

SYNTHRON Somatosensory algorithm dynamic link library manual

Chapter 1 Parameters

Note: The number of bytes corresponding to the following data types

Data type	float	long int	short	int
Number of bytes	4	4	2	4

1.1 Platform selection

The SIM_CUE_DLL.dll dynamic link library integrates somatosensory control algorithms for three platforms: six degrees of freedom, three plus one rotation axis, four degrees of freedom, and three degrees of freedom. The type of target platform can be selected by calling a function.

Function Choose_PlatformType(int PlatformType) set the type of the platform.

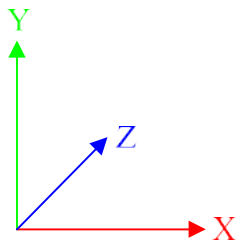
PlatformType = 0x00 means the target platform is 4 degree or 3 degree of platform ; PlatformType = 0x01 means the target platform is 6 degree of platform.

1.2 Game parameters

The game parameters are as follows. The following six parameters are real-time game parameters, which are refreshed in real time based on the real-time posture information of the moving entity in the game.

```
typedef struct    {
    float F32ReciveAlfaRad; /* unit: radian*/ float
    F32ReciveBetaRad; /* unit: radian */ float
    F32ReciveGammaRad; /* unit: radian*/
    float F32ReciveXAcceG; /* unit: Acceleration of
    gravity */ float F32ReciveYAcceG; /* unit:
    Acceleration of gravity */ float F32ReciveZAcceG;
    /* unit: Acceleration of gravity */
}DOF6_GAME_PARA;
```

For example: The axis XYZ in the game is as shown in the picture:



Taking a racing car as an example, the rotation angle of the car body around the X axis is $F32ReciveAlfaRad$, the rotation angle around the Y axis is $F32ReciveBetaRad$, the rotation angle around the Z axis is $F32ReciveGammaRad$, and the linear acceleration along the X axis is marked by the acceleration of gravity to $F32ReciveXAcceG$, along The linear acceleration along the Y axis to the acceleration of gravity is $F32ReciveYAcceG$, the linear acceleration along the Z axis to the gravitational acceleration is $F32ReciveZAcceG$. During the movement of the car, the rotation angle and linear acceleration change in real time, so it is necessary to refresh the game parameters in real time according to the game information.

Note :

1 、 6DOF motion platform: F32ReciveAlfaRad, F32ReciveBetaRad, F32ReciveGammaRad, F32ReciveXAcceG, F32ReciveYAcceG, F32ReciveZAcceG----All six values exist.

2 、 3+1 axis platform: F32ReciveAlfaRad, F32ReciveBetaRad, F32ReciveGammaRad, F32ReciveYAcceG----- Four values exist, the other two values should be set to 0.

3、 3DOF motion platform: F32ReciveAlfaRad, F32ReciveGammaRad, F32ReciveYAcceG---- Three values exist, the other three values should be set to 0.

The above is because the mechanical structure of the platform causes the platform to lack rotation and translation in one or several directions.

1.3 Special effect location information

Some position information can be superimposed on the basis of the somatosensory algorithm, the parameters are as follows.

```
typedef struct    {  
    long int  I32PlayXpos; //X axis position  
    information  
    long int  I32PlayYpos; //Y axis position  
    information  
    long int  I32PlayZpos; //Z axis position  
    information  
    long int  I32PlayUpos; //U axis position  
    information  
    long int  I32PlayVpos; //V axis position  
    information  
    long int  I32PlayWpos; //W axis position  
    information  
}DOF6_POSTION_PARA;
```

Special effect position information can be superimposed on the basis of somatosensory algorithm according to needs, such as vibration, jitter, etc. X~W correspond to 1~6 in turn axis.

1.4 Somatosensory parameters

Somatosensory parameters as below:

```
typedef struct {  
    float F32T1; /* Time factor T1 */  
    float F32T2; /* Time factor T2 */  
    float F32T3; /* Time factor T3 */  
    float F32T4; /* Time factor T4 */  
    float F32T5; /* Time factor T5 */  
    float F32T6; /* Time factor T6 */  
    float F32T7; /* Time factor T7 */  
    float F32T8; /* Time factor T8 */  
  
    float F32C1; /* Acceleration threshold 1 */  
    float F32C2; /* Acceleration threshold 2 */  
    float F32C3; /* Acceleration threshold 3 */  
    float F32C4; /* Acceleration threshold 4 */  
    float F32C5; /* Acceleration threshold 5 */  
    float F32C6; /* Acceleration threshold 6 */  
    float F32C7; /* Acceleration threshold 7 */  
}
```

```
float F32C8; /* Acceleration threshold 8 */
```

```
float F32K1; /* Scale Factor 1 */  
float F32K2; /* Scale Factor 2 */  
float F32K3; /* Scale Factor 3 */  
float F32K4; /* Scale Factor 4 */  
float F32K5; /* Scale Factor 5 */  
float F32K6; /* Scale Factor 6 */  
float F32K7; /* Scale Factor 7 */  
float F32K8; /* Scale Factor 8 */  
} DOF6_CUE_PARA;
```

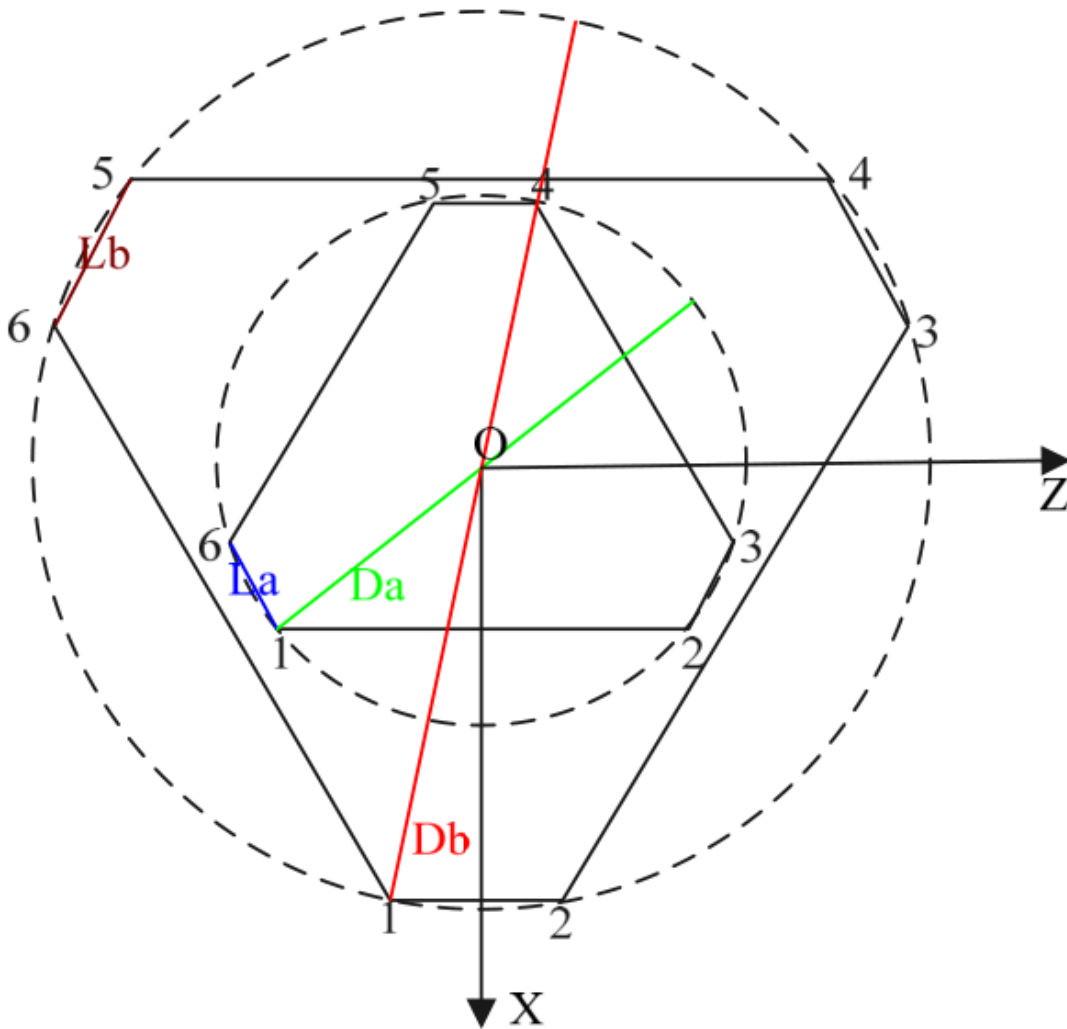
The initial values of somatosensory parameters can be assigned according to the default values. When the platform is to be debugged, these parameters can be adjusted according to the dynamic effects of the platform. If you want the platform to move softly, you can first increase the T1 parameter to 10~1000; reduce the K1 parameter to 0.01~0.1 ; Increase the T4 parameter to 0.5~10 for combined adjustment, otherwise, perform the opposite adjustment.

Note: Somatosensory parameters only affect acceleration

1.5. Mechanical parameters of the platform

The mechanical parameters of the platform are shown below.

```
typedef struct {  
    //Maximum movement range in the game  
    float F32AlfaMaxRad; /* Unit: radians */  
    float F32BetaMaxRad; /* Unit: radians */  
    float F32GammaMaxRad; /* Unit: radians */  
    float F32XAcceMaxG; /* Unit: acceleration of gravity */  
    float F32YAcceMaxG; /* Unit: gravity speed */  
    float F32ZAcceMaxG; /* Unit: acceleration of gravity */  
  
    //Three degrees of freedom platform  
    float F32TriangleHemMm; /* The length of the base of the triangle, in millimeters */  
    float F32HeightOfTriangleHemMm; /* The height of the base of the triangle, in millimeters */  
    float F32HeightToTopMarkMm; /* The distance from the vertex of the triangle to the center of the seat, unit: Mm */  
  
    //The size of the upper and lower planes of the six-degree-of-freedom platform  
    float F32StaticShortMm; /* Short side of the lower platform, unit: mm */  
    float F32StaticDiameterMm; /* Diameter of the lower platform, unit: mm */  
    float F32MovingShortMm; /* The short side of the upper platform, unit: mm */  
    float F32MovingDiameterMm; /* Diameter of the upper platform, unit: mm */  
  
    //The amount of rotation relative to the initial coordinates  
    float F32XAxisRotAngleDeg; /* Coordinate rotation angle */  
  
    //The mechanical structure of the platform determines its range of motion  
    float F32PlatformAlfaMaxDeg; /* Alfa maximum motion range, unit: degree */  
    float F32PlatformBetaMaxDeg; /* Beta maximum motion range, unit: degree */  
    float F32PlatformGammaMaxDeg; /* Gamma maximum motion range, unit: degree */  
}
```

As shown in the figure above, L_b is the short side of the lower platform $F32StaticShortMm$, and D_b is the diameter of the circumscribed circle of the six hinge points of the lower platform $F32StaticDiameterMm$, L_a is the short side of the upper platform $F32MovingShortMm$, D_a is the circumscribed diameter of the upper six hinge points $F32MovingDiameterMm$.

The amount of rotation relative to the initial coordinate only acts on the six-degree-of-freedom platform. As shown in Figure 3, the initial coordinates of the six-degree-of-freedom platform X. The positive direction of the axis is located on the vertical bisector of the hinge points 1 and 2 of the lower platform. If necessary, the positive direction of the X axis is located on the vertical bisector of the hinge points 3 and 4.

On the line, it is equivalent to rotating the coordinate system 120 degrees counterclockwise, that is, $F32XaxisRotAngleDeg$ is assigned a value of 120.

The mechanical structure of the platform determines its range of motion: For a three-degree-of-freedom platform, as shown in Figure 2, the platform rotates the most on the X axis.

The large angle is $F32PlatformAlfaMaxDeg$, the maximum angle of rotation around the Z axis is $F32PlatformGammaMaxDeg$,

The translation distance along the Y axis is $F32PlatformYMaxMm$. For a six-degree-of-freedom platform, as shown in Figure 3, the platform is around the X axis.

The maximum angle of rotation is $F32PlatformAlfaMaxDeg$, and the maximum angle of rotation around the Y axis is

$F32PlatformBetaMaxDeg$, the maximum angle of rotation around the Z axis is $F32PlatformGammaMaxDeg$,

along the X axis

The translation distance is F32PlatformXMaxMm, and the translation distance along the Y axis is F32PlatformYMaxMm, along the Z axis

The translation distance is F32PlatformZMaxMm.

The parameters of the electric cylinder: F32AccessDistanceUnitMm is the stroke of the electric cylinder, F32LeadDistanceUnitMm is the lead of the electric cylinder, F32MinLongofCylinderMm The minimum length of the electric cylinder is the length of the diagonal line between the two connected hinge points of the upper and lower platforms when the upper platform falls at the lowest height

The gear ratio is the reduction ratio when the motor and the electric cylinder are linked, and it is 1 if there is no reduction.

1.6 UDP port number and sampling time

The UDP port number and sampling time are shown in the figure below.

```
typedef struct {  
    float F32FlightSamplingPeriods; /* Sampling time, unit: second*/  
    short I16HostTxPort; /* Host sending UDP port */  
    short I16HostRxPort; /* Host receiving UDP port */  
    short I16MboxTxPort; /* MBOX sending UDP port */  
    short I16MboxRxPort; /* MBOX receiving UDP port */  
  
    short I16WhoAcceptCode;  
    short I16WhoReplyCode;  
    int I16BaseDoutCode; //Special effect output  
    long int I32PlayLine; //Check output  
}UDP_PORT_PARA;
```

F32FlightSamplingPeriods is the time interval at which the game data DOF6_GAME_PARA is updated.

I16HostTxPort, I16HostRxPort, I16MboxTxPort, I16MboxRxPort are UDP port numbers. For details, please refer to the MBOX manual.

I16WhoAcceptCode, I16WhoReplyCode means who receives it and who responds, please refer to the MBOX manual for details.

I16BaseDoutCode is special effect output, I32PlayLine is check output.

Chapter 2 Function

2.1. Platform type selection function

Select the type of platform to be controlled, which needs to be called at the beginning of the program. The input parameter PlatformType is 0x00: 3+1 axis platform\3 axis platform; 0x01: 6 axis platform. The return value of the function is the platform type.

2.2. The return value of the function

All the following functions have return values, and the return values are defined as follows:

0x20000000	Function call succeeded
0x20000001	The dongle failed to open, the function call failed
0x20000002	The dongle ID verification error, the function call failed
0x20000003	HAMC authentication error of dongle, function call failed
Other value	Other value function call failed

Table 1

2.3. Somatosensory parameter reset function

DOF6_Public_CueModule_Reset();

To reset the somatosensory parameters, it needs to be called at the beginning of the program. There are no parameters, and the return value is shown in Table 1.

2.4. Parameter transfer function

DOF6_Public_UserCueParaTranfer(DOF6_SYS_PARA *p);

Pass all parameters to the dynamic link library.

The return value is shown in Table 1.

2.5. Mechanical parameter initialization function

DOF6_Public_MechModule_InitCa();

Calculate the coordinates and median height of the platform.

There are no parameters, and the return value is shown in Table 1.

2.6. Open UDP port function

Public_OpenMboxUdpPort();

Open the UDP port.

There are no parameters, and the return value is shown in Table 1.

2.7. Special effect output function

Public_DoOutControl(short BaseDoutCode, short ExtDoutCode);

Special effect control commands are sent through the UDP port to control the special effect output of the platform, which can realize special effect output such as wind, spray, water spray, and flash.

BaseDoutCode is the basic 12 outputs, and ExtDoutCode is the extended 12 outputs.

The low 12bit of each data binary corresponds to 12 outputs. For example, the 0th bit corresponds to the 1st output. When the bit value is 1, it is on, and when it is 0, it is off.

B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
KEEP				12 digital output settings											

2.8. Somatosensory algorithm function

DOF6_Public_Cue2Inverse_Solution(DOF6_GAME_PARA *p, DOF6_POSTION_PARA *q, int PositonEnable);

Send data through UDP port to control platform movement.

The input parameter DOF6_GAME_PARA *p is the game parameter refreshed in real time.

The input parameter DOF6_POSTION_PARA is the number of special effect position pulses for six cylinders.

Enter the parameter PositonEnable to select whether to superimpose the special effect

DOF6_POSTION_PARA *q. When PositonEnable =1, the function superimposes the

DOF6_POSTION_PARA *q parameter; when PositonEnable =0, the function does not superimpose DOF6_POSTION_PARA *q.

The return value of the function is shown in Table 1.

2.9. Platform reset function

Public_ResetPlatform();

Send data through UDP port to control platform reset.

There are no parameters, and the return value is shown in Table 1

2.10. The platform reaches the median height function

Public_GoMiddlePlatform(long int GoMiddleTimeMs);

Send data through the UDP port to control the platform to reach the median height.

The input parameter GoMiddleTimeMs is the time for the platform to reach the median height, in milliseconds. The return value is shown in Table 1.

2.11. Platform reach zero function

Public_GoZeroPlatform(long int GoZeroTimeMs);

Send data through the UDP port to control the platform to reach the zero position.

The input parameter GoZeroTimeMs is the time when the platform reaches the zero point, in milliseconds. The return value is shown in Table 1.

2.12. Close UDP port function

Public_CloseMboxUdpPort();

Close the UDP port.

There are no parameters, and the return value is shown in Table 1.

2.13. Function call sequence

1. The program runs the initial call Choose_PlatformType(int PlatformType) function to select the type of control platform, and then call the DOF6_Public_CueModule_Reset() function to initialize the somatosensory parameters;
2. Assign values to all the parameters in the DOF6_SYS_PARA structure. The assignment process needs to be based on the actual situation of the platform game;
3. Call DOF6_Public_UserCueParaTransfer to transfer structure parameters;
4. Call the DOF6_Public_MechModule_InitCa() function to initialize the mechanical parameters;
5. Call the Public_OpenMboxUdpPort() function to open the UDP port;
6. Call the Public_GoMiddlePlatform(long int GoMiddleTimeMs) function to control the platform to reach the median height;
7. Call the Public_DoOutControl(short BaseDoutCode, short ExtDoutCode) function to control the special effects, if
There is no need to call the function without special effect output.
8. Call
DOF6_Public_Cue2Inverse_Solution(DOF6_GAME_PARA*p,DOF6_POSTION_PARA*q,int PositonEnable) function to control platform movement. This function needs to update the parameter DOF6_GAME_PARA regularly. This function can be put into the timer and send data to control platform movement at regular intervals;
9. Public_ResetPlatform() platform reset function, Public_GoZeroPlatform(long int GoZeroTimeMs) platform
The reset function can be called as needed;
10. When the program is closed, call the Public_CloseMboxUdpPort() function to close the UDP port.