2)	Yes (seeNote 3)	No
CAN Message block group	Yes	Yes	Yes	No
CAN Drivers (Vector) block group	Yes	Yes	Yes	No

Note 1. When using the QADC Analog In block, or the TOUCAN Transmit make sure that the block output is connected to the input of another block. Otherwise, the build process will terminate with an error.

Note 2. The PIL target does not support the TOUCAN Interrupt Generator block or the Can Calibration Protocol Block.

Note 3. The AE target does not support the Can Calibration Protocol Block.

Hardware Clock Configuration

- The Embedded Target for Motorola MPC555 supports a limited range of sample times, as follows.

  For an external clock frequency of 20Mhz:

  - Fastest sample time = 1.28e-5 sec.
  - Slowest sample time = 0.8388 sec.

  For an external clock frequency of 4 Mhz:

  - Fastest sample time = 6.4e-5 sec.
  - Slowest sample time = 4.1942 sec.

  Note that if you select a sample time slower than the slowest possible for your clock frequency, Simulink issues a warning message.

  Also note that the fastest sample time may not be achievable because timer overruns may occur, depending on your model.

- The current documentation does not include a procedure for configuring the Embedded Target for Motorola MPC555 for an MPC555 crystal frequency other than 20 MHz. This note provides that information.

  The MPC555 can operate with a crystal frequency of either 4 MHz or 20 MHz. By default, the Embedded Target for Motorola MPC555 is configured for a crystal frequency of 20 MHz.

  For the real-time target, you can set the crystal frequency from the System Configuration parameters of the MPC555 Resource Configuration block (see "System Configuration Parameters" in the *Embedded Target for Motorola MPC555 User's Guide*).

  If you are using the PIL target, we recommend that you use a board with a 20 MHz crystal frequency. We specifically recommend the Phytec PhyCORE-MPC555 board for use with the PIL target.

  If your hardware uses a 4 MHz crystal frequency, you must edit the System Configuration parameters of the MPC555 Resource Configuration block. Change the **Oscillator Frequency** parameter to 4 MHz, and change the **USIU_PLPRCR_B_MF** parameter to 4.

  In addition, it is necessary to re-configure the boot code and driver code. To do this:

1. In the directory

   ```
   matlabroot\toolbox\rtw\targets\mpc555dk\common\drivers\bootcode
   ```

   delete the files `gmd_ppc_cmf_300_A61_200.c` and `gmd_ppc_cmf_300_A61_200.o`.

2. In the same directory, make a copy of the file `gmd_ppc_cmf_300_A61_200_04.c`, and rename the copy to `gmd_ppc_cmf_300_A61_200.c` (replacing the `.c` file you deleted in step **1**).

3. Edit the file

   ```
   matlabroot/toolbox/rtw/targets/mpc555dk/common/drivers/libmods/timers/
   clocks_common.h
   ```

   and change the line

   ```
   #define MPC555_EXTERNAL_CLOCK_FREQ 20e6
   ```

   to

   ```
   #define MPC555_EXTERNAL_CLOCK_FREQ 4e6
   ```

4. Re-compile the boot code as described in "Boot Code Parameters for CAN Download" in the *Embedded Target for Motorola MPC555 User's Guide*. Recompilation produces a new `bootcode_flash.bin` file.

5. Program the updated `bootcode_flash.bin` onto the target hardware, as described in "Downloading Boot Code" in the *Embedded Target for Motorola MPC555 User's Guide*.

   To switch back to 20 MHz, just use the same procedure as above, but replace the file `gmd_ppc_cmf_300_A61_200.c` with the file `gmd_ppc_cmf_300_A61_200_20.c`.

PIL Target

- If you change the cross-compiler you use with the PIL target (from Diab to Metrowerks or vice versa), you should rebuild your PIL models in a clean directory, or delete all files from the models' code generation directories. The PIL build process expects to start with a clean directory, or a directory

created in the process of building with the same compiler. Leftover components built by a different compiler cause errors.

- The PIL co-simulation target has a **Download and run** option that uses your development environment to download the generated code to the target board as part of the build process. For more information, read the section "Overview of PIL Co-Simulation" in the *Embedded Target for Motorola MPC555 User's Guide* and work through the tutorials there.

  Once a subsystem has been built using the PIL target, it is possible to manually download the generated code to the target without repeating the entire build process. To do this, use the following procedure:

1. Locate the directory where the S-function, `subsystem_pil_sf.dll`, is located and add this directory to the MATLAB path.

2. Execute the following command at the MATLAB prompt:

   ```
   pildnload('subsystem_pil_sf')
   ```

- If you have difficulties using the PIL target **Download and run** option, or the `pildnload` utility, the problem may be caused by the `LPT` speed setting for the BDM port. This note describes how to configure the parallel port speed.
  - For SingleStep: For manual downloading, set the **Delay** parameter in the **Connection** tab of the **Debug** dialog (See "Configure SingleStep Parameters" in the *Embedded Target for Motorola MPC555 User's Guide*).
  - For the PIL target **Download and run** option using Diab/SingleStep, or for the `pildnload` utility: Speed values are obtained from the file

    ```
    matlabroot\toolbox\rtw\targets\mpc555dk\common\tools\diab\sdsgetport.m.
    ```

    In `sdsgetport.m`, edit the string

    ```
    '-p LPT1=6'
    ```

    and change the parameter value to the right of the = sign. If you are not sure what the correct speed value is, we suggest that you follow the procedure described in "Configure SingleStep Parameters" in the *Embedded Target for Motorola MPC555 User's Guide* until you obtain a **Delay** value that works. Then enter this number as the speed value.
  - For Metrowerks CodeWarrior: make sure that you have configured the **Remote Connections** settings as described in "Setting Up Your Installation with Metrowerks CodeWarrior" in the *Embedded Target for*

Motorola MPC555 User's Guide. If necessary, adjust the **Speed** property of the **MPC555DK Wiggler** configuration.

SingleStep Debugger

- If you are using the SingleStep debugger and connecting to the target board by a port other than LPT1, you must modify the following file:

  ```
  matlabroot/toolbox/rtw/targets/mpc555dk/common/tools/diab/getsdsport.m
  ```

  This file provides a default connection string used by SingleStep to connect to the target. To use a port other than LPT1, locate and change the connection string:

  ```
  '-p LPT1=6'
  ```

- The path to the SingleStep debugger, specified in the `TargetDebuggerExe` preference, must specify the full path to the executable file, on either an actual hard drive on your PC, or a mapped drive. Do not use a UNC.

Other Limitations and Warnings

- If you are using Metrowerks CodeWarrior, you do not need to set the `TargetCompilerPath` property of the target preferences. The `TargetCompilerPath` property is only required if you are using the Diab compiler.
- When the MPC555 Resource Configuration block (see "MPC555 Resource Configuration" in the *Embedded Target for Motorola MPC555 User's Guide*), is placed into a model, it modifies the `preloadfcn` callback of the model. If you wish to add a command to the `preloadfcn` callback of a model that already has an MPC555 Resource Configuration block, do not remove the commands that are already installed.

  Instead, copy the installed `preloadfcn` callback and append your commands. Then set the `preloadfcn` to the merged command. If you corrupt the `preloadfcn`, you can retrieve the command from any model has an MPC555 Resource Configuration block, as the `preloadfcn` will be the same for all models. You can retrieve the `preloadfcn` with the following command:

  ```
  plf = get_param(bdroot,'preloadfcn')
  ```

- The following warning, which is displayed at the end of a successful build process, should be ignored.

  ```
  dld: warning: Overlapping memory block stack
  ```

- Warnings similar to the following may appear when some of the Embedded Target for Motorola MPC555 demos are opened. You can ignore these warnings; the demo models will operate correctly.

  ```
  Warning: Loading model 'mpc555pil_fuelsys' generated with a newer
  version(4.10) of Simulink.
  Loaded Stateflow module Version 4.1 (R12.1) dated May 21 2001,
  01:15:03
  ```

- The **Help** link in the right-click pop-up menu for sublibraries of the Embedded Target for Motorola MPC555 block library (`mpc555library`) is not implemented in this release. Use the **Help** button in the `mpc555library` window instead.

  Similarly, the **Help** link in the right-click pop-up menu for some Embedded Target for Motorola MPC555 blocks is not implemented. This limitation affects:

  - the MPC555 Resource Configuration block.
  - the Can Calibration Protocol block.
  - Configuration Class blocks.

  For these blocks, the quickest route to online help is the “Block Reference” section of the Embedded Target for Motorola MPC555 documentation.

Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the The Embedded Target for Motorola MPC555 1.0 to Version 1.0.1.

ASAP2 Files Now Written to Build Directory

In the previous release, when generating an ASAP2 file, the Embedded Target for Motorola MPC555 real-time target wrote the ASAP2 file to the same directory as the source model.

In version 1.0.1, ASAP2 files are written to the build directory. If you have scripts or other procedures that assume the generation of ASAP2 files in the same directory as the source model, you should update them.