RPIt Custom Block Creation

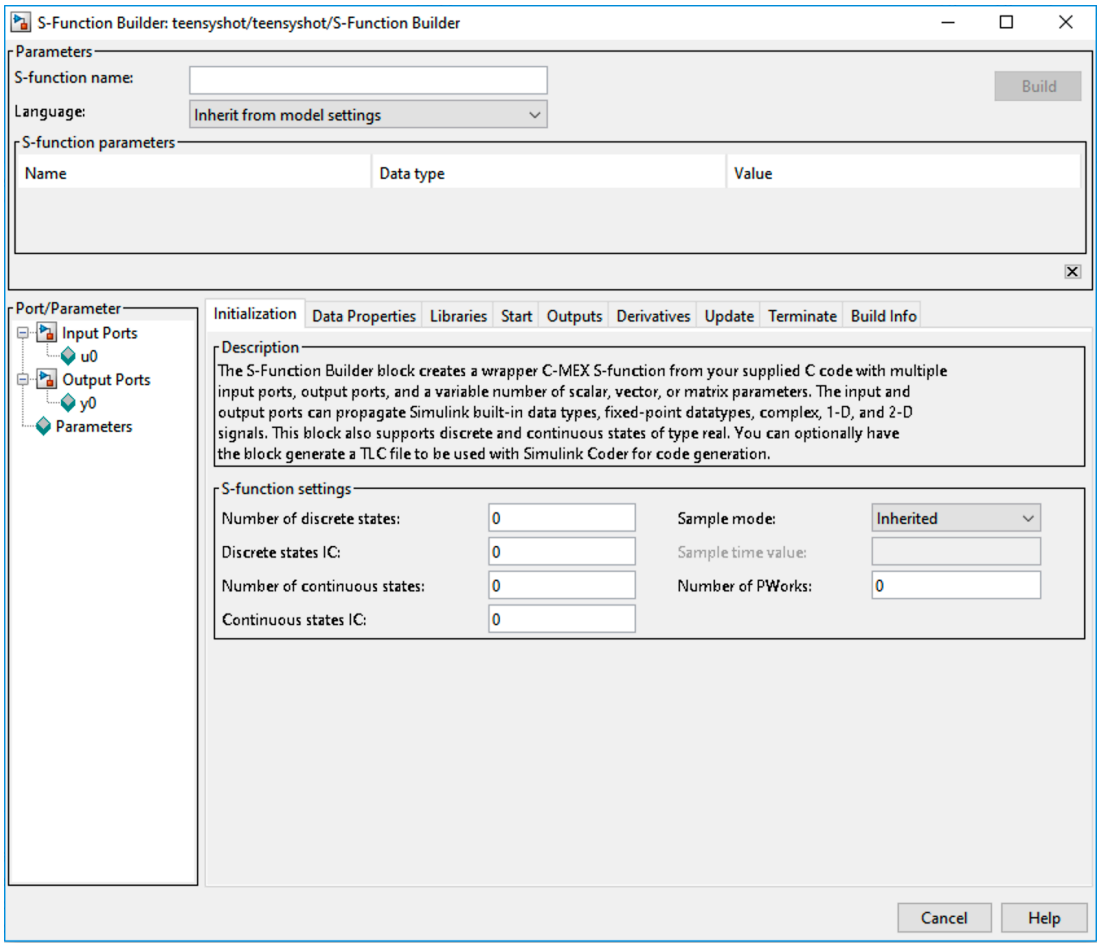


Figure 1 — S-Function Builder

In this tutorial we will learn how to create a custom bloc for RPIt. This is a 3-step procedure:

1. Definition of the bloc inputs, outputs and parameters with Simulink S-Function Builder.
2. Definition of the bloc appearance, configuration and help with Simulink mask editor.
3. Implementation of the Start, Terminate and Update methods of the block using your favorite code editor.

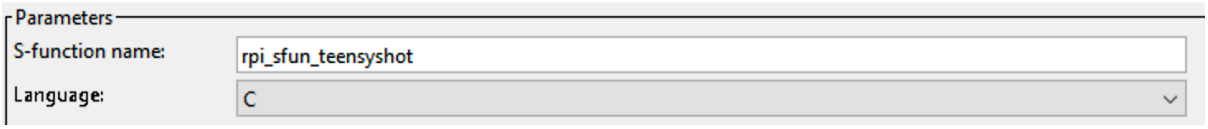
S-Function Builder

Definition of the bloc inputs, outputs and parameters

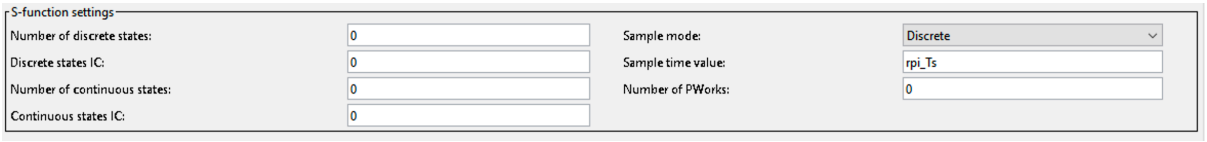
1. Create a blank Simulink document.

2. Drag in this document an S-Function Builder block and double-click it (Fig. 1).

3. Define the name of the S-function prefixed with “rpi_sfun_” and the programming language:

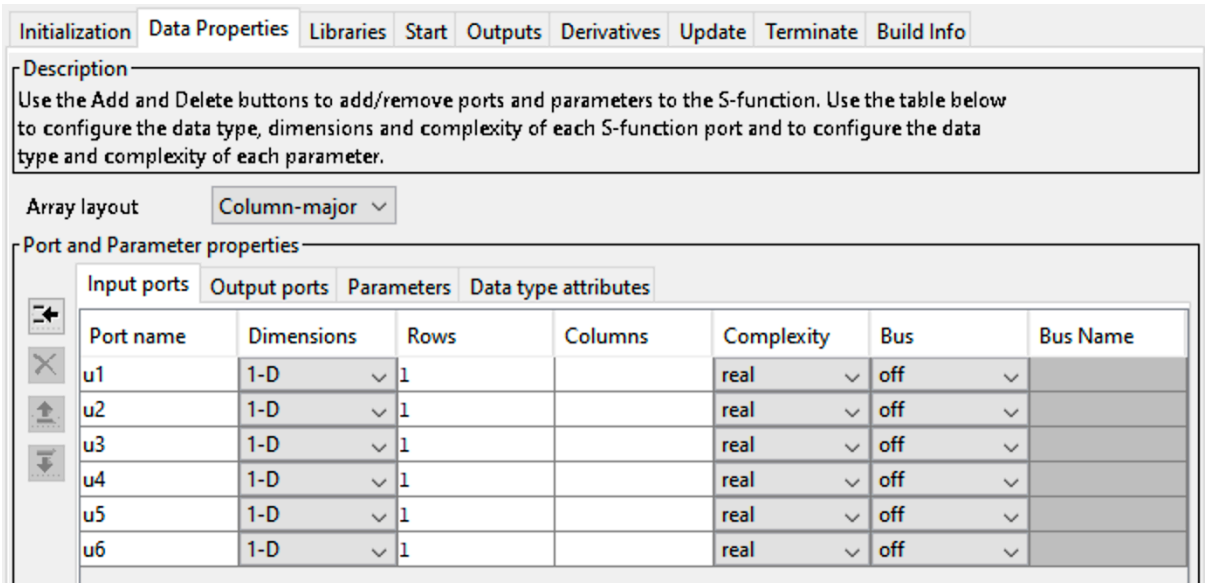


4. Define the sample mode and the sample time value as “rpi_Ts”:

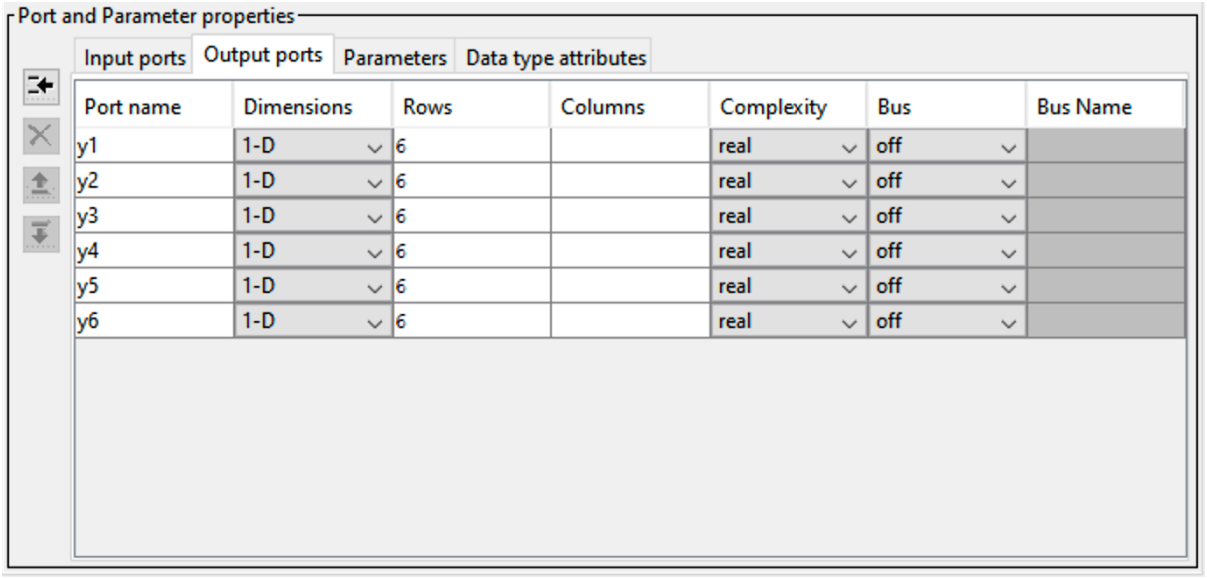


5. All defined parameters will appear in the “S-Function parameters” list. In order to be able to tune the sample time, associate the variable “rpi_Ts” with a mask tunable parameter “rpi_mask_Ts”:

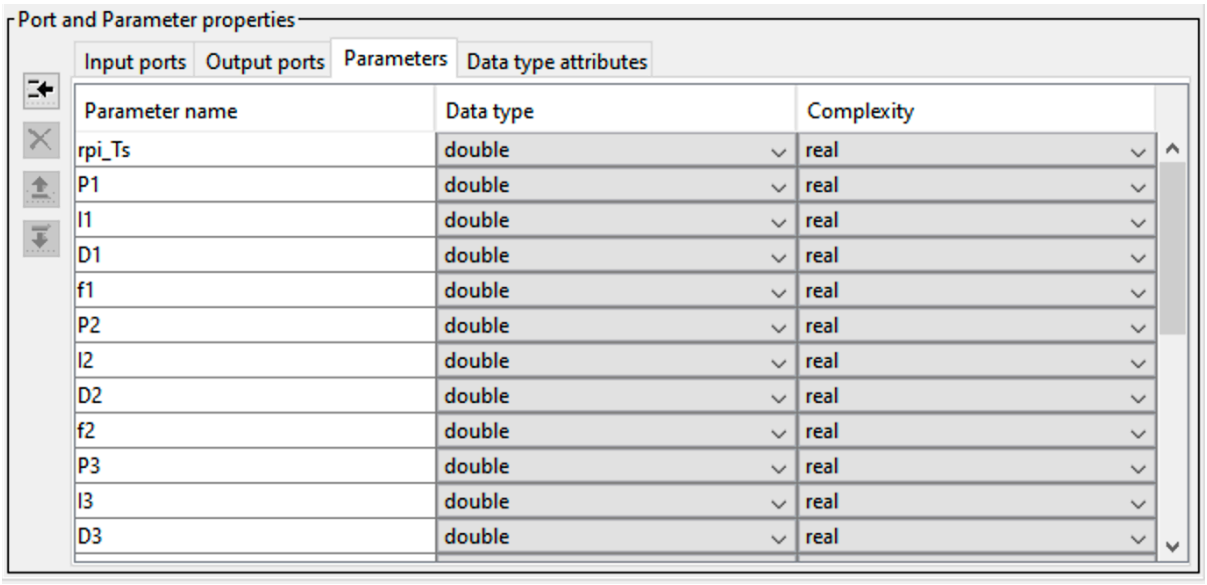
6. In the “Data properties” tab, define the input names and input types. These inputs may be vectors, in this case, define the number of rows in the “Rows” column. The port names will correspond to variable names in the automatically generated source code skeleton, so choose them carefully.



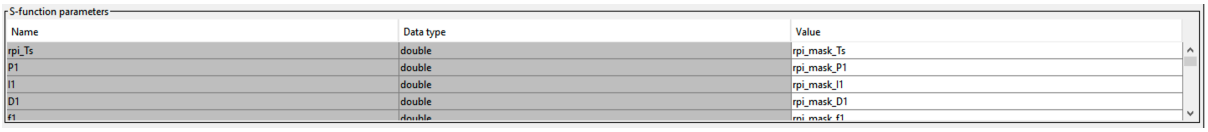
7. Define the output names and types:



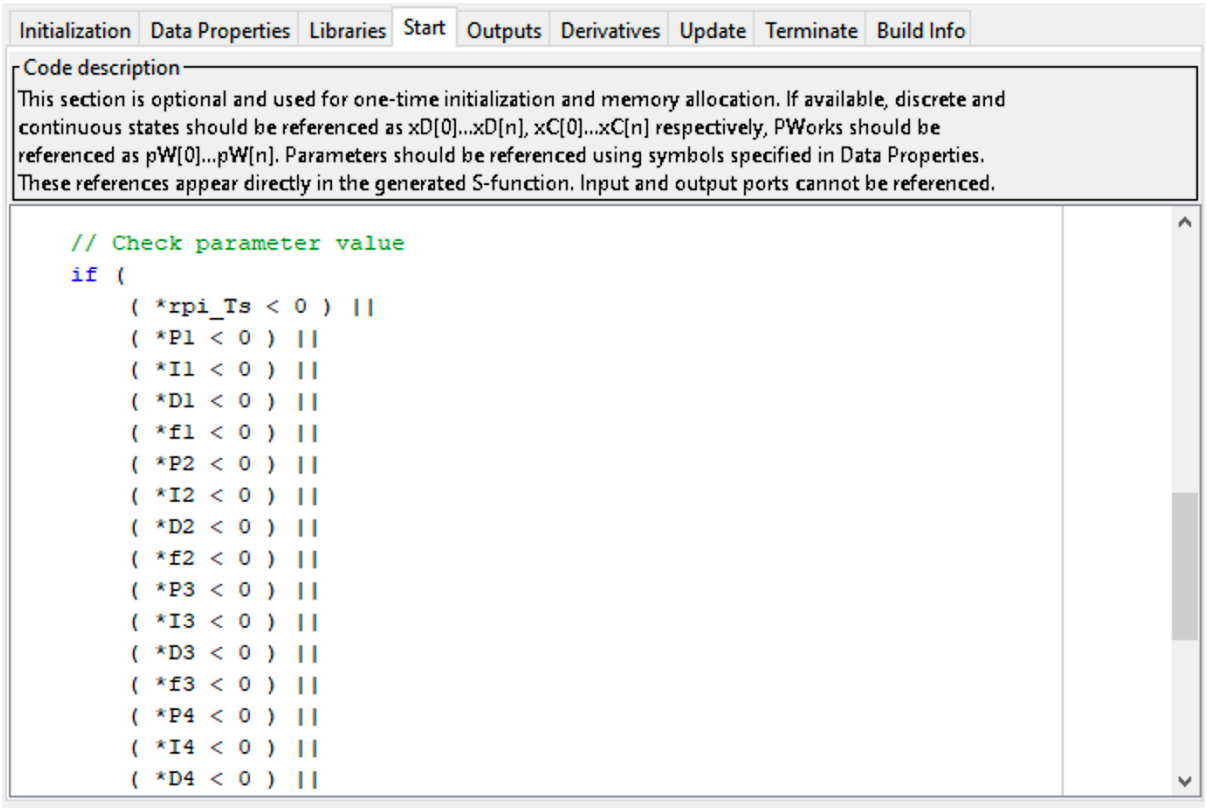
8. Define the parameters. These values (only real or integers are allowed, no text) will be tunable in the block dialog box. In external mode, all of these parameters may be tuned while the HIL experiment is running.



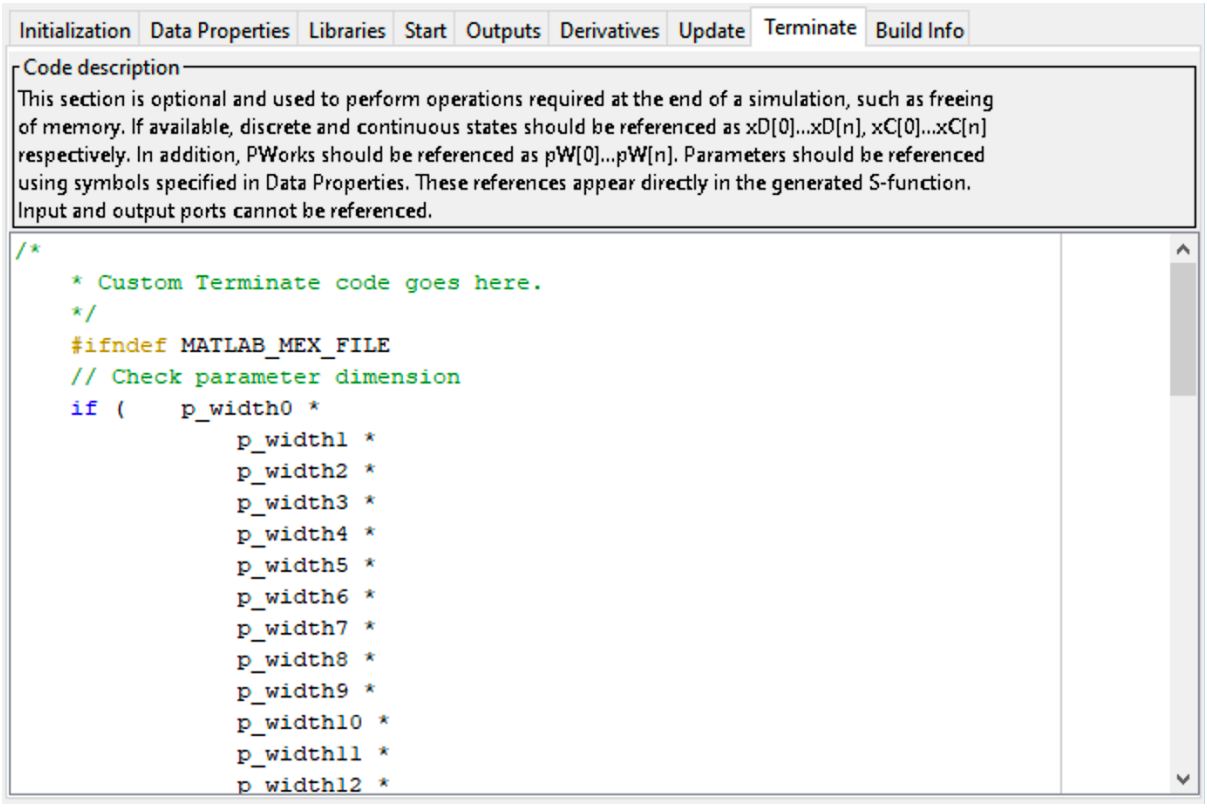
9. At each parameter defined in the previous step, give a specific name for the mask:



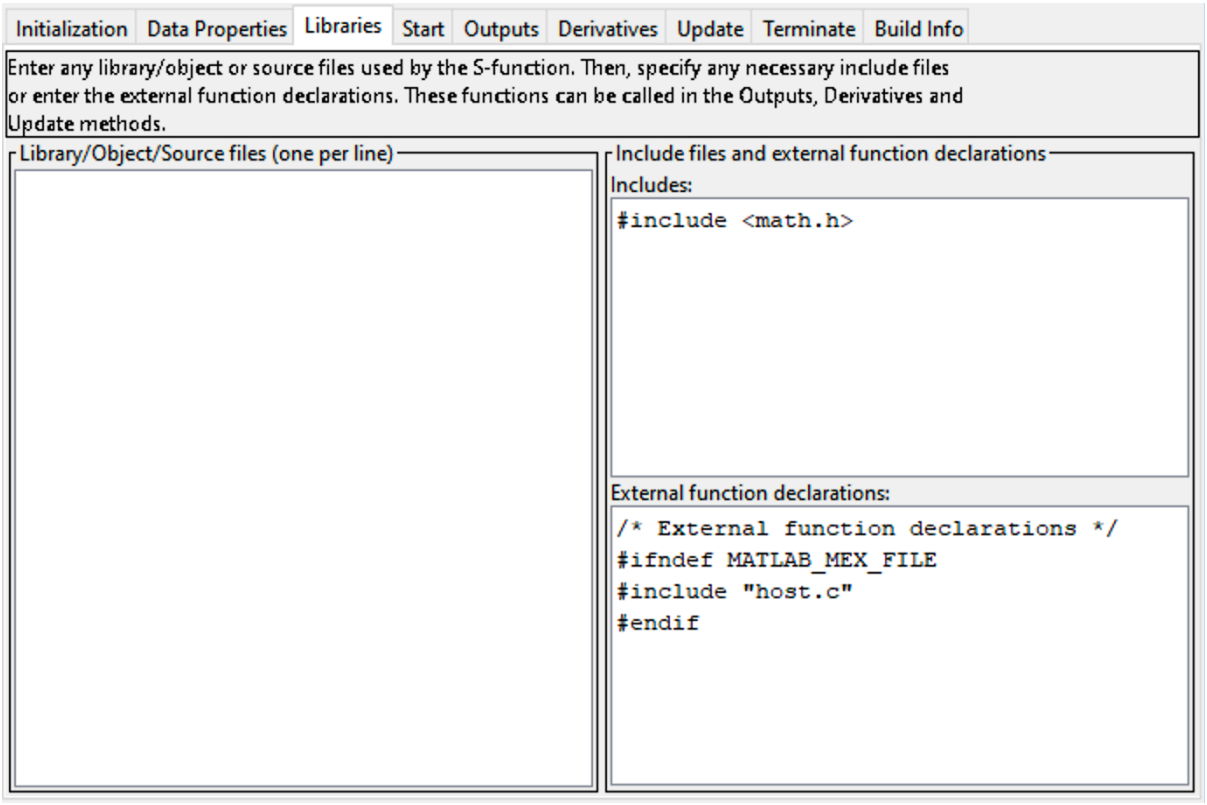
10. Define the Start method. This piece of code will be called in external mode at the launch of the experiment. You may insert a dummy line of code here to force the code generation engine to include this method. If this field is left blank, no “Start” method will be generated.



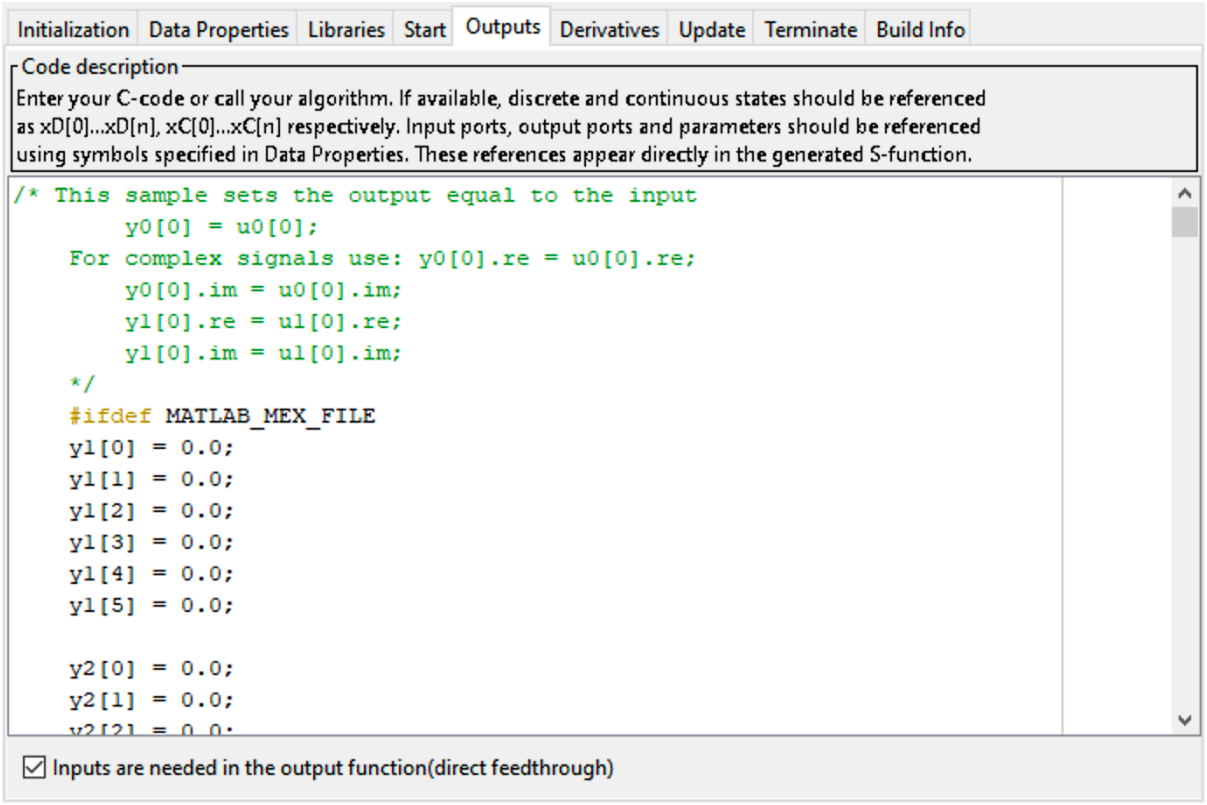
11. Define the Terminate method. Same rule as for the Start method applies.



12. In the “Libraries” tab, include all external code that should be compiled on the target. The “MATLAB_MEX_FILE” macro is used to detect if the code is compiled as a Mex file or as an external code. The Mex file should be minimalistic, it is only used by Simulink to check the type and number of inputs and outputs. For example, it is ok if the Mex file outputs “0” since this block is not relevant in simulation mode, but only in external HIL experiment mode.

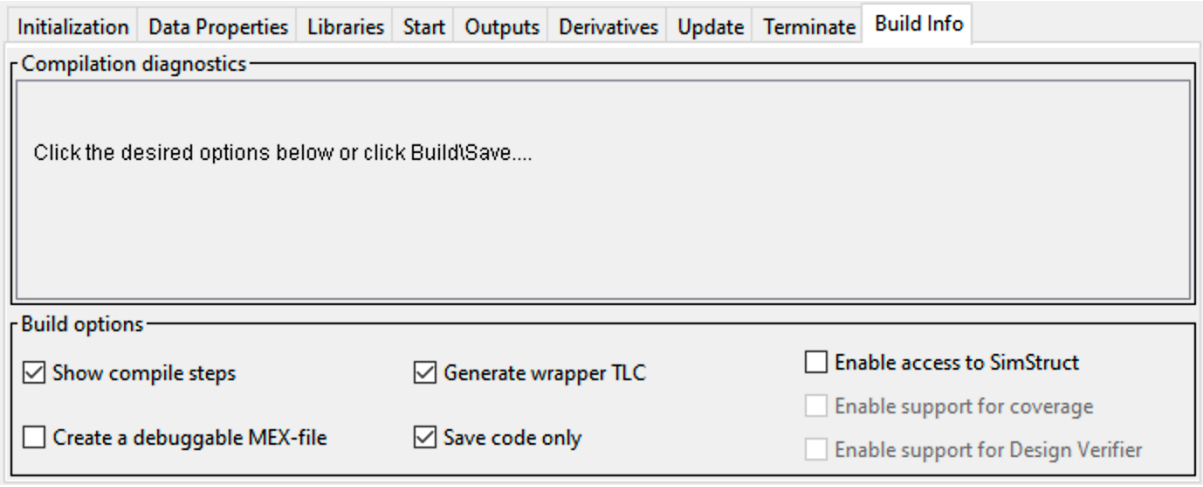


13. You may use the embedded editor to enter the update code in the “Outputs” tab. But you may also complete this step in the next stage within an IDE. This code is called at



each sample time. You should check the “Inputs are needed…” checkbox so that the update method has access to the current block inputs.

14. The derivative code is left blank. Go to the “Build info” tab and check the boxes accordingly to the following screenshot:



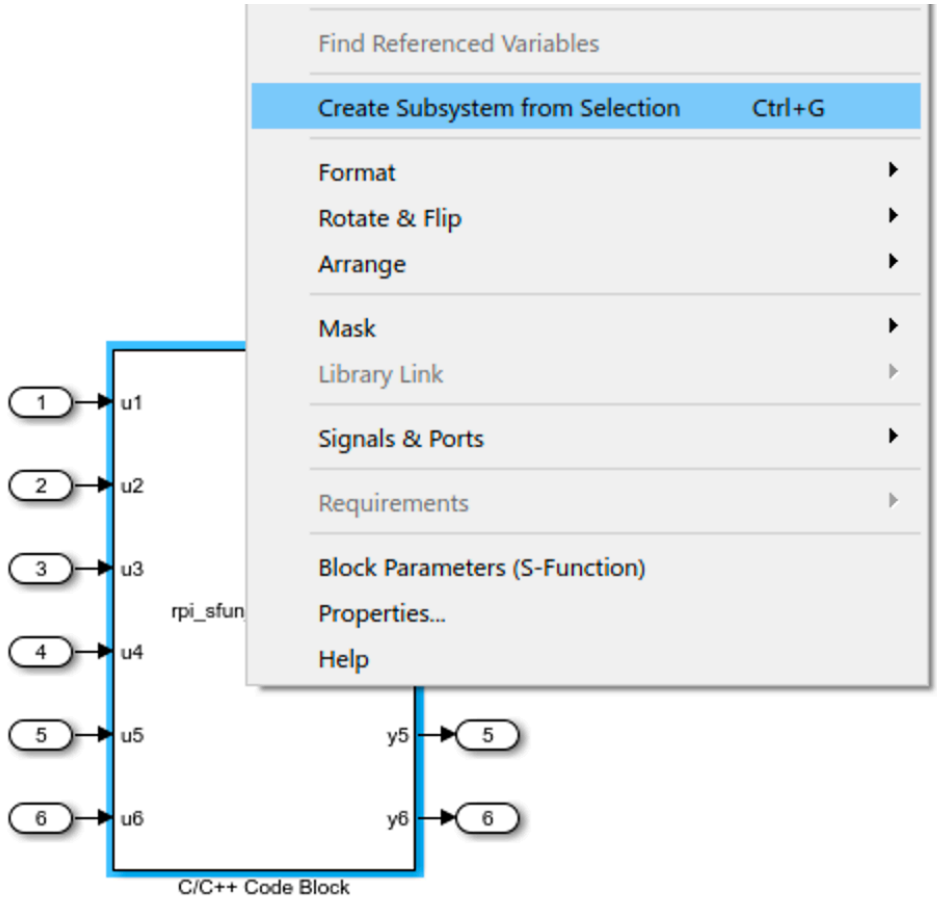
15. Click on the “Save” button. The following files will be created in the current directory:

- rpi_sfun_yourname_wrapper.c
- rpi_sfun_yourname.tlc
- rpi_sfun_yourname.c

Mask Creation

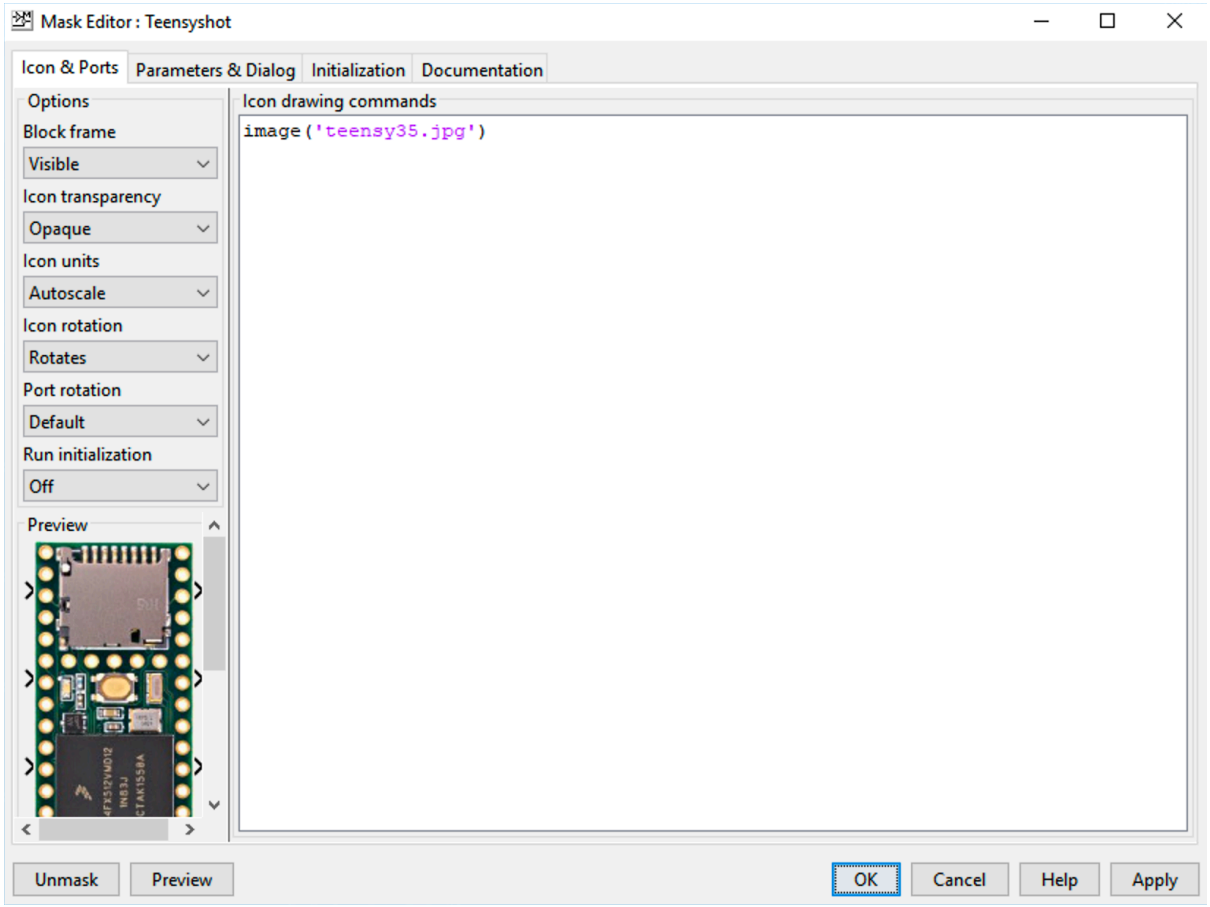
Creation of the block appearance

1. A mask cannot be applied directly to a S-Function Builder block. This block has to be first converted into a subsystem. To do so, right-click on the block and select “Create Subsystem from Selection.”

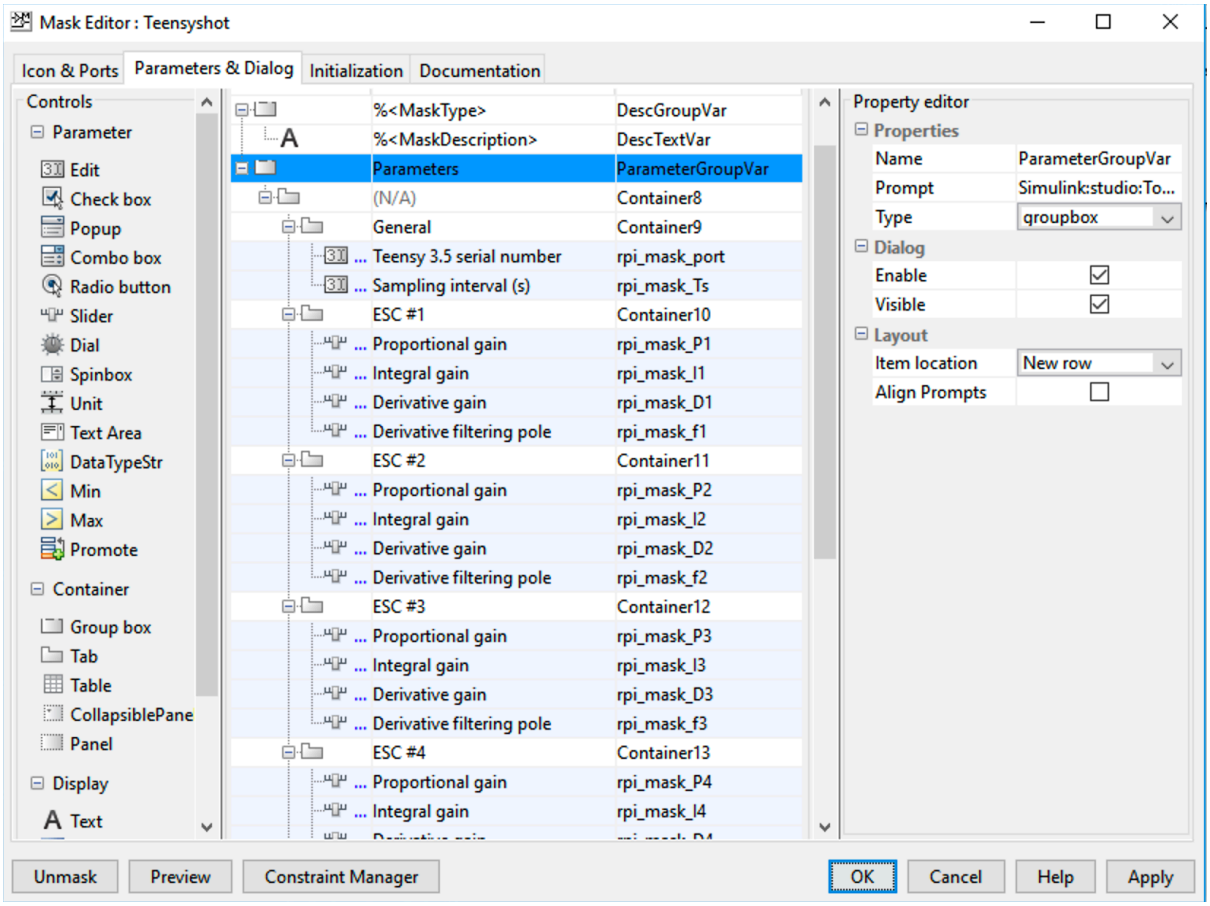


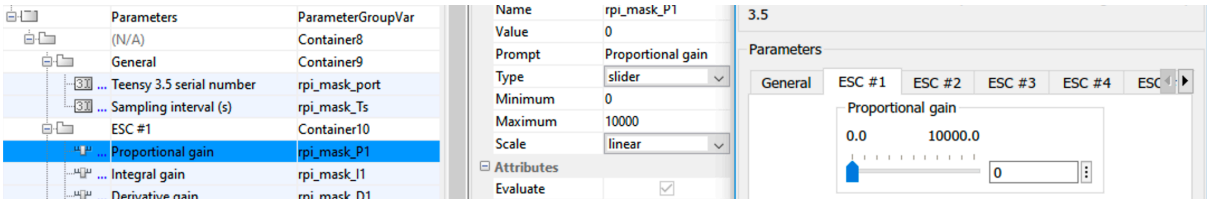
2. After that, you will be able to mask this block by right-clicking it and selecting “Create Mask”. After creating a mask, the default behavior of a double-click is to open a user-friendly dialog box that can be customized easily. The blocks under the mask is still accessible by right-clicking the block and selecting “Look under the Mask”. To edit the mask, right-click and select “Edit Mask”:

3. The first tab is dedicated to the block appearance in Simulink. Numerous customization may be made and they are well described in Matlab help. If you want to associate an image to the block, the image file should be in the same directory as the block or the block library.

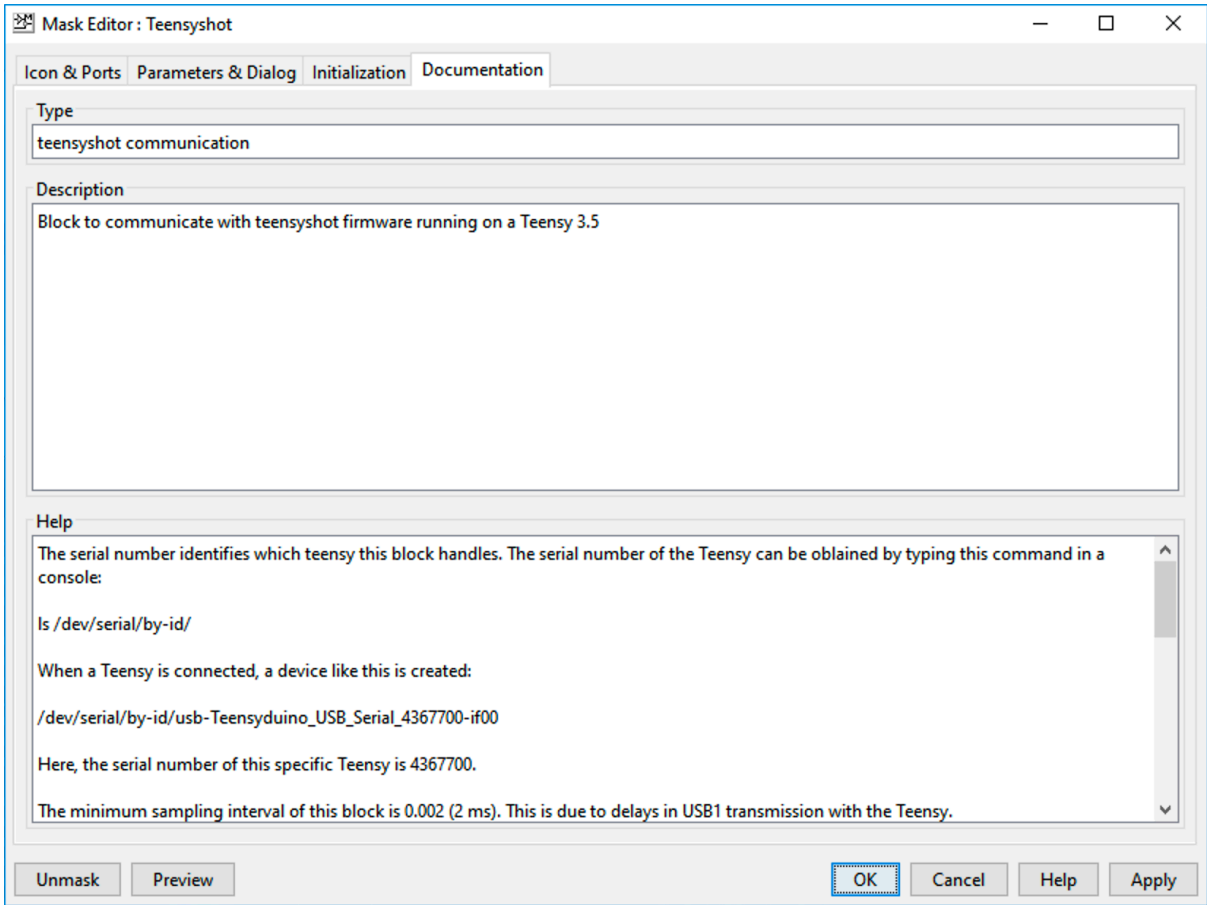


4. The second tab is the most important. It allows for the definition of the dialog box that pops up when double-clicking on the block in Simulink. You can easily preview your creation by pressing the “Preview” button. For each parameter, the parameter name should match the mask parameter name you have given to the variable in the S-Function Builder block in the previous stage. Set the “Tunable” attribute to “True” if you want to be able to tune this parameter on-the-fly.

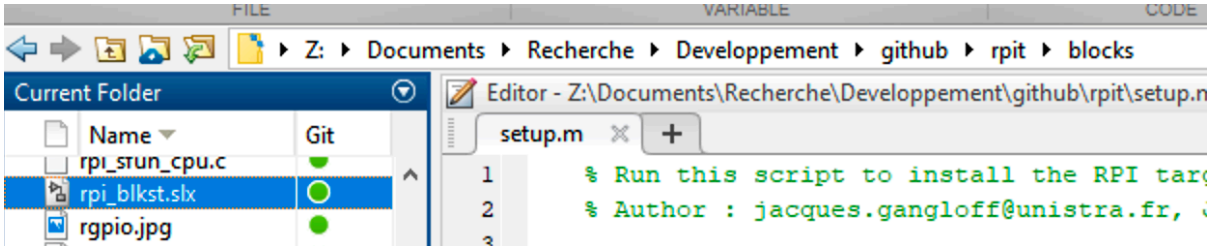




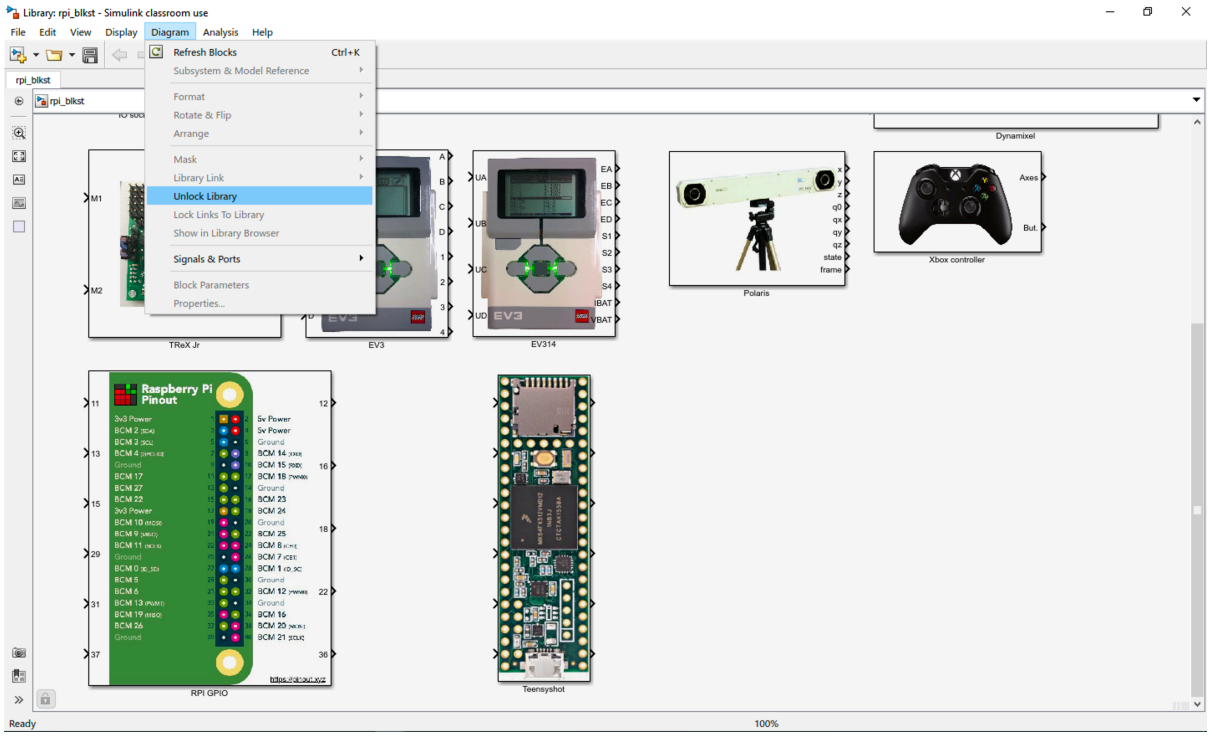
5. Edit the documentation for the block:



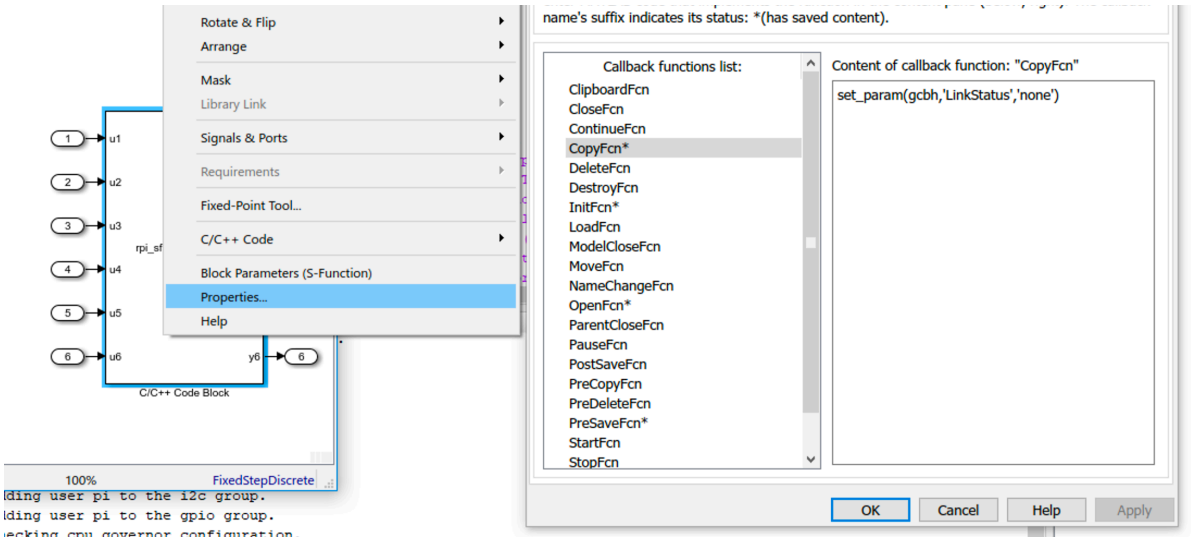
6. Close the mask editor and save the Simulink model. Now we will add the block to the RPIt library. To do so, open the file “rpi_blkst.slx.”



7. Unlock the library and drag the newly created block into the library.



8. To avoid an irrelevant warning, remove the line “set_param(gcbh,’LinkStatus’,’none')" in the “CopyFcn” callback function. This dialog box is accessible by selecting “Look under the Mask” and right-clicking on the block.



9. Save the library.

Coding the Block Methods

Implementation of the Start, Terminate and Update methods

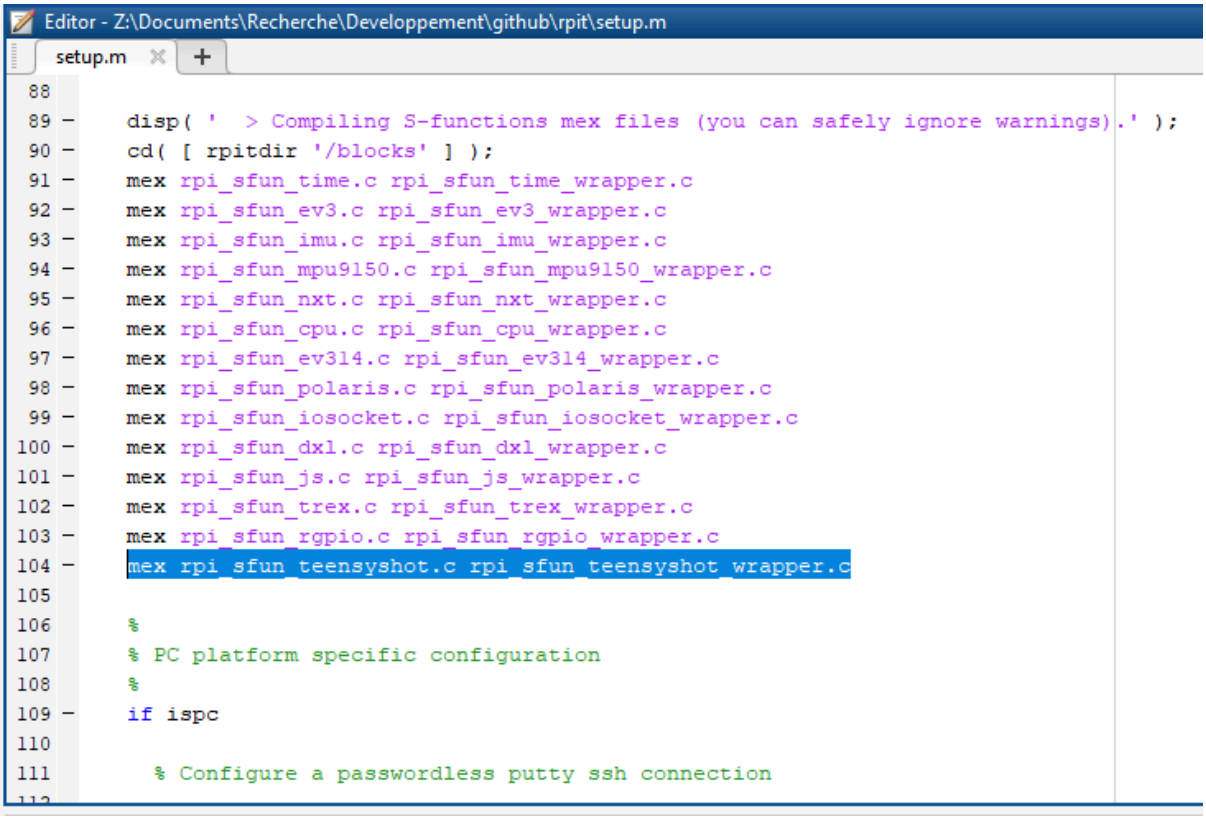
1. Copy the following files into the “blocks” directory:
   - rpi_sfun_yourname_wrapper.c
   - rpi_sfun_yourname.tlc
   - rpi_sfun_yourname.c
   - SFB__rpi_sfun_yourname__SFB.mat
   - Additional code and header files that are needed by the block.
   - If needed, the image file for the block picture.
2. Edit “rpi_sfun_yourname_wrapper.c” in your favorite code editor. This is the only file that should be edited. You will find 3 functions in this file, respectively the Start, Terminate and Update (Outputs) methods:
   - `void rpi_sfun_yourname_Start_wrapper(const real_T *rpi_Ts, const int_T p_width0, …)`
   - `void rpi_sfun_yourname_Terminate_wrapper(const real_T *rpi_Ts, const int_T p_width0, …)`
   - `void rpi_sfun_yourname_Outputs_wrapper(const real_T *u1, …);`
3. If you already edited some code within the S-Function Builder block, you should find this code into the respective functions. All the block parameters defined in the S-Function Builder are passed as parameters to the 3 functions. The tunable parameters will be updated automatically while the HIL experiment is running if the user interacts with the masked block dialog box. You may edit this code respecting those guidelines:
   - Rename all the types with the “_T” suffix. For example, rename double as real_T, uint8_t as unit8_T. This is the most portable way of defining the types.
   - In order to mute the warnings during target compilation, try to use each variable passed as parameter. For example, you could check the dimension of parameters by making some tests on the “p_width*” variable. For a scalar, “p_width*” should be equal to 1.
   - Check the minimum sampling interval: due to hardware limitations, each block has a minimum sampling interval. Document this value in the documentation.
   - Separate the mex-file code from the target code by using the “MATLAB_MEX_FILE” macro.

- The target code has no mean to inform Simulink that an error has occurred. You van use “fprintf” on the target, but stderr will be printed in a screen console. Read the RPIt quick start documentation to learn how to debug target code.

4. In the appendix, you will find a wrapper code example that could be inspiring.

5. Insert the following mex-file compilation instruction in “setup.m”:

- “mex rpi_sfun_yourname.c rpi_sfun_yourname_wrapper.c”



This will compile the mex-file of your block (only the code between defined “MATLAB_MEX_FILE” sections is compiled).

6. Rerun setup.m.

Appendix: Wrapper Code Example

```
/*
 * Include Files
 *
 */
#if defined(MATLAB_MEX_FILE)
#include "tmwtypes.h"
#include "simstruc_types.h"
#else
#include "rtwtypes.h"
#endif



/* %%%-SFUNWIZ_wrapper_includes_Changes_BEGIN --- EDIT HERE TO _END */
#include <math.h>
/* %%%-SFUNWIZ_wrapper_includes_Changes_END --- EDIT HERE TO _BEGIN */
#define u_width 1
#define y_width 1

/*
 * Create external references here.
 *
 */
/* %%%-SFUNWIZ_wrapper_externs_Changes_BEGIN --- EDIT HERE TO _END */
/* External function declarations */
#ifndef MATLAB_MEX_FILE
#include "host.c"
#endif
/* %%%-SFUNWIZ_wrapper_externs_Changes_END --- EDIT HERE TO _BEGIN */

/*
 * Start function
 *
 */
void rpi_sfun_teensyshot_Start_wrapper(const real_T *rpi_Ts, const int_T p_width0,
                       const real_T *P1, const int_T p_width1,
                       const real_T *I1, const int_T p_width2,
                       const real_T *D1, const int_T p_width3,
                       const real_T *f1, const int_T p_width4,
                       const real_T *P2, const int_T p_width5,
                       const real_T *I2, const int_T p_width6,
                       const real_T *D2, const int_T p_width7,
                       const real_T *f2, const int_T p_width8,
                       const real_T *P3, const int_T p_width9,
                       const real_T *I3, const int_T p_width10,
                       const real_T *D3, const int_T p_width11,
                       const real_T *f3, const int_T p_width12,
                       const real_T *P4, const int_T p_width13,
                       const real_T *I4, const int_T p_width14,
                       const real_T *D4, const int_T p_width15,
                       const real_T *f4, const int_T p_width16,
```

```
                    const real_T *P5, const int_T p_width17,
                    const real_T *I5, const int_T p_width18,
                    const real_T *D5, const int_T p_width19,
                    const real_T *f5, const int_T p_width20,
                    const real_T *P6, const int_T p_width21,
                    const real_T *I6, const int_T p_width22,
                    const real_T *D6, const int_T p_width23,
                    const real_T *f6, const int_T p_width24,
                    const uint32_T *Port, const int_T p_width25)
{
/* %%%-SFUNWIZ_wrapper_Start_Changes_BEGIN --- EDIT HERE TO _END */
 /*
        * Custom Start code goes here.
        */
        #ifndef MATLAB_MEX_FILE
        // Check parameter dimension
        if (    p_width0 *
                                p_width1 *
                                p_width2 *
                                p_width3 *
                                p_width4 *
                                p_width5 *
                                p_width6 *
                                p_width7 *
                                p_width8 *
                                p_width9 *
                                p_width10 *
                                p_width11 *
                                p_width12 *
                                p_width13 *
                                p_width14 *
                                p_width15 *
                                p_width16 *
                                p_width17 *
                                p_width18 *
                                p_width19 *
                                p_width20 *
                                p_width21 *
                                p_width22 *
                                p_width23 *
                                p_width24 *
                                p_width25 != 1 )
                return;

        // Check parameter value
        if (
                ( *rpi_Ts < 0 ) ||
                ( *P1 < 0 ) ||
                ( *I1 < 0 ) ||
                ( *D1 < 0 ) ||
                ( *f1 < 0 ) ||
                ( *P2 < 0 ) ||
                ( *I2 < 0 ) ||
                ( *D2 < 0 ) ||
```

```
                ( *f2 < 0 ) ||
                ( *P3 < 0 ) ||
                ( *I3 < 0 ) ||
                ( *D3 < 0 ) ||
                ( *f3 < 0 ) ||
                ( *P4 < 0 ) ||
                ( *I4 < 0 ) ||
                ( *D4 < 0 ) ||
                ( *f4 < 0 ) ||
                ( *P5 < 0 ) ||
                ( *I5 < 0 ) ||
                ( *D5 < 0 ) ||
                ( *f5 < 0 ) ||
                ( *P6 < 0 ) ||
                ( *I6 < 0 ) ||
                ( *D6 < 0 ) ||
                ( *f6 < 0 ) ||
                ( *Port == 0 )
                )
                return;

        // Open serial port
        Host_init_port( *Port );
        #endif
/* %%%-SFUNWIZ_wrapper_Start_Changes_END --- EDIT HERE TO _BEGIN */
}
/*
 * Output function
 *
 */
void rpi_sfun_teensyshot_Outputs_wrapper(const real_T *u1,
                        const real_T *u2,
                        const real_T *u3,
                        const real_T *u4,
                        const real_T *u5,
                        const real_T *u6,
                        real_T *y1,
                        real_T *y2,
                        real_T *y3,
                        real_T *y4,
                        real_T *y5,
                        real_T *y6,
                        const real_T *rpi_Ts, const int_T p_width0,
                        const real_T *P1, const int_T p_width1,
                        const real_T *I1, const int_T p_width2,
                        const real_T *D1, const int_T p_width3,
                        const real_T *f1, const int_T p_width4,
                        const real_T *P2, const int_T p_width5,
                        const real_T *I2, const int_T p_width6,
                        const real_T *D2, const int_T p_width7,
                        const real_T *f2, const int_T p_width8,
                        const real_T *P3, const int_T p_width9,
                        const real_T *I3, const int_T p_width10,
                        const real_T *D3, const int_T p_width11,
```

```
                        const real_T *f3, const int_T p_width12,
                        const real_T *P4, const int_T p_width13,
                        const real_T *I4, const int_T p_width14,
                        const real_T *D4, const int_T p_width15,
                        const real_T *f4, const int_T p_width16,
                        const real_T *P5, const int_T p_width17,
                        const real_T *I5, const int_T p_width18,
                        const real_T *D5, const int_T p_width19,
                        const real_T *f5, const int_T p_width20,
                        const real_T *P6, const int_T p_width21,
                        const real_T *I6, const int_T p_width22,
                        const real_T *D6, const int_T p_width23,
                        const real_T *f6, const int_T p_width24,
                        const uint32_T *Port, const int_T p_width25)
{
/* %%%-SFUNWIZ_wrapper_Outputs_Changes_BEGIN --- EDIT HERE TO _END */
 /* This sample sets the output equal to the input
               y0[0] = u0[0];
       For complex signals use: y0[0].re = u0[0].re;
               y0[0].im = u0[0].im;
               y1[0].re = u1[0].re;
               y1[0].im = u1[0].im;
       */
       #ifdef MATLAB_MEX_FILE
       y1[0] = 0.0;
       y1[1] = 0.0;
       y1[2] = 0.0;
       y1[3] = 0.0;
       y1[4] = 0.0;
       y1[5] = 0.0;

       y2[0] = 0.0;
       y2[1] = 0.0;
       y2[2] = 0.0;
       y2[3] = 0.0;
       y2[4] = 0.0;
       y2[5] = 0.0;

       y3[0] = 0.0;
       y3[1] = 0.0;
       y3[2] = 0.0;
       y3[3] = 0.0;
       y3[4] = 0.0;
       y3[5] = 0.0;

       y4[0] = 0.0;
       y4[1] = 0.0;
       y4[2] = 0.0;
       y4[3] = 0.0;
       y4[4] = 0.0;
       y4[5] = 0.0;

       y5[0] = 0.0;
       y5[1] = 0.0;
```

```
y5[2] = 0.0;
y5[3] = 0.0;
y5[4] = 0.0;
y5[5] = 0.0;

y6[0] = 0.0;
y6[1] = 0.0;
y6[2] = 0.0;
y6[3] = 0.0;
y6[4] = 0.0;
y6[5] = 0.0;
#else
int                 i, ret;
int16_T             RPM_r[ESCPID_MAX_ESC] = { 0 };
uint16_T            PID_P[ESCPID_MAX_ESC] = { 0 };
uint16_T            PID_I[ESCPID_MAX_ESC] = { 0 };
uint16_T            PID_D[ESCPID_MAX_ESC] = { 0 };
uint16_T            PID_f[ESCPID_MAX_ESC] = { 0 };
ESCPIDcomm_struct_t *comm;

// Check parameter dimension
if (    p_width0 *
                      p_width1 *
                      p_width2 *
                      p_width3 *
                      p_width4 *
                      p_width5 *
                      p_width6 *
                      p_width7 *
                      p_width8 *
                      p_width9 *
                      p_width10 *
                      p_width11 *
                      p_width12 *
                      p_width13 *
                      p_width14 *
                      p_width15 *
                      p_width16 *
                      p_width17 *
                      p_width18 *
                      p_width19 *
                      p_width20 *
                      p_width21 *
                      p_width22 *
                      p_width23 *
                      p_width24 *
                      p_width25 != 1 )
        return;

// Check parameter value
if (
        ( *rpi_Ts < 0 ) ||
        ( *P1 < 0 ) ||
        ( *I1 < 0 ) ||
```

```
        ( *D1 < 0 ) ||
        ( *f1 < 0 ) ||
        ( *P2 < 0 ) ||
        ( *I2 < 0 ) ||
        ( *D2 < 0 ) ||
        ( *f2 < 0 ) ||
        ( *P3 < 0 ) ||
        ( *I3 < 0 ) ||
        ( *D3 < 0 ) ||
        ( *f3 < 0 ) ||
        ( *P4 < 0 ) ||
        ( *I4 < 0 ) ||
        ( *D4 < 0 ) ||
        ( *f4 < 0 ) ||
        ( *P5 < 0 ) ||
        ( *I5 < 0 ) ||
        ( *D5 < 0 ) ||
        ( *f5 < 0 ) ||
        ( *P6 < 0 ) ||
        ( *I6 < 0 ) ||
        ( *D6 < 0 ) ||
        ( *f6 < 0 ) ||
        ( *Port == 0 )
        )
        return;

// Check for errors
if ( ( p_width0 > 1 ) || ( *rpi_Ts < 0.002 ) )      {
        y1[0] = 0.0;
        y1[1] = 0.0;
        y1[2] = 0.0;
        y1[3] = 0.0;
        y1[4] = 0.0;
        y1[5] = 0.0;

        y2[0] = 0.0;
        y2[1] = 0.0;
        y2[2] = 0.0;
        y2[3] = 0.0;
        y2[4] = 0.0;
        y2[5] = 0.0;

        y3[0] = 0.0;
        y3[1] = 0.0;
        y3[2] = 0.0;
        y3[3] = 0.0;
        y3[4] = 0.0;
        y3[5] = 0.0;

        y4[0] = 0.0;
        y4[1] = 0.0;
        y4[2] = 0.0;
        y4[3] = 0.0;
        y4[4] = 0.0;
```

```
        y4[5] = 0.0;

        y5[0] = 0.0;
        y5[1] = 0.0;
        y5[2] = 0.0;
        y5[3] = 0.0;
        y5[4] = 0.0;
        y5[5] = 0.0;

        y6[0] = 0.0;
        y6[1] = 0.0;
        y6[2] = 0.0;
        y6[3] = 0.0;
        y6[4] = 0.0;
        y6[5] = 0.0;

        return;
}

 // Initialize tunable PID data
for ( i = 0; i < ESCPID_MAX_ESC; i++ )  {
        switch( i )    {
                case 0:
                        RPM_r[i] = (int16_T)*u1;
                        PID_P[i] = (uint16_T)*P1;
                        PID_I[i] = (uint16_T)*I1;
                        PID_D[i] = (uint16_T)*D1;
                        PID_f[i] = (uint16_T)*f1;
                break;
                case 1:
                        RPM_r[i] = (int16_T)*u2;
                        PID_P[i] = (uint16_T)*P2;
                        PID_I[i] = (uint16_T)*I2;
                        PID_D[i] = (uint16_T)*D2;
                        PID_f[i] = (uint16_T)*f2;
                break;
                case 2:
                        RPM_r[i] = (int16_T)*u3;
                        PID_P[i] = (uint16_T)*P3;
                        PID_I[i] = (uint16_T)*I3;
                        PID_D[i] = (uint16_T)*D3;
                        PID_f[i] = (uint16_T)*f3;
                break;
                case 3:
                        RPM_r[i] = (int16_T)*u4;
                        PID_P[i] = (uint16_T)*P4;
                        PID_I[i] = (uint16_T)*I4;
                        PID_D[i] = (uint16_T)*D4;
                        PID_f[i] = (uint16_T)*f4;
                break;
                case 4:
                        RPM_r[i] = (int16_T)*u5;
                        PID_P[i] = (uint16_T)*P5;
                        PID_I[i] = (uint16_T)*I5;
```

```
                        PID_D[i] = (uint16_T)*D5;
                        PID_f[i] = (uint16_T)*f5;
                break;
                case 5:
                        RPM_r[i] = (int16_T)*u6;
                        PID_P[i] = (uint16_T)*P6;
                        PID_I[i] = (uint16_T)*I6;
                        PID_D[i] = (uint16_T)*D6;
                        PID_f[i] = (uint16_T)*f6;
                break;
                default:
                break;
        }
}

// Roundtrip USB communication with teensy
ret = Host_comm_update(       *Port,
                                                        RPM_r,
                                                        PID_P,
                                                        PID_I,
                                                        PID_D,
                                                        PID_f,
                                                        &comm );

// Update return values if no errors
if ( !ret )    {
        for ( i = 0; i < ESCPID_MAX_ESC; i++ )  {
                switch( i )    {
                        case 0:
                                y1[0] = (real_T)comm->err[i];
                                y1[1] = (real_T)comm->deg[i];
                                y1[2] = (real_T)comm->cmd[i];
                                y1[3] = (real_T)comm->volt[i];
                                y1[4] = (real_T)comm->amp[i];
                                y1[5] = (real_T)comm->rpm[i];
                        break;
                        case 1:
                                y2[0] = (real_T)comm->err[i];
                                y2[1] = (real_T)comm->deg[i];
                                y2[2] = (real_T)comm->cmd[i];
                                y2[3] = (real_T)comm->volt[i];
                                y2[4] = (real_T)comm->amp[i];
                                y2[5] = (real_T)comm->rpm[i];
                        break;
                        case 2:
                                y3[0] = (real_T)comm->err[i];
                                y3[1] = (real_T)comm->deg[i];
                                y3[2] = (real_T)comm->cmd[i];
                                y3[3] = (real_T)comm->volt[i];
                                y3[4] = (real_T)comm->amp[i];
                                y3[5] = (real_T)comm->rpm[i];
                        break;
                        case 3:
                                y4[0] = (real_T)comm->err[i];
```

```
                                        y4[1] = (real_T)comm->deg[i];
                                        y4[2] = (real_T)comm->cmd[i];
                                        y4[3] = (real_T)comm->volt[i];
                                        y4[4] = (real_T)comm->amp[i];
                                        y4[5] = (real_T)comm->rpm[i];
                                break;
                                case 4:
                                        y5[0] = (real_T)comm->err[i];
                                        y5[1] = (real_T)comm->deg[i];
                                        y5[2] = (real_T)comm->cmd[i];
                                        y5[3] = (real_T)comm->volt[i];
                                        y5[4] = (real_T)comm->amp[i];
                                        y5[5] = (real_T)comm->rpm[i];
                                break;
                                case 5:
                                        y6[0] = (real_T)comm->err[i];
                                        y6[1] = (real_T)comm->deg[i];
                                        y6[2] = (real_T)comm->cmd[i];
                                        y6[3] = (real_T)comm->volt[i];
                                        y6[4] = (real_T)comm->amp[i];
                                        y6[5] = (real_T)comm->rpm[i];
                                break;
                                default:
                                break;
                        }
                }
        }
        #endif
/* %%%-SFUNWIZ_wrapper_Outputs_Changes_END --- EDIT HERE TO _BEGIN */
}

/*
 * Terminate function
 *
 */
void rpi_sfun_teensyshot_Terminate_wrapper(const real_T *rpi_Ts, const int_T p_width0,
                        const real_T *P1, const int_T p_width1,
                        const real_T *I1, const int_T p_width2,
                        const real_T *D1, const int_T p_width3,
                        const real_T *f1, const int_T p_width4,
                        const real_T *P2, const int_T p_width5,
                        const real_T *I2, const int_T p_width6,
                        const real_T *D2, const int_T p_width7,
                        const real_T *f2, const int_T p_width8,
                        const real_T *P3, const int_T p_width9,
                        const real_T *I3, const int_T p_width10,
                        const real_T *D3, const int_T p_width11,
                        const real_T *f3, const int_T p_width12,
                        const real_T *P4, const int_T p_width13,
                        const real_T *I4, const int_T p_width14,
                        const real_T *D4, const int_T p_width15,
                        const real_T *f4, const int_T p_width16,
                        const real_T *P5, const int_T p_width17,
                        const real_T *I5, const int_T p_width18,
```

```
                    const real_T *D5, const int_T p_width19,
                    const real_T *f5, const int_T p_width20,
                    const real_T *P6, const int_T p_width21,
                    const real_T *I6, const int_T p_width22,
                    const real_T *D6, const int_T p_width23,
                    const real_T *f6, const int_T p_width24,
                    const uint32_T *Port, const int_T p_width25)
{
/* %%%-SFUNWIZ_wrapper_Terminate_Changes_BEGIN --- EDIT HERE TO _END */
 /*
        * Custom Terminate code goes here.
        */
        #ifndef MATLAB_MEX_FILE
        // Check parameter dimension
        if (    p_width0 *
                                p_width1 *
                                p_width2 *
                                p_width3 *
                                p_width4 *
                                p_width5 *
                                p_width6 *
                                p_width7 *
                                p_width8 *
                                p_width9 *
                                p_width10 *
                                p_width11 *
                                p_width12 *
                                p_width13 *
                                p_width14 *
                                p_width15 *
                                p_width16 *
                                p_width17 *
                                p_width18 *
                                p_width19 *
                                p_width20 *
                                p_width21 *
                                p_width22 *
                                p_width23 *
                                p_width24 *
                                p_width25 != 1 )
                return;

        // Check parameter value
        if (
                ( *rpi_Ts < 0 ) ||
                ( *P1 < 0 ) ||
                ( *I1 < 0 ) ||
                ( *D1 < 0 ) ||
                ( *f1 < 0 ) ||
                ( *P2 < 0 ) ||
                ( *I2 < 0 ) ||
                ( *D2 < 0 ) ||
                ( *f2 < 0 ) ||
                ( *P3 < 0 ) ||
```

```
                ( *I3 < 0 ) ||
                ( *D3 < 0 ) ||
                ( *f3 < 0 ) ||
                ( *P4 < 0 ) ||
                ( *I4 < 0 ) ||
                ( *D4 < 0 ) ||
                ( *f4 < 0 ) ||
                ( *P5 < 0 ) ||
                ( *I5 < 0 ) ||
                ( *D5 < 0 ) ||
                ( *f5 < 0 ) ||
                ( *P6 < 0 ) ||
                ( *I6 < 0 ) ||
                ( *D6 < 0 ) ||
                ( *f6 < 0 ) ||
                ( *Port == 0 )
                )
                return;

        // Close serial port
        Host_release_port( *Port );
        #endif
/* %%%-SFUNWIZ_wrapper_Terminate_Changes_END --- EDIT HERE TO _BEGIN */
}
```