

SimulinkSDK 使用说明

一、文件

- (1)、XINGYING_SimulinkDemo 文件夹-----客户端接收程序，供参考。
- (2)、lib 文件夹-----动态库相关文件，包括“.h”、“.lib”、“.dll”文件。

二、XINGYING_SimulinkDemo 操作流程（SDK 数据接收验证）

- (1)、设置电脑 IP 为 10.1.1.198，子网掩码为 255.255.255.0。关闭防火墙和网络拦截软件。
- (2)、以管理员的方式运行 XINGYING 动捕软件。
- (3)、设置菜单栏=》视图=》数据广播 =》网卡地址为“10.1.1.198”。
- (4)、勾选菜单栏=》视图=》数据广播 =》启用 SDK 点击开启。
- (5)、实时模式下，点击运行；

或者，后处理模式下，播放数据。

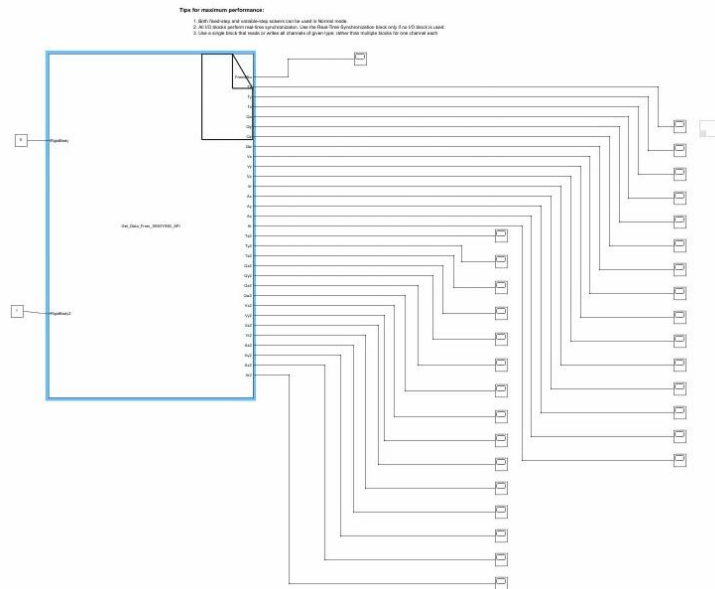
(6)、运行 XINGYING_SimulinkDemo 中的 XINGYING_SimulinkDemo.slx，即可接收到 SDK 数据。本程序默认在 10.1.1.198 的电脑上接收数据，接收的数据为第 iRigidBody 个刚体的数据，以及第 iRigidBody2 个刚体的数据。（帧号：FrameNo。第 iRigidBody 个刚体的位移信息：Tx，Ty，Tz；旋转信息：Qx，Qy，Qz，Qw；速度信息：Vx，Vy，Vz，Vr；加速度信息：Ax，Ay，Az，Ar；欧拉角信息：Roll，Pitch，Yaw；角速度信息：roll_Vel，pitch_Vel，Yaw_Vel）。第 iRigidBody2 个刚体的位移信息：Tx2，Ty2，Tz2；旋转信息：Qx2，Qy2，Qz2，Qw2；速度信息：Vx2，Vy2，Vz2，Vr2；加速度信息：Ax2，Ay2，Az2，Ar2；欧拉角信息：Roll2，Pitch2，Yaw2；角速度信息：roll2_Vel，pitch2_Vel，Yaw2_Vel）

注意：在后处理模式下，需要先关闭客户端接收程序，才能切换动捕数据。

iRigidBody 和 iRigidBody2 从 0 开始取值

三、XINGYING_SDK 在 Simulink 中使用的注意事项（以 demo 中的 S-Function Builder 方式说明）

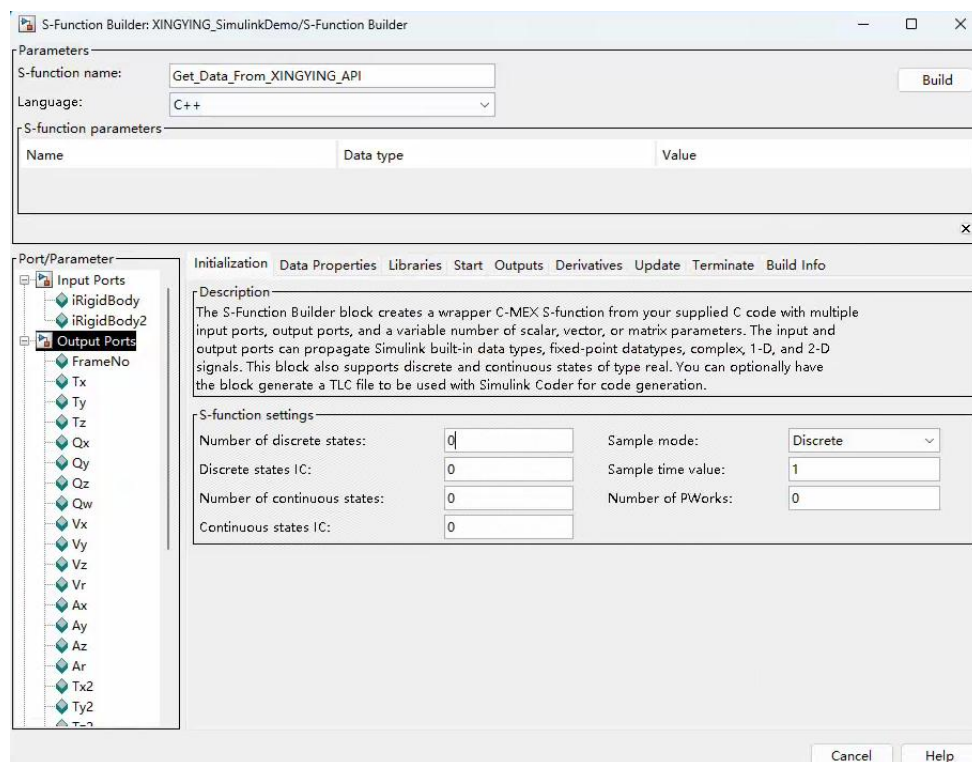
- (1)、需要安装 Visual Studio 2019
- (2)、本 Demo 在 Matlab 2020a 创建。（如果是其他版本的 Matlab，需要按照下列步骤，创建新的工程）



(3)、创建工程并添加 S-Function Builder 模块

(4)、设置代码生成的语言为 C++

(5)、设置采样



(6)、设置输入端口和输出端口，输入端口数据类型为 int32，输出端口数据类型为 double。

注意：如果需要多个输入或输出，可以在此添加输入或输出端口

S-Function Builder: XINGYING_SimulinkDemo/S-Function Builder

Parameters

S-function name:

Build

Language:

C++

S-function parameters

Name	Data type	Value

Port/Parameter

Input Ports

iRigidBody

iRigidBody2

Output Ports

FrameNo

Tx

Ty

Tz

Qx

Qy

Qz

Qw

Vx

Vy

Vz

Vr

Ax

Ay

Az

Ar

Tx2

Ty2

Tz2

Initialization

Data Properties

Libraries

Start

Outputs

Derivatives

Update

Terminate

Build Info

Description

Use the Add and Delete buttons to add/remove ports and parameters to the S-function. Use the table below to configure the data type, dimensions and complexity of each S-function port and to configure the data type and complexity of each parameter.

Array layout

Column-major

Port and Parameter properties

Input ports

Output ports

Parameters

Data type attributes

Port name	Dimensions	Rows	Columns	Complexity	Bus	Bus Name
iRigidBody	1-D	1		real	off	
iRigidBody2	1-D	1		real	off	

Cancel

Help

S-Function Builder: XINGYING_SimulinkDemo/S-Function Builder

Parameters

S-function name:

Build

Language:

C++

S-function parameters

Name	Data type	Value

Port/Parameter

Input Ports

iRigidBody

iRigidBody2

Output Ports

FrameNo

Tx

Ty

Tz

Qx

Qy

Qz

Qw

Vx

Vy

Vz

Vr

Ax

Ay

Az

Ar

Tx2

Ty2

Tz2

Initialization

Data Properties

Libraries

Start

Outputs

Derivatives

Update

Terminate

Build Info

Description

Use the Add and Delete buttons to add/remove ports and parameters to the S-function. Use the table below to configure the data type, dimensions and complexity of each S-function port and to configure the data type and complexity of each parameter.

Array layout

Column-major

Port and Parameter properties

Input ports

Output ports

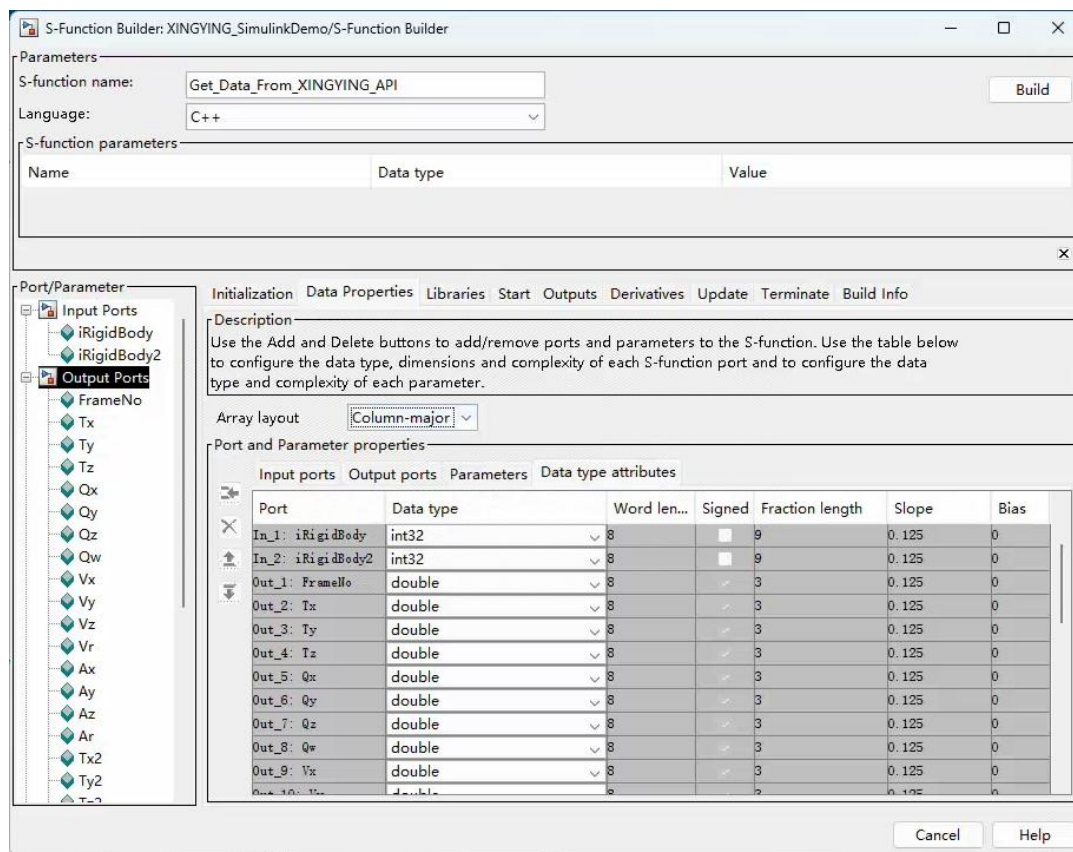
Parameters

Data type attributes

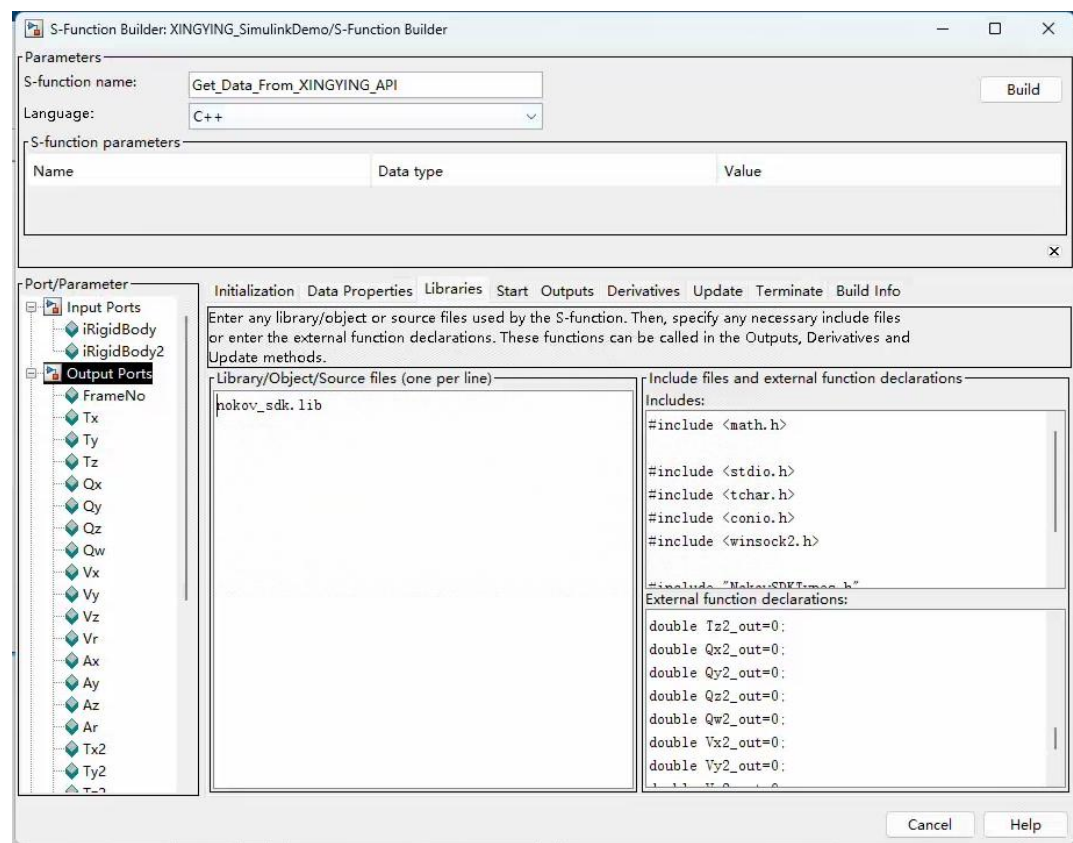
Port name	Dimensions	Rows	Columns	Complexity	Bus	Bus Name
FrameNo	1-D	1		real	off	
Tx	1-D	1		real	off	
Ty	1-D	1		real	off	
Tz	1-D	1		real	off	
Qx	1-D	1		real	off	
Qy	1-D	1		real	off	
Qz	1-D	1		real	off	
Qw	1-D	1		real	off	
Vx	1-D	1		real	off	
Vy	1-D	1		real	off	
Vz	1-D	1		real	off	

Cancel

Help



(7)、设置动态库 nokov_sdk.lib



(8)、包含头文件

```
#include <math.h>

#include <stdio.h>
#include <tchar.h>
#include <conio.h>
#include <winsock2.h>

#include "NokovSDKTypes.h"
#include "NokovSDKClient.h"
#include <iostream>
```

(9)、函数声明

```
unsigned int MyServersDataPort = 3130;
unsigned int MyServersCommandPort = 3131;
```

```
NokovSDKClient* theClient;
char szMyIPAddress[128] = "";
char szServerIPAddress[128] = "";
```

```
int iRigidBody_in=0;
int iRigidBody2_in=0;
int FrameNo_out=0;
```

```
int curFrmNo = 0;
double Tx_out=0;
double Ty_out=0;
double Tz_out=0;
double Qx_out=0;
double Qy_out=0;
double Qz_out=0;
double Qw_out=0;
double Vx_out=0;
double Vy_out=0;
double Vz_out=0;
double Vr_out=0;
double Ax_out=0;
double Ay_out=0;
double Az_out=0;
double Ar_out=0;
double Roll_out=0;
double Pitch_out=0;
double Yaw_out=0;
double roll_Vel_out=0;
```

```
double pitch_Vel_out=0;
double yaw_Vel_out=0;
```

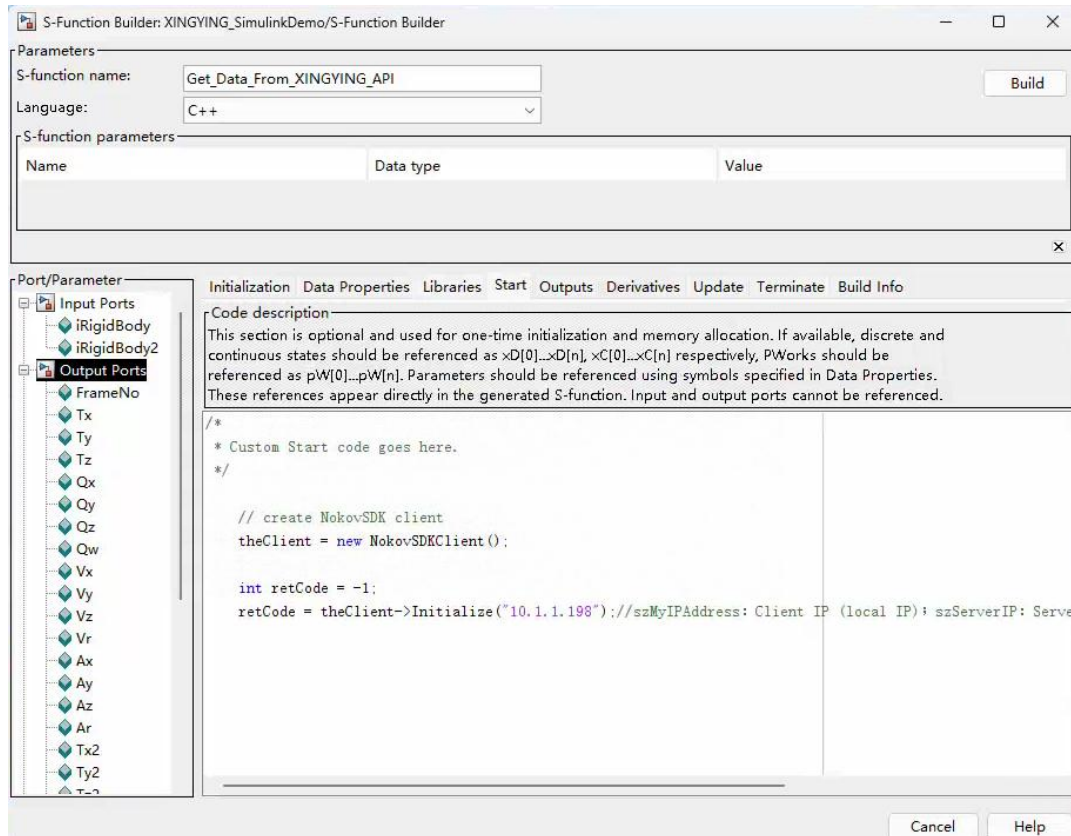
```
double Tx2_out=0;
double Ty2_out=0;
double Tz2_out=0;
double Qx2_out=0;
double Qy2_out=0;
double Qz2_out=0;
double Qw2_out=0;
double Vx2_out=0;
double Vy2_out=0;
double Vz2_out=0;
double Vr2_out=0;
double Ax2_out=0;
double Ay2_out=0;
double Az2_out=0;
double Ar2_out=0;
double Roll2_out=0;
double Pitch2_out=0;
double Yaw2_out=0;
double roll2_Vel_out=0;
double pitch2_Vel_out=0;
double yaw2_Vel_out=0;
```

注意：如果需要定义多个输出，可以在此增加多个输出的中间变量

(10)、初始部分添加的初始化代码

```
// create NokovSDK client
theClient = new NokovSDKClient();

int retCode = -1;
retCode = theClient->Initialize("10.1.1.198");//szMyIPAddress:Client
IP (local IP);szServerIP:Server IP (computer IP running motion capture
software)
```



(11)、输出部分添加的接收数据代码（如需同时输出更多刚体数据，可以对应修改或增加端口，并且增加数据输出代码）

```
iRigidBody_in = iRigidBody[0]; //the i-th RigidBody (RigidBody)
iRigidBody2_in = iRigidBody2[0]; //the j-th RigidBody (RigidBody)

// iMarkerset_in = iMarkerset[0]; //the i-th Markerset of the motion
capture data
// iSegments_in = iSegments[0]; //the j-th RigidBody (RigidBody)
// iMarkerset2_in = iMarkerset2[0]; //the i-th Markerset of the motion
capture data
// iSegments2_in = iSegments2[0]; //the j-th RigidBody (RigidBody)

sFrameOfMocapData* data = theClient->GetLastFrameOfMocapData();

if (data != NULL)
{
    curFrmNo = data->iFrame;

    sExtendData* pExtData = NULL;
    for (int i = 0; i < data->extendData.nExtendDataNum; i++)
    {
        if (data->extendData.extendData[i].type ==
```

```

ExtendDataRigidBody)
{
    pExtData = &data->extendData.extendData[i];
    break;
}
}

//the i-th RigidBody (RigidBody)
if(iRigidBody_in>=data->nRigidBodies)
{
    //When there is an input error, the output is 0
    Tx_out = 0;
    Ty_out = 0;
    Tz_out = 0;
    Qx_out = 0;
    Qy_out = 0;
    Qz_out = 0;
    Qw_out = 0;
    Vx_out = 0;
    Vy_out = 0;
    Vz_out = 0;
    Vr_out = 0;
    Ax_out = 0;
    Ay_out = 0;
    Az_out = 0;
    Ar_out = 0;
    Roll_out = 0;
    Pitch_out = 0;
    Yaw_out = 0;
    roll_Vel_out = 0;
    pitch_Vel_out = 0;
    yaw_Vel_out = 0;
}
else
{
    Tx_out = (real_T) data->RigidBodies[iRigidBody_in].x;
    Ty_out = (real_T) data->RigidBodies[iRigidBody_in].y;
    Tz_out = (real_T) data->RigidBodies[iRigidBody_in].z;
    Qx_out = (real_T) data->RigidBodies[iRigidBody_in].qx;
    Qy_out = (real_T) data->RigidBodies[iRigidBody_in].qy;
    Qz_out = (real_T) data->RigidBodies[iRigidBody_in].qz;
    Qw_out = (real_T) data->RigidBodies[iRigidBody_in].qw;
    if (pExtData != NULL)
    {

```

```

        // Vel
        Vx_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody_in].xVel;
        Vy_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody_in].yVel;
        Vz_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody_in].zVel;
        Vr_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody_in].rVel;
        // Acc
        Ax_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody_in].xAvel;
        Ay_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody_in].yAvel;
        Az_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody_in].zAvel;
        Ar_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody_in].rAvel;
        // Euler
        Roll_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody_in].roll;
        Pitch_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody_in].pitch;
        Yaw_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody_in].yaw;
        // Angular Vel
        roll_Vel_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody_in].rollVel;
        pitch_Vel_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody_in].pitchVel;
        yaw_Vel_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody_in].yawVel;
    }
}

//the j-th RigidBody (RigidBody)
if(iRigidBody2_in>=data->nRigidBodies)
{
    //When there is an input error, the output is 0
    Tx2_out = 0;
    Ty2_out = 0;
    Tz2_out = 0;
    Qx2_out = 0;
    Qy2_out = 0;
}

```

```

        Qz2_out = 0;
        Qw2_out = 0;
        Vx2_out = 0;
        Vy2_out = 0;
        Vz2_out = 0;
        Vr2_out = 0;
        Ax2_out = 0;
        Ay2_out = 0;
        Az2_out = 0;
        Ar2_out = 0;
        Roll2_out = 0;
        Pitch2_out = 0;
        Yaw2_out = 0;
        roll2_Vel_out = 0;
        pitch2_Vel_out = 0;
        yaw2_Vel_out = 0;
    }
    else
    {
        Tx2_out = (real_T) data->RigidBody2_in.x;
        Ty2_out = (real_T) data->RigidBody2_in.y;
        Tz2_out = (real_T) data->RigidBody2_in.z;
        Qx2_out = (real_T) data->RigidBody2_in.qx;
        Qy2_out = (real_T) data->RigidBody2_in.qy;
        Qz2_out = (real_T) data->RigidBody2_in.qz;
        Qw2_out = (real_T) data->RigidBody2_in.qw;
        if (pExtData != NULL)
        {
            // vel
            Vx2_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody2_in].xVel;
            Vy2_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody2_in].yVel;
            Vz2_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody2_in].zVel;
            Vr2_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody2_in].rVel;
            // acc
            Ax2_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody2_in].xAvel;
            Ay2_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody2_in].yAvel;
            Az2_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody2_in].zAvel;

```

```

        Ar2_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody2_in].rAvel;
        // Euler
        Roll2_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody2_in].roll;
        Pitch2_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody2_in].pitch;
        Yaw2_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody2_in].yaw;
        // Angular Vel
        roll2_Vel_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody2_in].rollVel;
        pitch2_Vel_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody2_in].pitchVel;
        yaw2_Vel_out = (real_T)
pExtData->extendData.rigidBodyExtend[iRigidBody2_in].yawVel;
    }
}

//Case2: Get Body Markerset
// if(iMarkerset[0]>=data->nSkeletons)
// {
//     //输入的Markerset 编号错误,输出赋值为 0
//     Tx_out = 0;
//     Ty_out = 0;
//     Tz_out = 0;
//     Qx_out = 0;
//     Qy_out = 0;
//     Qz_out = 0;
//     Qw_out = 0;
// }
// else
// {
//     Tx_out =
data->Skeletons[iMarkerset_in].RigidBodyData[iSegments_in].x;
//     Ty_out =
data->Skeletons[iMarkerset_in].RigidBodyData[iSegments_in].y;
//     Tz_out =
data->Skeletons[iMarkerset_in].RigidBodyData[iSegments_in].z;
//     Qx_out =
data->Skeletons[iMarkerset_in].RigidBodyData[iSegments_in].qx;
//     Qy_out =
data->Skeletons[iMarkerset_in].RigidBodyData[iSegments_in].qy;
//     Qz_out =

```

```

data->Skeletons[iMarkerset_in].RigidBodyData[iSegments_in].qz;
//          Qw_out =
data->Skeletons[iMarkerset_in].RigidBodyData[iSegments_in].qw;
//          }

}
else
{
    Sleep(5); //降低 CPU 占用
}

*FrameNo = (real_T) curFrmNo;
*Txx = (real_T) Tx_out;
*Ty = (real_T) Ty_out;
*Tz = (real_T) Tz_out;
*Qx = (real_T) Qx_out;
*Qy = (real_T) Qy_out;
*Qz = (real_T) Qz_out;
*Qw = (real_T) Qw_out;
*Vx = (real_T) Vx_out;
*Vy = (real_T) Vy_out;
*Vz = (real_T) Vz_out;
*Vr = (real_T) Vr_out;
*Ax = (real_T) Ax_out;
*Ay = (real_T) Ay_out;
*Az = (real_T) Az_out;
*Ar = (real_T) Ar_out;
*Roll = (real_T) Roll_out;
*Pitch = (real_T) Pitch_out;
*Yaw = (real_T) Yaw_out ;
*roll_Vel = (real_T) roll_Vel_out;
*pitch_Vel = (real_T) pitch_Vel_out;
*yaw_Vel = (real_T) yaw_Vel_out;

*Txx2 = (real_T) Tx2_out;
*Ty2 = (real_T) Ty2_out;
*Tz2 = (real_T) Tz2_out;
*Qx2 = (real_T) Qx2_out;
*Qy2 = (real_T) Qy2_out;
*Qz2 = (real_T) Qz2_out;
*Qw2 = (real_T) Qw2_out;
*Vx2 = (real_T) Vx2_out;
*Vy2 = (real_T) Vy2_out;
*Vz2 = (real_T) Vz2_out;

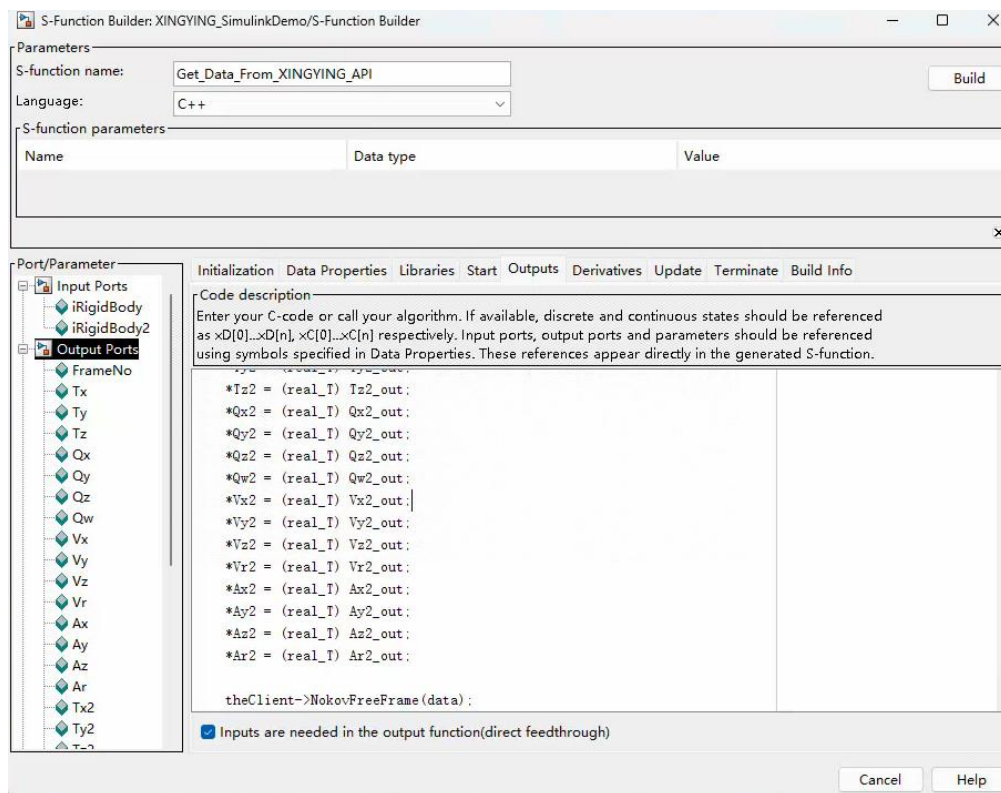
```

```

*Vr2 = (real_T) Vr2_out;
*Ax2 = (real_T) Ax2_out;
*Ay2 = (real_T) Ay2_out;
*Az2 = (real_T) Az2_out;
*Ar2 = (real_T) Ar2_out;
*Roll2 = (real_T) Roll2_out;
*Pitch2 = (real_T) Pitch2_out;
*Yaw2 = (real_T) Yaw2_out ;
*roll2_Vel = (real_T) roll2_Vel_out;
*pitch2_Vel = (real_T) pitch2_Vel_out;
*yaw2_Vel = (real_T) yaw2_Vel_out;

```

```
theClient->NokovFreeFrame(data);
```

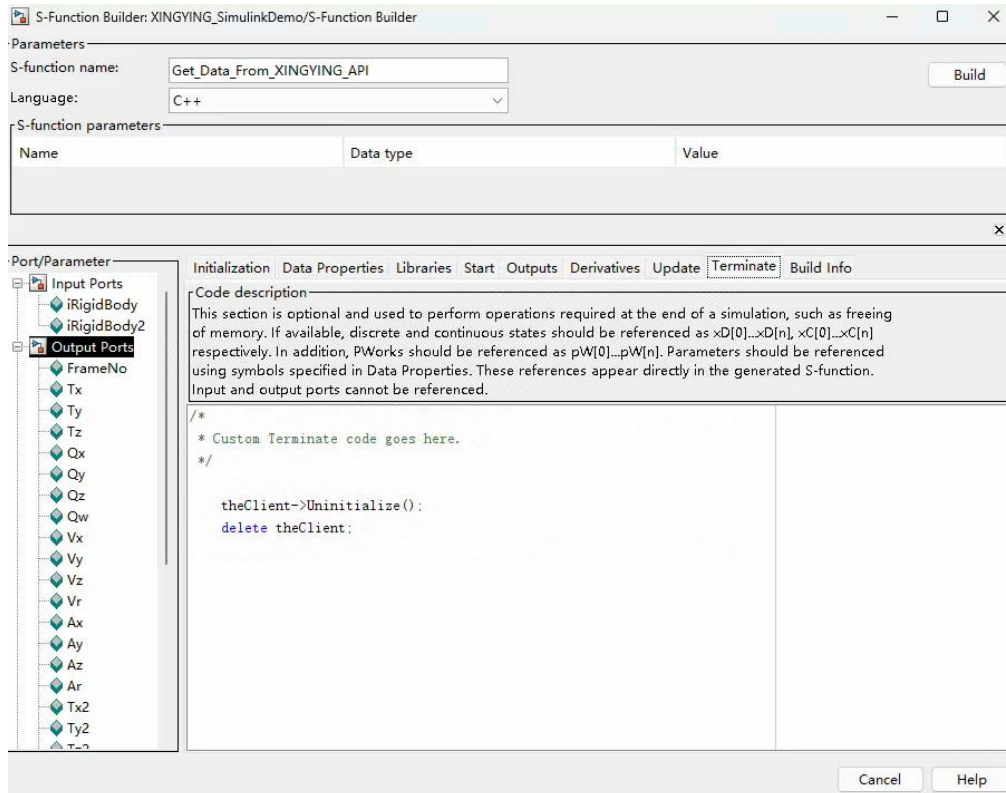


(12)、结束部分添加的代码

```

theClient->Uninitialize();
delete theClient

```



四、数据的具体解析

(1)、动捕系统连接初始化（输入 XINGYING 动捕软件 NIC Address）

```
int retCode = -1;
```

```
retCode = theClient->Initialize(szServerIP);//szMyIPAddress: Client IP (local IP); szServerIP: Server IP (computer IP running motion capture software)
```

(2)、获取数据描述信息

```
sServerDescription ServerDescription;
```

```
memset(&ServerDescription, 0, sizeof(ServerDescription));
```

```
theClient->GetServerDescription(&ServerDescription);
```

```
sDataDescriptions* ps = nullptr;
```

```
theClient->GetDataDescriptions(&ps);
```

dataDescription 描述信息解析:

```
ps->nDataDescriptions ----- 描述信息数量
```

```
const auto& dataDescription = ps->arrDataDescriptions[dsIndex];
```

```
dataDescription.type ----- 描述信息类型; 1---Descriptor_MarkerSet; 2---Descriptor_RigidBody; 3---Descriptor_Skeleton; 4---Descriptor_ForcePlate
```

```
dataDescription.Data.MarkerSetDescription->szName---描述信息中 Markerset 名称
```

```
dataDescription.Data.MarkerSetDescription->nMarkers---描述信息中 Markerset 包含的点的数量
```

dataDescription.Data.MarkerSetDescription->szMarkerNames[markerIndex]---描述信息中 Markerset 包含的第 markerIndex 个点的名称

dataDescription.Data.RigidBodyDescription->szName ---描述信息中刚体名称

dataDescription.Data.SkeletonDescription->szName ---描述信息中骨架名称

dataDescription.Data.SkeletonDescription->nRigidBodies ---描述信息中骨架包含的骨段数量

dataDescription.Data.SkeletonDescription->RigidBodies[boneIndex].ID;

--描述信息中骨架包含的第 boneIndex 个骨段的 ID

dataDescription.Data.SkeletonDescription->RigidBodies[boneIndex].szName---描述信息中骨架包含的第 boneIndex 个骨段的名称

dataDescription.Data.ForcePlateDescription->nChannels---描述信息中测力台的通道数量

dataDescription.Data.ForcePlateDescription->szChannelNames[channelIdx]---描述信息中测力台
的第 channelIdx 个通道的名称

theClient->FreeDataDescriptions(ps);//注意：需要释放 ps
ps = nullptr;

(3)、获取数据信息（注意：需要和（5）对应使用）

sFrameOfMocapData* data = theClient->GetLastFrameOfMocapData();

data数据信息解析：

data->iFrame-----数据的帧号

data->Timecode-----数据的时间码信息，需使用（4）解码

data->TimecodeSubframe-----数据的时间码信息，需使用（4）解码

data->iTimeStamp-----数据的时间戳信息

data->nMarkerSets---数据的Markerset数量

sMarkerSetData* markerset = &data->MocapData[i];

markerset->szName---数据的第i个Markerset的名称

markerset->nMarkers---数据的第i个Markerset包含的Marker的数量

markerset->Markers[i_Marker][0]---数据的第i个Markerset包含的第i_Marker个点的
的X坐标

markerset->Markers[i_Marker][1]---数据的第i个Markerset包含的第i_Marker个点的
的Y坐标

markerset->Markers[i_Marker][2]---数据的第i个Markerset包含的第i_Marker个点的
的Z坐标

data->nRigidBodies---数据的刚体的数量

data->RigidBodies[i].ID---数据的第i个刚体的ID

data->RigidBodies[i].x---数据的第i个刚体的X坐标

data->RigidBodies[i].y---数据的第i个刚体的Y坐标

```

data->RigidBody[i].z---数据的第i个刚体的z坐标
data->RigidBody[i].qx---数据的第i个刚体的旋转信息四元数Qx
data->RigidBody[i].qy---数据的第i个刚体的旋转信息四元数Qy
data->RigidBody[i].qz---数据的第i个刚体的旋转信息四元数Qz
data->RigidBody[i].qw---数据的第i个刚体的旋转信息四元数Qw

data->RigidBody[i].nMarkers---数据的第i个刚体包含的Marker的数量
data->RigidBody[i].MarkerIDs[iMarker]---数据的第i个刚体包含的第iMarker
个Marker的ID
data->RigidBody[i].Markers[iMarker][0]---数据的第i个刚体包含的第
iMarker个Marker的x坐标
data->RigidBody[i].Markers[iMarker][1]---数据的第i个刚体包含的第
iMarker个Marker的y坐标
data->RigidBody[i].Markers[iMarker][2]---数据的第i个刚体包含的第
iMarker个Marker的z坐标

data->extendData.nExtendDataNum---数据的扩展数据的数量
data->extendData.extendData[i]---数据的第i个扩展数据
data->extendData.extendData[i].type---数据的第i个扩展数据的类型
sExtendData* pExtData = NULL;
for (int i = 0; i < data->extendData.nExtendDataNum; i++)
{
    if (data->extendData.extendData[i].type == ExtendDataRigidBody)
    {
        pExtData = &data->extendData.extendData[i];
        break;
    }
}
pExtData->extendData.rigidBodyExtend[iRigidBody_in].xVel---扩展数据的第
iRigidBody_in个刚体的x方向速度信息
pExtData->extendData.rigidBodyExtend[iRigidBody_in].yVel---扩展数据的第
iRigidBody_in个刚体的y方向速度信息
pExtData->extendData.rigidBodyExtend[iRigidBody_in].zVel---扩展数据的第
iRigidBody_in个刚体的z方向速度信息
pExtData->extendData.rigidBodyExtend[iRigidBody_in].rVel---扩展数据的第
iRigidBody_in个刚体的速度信息
pExtData->extendData.rigidBodyExtend[iRigidBody_in].xAvel---扩展数据的
第iRigidBody_in个刚体的x方向加速度信息
pExtData->extendData.rigidBodyExtend[iRigidBody_in].yAvel---扩展数据的
第iRigidBody_in个刚体的y方向加速度信息
pExtData->extendData.rigidBodyExtend[iRigidBody_in].zAvel---扩展数据的
第iRigidBody_in个刚体的z方向加速度信息
pExtData->extendData.rigidBodyExtend[iRigidBody_in].rAvel---扩展数据的
第iRigidBody_in个刚体的加速度信息

```

pExtData->extendData.rigidBodyExtend[iRigidBody_in].roll;---扩展数据的第iRigidBody_in个刚体的横滚角信息
pExtData->extendData.rigidBodyExtend[iRigidBody_in].pitch;---扩展数据的第iRigidBody_in个刚体的俯仰角信息
pExtData->extendData.rigidBodyExtend[iRigidBody_in].yaw;---扩展数据的第iRigidBody_in个刚体的偏航角信息
pExtData->extendData.rigidBodyExtend[iRigidBody_in].rollVel;---扩展数据的第iRigidBody_in个刚体的横滚角速度信息
pExtData->extendData.rigidBodyExtend[iRigidBody_in].pitchVel;---扩展数据的第iRigidBody_in个刚体的俯仰角速度信息
pExtData->extendData.rigidBodyExtend[iRigidBody_in].yawVel;---扩展数据的第iRigidBody_in个刚体的偏航角速度信息

data->nSkeletons---数据的骨架的数量
data->Skeletons[i].skeletonID---数据的第i个骨架的ID
data->Skeletons[i].nRigidBodies---数据的第i个骨架包含的骨段的数量
data->Skeletons[i].RigidBodyData[j].ID---数据的第i个骨架中第j段骨段的ID
data->Skeletons[i].RigidBodyData[j].x---数据的第i个骨架中第j段骨段的X坐标
data->Skeletons[i].RigidBodyData[j].y---数据的第i个骨架中第j段骨段的Y坐标
data->Skeletons[i].RigidBodyData[j].z---数据的第i个骨架中第j段骨段的Z坐标
data->Skeletons[i].RigidBodyData[j].qx---数据的第i个骨架中第j段骨段的旋转信息四元数Qx
data->Skeletons[i].RigidBodyData[j].qy---数据的第i个骨架中第j段骨段的旋转信息四元数Qy
data->Skeletons[i].RigidBodyData[j].qz---数据的第i个骨架中第j段骨段的旋转信息四元数Qz
data->Skeletons[i].RigidBodyData[j].qw---数据的第i个骨架中第j段骨段的旋转信息四元数Qw

data->Skeletons[i].RigidBodyData[j].nMarkers---数据的第i个骨架中第j段骨段包含的Marker数量
data->Skeletons[i].RigidBodyData[j].MarkerIDs[iMarker]---数据的第i个骨架中第j段骨段包含的第iMarker个Marker的ID
data->Skeletons[i].RigidBodyData[j].Markers[iMarker][0]---数据的第i个骨架中第j段骨段包含的第iMarker个Marker的X坐标
data->Skeletons[i].RigidBodyData[j].Markers[iMarker][1]---数据的第i个骨架中第j段骨段包含的第iMarker个Marker的Y坐标
data->Skeletons[i].RigidBodyData[j].Markers[iMarker][2]---数据的第i个骨架中第j段骨段包含的第iMarker个Marker的Z坐标

data->nOtherMarkers---数据的未命名Marker数量
data->OtherMarkers[i][0]---数据的第iMarker个未命名Marker的X坐标
data->OtherMarkers[i][1]---数据的第iMarker个未命名Marker的Y坐标
data->OtherMarkers[i][2]---数据的第iMarker个未命名Marker的Z坐标

```
data->nAnalogdatas---数据的模拟通道数量  
data->Analogdata[i]---数据的第i个模拟通道的数据
```

```
data->nLabeledMarkers---数据的所有命名点的数量  
data->LabeledMarkers[iLabeledMarkers].x---数据的第iLabeledMarkers个命名点的X坐标  
data->LabeledMarkers[iLabeledMarkers].y---数据的第iLabeledMarkers个命名点的Y坐标  
data->LabeledMarkers[iLabeledMarkers].z---数据的第iLabeledMarkers个命名点的Z坐标  
data->LabeledMarkers[iLabeledMarkers].size---数据的第iLabeledMarkers个命名点的大小
```

(4)、时间信息解码

```
int hour, minute, second, frame, subframe;  
bool bValid = theClient->DecodeTimecode(data->Timecode,  
data->TimecodeSubframe, &hour, &minute, &second, &frame, &subframe);
```

(5)、释放数据信息内存（注意：需要和（3）对应使用）

```
theClient->NokovFreeFrame(data);
```

(6)、结束部分添加的代码，与动捕系统断开连接

```
theClient->Uninitialize();
```

五、数据说明

(1)、坐标系为右手坐标系

(2)、对于 **Markerset** 中已经定义的 **Marker**，由于软件操作问题、**Marker** 遮挡等原因，造成丢点或识别不到的情况，X、Y、Z 坐标值会填充为 9999999.000000