RTMaps TLC Target for Simulink

v2.7.1

Intempora S.A.S.

Contents

1 Disclaimer

INTEMPORA S.A.S. MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND SPECIFICALLY DISCLAIMS ANY IMPLIEDWARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE.

Further, Intempora S.A.S. reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Intempora S.A.S. to notify any person of such revision or changes.

RTMaps (Real Time Multisensor Applications) is a registered trademark of Intempora S.A.S. All other company, product, or service names mentioned in this book may be trademarks or service marks of their respective owners.

Intempora S.A.S.
19, rue Diderot
92130 Issy les Moulineaux
FRANCE
Tel: +33 1 41 90 03 59
Fax: +33 1 41 90 08 39
https://www.intempora.com

2 Introduction

With the RTMaps TLC Target for Simulink, you can automatically compile a Simulink model into an RTMaps component:

- This target converts the model's inports and outports into the inputs and outputs of the generated RTMaps component
- Fixed-size vectors of `double`, `single`, `(u)int(8,16,32)`, `boolean` and **bus objects** are currently supported on the model's INports and OUTports
- For **variable-size inputs and outputs**, please refer to the "Handling Variable-Size I/O" section of this document
- For handling **image data**, please refer to the "Handling Images (IplImage)" section of this document
- Not all Simulink blocks are supported
- Matlab versions from **R2016a** are supported on **Windows** and **Ubuntu Linux**

3 Prerequisites

3.1 RTMaps

In order to **compile** the generated RTMaps component, the following is required:

- RTMaps Developer version >= 4.7.0
- The `sdk_rtmaps_input_reader_dev` archive version >= 0.3.1
  (can be downloaded using the RTMaps Updater)
- The `rtmaps_mathworks` archive version >= 3.19.2
  (can be downloaded using the RTMaps Updater)

In order to **load and run** the compiled RTMaps component, the following is required:

- RTMaps (Runtime or Studio) version >= 4.2.3
- The `sdk_rtmaps_input_reader` archive version >= 0.3.1
  (can be downloaded using the RTMaps Updater)

3.2 Simulink

The following Mathworks products are required in order to generate and compile the code:

- Matlab

- Simulink
- Matlab Coder
- Simulink Coder

In order to check whether those products are installed, you can run the `ver` command in Matlab.

The resulting RTMaps component can run without having those products installed.

3.3 C and C++ Compiler

In order to build the RTMaps package, a C++11 compiler is needed:

- Depending on your operating system:
  - Windows: Microsoft Visual C++ >= 2015
  - Ubuntu Linux: The distribution's default version of GCC (i.e. GCC >= 5 for the supported Ubuntu distributions)
- The compiler has also to be supported by your version of Matlab (cf. "Supported Compilers" under System Requirements on Mathworks' website)

Once a compiler is installed, it has to be configured as the C **and** C++ compiler in Matlab using the following commands:

```
mex -setup C    % then select a supported compiler by e.g. clicking on
    it
mex -setup C++  % then select the same version of the supported
    compiler by e.g. clicking on it

% on some older versions of Matlab, you might have to execute the
    following command instead
mex -setup      % then select a supported compiler by e.g. clicking on
    it
```

For instance, under Windows, **the same version** of Microsoft Visual C++ must be set for both C and C++.

3.4 Proper Access Rights to the "samples" Folder

In order to test and compile the included samples, you must have "write" access to the `<RTMaps Install Dir>/tools/simulink/rtmaps_tlc_target/samples` folder.

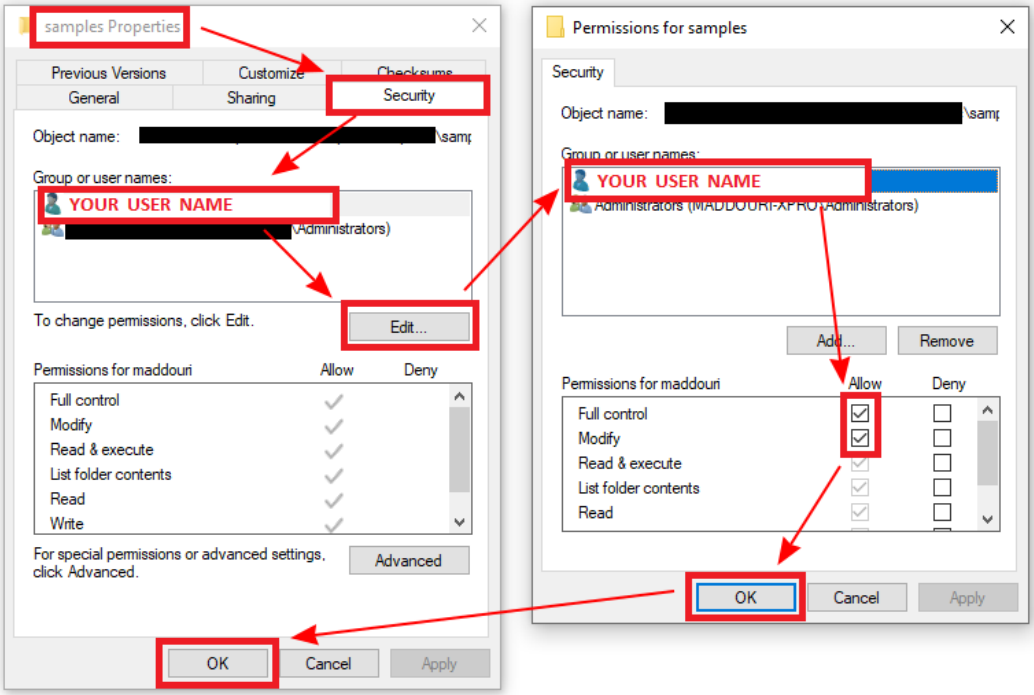
Usually, you can ensure that by doing the following:

- On Linux

```
sudo chown $USER:$USER -R "<RTMaps Install Dir>/tools/simulink/
    rtmaps_tlc_target/samples"
```

- On Windows:

1. Right-click on `<RTMaps Install Dir>/tools/simulink/rtmaps_tlc_target/samples`
2. Click on “Properties”
3. Configure the security properties as shown in the following screenshot



4 Setup

4.1 Installing

1. Run the `install_rtmaps_target` function found under

```
<RTMaps Install Dir>/tools/simulink/rtmaps_tlc_target/rtmaps_tlc
```

Either drag and drop the file into the Matlab window or run the following command in Matlab: `matlab run('<RTMaps Install Dir>/tools/simulink/rtmaps_tlc_target/rtmaps_tlc/install_rtmaps_target')`

2. Make sure that **the same version** of a C **and** C++ compiler is set using the following commands:

```
mex -setup C    % then select a supported compiler by e.g.
    clicking on it
mex -setup C++  % then select the same version of the supported
    compiler by e.g. clicking on it

% on some older versions of Matlab, you might have to execute the
    following command instead
mex -setup      % then select a supported compiler by e.g.
    clicking on it
```

3. Done! If Simulink was running, then restart it in order for the changes to take effect

4.2 Testing

In order to test the newly-installed RTMaps TLC Target for Simulink, just run the following command:

```
test_rtmaps_target
```

This command executes the `run_simple_test.m` file available under:

```
<RTMaps Install Dir>/tools/simulink/rtmaps_tlc_target/samples/
    simple_test
```

This test creates a temporary Simulink model that has 1 Inport, 1 Gain (gain = 2) and 1 Outport. The Inport is connected to the Gain's input and the Outport is connected to the Gain's output.

If the RTMaps TLC Target for Simulink is installed and the Matlab environment configured correctly, then the new `simple_test.pck` package should be generated in the current working directory. You can test the package using the `simple_test.rtd` in RTMaps.

If the test fails, then make sure that:

1. You have gathered the prerequisites mentioned in the "Prerequisites" section
2. Properly installed the RTMaps TLC Target for Simulink by running the `install_rtmaps_target` script
3. You have "write" access to the `samples/simple_test` folder. If you don't or if you are not sure, then, you can copy `samples/simple_test` to e.g. your home folder and run `run_simple_test.m` from there

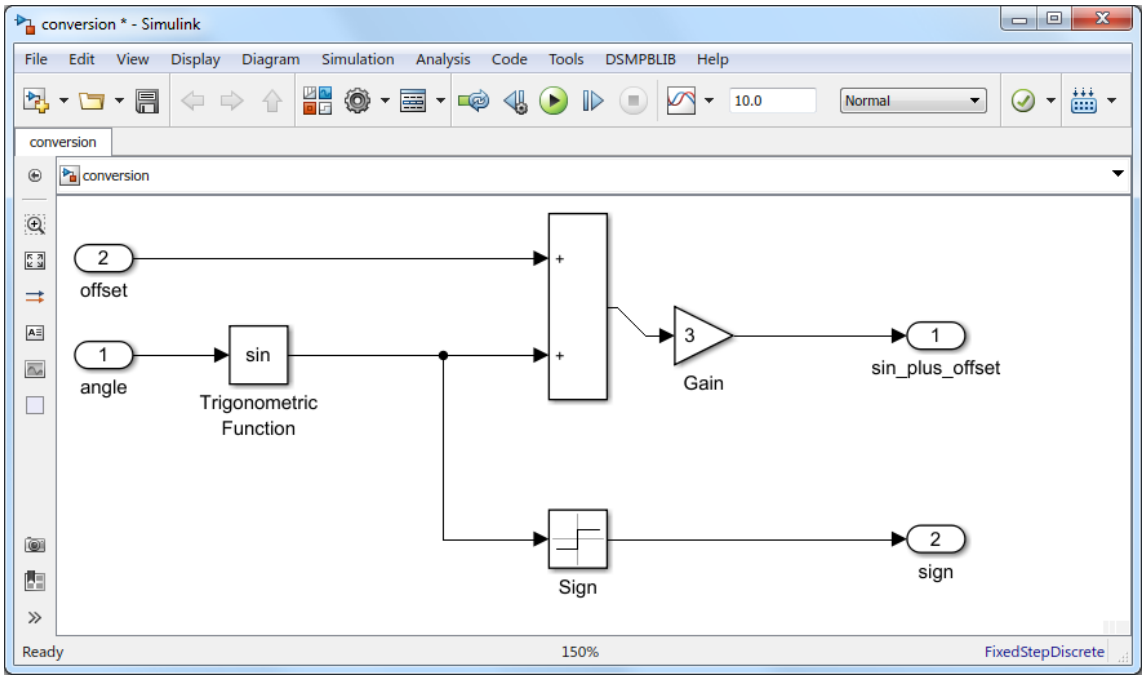
4.3 Uninstalling

Simply use the `uninstall_rtmaps_target` function in Matlab then restart Simulink in order to apply the changes.

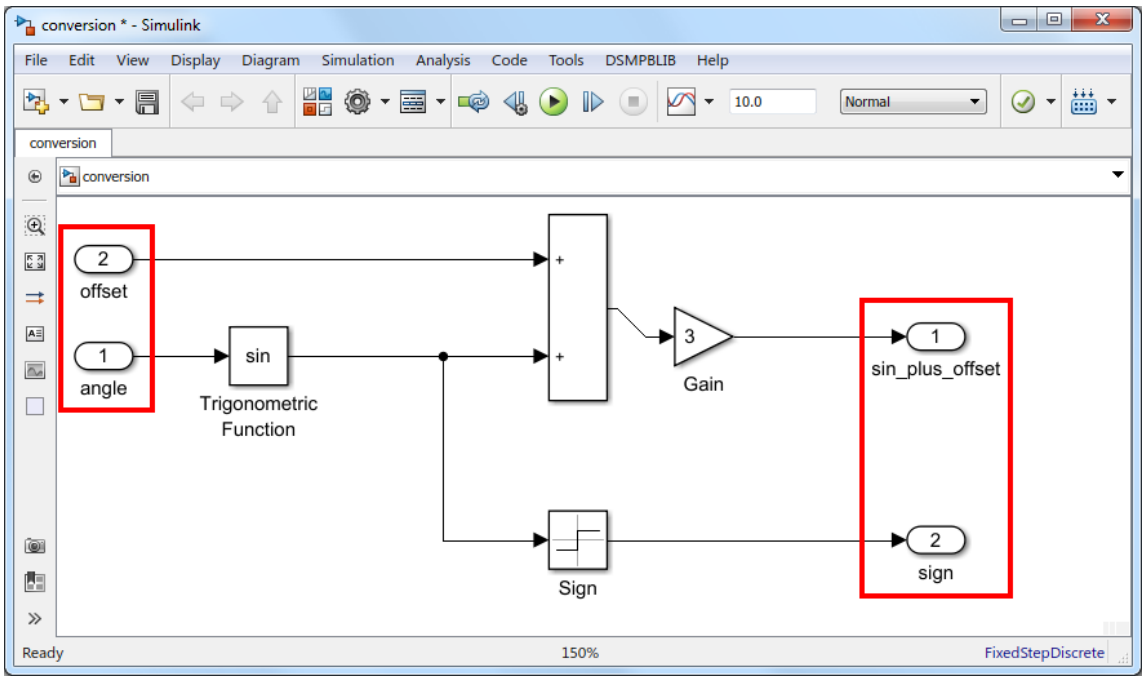
5 Usage: Generating an RTMaps Package

5.1 From a Simulink Model, Using the Simulink GUI

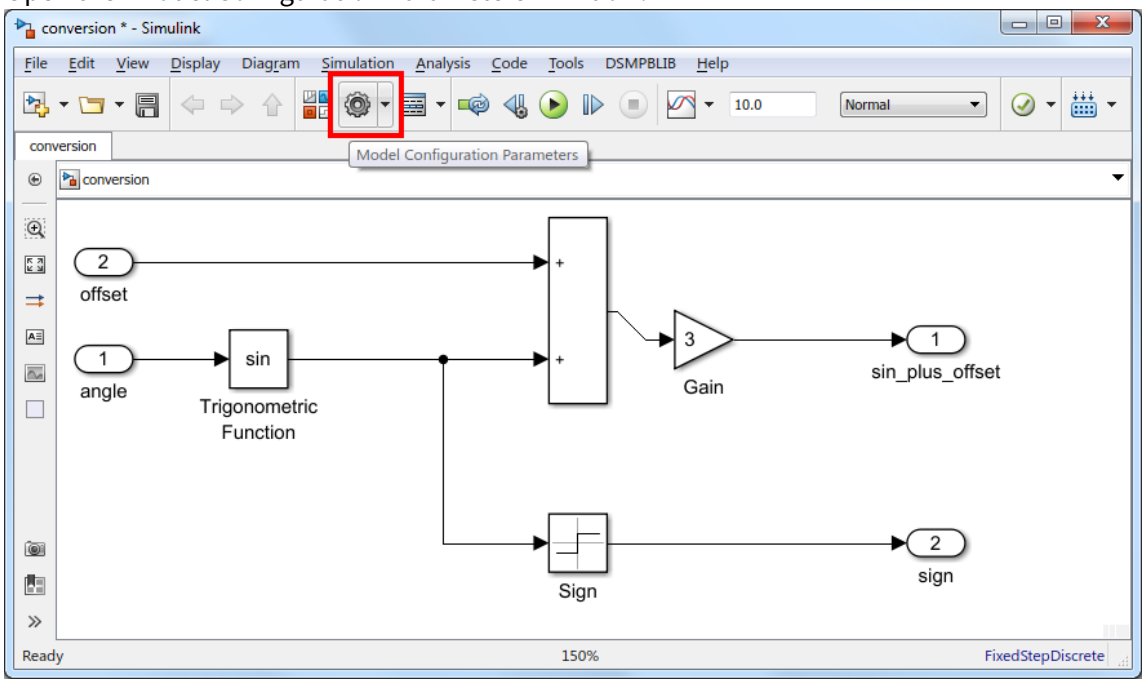
1. Start Simulink
2. Create a new model or load an already existing one. For instance, the following is the `conversion` example provided under `<RTMaps Install Dir>/tools/simulink/rtmaps_tlc_target/samples/conversion`

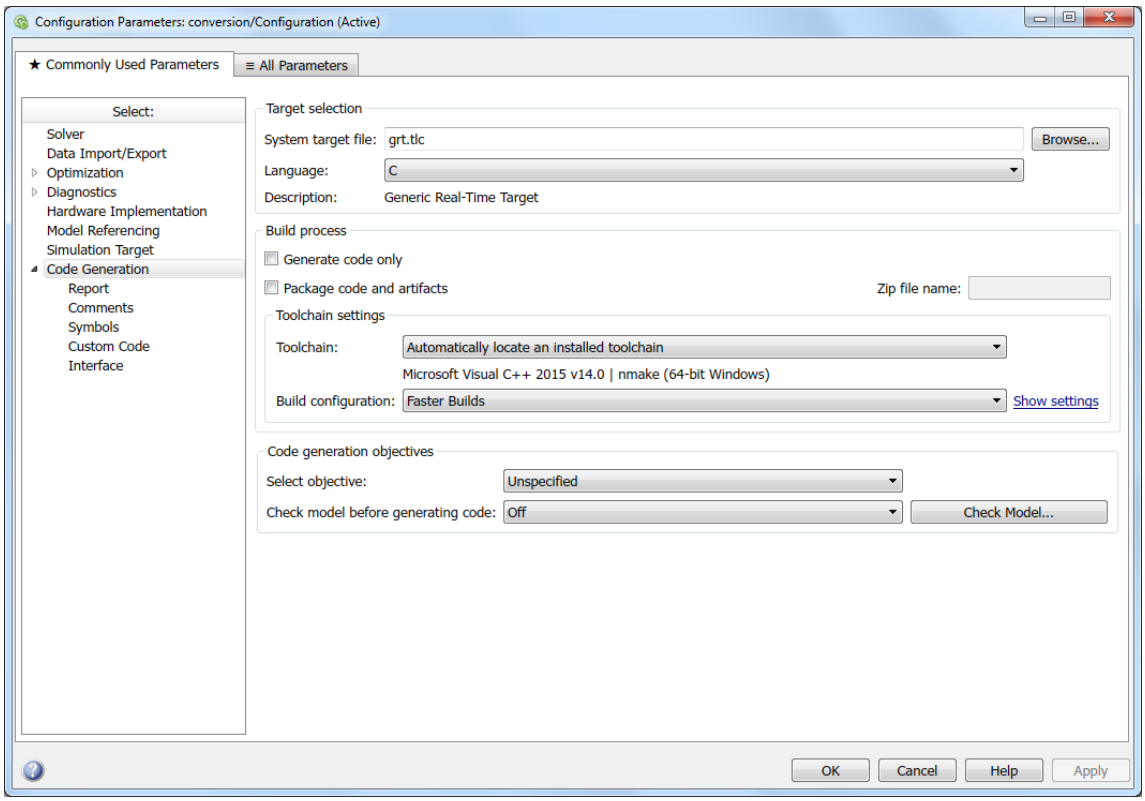


3. Make sure that the model contains **at least 1 Inport and 1 Outport**. This is currently required by the current version of the RTMaps TLC Target for Simulink in order to generate and compile the RTMaps component's code. If your model doesn't require either Inports or Outports, you can simply add 1 Inport and 1 Outport then connect them to each other.

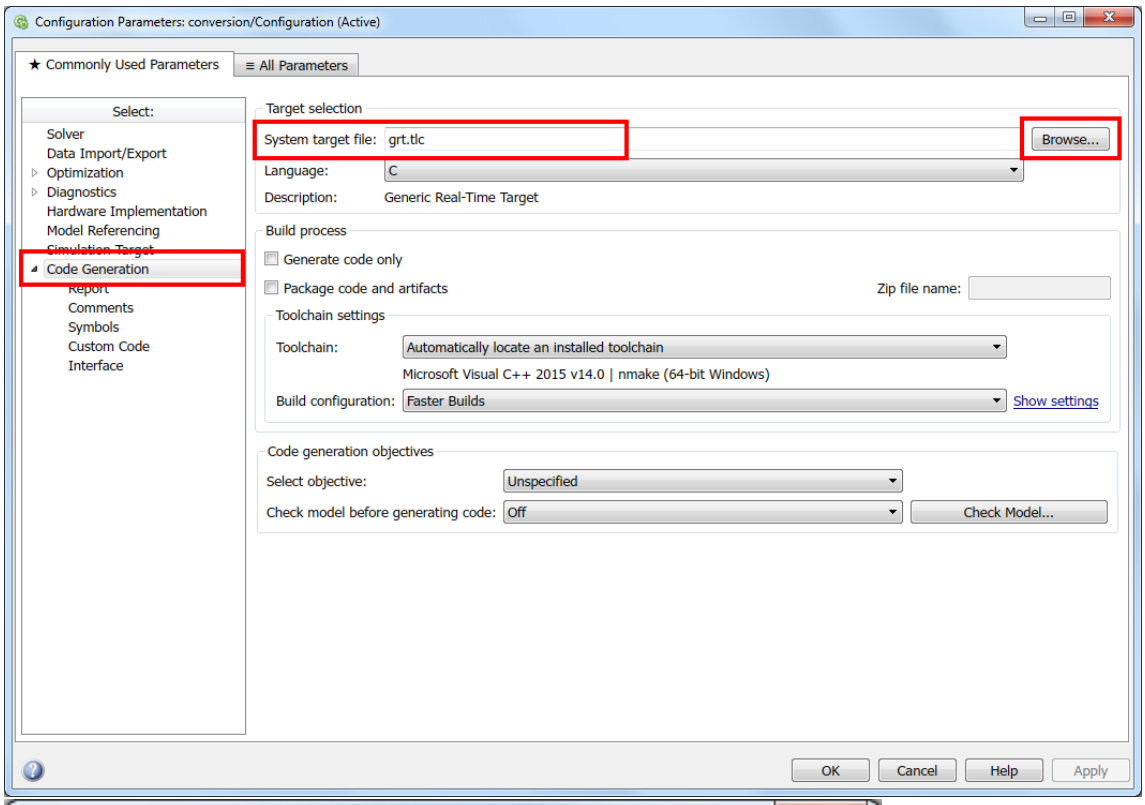


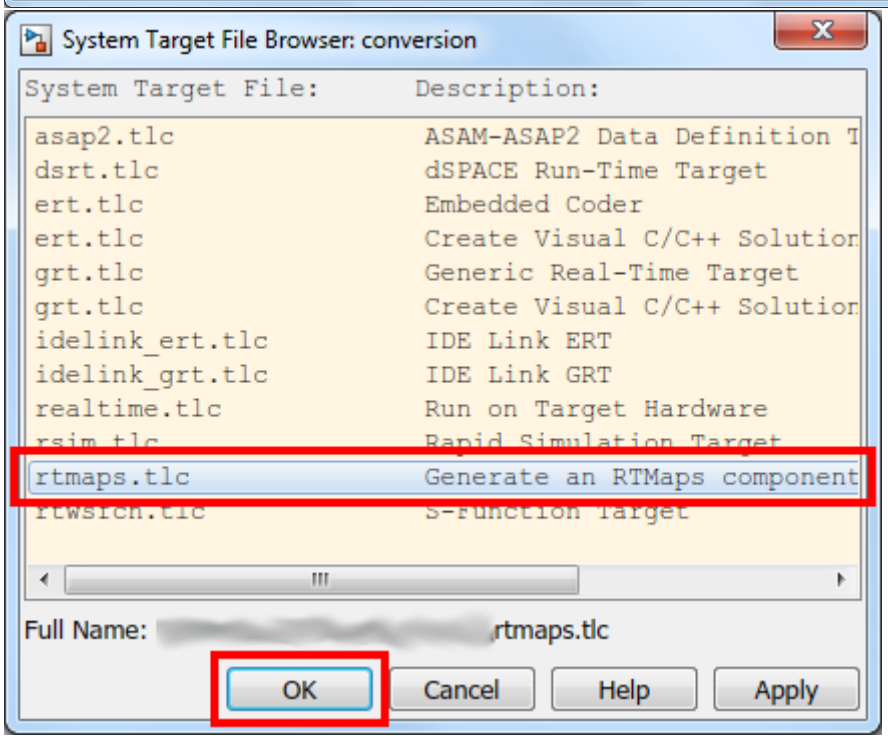
4. Open the “Model Configuration Parameters” window.

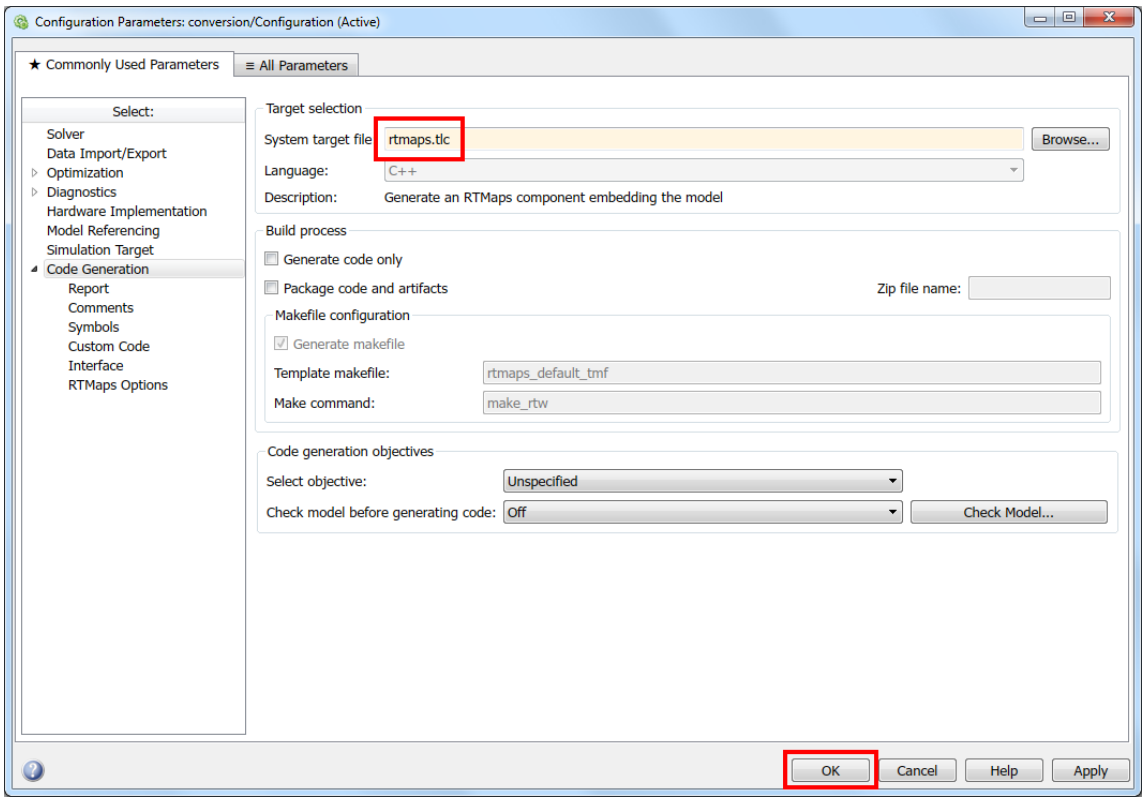




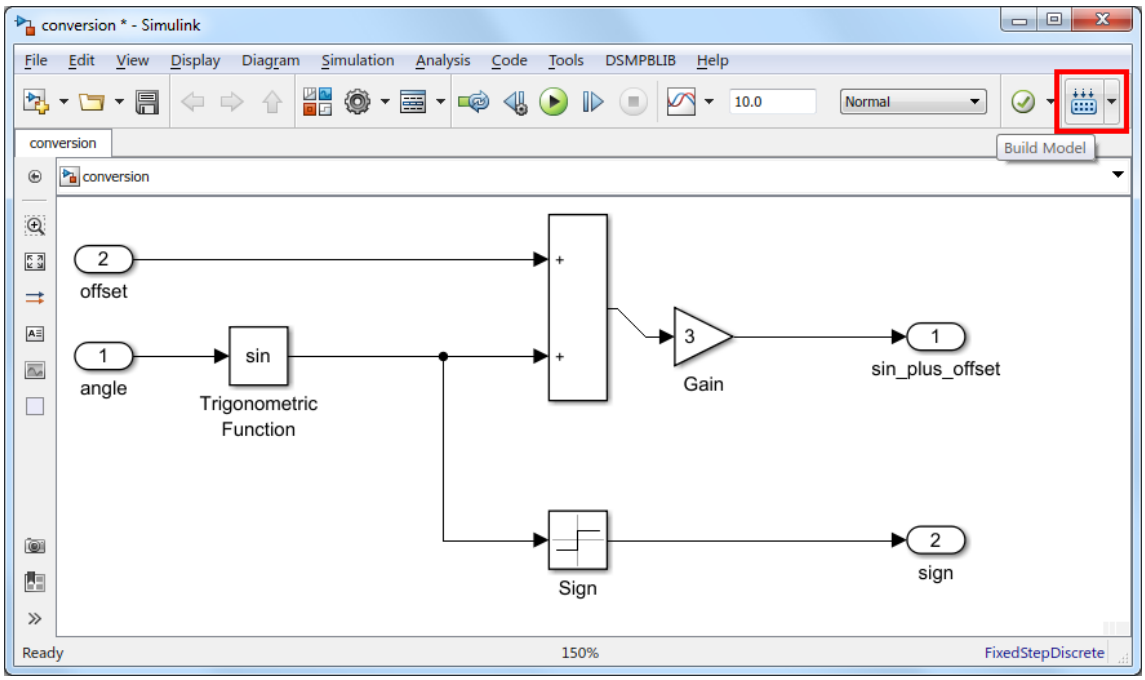
5. In the “Model Configuration Parameters” window, under the “Code Generation” left menu item, enter `rtmaps.tlc` (**all lowercase letters**) in the “System target file” edit box then press the “Enter” key on your keyboard. Alternatively, you can press the “Browse…” button then select the `rtmaps.tlc` (**all lowercase letters**) target from the “System target file browser” window.


Configuration Parameters: conversion/Configuration (Active)
★ Commonly Used Parameters
≡ All Parameters
Select:
Solver
Data Import/Export
Optimization
Diagnostics
Hardware Implementation
Model Referencing
Simulation Target
Code Generation
Report
Comments
Symbols
Custom Code
Interface
Target selection
System target file: grt.tlc
Browse...
Language: C
Description: Generic Real-Time Target
Build process
Generate code only
Package code and artifacts
Zip file name:
Toolchain settings
Toolchain: Automatically locate an installed toolchain
Microsoft Visual C++ 2015 v14.0 | nmake (64-bit Windows)
Build configuration: Faster Builds
Show settings
Code generation objectives
Select objective: Unspecified
Check model before generating code: Off
Check Model...
OK
Cancel
Help
Apply

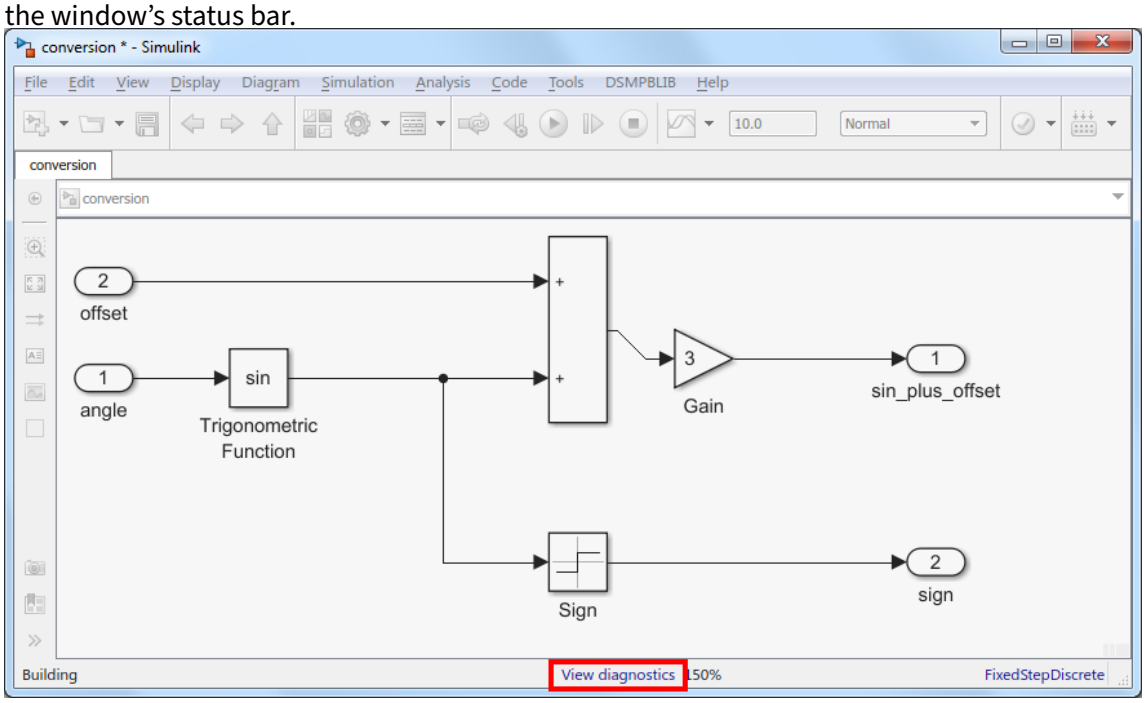

System Target File Browser: conversion
System Target File: Description:
asap2.tlc ASAM-ASAP2 Data Definition T
dsrt.tlc dSPACE Run-Time Target
ert.tlc Embedded Coder
ert.tlc Create Visual C/C++ Solution
grt.tlc Generic Real-Time Target
grt.tlc Create Visual C/C++ Solution
idelink_ert.tlc IDE Link ERT
idelink_grt.tlc IDE Link GRT
realtime.tlc Run on Target Hardware
rsim.tlc Rapid Simulation Target
rtmaps.tlc Generate an RTMaps component
rtwsfcn.tlc S-Function Target
Full Name: rtmaps.tlc
OK
Cancel
Help
Apply

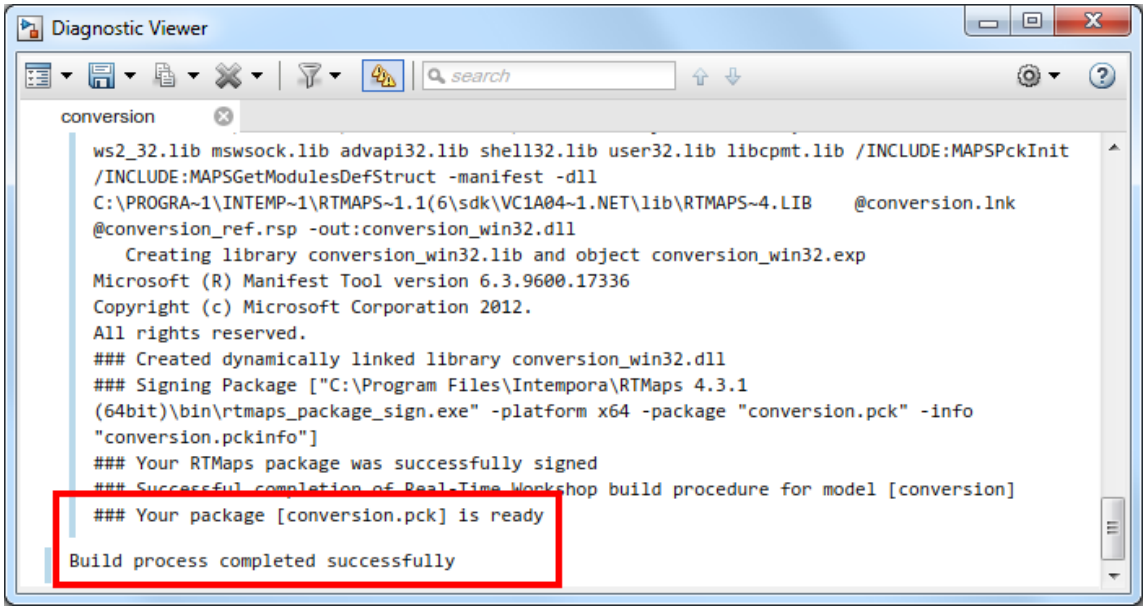


6. A new left menu entry named “RTMaps Options” is created. You can use it to check various informations about the RTMaps TLC Target for Simulink.
7. Press the “OK” button in the “Model Configuration Parameters” window to use of the new target.
8. In the Simulink window, press the “Build Model” button

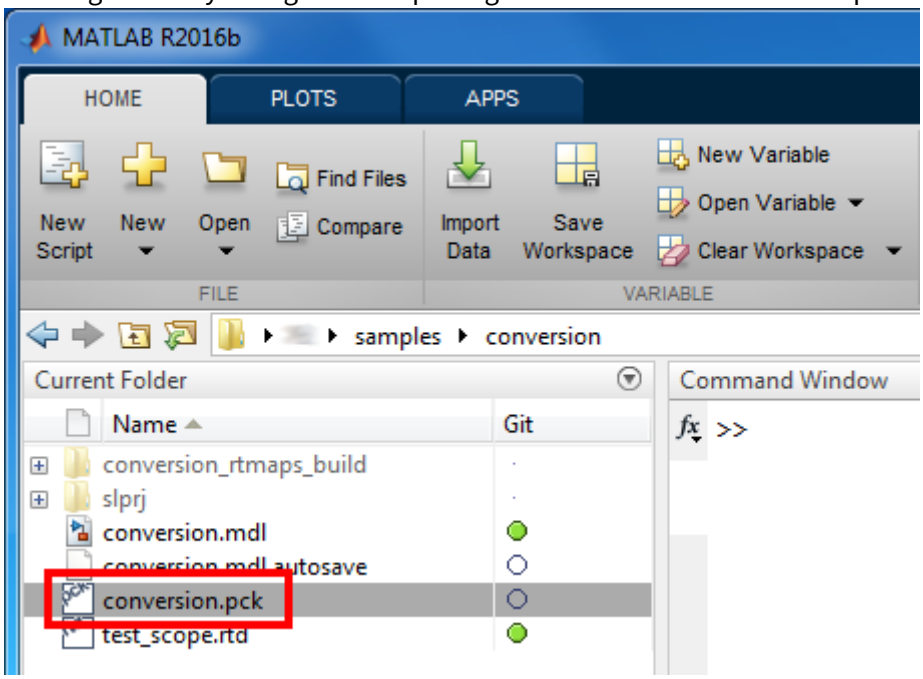


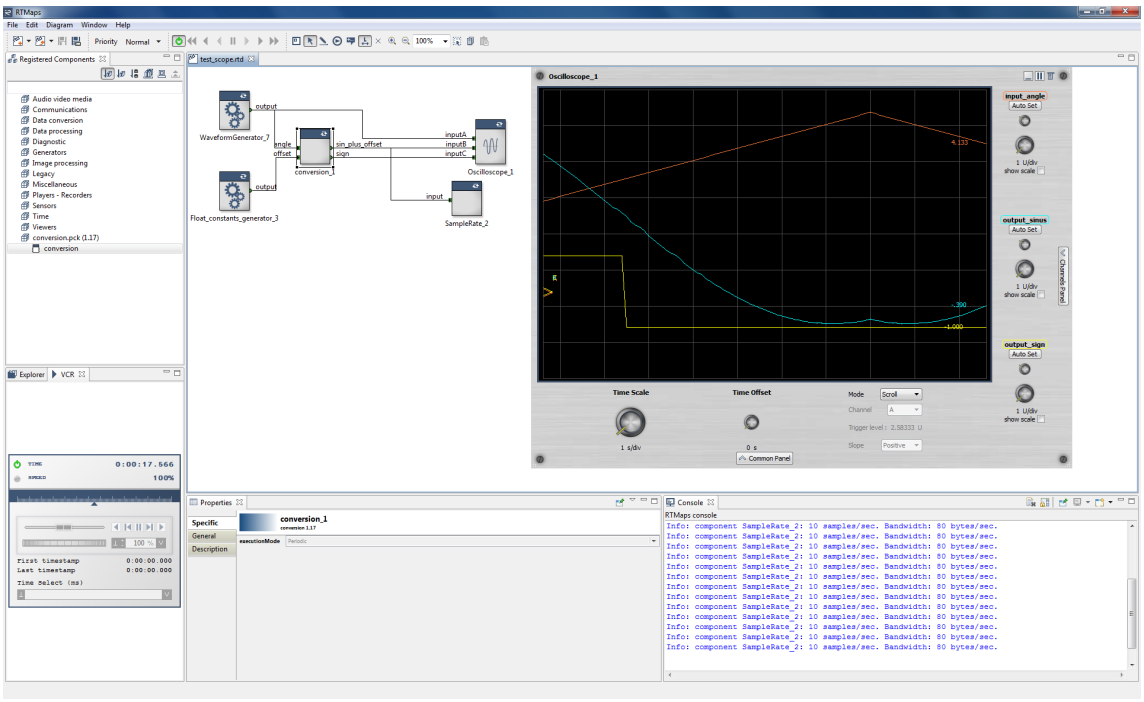
9. To monitor the code generation and compilation process, click on the “View diagnostics” link in the window’s status bar.





10. If the build succeeds, a new package having the same name as the model is created in the current working directory. The generated package can then be loaded in RTMaps.





5.2 From a Simulink Model, Using `rtmaps_model2pck`

To quickly generate an RTMaps package from an existing `.mdl` or `.slx` Simulink model, you can use the `rtmaps_model2pck` function from the Matlab command window:

```
1 rtmaps_model2pck('model')   % model name without the ".mdl" or ".slx"
     extension
2 rtmaps_model2pck(model)     % model handle
```

This will generate `model.pck` from `model.mdl` or `model.slx`.

To customize the model's parameters first, `rtmaps_model2pck` accepts the following syntax:

```
1 rtmaps_model2pck('model', 'Param1Name', Param1Value, 'Param2Name',
     Param2Value, ...)
```

For further details, type:

```
1 help rtmaps_model2pck
```

5.3 From a Matlab Function, Using `rtmaps_function2pck`

To quickly generate an RTMaps package from an existing `.m`, `.p` or built-in function, you can use the `rtmaps_function2pck` function from the Matlab command window:

```
rtmaps_function2pck('func')  % function name
rtmaps_function2pck(@func)   % function handle
```

This will generate `func_m.pck`, `func_p.pck` or `func_b.pck` depending on whether `'func'` is an `.m`, `.p` or built-in function.

In order to achieve this, `rtmaps_function2pck` generates of a *temporary* Simulink model that contains a "MATLAB Function" block embedding `func`. Then, the temporary model is converted into an RTMaps package using `rtmaps_model2pck`.

To customize the temporary model's parameters, `rtmaps_function2pck` accepts the following syntax:

```
rtmaps_function2pck('func', 'Param1Name', Param1Value, 'Param2Name',
    Param2Value, ...)
```

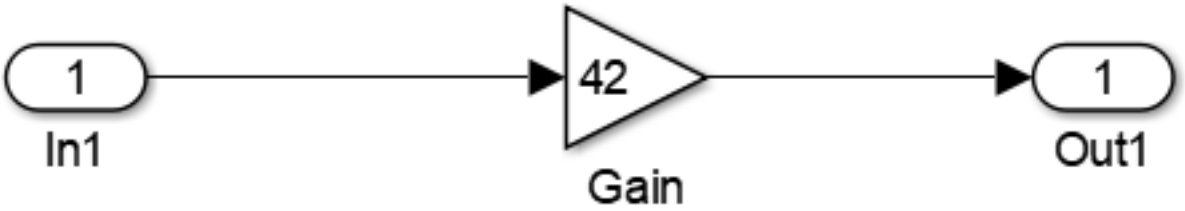
For further details, type:

```
help rtmaps_function2pck
```

6 Handling Variable-Size I/O

In order to handle variable-size I/O, you must define special integer INports and/or OUTports that have the `_size` suffix appended to their name.

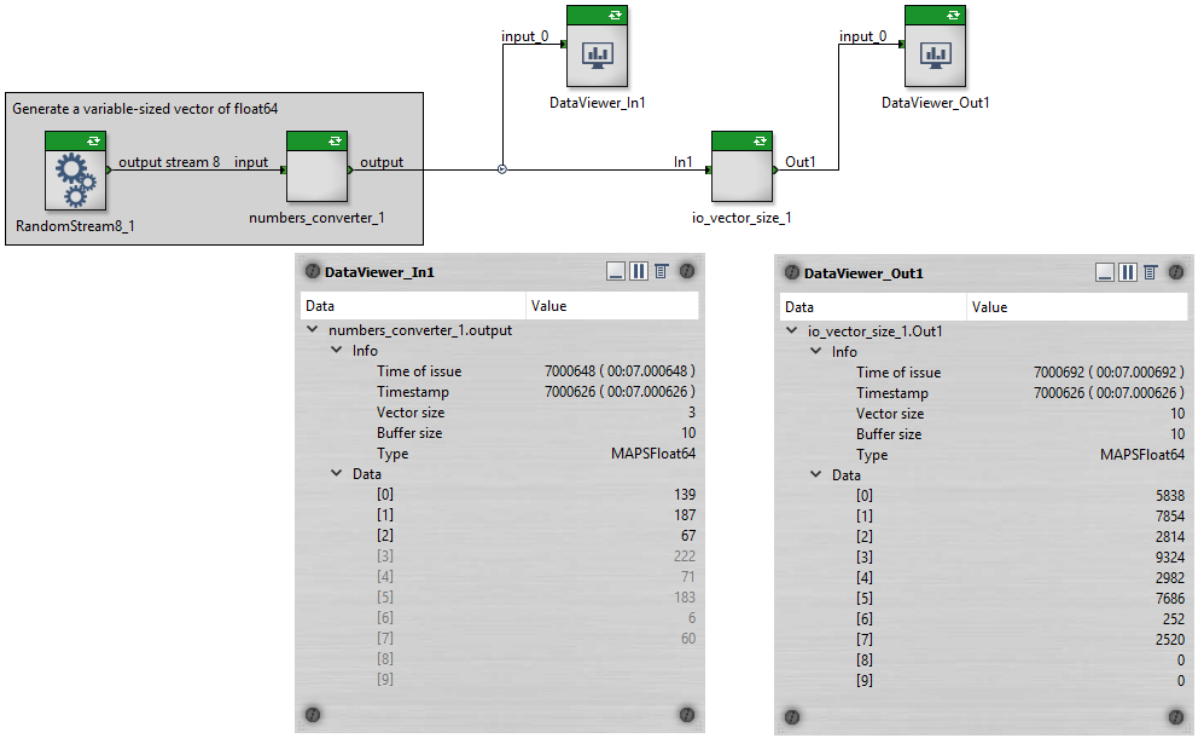
For example, suppose that you have the following model



such that `In1` has a width of 10 and `Out1` also has a width of 10. (The fact that both the inport and the outport have the same width is **just an example** here. In other usecases, **they can have different widths**)

The generated RTMaps component would be

When you use such a component, it would always read 10 elements on its input and write 10 elements on its output as shown below



Note in the above screenshot of RTMaps, even though the upstream component (`numbers_converter_1`) is writing a vector containing 3 elements only, the generated `io_vector_size_1` component is still going to read 10 elements. One of the goals of this section is to show you how to configure your Simulink model so that the generated `io_vector_size_1` reads only what the upstream component has written (i.e. 3 elements in the above example).

Now, if you want to communicate to your RTMaps diagram the facts that

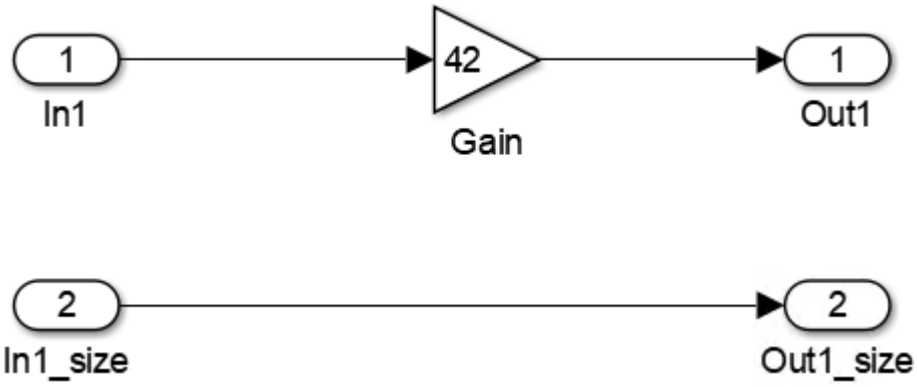
1. The RTMaps component should actually read a number of elements that is <= 10 from its input
2. It might actually write a number of elements that is <= 10 on its output

then, **in your Simulink model**, you must

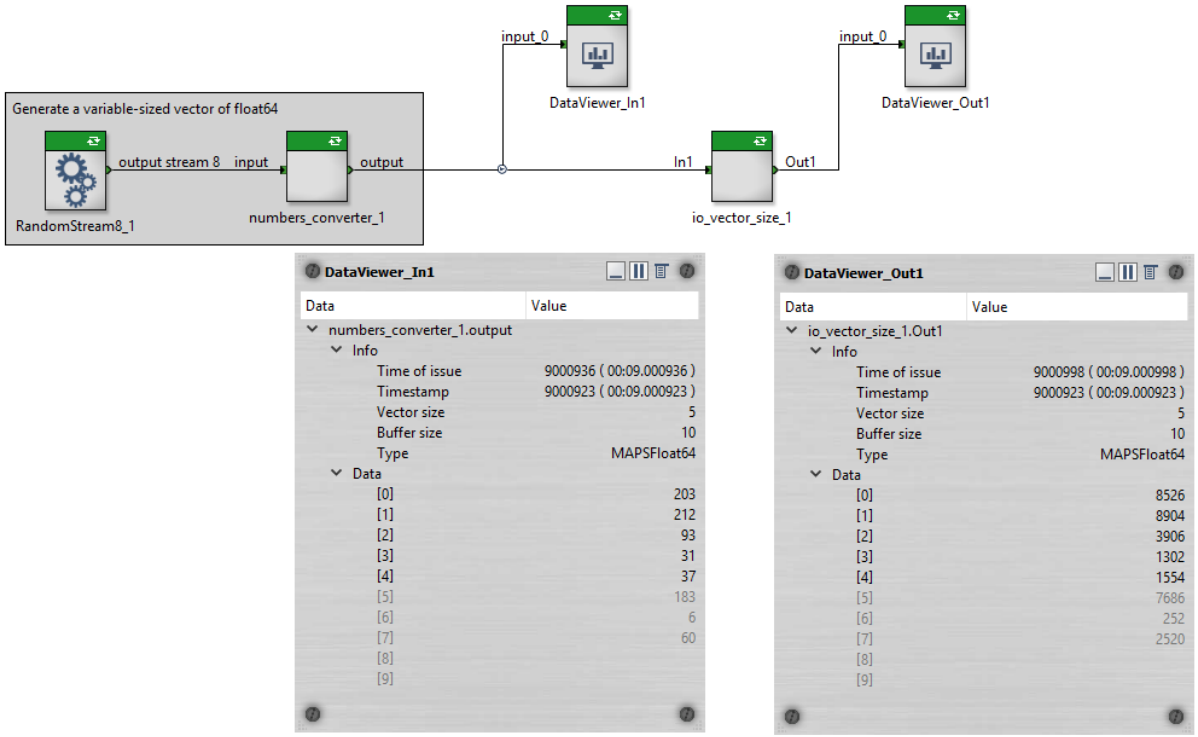
1. Define a new inport that is named `In1_size`. It will indicate, to the Simulink model, the number of elements that have been read on the input of the RTMaps component
2. Define a new outport that is named `Out1_size`. It will indicate, to the RTMaps component, the number of elements that must be read from the Simulink outport and written on the output of the RTMaps component
3. Implement, the logic for handling `In1_size` and computing `Out1_size` in the Simulink model

For the example shown in this section, a possible *(albeit trivial and not necessarily representative of*

what a real-world model might contain) could be the following solution in which we just transfer the vector size from the inport to the outport



If you generate and use and RTMaps component from such a model, you will get the following result in RTMaps



To sum up:

- An RTMaps component's input (respectively output) named `X` corresponds to the Simulink model's inport (respectively outport) that is named `X`
- The vector size of `X` in RTMaps is equal to the value of the `X_size` inport or outport in Simulink
- The buffer size of `X` in RTMaps is equal to the "width" of the corresponding `X` inport or outport in Simulink

Finally:

- The example shown in this section is available in `samples/io_vector_size`
- The vector size of a component's input or output is one of many "meta information" that can be specified. Please refer to "Supported I/O Meta Information" (in the "Reference" section of this document) for the complete list of meta information that can be used

7 Handling Images (IplImage)

In order to declare an image port on your model, you must define the ports that are shown in the following table:

Port Name	Port Type	Port Width
`MyPort_IplImage`	`uint8`	`[img_height img_width channel_count]`
`MyPort_IplImageSize`	`int32`	2 (contains `[img_height img_width]` in that order)
`MyPort_IplImageChannelSeq`	`uint8`	4 (e.g. `'GRAY'`, `'RGB'`, etc.)

The input or output of the generated RTMaps component that corresponds to the ports that are shown in the above table will be named `MyPort_IplImage`.

Currently, the following restrictions apply to the **generated RTMaps component**:

- Input images **must**:
  - Be an `IplImage`
  - Be planar (i.e. `IplImage::dataOrder == IPL_DATA_ORDER_PLANE`)
  - Use `uint8` for data representation (i.e. `IplImage::depth == IPL_DEPTH_8U`)
- Output images **will**:
  - Be an `IplImage`
  - Be planar (i.e. `IplImage::dataOrder == IPL_DATA_ORDER_PLANE`)
  - Use `uint8` for data representation (i.e. `IplImage::depth == IPL_DEPTH_8U`)
  - Be aligned on 8 bytes (i.e. `IplImage::align == IPL_ALIGN_QWORD`)

Regarding the images that are manipulated by **the Simulink model**:

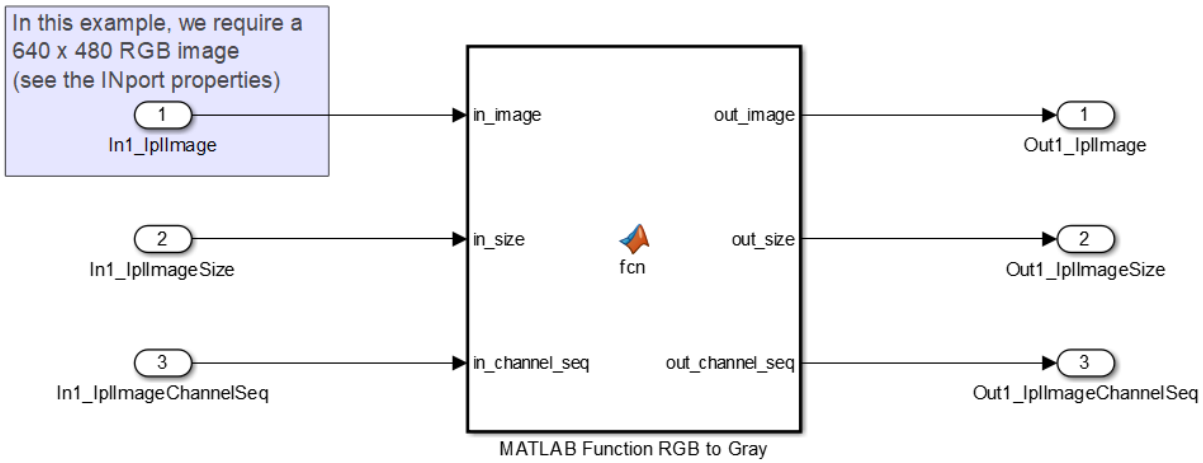
- Images read from an INport (e.g. called `My_INport_IplImage`) **will**:

  - Have the same dimensions (i.e. width, height and channel count) as the IplImage of the corresponding input of the generated RTMaps component
  - Not be aligned (which should be consistent with how matrices usually are in Matlab)
  - Have the `uint8` data type

- Images written to an OUTport (e.g. called `My_OUTport_IplImage`) **must**:

  - Have dimensions (i.e. width, height and channel count) that are consistent with the values that are contained in `My_OUTport_IplImageSize` and `My_OUTport_IplImageChannelSeq` (**this is very important**)
  - Not be aligned (this should be the default behavior of Matlab matrices)
  - Have the `uint8` data type

To illustrate the above, let's use the provided `samples/iplimage_io` sample:



In this example, the "MATLAB Function" block simply reads an RGB image (`in_image`), converts it to a gray image and outputs the result to `out_image`.

The generated RTMaps component looks like the following:

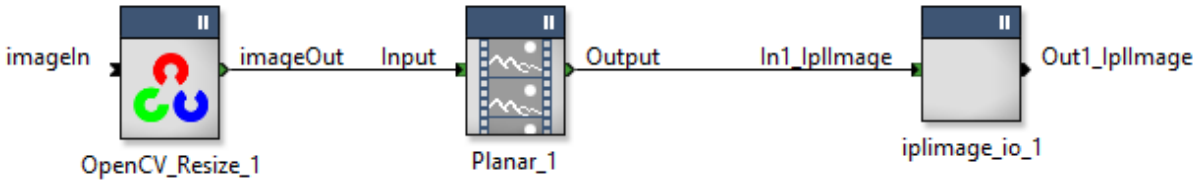
> Note: If you get an "Error: Unable to load package…" error message in the RTMaps Console when trying to register the generated package in RTMaps, it is likely that a Matlab dependency was not found. A typical example is `libmwrgb2gray_tbb.dll` when using Matlab 2020b on Windows 64bit. In this case, you might want to consider the following:

1. Find where `libmwrgb2gray_tbb.dll` is located. For this example, let's suppose that it is in
   `<Matlab DLL Dir> = C:/Program Files/MATLAB/R2020b/bin/win64`
2. Either
   - Copy `<Matlab DLL Dir>/libmwrgb2gray_tbb.dll` to `<Generated .pck Dir>`
   - Or copy `<Matlab DLL Dir>/libmwrgb2gray_tbb.dll` to `<Generated .pck Dir>/bin`
   - Or add `<Matlab DLL Dir>` to your system's PATH
3. Restart RTMaps

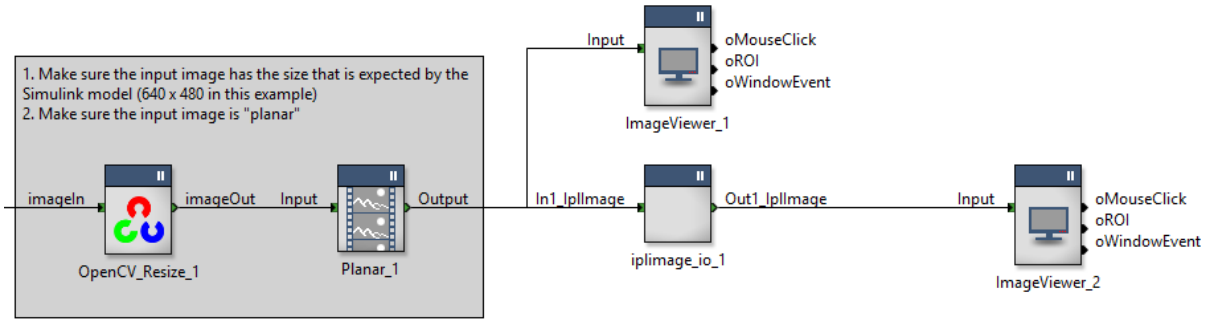
In order to comply with the mentioned input restrictions, you could:

1. Use an OpenCV_Resize component (from the `rtmaps_image_processing_opencv.pck` package) to make sure that the image has the expected size (iff your original image might have a different size)
2. Use a "Planar" component (from the components that are loaded by default) to make sure that the input image is always planar

The above 2 points, when applied, would result in the following setup on your diagram



Finally, to better understand how to use images on your model's I/O, please take the time to analyse, try and even modify the provided `samples/iplimage_io` sample.

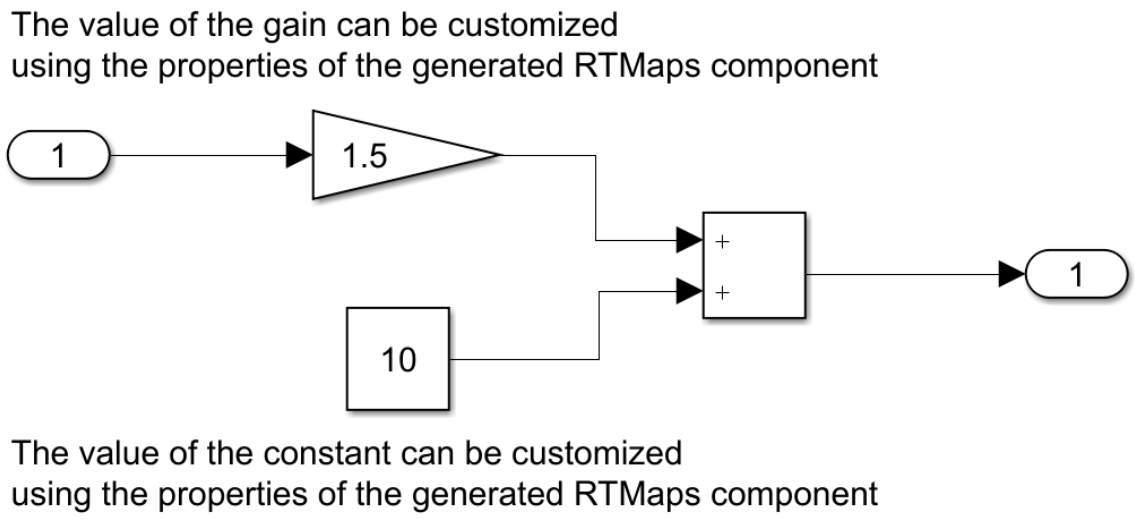


8 Model Parameters

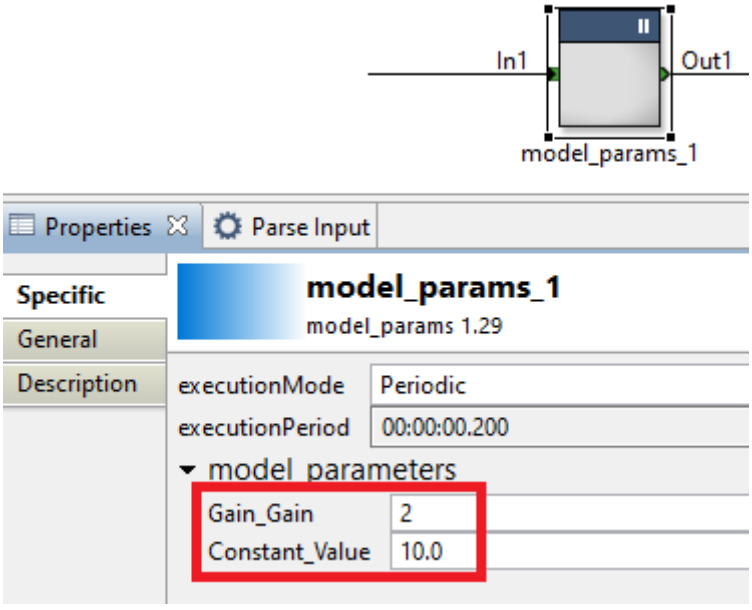
Some models make it possible to customize the parameters of their blocs. When that is the case, the generated RTMaps component will automatically expose those parameters as properties.

The `samples/model_params` sample shows an example of such models:

- Simulink Model



- RTMaps Component



Note that:

1. Not all model parameters are supported. This is a TLC-specific limitation (e.g. TLC might choose

not to make certain parameters tunable in order to generate more efficient code).

2. The following parameters are not supported and will not be exposed in the generated RTMaps component
    1. Parameters that have `width != 1`
    2. Parameters of "bus object" type
3. Model parameters are instance-specific starting from Matlab R2021a. To make sure this is enabled, in Code Mappings, set the "Data Visibility" for "Model parameter arguments" to "private". https://www.mathworks.com/matlabcentral/answers/461376-does-embedded-coder-support-instance-specific-parameters-for-top-model-when-generating-c-code In previous versions, the model parameters will be shared across component instances.

9 Setting the Value of an Input Using the Properties of Its Component

It can be useful to fix the value of an input using the properties of the generated component (instead of connecting that input to e.g. the output of a "number generator" component). To do so, you must:

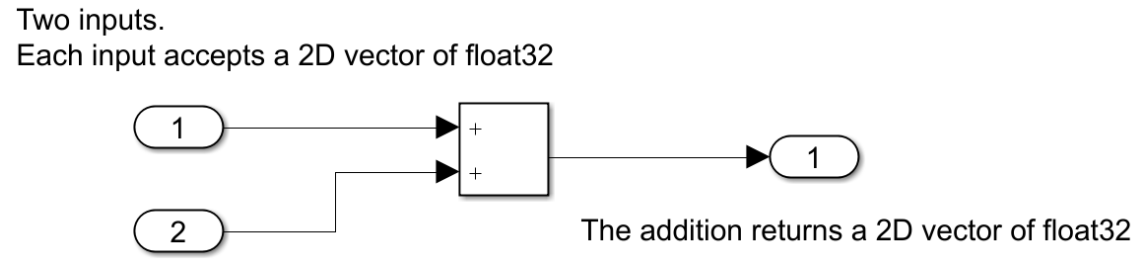
1. Make sure the model has more than 1 InPort
2. Enable the `p_force_value_of_IN_PORT_NAME` property of generated RTMaps component
3. Enter the value that you want to set for the `IN_PORT_NAME` input (or the sequence of values, seperated by spaces, in case of non-scalar input) in the `p_value_of_IN_PORT_NAME` property

Note:

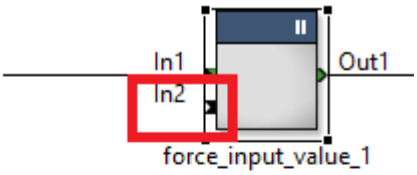
- If the component has multiple inputs, then at least one of them must read its values from an actual RTMaps component output (i.e have its `p_force_value_of_IN_PORT_NAME` property disabled). This is necessary for implementing the selected `executionMode` of the component.
- If `p_force_value_of_IN_PORT_NAME` is enabled and the `IN_PORT_NAME` input is connected, then the value that is going to be used is that of the `p_value_of_IN_PORT_NAME` property
- The number of values (separated by space) that are entered in the `p_value_of_IN_PORT_NAME` property must match the expected "Port dimensions" of the `IN_PORT_NAME` InPort of the Simulink model
- The value of an IplImage input cannot be set using the properties of the generated RTMaps component
- The value of a bus/struct input cannot be set using the properties of the generated RTMaps component

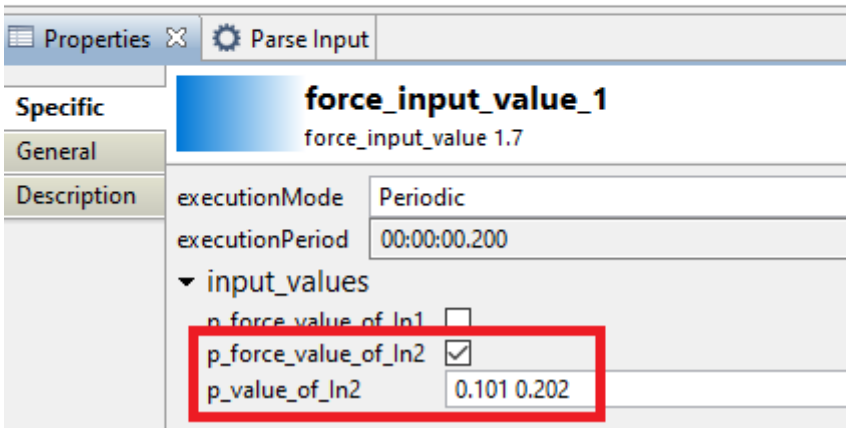
The `samples/force_input_value` sample shows an example of how to use this feature:

- Simulink Model



- RTMaps Component



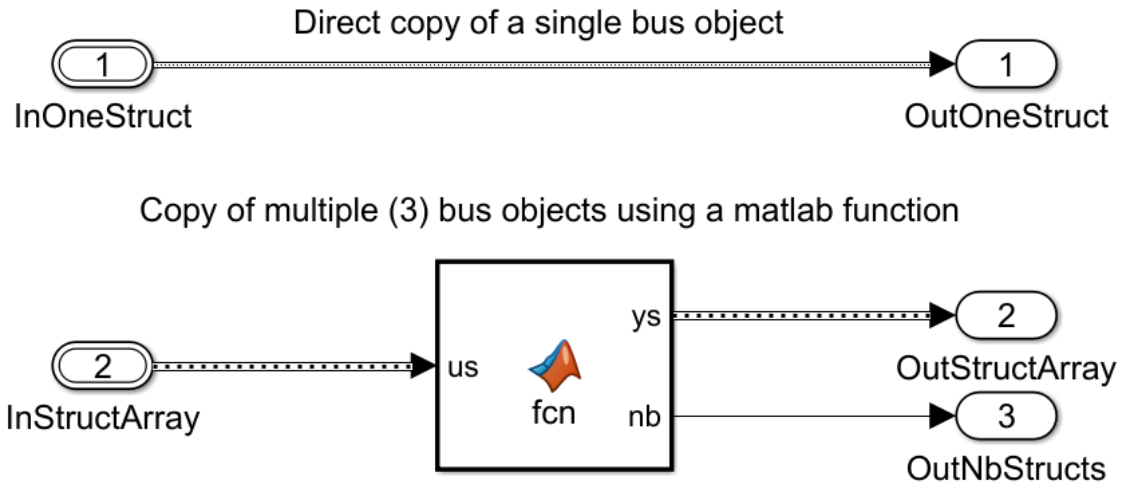


10 Handling Bus Objets (Custom Structs)

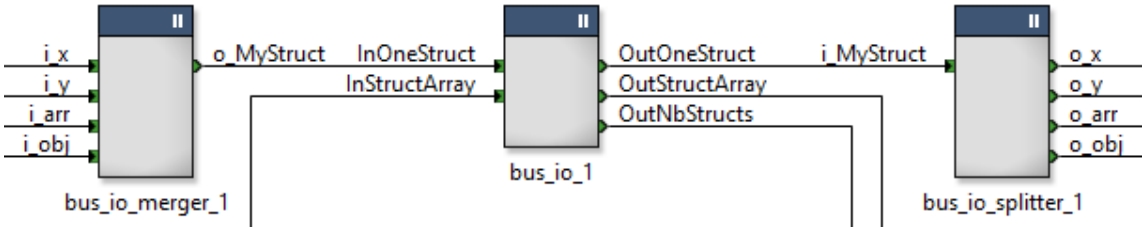
If your model uses Bus Objects for some (or all) of its InPorts and/or OutPorts, then the generated RTMaps component will use the corresponding (TLC-generated) custom C `structs` for its Inputs and/or Outputs.

The `samples/bus_io` sample shows an example of how to use this feature:

- Simulink Model

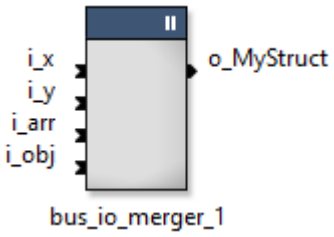


- RTMaps Components

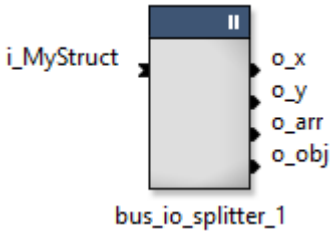


As you can see in the `samples/bus_io` sample, in addition to the usual main `<model_name>` *component*, the generated `model_name.pck` *package* will also contain 2 more components to help you in manipulating the custom structs of the main component:

- `<model_name>_merger`: generates (on the component output) custom strucs from their individual fields (on the component inputs)



- `<model_name>_splitter`: reads custom structs (on the component input) and splits them into their fields (on the component outputs)



11 Final Notes

Useful Links:

- Mathworks documentation on Simulink Coder
- Troubleshooting Simulink Coder's Compiler Issues

If you face issues with the installation or usage of the RTMaps TLC Target for Simulink, feel free to contact Intempora's technical support via our HelpDesk: support.intempora.com

12 Reference

12.1 Data Type Mapping

The following table lists the type mapping between the ports of the Simulink model and the corresponding inputs and outputs of the generated RTMaps component.

Simulink Type	RTMaps Type
`double`	`float64` (a.k.a. **`double`**)
`single`	`float32` (a.k.a. **`float`**)
`int8`	`int8`
`int16`	`int16`
`int32`	`int32`
`uint8`	`uint8`
`uint16`	`uint16`
`uint32`	`uint32`
`boolean`	`int32` *

** Please note that **`boolean`** is indeed mapped to `int32` on the I/O of the generated RTMaps component (**`true`** is mapped to `1` and **`false`** is mapped to `0`)*

As mentioned in the "New and Legacy Input/Output Data Types" article, in order to to use the "unsigned" integer types in RTMaps, it is highly recommended that you upgrade to RTMaps >= 4.5.4 (i.e. with Remote Studio).

12.2 Supported I/O Meta Information

The following table shows the list of supported I/O meta information, their data types as well as the port suffixes that must be used in order to manipulate them.

Simulink Port Name	Simulink Port Width	RTMaps Component I/O Meta Information	RTMaps Data Type
`MyPort_size`	1	Vector Size	`int32`
`MyPort_ts`	1	Timestamp (microseconds)	`int64`
`MyPort_freq`	1	Frequency (user-defined unit)	`int64`
`MyPort_qual`	1	Quality	`int32`
`MyPort_misc1`	1	Misc1	`int32`
`MyPort_misc2`	1	Misc2	`int32`
`MyPort_misc3`	1	Misc3	`int32`

Please note that

1. A valid, fixed-width port named `MyPort` must also be added to your Simulink model. The width of that port will be the “Buffer Size” of the corresponding input or output of the generated RTMaps component
2. The data types are provided as a guideline so that you can set, in Simulink, a type that can be cast (i.e. converted) properly to the RTMaps type. For instance, since Simulink inports and outports don’t support the `int64` data type, you can use `int32` or `uint32` instead and the generated RTMaps component will perform the necessary casts using the `static_cast<>()` C++ operator if necessary

Finally, a simple Simulink model implementing some of the supported meta information is available in `samples/io_meta_info`.

13 Known Limitations and Issues

13.1 Building a Model that was Created using v1.x of the RTMaps TLC Target for Simulink

In order to build a model that was originally created using the previous 1.x version of the RTMaps TLC Target for Simulink, please make sure that you change the “System Target File” property of the “Model

Configuration Parameters" window to `rtmaps.tlc` (**all lowercase letters**).

13.2 Supported Simulink Blocks for Code Generation

Not all Simulink Blocs are supported for code generation using TLC. Please refer to Mathworks' website and documentation for more details.

13.3 Supported Matlab Functions and Objects for Code Generation

The following link indicates which Matlab functions and objects are supported for code generation:

- https://www.mathworks.com/help/coder/ug/functions-and-objects-supported-for-cc-code-generation.html

13.4 Supported Compilers by Matlab and Simulink for Code Generation

The following link (under "Supported Compilers > Details" indicates which specific compiler versions are supported by Matlab and Simulink for Code Generation. Note that, from those compilers, `rtmaps.tlc` supports only a subset of the `Visual C++` versions on Windows and the `GCC` versions on Linux, as mentioned in the "Prerequisites" section above.

- https://www.mathworks.com/support/requirements/previous-releases.html

13.5 "Error: Unable to load package" when Registering the Generated Package in RTMaps

This error usually means that the generated RTMaps package depends on a shared library (typically a Matlab-provided library) that cannot be found by RTMaps. In this case, please refer to the following article for possible solutions:

- https://support.intempora.com/hc/en-us/articles/115000581313

13.6 Compile error when opening a model saved with an older version of the RTMaps TLC target

As some configuration parameters are cached into the model file, there might be code generation issues when using a model saved with an older version of the RTMaps TLC target or an older Simulink version. In that case, you have to force reloading the RTMaps code generation parameters. In the

“Model Configuration Parameters” window, under the “Code Generation” left menu item, in the “System target file” field, click “Browse”, select another TLC file, click “Apply”, and select “rtmaps.tlc” again, click “Apply” and “Ok”. Some settings may also be cached in the build folder, make sure to delete the "[ProjectName]_rtmaps_build" directory. The code should now be generated correctly.