1.实验名称及目的

1.1实验名称

Simulink调用C代码实验（基于MATLAB coder）

1.2实验目的

通过该例程介绍Simulink模型调用已有的外部C代码的几种方法。MATLAB的代码生成器（Coder）允许将MATLAB代码转换为C/C++代码。这对于性能要求高的应用（如嵌入式系统）特别有用，因为C/C++代码通常比MATLAB代码运行得更快。同时，通过调用外部C函数，可以利用已有的高效C代码库或者专门优化的C函数，从而提升计算性能。此外，这种方法还可以复用现有的C代码，避免重写相同的逻辑。

1.3关键知识点

将C代码嵌入Simulink仿真

Simulink可以利用已有的C代码实现一些复杂或特定的功能，从而提高仿真的效率和准确性。例如，可以通过调用外部C代码实现一些数学函数、信号处理算法、硬件接口等。这样，不仅可以避免重复编写相同的逻辑，还可以方便地将生成的代码部署到嵌入式系统中。simulink提供了多种方式来调用外部C代码，如使用S-Function Builder模块、编写自定义的S函数、使用Legacy Code Tool或者使用Embedded Coder。

Simulink调用C代码实验原理

MATLAB Coder 快速入门 - MathWorks 中国

Simulink调用C代码有多种方式，以MATLAB Function调用为例，MATLAB Function模块可以编写基于MATLAB语言的函数，该函数可以在Simulink仿真中执行。如果需要将c代码嵌入到MATLAB Function中，可以使用coder.ceval函数来调用c函数，需要指定c函数的名称、输入参数。coder.ceval函数会在生成代码时，将MATLAB语言转换为对应的c语言，并保持与MATLAB Function中的逻辑一致。这样就可以实现在Simulink仿真中，同时运行MATLAB语言和c语言的功能。

配置源代码文件

确保Typedef.h和相应的C源文件位于当前工作目录或指定的路径中。

Typedef.h头文件应该包含GetMaxValue和SquareOperation函数的声明

Typedef.h

```
\#ifndef \_TYPEDEF_H

\#define \_TYPEDEF_H

\#define EXLIB_API \__declspec(dllexport)

typedef unsigned char uint8_t;

typedef short unsigned int uint16_t;

EXLIB_API extern uint16_t MaxValue;

EXLIB_API extern uint16_t Result;

EXLIB_API extern uint16_t GetMaxValue(uint16_t Input1, uint16_t Input2);

EXLIB_API extern uint16_t SquareOperation(uint16_t Input);

\#endif
```

相应的C源文件应该包含这些函数的定义

Max.c

```
\#include "Typedef.h"

uint16_t GetMaxValue(uint16_t Input1, uint16_t Input2)

{

if(Input1 \> Input2)

{

return Input1;

}

else

{

return Input2;

}

}
```

Square.c

```
\#include "Typedef.h"

uint16_t MaxValue = 4500;

uint16_t Result;

uint16_t SquareOperation(uint16_t Input)

{

uint16_t Result = Input \* Input;

if(Result \> MaxValue)

{

Result = MaxValue;

}

return Result;

}
```

2.实验效果

运行Simulink仿真时通过调用已有的外部C函数实现特定计算操作，在Simulink的Display模块中观测运算结果。

3.文件目录

例程目录：[安装目录]\RflySimAPIs\4.RflySimModel\0.ApiExps\2.UserDefinedC++\1.CallC

文件夹/文件名称	说明	
..\Readme.pdf	Simulink调用c代码实验原理	
	Matlab Coder使用指南	
Code	Typedef.h	C代码的头文件
	Square.c	C代码的源码，包含计算平方的函数
	Max.c	C代码的源码，包含计算最大值的函数
	mfunc.slx	通过m函数的方式在simulink中调用c代码
	dll.slx	通过生成dll的方式在simulink中调用c代码
	SfBuilder.slx	通过S函数的方式在simulink中调用c代码
	state.slx	**通过状态图表的方式在Simulink中调用c代码，最低需要MATLAB2021B**
	caller.slx	**通过C caller模块的方式在simulink中调用c代码，最低需要MATLAB2021B**

4.运行环境

4.1 软件要求

Windows 10及以上版本；RflySim工具链；MATLAB 2017B及以上。

> ①：若使用Pixhawk 6X飞控，平台安装时的编译命令为：px4_fmu-v6x_default，推荐PX4固件版本为：1.12.3。其他配套飞控及编译命令请见：https://rflysim.com/doc/zh/1/Hardware.html

4.2 硬件要求

笔记本/台式电脑① 1台；\ \台；\ \台。

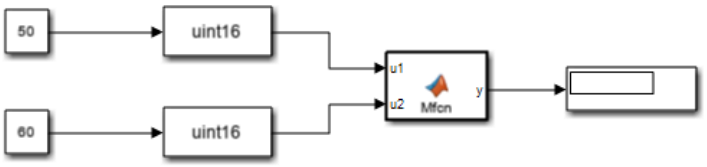
> ①：推荐配置请见： https://rflysim.com/

5.实验步骤

5.1. Matlab function调用

Step 1：搭建模型

加入MATLAB Function模块、Data Type Conversion模块、Display模块，连线如下



Step 2：配置Simulation Target调用

打开模型配置参数：Ctrl+E

选择Simulation Target：导航到Simulation Target

添加头文件：在Custom Code>Include Header 中添加：

#**include** "Typedef.h"

添加源文件：在Custom Code>Source files中添加：

Max.c

Square.c

Step 3：编写代码

在MATLAB Function模块编写如下程序来调用c代码

function y = Mfcn(u1,u2)

y = 0;

coder.updateBuildInfo('addIncludePaths','[pwd]');

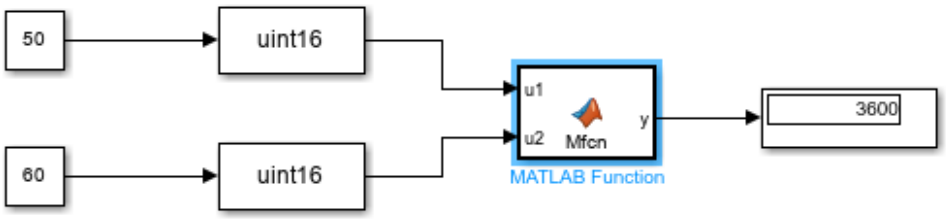
coder.cinclude('Typedef.h');

y = coder.ceval('GetMaxValue',u1,u2);

y = coder.ceval('SquareOperation',y);

Step 4：运行仿真

可见结果显示正确。



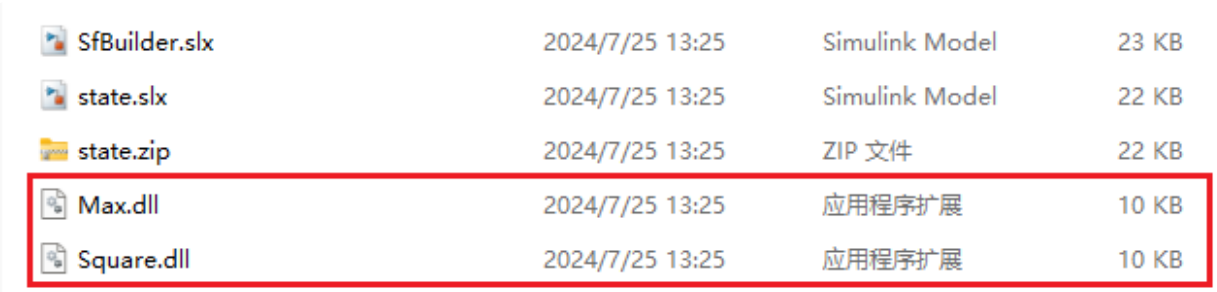
5.2. dll调用

Step 1：生成dll文件

可以使用以下指令生成C代码对应的dll文件

mex('LDEXT=.dll','LINKEXPORT=','LINKEXPORTVER=','LINKLIBS=','Square.c')

mex('LDEXT=.dll','LINKEXPORT=','LINKEXPORTVER=','LINKLIBS=','Max.c')



Step 2：模型属性配置

示例模型同上，进入模型属性设置界面，在Model Properties中加载dll文件

Step 3：编写代码

在MATLAB Function模块编写如下程序来调用c代码

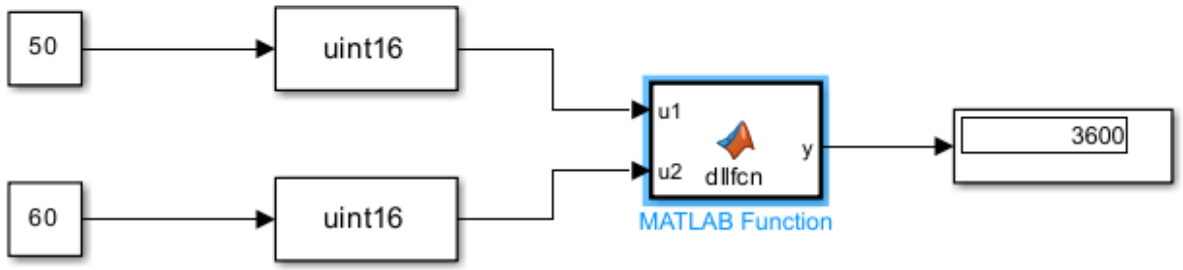
function y = dllfcn(u1,u2)

coder.extrinsic('calllib');

```
y = 0;

y = calllib('Max', 'GetMaxValue', u1, u2);

y = calllib('Square', 'SquareOperation', y);

end
```

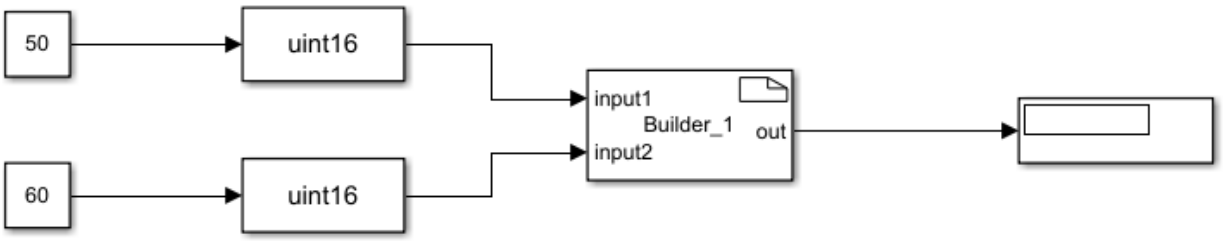
Step 4：运行仿真

结果如下：



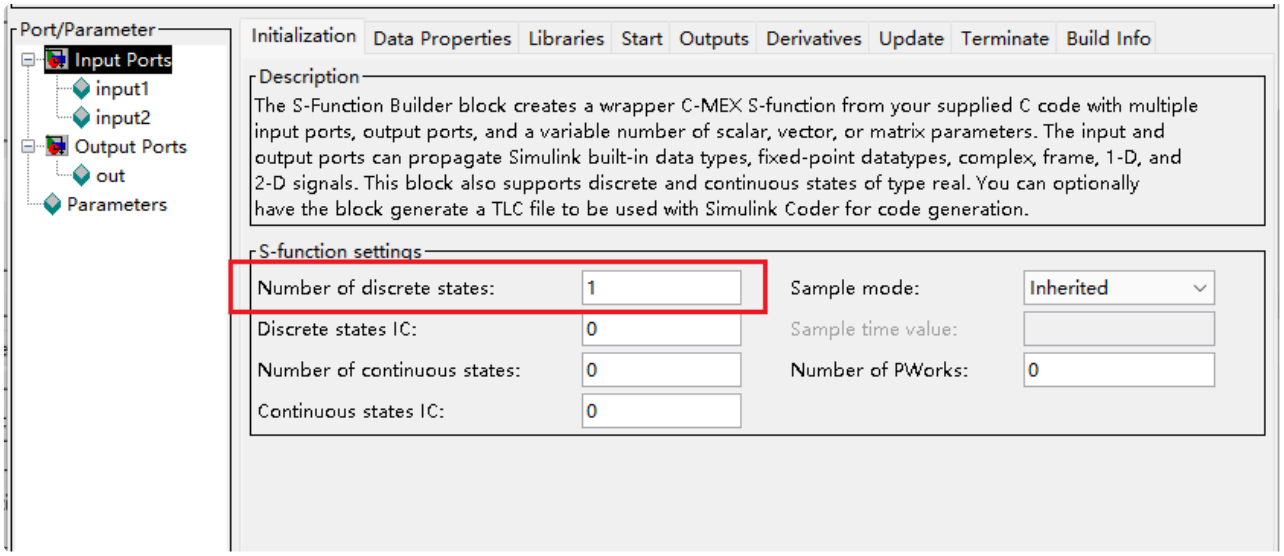
5.3. Sfunction Builder调用

Step 1：搭建模型

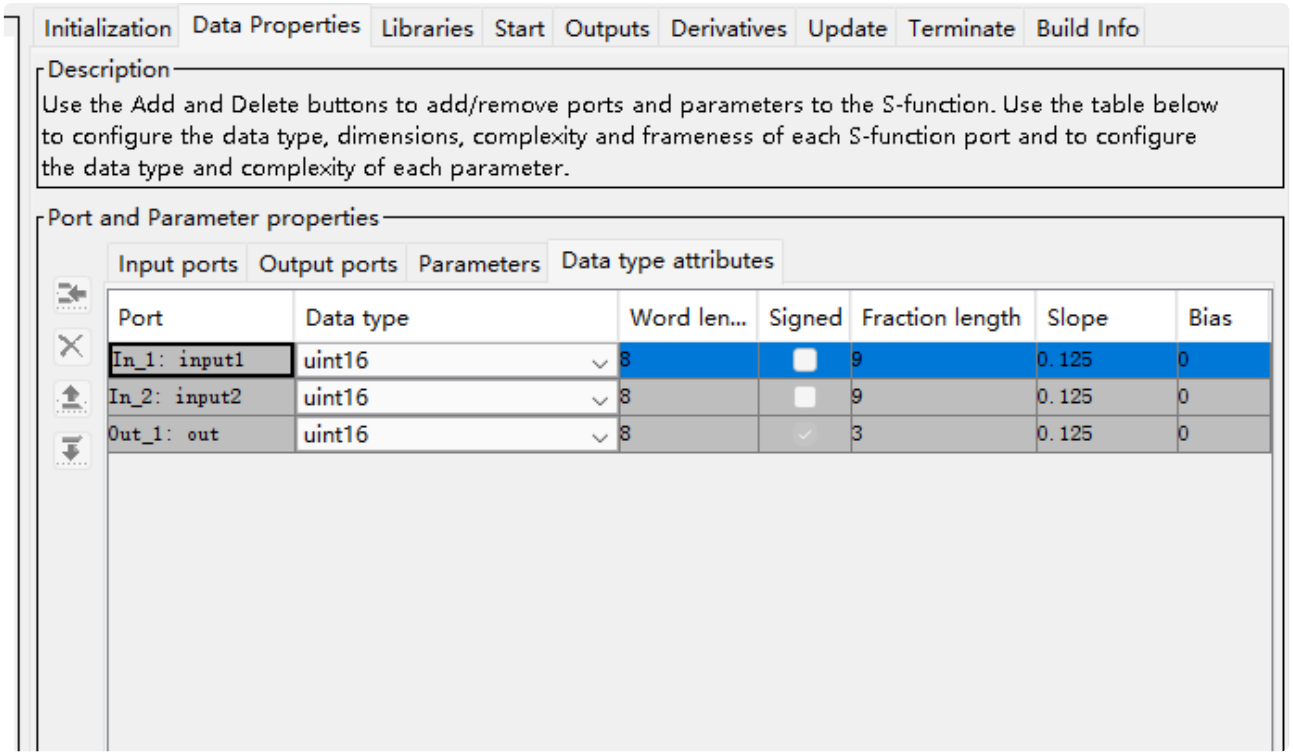


Step 2：Sfunction Builder配置

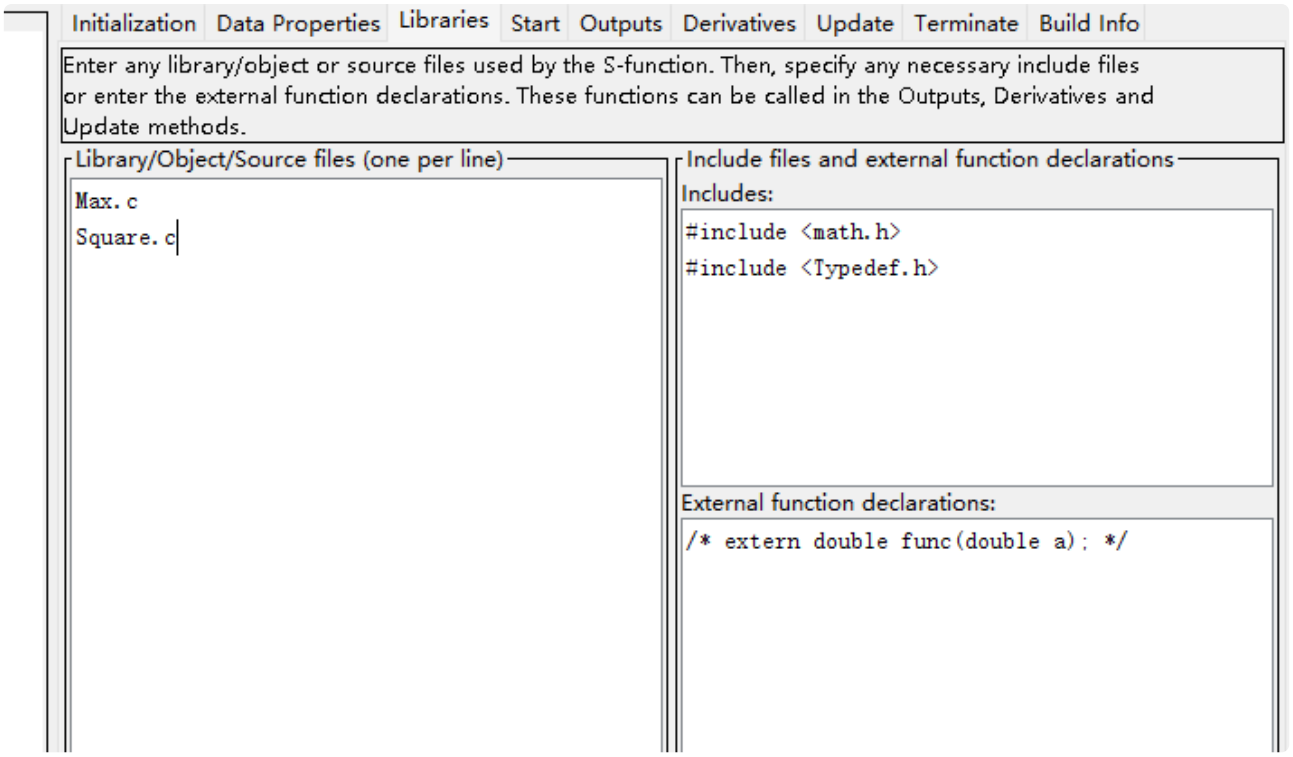
采用定步长仿真，离散状态个数设为1。



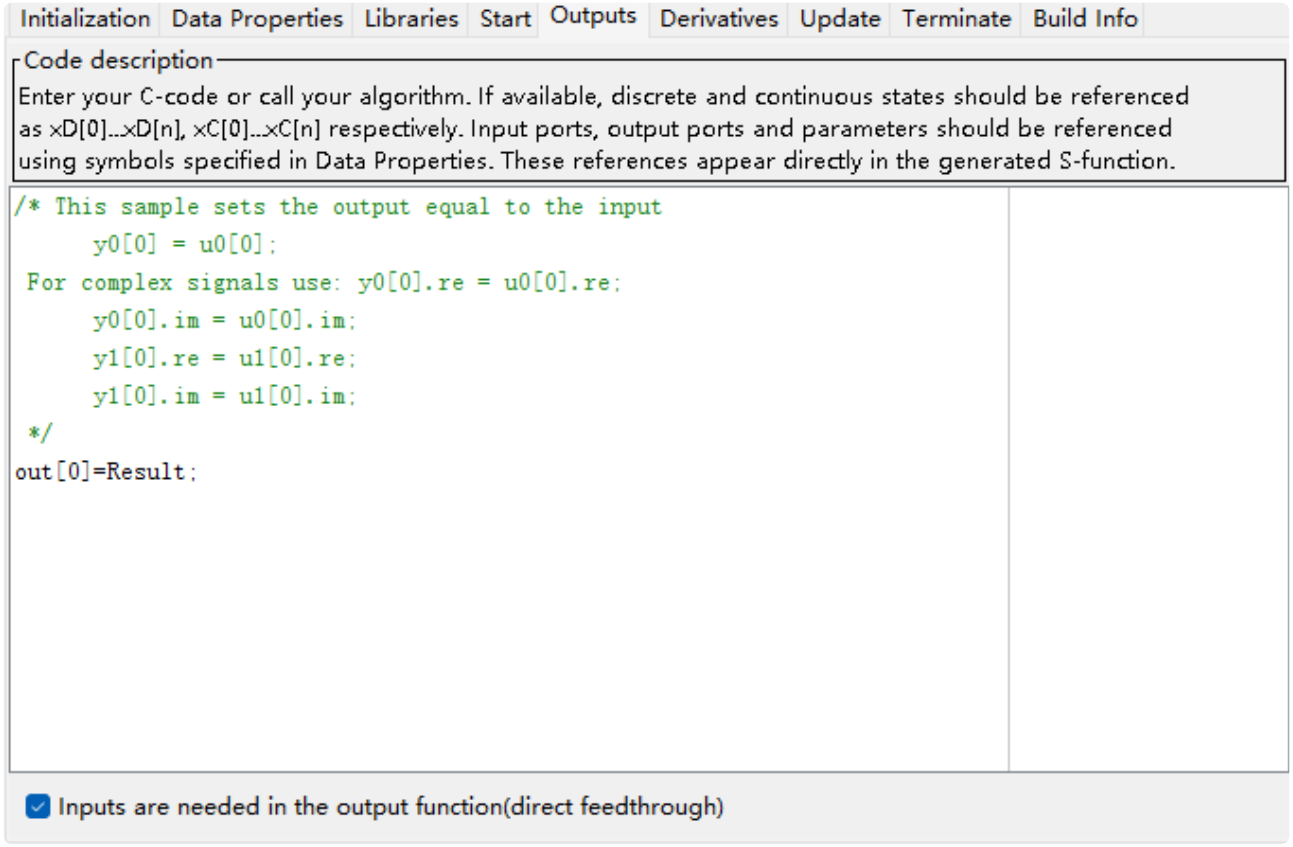
输入输出的数据属性如下：



加入源文件与头文件：


Initialization
Data Properties
Libraries
Start
Outputs
Derivatives
Update
Terminate
Build Info
Enter any library/object or source files used by the S-function. Then, specify any necessary include files or enter the external function declarations. These functions can be called in the Outputs, Derivatives and Update methods.
Library/Object/Source files (one per line)
Max.c
Square.c
Include files and external function declarations
Includes:
#include <math.h>
#include <Typedef.h>
External function declarations:
/* extern double func(double a); */

设置数据输出：


Initialization
Data Properties
Libraries
Start
Outputs
Derivatives
Update
Terminate
Build Info
Code description
Enter your C-code or call your algorithm. If available, discrete and continuous states should be referenced as xD[0]...xD[n], xC[0]...xC[n] respectively. Input ports, output ports and parameters should be referenced using symbols specified in Data Properties. These references appear directly in the generated S-function.
/* 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;
*/
out[0]=Result;
Inputs are needed in the output function(direct feedthrough)

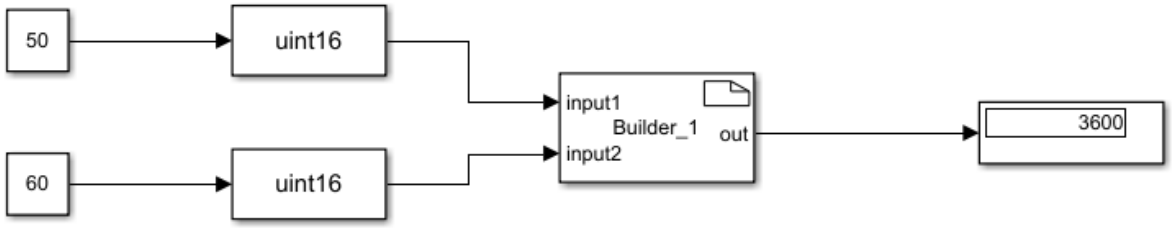
Initialization | Data Properties | Libraries | Start | Outputs | Derivatives | Update | Terminate | Build Info

Code description

This section is optional and used to update the discrete states. It is called only if the S-function has one or more discrete states. The states of the S-function are of type double and must be referenced as xD[0]...xD[n]. Input ports, output ports and parameters should be referenced using symbols specified in Data Properties. These references appear directly in the generated S-function.

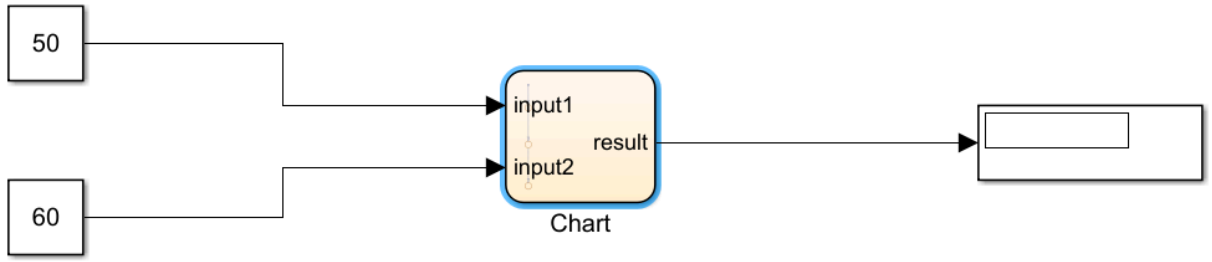
```
/*
 * Code example
 *   xD[0] = u0[0];
 */
Result=GetMaxValue(input1[0],input2[0]);
Result=SquareOperation(Result);
```

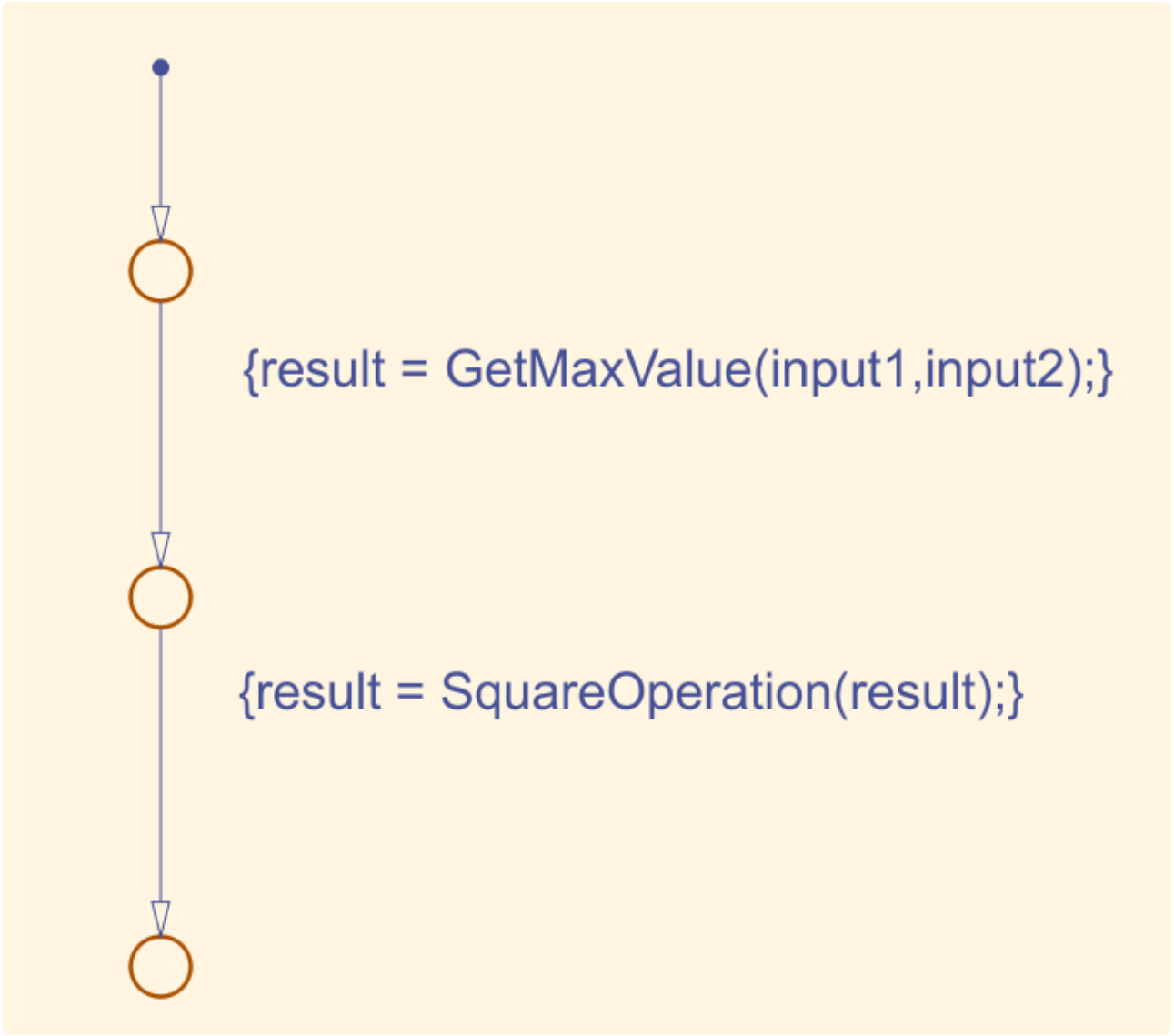
Step 3：运行仿真



5.4. 状态图表调用

Step 1：搭建模型





Step 2：模型配置

选择Simulation Target：导航到Simulation Target

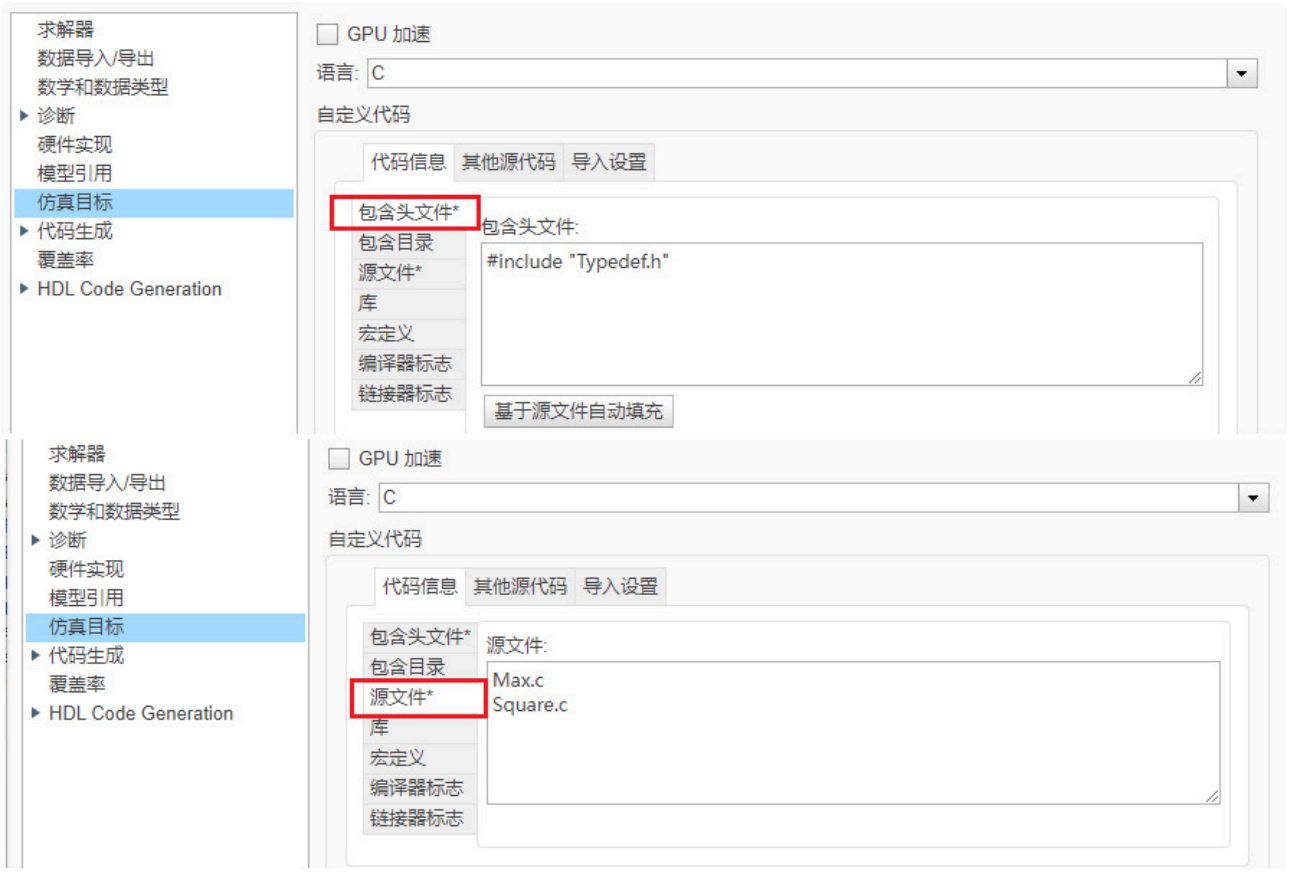
添加头文件：在Custom Code>Include Header 中添加：

#**include** "Typedef.h"

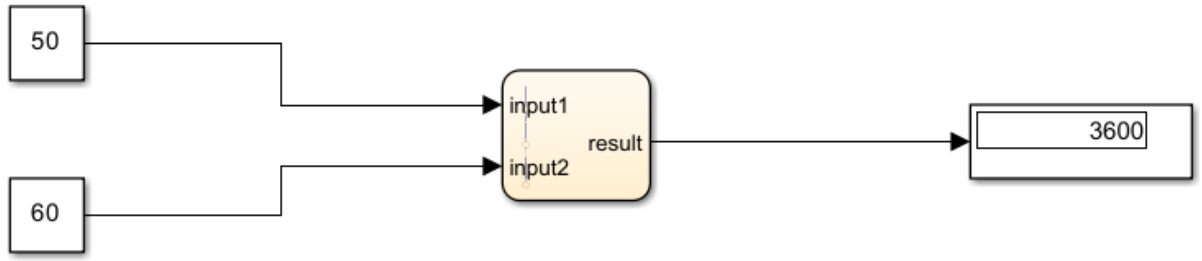
添加源文件：在Custom Code>Source files中添加：

Max.c

Square.c



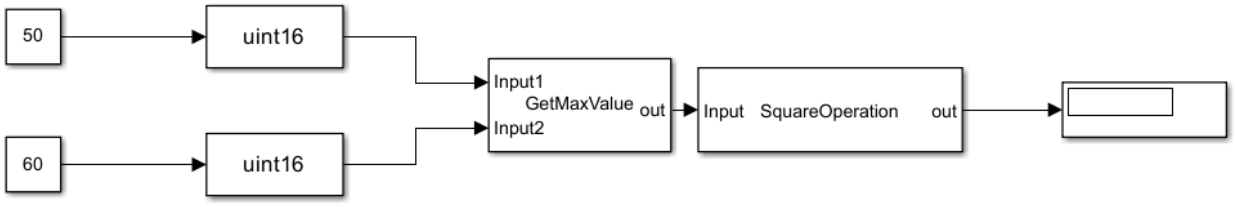
Step 3：运行仿真

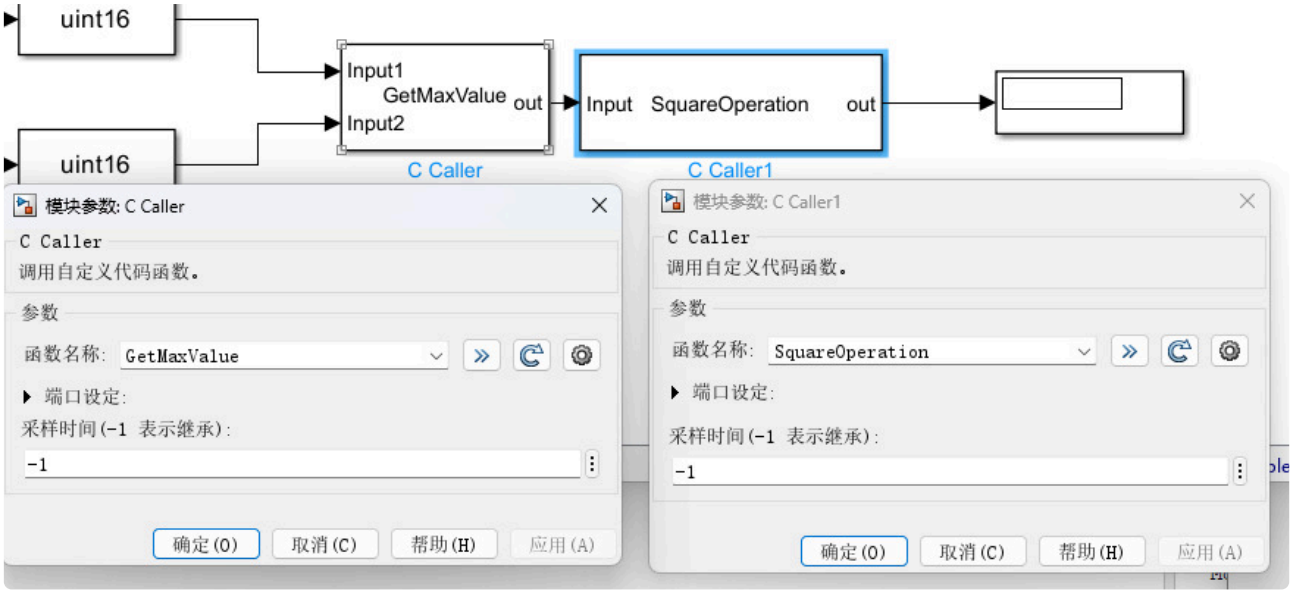


注意：最低需要MATLAB2021B。

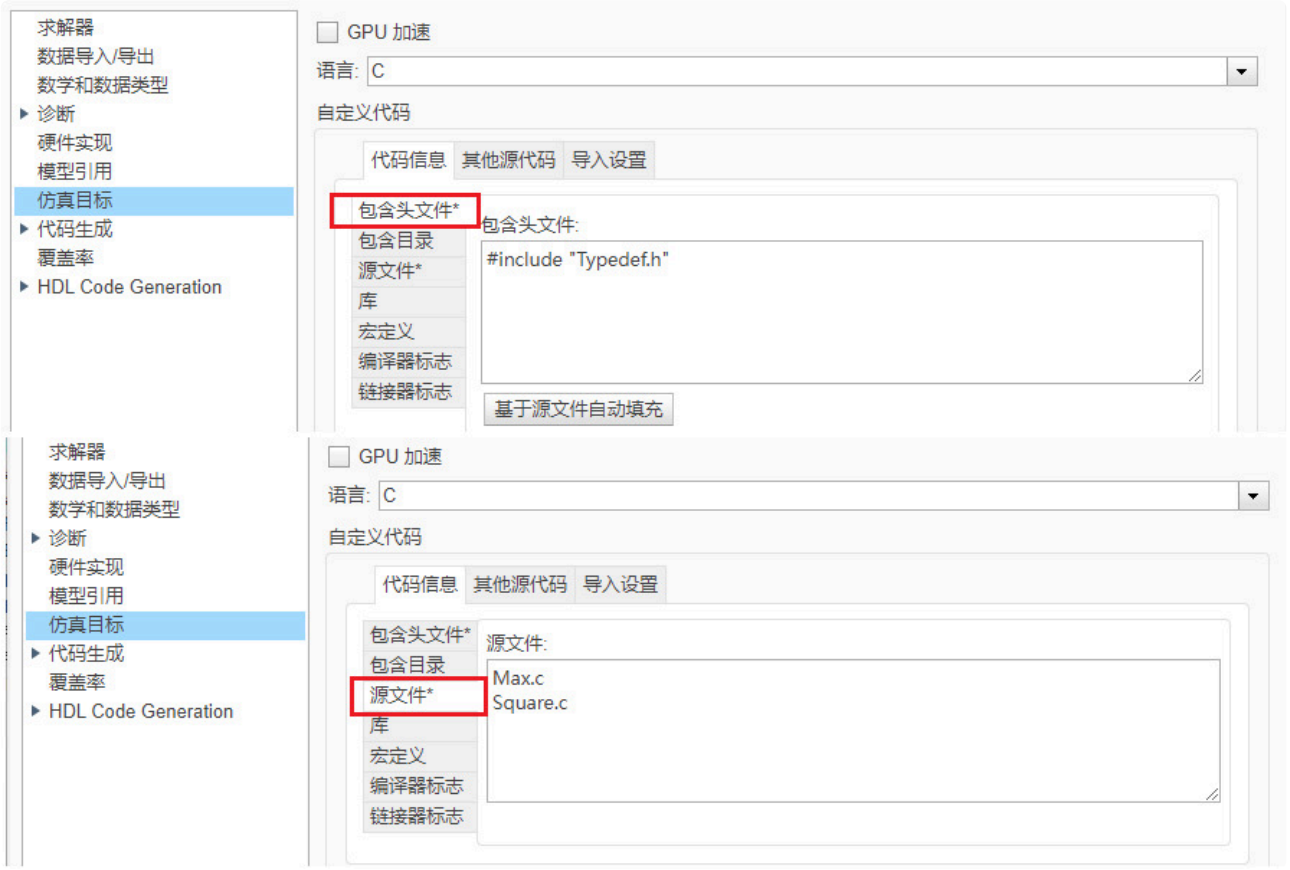
5.5. C caller调用

Step 1：搭建模型

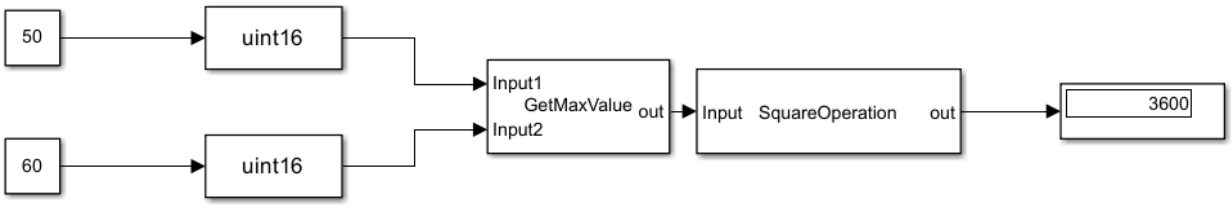




Step 2：模型配置



Step 3：运行仿真



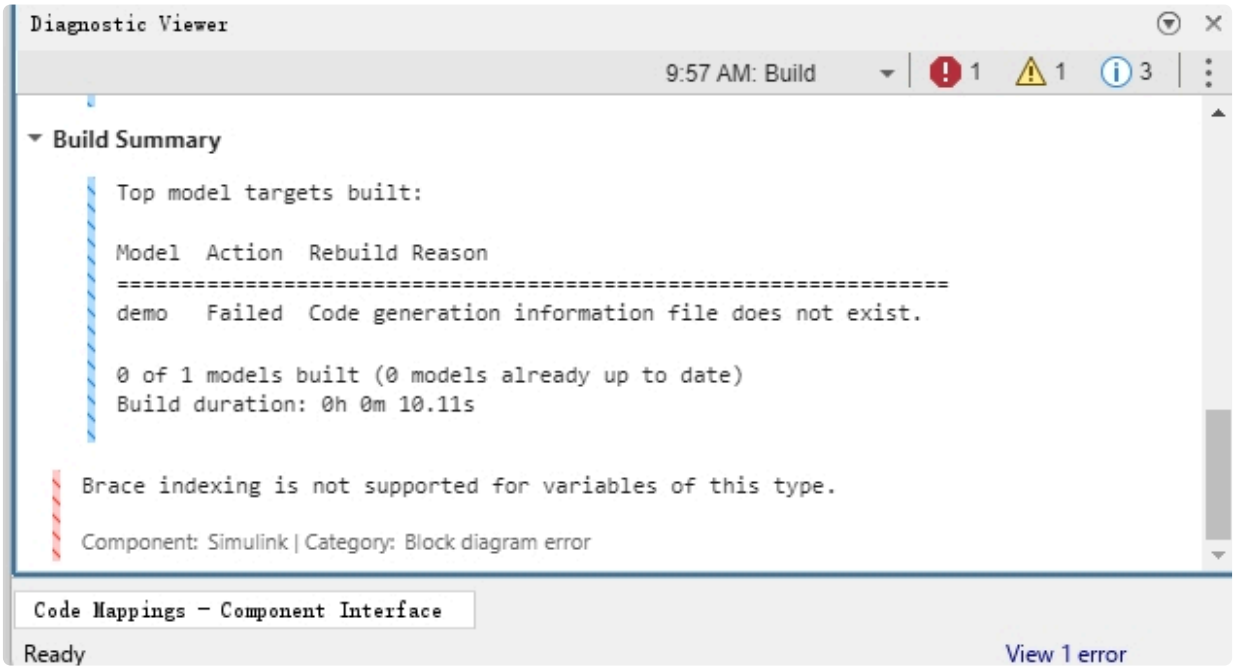
注意：最低需要MATLAB2021B。

6.参考资料

1. MATLAB官网Embedded Coder简介： https://ww2.mathworks.cn/products/embedded-coder.html
   Simulink调用外部C代码的几种方法_simulink如何利用interpreted matlab
2. function模块调用c语言代码-CSDN博客
3.

7.常见问题

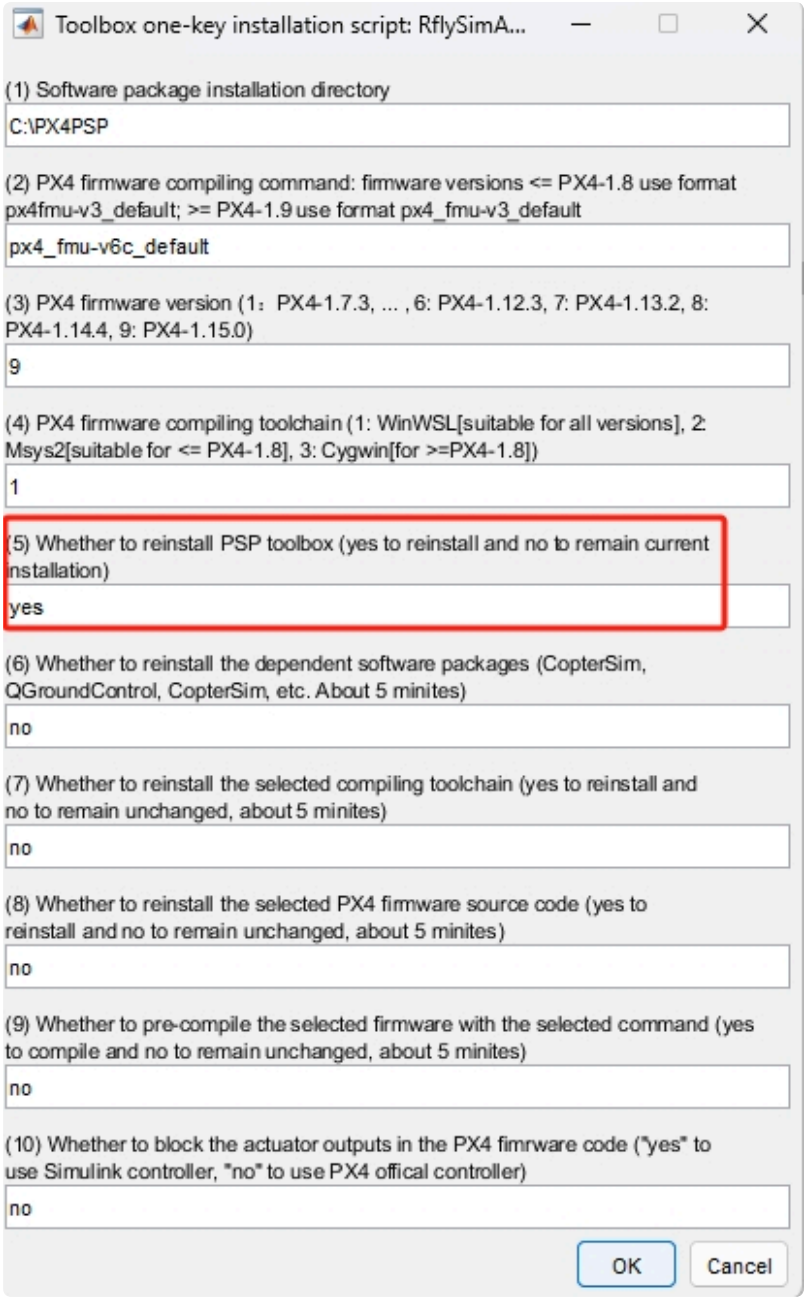
Q1：未正确安装visual studio c++编译环境并配置mex，导致Simulink文件编译失败



A1：首先将低于当前MATLAB版本的Visual Studio C++编译环境安装到VS默认安装目录，然后在MATLAB的命令行窗口中输入指令“mex -setup”，一般来说会自动识别并安装上支持的编译器，命令行显示“MEX 配置使用‘Microsoft Visual C++ 2017’以进行编译”的字样说明安装正确。详细环境配置参考”[RflySim平台安装目录]\RflySimAPIs\4.RflySimModel\API.pdf“中的环境配置

Q2：编译报错，无法加载库文件

A2：这可能是由于安装平台时PX4PSP工具箱未更新到最新版，更新RflySim安装包后按照如下配置重新安装平台即可


Toolbox one-key installation script: RflySimA...
(1) Software package installation directory
C:\PX4PSP
(2) PX4 firmware compiling command: firmware versions <= PX4-1.8 use format px4fmu-v3_default; >= PX4-1.9 use format px4_fmu-v3_default
px4_fmu-v6c_default
(3) PX4 firmware version (1：PX4-1.7.3, ... , 6: PX4-1.12.3, 7: PX4-1.13.2, 8: PX4-1.14.4, 9: PX4-1.15.0)
9
(4) PX4 firmware compiling toolchain (1: WinWSL[suitable for all versions], 2: Msys2[suitable for <= PX4-1.8], 3: Cygwin[for >=PX4-1.8])
1
(5) Whether to reinstall PSP toolbox (yes to reinstall and no to remain current installation)
yes
(6) Whether to reinstall the dependent software packages (CopterSim, QGroundControl, CopterSim, etc. About 5 minites)
no
(7) Whether to reinstall the selected compiling toolchain (yes to reinstall and no to remain unchanged, about 5 minites)
no
(8) Whether to reinstall the selected PX4 firmware source code (yes to reinstall and no to remain unchanged, about 5 minites)
no
(9) Whether to pre-compile the selected firmware with the selected command (yes to compile and no to remain unchanged, about 5 minites)
no
(10) Whether to block the actuator outputs in the PX4 fimrware code ("yes" to use Simulink controller, "no" to use PX4 offical controller)
no
OK
Cancel