

SimulinkSDK instruction manual

1. folder

- (1)、XINGYING_SimulinkDemo ----- The source code of client receiving program for reference.
- (2)、lib ----- The directory of library.

2. XINGYING_SimulinkDemo SDK data receiving

- (1)、Set the PC IP address to 10.1.1.198 and subnet mask to 255.255.255.0. Disable the firewall and network blocking software.
- (2)、Run XINGYING motion capture software as an administrator.
- (3)、Select the data broadcast interface, select "NIC Address: 10.1.1.198".
- (4)、check "Setting","SDK Enabled".XINGYING sends motion capture data.
- (5)、In live mode, click run;Or, in post-process mode, play back data.
- (6)、Run XINGYING_SimulinkDemo.slx to receive SDK data.

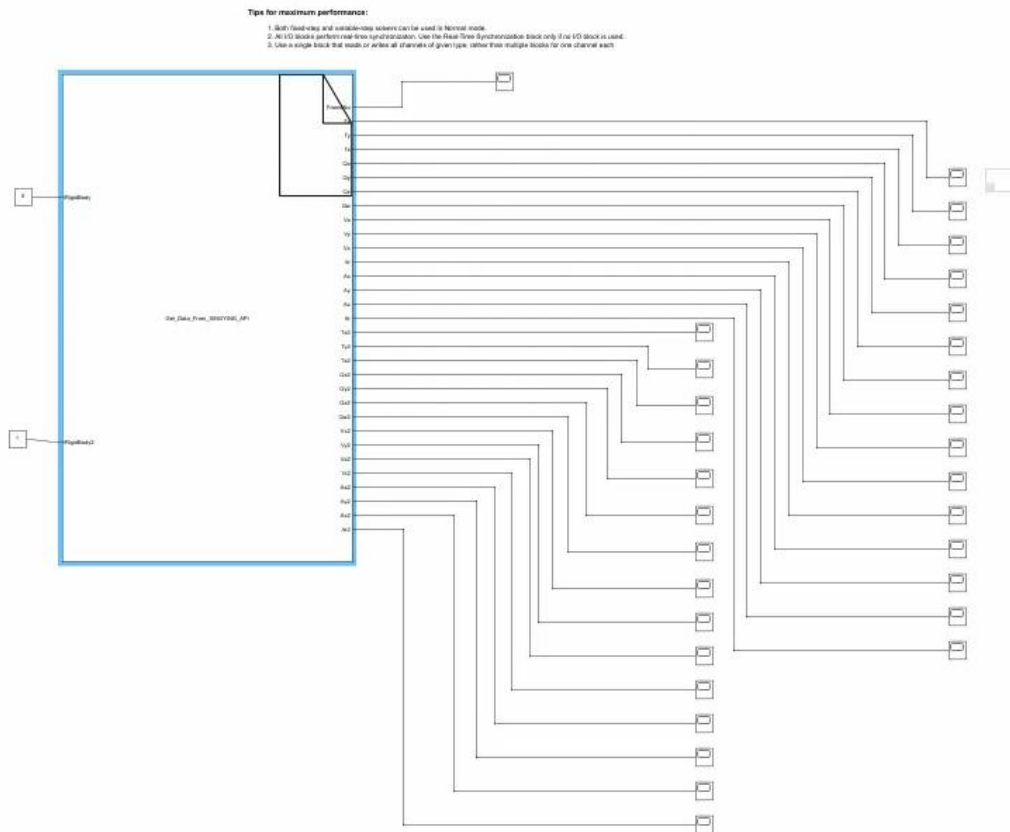
This program receives data by default on the computer at 10.1.1.198, and the data of the iRigidBody rigid body and the data of the iRigidBody2 rigid body. (Frame number: FrameNo; Displacement information of the iRigidBody rigid body: Tx, Ty, Tz; Rotation information: Qx, Qy, Qz, Qw; Speed information: Vx, Vy, Vz, Vr; Acceleration information: Ax, Ay, Az, Ar. Displacement information of the second rigid body iRigidBody: Tx2, Ty2, Tz2; Rotation information: Qx2, Qy2, Qz2, Qw2; Speed information: Vx2, Vy2, Vz2, Vr2; Acceleration information: Ax2, Ay2, Az2, Ar2)

Note: In post-process mode, you need to close the client receiver before switching the motion capture data.

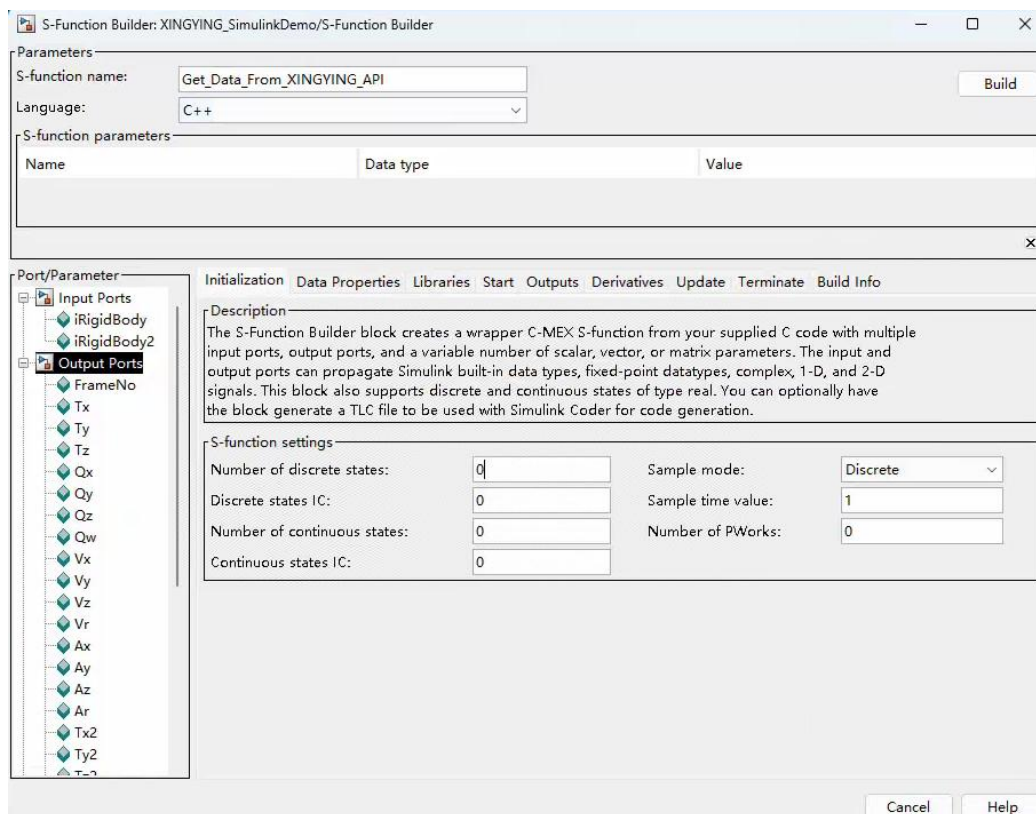
iRigidBody and iRigidBody2 Start taking values from 0

3. Precautions for using XINGYING_SDK in Simulink (explained using the S-Function Builder method in the demo)

- (1)、Visual Studio 2019 needs to be installed
- (2)、This demo was created in Matlab 2020a.(If it is another version of Matlab, you need to follow the steps below to create a new project)

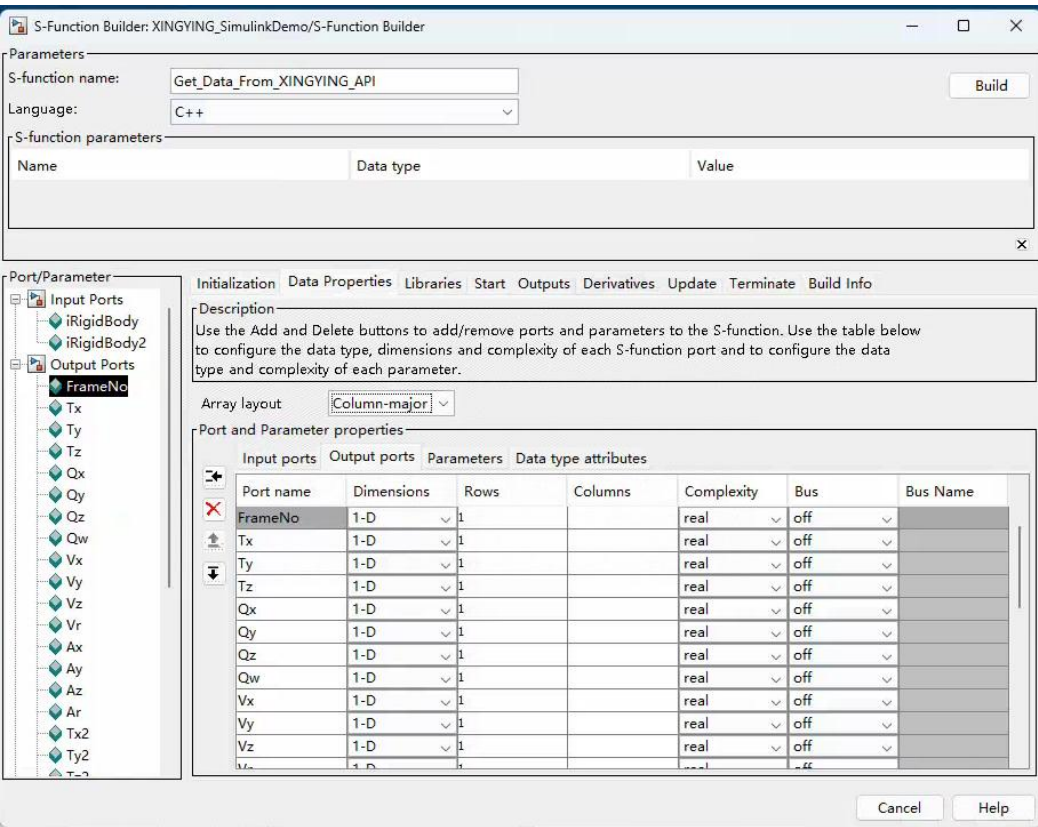
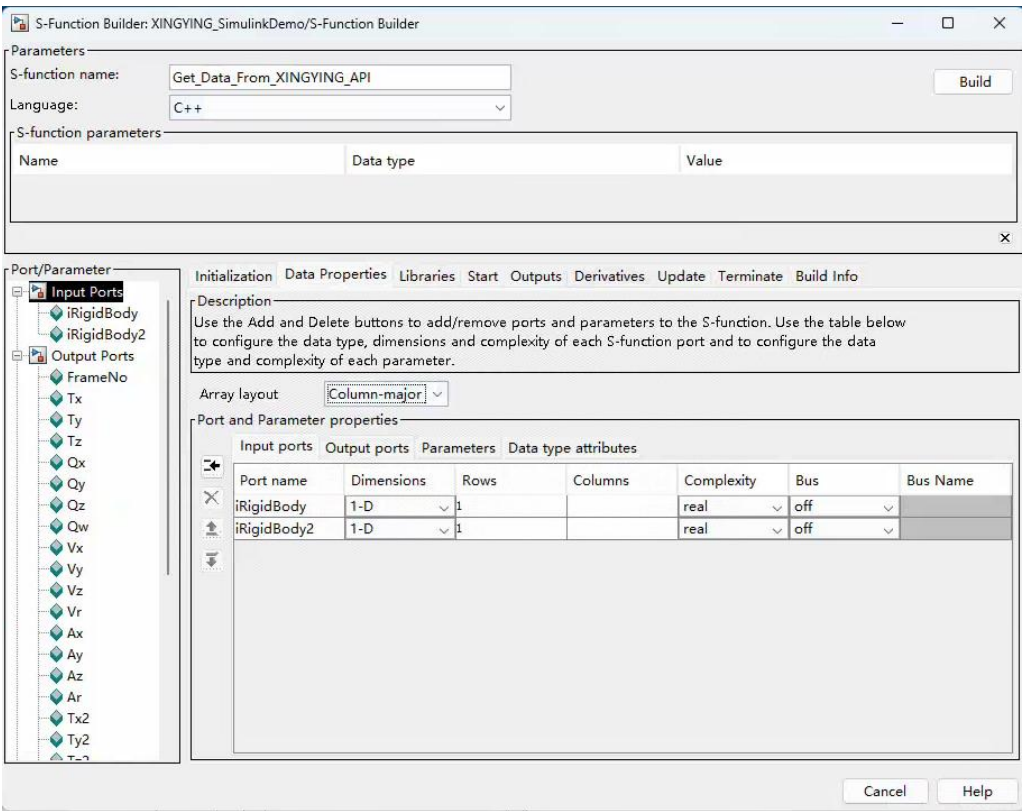


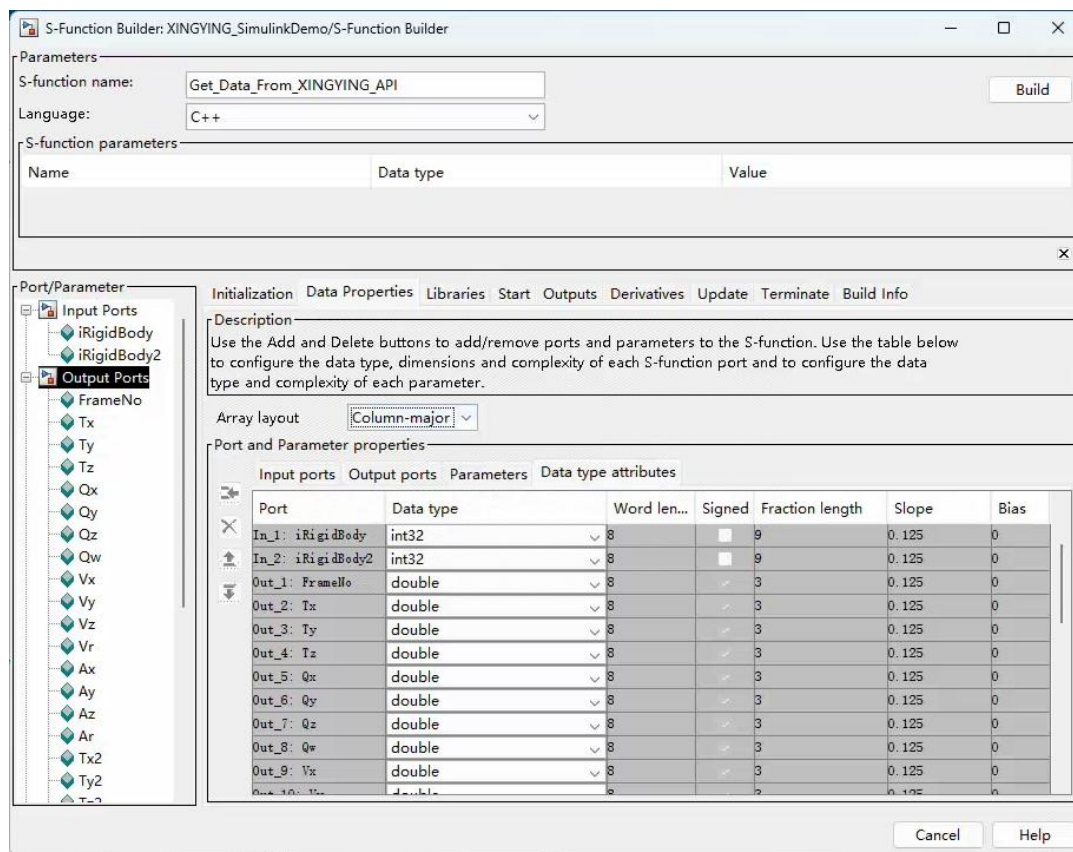
- (3)、Create a project and add the S-Function Builder module
- (4)、Set the programming language to C++
- (5)、Set sampling



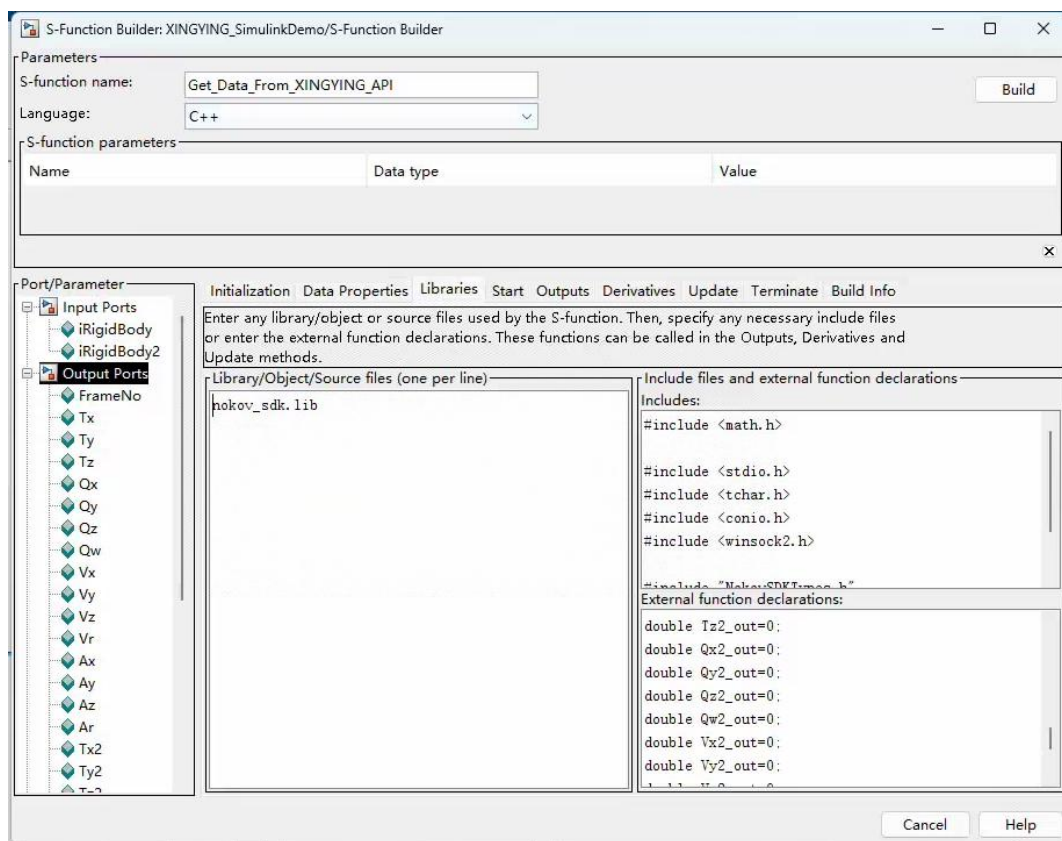
(6)、Set the input port and output port, the data type of the input port is int32, and the data type of the output port is double

Note: If multiple inputs or outputs are needed, you can add input or output ports here.





(7)、Include dynamic library
nokov_sdk.lib



(8)、Include header file

```
#include <math.h>
```

```
#include <stdio.h>
```

```
#include <tchar.h>
```

```
#include <conio.h>
```

```
#include <winsock2.h>
```

```
#include "NokovSDKTypes.h"
```

```
#include "NokovSDKClient.h"
```

```
#include <iostream>
```

(9)、Function declaration

```
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;

```

Note: If multiple outputs need to be defined, intermediate variables can be added here.

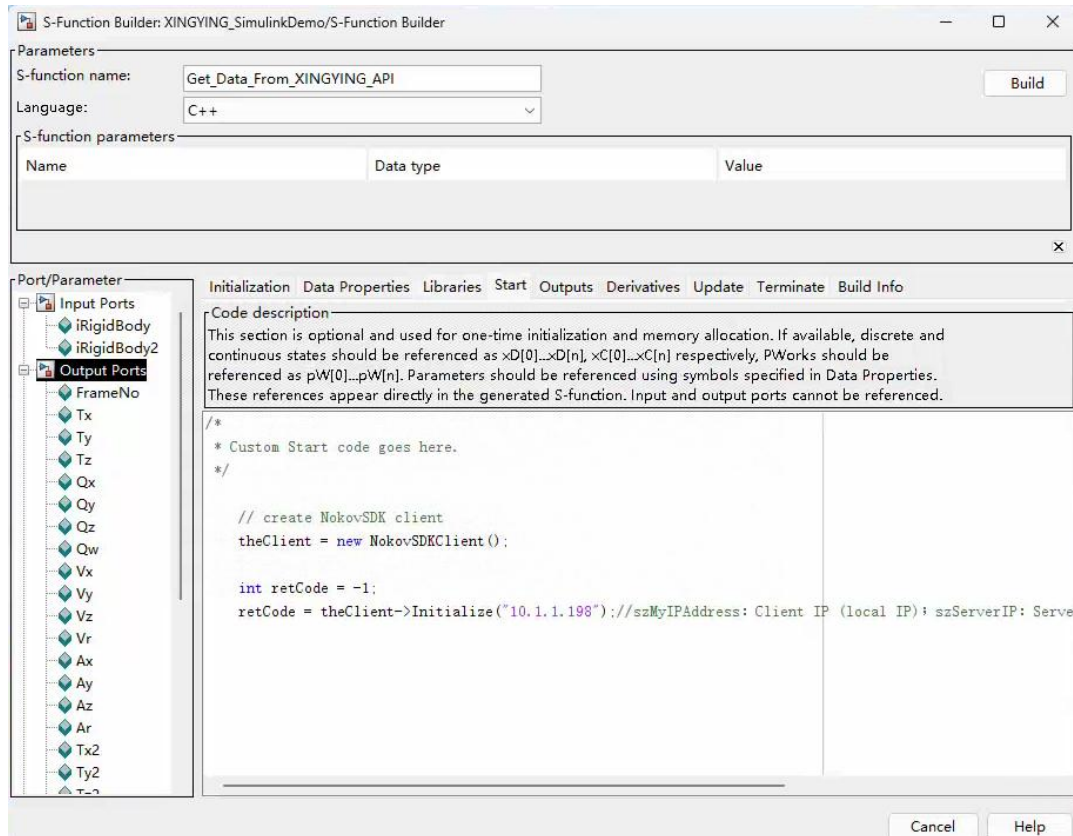
(10) 、 Initialization code added in the initial part

```
// 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)、 Code for receiving data added in the output part (if you need to output more skeletal data, you can modify or add ports accordingly, and add data output code)

```
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)
{
    Tx_out = (real_T) data->RigidBody[iRigidBody_in].x;
    Ty_out = (real_T) data->RigidBody[iRigidBody_in].y;
    Tz_out = (real_T) data->RigidBody[iRigidBody_in].z;
    Qx_out = (real_T) data->RigidBody[iRigidBody_in].qx;
    Qy_out = (real_T) data->RigidBody[iRigidBody_in].qy;
    Qz_out = (real_T) data->RigidBody[iRigidBody_in].qz;
    Qw_out = (real_T) data->RigidBody[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;
}
else
{
    Tx2_out = (real_T) data->RigidBodies[iRigidBody2_in].x;
    Ty2_out = (real_T) data->RigidBodies[iRigidBody2_in].y;
    Tz2_out = (real_T) data->RigidBodies[iRigidBody2_in].z;
    Qx2_out = (real_T) data->RigidBodies[iRigidBody2_in].qx;
    Qy2_out = (real_T) data->RigidBodies[iRigidBody2_in].qy;
    Qz2_out = (real_T) data->RigidBodies[iRigidBody2_in].qz;
    Qw2_out = (real_T) data->RigidBodies[iRigidBody2_in].qw;
    if (pExtData != NULL)
    {
        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;
*Tx = (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;

*Tx2 = (real_T) Tx2_out;
*Ty2 = (real_T) Ty2_out;
*Tz2 = (real_T) Tz2_out;
*Qx2 = (real_T) Qx2_out;

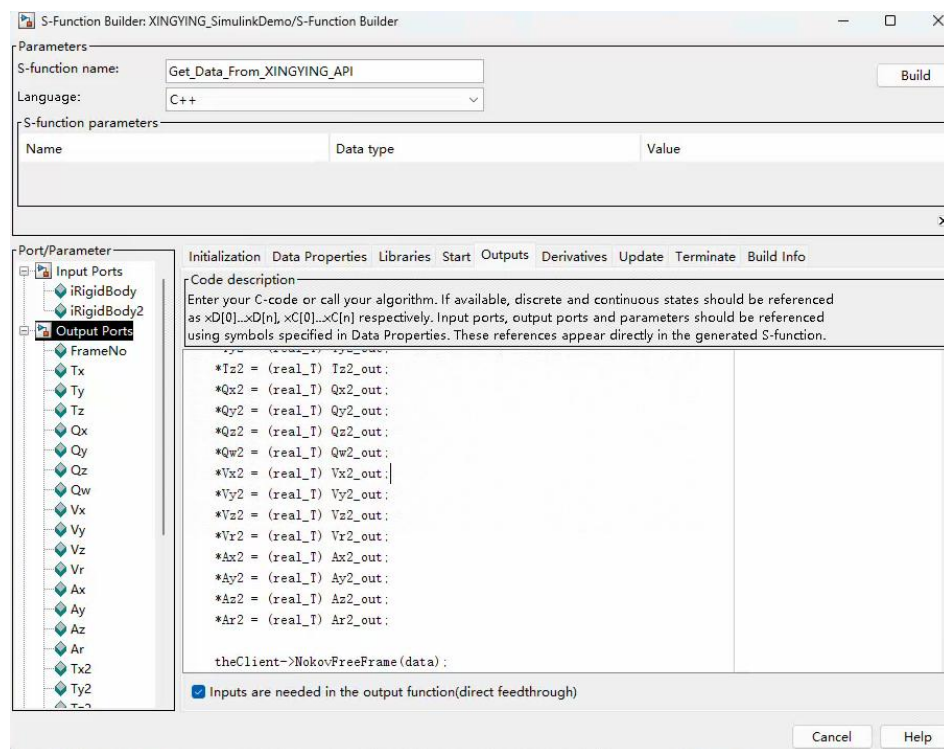
```

```

*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;

```

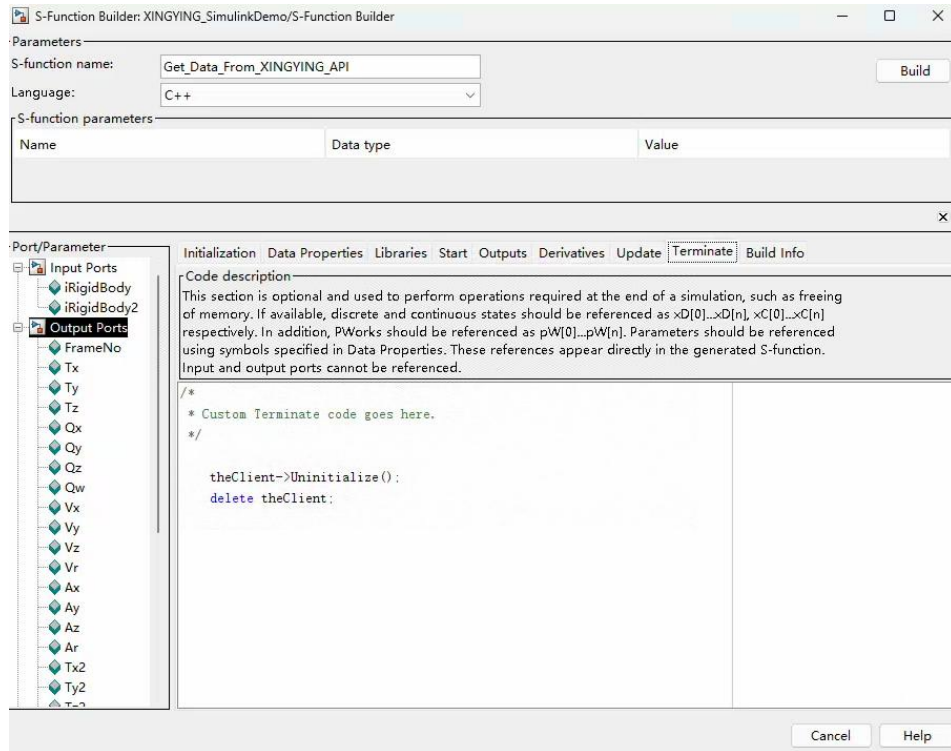
```
theClient->NokovFreeFrame(data);
```



(12) 、Add code to the end part

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

```
delete theClient
```



4. Analysis of the data

(1)、Initialize (XINGYING software NIC Address)

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

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

(2)、Get Data Descriptions

```
sServerDescription ServerDescription;
memset(&ServerDescription, 0, sizeof(ServerDescription));
theClient->GetServerDescription(&ServerDescription);
```

```
sDataDescriptions* ps = nullptr;
theClient->GetDataDescriptions(&ps);
```

dataDescription note:

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

dataDescription.type ----- Description information type; 1---Descriptor_MarkerSet; 2---Descriptor_RigidBody; 3---Descriptor_Skeleton; 4---Descriptor_ForcePlate

dataDescription.Data.MarkerSetDescription->szName--- Markerset name in description information

dataDescription.Data.MarkerSetDescription->nMarkers--- The number of points contained in the

MarkerSet in the description information

dataDescription.Data.MarkerSetDescription->szMarkerNames[markerIndex]--- The name of the markerIndex-th point contained in the MarkerSet in the description information

dataDescription.Data.RigidBodyDescription->szName --- Rigid body name in description information

dataDescription.Data.SkeletonDescription->szName --- Skeleton name in description information

dataDescription.Data.SkeletonDescription->nRigidBodies --- The number of bone segments contained in the skeleton in the description information

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

-- The ID of the boneIndex-th bone segment contained in the skeleton in the description information

dataDescription.Data.SkeletonDescription->RigidBodies[boneIndex] .szName---The name of the boneIndexth bone segment contained in the skeleton in the description information

dataDescription.Data.ForcePlateDescription->nChannels--- The number of channels of the force plate in the description information

dataDescription.Data.ForcePlateDescription->szChannelNames[channelIdx]--- The name of the channelIdxth channel of the force plate in the description information

theClient->FreeDataDescriptions(ps);// Note: need to release ps
ps = nullptr;

(3)、Get data information (note: need to be used correspondingly with (5))

sFrameOfMocapData* data = theClient->GetLastFrameOfMocapData();

data information analysis:

data->iFrame----- Data frame number

data->Timecode----- The time code information of the data requires decoded with(4)

data->TimecodeSubframe----- The time code information of the data requires to be decoded with (4)

data->iTimeStamp----- Data timestamp information

data->nMarkerSets--- Number of MarkerSets of data

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

markerset->szName--- The name of the i-th MarkerSet of the data

markerset->nMarkers--- The number of Markers contained in the i-th MarkerSet of the data

markerset->Markers[i_Marker][0]--- The X value of the i_Marker point contained in the i-th MarkerSet of the data

markerset->Markers[i_Marker][1]--- The Y value of the i_Marker point contained in the i-th MarkerSet of the data

markerset->Markers[i_Marker][2]--- The Z value of the i_Marker point contained in the i-th MarkerSet of the data

data->nRigidBody--- The number of rigid bodies in the data
data->RigidBody[i].ID--- The ID of the i-th rigid body of the data
data->RigidBody[i].x--- The X value of the i-th rigid body of the data
data->RigidBody[i].y--- The Y value of the i-th rigid body of the data
data->RigidBody[i].z--- The Z value of the i-th rigid body of the data
data->RigidBody[i].qx--- The rotation information quaternion Qx of the i-th rigid body of the data
data->RigidBody[i].qy--- The rotation information quaternion Qy of the i-th rigid body of the data
data->RigidBody[i].qz--- The rotation information quaternion Qz of the i-th rigid body of the data
data->RigidBody[i].qw--- The rotation information quaternion Qw of the i-th rigid body of the data

data->RigidBody[i].nMarkers--- The number of Markers contained in the i-th rigid body of the data
data->RigidBody[i].MarkerIDs[iMarker]--- The ID of the iMarker-th Marker contained in the i-th rigid body of the data
data->RigidBody[i].Markers[iMarker][0]--- The X value of the iMarker-th Marker contained in the i-th rigid body of the data
data->RigidBody[i].Markers[iMarker][1]--- The Y value of the iMarker-th Marker contained in the i-th rigid body of the data
data->RigidBody[i].Markers[iMarker][2]--- The Z value of the iMarker-th Marker contained in the i-th rigid body of the data

data->extendData.nExtendDataNum--- The number of Extend Data
data->extendData.extendData[i]---the i-th Extend Data
data->extendData.extendData[i].type--- the type of the i-th Extend Data
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---The X-direction velocity value of the iRigidBody_in rigid body in the extended data
pExtData->extendData.rigidBodyExtend[iRigidBody_in].yVel---The Y-direction velocity value of the iRigidBody_in rigid body in the extended data
pExtData->extendData.rigidBodyExtend[iRigidBody_in].zVel---The Z-direction velocity value of the iRigidBody_in rigid body in the extended data
pExtData->extendData.rigidBodyExtend[iRigidBody_in].rVel---The velocity value of the iRigidBody_in rigid body in the extended data
pExtData->extendData.rigidBodyExtend[iRigidBody_in].xAvel---The X-direction acceleration value of the iRigidBody_in rigid body in the extended data
pExtData->extendData.rigidBodyExtend[iRigidBody_in].yAvel--- The Y-direction acceleration value of the iRigidBody_in rigid body in the extended data

pExtData->extendData.rigidBodyExtend[iRigidBody_in].zAvel--- The Z-direction acceleration value of the iRigidBody_in rigid body in the extended data
pExtData->extendData.rigidBodyExtend[iRigidBody_in].rAvel--- The acceleration value of the iRigidBody_in rigid body in the extended data
pExtData->extendData.rigidBodyExtend[iRigidBody_in].roll;--- Roll angle of the iRigidBody_in-th rigid body in extended data
pExtData->extendData.rigidBodyExtend[iRigidBody_in].pitch;--- Pitch angle of the iRigidBody_in-th rigid body in extended data
pExtData->extendData.rigidBodyExtend[iRigidBody_in].yaw;--- Yaw angle of the iRigidBody_in-th rigid body in extended data
pExtData->extendData.rigidBodyExtend[iRigidBody_in].rollVel;---Roll angular velocity of the iRigidBody_in-th rigid body in extended data
pExtData->extendData.rigidBodyExtend[iRigidBody_in].pitchVel;--- Pitch angular velocity of the iRigidBody_in-th rigid body in extended data
pExtData->extendData.rigidBodyExtend[iRigidBody_in].yawVel;---Yaw angular velocity of the iRigidBody_in-th rigid body in extended data

data->nSkeletons--- The number of data skeletons
data->Skeletons[i].skeletonID--- The ID of the i-th skeleton of the data
data->Skeletons[i].nRigidBody--- The number of bone segments contained in the i-th skeleton of the data
data->Skeletons[i].RigidBodyData[j].ID--- The ID of the j-th bone segment in the i-th skeleton of the data
data->Skeletons[i].RigidBodyData[j].x--- The X value of the j-th bone segment in the i-th skeleton of the data
data->Skeletons[i].RigidBodyData[j].y--- The Y value of the j-th bone segment in the i-th skeleton of the data
data->Skeletons[i].RigidBodyData[j].z--- The Z value of the j-th bone segment in the i-th skeleton of the data
data->Skeletons[i].RigidBodyData[j].qx--- The rotation information quaternion Qx of the j-th bone segment in the i-th skeleton of the data
data->Skeletons[i].RigidBodyData[j].qy--- The rotation information quaternion Qy of the j-th bone segment in the i-th skeleton of the data
data->Skeletons[i].RigidBodyData[j].qz--- The rotation information quaternion Qz of the j-th bone segment in the i-th skeleton of the data
data->Skeletons[i].RigidBodyData[j].qw--- The rotation information quaternion Qw of the j-th bone segment in the i-th skeleton of the data

data->Skeletons[i].RigidBodyData[j].nMarkers--- The number of Markers contained in the j-th bone segment in the i-th skeleton of the data
data->Skeletons[i].RigidBodyData[j].MarkerIDs[iMarker]--- The ID of the i-th Marker contained in the j-th bone segment in the i-th skeleton of the data
data->Skeletons[i].RigidBodyData[j].Markers[iMarker][0]--- The X value of the i-th Marker contained in the j-th bone segment in the i-th skeleton of the data

data->Skeletons[i].RigidBodyData[j].Markers[iMarker][1]--- The Y value of the i-th Marker contained in the j-th bone segment in the i-th skeleton of the data
data->Skeletons[i].RigidBodyData[j].Markers[iMarker][2]--- The Z value of the i-th Marker contained in the j-th bone segment in the i-th skeleton of the data

data->nOtherMarkers--- The number of unnamed markers in the data
data->OtherMarkers[i][0]--- The X value of the iMarker unnamed Marker of the data
data->OtherMarkers[i][1]--- The Y value of the iMarker unnamed Marker of the data
data->OtherMarkers[i][2]--- The Z value of the iMarker unnamed Marker of the data

data->nAnalogdatas--- Number of analog channels of data
data->Analogdata[i]--- The data of the i-th analog channel of the data

data->nLabeledMarkers--- The number of all named points of the data
data->LabeledMarkers[iLabeledMarkers].x--- The X value of the iLabeledMarkers-th named point of the data
data->LabeledMarkers[iLabeledMarkers].y--- The Y value of the iLabeledMarkers-th named point of the data
data->LabeledMarkers[iLabeledMarkers].z--- The Z value of the iLabeledMarkers-th named point of the data
data->LabeledMarkers[iLabeledMarkers].size--- The size of the iLabeledMarkers-th named point of the data

(4)、Time information decoding

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

(5)、Release data information memory (note: To be used in correspondence with (3))

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

(6)、The code added at the end to disconnect from the motion capture system

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

5. Data description

(1)、The coordinate system is a right-handed system

(2)、For Markers that have been defined in the Markerset, if points are lost or cannot be recognized due to software operation problems, Marker occlusion, etc., the X, Y, and Z coordinate values will be filled in as 9999999.000000