Motion Controller
API Function Programming Manual

Version 1.5

2017.5.9

Copyright

The Manual's information and data are for reference only. The Company reserves the final interpretation right of data. We may upgrade design and function at all times without a prior notice.

Pay attention to safety while debugging the Machine! The user must design valid safety protection device in the Machine or set error handling program in the software; otherwise, the losses will not be undertaken by Optics Focus.

Table of Contents

Document Version

Version No.	Revision date	Remarks
V1.5	2017-5-9	

Chapter 1 Introduction to Use of API Function

1.1 Architecture of API Programming System Software

Based on dynamic link library (DLL) provided by Leadshine, the user may use the API function from DLL to realize the functions required. The so-called API programming means application codes are programmed in PC and the relevant functions are realized through API function from DLL. For example, the DLL provided by SMC606 includes three files, i.e. LTSMC.dll, LTSMC.h and LTSMC.lib; presently, it supports the dynamic libraries of multiple programming language versions, including C#, VB, VC, VB.NET, VC.NET, LABVIEW and DELPHI under MICROSOFT WINDOWS system, as well as Xcode environment under MAC system.

SMC600 series controller provides DLL for different application environment:

The DLL in folder "WINCE_DLL" applies to programming under WINCE environment;

The DLL in "WINDOWS_PC_DLL_32" applies to programming under 32-bit environment;

The DLL in "WINDOWS_PC_DLL_64" applies to programming under 64-bit environment.

In Windows system, the user may develop the corresponding programs by using any tools which support DLL. The use of motion controller DLL in development tool is introduced by taking Visual C++, Visual Basic and C3 as example below.

In MAC system, the user may have programing in Xcode environment.

1.2 Introduction to Development based on VC 6.0 Software

The general method of developing application software via VC is introduced by taking the example of writing a PT application software under Visual C++ 6.0 as follows:

1) Open Visual C++6.0.
2) Create a project.
3) Select MFC APP Wizard(exe).
4) Select project saving route, such as E:\.
5) Enter the project name, such as SMC_EXAMPLE, as shown in Fig. 1.1.

Fig. 1.1 New project

6) Select “Basic dialogue box” in application program type, click “Finish” key to create a project.

7) Find the files LTSMC.h, LTSMC.lib and LTSMC.dll in the header file directory of data CD DLL, and copy themto the directory of :\SMC_EXAMPLE.

8) Enter the menu, select “Project” → “Add a project” → “File”, select the file LTSMC.lib and LTSMC.h and add them into the project.

9) Open the file SMC_EXAMPLE.cpp and SMC_EXAMPLEDlg.cpp, add the corresponding phrase “#include“LTSMC.h” at the beginning part of program, as shown in Fig. 1.2.

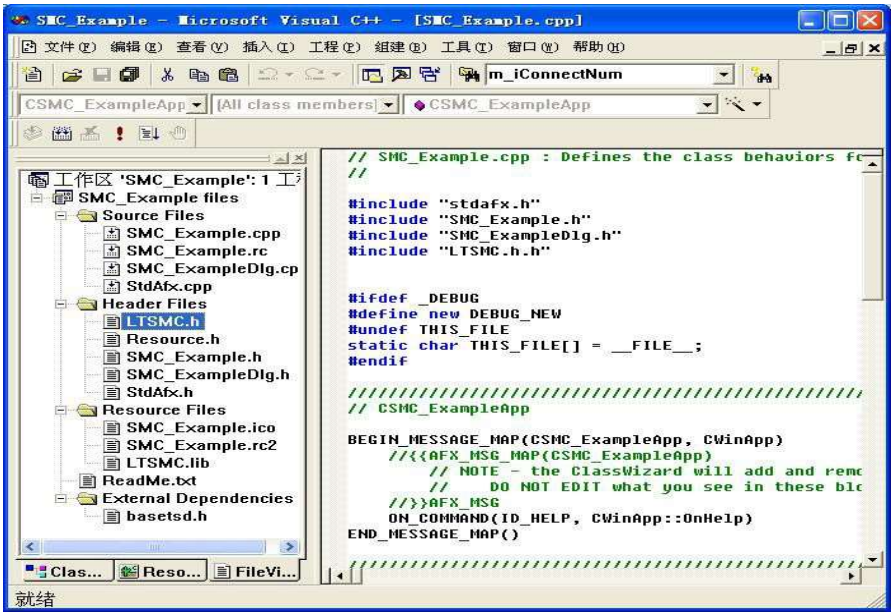


Fig. 1.2 Add header file in program

10) Add “Start” and “Stop” button; rename them as “IDC_BUTTON_Start” and “IDC_BUTTON_Stop” respectively, as shown in Fig. 1.3.

Fig. 1.3 Add dialogue box

11) Double click the window interface, add the code in the function of CSMC_EXAMPLEDlg:CSMC_EXAMPLEDlg(CWnd* pParent/* = NULL*/):CDialog(CSMC_EXAMPLEDlg:IDD, pParent) and connect them via serial port:

smc_board_init(0, 1, “com1”, 115200)

12) Double click “Start” button and add the code in the function of CSMC_EXAMPLEDlg:OnBUTTONStart():

smc_set_profile_unit(0, 0, 500, 5000, 0.01, 0.01, 500);

smc_pmove_unit(0, 0, 200000, 0);

13) Double click “Stop” button and enter the code in the function event of CSMC_EXAMPLEDlg: OnBUTTONStop():

smc_stop(0, 0, 0); as shown in Fig. 1.4:

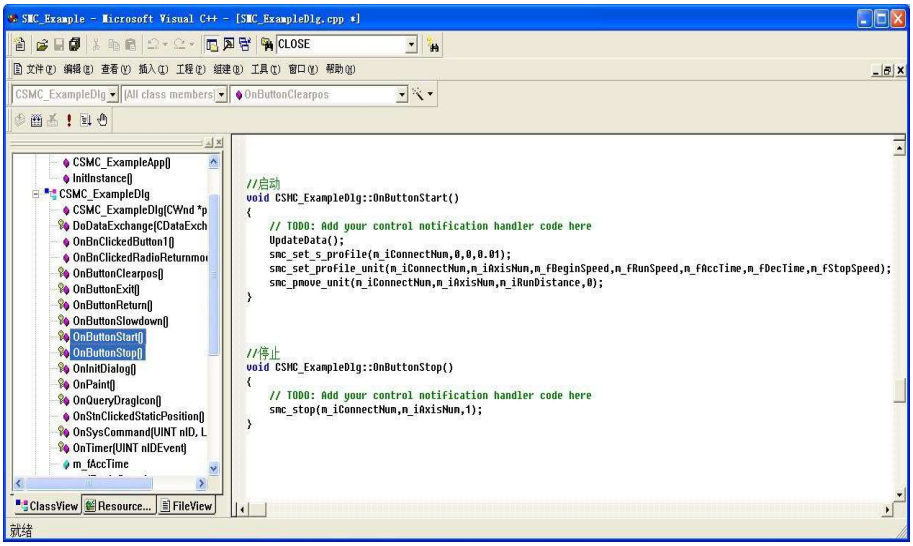


Fig. 1.4 Call library function of motion controller in program

15) Add a member function OnCancel in CSMC_EXAMPLEDlg and add the code in the function of OnCancel as follows:

CDialog:OnCancel ();

smc_board_close(0);

16) Once the program is compiled, run it and it will show the interface as shown in Fig. 1.5. Press "Start" button and the 0th axis will output the pulse with length of 200000; press "Stop" button during motion to slow down and stop pulse output.

Fig. 1.5 Interface of program running

1.3 Introduction to Development based on VC 6.0 Software

The general method of developing application software via VB is introduced by taking the example of writing a PT application software under Visual Basic 6. environment as follows:

1) Create a new directory in the magnetic disk, such as E:\test1

2) Open Visual Basic 6.0, create a "Label EXE" project, add "Start" and "Stop" button on the dialogue box, and modify them into "CB_Start" and "CB_Stop" respectively, as shown in Fig. 1.6.

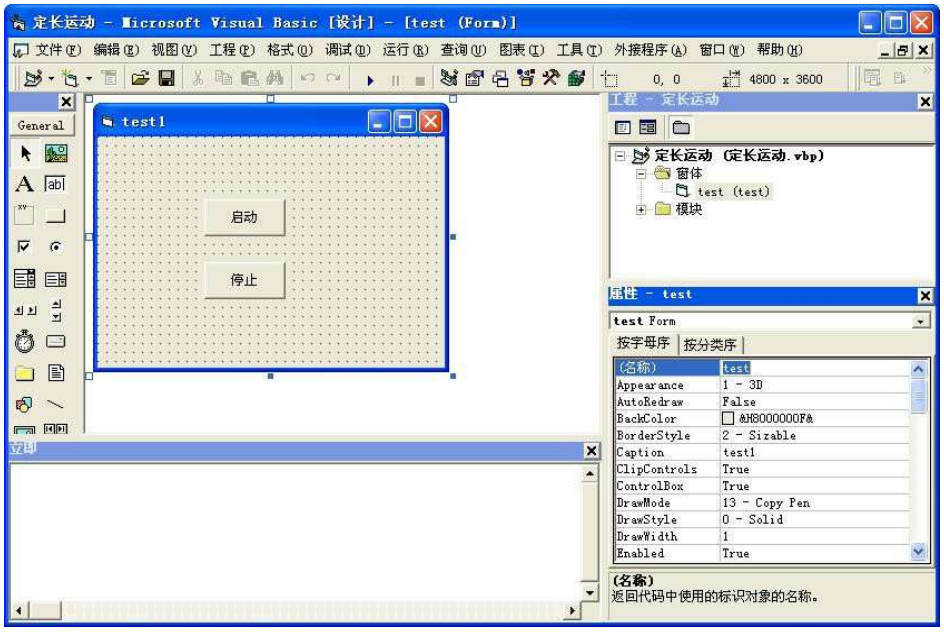


Fig. 1.6 Modify dialogue box

3) Save the project in the directory of E:\test1.

4) Find the file LTSMC.bas and LTSMC.dll in the header file directory of data CD DLL, and copy them to the directory of testl.

5) Select “Select Project → Add module → Existing, find” in the menu, find the file LTSMC.bas in the directory of test1 and add it in the project, as shown in Fig. 1.7.

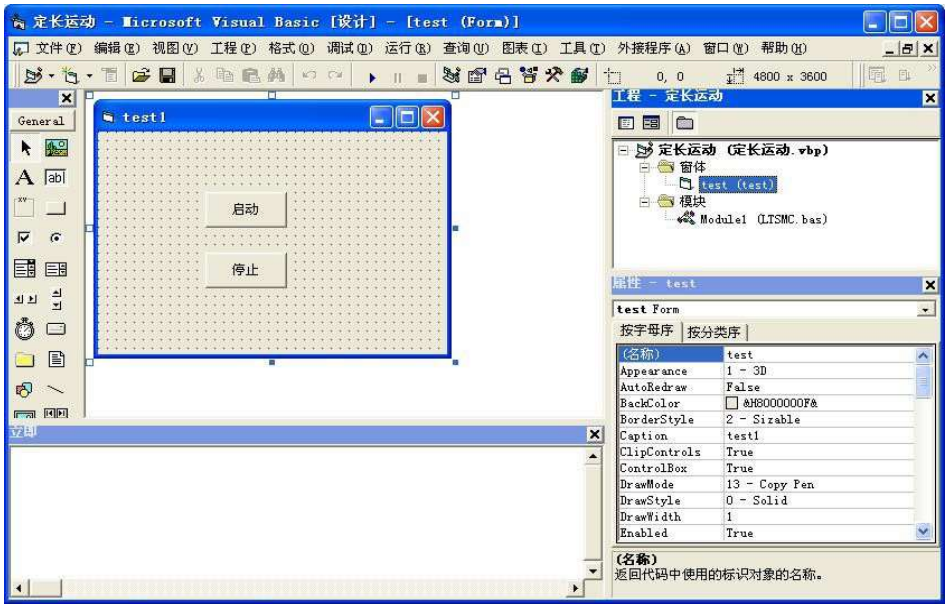


Fig. 1.7 Add header file

6) Connect the controller via Ethernet port. Refer to the specific function instruction in Fig. 1.8.

(1) Double click window control to add the code in the event of Form_Load

```
smc_board_init 0, 2, "192.168.5.11", 0
```

(2) Double click “Start” button to add the code in the event of CB_Start_Click as follows:

```
smc_set_profile_unit 0, 0, 500, 5000, 0.01, 0.01, 500
smc_pmove_unit 0, 0, 200000, 0
```

(3) Double click “Stop” button to add the code in the event of CB_Stop_Click() as follows:

```
smc_stop 0, 0, 0
```

(4) Add code in the event of Form_Unload

```
Smcboardclose (0)
```

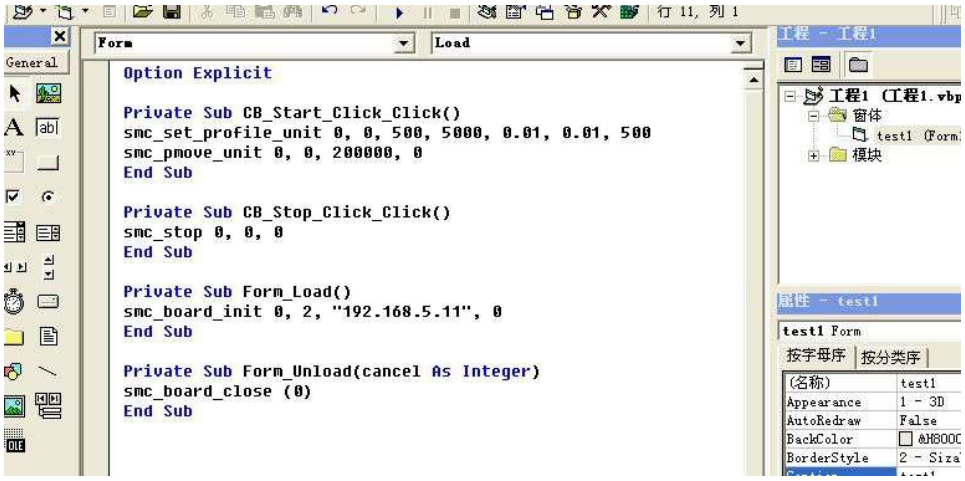


Fig. 1.8 Call library function of motion controller in program

7) Once the program is compiled, run it to display the interface as shown in Fig. 1.9. Press "Start" button and the 0th axis will output the pulse with length of 200000; press "Stop" button during motion to slow down and stop pulse output.

Fig. 1.9 Interface of program running (VB)

1.4 Introduction to Development based on C# Software

The general method of developing application software via c# is introduced by taking the example of writing a PT application software under c# environment as follows:

1) Create a new directory in the magnetic disk, such as E: \ C_Sharp

2) Open C#2020, create a project of "windows window application program", add "Start" and "Stop" button on the dialogue box and modify them into "CB_Start" and "CB_Stop" respectively, as shown in Fig. 1.10.

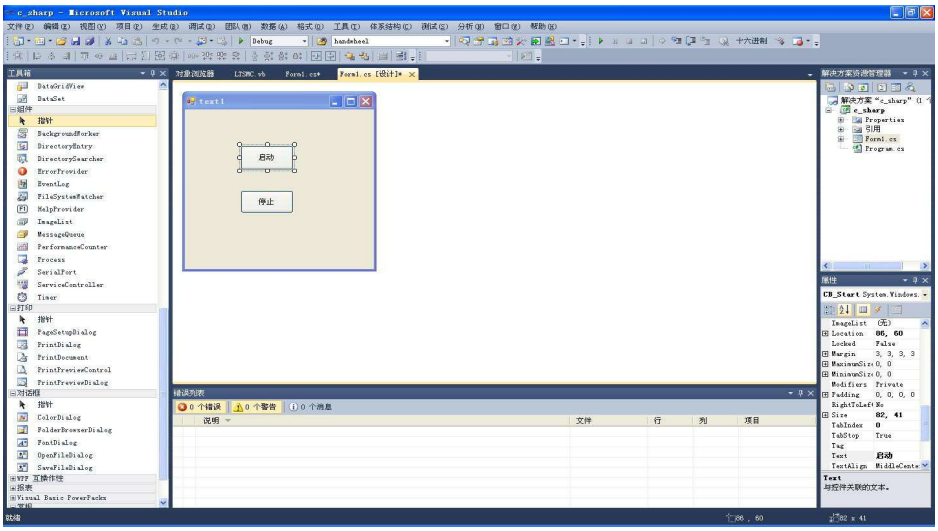


Fig. 1.10 Program editing interface

3) Save the project in the directory of E:\C_Sharp.

4) Find the file LTSMC.CS and LTSMC.dll in the header file directory of data CD DLL, and copy them to the directory of ..BIN/DEBUG.

5) Right click project name “C_Sharp” → “Add” → “Existing item” in menu, find the file LTSMC.cs in the directory of test1 and add it in the project.

6) Add header file and event; add header file usingLeadshine; (Leadshine is the namespace name in LTSMC.CS) and connect the controller via Ethernet port.

(1) Double click “Start” button in FORM and write code in the software editor

```
private void CB_Start_Click(object sender, EventArgs e)
{
    ushort CardNo = 0;              //Card No.
    ushort axis = 0;                //Motion axis No.
    double start_speed = 0;         //Starting speed
    double speed = 1000;            //Max. running speed
    double stop_speed = 0;          //Stop speed
    double tacc = 0.1;              //Acceleration
    double tdec = 0.1;              //Deceleration
    double s_pare = 0.05;           //s-shaped smoothing factor
    double dist = 10000;            //Motion distance
    LTSMC.smc_set_profile_ui\it(CardNo, axis, start_speed, speed, tacc, tdec, stop_speed);    //Set speed
    parameter
    LTSMC.smc_set_s_profile(CardNo, axis, 0, s_pare);       //Set s smoothing factor
    LTSHC.smc_pmove_unit(CardNo, axis, dist, 0);    //Start fixed-length motion
}
```

(2) Double “Stop” button in FORM and write code in software editor

```
private void CB_Stop_C1ick(object sender, EventArgs e)
{
     ushort CairdWo = 0;          //Card No.
    ushort axis = 0;              //Return axis
    ushort mode = 0;              //Stop mode,  0: Decelerate to stop;  1: Emergency stop
    LTSMC.smc_stop(CardNo, axis, mode);      //Stop motion
}
```

(3) Double the frame of window form, write code in the software editor and connect the controller via Ethernet port.

```
private void Forml_Load(objectsender, EventArgs e)
```

```
{
    ushort CardNo = 0;
    //short res = LTSMC.smc_board_init(CardNo, 1, "C0M1", 115200);  //Type of serial port connection is 1
    short res = LTSMC.smc_board_init(CardNo, 2, "192.168.5.11", 0);    //Type of Ethernet port connection is 2
    if(res != 0)
    {
       MessageEox.Show(string.Format ("Controller connection failure,  error code: {0}",  res),  "error");
    }
}
```

7) Click "Run program" or press F5 key to start auto running of program, as shown in interface below: Click "Start" to run the controller axis, or click "Stop" to stop running, as shown in Fig. 1.11.

Fig. 1.11 Program running interface (C#)

Chapter 2 Functioning

2.1 Parameter setting

2.1.1 Controller initialization

Make sure to call the function of smc_board_init to distribute resources for motion controller before operating the motion controller. Call the function of smc_board_close to release the PC system resources occupied by motion controller when program has finished the operation of motion controller, so the occupied resources can be used by other equipment.

Relevant functions:

Name	Function	Reference
smc_board_init	Controller initialization function	Section 3.1
smc_board_close	Controller shutdown function	

⚠Note: Make sure to initialize the controller before controller is connected; the connection mode includes Ethernet port and serial port.

Example 1: Assuming that computer serial port is “C0M1”, serial port baud rate is 115200, stop bit is 2 and no check 0

```
short iret = smc_board_init(0, 1, "COM1", 115200);     //Default connection mode of serial port
short iret = smc_board_init_ex(0, 1, "COM1", 115200, 8, 0, 2);   //Advanced connection of serial port
```

Example 2: Assuming that controller IP is “192.168.5.11” and have connection via Ethernet port.

```
Short iret = smc_board_init(0, 2, "192.168.5.11", 0);     //Connection via Ethernet port
```

2.1.2 Setting of pulse output mode

The stepping/servo motor is controlled by Leadshine controller through command pulse. In consideration of various signal interfaces (6 types in general) from different manufacturers of motor drive, set the motor drive of motion controller based on the type of signals received by motor drive, and set the pule output mode of motion controller correctly by using the function of smc_set_pulse_outmode, so as to ensure normal functioning of motor.

Relevant functions:

Name	Function	Reference
smc_set_pulse_outmode	Set the pulse output mode of designated axis	Section 3.2
smc_get_pulse_outmode	Read the pulse output mode of designated axis	

2.1.3 Setting of pulse equivalent

Pulse equivalent setting function, which is provided by Leadshine controller, is used to define the position (displacement) unit; the advanced motion functions are also available.

Relevant functions:

Name	Function	Reference
smc_set_equiv	Set pulse equivalent	Section 3.3
smc_get_equiv	Read pulse equivalent	

2.1.4 Reverse backlash compensation

The function of reverse clearance compensation is provided by Leadshine controller, to lower the influence of the reverse clearance of mechanical rotation. The relevant functions:

Name	Function	Reference
smc_set_backlash_unit	Set reverse clearance value	Section 3.4
smc_get_backlash_unit	Read reverse clearance	

2.1.5 Axis IO mapping

The function of axis IO mapping configuration is supported by Leadshine controller, to configure the specific axis IO signal to any hardware input; for example, use the limit interface as the original point signal. This function can simplify wiring and wire change at site.

Relevant functions of axis IO mapping:

Name	Function	Reference
smc_set_axis_io_map	Set axis IO mapping relationship	Section 3.26
smc_get_axis_io_map	Read setting of axis IO mapping relationship	

Example 1: Set the original point interface of the 2nd axis as the positive limit signal of the 0th axis. (Example is c++ compiling environment)

```
int main(int argc, char* argv[])
{
/************************Variable definition*****************************/
    WORD CardNo = 0;                    //Connection No.
    WORD Axis = 0;                      //Designated axis No.: The 0th axis
    WORD IoType = 0;                    //Type of designated axis IO signal: Positive limit signal
    WORD MapIoType = 2;                 //Type of axis IO mapping: Original point signal
    WORD MapIoIndex = 2;                //Index number of axis IO mapping: The 2nd axis
    Double filter = 0;                  //Filter time
    short ret;
/*********************Function call and execution************************/
```

```
    //Step 1: Set mapping parameter
    ret = smc_set_axis_io_map(CardNo, Axis, IoType, MapIoType, MapIoIndex, filter);
    //Step 2: Read mapping parameter
    ret = smc_get_axis_io_map(CardNo, Axis, IoType, &MapIoType, &MapIoIndex, &filter);
        printf(" axis IO mapping type = %d\n", MapIoType);
        printf(" axis IO mapping index number = %d\n", MapIoIndex);
        printf(" filter time = %f\n",  filter);
}
```

2.1.6 Example of parameter setting

Example: Apply initialization before using controller (connect Leadshine controller via Ethernet port), set the pulse mode as Mode 0 andpulse equivalent as 10, enable PT motion and shut down the controller after motion completion.

```
int main(int argc, char* argv[])
{
/*************************Variable definition*****************************/
    short ret;                          //Return value of error code
    WORD ConnectNo = 0;                 //Connection number, ranged 0-7
    WORD type = 2;                      //Link type: 1 – Serial port; 2 – Ethernet port
    Char *pconnectstring = "192.168.5.11";    //Controller IP address
    DWORD baud = 0;
    WORD axis = 0;                      //Motion axis
    WORD outmode = 0;                   //Pulse output mode
    double equiv = 10;                  //Pulse equivalent
    double backlash = 10;               //Reverse clearance
    WORD posi_mode = 0;                 //0: Relative mode; 1: Absolute mode
    double read_pos;                    //Read value of command pulse counter

/*********************Function call and execution*************************/
    //Step 1: Initialize controller connection and connect the controller via Ethernet port
    ret = smc_board_init(ConnectNo, type, pconnectstring, baud);
    //Step 2: Set the initial parameter of controller
    ret = smc_set_pulse_outmode(ConnectNo, axis, outmode);      //Set pulse output mode
    ret = smc_set_equiv(ConnectNo, axis, equiv);  //Set pulse equivalent
    ret = smc_set_backlash_unit(ConnectNo, axis, backlash);       //Set reverse clearance value
    //Step 3: Shut down controller to release system resources
    ret = smc_board_close(ConnectNo);
}
```

2.2 Motion function

2.2.1 Point location (PT) motion

It mainly includes fixed-length motion, constant-speed motion, online variable-speed motion and online variable-position motion.

2.2.1.1 Parameter setting

While executing the control command of PT motion, Leadshine controller can make the motor

undergo PT motion based on T-shaped speed curve or S-shaped speed curve. Meanwhile, it can set the speed parameters such as starting speed, stop speed, acceleration and deceleration.

Relevant functions:

Name	Function	Reference
smc_set_profile_unit	Set speed curve of single-axis motion	Section 3.6
smc_get_profile_unit	Read speed curve of single-axis motion	
smc_set_s_profile	Set smoothing time parameter of Section S of single-axis speed curve	
smc_get_s_profile	Read smoothing time parameter of Section S of single-axis speed curve	

⚠Note: (1) The product between the set max. speed and the set pulse equivalent must be lower than 2MHz, for the max. pulse output frequency of motion controller is 2MHz.

(2) Smoothing time of single-axis speed curve S; it will be T-shaped curve, if the value is 0, or S-shaped smooth curve, if it is not 0; the S smoothing period is ranged 0-1S.

The speed curve can be divided into symmetric and asymmetric depending on whether the acceleration and deceleration section are the same or not; it can also be divided into T-shaped and S-shaped by shape, as shown in Fig. 2.1 and Fig. 2.2.

“Starting speed”: Set the initial speed of single-axis motion; unit: unit/s.

“Running speed”: Set the max. running speed of single-axis motion; unit: unit/s.

“Stop speed”: Set the stop speed of single-axis motion; unit: unit/s.

“Acceleration”: Set the period consumed from initial speed of single-axis motion to the highest running speed (Tacc)

“Deceleration”: Set the period consumed from the max. running speed of single-axis motion to the stop speed (Tdec)

“S-section time”: Set the time parameter of S Section of single-axis speed curve. See spara in diagram below:

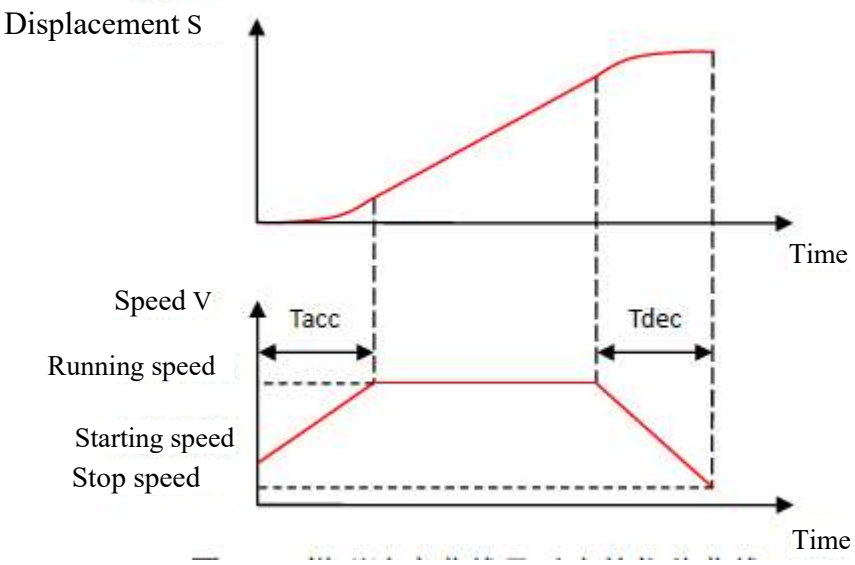


Fig. 21 Trapezoidal speed curve and corresponding displacement curve time

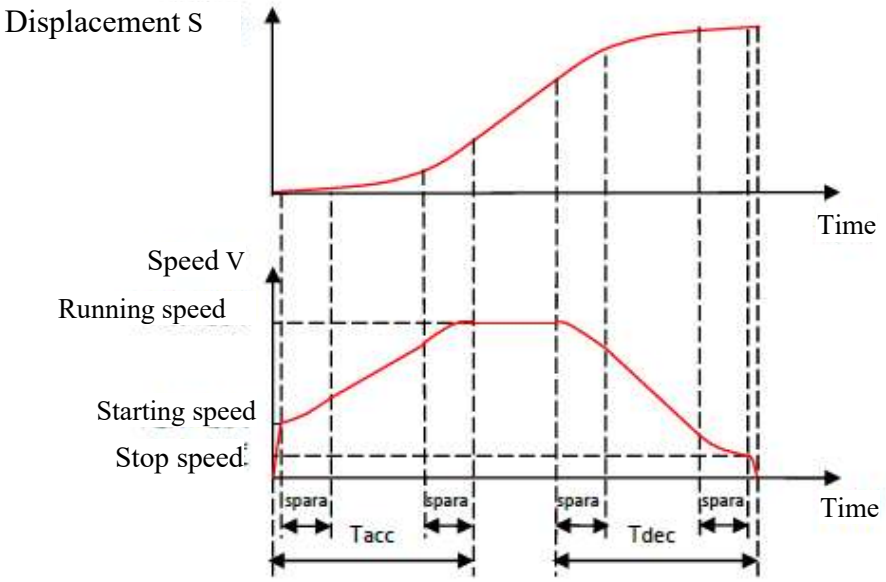


Fig. 2.2 S-shaped speed curve and corresponding displacement curve

2.2.1.2 Fixed-length motion

Fixed-length motion: Controlled by the motion controller, the motion platform moves ata preset speed from the current position and stops accurately at the preset position.

In fixed-length motion mode, all axises support independent setting of motion parameters, such as target position, target speed, acceleration, deceleration, starting speed, stop speed and S-section parameter, as well as independent motion or stop.

Relevant functions:

Name	Function	Reference
smc_pmove_unit	Point location (PT) motion	Section 3.6

2.2.1.3 Constant-speed motion

In constant-speed motion mode, the motor, which is controlled by controller, can accelerate to the max. speed from the starting speed through the trapezoid or S-shaped speed curve, and then keep moving along at such a speed; it will not decelerate or stop at the starting speed curve, until it receives stop command or limit signal.

Relevant functions:

Name	Function	Reference
smc_vmove	Constant-speed motion of designated axis	Section 3.6
smc_stop	Stop of designated axis	

⚠Note: Constant-speed motion needs to be stopped manually; otherwise, it will continue the motion.

2.2.1.4 Online variable speed and position motion

Both speed and position may change in real time during PT motion.

Relevant functions:

Name	Function	Reference
smc_reset_target_position_unit	Online change of target position of designated axis	Section 3.6
smc_update_target_position_unit	Forced change of target position of designated axis	
smc_change_speed_unit	Online change of motion speed of designated axis	

⚠Note: 1)Online variable motion applies to PT motion; online variable-speed motion applies to PT and constant-speed motion.

2) The target position after online position change is the absolute coordinate position, no matter whether the current motion mode is absolute coordinate or relative coordinate mode.

3) Online variable speed supports speed change setting; the set variable speed time is changed from the current speed to a new one. Now, the controller will recalculate the period from starting speed to top speed, as well as the period from the top speed to stop speed. In other words, the acceleration/deceleration will be recalculated. Once variable speed is established, the default running speed of this axis will be rewritten as New_Vel, and both acceleration/deceleration will be covered by the calculated value of controller.

4) Online position change will be executed only if it is under motion status; while forced position change can be executed whether the current axis is under the motion status.

Example 1: Apply PT motion for certain section, motion distance is 10,000units, speed curve is S-shaped speed curve, starting speed is 0, top speed is 1,000, acceleration is 0.1S, deceleration is

0.2S; the speed will change to 2,000 after running for a certain period and then become 0 after a certain period.

```
int main(int argc, char* argv[])
{
/************************Variable definition*****************************/
    WORD ConnectNo = 0;          //Connection number, ranged 0-7
    WORD ret = 0;                //Return error code
    WORD axis = 0;               //Motion axis
    double Max_Vel = 1000;       //Max. running speed
    double Tacc = 0.1;           //Acceleration
    double Tdec = 0.2;           //Deceleration
    double Min_Vel = 0;          //Starting speed
    double Stop_Vel = 0;         //Stop speed
    double s_para = 0.1;         //S-shaped smoothing factor
    double Dist = 10000;         //Motion distance
    WORD posi_mode = 0;          //0: Relative mode; 1: Absolute mode
    double New_Vel = 2000;       //Speed after online speed change
    double Taccdec = 0.1;        //Acceleration after online speed change

/*********************Function call and execution**************************/
    //Step 1: Set speed curve of single-axis motion
    ret = smc_set_profile_unit(ConnectNo, axis, Min_Vel, Max_Vel, Tacc, Tdec, Stop_Vel);
    //Step 2: Set parameter of smoothing Section S of single-axis speed curve
    ret = smc_set_s_profile(ConnectNo, axis, 0, s_para);
    //Step 3: Start fixed-length motion
    ret = smc_pmove_unit(ConnectNo, axis, Dist, posi_mode);
    //Step 4: Start online speed change
    Sleep(500);          //Delay a certain period
    ret = smc_change_speed_unit(ConnectNo, axis, New_Vel, Taccdec);
    //Step 5: Start online speed change and change target position to 0
    Sleep(500);          //Delay a certain period
    ret = smc_reset_target_position_unit(ConnectNo, axis, 0);
    }
```

Example 2: Apply reverse constant-speed motion for a certain period and the speed will become 2000, the speed change will become positive value, motion direction is positive and it will stop motion after a certain period.

```
int main(int argc, char* argv[])
{
/************************Variable definition*****************************/
    WORD ConnectNo = 0;          //Connection number, ranged 0-7
    WORD ret = 0;                //Return error code
    WORD axis = 0;               //Motion axis
    double Min_Vel = 0;          //Starting speed
    double Max_Vel = 1000;       //Max. running speed
    double Tacc = 0.1;           //Acceleration
    double Tdec = 0.2;           //Deceleration
    double Stop_Vel = 0;         //Stop speed
```

```
    double s_para = 0;              //S-shaped smoothing factor
    double read_pos;                //Read value of command pulse counter
    WORD dir = 0;                   //Reverse motion
    double NewVel = 2000;           //Value after speed change

/**********************Function call and execution**************************/
    //Step 1: Set speed curve of single-axis motion
    ret = smc_set_profile_unit(ConnectNo, axis, Min_Vel, Max_Vel, Tacc, Tdec, Stop_Vel);
    //Step 2: Set parameter of smoothing Section S of single-axis speed curve
    ret = smc_set_s_profile(ConnectNo, axis, 0, s_para);
    //Step 3: Start constant-speed motion
    ret = smc_vmove(ConnectNo, axis, dir);
    //Step 4: Start online variable-speed motion
    Sleep(500);         //Delay a certain period
    ret = smc_change_speed_unit(ConnectNo, axis, New_Vel, 0.1);
    //Step 5: Stop motion immediately
    Sleep(500);         //Delay a certain period
    ret = smc_emg_stop(ConnectNo);
    }
```

2.2.2 Homing motion

The original point of motion coordinate system needs setting before precise motion control. The motion platform is provided with original point sensor (also called original point switch) to use the input source from the original point signal.

Leadshine controller provides 10 homing modes:

Method: One-step homing

Realize homing at a preset speed in this method; it applies to scenarios with short stroke and high safety. Motion process: The motor moves from the initial position to the original point at constant speed; when it reaches the original point switch, the original point signal will be triggered and motor will stop immediately (process 0); the stop position will be set as the original point position, as shown in Fig. 2.9.

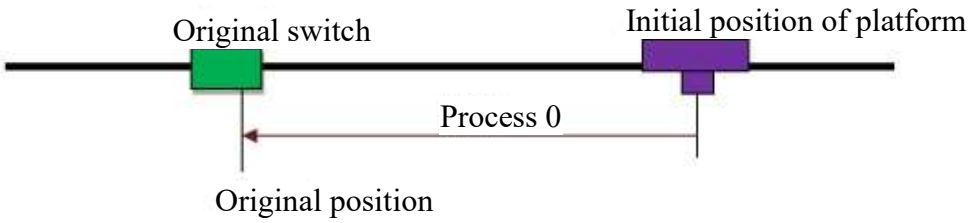


Fig. 2.9 Schematic diagram of one-step homing

Method 1: One-step homing and return

In this method, motion will be executed at mode 1 first and then homing will be enabled at reverse direction to reach the edge position of original point switch; when the original point signal is invalid

for the first time, the motor will stop immediately; set the stop position as the original point position, as shown in Fig. 2.10.

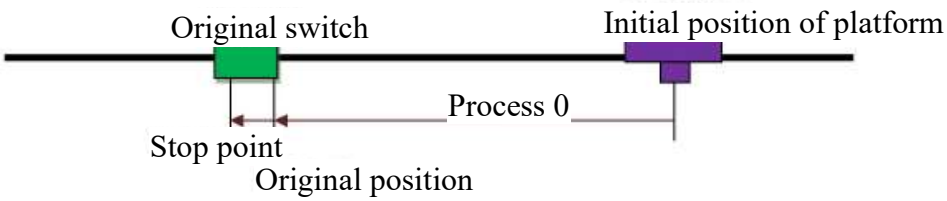


Fig. 2.10 Schematic diagram of one-step homing and return

Method 2: Two-step homing

As shown in Fig. 2.11, this method is a combination of Method 0 and 1. It will execute homing and search in Method 1 firstly, and then perform one-step homing in Method 0. Refer to the introduction of Method 0 and 1.

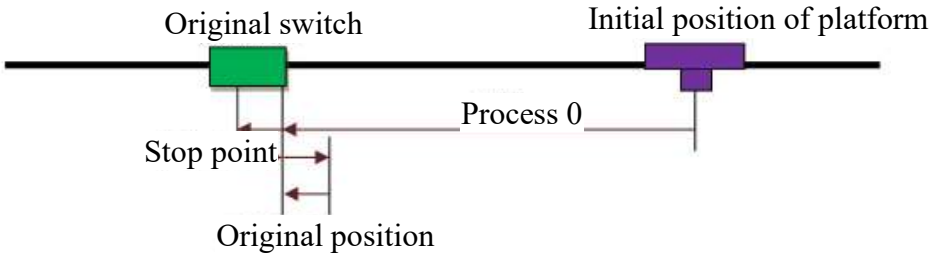


Fig. 2.11 Schematic diagram of two-step homing

Method 3: One-step homing and search of EZ signal

When homing signal is found during this homing process, the motor will not stop until the EZ signal of this axis occurs. See Fig. 2.12 for the homing process.

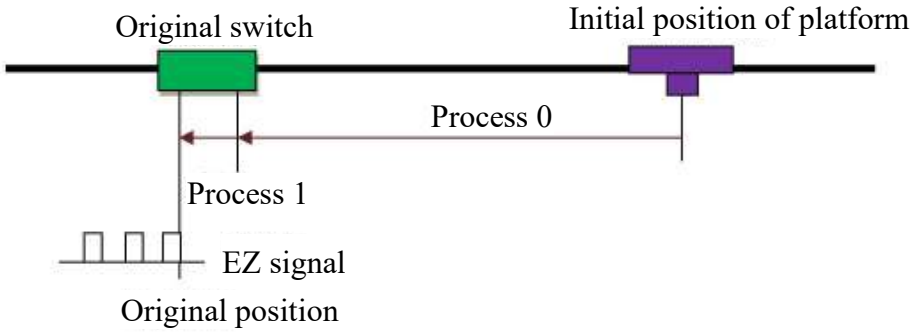


Fig. 2.12 Schematic diagram of one-step homing and search of one EZ signal

Method 1: Record 1 EZ signal during homing

When EZ signal of this axis is detected during homing process in this method, the motor will stop. See Fig. 2.13 for the homing process.

Figure2.13 Schematic diagram of recording 1 EZ signal during ho过程 1

Method 5: One-step homing and search of EZ signal

When original point signal is searched during this homing process, it will decelerate and stop, and then find EZ reversely at the search speed and then motor will stop. See Fig. 2.14 for the homing process.

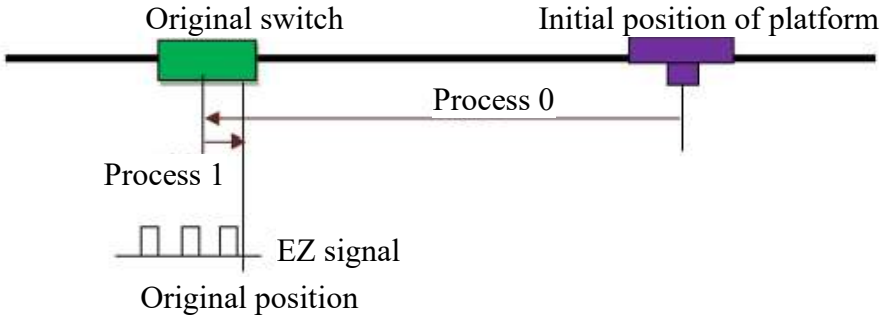


Fig. 2.14 One-step homing and search one EZ

Method 6: Original point latch

As shown in Fig. 2.5, the motor has homing at the set speed firstly; when edge of original point switch is triggered, the current position will be latched and motor will decelerate and stop. When motor is decelerated and stopped, it will have reverse search of latch position, then move to the latch position and motor will stop.

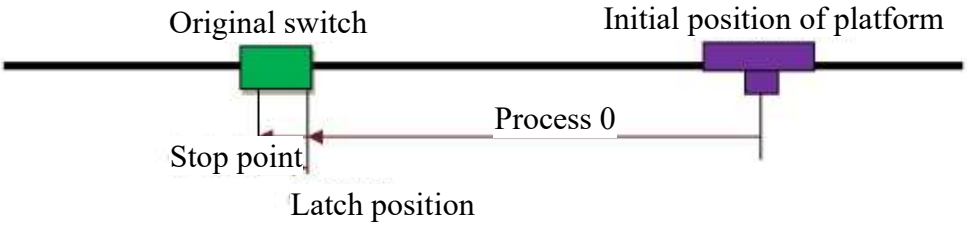


Fig. 2.15 Schematic diagram of homing with original point latch

Method 7: Original point latch and homodromous EZ latch

In this mode, it will firstly execute the original point latch and homing according to Mode 6, and then move along with the set homing direction until EZ signal is generated; when EZ signal is generated, latch current position and execute deceleration and stop; when motor is decelerated and stopped, apply reverse search of EZ latch position, move the latch position and motor will stop. See Fig. 2.16 for the homing process.

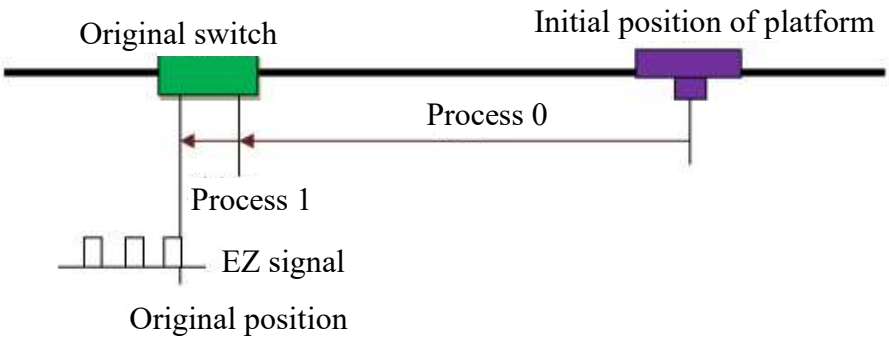


Fig. 2.16 Schematic diagram of original point latch and homodromous latch homing

Method 8: Record one EZ latch

When EZ effective edge is detected during home, latch the current position, execute deceleration and stop; when motor is decelerated and stopped, have reverse search of EZ latch position, move to the latch position and the motor will stop. See Fig. 2.17 for the homing process.

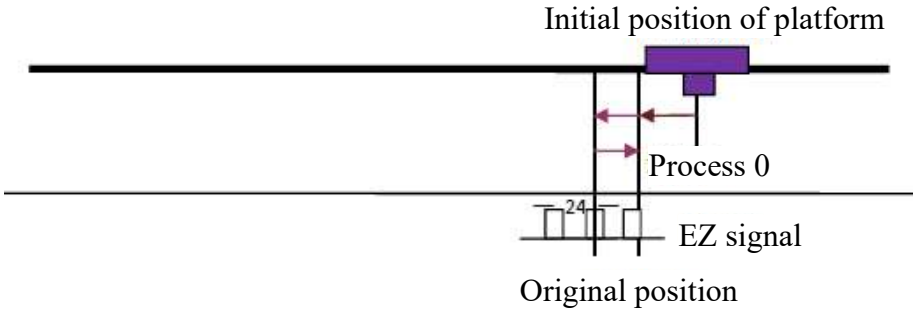


Fig. 2.17 Schematic diagram of recording one EZ latch homing

Method 9: Original point latch and reverse EZ latch

In this mode, it will firstly execute original point latch and homing according to Mode 6, then move to the reverse direction of homing direction until EZ signal is generated; when EZ signal is generated, latch the current position, decelerate and stop; when motor is decelerated and stopped, apply reverse research of EZ latch position, move to latch position and motor is stopped. See Fig. 2.8 for the homing process.

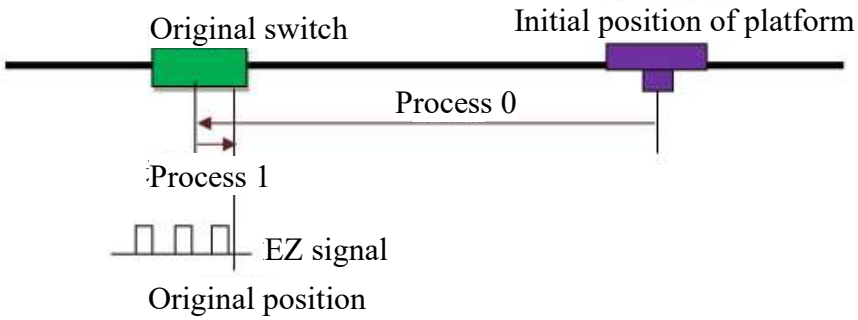


Fig. 2.18 Original point latch and reverse EZ latch for homing

Relevant functions of homing motion:

Name	Function	Reference

smc_set_home_pin_logic	Set effective level of original point signal	Section 3.7
smc_get_home_pin_logic	Read setting of original point setting	
smc_set_homemode	Set homing mode	
smc_get_homemode	Read homing mode	
smc_set_ez_count	Set homing EZ number	
smc_get_ez_count	Read homing EZ number	
smc_set_home_position_unit	Set deviation position value after homing	
smc_get_home_position_unit	Read deviation position value after homing	
smc_set_home_profile_unit	Set homing speed parameter	
smc_get_home_profile_unit	Read homing speed parameter	
smc_set_el_home	Original point switch function at limit position	
smc_home_move	Start homing at designated direction and speed	
smc_get_home_result	Read homing status	

Example: Homing of Axis 0

```
int main(int argc, char* argv[])
{
/************************Variable definition*****************************/
    short ret = 0;                  //Error return
    short ConnectNo = 0;            //Link number,  range: 0~7
    WORD axis = 0;                  //Motion axis number,  range: 0~Max. axis number -1
    double Start_Vel = 1000;        //Initial speed of homing,  range: 0~2MHz frequency
    double Max_Vel = 1000;          //Homing speed,  range: 0~2MHz frequency
    double Tacc = 0.1;              //Acceleration,  unit s: Range: 0.001~10s
    WORD org_logic = 0;             //Set effective level of original point: 0 – Low level;  1 – High
    level
    double filter = 0;              //Filter time is 0,  reserved parameter,  meaningless
    WORD home_dir = 1;              //Set homing direction: 0 – Negative direction;  1 – Forward
    direction
    WORD mode = 0;                  //Set homing mode as one-step homing
    WORD Source = 0;                //Set counting source as pulse counting
    WORD enable = 0;                //Set counting enabling after homing
    double position = 100;          //Set counting value after homing

/*********************Function call and execution**************************/
    //Step 1: Set level parameter of homing
    ret = smc_set_home_pin_logic(ConnectNo, axis, org_logic, filter);
    //Step 2: Set homing mode
    ret = smc_set_homemode(ConnectNo, axis, home_dir, 1, mode, Source);
    //Step 3: Set counting position value after homing
    ret = smc_set_home_position_unit(ConnectNo, axis, enable, position);
    //Step 4: Set speed parameter of homing
    ret = smc_set_home_profile_unit(ConnectNo, axis, Start_Vel, Max_Vel, Tacc, 0); //  Step 5:
```

```
    Start homing
    ret = smc_home_move(ConnectNo, axis);
}
```

2.2.3 PVT motion

Leadshine controller provides two PVT modes to set the single-axis speed freely, i.e. PTT motion mode and PTS motion mode; in which, PTT motion mode applied to setting of single-axis trapezoidal speed, while PTS motion mode applied to setting of single-axis S-shaped speed.

2.2.3.1 Setting of single-axis speed

(1) PTT motion mode

With high flexibility, PTT mode can be used for setting the single-axis speed freely. The user may enter the position and time parameter directly to describe the law of motion.

Relevant functions:

Name	Function	Reference
smc_ptt_table_unit	Send data to the designated data table by means of PTT description	Section 3.8
smc_pvt_move	Start PVT motion	

Example: Motion at PTT mode

Set the speed curve as shown in Fig. 2.4 and this speed curve can be set easily through PTT mode. Firstly, calculate the displacement of all sections, i.e. the area of speed curve and time axis: P1 = 1500(unit), P2 = 4000(unit) , P3 = 8500(unit) , P4 = 24000(unit) , P5 = 27000(unit) , P6 = 3000(unit).

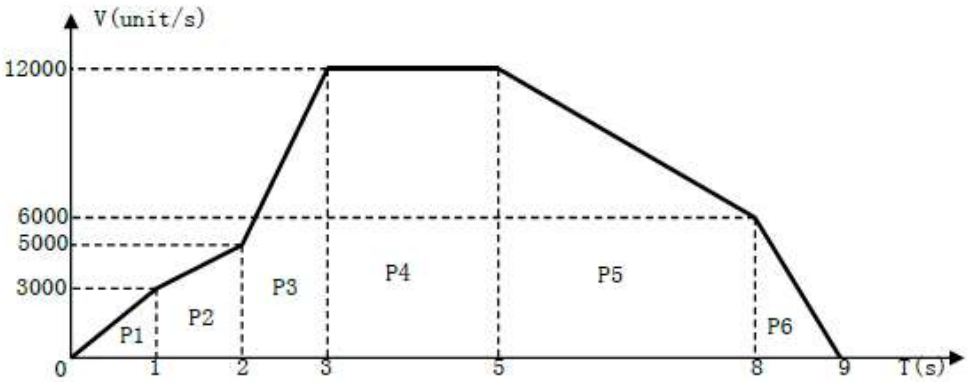


Fig. .24 Setting of V-T curve at PTT mode

Accumulate the displacements of all sections to acquire the position and time of all points at PTT mode, as shown in Fig. 2.5.

Table 2.5 Array data at PTT mode

S/N	Position P (unit)	Time T(s)
0	0	0
1	1500	1
2	5500	2
3	14000	3
4	38000	5
5	65000	8
6	68000	9

Do programming as follows:

```
int main(int argc, char* argv[])
{
/************************Variable definition*****************************/
    WORD MyCardNo = 0;          //Connection number,  ranged 0--7
    WORD ret= 0;                //Return error code
    WORD My_AxisList= 0;        //Motion axis
    WORD MyCount = 7;           //Motion point
    double MyPTime[7];          //Time parameter of motion point
    double MyPPos[7];           //Position parameter of motion point
    MyPTime[0] = 0;
    MyPPos[0] = 0;
    MyPTime[1] = 1;
    MyPPos[1] = 1500;
    MyPTime[2] = 2;
    MyPPos[2] = 5500;
    MyPTime[3] = 3;
    MyPPos[3] = 14000;
    MyPTime[4] = 5;
    MyPPos[4] = 38000;
    MyPTime[5] = 8;
    MyPPos[5] = 65000;
    MyPTime[6] = 9;
    MyPPos[6] = 68000;
    WORD MyAxisNum = 1;         //Number of axises engaged in PVT motion is 1

/*********************Function call and execution************************/
    //Step 1: Set PTT motion parameter
    ret = smc_ptt_table_unit(MyCardNo, My_AxisList, MyCount, MyPTime, MyPPos);
    //Step 2: Start PTT motion
    ret = smc_pvt_move(MyCardNo, MyAxisNum, &My_AxisList);
}
```

PTT motion result (position – time curve, acceleration – time curve) is as shown in Fig. 2.6 and 2.7.

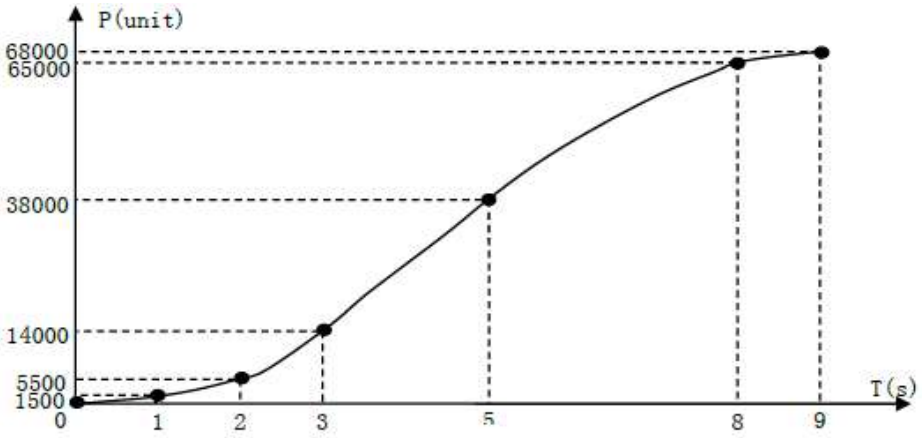


Fig. 2.6 Displacement curve of PTT motion

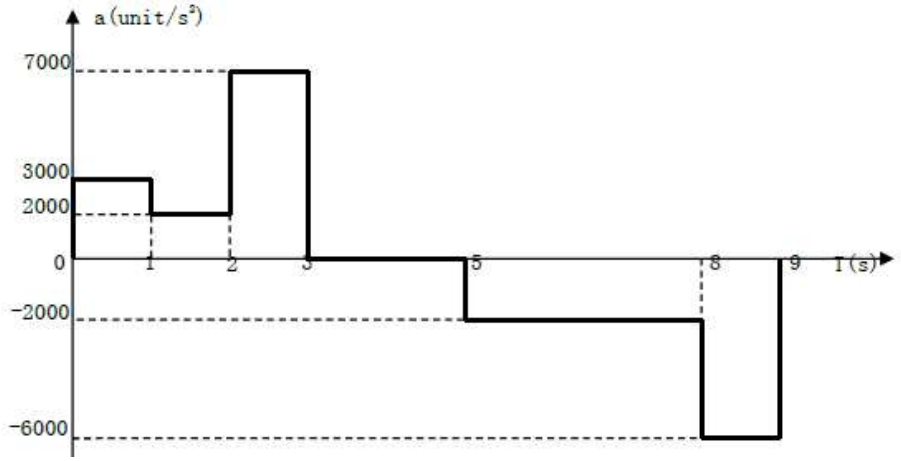


Fig. 2.7 Acceleration curve of PTT motion

(2) PTS motion mode

As the extended function mode of PTT, PTS motion mode can ensure smooth speed transition of all data points. User can enter the position, time and parameter to describe the law of motion. The percentage parameter of data point: The percentage of acceleration change between two adjacent 2 data points in the speed change time.

Relevant functions:

Name	Function	Reference
smc_pts_table_unit	Send data to the designated table by means of PTS description	Section 3.8
smc_pvt_move	Start PVT motion	

Example: Motion at PTS mode

According to Fig. 2.9, the acceleration curve in example has sudden change; so, there may have impact during the motion. Motion at PTS mode can be adopted in order to have smoother speed curve than that of PTT mode.

See Fig. 2.8 for the speed percentage parameter of all points; the corresponding acceleration – time

curve is shown in Fig. 2.9, and the acceleration/deceleration of each section is the percentage between time and speed of that section; the max. acceleration of one section can be different from the acceleration at PTT mode, for it is provided with smoothing based on the internal algorithm. See Fig. 2.10 and 2.11 for the corresponding position – time curve and speed – time curve.

S/N	P(unit)	T(s)	Percent(%)
0	0	0	0
1	1500	1	20
2	5500	2	40
3	14000	3	60
4	38000	5	0
5	65000	8	20
6	68000	9	80

Fig. 2.8 Acceleration curve at PTS mode

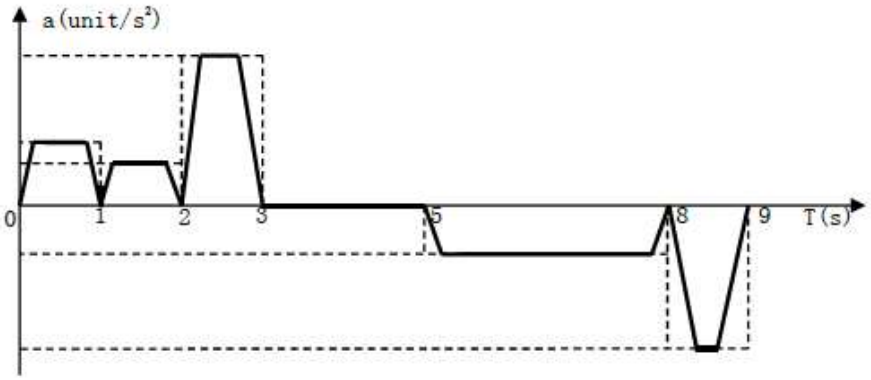


Fig. 2.9 Acceleration curve at PTS mode

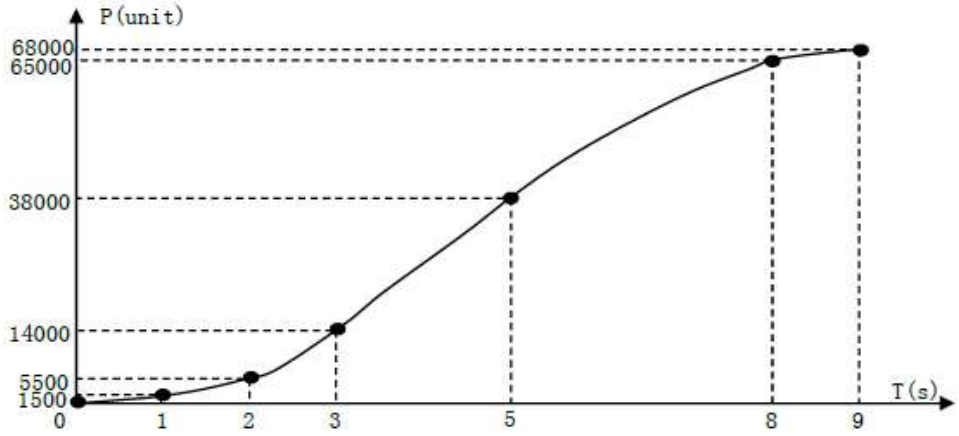


Fig. 2.10 Displacement curve acquired at PTS mode

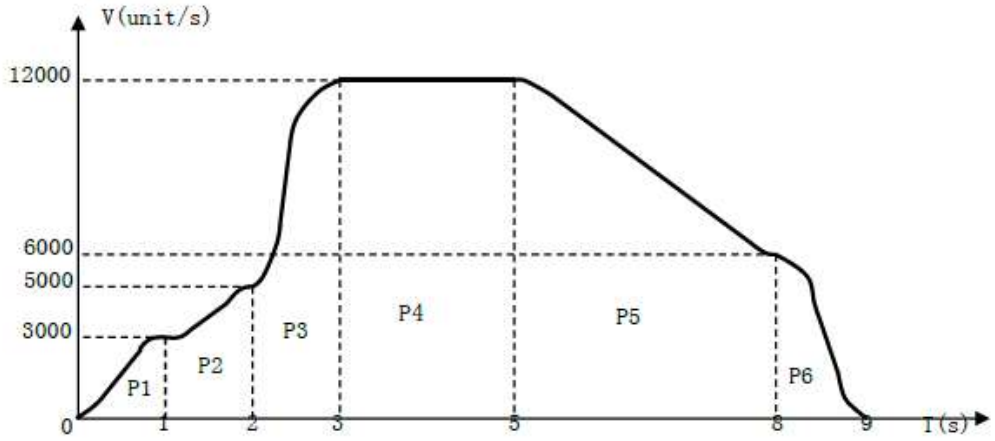


Fig. 2.11 Speed curve acquired from PTS motion

Do programming as follows:

```
int main(int argc, char* argv[])
{
/************************Variable definition*****************************/
    short ret = 0;                  //Error return
    short MyCardNo = 0;             //Controller connection number
    WORD My_AxisList = 0;           //Motion axis
    WORD MyCount = 7;               //Motion point
    double MyPTime[6];              //Time parameter of motion point
    double MyPPos[6];               //Position parameter of motion point
    double MyPPer[6];               //Percentage of motion acceleration
    MyPTime[0] = 0;
    MyPPos[0] = 0;
    MyPPer[0] = 0;
    MyPTime[1] = 1;
    MyPPos[1] = 1500;
    MyPPer[1] = 20;
    MyPTime[2] = 2;
    MyPPos[2] = 5500;
    MyPPer[2] = 0;
    MyPTime[3] = 3;
    MyPPos[3] = 14000;
    MyPPer[3] = 60;
    MyPTime[4] = 5;
    MyPPos[4] = 38000;
    MyPPer[4] = 0;
    MyPTime[5] = 8;
    MyPPos[5] = 65000;
    MyPPer[5] = 20;
    MyPTime[6] = 9;
    MyPPos[6] = 68000;
    MyPPer[6] = 80;
    WORD MyAxisNum = 1;             //Number of axises engaged in PVT motion is 1

/*********************Function call and execution**************************/
```

```
    //Step 1: Set PTS motion parameter
    ret = smc_pts_table_unit(MyCardNo, My_AxisList, MyCount, MyPTime, MyPPos, MyPPer);
    //Step 2: Start PTS motion
    ret = smc_pvt_move(MyCardNo, MyAxisNum, &My_AxisList);
}
```

2.2.3.2 Function of setting multi-axis speed

Leadshine controller supports planning of PVT advanced motion curve; the user may try to set the motion track through PVT if some special motion tracks fail to be satisfied by the single-axis motion and interpolated motion.

Leadshine controller provides two PVT modes to realize multi-axis track planning, i.e. PVT and PVTS motion mode. PVT mode applies to the setting of track which has requirements for the position, time and speed of all points, while PVTS mode applies to the setting of track which has requirements for position and time of all points, but no requirements for speed.

(1) PVT motion mode

The position, speed and time parameter of a series data points are used in PVT mode to describe the law of motion.

Relevant functions:

Name	Function	Reference
smc_pvt_table_unit	Send data to the designated data sheet by means of PVT description	Section 3.8
smc_pvt_move	Start PVT motion	

⚠Note: The ideal track curve can be hardly realized if the P, V and T data of all points are set unreasonably.

The actual track can be closer to the ideal track if more points are selected on the ideal track.

Example: Motion at PVT mode

As shown in Fig. 2.12, the user may design the track by using PVT function to move the motion platform along with the elliptical orbit, for the elliptic interpolation function is not provided in the Leadshine controller.

Set the major semi-axis of ellipse as 9000unit long and minor semi-axis as 7000unit long; the angular speed ω of elliptical orbit is constant, and total period of track motion is 10s.

Obviously, the equation of this ellipse is:

$$\begin{cases} x = 9000\cos(\theta) + 9000 \\ y = 7000\sin(\theta) \end{cases}$$

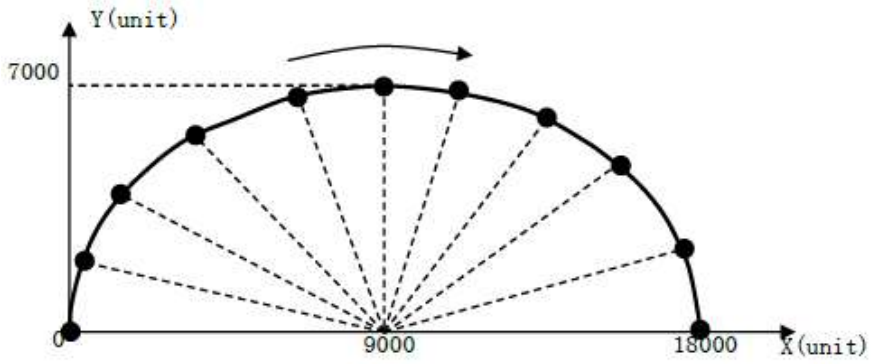


Fig. 2.12 Track and segmentation of first half ellipse

The steps for acquiring first half elliptical orbit at PVT mode:

1. Divide this track into ten sections with equal arc angles; calculate the coordinates of each point, i.e. Value P. The procedure is as follows:

```
short ret;
WORD MyCount = 7;              //Motion point
double MyPPosX[11];            //Define array to store the position data of PVT (Axis X)
double MyPPosY[11];            //Define array to store the position data of PVT (Axis Y)
WORD a, b, i;
      a = 9000;                //Define major semi axis of ellipse
      b = 7000;                //Length of minor semi axis
const double PI = 3.14159265358979323846;
    for(i = 0; i<11; i++)      //Calculate the X and Y coordinate of each point
      {

          MyPPosX[i] = a * cos((10-i) * PI/10)+ a;
          MyPPosY[i] = b * sin((10-i) * PI/10);
      }
      MyPPosX[0] = 0;      //Position of the 1st point is 0
      MyPPosY[0] = 0;
```

2. Calculate the corresponding speed (value V) and time (value T) of each point according to the coordinates (i.e. Value P) of each point. Apply derivation of elliptic equation formula to acquire the speed component at direction of X and Axis Y:

$$\begin{cases} \dot{x} = -9000\sin(\theta)\dfrac{d\theta}{dt} \\ \dot{y} = 7000\cos(\theta)\dfrac{d\theta}{dt} \end{cases}$$

The angular speed ω is listed in equation above $\frac{d\theta}{dt}$.

The speed component at X and Axis Y direction of each point can be calculated through the following procedure:

```
double MyPVelX[11];      //Define array to store the speed data of PVT (Axis X)
double MyPTimeX[11];     //Define array to store the time data of PVT (Axis X)
double MyPVelY[11];      //Define array to store the speed data of PVT (Axis Y)
double MyPTimeY[11];     //Define array to store the time data of PVT (Axis Y)
double MyWVel;           //Define angular speed
  for(i = 0; i<11; i++)
    {
        MyPTimeX[i] = i;              //Store the time data of Axis X
        MyPTimeY[i] = i;              //Store the time data of Axis Y
    }
MyWVel = -PI/10;                      //Calculate angular speed
MyPVelX[0] = MyPVelX[10] = 0;         //Set the Axis X of starting and end point as 0
MyPVelY[0] = MyPVelY[10] = 0;         //Set the Axis Y speed of starting and end point as 0
for(i = 0; i<9; i++)
{

    MyPVelX[i+1] = -a * sin((10-i-1) * PI/10) * MyWVel;
                    //Calculate the Axis X speed of other points
    MyPVelY[i+1] = b * cos((10-i-1) * PI/10) * MyWVel;
                    //Calculate the Axis Y speed of other points
}
```

Calculate the P, V and T data of X and Axis Y of all points, as shown in Table 2.13.

Fig. 2.13 PVT data

	Axis X			Axis Y		
S/N	P(unit)	V(unit/s)	T(s)	P(unit)	V(unit/s)	T(s)
0	0	0	0	0	0	0
1	440	873.731	1	2163	2091.479	1
2	1719	1661.927	2	4115	1779.117	2
3	3710	2287.443	3	5663	1292.603	3
4	6219	2689.048	4	6657	679.560	4
5	9000	2827.431	5	7000	-0.003	5
6	11781	2689.046	6	6657	-679.566	6
7	14290	2287.438	7	5663	-1292.608	7
8	16281	1661.921	8	4114	-1779.120	8
9	17560	873.724	9	2163	-2091.481	9
10	18000	0	10	0	0	10

3. Transfer groups of data to data table by using the function of smc_pvt_table_unit.

The procedure is as follows:

```
WORD MyCardNo;
WORD My_AxisList[2];          //Define the variable in list of PVT motion axis
intMyCountX;                  //Define number variable of PVT data point of Axis X
intMyCountY;                  //Define number variable of PVT data point of Axis Y
MyCardNo = 0;                 //Connection No.
My_AxisList[0] = 0;           //X and Axis Y) engaged in PVT motion
My_AxisList[1] = 1;
MyCountX = 11;
MyCountY = 11;
ret = smc_pvt_table_unit(MyCardNo, 0, MyCountX, MyPTimeX, MyPPosX, MyPVelX);
//Transfer PVT data to Axis X by means of PVT description

ret = smc_pvt_table_unit(MyCardNo, 1, MyCountY, MyPTimeY, MyPPosY, MyPVelY);
//Transfer PVT data to Axis Y by means of PVT description
```

4. Execute PVT motion by using the function of smc_pvt_move

The procedure is as follows:

```
WORD My_AxisNum = 2;                  //Number of axises engaged in PVT motion is 2
ret = smc_pvt_move(MyCardNo, My_AxisNum, My_AxisList);    //Start two-axis PVT motion
```

Once the procedure above is executed, acquire the PVT motion track as shown in Fig. 2.14.

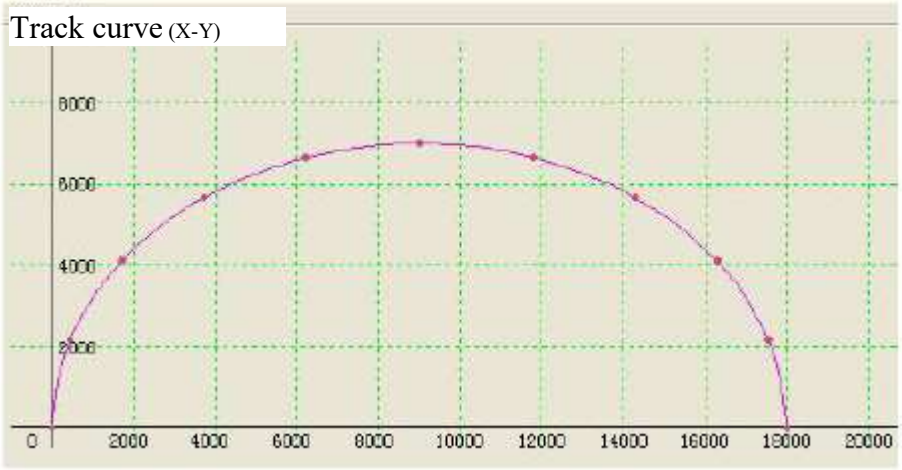


Fig. 2.14 The first half elliptical orbit acquired in PVT mode

(2) PVTS motion mode

Only position and time of data point on ideal track, as well as the starting and end speed, need to be defined in PVTS motion mode. The motion controller will calculate the track position and progress between all points based on the position and time parameter of data point, to ensure the continuity of speed and acceleration of all tracks. Relevant functions:

Name	Function	Reference
smc_pvts_table_unit	Transfer data to the designated data sheet by means of PVTS description	Section 3.8

smc_pvt_move	Start PVT motion	

Example: Motion at PVTS mode

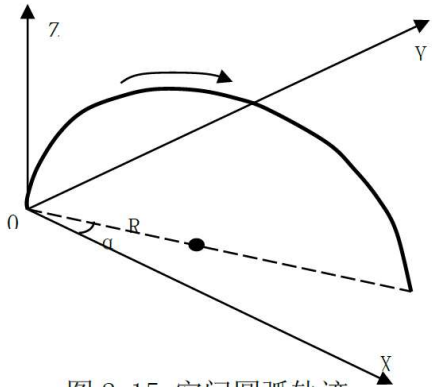


Fig. 2.15 Arc track of space

Design the arc track of space as shown in Fig .2.15. Set radius R = 15000unit, the inclined angle between track on XY plane and Axis X is a = n/6 and total period of track motion is 10s.

Obviously, the equation of this space arc is:

$$\begin{cases} x = 15000\cos(\frac{\pi}{6})\cos(\theta) + 15000\cos(\frac{\pi}{6}) \\ y = 15000\sin(\frac{\pi}{6})\cos(\theta) + 15000\sin(\frac{\pi}{6}) \\ z = 15000\sin(\theta) \end{cases}$$

If there are no accurate requirements for the speed of points on track, the track in PVTS mode should be designed as follows:

1. Divide this track into 10 sections with the same angle, calculate the position coordinate of each point, i.e. Value P.

2. Calculate the Value T of each point based on Value P of them. Set the same motion time of each track.

3. Set starting speed and end speed as 0.

4. Transfer groups of data to the data sheet by using the function of smc_pvts_table_unit.

5. Execute PVT motion by using the function of smc_pvt_move.

Do programming as follows:

```
int main(int argc, char* argv [])
{
/************************Variable definition*****************************/
    short ret;                          //Return error code
    double MyPTimeX[11], MyPVelBeginX, MyPVelEndX;     //Define variable (Axis X)
    double MyPTimeY[11], MyPVelBeginY, MyPVelEndY;     //Define variable (Axis Y)
    double MyPTimeZ[11], MyPVelBeginZ, MyPVelEndZ; ;    //Define variable (Axis Z)
    WORD MyCountX, MyCountY, MyCountZ, MyCardNo, My_AxisNum;
    double MyPPosX[11], MyPPosY[11], MyPPosZ[11];   //Define groups
    WORD My_AxisList[3];       //Define list of motion axis
```

```
    WORD R = 15000;             //Define circle radius
    WORD i;
    const double pi = 3.14159265358979323846;             //Define circumference
    MyCountX =
    MyCountY =
    MyCountZ = 11;                                        //Set data point of X,  Y and Axis Z
    MyPVelBeginX = MyPVelEndX = 0;                        //Set starting/end speed of Axis X as 0
    MyPVelBeginY = MyPVelEndY = 0;                        //Set starting/end speed of Axis Y as 0
    MyPVelBeginZ = MyPVelEndZ = 0;                        //Set starting/end speed of Axis Z as 0
    MyCardNo = 0;                                         //Connection No.
    My_AxisList[0] = 0;               //No. 0,  1 and 2 axis (i.e. X,  Y and Axis Z) are engaged in
    PVT motion
    My_AxisList[1] = 1;
    My_AxisList[2] = 2;
    My_AxisNum = 3;                                       //3 axises
     are engaged in PVT motion

/**********************Function call and execution*************************/
//Step 1: Calculate the position coordinates of Axis X,  Y and Z
for(i = 0; i<11; i++)
{
   MyPPosX[i] = R * cos(pi/6) * cos((10-i) * pi/10)+R * cos(pi/6);
   //Calculate the position coordinates of points on X axis
   MyPPosY[i] = R * sin(pi/6) * cos((10-i) * pi/10)+R * sin(pi/6);
   //Calculate the position coordinates of points on Y axis
   MyPPosZ[i] = R * sin((10-i) * pi/10);       //Calculate the position coordinates of points on Z axis
   }
   MyPPosX[0] = MyPPosY[0] = MyPPosZ[0] = 0;     //Position of the first point is 0

   //Step 2: Calculate the time of points on Axis X,  Y and Z
for(i = 0; i<11; i++)
   {
   MyPTimeX[i] = i;                    //Calculate the time of points on Axis X
   MyPTimeY[i] = i;                    // Calculate the time of points on Axis Y
   MyPTimeZ[i] = i;                    Calculate the time of points on Axis Z
   }

//Step 3: Transfer the PVTS data to Axis X,  Y and Z
   ret = smc_pvts_table_unit(MyCardNo, 0, MyCountX, MyPTimeX, MyPPosX, MyPVelBeginX,
   MyPVelEndX);                  //Transfer PVTS data to Axis X
   ret = smc_pvts_table_unit(MyCardNo, 1, MyCountY, MyPTimeY, MyPPosY, MyPVelBeginY,
   MyPVelEndY);                  //Transfer PVTS data to Axis Y
   ret = smc_pvts_table_unit(MyCardNo, 2, MyCountZ, MyPTimeZ, MyPPosZ, MyPVelBeginZ,
   MyPVelEndZ);                  //Transfer PVTS data to Axis Z

//Step 4: Start PVTS motion of Axis X,  Y and Z
      ret = smc_pvt_move(MyCardNo, My_AxisNum, My_AxisList);
}
```

The space arc track acquired in PVTS mode is as shown in Fig. 2.16.

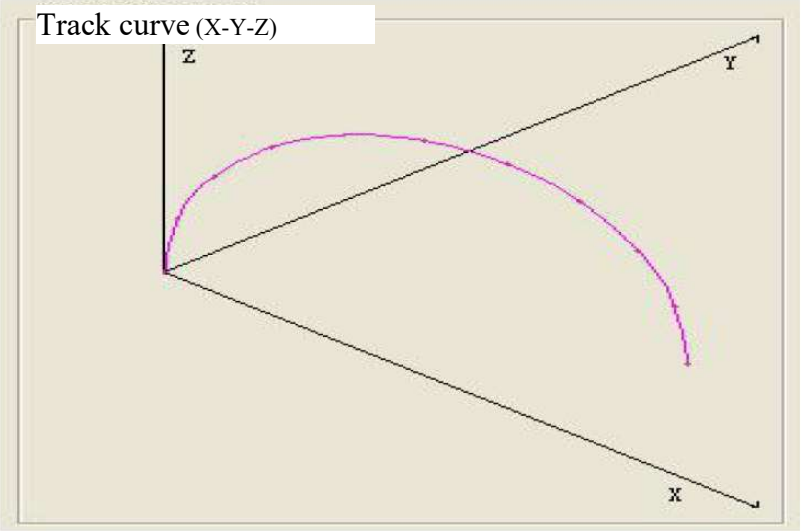


Fig. 2.16 Space arc track acquired by motion in PVTS mode

2.2.4 Interpolation motion

The multi-axis coordinated motion can be realized in interpolated motion mode, in order to realize certain motion track. The interpolated motion mode can be further divided into multi-axis interpolation and continuous interpolation.

Multi-axis interpolation can realize the motions such as single-section straight-line interpolation, single-section arc interpolation and single-section spiral interpolation.

The continuous interpolation, which is divided into lookbehind and lookahead ones, can realize the continuous running of interpolation curve, smooth transition of speed, reduce machine vibration, improve the processing efficiency and precision.

2.2.4.1 Parameter setting

The setting of parameters, such as speed, acceleration/deceleration and smooth Section S time of interpolated motion, can be realized by different functions.

The interpolation speed curve includes T-shaped and S-shaped curve; see Fig. 2.17 for the setting of speed curve.

“Starting speed”: Set the starting speed of interpolated motion.

“Interpolation speed”: Set the max. running speed during interpolated motion, it is the speed of interpolated axis.

“End speed”: Set the stop speed of interpolated motion.

“Acceleration”: Set the period Tdec consumed from starting speed to the top running speed during interpolated motion

“Deceleration”: Set the period Tdec consumed from top running speed to the stop speed during interpolated motion

"Section S time": Set the time parameter (spara) of Section S of interpolation speed curve.

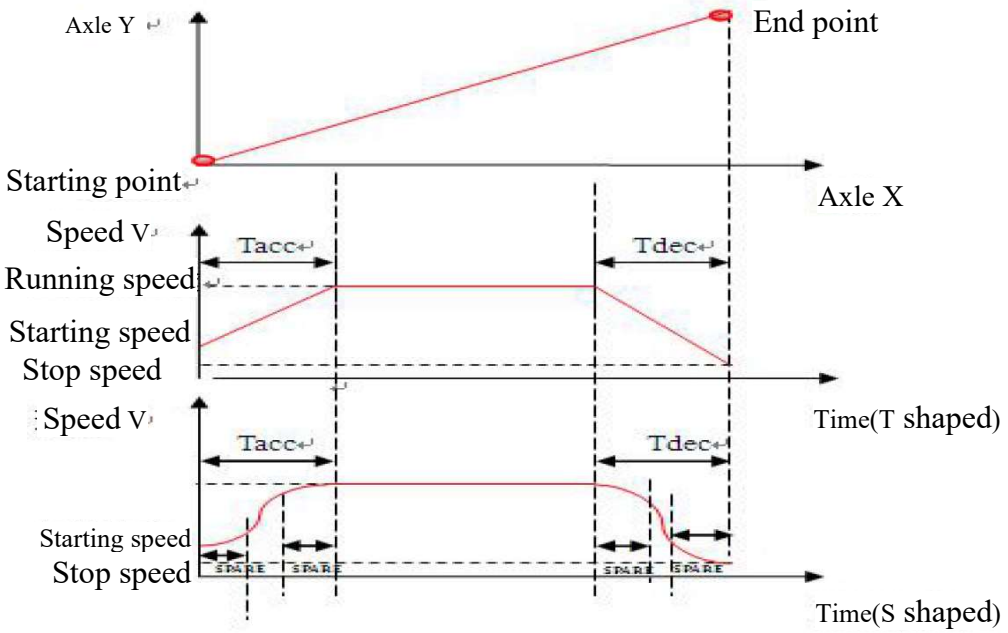


Fig. 2.17 Diagram of T-shaped and S-shaped speed curve and corresponding positions

Relevant functions:

Name	Function	Reference
smc_set_vector_profile_unit	Set speed curve of interpolated motion	Section 3.9
smc_get_vector_profile_unit	Read the speed curve of interpolated motion	
smc_set_vector_s_profile	Set the smoothing time of speed curve of interpolated motion	
smc_get_vector_s_profile	Read the smoothing time of speed curve of interpolated motion	

2.2.4.2 Single-section interpolation

The single-section interpolated motion can realize the multi-axis coordinated motion, such as 2-6 axis straight-line interpolation, plane arc interpolation, space arc interpolation and spiral interpolation. If over 3 axises are engaged in space arc interpolation or spiral interpolation, the subsequent axis will have linear motion along with the first three axises. Arc interpolation supports arc speed limit function, that is to limit the running speed, to keep the acceleration at the set value and reduce the vibration.

Relevant functions:

Name	Function	Reference
smc_line_unit	Straight-line interpolated motion	Section 3.10
smc_arc_move_center_unit	Arc interpolated motion in circle center +end point mode	
smc_arc_move_radius_unit	Arc interpolated motion in radius + end point	

	mode	
Name	Function	Reference
smc_arc_move_3points_unit	Arc interpolated motion in 3-point mode	

Example: Straight-line interpolation of Axis X and Y

```
int main(int argc, char* argv [])
{
/************************Variable definition*****************************/
    short MyCardNo = 0;                 //Connection number
    short ret;                          //Return error code
    WORD Myposi_mode = 0;               //0: Relative mode;  1: Absolute mode
    WORD MyCrd = 0;                     //Coordinate system engaged in interpolated motion
    WORD AxisArray[2];                  //Define axis
    AxisArray[0] = 0;                   //Define interpolated Axis 0 as Axis X
    AxisArray[1] = 1;                   //Define interpolated Axis 1 as Axis Y
    double MyMin_Vel = 0;               //Starting speed 0
    double MyMax_Vel = 3000;            //Top speed of interpolated motion
    double MyTacc = 0.2;                //Acceleration of interpolated motion
    double MyTdec = 0.1;                //Deceleration of interpolated motion
    double MyStop_Vel = 0;              //Stop speed of interpolated motion
    WORD MySmode = 0;                   //Reserved parameter,  fixed value is 0
    double MySpara = 0.05;              //Smoothing time is 0.05s
    WORD MyaxisNum = 2;                 //Number of axises of interpolated motion is 2
    double Dist[2];
    Dist[0] = 10000;                    //Define motion distance of Axis X
    Dist[1] = 8000;                     //Motion distance of Axis Y

/*********************Function call and execution*************************/
    //Step 1: Set the speed parameter of interpolated motion
    ret = smc_set_vector_profile_unit(MyCardNo, MyCrd, MyMin_Vel, MyMax_Vel, MyTacc,
    MyTdec, MyStop_Vel);
    //Step 2: Set the smoothing parameter of interpolated motion
    ret = smc_set_vector_s_profile(MyCardNo, MyCrd, MySmode, MySpara);
    //Step 3: Start interpolated motion
    ret = smc_line_unit(MyCardNo, MyCrd, MyaxisNum, AxisArray, Dist, Myposi_mode);
    }
```

Example 2: Arc interpolation of Axis X and Y, circle center +end point mode

```
    int main(int argc, char* argv [])
    {
/************************Variable definition*****************************/
    short MyCardNo = 0;                 //Connection No.
    short ret;                          //Return error code
    WORD Myposi_mode = 0;               //0: Relative mode;  1: Absolute mode
    WORD MyCrd = 0;                     //Coordinate system engaged in interpolated motion
    WORD AxisArray[2];
    AxisArray[0] = 0;                   //Define interpolated Axis 0 as Axis X
    AxisArray[1] = 1;                   //Define interpolated Axis 1 as Axis Y
    double MyMin_Vel = 0;               //Reserved parameter
```

```
double MyMax_Vel = 3000;        //Max. vector speed of interpolated motion 3000unit/s
double MyTacc = 0.2;            //Acceleration of interpolated motion is 0.2s
double MyTdec = 0.1;            //Motion deceleration
double MyStop_Vel = 0;          //Stop speed
WORD MySmode = 0;               //Reserved parameter,  fixed value is 0
double MySpara = 0.0;           //Smoothing time is 0.05s
WORD MyaxisNum = 2;             //Number of axises of interpolated motion is 2
WORD dir = 0;                   //Arc direction;  0: Clockwise;  1: Anticlockwise
long cic = 0;                   //Arc cycle
double cen[2];                  //Define coordinate of circle center
cen[0] = 10000;                 //Define circle center coordinate of Axis X
cen[1] = 0;                     // Define circle center coordinate of Axis Y
double Dist[2];                 //Define the coordinates of end points
Dist[0] = 0;                    //Define end point of Axis X motion
Dist[1] = 0;                    //Define end point of Axis Y motion

/**********Function call and execution**********/
  //Step 1: Set the speed parameter of interpolated motion
  ret = smc_set_vector_profile_unit(MyCardNo, MyCrd, MyMin_Vel, MyMax_Vel, MyTacc,
  MyTdec, MyStop_Vel);
  //Step 2: Set time parameter S
  ret = smc_set_vector_s_profile(MyCardNo, MyCrd, MySmode, MySpara);
  //Step 3: Execute arc interpolated motion
  ret = smc_arc_move_center_unit(MyCardNo, MyCrd, MyaxisNum, AxisArray, Dist, cen, dir, cic,
  Myposi_mode);
  }
```

2.2.4.3 Continuous interpolation

The controller provides continuous interpolation motion, including lookahead (Mode 1) and lookbehind mode (Mode 0 and mode 2 have different algorithms). The continuous interpolation command supports straight-line interpolation, arc interpolation, spiral interpolation and IO control; as the difference, algorithms can be well applied in small-section track and it is smoother at the connection position.

The continuous interpolation provides arc speed limit function for arc, in order to limit the running speed and keep the acceleration within the set range. Arc limit is enabled by default in lookahead (Mode 1); arc speed limit function is not supported in lookbehind mode; the arc speed limit function can be set freely in lookbehind mode 2. The detailed functions are shown in table below:

Continuous interpolation function	**Lookbehind (Mode 0)**	**Lookahead (Mode 1)**	**Lookbehind (Mode 2)**
Interpolation system	4pcs	2pcs	2pcs
Track of long line section	Supported	Supportd	Supportd
Track of small line section	Not supported	Supportd	Not supported
Arc speed limit	Not supported	Supported	Free setting
Blend function	Supported	Not supported	Not supported

Type T/S curve	Type T and S supported	Type T supported	Type T and S supported
Acceleration/deceleration	Free setting	Symmetric curve	Free setting
Start/stop speed	Free setting	Not supported	Free setting
Speed ratio	Support (take effect on the next track)	Support (take effect immediately)	Support (take effect immediately)
Interpolation delay	Supported	Supported	Supportdd
IO waits for input	Supported	Supported	Supported
IO instant output	Supported	Supported	Supported
IO lead and lag output	Supported	Supportd	Supported

Besides, Leadshine controller supports two coordinate systems, of which, the continuous buffer area support cache of 5, 000 instructions at most. The speed of two coordinate systems can be set independently; they can have interpolation action independently and continuously during continuous interpolation, which means, two groups can have continuous interpolation motion at the same time.

General steps for realizing basic and continuous interpolation motion:

1) If small-section lookahead function is required, set the continuous lookahead mode and parameter by using the function of smc_conti_set_lookahead _mode;
2) Open the buffer zone of continuous interpolation by using the function of smc_conti_open_list;
3) Set the speed curve of continuous interpolation by using the function of smc_set_vector_speed_unit and smc_set_vector_s_profile;
4) Prepare the command of continuous interpolation motion;
5) Start continuous interpolation motion by using the function of smc_conti_start_list;
6) Close the buffer zone of continuous interpolation by using the function of smc_conti_close_list.

Attention: The command of setting lookahead mode of continuous interpolation smc_conti_set_lookahead _mode and parameter instruction should be called before opening the buffer zone of continuous interpolation.

The functions which are called in continuous interpolation are listed below:

a. The functions for continuous interpolation initialization and status detection are shown in table below.

Name	Function	Reference
smc_conti_set_lookahead_mode	Set the lookahead mode and parameters of continuous interpolation	Section 3.11
smc_conti_get_lookahead_mode	Read lookahead mode and parameter of continuous interpolation	
smc_conti_open_list	Open buffer zone of continuous interpolation	
smc_conti_close_list	Close buffer zone of continuous interpolation	

smc_conti_start_list	Start continuous interpolation
smc_conti_pause_list	Pause continuous interpolation
smc_conti_stop_list	Stop continuous interpolation
smc_conti_remain_space	Inquiry the remaining interpolation space of buffer zone
smc_conti_read_current_mark	Read the current interpolation section in buffer zone of continuous interpolation
smc_conti_get_run_state	Read the status of continuous interpolation motion
smc_check_done_multicoor	Detect the status of continuous interpolation motion

b. The relevant functions of continuous interpolation motion are shown in table below.

Name	Function	Reference
smc_conti_line_unit	Straight-line interpolation command in continuous interpolation	Section 3.11
smc_conti_arc_move_center_unit	The spiral continuous interpolation motion expanded based on circle center and arc	
smc_conti_arc_move_radius_unit	The cylindrical and spiral continuous interpolation motion expanded based on radius arc	
smc_conti_arc_move_3points_unit	The cylindrical and spiral continuous interpolation motion expanded based on 3-point arc	

c. The relevant functions of continuous interpolation IO are shown in table below.

Name	Function	Reference
smc_conti_set_pause_output	Set IO output status when continuous interpolation is paused or stopped due to a fault	Section 3.13
smc_conti_get_pause_output	Read IO output status when continuous interpolation is paused or stopped due to a fault	
smc_conti_wait_input	Wait for IO input in continuous interpolation	
smc_conti_delay_outbit_to_start	IO lag output relative to track starting point during continuous interpolation	
smc_conti_delay_outbit_to_stop	IO lag output relative to track end point during continuous interpolation	
smc_conti_ahead_outbit_to_stop	IO advanced output relative to track end point during continuous interpolation	
smc_conti_write_outbit	Instant IO output in buffer zone during continuous interpolation	
smc_conti_clear_io_action	Clear the unexecuted IO action in section	

d. Other functions of continuous interpolation are shown in table below.

Name	Function	Reference
smc_conti_delay	Pause delay command during continuous interpolation.	3.11
smc_conti_change_speed_ratio	Dynamic adjustment of speed ratio of continuous interpolation	
smcsetarclimit	Set arc interpolation function	

smc_get_arc_limit	Read parameter of arc interpolation function
smc_conti_set_blend	Set enabling status of Blend corner transition mode during continuous interpolation
smc_conti_get_blend	Read enabling status of Blend corner transition mode during continuous interpolation

Example 1: Continuous interpolation motion, straight line + arc (lookahead motion)

```
int main(int argc, char* argv [])
{
/*************************Variable definition*****************************/
    short MyCardNo = 0;                    //Connection No.
    short ret;                             //Return error code
    WORD Myposi_mode = 1;                  //0: Relative mode;  1: Absolute mode
    WORD MyCrd = 0;                        //Coordinate system engaged in interpolated motion
    WORD AxisArray[2];
    AxisArray[0] = 0;                      //Define interpolated Axis 0 as Axis X
    AxisArray[1] = 1;                      //Define interpolated Axis 1 as Axis Y
    double MyMin_Vel = 0;                  //Initial speed of interpolation
    double MyMax_Vel = 30000;              //Max. vector speed of interpolated motion 3000unit/s
    double MyTacc = 0.2;                   //Acceleration of interpolated motion is 0.2s
    double MyTdec = 0.1;                   //Deceleration of interpolated motion is 0.1s
    double MyStop_Vel = 0;                 //Stop speed of interpolation
    WORD MySmode = 0;                      //Reserved parameter,  fixed value is 0
    double MySpara = 0.0;                  //Smoothing time is 0.05s
    WORD MyaxisNum = 2;                    //Number of axises of interpolated motion is 2
    WORD enable = 1;                       //Enable Blend function or not;  0: Disable;  1: Enable
    double Dist[2];                        //Define the coordinates of end points
    Dist[0] = 1000;                        //Define end point of Axis X motion
    Dist[1] = 1000;                        //Define end point of Axis Y motion
    WORD dir = 0;                          //Arc direction;  0: Clockwise;  1: Anticlockwise
    long cic = 0;                          //Arc cycle
    double cen[2];                         //Define coordinate of circle center
    cen[0] = 10000;                        //Define circle center coordinate of Axis X
    cen[1] = 0;                            // Define circle center coordinate of Axis Y
    double Dist1[2];                       //Define the coordinates of end points
    Dist1[0] = 20000;                      //Define end point of Axis X motion
    Dist1[1] = 0;                          //Define end point of Axis Y motion
    WORD mode = 1;                         //Define enabling parameter of continuous interpolation;
    0: Continuous;  1: Lookahead motion
    long LookaheadSegment = 200;           //Define number of interpolation section: 200 sections
    double PathError = 1;                  //Define track error: 1unit
    double LookaheadAcc = 10000;           //Define turning acceleration: 10000unit/s2
    ArcLimit = 1;                          //Enable arc speed limit;  0: Disable;  1: Enable

/*********************Function call and execution*************************/
    //Step 1: Set speed parameter and Time S parameter of interpolated motion
    ret = smc_set_vector_profile_unit(MyCardNo, MyCrd, MyMin_Vel, MyMax_Vel, MyTacc,
    MyTdec, MyStop_Vel);
    ret = smc_set_vector_s_profile(MyCardNo, MyCrd, MySmode, MySpara);
```

```
//Step 2: Enable arc speed limit function
ret = smc_set_arc_limit(MyCardNo, MyCrd, ArcLimit, 0, 0);
//Step 3: Set lookahead parameter
ret = smc_conti_set_lookahead_mode(MyCardNo, MyCrd, mode, LookaheadSegment, PathError,
LookaheadAcc);
//Step 4: Open continuous interpolation
ret = smc_conti_open_list(MyCardNo, MyCrd, MyaxisNum, AxisArray);
//Step 5: Start continuous interpolation
ret = smc_conti_start_list(MyCardNo, MyCrd);
//Step 6: Add straight-line interpolation section
ret = smc_conti_line_unit(MyCardNo, MyCrd, MyaxisNum, AxisArray, Dist, Myposi_mode, 0);
//Step 7: Add arc interpolation section
ret = smc_conti_arc_move_center_unit(MyCardNo, MyCrd, MyaxisNum, AxisArray, Dist1, cen,
dir, cic, Myposi_mode, 0);
//Step 8: Close buffer zone of continuous interpolation
ret = smc_conti_close_list(MyCardNo, MyCrd);
}
```

Example 2: IO function during continuous interpolation (lookbehind motion)

```
int main(int argc, char* argv [])
 {
/************************Variable definition*****************************/
   short ret;                          //Error return
   short MyCardNo = 0;                 //Connection No.
   WORD Myposi_mode = 1;               //0: Relative mode;  1: Absolute mode
   WORD MyCrd = 0;                     //Coordinate system engaged in interpolated motion
   WORD AxisArray[2];
   AxisArray[0] = 0;                   //Define interpolated Axis 0 as Axis X
   AxisArray[1] = 1;                   //Define interpolated Axis 1 as Axis Y
   double MyMin_Vel = 0;               //Initial speed of interpolation
   double MyMax_Vel =  = 3000;         //Max. vector speed during interpolated motion is
   3000unit/
   double MyTacc = 0.2;                //Acceleration of interpolated motion is 0.2s
   double MyTdec = 0.1;                //Deceleration of interpolated motion is 0.1s
   double MyStop_Vel = 0;              //Stop speed of interpolation
   WORD MySmode = 0;                   //Reserved parameter,  fixed value is 0
   double MySpara = 0.0;               //Smoothing time is 0.05s
   WORD MyaxisNum = 2;                 //Number of axises of interpolated motion is 2
   WORD enable = 1;                    //Enable Blend function or not;  0: Disable;  1: Enable
   double Dist[2];                     //Define the coordinates of end points
   Dist[0] = 1000;                     //Define end point of Axis X motion
   Dist[1] = 1000;                     //Define end point of Axis Y motion
   WORD MyBitno = 1                    //'General output port 1
   WORD MyLevel = 0;                   //'Output level: Low level
   WORD MyDelayMode = 1;               //'Lag mode: Lag position
   WORD MyDelayVal = 100;              //'Lag position is 100unit
   double MyRevTime = 0.5;             //'Level delay rotation period is 0.5s
   WORD dir = 0;                       //Arc direction;  0: Clockwise;  1: Anticlockwise
   long cic = 0;                       //Arc cycle
   double cen[2];                      //Define coordinate of circle center
```

```
    cen[0] = 10000;                    //Define circle center coordinate of Axis X
    cen[1] = 0;                        // Define circle center coordinate of Axis Y
    double Dist1[2];                   //Define the coordinates of end points
    Dist1[0] = 20000;                  //Define end point of Axis X motion
    Dist1[1] = 0;                      //Define end point of Axis Y motion
    WORD bitno = 1;                    //'Input port 0
    WORD sta = 0;                      //'Input port level: Low level
    WORD TimeOut = 2;                  //'Timeout
    WORD mode = 0;                     //Define enabling parameter of continuous
    interpolation;  0: Continuous;  1: Lookahead motion
    long LookaheadSegment = 200;       //Define number of interpolation section
    double PathError = 0;              //Define track error
    double LookaheadAcc = 1000000;     //Define acceleration of bend

/**********Function call and execution**********/
    //Step 1: Set speed curve of interpolated motion
    ret = smc_set_vector_profile_unit(MyCardNo, MyCrd, MyMin_Vel, MyMax_Vel, MyTacc,
    MyTdec, MyStop_Vel);
    //Step 2: Set the parameter value of Section S of interpolation speed curve
    ret = smc_set_vector_s_profile(MyCardNo, MyCrd, MySmode, MySpara);
    //Step 3: Set lookahead parameter
    ret = smc_conti_set_lookahead_mode(MyCardNo, MyCrd, mode, LookaheadSegment,
    PathError, LookaheadAcc);
    //Step 4: Open continuous interpolation
    ret = smc_conti_open_list(MyCardNo, MyCrd, MyaxisNum, AxisArray);
    //Step 5: Start continuous interpolation
    ret = smc_conti_start_list(MyCardNo, MyCrd);
    //Step 6: Enable Blend function
    ret = smc_conti_set_blend(MyCardNo, MyCrd, enable);
    //Step 7: Add straight-line interpolation section
    ret = smc_conti_line_unit(MyCardNo, MyCrd, MyaxisNum, AxisArray, Dist, Myposi_mode,
    0);
    //Step 8: Lag 100unit relative to starting point of track section,  output port 1 has output of low
    level and low level lasts for 0.5s.
    ret = smc_conti_delay_outbit_to_start(MyCardNo, MyCrd, MyBitno, MyLevel, MyDelayVal,
    MyDelayMode, MyRevTime); //
    //Step 10: Add arc interpolation section
    ret = smc_conti_arc_move_center_unit(MyCardNo, MyCrd, MyaxisNum, AxisArray, Distl,
    cen, dir, cic, Myposi_mode, 0);
    //Step 9: Wait for delay input IO
    ret = smc_conti_wait_input(MyCardNo, MyCrd, bitno, sta, TimeOut, 0);
    //Step 11: Close buffer zone of continuous interpolation
    ret = smc_conti_close_list(MyCardNo, MyCrd);
    }
```

2.2.5 Handwheel motion

Leadshine controller provides a powerful handwheel pulse control function; it can realize ideal handwheel effects for use through flexible configuration, even if there's only one interface. The

controller handwheel provides two modes, i.e. default mode and advanced mode.

To be specific, in default mode, the handwheel function has one-to-one correspondence to the icons on handwheel, such as single-axis control of Axis x, y, z, 4, 5 and 6; it has 3 rate gears: Control at 1, 10 and 100 times.

In advanced mode, the axis level is not controlled by the fixed single axis; instead, the axis group related to each axis level can be set through the function, and the speed level is not the fixed 1, 10 and 100 times. For example: If handwheel is used in three conditions, 1: Both Axis x and y will have motion simultaneously; 2: Both of Axis z and u will have motion simultaneously; 3. Four axises will have motion simultaneously. So, Axis x handwheel can be associated with Axis x and y; meanwhile, the rate of three levels can be set independently and not shared with the level (y, z and u) of handwheels of other axises. Axis y handwheel signal is associated with Axis z and u, and rate can be set independently. Axis z can be associated with Axis x, y, z and u, and rate can be set independently. So, when handwheel axis is pulled to Axis x switch, the motion of both Axis x and y can be controlled simultaneously; when it is pulled to switch of Axis z, the motion of 4 axises can be controlled simultaneously.

Different modes are provided to satisfy the varying operation conditions, to greatly increase the handwheel functions and satisfy the requirements of user. The handwheel is shown in Fig. 2.18 below.

Fig. 2.18 Appearance of handheld manual pulse generator

Relevant functions of handwheel motion:

Name	Function	Reference
smc_handwheel_set_axislist	Set the specific motion axis under the selected level of the same axis	3.17
smc_handwheel_get_axislist	Read the specific motion axis under the selected level of the same axis	
smc_handwheel_set_ratiolist	Set the rate level of selected handwheel of the same axis	
smc_handwheel_get_ratiolist	Read the rate level of selected handwheel of the same axis	
smc_handwheel_set_mode	Set handwheel motion mode, hardware or software	
smc_handwheel_get_mode	Read handwheel motion mode, hardware or software	
smc_handwheel_set_index	Set axis and rate level of handwheel motion	
smc_handwheel_get_index	Read axis and rate level of handwheel motion	
smc_handwheel_move	Start handwheel motion	

smc_handwheel_stop	Stop handwheel motion	

Example 1: Handwheel motion. Axis level 0 of motion, rate level 0, motion axis 0, 1 and 2 are running at 1 times of speed. 5s later, change axis and select level 1, rate level 2, motion axis 4 and 5 are running at 100 times of speed.

```
int main(int argc, char* argv [])
{
/************************Variable definition*****************************/
   short ret = 0;                          //Error return
   short MyCardNo = 0;
   WORD Myposi_mode = 1;                   //0: Relative mode;  1: Absolute mode
   WORD MyCrd = 0;                         //Coordinate system engaged in interpolated motion
   double MyMin_Vel = 0;                   //Initial speed of interpolation
   double MyMax_Vel = 3000;                //Max. vector speed of interpolated motion 3000unit/s
   double MyTacc = 0.2;                    //Acceleration of interpolated motion is 0.2s
   double MyTdec = 0.1;                    //Deceleration of interpolated motion is 0.1s
   double MyStop_Vel = 0;                  //Stop speed of interpolation
   WORD MySmode = 0;                       //Reserved parameter,  fixed value is 0
   double MySpara = 0.0;                   //Smoothing time is 0.05s
   WORD AxisSelIndex = 0;                  //Select level 0 of handwheel axis
   WORD RatioSelIndex = 0;                 //Rate level 0,  1 times of speed
   WORD AxisSelIndex_1 = 1;                //Select level 1 of handwheel axis
   WORD RatioSelIndex_1 = 2;               //Rate level,  100 times of speed
   WORD InMode = 1;                        //Input pulse mode,  0: Pulse + direction,  1: Phase AB pulse
   WORD IfHardEnable = 0;                  //Motion mode: 0: Software control;  1: Hardware control
   WORD AxisNum = 3;                       //Handwheel motion axis: 3 axises
   WORD AxisArray[3];                      //Motion axis at axis level 0: 3 axises
   AxisArray[0] = 0;                       //Motion axis 0
   AxisArray[1] = 1;                       //Motion axis 1
   AxisArray[2] = 2;                       //Motion axis 2
   WORD AxisNum_1 = 2;                     //Motion axis of axis level 1: 2 axises
   WORD AxisArray_1[2];                    //List of handwheel motion axises
   AxisArray_1[0] = 4;                     //Motion axis 4
   AxisArray_1[1] = 5;                     //Motion axis 5
   WORD StartRatioIndex = 0;               /Starting value of multiplication
   WORD RatioSelNum = 3;                   /'Set ratio value of 3
   double RatioList[3];                    //'Set rate list
   RatioList[0] = 1;
   RatioList[1] = 10;
   RatioList[2] = 100;

/*********************Function call and execution**************************/
   //Step 1: Set speed curve of interpolated motion
   ret = smc_set_vector_profile_unit(MyCardNo, MyCrd, MyMin_Vel, MyMax_Vel, MyTacc,
   MyTdec, MyStop_Vel);
   //Step 2: Set smoothing time of interpolation speed curve
   ret = smc_set_vector_s_profile(MyCardNo, MyCrd, MySmode, MySpara);
   //Step 3: Set axis multiplication level 0
   ret = smc_handwheel_set_index(MyCardNo, AxisSelIndex, RatioSelIndex);
```

```
    //Step 4: Set handwheel motion mode,  hardware or software mode
    ret = smc_handwheel_set_mode(MyCardNo, InMode, IfHardEnable);

    //Step 5: Set handwheel level and list of motion axis
    ret = smc_handwheel_set_axislist(MyCardNo, AxisSelIndex, AxisNum, AxisArray;
ret = smc_handwheel_set_axislist(MyCardNo,AxisSelIndex_1,AxisNum_1,AxisArray_1);
    //Set detailed motion axis of axis level 1

    //Step 6: Set handwheel rate level and rate value of corresponding axises
    ret = smc_handwheel_set_ratiolist(MyCardNo, AxisSelIndex, StartRatioIndex, RatioSelNum,
    RatioList);
    ret = smc_handwheel_set_ratiolist(MyCardNo, AxisSelIndex_1, StartRatioIndex, RatioSelNum,
    RatioList);  //Set the corresponding handwheels rates at axis level 1

    //Step 7: Start handwheel motion
    ret = smc_handwheel_move(MyCardNo, 0);
    //Step 8: Switch handwheel motion to level 1 5s later
    Sleep(5000);
    ret = smc_handwheel_set_index(MyCardNo, AxisSelIndex_1, RatioSelIndex_1); }
```

2.2.6 Electronic cam

Leadshine controller supports single axis or multiple axises to have motion along with command position or encoder feedback position of main axis based on the cam relationship.

Data sheet of electronic cam, relative position mode is adopted; Mpos1 and Spos1 are fixed as 0; the main axis position must increase or decrease along with one direction.

Position of master axis	Mpos1	Mpos2	……	Mposn-1	MposN
Position of slave axis	Spos1	Spos2	……	Sposn-1	SposN

Control procedure of electronic cam

1. Download data sheet of cam

2. Start cam motion of slave axis, the slave axis enters cam motion mode and motion status, the current point is the starting point of follow-up motion; the max. follow-up distance equals to the max. distance of master axis in cam sheet;

3. Start motion of master axis, the slave axis will start follow-up motion and it will stop when exceeding the follow-up range.

4. Stop cam motion of slave axis and the slave axis will quit the cam follow-up mode.

Relevant commands:

Name	Function	Reference
smc_cam_table_unit	Set cam sheet	Section 3.17
smc_cam_move	Start motion of electronic cam	

Example:

```
WORD ConnectNo, MasterAxisNo, SlaveAxisNo, Count;
double MasterPos[100], SlavePos[100];
WORD SrcMode, MasterTargetPos;
ConnectNo = 0;                          //Link No.
MasterAxisNo = 0;                       //Master axis No.
SlaveAxisNo = 1;                        //Slave axis No.
SrcMode = 0;                            //Master axis position mode: 0 – Command position;  1-Feedback position
Count = 11;                             //Data number
//Fill data sheet of electronic cam,  1000 groups maximally;  reserve the data of the 1st group and it must be fixed as (0,  0)
for(i = 0; i<Count; i++)
{
      MasterPos[i] = I * (-1000);
      SlavePos[i] = (-100) * (i % 2);
}
//Add electronic cam sheet
iret =
smc_cam_table_unit(ConnectNo,  MasterAxisNo,  SlaveAxisNo,  Count,  MasterPos,  SlavePos,  SrcMode);
//Start motion of electronic came of slave axis
iret = smc_cam_move(ConnectNo, SlaveAxisNo);
//Control motion of master axis
iret = smc_pmove_unit(ConnectNo, MasterAxisNo, MasterPos[Count-1], 0);
```

2.3 General IO function

2.3.1 General IO control

User may use the digital I/O port on Leadshine controller to detect the input signals, such as switching signal and sensor signal, or control the output devices such as relay and solenoid valve.

Leadshine controller supports I/O delay rotation function. Once this function is enabled, it will firstly output a signal that is reverse to the current level and, after finishing the set delay, enable auto rotation of level for once.

Leadshine controller supports input of IO counting function, with which, user may set input IO as counter.

Control relevant functions through general IO:

Name	Function	Reference
smc_read_inbit	Release the level status of certain input port	Section 3.15
smc_write_outbit	Set the output level of certain port	

smc_read_outbit	Read the level status of certain output port	
smc_read_inport	Read the level status of all input ports	
smc_read_outport	Read the level status of all output ports	
smc_write_outport	Set the level status of all output ports	
smc_reverse_outbit	Delayed rotation of IO output	
smc_set_io_count_mode	Set IO counting mode	
smc_get_io_count_mode	Read IO counting mode	
smc_set_io_count_value	Reset IO counting	
smc_get_io_count_value	Read IO counting	

⚠Note: Call the function of smc_read_inbit() to read the level of certain general input port; call the function of smc_read_inport() to read the level of all input ports in one time.

2.3.2 Virtual IO mapping

Leadshine controller supports virtual IO mapping function, which can be used to filter the specific and general IO input interface; besides, the level status of this port can be read through specific function after filtering.

Correlation function of virtual IO mapping:

Name	Function	Reference
smc_set_io_map_virtual	Set virtual IO mapping relationship	Section 3.27
smc_get_io_map_virtual	Read the setting of virtual IO mapping relationship	
smc_read_inbit_virtual	Read the level status of virtual IO port after filtering	

Example 1: Read common IO status

```
int main(int argc, char* argv[])
{
/*************************Variable definition*****************************/
    short ret = 0;                      //Error return
    WORD MyCardNo = 0;                  //Connection No.
    WORD MyInport = 0;                  //Enter port group number
    short vale;
    WORD bitno = 0;                     //IO port 0
    WORD stat = 0;                      //Low level status 0
    WORD portno = 0;                    //io group number
    DWORD port_value = 3;               //Write value 2 and convert to binary 11
    WORD mode = 2;          //IO counting mode;  0: Disable;  1: Rising edge counting;  2: Failing
    edge counting
    bitno = 0;                          //Output port 0
    double filter_time = 0;             //Filtering time,  unit: s
    DWORD CountValue = 0;               //Count value
```

```
    double reverse_time = 1;                //Rotation time,  unit: s

/**********************Introduction to function call and execution**********************/
    //1. Read return value of input IO level status
    long MyInportValue = smc_read_inport(MyCardNo, MyInport);
    printf("Input return value of IO level status = %d\n",  MyInportValue);
    //2. Set output level of single port
    ret = smc_write_outbit(MyCardNo, bitno, stat);
    //3. Read level status of single port
    vale = smc_read_outbit(MyCardNo, bitno);
    printf("Read level status of port = %d\n", vale);  //Print level status value of single port
    //4. Set level status of all output ports
    smc_write_outport(MyCardNo, portno, port_value);
    //5. Read level status of all output ports
    vale = smc_read_outport(MyCardNo, portno);
    printf("Level status of all output ports = %d\n", vale);  //Print level value of all output ports
    //6. Set counting mode and read count value
    ret = smc_set_io_count_mode(MyCardNo, bitno, mode, filter_time);
    ret = smc_get_io_count_value(MyCardNo, bitno, &CountValue);
    printf("Level status of all output ports = %d\n",  CountValue);  print count value
    //7. Set level rotation
    ret = smc_reverse_outbit(MyCardNo,  bitno,  reverse_time);  //Level rotation
}
```

2.4 Special IO function

2.4.1 Encoder detection

Each axis of Leadshine controller provides one encoder input port to detect platform displacement or motor rotation angle. The encoder has three signals, i.e. EA, EB and EZ and pulse counting signal is inputted through EA and EB port; it can receive two types of pulse signals: Forward and negative pulse input or Phase A/B orthogonal signal; EZ signal refers to the zero signal of encoder. The encoder appearance is as shown in Fig. 2.19.

Rotary encoder Grating ruler

Fig. 2.19 Encoder appearance

The probe and encoder are combined; Leadshine controller can realize position detection of workpiece through position trigger function, as shown in Fig. 2.20; in other words, a trigger signal will be generated when probe contacts the workpiece and when this signal is received by the controller,

the current position of encoder will be recorded immediately; the dimensions of this workpiece can be acquired by recording a series of data of workpiece and then handling them through software.

Fig. 2.20 Workpiece and probe under test

Correlation function of encoder:

Name	Function	Reference
smc_set_counter_inmode	Set the counting mode of encoder	Section 3.18
smc_get_counter_inmode	Read the counting mode of encoder	
smc_set_encoder_unit	Set the encoder pulse counting of designated axis	
smc_get_encoder_unit	Read the encoder pulse counting of designated axis	
smc_set_ez_mode	Set the EZ signal level of designated axis	
smc_get_ez_mode	Read the EZ signal level of designated axis	
smc_set_counter_reverse	Set the reverse phase of Phase AB count value	
smc_get_counter_reverse	Read the reverse phase mode of Phase AB count value	

Example: Encoder detection

```
int main(int argc, char* argv[])
{
/************************Variable definition*****************************/
   short ret = 0;                          //Error return
   short res = 0;
   WORD MyCardNo = 0;                      //Connection No.
   WORD Myaxis = 0;                        //Axis No.
   WORD Mymode = 3;                        //Set the counting mode of encoder as 4 times of frequency,
   Phase AB
   double Myencoder_value = 0;
/*********************Function call and execution**************************/
   //Step 1: Set the encoder counting mode of No. 0 axis
   ret = smc_set_counter_inmode(MyCardNo, Myaxis, Mymode);
   //Step 2: Set the encoding value of No. 0 axis as
   100res = smc_set_encoder_unit(MyCardNo, Myaxis, 100);
   //Step 3: Read the encoding value of No. 0 axis
   res = smc_get_encoder_unit(MyCardNo, Myaxis, &Myencoder_value);
}
```

2.4.2 Position latch

Leadshine controller supports latch of multiple high-speed positions, to realize accurate locating, including single-time latch and continuous latch. Single latch: One latch is supported when latch signal becomes effective; the latch mark should be reset before secondary latch.

Continuous latch: Multiple positions can be provided with latch in turns. It is not required to rest latch mark after each latch. When latch times exceed 1, 000, the earliest trigger position will be eliminated to save the latest trigger position. The interval of continuous latch should be higher than 1ms. The correlation function of high-speed latch:

Name	Function	Reference
smc_set_ltc_mode	Set the LTC signal of designated axis	Section 3.19
smc_get_ltc_mode	Read the LTC signal setting of designated axis	
smc_set_latch_mode	Set latch mode: Single and continuous latch	
smc_get_latch_mode	Read latch mode	
smc_get_latch_value_unit	Read value of encoder latch from the controller	
smc_get_latch_flag	Read the latch times of designated axis from the controller	
smc_reset_latch_flag	Reset the latch mark position of designated controller	

Correlation function of original point latch:

Name	Function	Reference
smc_set_homelatch_mode	Set latch mode of original point	Section 3.20
smc_get_homelatch_mode	Read latch mode setting of original point	
smc_reset_homelatch_flag	Clear latch mark of original point	
smc_get_homelatch_flag	Read latch mark of original point	
smc_get_homelatch_value_unit	Read latch value of original point	

Example 1: Latch of multiple high-speed positions

```
int main(int argc, char* argv [])
{
/************************Variable definition*****************************/
    WORD MyCardNo = 0;              //Connection No.
    WORD Myaxis = 0;                //Axis No.
    short ret;                      //Return error code
    WORD Myltc_logic = 0;           //Set LTC trigger mode as failing edge trigger
    WORD Myltc_mode = 0;            //Reserved value 0
    double Myfilter = 0;            //Reserved parameter
    WORD ltc_mode = 2;              //Multiple-latch mode;  0: Single latch;  2: Continuous latch
    WORD Mylatch_source = 0;        //Set latch source as command position
    WORD triger = 0; inti = 0;      //Trigger channel,  fixed value is 0
```

```
/**********************Function call and execution*************************/
   //Step 1: Set LTC signal of No. 0 axis and trigger method is failing edge trigger
   ret = smc_set_ltc_mode(MyCardNo, Myaxis, Myltc_logic, Myltc_mode, Myfilter);
   //Step 2: Set the latch parameter of No. 0 axis
   ret = smc_set_latch_mode(MyCardNo, Myaxis, ltc_mode, Mylatch_source, triger);
   //Step 3: Reset the latch status of No. 0 axis
   ret = smc_reset_latch_flag(MyCardNo, Myaxis);
   //Step 4: Set speed parameter
   ret = smc_set_profile_unit(MyCardNo, Myaxis, 0, 5000, 0.1, 0.1, 0);
   //Step 5: Start fixed-length motion and wait for motion stop
   ret = smc_pmove_unit(MyCardNo, Myaxis, 10000, 0);
   while(smc_check_done(MyCardNo, Myaxis) == 0);        //Wait for motion stop
   //Step 6: Read latch number
   intMy_latch_flag = smc_get_latch_flag(MyCardNo, Myaxis);
   printf("Latch number = %d\n",  My_latch_flag);      //Print latch number
   //Step 7: Read latch number if(My_latch_flag>0)
   {
      double* My_latch_Value = newdouble [My_latch_flag];
      for(i = 0; i< = My_latch_flag; i++)
         {
            smc_get_latch_value_unit(MyCardNo, Myaxis, &My_latch_Value[i]);
            printf("My_latch_Value = %f\n", My_latch_Value[i]);    //Print latch number
         }
   }
}
```

Example 2: Latch original point; the latch mark of original pint is effective when original point signal, which has contact during motion, is effective

```
int main(int argc, char* argv [])
{
/**************************Variable definition*****************************/
   short ret = 0;                          //Error return
   WORD MyCardNo = 0;                      //Connection No.
   WORD Myaxis = 0;                        //Axis No.
   WORD enable = 1;                        //Enable original point latch;  0: Disable;  1: Enable
   WORD logic = 0;                         //Set LTC trigger mode as failing edge trigger
   WORD source = 0;                        //Set latch source;  0: Command position;  1: Encoder position
   double pos = 0;                         //Read latch value of original point
/**********************Function call and execution*************************/
   //Step 1: Set latch mode
   ret = smc_set_homelatch_mode(MyCardNo, Myaxis, enable, logic, source);
   //Step 2: Reset latch status of No. 0 axis
   ret = smc_reset_homelatch_flag(MyCardNo, Myaxis);
   //Step 3: Set speed parameter
   ret = smc_set_profile_unit(MyCardNo, Myaxis, 0, 500, 0.1, 0.1, 0);
   //Step 4: Start fixed-length motion
   ret = smc_pmove_unit(MyCardNo, Myaxis, 10000, 0);
```

```
    while(smc_check_done(MyCardNo, Myaxis) == 0);        //Wait for motion stop
    //Step 5: Judge latch mark and read latch value
    if((smc_get_homelatch_flag(MyCardNo, Myaxis))>0)
        {
            smc_get_homelatch_value_unit(MyCardNo, Myaxis, &pos);
            printf("Original point latch value = %f\n",  pos);
        }
}
```

2.4.3 Position comparison and output

Leadshine provides the function of output signal triggered by position, including comparison of single-axis low speed position, comparison of single-axis high-speed position, comparison of 1 group 2D low-speed position. When motor reaches the preset position, it will trigger the specific output port automatically. This function can facilitate the control of dispensing valve and trigger of camera shutter. The correlation functions for comparison of one-dimensional low-speed position are shown in table below.

Leadshine controller provides the position comparison function. The general steps of position comparison:

1. Configure comparator;
2. Clear comparator;
3. Add/update comparison position;
4. Start motion and view comparison status.

Correlation function of one-dimensional low-speed position comparison;

Name	Function	Reference
smc_compare_set_config	Set one-dimensional position comparator	Section 3.22
smc_compare_get_config	Read settings of one-dimensional position comparator	
smc_compare_clear_points	Clear one-dimensional position comparison points	
smc_compare_add_point_unit	Add one-dimensional position comparison points	
smc_compare_get_current_point_unit	Read the position of current one-dimensional comparison point	
smc_compare_get_points_runned	Inquiry the number of one-dimensional comparison points which have been compared	
smc_compare_get_points_remained	Inquiry the number of one-dimensional comparison points which can be added	

⚠Note: (1) Position comparison of each axis is carried out independently.

(2) When performing position comparison, the trigger of each comparison point is executed by the

sequence of added comparison points, which means, if one comparison point is not triggered for comparison action, the following comparison points will not work.

Correlation function of two-dimensional low-speed position comparison;

Name	Function	Reference
smc_compare_set_config_extern	Set two-dimensional position comparator	Section 3.22
smc_compare_get_config_extern	Read parameters of two-dimensional comparator	
smc_compare_clear_points_extern	Clear comparison points of two-dimensional position	
smc_compare_add_point_extern_unit	Add comparison points of two-dimensional position	
smc_compare_get_current_point_extern_unit	Read that position of the comparison point of the current two-dimensional position	
smc_compare_get_points_runned_extern	Inquiry the number of two-dimensional comparison points which have been compared	
smc_compare_get_points_remained_extern	Inquiry the number of two-dimensional comparison points which can be added	

Correlation function of dimension high-speed position comparison

Name	Function	Reference
smc_hcmp_set_mode	Set the high-speed comparison mode	Section 3.23
smc_hcmp_get_mode	Read the settings of high-speed comparison mode	
smc_hcmp_set_config	Configure the high-speed comparator	
smc_hcmp_get_config	Read the configuration of high-speed comparator	
smc_hcmp_set_liner_unit	Set the parameters of high-speed comparison linear mode	
smc_hcmp_get_liner_unit	Read the parameters of high-speed comparison linear mode	
smc_hcmp_clear_points	Clear the high-speed position comparison point	
smc_hcmp_add_point_unit	Add/update high-speed comparison position	
smc_hcmp_get_current_state_unit	Read high-speed comparison status	
smc_write_cmp_pin	Control the output of designated CMP port	
smc_read_cmp_pin	Read the level of designated CMP port	

⚠Note: (1) The position comparison of each comparator is carried out independently. If OUT16 and OUT17 are no longer compared through position comparison, we need to enable the disable mode under the instruction of SMCHcmpSetMode(Myhcmp, 0) and release the comparison ports, otherwise 0UT16 and 0UT17 may not be used when being used as

independent output port.

(2) When performing position comparison in the queue and linear comparison mode, the triggering of each comparison point is performed by sequence of added comparison points, which means, if one comparison point is not triggered for comparison action, the subsequent comparison points will not be triggered.

Example 1: Comparison of one-dimensional low-speed position

```
int main(int argc, char* argv[])
{
/************************Variable definition*****************************/
   short ret = 0;                          //Error return
   WORD MyCardNo = 0;                      //Connection No.
   WORD Myaxis = 0;                        //Axis No.
   WORD enable = 1;                        //Enable original point latch;  0: Disable;  1: Enable
   WORD source = 0;                        //Set latch source,  where 0 is instruction position and 1 is
   encoder bit.
   double Mypos = 100;                     /Set comparison position
   WORD Mydir = 1;                         //Comparison mode;  0: Less than or equal to,  1: Higher than
   or equal to
   WORD Myaction = 3;                      //Set the trigger function as reverse IO level
   WORD Myactpara = 0;                     //Set the trigger function of output IO port 0.
   double pos = 0;  //comparison value
   long pointNum = 0;  //Number of compared points
   long pointNum1 = 0;  //Number of remaining comparison points

/*********************Function call and execution**************************/
   //Step 1: Clear comparison points
   ret = smc_compare_clear_points(MyCardNo, Myaxis);
   //Step 2: Configure the comparator
   ret = smc_compare_set_config(MyCardNo, Myaxis, enable, source);
   //Step 3: Configure comparison point parameters
   ret = smc_compare_add_point_unit(MyCardNo, Myaxis, Mypos, Mydir, Myaction, Myactpara);
   //Step 4: Set the motion speed parameters
   ret = smc_set_profile_unit(MyCardNo, Myaxis, 0, 500, 0.1, 0.1, 0);
   //Step 5: Start motion
   ret = smc_pmove_unit(MyCardNo, Myaxis, 1000, 0);
   while(smc_check_done(MyCardNo, Myaxis) == 0);        //Wait for motion stop
   //Step 6: Read the position value of the current comparison point
   ret = smc_compare_get_current_point_unit(MyCardNo, Myaxis, &pos);
   Printf ("Read current comparison point = %f\n",  pos);     //Print comparison point value
   //Step 7: Query the compared points
   ret = smc_compare_get_points_runned(MyCardNo, Myaxis, &pointNum);
   Printf ("compared points = %d\n",  pointnum);
   //Step 8: Inquiry the remaining number of comparison points
   ret = smc_compare_get_points_remained(MyCardNo, Myaxis, &pointNum1);
   Printf ("Number of comparison points which can be added = %d\n",  pointnum1);
}
```

Example 2: Comparison of one-dimensional high-speed position, queue mode

```
int main(int argc, char* argv [])
{
/************************Variable definition*****************************/
   short ret = 0;                    //Error return
   WORD MyCardNo = 0;                //Connection No.
   WORD Myaxis = 0;                  //Axis No.
   WORD hcmp = 0;                    //High speed comparator
   WORD source = 0;                  //Set comparison source, where 0 is instruction position and
   1 is encoder position
   WORD Logic = 0;                   //Low level is effective
   long MyTime = 0;                  //Pulse width, unit: us
   double cmp_pos = 1000;            //Add comparison point 0
   double cmp_pos1 = 2000;           //Add comparison point 1
   double cmp_pos2 = 5000;           //Add comparison point 2
   long remained_points = 0;         //Return the number of comparison points which can be
   added
   double current_point = 0;         //Return the position of the current comparison point; unit:
   unit
   long runned_points = 0;           //Return the compared points
/*********************Function call and execution**************************/
   //Step 1: Set comparator parameters
   ret = smc_hcmp_set_config(MyCardNo, hcmp, Myaxis, source, Logic, MyTime);
   //Step 2: Set the comparator mode
   WORD cmp_mode = 4;  //0: Disabled (default value, 1: Equal to, 2: Lower than, 3: Greater than,
   4: Queues, 5: Linear)
   ret = smc_hcmp_set_mode(MyCardNo, hcmp, cmp_mode);
   //Step 3: Clear all added comparison points of high-speed position
   ret = smc_hcmp_clear_points(MyCardNo, hcmp);
   //Step 4: Add the value of position comparison point
   ret = smc_hcmp_add_point_unit(MyCardNo, hcmp, cmp_pos);
   ret = smc_hcmp_add_point_unit(MyCardNo, hcmp, cmp_pos1);
   ret = smc_hcmp_add_point_unit(MyCardNo, hcmp, cmp_pos2);
   //Step 5: Set speed parameters
   ret = smc_set_profile_unit(MyCardNo, Myaxis, 0, 500, 0.1, 0.1, 0);
   //Step 6: Start motion
   ret = smc_pmove_unit(MyCardNo, Myaxis, 1000, 0);
   while(smc_check_done(MyCardNo, Myaxis) == 0);       //Wait for motion stop
   //Step 7: Read the position comparison status
   ret = smc_hcmp_get_current_state_unit(MyCardNo, hcmp, &remained_points, &current_point,
   &runned_points);
   Printf ("Read current comparison point = %f\n", current _ point);
   Printf ("compared points = %d\n", running _ points);
   Printf ("Number of comparison points which can be added = %d\n", remaining _ points);
}
```

2.4.4 PWM output

The Leadshine controller provides PWM output function. As shown in Fig. 2.21, the period of PWM waveform output by Leadshine controller is t2 (frequency is 1/t2), the duty ratio is t1/t2, and the amplitude is V1 = 5V.

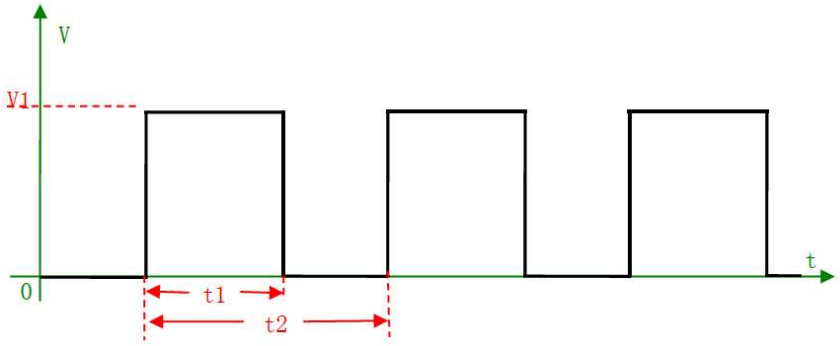


Fig. 2.21. Schematic diagram of PWM output

Basic principle of PWM (Pulse Width Modulation): Control the switching devices of inverter circuit, to acquire a series of pulses with equal amplitude at the output end, and these pulses can be used to replace sine waves or required waveforms.

Association function of PWM function:

Name	Function	Reference
smc_set_pwm_output	Set immediate output of PWM	Section 3.14
smc_get_pwm_output	Read the current output state of PWM	

Example: PWM output function

```
int main(int argc, char* argv[])
{
/*************************Variable definition*****************************/
   short ret = 0;                    //Error return
   WORD MyCardNo = 0;                //Connection No.
   WORD MyPwmNo = 0;                 //PWM output channel is 0
   double MyfDuty = 0.5;             //PWM output duty cycle
   double MyfFre = 10000;            //PWM output frequency
   double MyfDuty1;                  //PWM output duty cycle
   double MyfFre1;                   //PWM output frequency

/*********************Function call and execution**************************/
   //Step 1: Set PWM output parameters
   ret = smc_set_pwm_output(MyCardNo, MyPwmNo, MyfDuty, MyfFre);
   //Step 1: Read and print PWM output parameters
   ret = smc_get_pwm_output(MyCardNo, MyPwmNo, &MyfDuty1, &MyfFre1);
   Printf("PWM output duty ratio = %f\n ", myfduty1);
   Printf("PWM output frequency = %f\n ", myfre1);
}
```

2.4.5 Specific function of servo

It is very convenient to use servo motor control signals SEVON, RDY, ALM, INP and ERC on the Leadshine controller when the equipment has been fitted with an AC servo motor, SEVON is the control signal output by the controller to the servo motor driver. When the SEVON signal is invalid, the servo driver will not be enabled and the motor will be in free state. When the SEVON signal is valid, the servo driver will be enabled while the motor is locked; wait for command pulse signal.

The ALM signal is a status signal sent from the servo motor driver to the controller, so as to report the error of servo driver or motor. When the controller receives the ALM signal, it will stop sending pulses immediately, and it is a hardware processing process.

INP signal is a status signal sent from the servo motor driver to the controller, so as to inform the motion controller that servo motor has stopped.

Generally, servo motor driver has a position deviation counter to record the deviation between command pulse and position feedback pulse and could controlmotor movement. In this way, the position deviation tends to zero, but the actual position of the motor always lags the command pulse. As a result, when the instruction pulse of motion controller is sent, the servo motor does not stop immediately but continues to move, until the position deviation tends to zero, as shown in Fig. 2.22; then the driver will send out an INP signal.

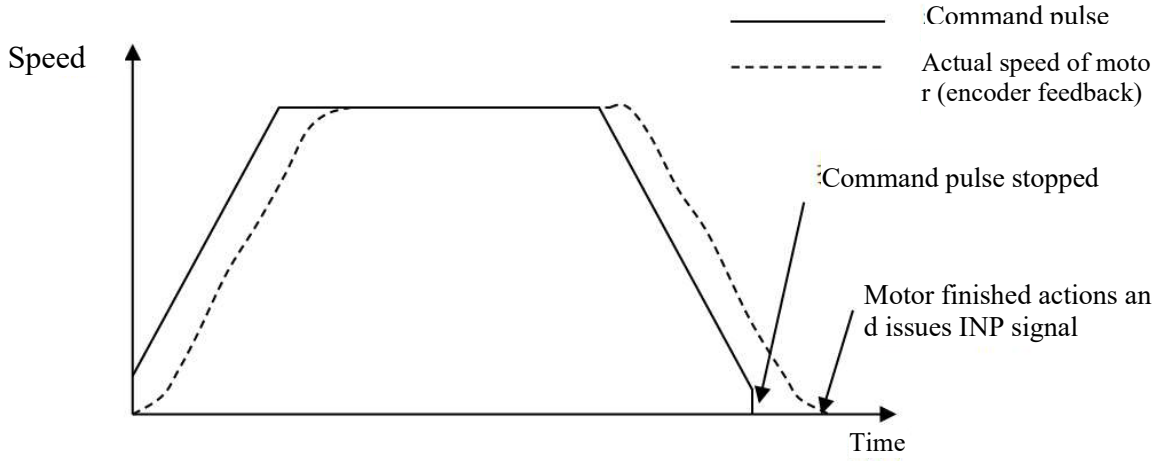


Fig. 2.22. INP signal when servo positioning is completed

The ERC signal is a control signal output by the controller to the servo motor driver.

Servo driver can drive motor movement based on the error between the target position of motor (the target position of motor) and the current position (the position reached by motor). The motor will stop moving if this error is zero. When the servo driver receives the ERC signal from the motion controller, it will clear the error and stop motor from running immediately.

Correlation function of specific IO:

Name	Function	Reference
smc_set_alm_mode	Sets the ALM signal of specified axis	Section 3.16.
smc_get_alm_mode	Read the ALM signal setting of the specified axis	
smc_read_alarm_pin	Read the ALARM port level of the specified axis	
smc_set_inp_mode	Sets the INP signal for the specified axis	
smc_get_inp_mode	Read the INP signal settings for the specified axis	
smc_read_inp_pin	Read the INP port level of the specified axis	
smc_read_rdy_pin	Read the level status of RDY port of the specified axis	
smc_write_sevon_pin	Control the output of servo enable port of the specified axis	
smc_read_sevon_pin	Read the level of servo enable port of the specified axis	
smc_write_erc_pin	Control the ERC signal output of the specified axis	
smc_read_erc_pin	Read the ERC port level status of the specified axis	
smc_read_org_pin	Read the 0RG port level of the specified axis	
smc_read_elp_pin	Read the ELP port level of the specified axis	
smc_read_eln_pin	Read the ELN port level of the specified axis	
smc_read_emg_pin	Read the EMG port level of the specified axis	

⚠Note: (1) IO mapping is required before use if the controller has no corresponding hardware interface; for example, mapping is required by the INP signal of SMC606 controller prior to use.

(2) When the axis number of some functions is set as 255, similar parameters of all axes will be set. The functions are smc_set_el_mode, smc_set_alarm_mode, smc_set_inp_mode, smc_set_home_pin_logic, smc_set_ez_mode and smc_set_io_dstp_mode.

2.4.6 Limit function

Both hardware and software limit functions are provided by Leadshine controller. The user can set the effective level of limit switch according to the hardware limit switch of device, or directly set the software limit position value by software.

Association function of limit function:

Name	Function	Reference
smc_set_el_mode	Set limit switch signal	Section 3.24
smc_get_el_mode	Read the limit switch signal settings	
smc_set_softlimit_unit	Set soft limit parameters	

smc_get_softlimit_unit	Read soft limit parameters	

Example: Set limit switch parameters

```
int main(int argc, char* argv [])
{
/************************Variable definition*****************************/
    WORD MyCardNo = 0;              //Connection No.
    short ret = 0;                  //Error return
    WORD Myaxis = 0;                //Axis No.
    WORD Myel_enable = 1;           //Enable positive and negative limit
    WORD Myel_logic = 0;            //Low level of positive and negative limit is effective
    WORD Myel_mode = 0;             //Stop mode of positive and negative limit: Immediate stop
    Myel_enable = 1;                //Enable status, 0: Disable, 1: Enable
    WORD source_sel = 0;            //Counter selection, 0: Instruction position counter 1:
    Encoder counter
    WORD SL_action = 1;             //Limit stop mode, 0: Immediate stop, 1: Deceleration stop
    double P_limit = 1000;          //Positive limit position, unit
    double N_limit = -1000;
/*********************Function call and execution*************************/
    //Step 1: Set the hardware limit signal of Axis 0.
    ret = smc_set_el_mode(MyCardNo, Myaxis, Myel_enable, Myel_logic, Myel_mode);
    //Step 2: Read the hardware limit signal parameters of Axis 0.
    ret = smc_get_el_mode(MyCardNo, Myaxis, &Myel_enable, &Myel_logic, &Myel_mode);
    Printf ("read hardware limit setting parameter = %d%d%d\n", Myel_enable, Myel_logic, myel _
    mode);
    //Step 3: Set soft limit parameters
    ret = smc_set_softlimit_unit(MyCardNo, Myaxis, Myel_enable, source_sel, SL_action, N_limit,
    P_limit);
    //Step 4: Read the soft limit parameters
    ret = smc_get_softlimit_unit(MyCardNo, Myaxis, &Myel_enable, &source_sel, &SL_action,
    &N_limit, &P_limit);
    Printf ("Soft limit position = %f%f\n", N_limit, p _ limit);     //print soft limit set value
}
```

2.4.7 Emergency stop function

The Leadshine controller has been fitted with motion emergency stop function, corresponding hardware interface circuit for wiring, and calls the emergency stop switch to set the function of smc_set_emg_mode.

Association function of emergency stop:

Name	Function	Reference
smc_set_emg_mode	Set EMG emergency stop signal	Section 3.25
smc_get_emg_mode	Read setting of EMG emergency stop signal	

⚠Note: IO mapping is required before use if the controller has no emergency stop hardware

interface, such as SMC606 controller.

Example: Set the interface of general input port 0 as the mapping entry of the emergency stop signal of Axis 0, and the low level is valid.

```
int main(int argc, char* argv [])
{
/************************Variable definition*****************************/
    WORD MyCardNo = 0;              //Connection No.
    WORD Myaxis = 0;                //Axis No.
    short ret = 0;                  //Error return
    WORD Myenable = 1;              //Enable emergency stop signal
    WORD Mylogic = 0;               //The low level of emergency stop signal is effective.

/*********************Function call and execution**************************/
    //Step 1: Set the axis IO mapping, use the general input 0 as the emergency stop signal of each axis
    ret = smc_set_axis_io_map(MyCardNo, Myaxis, 3, 6, 0, 0);
    //Step 2: Set EMG enabling, low level is effective
    ret = smc_set_emg_mode(MyCardNo, Myaxis, Myenable, Mylogic);
    //Step 3: Read EMG enabling, and the low level is effective.
    ret = smc_get_emg_mode(MyCardNo, Myaxis, &Myenable, &Mylogic);
    Printf ("Emergency stop signal parameter, enable, effective level = %d%d\n", Myenable, mylogic);
    //Step 4: Start fixed-length motion
    ret = smc_set_profile_unit(MyCardNo, Myaxis, 0, 1000, 0.1, 0.1, 0);
    ret = smc_set_s_profile(MyCardNo, Myaxis, 0, 0.05);
    ret = smc_pmove_unit(MyCardNo, Myaxis, 10000, 1);
}
```

Motion result: After running the program, the motion will stop immediately when the IN port 0 is at low level.

2.5 Document function

The file function of controller is designed to control uploaded and downloaded files (such as basic files, G code files and parameter files).

Association function of file:

Name	Function	Reference
smc_download_file	Download local files to FLASH.	Section 3.29
smc_download_memfile	Download memory files to FLASH.	
smc_upload_file	Upload FLASH file to local file.	
smc_upload_memfile	Upload FLASH file to memory file.	
smc_download_file_to_ram	Download the local file to RAM, not saved after power failure.	
smc_download_memfile_to_ram	Download the memory file to RAM, not saved	

	after power failure.	
smc_get_progress	File download progress	

2.6 Register operation function

The Leadshine register BIT has 10,000 bits, while the register REG has 10,000 words. Their addresses are assigned as follows:

BIT00000~BIT09999: Customer-defined area.

BIT10000~BIT10299: Digital input port level. Arranged from port 0 by sequence, 0- off (high level), 1- on (low level).

BIT10300~BIT10399: Status of negative limit input signal. 0- normal, 1- alarm.

BIT10400~BIT10499: Status of positive limit input signal. 0- normal, 1- alarm.

Bit10500 ~ Bit10599: Status of Home input signal. 0- normal, 1- alarm.

Bit10600 ~ Bit10699: Status of alarm input signal. 0- normal, 1- alarm.

BIT11000~BIT11399: Digital output port level. Arranged from port 0 by sequence, 0- off (high level), 1- on (low level).

REG00000~REG09999: Customer defined area.

Reg10000 ~ REG10001: Current position of Axis 0, unit: pulse, type: int32.

REG10002 ~ REG10003: Current position of Axis 1, unit: pulse, type: int32.

Reg10004 ~ REG10005: Current position of Axis 2, unit: pulse, type: int32.

Reg 10006 ~ REG 10007: Current position of Axis 3, unit: pulse, type: int32.

Reg 10008 ~ REG 10009: Current position of Axis 4, unit: pulse, type: int32.

Reg 10010 ~ REG 10011: Current position of Axis 5, unit: pulse, type: int32.

REG 10100 ~ REG 10101: Current position of Axis 0, unit, type: float.

REG 10102 ~ REG 10103: Current position of Axis 1, unit, type: float.

REG 10104 ~ REG 10105: Current position of Axis 2, unit, type: float.

REG 10106 ~ REG 10107: Current position of Axis 3, unit, type: float.

REG 10108 ~ REG 10109: Current position of Axis 4, unit, type: float.

REG 10110 ~ REG 10111: Current position of Axis 5, unit, type: float.

REG 10200 ~ REG 10201: Current speed of Axis 0, unit: pulse/s, type: int32.

Reg 10202 ~ REG 10203: Current speed of Axis 1, unit: pulse/s, type: int32.

REG 10204 ~ REG 10205: Current speed of Axis 2, unit: pulse/s, type: int32.

REG 10206 ~ REG 10207: Current speed of Axis 3, unit: pulse/s, type: int32.

REG 10208 ~ REG 10209: Current speed of Axis 4, unit: pulse/s, type: int32.

REG 10210 ~ REG 10211: Current speed of Axis 5, unit: pulse/s, type: int32.

REG 10300 ~ REG 10301: Current speed of Axis 0, unit/s, type: float.

REG 10302 ~ REG 10303: Current speed of axis 1, unit/s, type: float.

REG 10304 ~ REG 10305: Current speed of Axis 2, unit/s, type: float.

REG 10306 ~ REG 10307: Current speed of Axis 3, unit/s, type: float.

REG 10308 ~ REG 10309: Current speed of Axis 4, unit/s, type: float.

REG 10310 ~ REG 10311: Current speed of Axis 5, unit/s, type: float.

REG10400 ~ REG10401: Current position of Encoder 0, unit: pulse, type: int32.

REG 10402 ~ REG 10403: Current position of Encoder 1, unit: pulse, type: int32.

REG 10404 ~ REG 10405: Current position of Encoder 2, unit: pulse, type: int32.

REG 10406 ~ REG 10407: Current position of Encoder 3, unit: pulse, type: int32.

REG 10408 ~ REG 10409: Current position of Encoder 4, unit: pulse, type: int32.

REG 10410 ~ REG 10411: Current position of Encoder 5, unit: pulse, type: int32.

REG 10500 ~ REG 10501: Current position of Encoder 0, unit, type: float.

REG 10502 ~ REG 10503: Current position of Encoder 1, unit, type: float.

REG 10504 ~ REG 10505: Current position of Encoder 2, unit, type: float.

REG 10506 ~ REG 10507: Current position of Encoder 3, unit, type: float.

REG 10508 ~ REG 10509: Current position of Encoder 4, unit, type: float.

REG 10510 ~ REG 10511: Current position of Encoder 5, unit, type: float.

Correlation function of register operation:

Name	Function	Reference
smc_set_modbus_0x	Write bit register	Section 3.30.
smc_get_modbus_0x	Read bit register	
smc_set_modbus_4x	Write word register	
smc_get_modbus_4x	Read word register	

Example: Register operation

```
int main(int argc, char* argv[])
{
/************************Variable definition*****************************/
    short ret = 0;                      //Error return
    WORD MyCardNo = 0;                  //Connection No.
    WORD start = 0;                     //Register head address.
    WORD inum = 3;                      //Number of registers
    charpdata = 5;                      //Send data value.
    charpdata1;
    WORD inum1 = 2;                     //Number of registers
    WORD pdata2[2];                     //Send data value.
    pdata2[0] = 1;
    pdata2[1] = 2;
```

```
    WORD pdata3[2];

/**********************Function call and execution*************************/
    //Step 1: Write a bit register,  in which,  the values of Register 0,  1 and 2 are 1,  0 and 1 respectively.
    ret = smc_set_modbus_0x(MyCardNo, start, inum, &pdata);
    //Step 2: Read the bit register.
    ret = smc_get_modbus_0x(MyCardNo, start, inum, &pdata1);
    //Step 3: Write the register.
    ret = smc_set_modbus_4x(MyCardNo, start, inum1, pdata2);
    //Step 4: Read the word register.
    ret = smc_get_modbus_4x(MyCardNo, start, inum1, pdata3);
}
```

2.7 Controller networking

The Leadshine controller can be connected to multiple controllers simultaneously through the devices such as switch, and each controller can operate independently, as shown in diagram below.

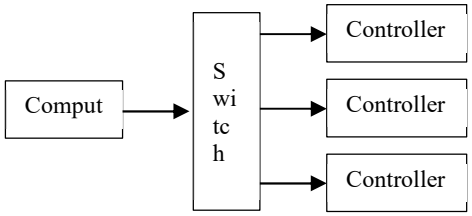


Fig. 2.22. Schematic diagram of controller networking.

If multiple SMC606 controllers are connected, Controller 0 will undergo PT motion and Controller 1 will undergo linear interpolation motion. The realization steps are as follows:

(1) Set the IP address of each controller in the same network segment, but their last bit is different; for example, set the IP of Controller 0 as 192.168.5.11 and Controller 1 as 192.168.5.22.

(2) Write a program and connect two controllers.

(3) Edit instructions to control each controller to have different actions.

⚠Note: The IP values (the last bit) of each group of network controllers should be different before the controllers are connected.

Example: Connect 2 controllers through the switch. Controller 0 executes fixed-length motion, and after the motion is completed, it will conduct linear interpolation motion for Controller 1.

```
/************************Variable definition*****************************/
    WORD ConnectNo = 0;                 //Connection number of Controller 0,  ranged 0-7.
    WORD ConnectNo2 = 1;                //Connection number of Controller 1,  ranged 0-7.
    WORD type = 2;                      //Link type: 1 – Serial port;  2 – Ethernet port
    char *pconnectstring = "192.168.5.11";     // IP address of Controller 0
    char *pconnectstring2 = "192.168.5.22";    // IP address of Controller 1
```

```
    DWORD baud = 0;
    WORD ret;                          //Return error value
    WORD axis = 0;                     //Fixed length motion axis
    double vel = 2000;                 //Fixed length motion speed
    double dist = 2000;                //Fixed motion distance
    WORD MyCrd = 0;                    //Interpolation system 0
    double MyMax_Vel = 1000;           //Interpolation speed
    double MyTacc = 0.1;               //Acceleration and deceleration
    double MySpara = 0.05;             //Time of Section S
    WORD AxisArray[2];                 //Define axis
    AxisArray[0] = 0;                  //Define interpolated Axis 0 as Axis X
    AxisArray[1] = 1;                  //Define interpolated Axis 1 as Axis Y
    double pos[2];
    pos[0] = 10000;                    //Define motion distance of Axis X
    pos[1] = 8000;                     //Motion distance of Axis Y
    WORD MyaxisNum = 2;                //Interpolation axis number.
/**********************Function call and execution**************************/
    //Step 1: Connect Controller 0 and 1
    ret = smc_board_init(ConnectNo, type, pconnectstring, baud);
    //Initialize link of Controller 0
    if(ret! = 0)
    {printf("Controller 0 initialization failed: smc_board_init = %d\n ", ret);  }
    else
    {printf("Controller 0 initialization succeeded \ n ");  }
    ret = smc_board_init(ConnectNo2, type, pconnectstring2, baud);
    if(ret! = 0)
    {printf("Controller 1 initialization failed: smc_board_init = %d\n ", ret);  }
    else
    {printf("Controller 1 initialization succeeded \ n ");  }
    //Step 2: Set Axis 0 and Axis 1 of Controller 0 and 1 as 0
    ret = smc_set_position_unit(ConnectNo, 0, 0);  //Clear position
    ret = smc_set_position_unit(ConnectNo, 1, 0);
    ret = smc_set_position_unit(ConnectNo2, 0, 0);
    ret = smc_set_position_unit(ConnectNo2, 1, 0);
    //Step 3: Set the fixed-length motion speed parameter of Controller 0
    ret = smC_Set_profile_unit(ConnectNo, axis, 0, vel, 0.1, 0.1, 0);
    ret = smC_Set_S_profile(ConnectNo, axis, 0, MySpara);
    //Step 4: Set interpolation motion speed parameter of Controller 1
    ret = smc_set_vector_profile_unit(ConnectNo2, MyCrd, 0, MyMax_Vel, MyTacc, MyTacc, 0);
    ret = smc_set_vector_s_profile(ConnectNo2, MyCrd, 0, MySpara);
    //Step 5: Start the fixed-length motion of Controller 0
    ret = smc_pmove_unit(ConnectNo, axis, dist, 0);
    //Step 6: Wait for the stop of fixed-length motion of controller
    while(smc_check_done(ConnectNo, axis) == 0);
    //Step 7: Start interpolation motion
    ret = smc_line_unit(ConnectNo2, MyCrd, MyaxisNum, AxisArray, pos, 0);
```

2.8 Control function of BASIC program

The BASIC program control function of Leadshine controller is designed to read, modify, run, stop

and debug the BASIC program stored in the controller.

Association function of BASIC program control:

Name	Function	Reference
smc_read_array	Read array values by index	Section 3.32.
smc_modify_array	Modify array values by index	
smc_read_var	Read variable value	
smc_modify_var	Modify variable value	
smc_get_stringtype	Read variable type	
smc_basic_run	Run	
smc_basic_stop	Stop	
smc_basic_pause	Pause	
smc_basic_step_run	Single-step run	
smc_basic_step_over	Run to the next breakpoint	
smc_basic_continue_run	Continue running	
smc_basic_state	Current state	
smc_basic_current_line	Current execution line	
smc_basic_break_info	Breakpoint information	
smc_basic_message	Read output information	
smc_basic_command	Online command	

2.9 Control function of G code program

The G code program control function of Leadshine controller is designed to control the G code program in controller, read, delete, run, stop and check the G code program in controller.

Association function of G code program control:

Name	Function	Reference
smc_gcode_check_file	Check whether the file exists	Section 3.33
smc_gcode_delete_file	Delete a file	
smc_gcode_clear_file	Delete all files	
smc_gcode_get_first_file	Read the first file name	
smC_gcode_get_next_file	Read the next file name	
smc_gcode_start	Start	
smc_gcode_stop	Stop	
smc_gcode_pause	Pause	
smc_gcode_state	Read current status	

smc_gcode_set_current_file	Set current file	
smc_gcode_get_current_file	Read current file name	
smc_gcode_current_line	Read the current running line	
smc_gcode_get_file_profile	Read attribute of G code operation file	

2.10 Bus control function

The Leadshine controller supports bus functions, including CANopen bus and EtherCAT bus. Bus control and pulse control are the same in most range, but there are some differences at the same time, which will be explained below in detail.

2.10.1 Enable motor

In bus mode, all axises should be enabled before start of motion, whether it is bus servo or bus stepping. Unlike the pulse controller which can open the servo enabling signal by enabling the motor, the axis of master controller is enabled by sending the set axis enable command, which is completed after the bus axis state machine (state_machine) becomes "operation enabled". The specific codes are as follows:

Note: The VC++ is used as example for the code in this part in order to better illustrate this function which is for reference only. Please use it according to the actual situation.

```
voidCDMCd1Dlg: OnButtonEnable()                    //Axis enable operation function.
{
   //TODO:Add your control notification handler code here
   time_tt1, t2;                          //Set the time monitoring variable to prevent endless loop while waiting for the change of axis state machine
   unsigned long errcode = 0;                          //Bus error code
   unsigned short statemachine = 0;                    //Bus state machine
   nmcs_get_errcode(m_nConnectNo, 2, &errcode);        //Acquire bus status
   if(errcode == 0)//Enable operation is not allowed until bus is normal
   {
      nmcs_set_axis_enable(m_nConnectNo, m_nAxis); // Set to enable the specified axis, where m_nAxis is the current specified axis.
      nmcs_get_axis_state_machine(m_nConnectNo, m_nAxis, &statemachine);     //Acquire axis state machine t1 = time (NULL);       //Set time
      while(statemachine! = 4)       //Monitor the value of axis state machine. The axis state machine is in a ready state if the value is equal to 4
         {
            t2 = time(NULL);
            if(t2-t1>3)     //3 seconds to prevent endless loop
            {
               GetDlgItem(IDC_STATIC_BSState)->SetWindowText ("Enable timeout, please check the device"); return;
            }
            nmcs_set_axis_enable(m_nConnectNo, m_nAxis); //Set axis enable.
            nmcs_get_axis_state_machine(m_nConnectNo, m_nAxis, &statemachine);     //Acquire state machine.
```

```
        }

    }
    else  //The bus has no response to the enable operation in abnormal state.
    {
        GetDlgItem(IDC_STATIC_BSState)->SetWindowText ("Bus error,  operation prohibited!") );
        return;
    }
}
```

2.10.2 Reset motor

The pulse controller has been fitted with a reset mode, which is very much different from that of bus controller. The reset is controlled by pulse controller (read limit signal of origin point, control mode, etc.) in the controller part. The control of reset by bus controller includes initiating reset and waiting for the completion of reset, and all the intermediate reset processes are handled by drive. The Leadshine controller supports 34 standard homing modes. Please refer to EtherCAT homing standard for details. It is illustrated by the following examples.

Note: The VC++ is used as example for the code in this part in order to better illustrate this function which is for reference only. Please use it according to the actual situation.

```
voidCDMCd2Dlg: OnZero()                    //Axis homing operation
{
  //TODO:Add your control notification handler code here
  UpdateData(true);                        // Refresh parameter
  short iret = 0;                          // Function return value is used to check whether the function
  is executed correctly
  unsigned short statemachine = 0;         // Symbol of axis state machine
  unsigned long errcode = 0;               //Bus error code
  iret = nmcs_get_errcode(m_nConnectNo, 2, &errcode);          //Acquire bus status
  if(errcode! = 0)                         // homing of bus error prevention
  {
  MessageBox (“Bus error” “Error”); return;
  }
  iret = nmcs_get_axis_state_machine(m_nConnectNo, m_nAxis, &statemachine);
  if(statemachine! = 4)                    // Axis state machine error, homing prevention
  {
    MessageBox (“Axis state machine error”; “Error”)
    return;
  }
  if(smc_check_done(m_nConnectNo, m_nAxis) == 0) // No operation when axis is in motion
      return;
  // Set homing parameters
  smc_set_home_profile_unit(m_nConnectNo,  m_nAxis,  m_nSpeedmin,  m_nSpeedmax,  m_nAcc,
  m_nDec);
  // Set homing mode
  iret = smc_set_homemode(m_nConnectNo, m_nAxis, m_nPositive, m_nLowspeed, m_nHome, 1);
  // Start homing motion
  iret = smc_home_move(m_nConnectNo, m_nAxis);
```

```
    //Judge current axis state
    while(smc_check_done(m_nConnectNo, m_nAxis) == 0)
    {
        AfxGetApp()->PumpMessage();
        GetDlgItem(IDC_BUTTON1)->EnableWindow(false);
    }
    WORD state = 0;
    iret = smc_get_home_result(m_nConnectNo, m_nAxis, &state);
    if(state == 1)    // The command is cleared after homing
    {
        iret = smc_set_position_unit(m_nConnectNo, m_nAxis, 0);
    }
    GetDlgItem(IDC_BUTTON1)->EnableWindow(true);
    UpdateData(false);
}
```

2.10.3 IO control and motor motion

There is no substantial difference between bus controller and pulse controller in IO control and axis motion mode, both of which are the same in terms of function controlling. Explaination is made below from such aspects as motion of constant length and continuous motion, reset position and deceleration stop by some examples.

```
// Execute motion of constant length and continuous motion
void CDMCd1Dlg: OnButtonDo()
{
    //TODO:Add your control notification handler code here
    UpdateData(true);     // Refresh parameter
    short iret = 0;
    unsigned short statemachine = 0;
    unsigned long errcode = 0;
    iret = nmcs_get_errcode(m_nConnectNo, 2, &errcode);
    if(errcode! = 0)
    {
        MessageBox ("Bus error", "Error")
        return;
    }
    iret = nmcs_get_axis_state_machine(m_nConnectNo, m_nAxis, &statemachine);
    if(statemachine! = 4)
    {
        MessageBox ("Axis state machine error"; "Error")
        return;
    }
    // Note: The part above is bus control mode, under which bus state and axis state machine need detecting;
    the following bus control modes are shared with pulse control modes
    if(smc_check_done(m_nConnectNo, m_nAxis) == 0)//  moving
    return;
    iret = smc_set_equiv(m_nConnectNo, m_nAxis, 1); //  Set pulse equivalent

    // Set pulse mode (the pulse mode here is fixed as P+D Direction: pulse + direction)
    iret = smc_set_pulse_outmode(m_nConnectNo, m_nAxis, 0);
    // Set uniaxial motion speed parameters
    smc_set_profile_unit(m_nConnectNo,  m_nAxis,  m_nSpeedMin,  m_nSpeed,  m_nAcc,  m_nDec,
```

```
    m_nSpeedStop);
    // Set time of S section
    iret = smc_set_s_profile(m_nConnectNo, m_nAxis, 0, m_nSPara);
    if(m_nActionst == 0)
    {
        iret = smc_pmove_unit(m_nConnectNo, m_nAxis, m_nPulse*(m_bLogic?1:-1), 0);
        // Relative constant length motion
    }
    else
    {
        iret = smc_vmove(m_nConnectNo, m_nAxis, m_bLogic?1:0);    // Constant motion
    }
        UpdateData(false);
}
// Execute command of position clearing; the pulse mode is consistent with bus mode
void CDMCd1Dlg: OnButtonClear()
{
    //TODO:Add your control notification handler code here
    for(inti = 0; i<4; i++)
    {
        smc_set_position_unit(m_nConnectNo, i, 0);                    // Command position is cleared
    }
}
// Execute deceleration stop time
void CDMCd1Dlg: OnButtonDecstop()
{
    //TODO:Add your control notification handler code here
    UpdateData(true);     // Refresh parameter
    smc_set_dec_stop_time(m_nConnectNo, m_nAxis, m_nDec);     // Set 10ms deceleration stop time
    smc_stop(m_nConnectNo, m_nAxis, 0);     // deceleration stop
}
```

2.10.4 Bus status

The bus controller is of master-slave structure, which is connected by network cables. Therefore it is necessary to scan bus status in real time in the program (make scanning using timer or independent task in general) so as to cope with any abnormal problem that may happen. The corresponding operations can only be performed when the bus status is normal; otherwise, take corresponding measures, such as stopping the current operation, prompting alarm, etc. As shown in the following example:

```
// Scan bus state
void CDMCd2Dlg:OnTimer(UINTn IDEvent)
{
    //TODO:Add your message handler code here and / or call default
    unsigned long TotalAxis = 0;                    // The number of axises connected on the bus
    unsigned long errcode = 0;                      // Bus state
    short iret = 0;
    iret = nmcs_get_errcode(m_nConnectNo, 2, &errcode);      //Acquire bus status
    CStringstring;
    if(errcode == 0)                                // Status 0 means bus is normal
```

```
    {
        nmcs_get_total_axes(m_nConnectNo, &TotalAxis);      // Acquire the number of axises on the
        current bus
        string.Format("Bus state: Number of normal connecting axises: %id"TotalAxis);
        GetDlgItem(IDC_STATIC_BUSState)->SetWindowText(string);
        m_nStatus = smc_check_done(m_nConnectNo, m_nAxis);     //Judge current axis state
        GetDlgItem(IDC_EDIT_STATUS)->SetWindowText(m_nStatus? “Static”: “Motion”);
        WORD state = 0;
        smc_get_home_result(m_nConnectNo, m_nAxis, &state); //  Acquire hominging motion results
        GetDlgItem(IDC_EDIT_STATUS2)->SetWindowText(state?“Complete”: “Uncompleted”);

GetDlgItem(IDC_EDIT_HomeLogicState)->SetWindowText(smc_read_org_pin(m_nConnectNo,m_nAxi
s)?"  High level ":"  Low level ");

GetDlgItem(IDC_EDIT_EZLogicState)->SetWindowText(smc_read_ez_pin(m_nConnectNo,m_nAxis)?"
High level ":"  Low level ");

        CStringstrpos;
        CStringstrspeed;
        smc_get_position_unit(m_nConnectNo, m_nAxis, &m_fUposition);      // Acquire current axis
        position
        smc_read_current_speed_unit(m_nConnectNo, m_nAxis, &m_NowSpe);       // Acquire current axis
        speed

        strpos.Format("%.3lf", m_fUposition);
        strspeed.Format("%.3f", m_NowSpe);

        GetDlgItem(IDC_EDIT_XPOSITION)->SetWindowText(strpos);
        GetDlgItem(IDC_EDIT_MyNowSpe)->SetWindowText(strspeed);
    }
    Else     // Stop the bus which is abnormal and then give out alarm prompt
    {
        smc_emg_stop(m_nConnectNo);                          // Stop all axises immediately
        string.Format( “Bus error:%lu”, errcode);
        GetDlgItem(IDC_STATIC_BUSState)->SetWindowText(string);
    }
    CDialog:OnTimer(nIDEvent);

}
```

The following error messages many appear while using the bus controller. Some of the error messages may disappear after powering on the controller again and initializing the bus; some error messages can be removed with functions.

S/N	Bus error	Causes and solutions
1	000e	Bus initialization; it will automatically disappear if initialization is normal; check the line, if any fault is reported constantly.
2	001e	Add or lose slave station; rescanning required

3	0009	The controller should reconnect the first slave station after losing connection
4		

```
// Clear bus error operation
void CDMCd2Dlg:OnButtonBusrst()
{
	//TODO:Add your control notification handler code here
	unsigned long errcode = 0;
	nmcs_get_errcode(m_nConnectNo, 2, &errcode);
	if(errcode! = 0)
	{
		nmcs_clear_errcode(m_nConnectNo, 0);		// Clear bus error
	}
	else
	{
		MessageBox("Bus normal", "Error");
		return;
	}
}
```

Chapter 3 Function list

3.1 Communication connection function

short smc_board_init(WORD ConnectNo, WORD type, char *pconnectstring, DWORD baud)

Function: The controller links initialization functions to allocate system resources

Parameter: ConnectNo Designated link No. (0-7), default value 0

Type Link type: 1- serial port, 2-Ethernet port

Pconnectstring The link string corresponds to the IP address of the controller or the corresponding COM port

Baud Baud rate, default value:115200

Return value: 0: Link suceeds, Non-zero: link failure error code

Application scope: Full series of controllers

⚠Note: The data bit must be 8 bits when using the API function dynamic library. Use it under the default parameters of stop bit and parity bit. smc_board_init_ex. Function is available, if any parameter is changed.

Example 1: Initialize the Ethernet port and serial port

```
short iret = smc_board_init(0, 2, "192.168.5.11", 0)		// Initialize the Ethernet port
short iret = smc_board_init(0, 1, "COM1", 115200)		// Initialize the serial port
```

short smc_board_init_ex(WORD ConnectNo, WORD type, char* pconnectstring, DWORD dwBaudRate, DWORD dwByteSize, DWORD dwParity, DWORD dwStopBits)

Function: The high-level link initialization function of controller, which is used to allocate system resources

Parameter:	ConnectNo	Designated link No. (0-7), default value 0
	Type	Link type: 1- serial port, 2-Ethernet port
	Pconnectstring	The link string corresponds to the IP address of the controller or the corresponding COM port
	dwBaudRate	Baud rate, default value:115200
	dwByteSize	8: Data: 8
	dwParity	Check bit, 0: No check, 1: Odd parity check, 2: Even parity check
	dwStopBits	Stop bit, 1: Stop bit 1, 2: Stop bit 2

Return value: 0: Link suceeds, non-zero Link failed; error code:

Application scope: Full series of controllers

⚠Note: The data bit must be 8 bits when using the API function dynamic library. It is used when the stop bit and parity bit is modified by the smc_set_com function.

short smc_board_close(WORD ConnectNo)

Function: The controller closes the function to release system resources

Parameter: ConnectNo Designed link No. (0-7),

Return value: Error code

Application scope: Full series of controllers

short smc_set_connect_timeout(DWORD timems);

Function: Network connection timeout

Parameter: timems time-out period, (ms); the default is 5 s if the timeout is equal to 0 or the function has not been called

Return value: Error code

Application scope: Full series of controllers

short smc_get_release_version(WORD ConnectNo, char* ReleaseVersion)

Function: Read the release version No.

Parameter: ConnectNo Designated link No. (0-7), default value 0

ReleaseVersion Return the release version No. of the controller

Return value: Error code

Application scope: Full series of controllers

short smc_get_card_version(WORD ConnectNo, DWORD* CardVersion)

Function: Obtain the hardware version of the controller

Parameter: ConnectNo Designated link No. 0-7, default value 0

CardVersion Return the hardware version No. of controller

Return value: Application scope of error code: Full series of controllers

short smc_get_card_soft_version(WORD ConnectNo, DWORD* FirmID, DWORD* SubFirmID)

Function: Obtain the controller firmware version

Parameter: ConnectNo Designated link No. 0-7, default value 0

FirmID Return to the controller firmware type

SubFirmID Return the firmware version No. of controller

Return value: Error code

Application scope: Full series of controllers

⚠Note: The type of FirmID is subject to hexadecimal system

short smc_get_card_lib_version(DWORD* LibVer)

Function: Obtain the version No. of the controller dynamic library file

Parameter: LibVer Return the library version No.

Return value: Error code

Application scope: Full series of controllers

short smc_get_total_axes(WORD ConnectNo, DWORD* TotalAxis)

Function: Obtain the number of axises for the current controller

Parameter: ConnectNo Designated link No. 0-7, default value 0

TotalAxis Return the number of controller axises at present

Return value: Error code

Application scope: Full series of controllers

short smc_set_debug_mode(WORD mode, const char* FileName)

Function: Function calls print out settings

Parameter: mode Prints the enable status, 0: Disabled; 1: Enabled

FileName File saving path:

Parameter file name + suffix: Relative path

Complete description of parameter file path + file name suffix: Absolute path

Return value: Error code

Application scope: Full series of controllers

⚠Note: Calling can be monitored after printout is enabled. When user calls function, it will output relevant information, which will also be saved in the designated file path.

short smc_get_debug_mode(WORD mode, char* FileName)

Function: Read function calling and set printout

Parameter: mode Return printout enabling status

FileName Return file saving path

Return value: Error code

Application scope: Full series of controllers

short smc_set_ipaddr(WORD ConnectNo, const char* IpAddr)

Function: Set a new IP address for the controller

Parameter: ConnectNo Designated link No. 0-7, default value 0

IpAddr Character string of new IP address, such as “192.168.5.11”

Return value: Error code

Application scope: Full series of controllers

short smc_get_ipaddr(WORD ConnectNo, char* IpAddr)

Function: Read IP address of the controller

Parameter: ConnectNo Designated link No. 0-7, default value 0

IpAddr Return character string of IP address, such as “192.168.5.11”

Return value: Error code

Application scope: Full series of controllers

short smc_set_com(WORD ConnectNo, WORD com, DWORD dwBaudRate, WORD wByteSize, WORD wParity, WORD wStopBits);

Function: Set COM port parameters of the controller

Parameter: ConnectNo Designated link No. 0-7, default value 0

Com Com port: 1-RS232 and 2-RS485

dwBaudRate Baud rate, such as 9600, 1200, 115200, etc. 115200

wByteSize Data bits: 7 and 8. Default value: 8.

wParity Check bit: 0- No check, 1- Odd parity check, 2-Even parity check

wStopBits Stop bit: 1and 2

Return value: Error code

Application scope: Full series of controllers

⚠Note: The data bit must be 8 bits when using the API function dynamic library.

short smc_get_com(WORD ConnectNo, WORD com, DWORD* dwBaudRate, WORD* wByteSize, WORD* wParity, WORD* dwStopBits)

Function: Read COM port parameters of the controller

Parameter: ConnectNo Designated link No. 0-7, default value 0

Com Com port: 1-RS232 and 2-RS485

dwBaudRate Return Baud rate, such as 9600, 19200, etc. 115200 等。

wByteSize Return data bits: 7 and 8

wParity Return Check bit: 0- No check, 1- Odd parity check, 2-Even parity check

wStopBits Return stop bits: 1 and 2

Return value: Error code

Application scope: Full series of controllers

3.2 Pulse mode

short smc_set_pulse_outmode(WORD ConnectNo, WORD axis, WORD outmode)

Function: Set pulse output mode of designated axis

Parameter: ConnectNo Designated link No. 0-7, default value 0

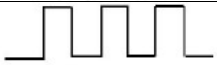
axis Designated axis No., value range: 0- Max. number of axises of the controller -1

outmode Select pulse output mode. The values are shown in Table 3.1

Return value: Error code

Application scope: Pulse type full series controller

Table 3.1 Pulse output mode of command

Output mode of pulse	Positive direction pulse		Negative direction pulse	
	PULSE output end	DIR output end	PULSE output end	DIR output end
0		High level		Low level
1		High level		Low level
2		Low level		High level
3		Low level		High level
4		High level	High level	
5		Low level	Low level	
6	PULSE output end DIR output end	 	PULSE output end DIR output end	 

⚠Note: 1. Call smc_set_pulse_outmode to set the controller pulse output mode according to the pulse receiving mode of the driver before calling the motion function (such as smcjmove) to output pulses

2. 300 and 600 series output of AB phase supported

short smc_get_pulse_outmode(WORD ConnectNo, WORD axis, WORD* outmode)

Function: Read the pulse output mode settings of the designated axis

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

outmode Return pulse output mode

Return value: Error code

Application scope: Pulse type full series controller

3.3 Pulse equivalent

short smc_set_equiv(WORD ConnectNo, WORD axis, double equiv)

Function: Set the value of pulse equivalent

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

equiv Pulse equivalent (pulse/unit)

Return value: Error code

Application scope: Full series of controllers

⚠ Note: 1) This function applies to advanced motion functions (include motion of point , interpolation and continuous interpolation)

2) Set the pulse current value of each motion axis before using advanced motion function to move. The value cannot be 0

short smc_get_equiv(WORD ConnectNo, WORD axis, double* equiv)

Function: Set the return to pulse current value

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

equiv Set the return to pulse current value

Return value: Error code

Application scope: Full series of controllers

3.4 Backlash setting

short smc_set_backlash_unit(WORD ConnectNo, WORD axis, double backlash)

Function: Set reverse clearance value

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

backlash Reverse clearance value, unit: unit

Return value: Error code

Application scope: Pulse type full series controller

short smc_get_backlash_unit(WORD ConnectNo, WORD axis, double * backlash)

Function: Read setting of reverse clearance value

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

backlash Set the value of return to reverse clearance

Return value: Error code

Application scope: Pulse type full series controller

3.5 Status monitoring function

short smc_check_done(WORD ConnectNo, WORD axis)

Function: Detect the motion status of designated axis

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Return value: 0: Designated axis is running, 1: The designated axis has been stopped

Application scope: Full series of controllers

⚠Note: The function is applied to uniaxial and PVT motion

short smc_check_done_multicoor(WORD ConnectNo, WORD Crd)

Function: Detect the motion state of the coordinate system

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Specify the frame number of the controller (Value range: 0~1)

Return value: Status of coordinate system, 0:operating, 1: Normal stop

Application scope: Full series of controllers

⚠Note: This function applies to interpolation motion

DWORD smc_axis_io_enable_status(WORD ConnectNo, WORD axis)

Function: Read the enable status of the designated axis special signal

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Return value: Refer to Table 3.2. bit 0 means forbidding, bit 1 means permit

Application scope: Pulse type full series controller

⚠ Note: The function can be used to read whether the special signal of axis is under enabling status. Its return value is decimal. After converting to base 2, check the value of each bit. The usage is similar to command smc_axis_io_status.

DWORD smc_axis_io_status(WORD ConnectNo, WORD axis)

Function: Read the state of motion signal about the designated axis

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Return value: Refer to Table 3.2

Application scope: Full series of controllers

Table 3.2 The signal status of axis motion

Bit number	Signal name	Description
0	ALM	1: Servo alarm signal ALM is ON; 0: OFF
1	EL+	1: Positive hard limit signal +EL is ON; 0: OFF
2	EL-	1: Negative hard limit signal -EL is ON; 0: OFF
3	EMG	1: Emergency-stop signal EMG is ON; 0: OFF
4	ORG	1: Origin signal ORG is ON; 0: OFF
6	SL+	1: Positive soft limit signal +SL is ON; 0: OFF
7	SL-	1: Negative software limit signal-SL is ON; 0: OFF
8	INP	1: Servo position signal INP is ON; 0: OFF
9	EZ	1: EZ signal is ON; 0: OFF
10	RDY reserve	1: Servo ready signal RDY is ON (SMC100 controller);
11	DSTP	1: deceleration stop signal DSTP is ON (SMC100 Controller only)
Other bits reserved	Reserved	

short smc_get_axis_run_mode(WORD ConnectNo, WORD axis, WORD* run_mode)

Function: Read the axis motion mode

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

run_mode Return to motion mode:

0: Free

1: Pmove

2: Vmove

3: Hmove

4: Handwheel

5: Ptt/Pts

6: Pvt/Pvts

7: Gear

8: Cam

9: Line

10: Continue

Return value: Error code

Application scope: Full series of controllers

short smc_set_position_unit(WORD ConnectNo, WORD axis, double pos)

Function: Set the value of current command position counter

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Pos Positional value, unit: unit

Return value: Error code

Application scope: Full series of controllers

short smc_get_position_unit(WORD ConnectNo, WORD axis, double* pos)

Function: Read the value of current command position counter

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

pos Return to current positional value, unit: unit

Return value: Error code

Application scope: Full series of controllers

short smc_read_current_speed_unit(WORD ConnectNo, WORD axis, double* current_speed)

Function: Read current axis speed

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

current_speed Return the speed value , unit: unit/s. Read-back speed. The read value is marked with a sign. Positive indicates positive motion while negative indicates negative motion

Return value: Application scope of error code: Full series of controllers

⚠Note: The vector speed is read using this function in course of interpolation and continuous interpolation motion.

short smc_get_stop_reason(WORD ConnectNo, WORD axis, long *StopReason)

Function: Read axis stop cause

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

StopReason Stop cause:

0: Normal stop

1: ALM immediately stop, IMD_STOP_AT_ALM

2: ALM deceleration stop, DEC_STOP_AT_ALM

3: LTC external triggering stops immediately, IMD__STOP_AT_LTC

4: EMG immediately stop, IMD__STOP_AT_EMG

5: Positive hard limit bit stops immediately, IMD__STOP_AT_ELP

6: Negative hard limit bit stops immediately, IMD__STOP_AT_ELN

7: Positive hard limit bit decelerates to stop, DEC_STOP_AT_ELP

8: Negative hard limit bit decelerates to stop, DEC_STOP_AT_ELN

9: Positive soft limit bit stops immediately, IMD__STOP_AT_SOFT_ELP

10: Negative soft limit bit stops immediately, IMD__STOP_AT_SOFT_ELN

11: Positive soft limit bit decelerates to stop, DEC_STOP_AT_SOFT_ELP

12: Negative soft limit bit decelerates to stop, DEC_STOP_AT_SOFT_ELN

13: Command stops immediately, IMD__STOP_AT_CMD

14: Command decelerates to stop, DEC_STOP_AT_CMD

15: Immediate stop for other reasons, IMD__STOP_AT_OTHER

16: Deceleration stop for other reasons, DEC_STOP_AT_OTHER

17: Immediate stop for unknown reasons, IMD__STOP_AT_UNKOWN

18: Deceleration stop for unknown reasons

DEC_STOP_AT_UNKOWN

19: External IO deceleration stop DEC_STOP_AT_DEC

Return value: Error code

Application scope: Full series of controllers

short smc_clear_stop_reason(WORD ConnectNo, WORD axis)

Function: Clear the stop reason of axis

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Return value: Error code

Application scope: Pulse type full series controller

short smc_get_target_position_unit(WORD ConnectNo, WORD axis, double* pos)

Function: Read the current target position

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Return value: Positional value, unit: unit

Application scope: Full series of controllers

short smc_set_workpos_unit(WORD ConnectNo, WORD axis, double pos)

Function: Set the current workpiece origin

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Pos Set positional value, unit: unit

Return value: Error code

Application scope: Full series of controllers

short smc_get_workpos_unit(WORD ConnectNo, WORD axis, double* pos)

Function: Read the origin of current workpiece

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the

controller -1

Pos Return to set the positional value, unit: unit

Return value: Error code

Application scope: Full series of controllers

3.6 Inching function

short smc_set_profile_unit(WORD ConnectNo, WORD axis, double Min_Vel, double Max_Vel, double Tacc, double Tdec, double Stop_Vel)

Function: Set uniaxial speed curve (time mode)

Parameter: ConnectNo Designated link No. 0-7, default value 0

Return value: Application scope of error code: Full series of controllers

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Min_Vel Starting speed, unit: unit/s

Max_Vel max. speed, unit: unit/s

Tacc Acceleration time, unit: s

Tdec Deceleration time, unit: s

Stop_Vel Stop speed, unit: unit/s

⚠Note: Since the max. pulse output frequency of the motion controller is 2MHz, the product of the max. speed set and the pulse equivalent set value must be less than 2MHz.

short smc_set_profile_unit_acc(WORD ConnectNo, WORD axis, double Min_Vel, double Max_Vel, double acc, double dec, double Stop_Vel)

Function: Set uniaxial motion speed curve (acceleration mode)

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Min_Vel Starting speed, unit: unit/s

Max_Vel max. speed, unit: unit/s

acc Acceleration, unit: unit/s^2

dec Deceleration, unit: unit/s^2

Stop_Vel Stop speed, unit: unit/s

Return value: Error code

Application scope: Full series of controllers

⚠Note: Since the max. pulse output frequency of the motion controller is 2MHz, the product of the max. speed set and the pulse equivalent set value must be less than 2MHz.

short smc_get_profile_unit(WORD ConnectNo, WORD axis, double* Min_Vel, double* Max_Vel, double* Tacc, double* Tdec, double* Stop_Vel)

Function: Read uniaxial motion speed curve

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	axis	Designated axis No., value range: 0- Max. number of axises of the controller -1
	Min_Vel	Return to the starting speed setting
	Max_Vel	Return to the max. speed setting
	Tacc	Return to acceleration time setting
	Tdec	Return to deceleration time setting
	Stop_Vel	Return to stop speed setting

Return value: Error code

Application scope: Full series of controllers

short smc_set_s_profile(WORD ConnectNo, WORD axis, WORD s_mode, double s_para)

Function: Set the parameter value of S section of the uniaxial speed curve

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	axis	Designated axis No., value range: 0- Max. number of axises of the controller -1
	s_mode	Reserved parameter, fixed value is 0
	s_para	Time of S section, unit: s; rang: 0~1 s

Return value: Error code

Application scope: Pulse type full series controller

short smc_get_s_profile(WORD ConnectNo, WORD axis, WORD s_mode, double* s_para)

Function: Read the parameter value of S section of the uniaxial speed curve

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	axis	Designated axis No., value range: 0- Max. number of axises of the controller -1
	s_mode	Reserve parameter 0

s_para Return to set the time of S section, unit: s;

Return value: Error code

Application scope: Pulse type full series controller

short smc_pmove_unit(WORD ConnectNo, WORD axis, double Dist, WORD posi_mode)

Function: Motion of constant length

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Dist Target position, unit: unit

posi_mode Motion mode, 0: Relative coordinate mode, 1: Absolute coordinate mode

Return value: Error code

Application scope: Full series of controllers

short smc_vmove(WORD ConnectNo, WORD axis, WORD dir)

Function: Continuous motion of designated axis

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

dir Motion direction, 0: Negative direction, 1: Positive direction

Return value: Error code

Application scope: Full series of controllers

short smc_reset_target_position_unit(WORD ConnectNo, WORD axis, double New_Pos)

Function: Change the current target position of the designated axis online

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

New_Pos New target position, unit: unit

Return value: Error code

⚠Note: 1) The function is applied to the displacement in the motion state of axis PMOVE motion only.

2) Parameter New_Pos is an absolute positional value, no matter whether the current motion is absolute or relative coordinate mode.

Application scope: Full series of controllers

short smc_update_target_position_unit(WORD ConnectNo, WORD axis, double New_Pos)

Function: Change the current target position of the designated axis compulsorily

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

New_Pos New target position, unit: unit

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: 1) This function is applied to the displacement of axis in PM0VE or free motion state.

2) Parameter New_Pos is an absolute positional value, no matter whether the current motion is absolute or relative coordinate mode.

short smc_change_speed_unit(WORD ConnectNo, WORD axis, double New_Vel, double Taccdec)

Function: Change the current motion speed of the designated axis online

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

New_Vel New operation speed, unit: unit/s

Taccdec Variable speed time, unit: s

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: 1) The function is applied to variable speed in uniaxial motion.

2) The time of variable speed is from the current speed to a new one. At that time, the controller will recalculate the time from the starting speed to the max. speed, and from the max. speed to stop, i.e., the acceleration and deceleration time can be recalculated.

3) Once the variable speed is confirmed, the default operation speed will be changed into New_Vel. The acceleration and deceleration time will also be covered by the controller's new calculated value, i.e. when reading back smc_get_profile_unit to speed parameter, its value may differ from the setting value of smc_set_profile_unit.

4) Variable speed value can only be positive under point bit motion and the direction of motion is separate from the variable speed value.

5) In constant speed motion, the variable speed value is related to its running direction. If the starting operating direction is positive, the motion will be the negative direction. If the variable speed is positive, the motion direction will remain unchanged, and the speed will be subject to the setting variable speed value. If the starting direction is negative, when the speed is positive, the motion will move toward positive direction; if the variable speed is negative, the direction of motion will remain unchanged and speed will be subject to to the setting variable speed value

3.7 Homing action function

short smc_set_home_pin_logic(WORD ConnectNo, WORD axis, WORD org_logic, double filter)

Function: Set ORG origin signal

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

org_logic Effective level of ORG signal, 0: low effective, 1: high effective

filter Reserved parameter, fixed value is 0

Return value: Error code

Application scope: Pulse type full series controller

⚠Note: If the axis number of this function is 255, set homing signal parameters of all axises.

short smc_get_home_pin_logic(WORD ConnectNo, WORD axis, WORD* org_logic, double filter)

Function: Read ORG origin signal setting

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

org_logic Return to the effective level of ORG signal

filter Reserved parameter

Return value: Error code

Application scope: Pulse type full series controller

short smc_set_homemode(WORD ConnectNo, WORD axis, WORD home_dir, double vel_mode, WORD mode, WORD source)

Function: Set homing mode

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

home_dir Homing origin direction, 0: Negative, 1: Positive

vel_mode Homing origin speed mode, default value: 1

Mode Homing origin mode:

①The bus-type controller homing mode is set by referring to drive

②The homing modes supported by SMC100 series controllers are as follows (SMC102 only supports Mode 0,1 and 2 only) :

0: Primary homing, that is, the platform moves toward the direction of the origin sensor at a high speed, and the motor stops immediately when triggering the origin sensor.

1: Primary homing and search, which means the platform moves towards the origin sensor at a high speed, and the motor reverses at a low speed when triggering the origin sensor; the motor immediately stops after exiting the trigger area of the origin sensor.

2: Secondary homing, which means the platform moves toward the direction of the origin sensor at a high speed, and the motor reverses at a low speed when triggering the origin sensor; it moves to the origin sensor at a low speed again after exiting the trigger area of the origin sensor; the motor stops immediately when triggering the origin sensor.

3: After primary homing, record an EZ pulse for homing, which means the platform moves toward the direction of the origin sensor at a high speed. The motor continues to move forward at a low speed when triggering the origin sensor. The motor immediately stops when triggering the EZ signal on the encoder.

4: Record an EZ pulse for homing, and the number of EZ signal triggers is 1. The motor moves forward at a low speed. The motor stops immediately when triggering the EZ signal on the encoder.

5: After primary homing, record a reverse EZ pulse for homing, which means the platform moves toward the direction of the origin sensor at a high speed. The motor moves backward at a low

speed, when triggering the origin sensor. The motor stops immediately when triggering the EZ signal on the encoder. (Reservation)

③The homing mode supported by the SMC300, SMC600 series is as follows:

0: Primary homing, which means the platform moves toward the direction of the origin sensor at a high speed. The motor stops immediately when triggering the origin sensor.

1: Primary homing and search, which means the platform moves toward the direction of the origin sensor at a high speed, and the motor reverses at a low speed when triggering the origin sensor. The motor stops immediately after exiting the triggering area of the origin sensor.

2: Secondary homing, which means the platform moves toward the direction of the origin sensor at a high speed, and the motor reverses at a low speed when triggering the origin sensor; it moves to the origin sensor at a low speed again after exiting the triggering area of the origin sensor; the motor stops immediately when triggering the origin sensor.

3: Primary homing and EZ homing method, which means the platform moves toward the direction of the origin sensor at a high speed. The motor continues to move forward at a low speed when triggering the origin sensor. The motor stops immediately when triggering the EZ signal on the encoder.

4: EZ alone homing. The trigger times of EZ signal is 1. The motor moves forward at a low speed, and the motor stops immediately when triggering the EZ signal on the encoder.

5: Primary homing and search to EZ. In the motion process of homing under such process, deceleration will stop when searching the origin signal, and then use the reverse speed to search EZ to take effect reversely. Motor will stop at this time.

6: Origin latch. The motor returns to the origin at a set speed. Latch the current position and the motor will slow down at the same time when triggering the edge of the origin switch. After completing the deceleration stop of the motor, search the latch position reversely and move to the position, and then the motor stops.

7: Origin latch and the EZ latch in the same direction. This mode first executes the origin latch homing in mode 3, and then continues to run along the homing direction to the EZ signal generation. When generating the EZ signal, latch the current position and execute the deceleration stop. After the deceleration stop of the motor, search reversely to the EZ latch position. The motor will stop after moving to the latch position.

8: Record an EZ latch alone. In the process of homing, detect effective edge of EZ, latch the current position, and execute deceleration stop. After the motor deceleration stops, search the latch position of EZ reversely, and the motor stops after moving to the latch position.

9: Origin latch and reverse EZ latch. This mode first executes the origin latch homing in mode

3, and then runs to the EZ signal generation in the opposite direction of the set homing direction. When generating the EZ signal, latch the current position and execute the deceleration stop. After the deceleration stop of the motor, search the latch position of EZ reversely, and move to the latch position. The motor stops the Source homing count source, 0: command position counter, 1: encoder counter.

Return value: Error code

Application scope: Full series of controllers

⚠Note: When the homing mode is equal to 4, fix the homing speed mode as the low-speed homing.

short smc_get_homemode(WORD ConnectNo, WORD axis, WORD* home_dir, double* vel, WORD* mode, WORD* source)

Function: Read homing mode

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	axis	Designated axis No., value range: 0- Max. number of axises of the controller -1
	home_dir	Return to homing direction
	vel	Return to origin speed mode, default value: 1
	mode	Return to the origin point mode
	Source	Return to zero count source, 0: Instruction position counter, 1: Encoder counter

Return value: Error code

Application scope: Full series of controllers

short smc_set_ez_count(WORD CardNo, WORD axis, WORD Count)

Function: Set the number of EZ

Parameter:	CardNo	Control card number
	axis	Designated axis No., value range: 0- Max. number of axises of the controller -1
	Count	Set EZ number

Return value: Error code

Scope of application: SMC300 and SMC600 series.

short smc_get_ez_count(WORD CardNo, WORD axis, WORD* Count)

Function: Set the number of EZ

Parameter: CardNo Control card number

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Count Number of returned EZ

Return value: Error code

Scope of application: SMC300 and SMC600 series.

short smc_set_home_position_unit(WORD ConnectNo, WORD axis, WORD enable, double position);

Function: Set the offset position value after homing

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

enable Enabling parameters

0: Disabled.

1: Clear 0 first, and then move to the specified position (relative position).

2: Move to the specified position (relative position) firstly, and then clear 0.

position Set homing position

Return value: Error code

Scope of application: SMC300 and SMC600 series.

⚠Note: The running speed and acceleration/deceleration time of offset position are the set values during fixed-length motion.

short smc_get_home_position_unit(WORD ConnectNo, WORD axis, WORD* enable, double* position);

Function: Read the offset position value after homing

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

enable Enabling parameters

0: Disabled.

1: Clear 0 first, and then move to the specified position (relative

position).

2: Move to the designated position firstly, and then clear 0.

position Read the set value of the origin point position

Return value: Error code

Scope of application: SMC300 and SMC600 series.

short smc_set_home_profile_unit(WORD ConnectNo, WORD axis, double Low_Vel, double High_Vel, double Tacc, double Tdec)

Function: Set speed parameter of homing

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Low_Vel Set the starting speed of homing

High_Vel Set the running speed of homing

Tacc Set the homing acceleration and deceleration time, unit: s.

Tdec Reserved value 0.

Return value: Error code

Application scope: Full series of controllers

short smc_get_home_profile_unit(WORD ConnectNo, WORD axis, double* Low_Vel, double* High_Vel, double* Tacc, double* Tdec);

Function: Read the homing parameter

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Low_Vel Read the starting speed of the origin point

High_Vel Read the running speed if the original point

Tacc Read the homing acceleration and deceleration time, unit: s.

Tdec Reserved value 0.

Return value: Error code

Application scope: Full series of controllers

short smc_set_el_home(WORD ConnectNo, WORD axis, WORD mode)

Function: Switch function when limit position is original point

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

mode Switching mode: 0- no switching, 1- positive limit as origin point, 2- negative limit as origin point

Return value: Error code

Application scope: Pulse type full series controller

short smc_home_move(WORD ConnectNo, WORD axis)

Function: Homing

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Return value: Error code

Application scope: Full series of controllers

short smc_get_home_result(WORD ConnectNo, WORD axis, WORD* state);

Function: Read the motion state of origin point

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

state Homing motion state, 0: Unfinished, 1: Finished

Return value: Error code

Application scope: Full series of controllers

3.8 PVT motion function

short smc_ptt_table_unit(WORD ConnectNo, WORD axis, DWORD count, double* pTime, double* pPos)

Function: Transfer data to the specified data sheet by means of PTT mode

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

count Number of data points, each data sheet has 1000 storage spaces, and each data point occupies 1 storage space.

pTime	Data point time array, unit: s (Precision: ms);
pPos	Data point position array, unit: unit;

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: (1) The position and time in the first group of downloaded data (the starting point) must be 0; the data in the array are based on the data at the starting point.

(2) All the data should be transferred at one time when calling this function to transfer data to the data sheet, for the original data in the data sheet will be deleted. Do not update the data sheet if the axis of the data sheet is moving.

short smc_pts_table_unit(WORD ConnectNo, WORD axis, DWORD count, double* pTime, double* pPos, double* pPercent)

Function: Transfer data to specified data sheet in PTS mode.

Parameter: ConnectNo	Designated link No. 0-7, default value 0
axis	Designated axis No., value range: 0- Max. number of axises of the controller -1
count	Number of data points, each data sheet has 1000 storage spaces, and each data point occupies 1 storage space.
pTime	Data point time array, unit: s (Precision: ms);
pPos	Data point position array, unit: unit;
pPercent	Data point percentage array, range of percentage: [0, 100];

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: (1) The position and time in the first group of downloaded data (i.e. starting point) must be 0; The data in the array are based on the data at the starting point.

(2) All the data should be transferred at one time when calling this function to transfer data to the data sheet, for the original data in the data sheet will be deleted. It is forbidden to update the data sheet if the axis using the data sheet is moving.

short smc_pvt_table_unit(WORD ConnectNo, WORD axis, DWORD count, double* pTime, double* pPos, double* pVel)

Function: Transfer data to specified data sheet in PVT mode.

Parameter: ConnectNo	Designated link No. 0-7, default value 0

	axis	Designated axis No., value range: 0- Max. number of axises of the controller -1
	count	Number of data points, each data sheet has 5000 storage spaces, and each data point occupies 1 storage space.
	pTime	Data point time array, unit: s (Precision: ms);
	pPos	Data point position array, unit: unit;
	pVel	Data point speed array, unit/s;

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: (1) The position, time and speed in the first group of downloaded data (i.e. starting point) must be 0; The data in the array are based on the data at the starting point

(2) All the data should be transferred at one time when calling this function to transfer data to the data sheet, for the original data in the data sheet will be deleted. It is forbidden to update the data sheet if the axis using the data sheet is moving.

short smc_pvts_table_unit(WORD ConnectNo, WORD axis, DWORD count, double* pTime, double* pPos, double velBegin, double velEnd)

Function: Transfer data to specified data sheet in PVTS mode

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	axis	Designated axis No., value range: 0- Max. number of axises of the controller -1
	count	Number of data points, each data sheet has 5000 storage spaces, and each data point occupies 1 storage space.
	pTime	Data point time array, unit: s (Precision: ms);
	pPos	Data point position array, unit: unit;
	velBegin	Set the speed of the first point, unit: unit/s
	velEnd	Set the speed of the last point; unit: unit/s

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: (1) The position, time and speed in the first group of downloaded data (i.e. starting point) must be 0; The data in the array are based on the data at the starting point

(2) All the data should be transferred at one time when calling this function to transfer data to the data sheet, for the original data in the data sheet will be deleted. It is forbidden to

update the data sheet if the axis using the data sheet is moving.

short smc_pvt_move(WORD ConnectNo, WORD AxisNum, WORD* AxisList)

Function: Start PVT motion

Parameter: ConnectNo Designated link No. 0-7, default value 0

AxisNum Axis number, value range: 1- Max. axis number of controllers

AxisList List of axes, array

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

3.9 Function of interpolation motion parameter

short smc_set_vector_profile_unit(WORD ConnectNo, WORD Crd, double Min_Vel, double Max_Vel, double Tacc, double Tdec, double Stop_Vel)

Function: Set interpolation motion speed parameter (time mode).

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

Min_Vel Starting speed, unit: unit/s

Max_Vel max. speed, unit: unit/s

Tacc Acceleration time, unit: s

Tdec Deceleration time, unit: s

Stop_Vel Stop speed, unit: unit/s

Return value: Error code

Application scope: Pulse type full series controller

short smc_set_vector_profile_unit_acc(WORD ConnectNo, WORD Crd, double Min_Vel, double Max_Vel, double acc, double dec, double Stop_Vel)

Function: Set interpolation motion speed parameter (acceleration mode)

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

Min_Vel Starting speed, unit: unit/s

Max_Vel max. speed, unit: unit/s

acc Acceleration time, unit: unit/s~2.

dec Deceleration time, unit: unit/s~2.

Stop_Vel Stop speed, unit: unit/s

Return value: Error code

Application scope: Pulse type full series controller

short smc_get_vector_profile_unit(WORD ConnectNo, WORD Crd, double* Min_Vel, double* Max_Vel, double* Tacc, double* Tdec, double* Stop_Vel)

Function: Read interpolation motion speed parameter

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	Crd	Coordinate system number, value range: 0~1
	Min_Vel	Return the starting speed value; unit: unit/s
	Max_Vel	Return the max. speed value; unit: unit/s
	Tacc	Return the acceleration time value, unit: s
	Tdec	Return the deceleration time value, unit: s
	Stop_Vel	Return the stop speed value, unit: unit/s

Return value: Error code

Application scope: Pulse type full series controller

short smc_set_vector_s_profile(WORD ConnectNo, WORD Crd, WORD s_mode, double s_para)

Function: Set the smoothing time of interpolation velocity curve

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	Crd	Coordinate system number, value range: 0~1
	s_mode	Reserved parameter, fixed value is 0
	s_para	Smoothing time, unit: s, range: 0~1

Return value: Error code

Scope of application: Controllers other than SMC100 series

short smc_get_vector_s_profile(WORD ConnectNo, WORD Crd, WORD s_mode, double* s_para) function: read the set smoothing time of interpolation velocity curve

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	Crd	Coordinate system number, value range: 0~1
	s_mode	Reserved parameter, fixed value is 0
	s_para	Back to smoothing time setting

Return value: Error code

Scope of application: Controllers other than SMC100 series

short smc_set_vector_decstop_time(WORD ConnectNo, WORD Crd, double time)

Function: Set the time parameter when interpolation has abnormal deceleration and stop

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

time Deceleration stop time, unit: s.

Return value: Error code

Scope of application: Controllers other than SMC100 series

short smc_get_vector_decstop_time(WORD ConnectNo, WORD Crd, double* time)

Function: Read the interpolation abnormal deceleration stop time parameter

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

time Return the deceleration stop time, unit: s.

Return value: Error code

Scope of application: Controllers other than SMC100 series

short smc_set_arc_limit(WORD ConnectNo, WORD Crd, WORD Enable, double MaxCenAcc, double MaxArcError)

Function: Set the speed limit function of circular interpolation

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

Enable Enabling parameters, 0: unlimited speed, 1: circular speed limit

MaxCenAcc Reserve parameter 0

MaxArcError Reserve parameter 0

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: Arc speed limit only applies to continuous interpolation mode 1 and 2.

short smc_get_arc_limit(WORD ConnectNo, WORD Crd, WORD* Enable, double* MaxCenAcc, double* MaxArcError);

Function: Read the speed limit function of arc interpolation

Parameter: ConnectNo Designated link No. 0-7, default value 0

	Crd	Coordinate system number, value range: 0~1
	Enable	Return the enabling parameters, 0: Unlimited speed, 1: Arc speed limit
	MaxCenAcc	Reserve parameter 0
	MaxArcError	Reserve parameter 0

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

3.10 Function of single-section interpolation motion

short smc_line_unit(WORD ConnectNo, WORD Crd, WORD AxisNum, WORD* AxisList, double* Target_Pos, WORD posi_mode)

Function: Linear interpolation motion

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	Crd	Coordinate system number, value range: 0~1
	AxisNum	Motion axis number, value range: 2~ max. axis number of controllers
	AxisList	Axis number list
	Target_Pos	List of target locations, unit: unit.
	posi_mode	Motion mode, 0: Relative coordinate mode, 1: Absolute coordinate mode

Return value: Error code

Application scope: Pulse type full series controller

short smc_arc_move_center_unit(WORD ConnectNo, WORD crd, WORD AxisNum, WORD* AxisList, double* Target_Pos, double* Cen_Pos, WORD Arc_Dir, long Circle, WORD posi_mode)

Function: Spiral interpolation motion extended based on circle center + arc end point mode (it can be used for two-axis arc interpolation)

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	Crd	Coordinate system number, value range: 0~1
	AxisNum	Motion axis number, value range: 2~ max. axis number of controllers
	AxisList	Axis number list
	Target_Pos	Target location array, unit: unit.
	Cen_Pos	Circle center position array, unit: unit.
	Arc_Dir	Arc direction, 0: Clockwise, 1: Counterclockwise.
	Circle	Number of turns:

Negative number: Concentric interpolation is performed at this time. The absolute value 1 of this value plus means the number of concentric circles. For example, -1 means 2 circles of concentric interpolation, -2 means 3 circles of concentric interpolation …

Natural number: It means spiral interpolation is performed at this time. The value represents the turn number of spirals. For example, 0 means 0 spiral interpolation, and 1 means 1 spiral interpolation …

posi_mode Motion mode, 0: Relative coordinate mode, 1: Absolute coordinate mode

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: 1) When axis number is 2, the first two axes of the axis list are interpolated by plane spiral or concentric circles.

2) When the axis number is 3 and the motion track is spiral interpolation, the plane of the first two axes in axis list is regarded as the base plane to perform plane spiral interpolation. Meanwhile, the third axis of the axis list moves to a specified height, and the difference between the end position and the starting position of axis is the height of the spiral line segment relative to the base plane.
3) When the axis number is higher than 3 and motion track is spiral interpolation, the first three axes are listed for spiral interpolation, while the subsequent axes have linear follow-up motion, and the motion time equals to that of the first three axes.
4) When the motion track is spiral interpolation:

It is a blooming spiral when the half distance from the starting point to the end point is greater than the distance from the starting point to the circle center on the base plane composed of the first two axises in axis list.

It is a convergent spiral when the half distance from the starting point to the end point is less than the distance from the starting point to the circle center on the base plane composed of the first two axes of axis list.

It is circular interpolation (it is cylindrical spiral if the number of interpolation axes is 3) when half of the distance from the starting point to the end point is equal to the distance from the starting point to the circle center on the base plane composed of the first two

axes of the list

short smc_arc_move_radius_unit(WORD ConnectNo, WORD crd, WORD AxisNum, WORD* AxisList, double* Target_Pos, double Arc_Radius, WORD Arc_Dir, long Circle, WORD posi_mode)

Function: Spiral interpolation motion extended by radius + arc end point mode (it can be used for two-axis arc interpolation)

Parameter:		
	ConnectNo	Designated link No. 0-7, default value 0
	Crd	Coordinate system number, value range: 0~1
	AxisNum	Motion axis number, value range: 2~ max. axis number of controllers
	AxisList	Axis number list
	Target_Pos	Target location array, unit: unit.
	Arc_Radius	Circular arc value, unit: unit:
	Arc_Dir	Arc direction, 0: Clockwise, 1: Counterclockwise.
	Circle	Number of turns, value range: higher than or equal to 0. The value means the number of turns of spiral. For example, 0 means 0 spiral interpolation, and 1 means 1 spiral interpolation …
	posi_mode	Motion mode, 0: Relative coordinate mode, 1: Absolute coordinate mode

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: 1) When the axis number is 2, the first two axes of axis list should be interpolated by plane arc.

2) When the axis number is 3, the plane of the first two axes in the axis list is regarded as the base plane to carry out plane arc interpolation; meanwhile, the third axis of axis list moves to the specified height; the difference between the end position and the starting position of the axis is the height of the cylindrical spiral line segment relative to the base plane.
3) When the axis number is greater than 3, the first three axes in axis list are provided with cylindrical spiral interpolation, while the subsequent axes have linear follow-up motion, and the motion time is equal to that of the first three axes.

short smc_arc_move_3points_unit(WORD ConnectNo, WORD Crd, WORD AxisNum, WORD* AxisList, double* Target_Pos, double* Mid_Pos, long Circle, WORD posi_mode)

Function: Spiral interpolation motion extended by three-point circular arc mode (it can be used for spatial circular arc interpolation of two axes and three axes).

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

AxisNum Motion axis number, value range: 2~ max. axis number of controllers

AxisList Axis number list

Target_Pos Target location array, unit: unit.

Mid_Pos Intermediate position array, unit: unit.

Circle Number of turns:

Negative number: It means that spatial arc interpolation is performed at this time.

The absolute value of this value minus 1 means the turn number of space arc. For example, -1 means 0 space arcs, and -2 means 1 space arc …

Natural number: It means that cylindrical spiral interpolation is performed at this time.

The value means the number of turns of spiral. For example, 0 means 0 spiral interpolation, and 1 means 1 spiral interpolation …

posi_mode Motion mode, 0: Relative coordinate mode, 1: Absolute coordinate mode

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: 1) When the axis number is 2, the first two axes of axis list should be interpolated by plane arc.

2) When the axis number is 3 and the motion track is cylindrical spiral interpolation, the plane of the first two axes in axis list is regarded as the base plane to perform the plane arc interpolation. Meantime, the third axis of axis list moves to the specified height; the difference between the end and starting position of axis is the height of the cylindrical spiral line segment relative to the base plane.

3) When the axis number is greater than 3, cylindrical spiral interpolation or spatial circular arc interpolation is performed on the first three axes of axis list, while the subsequent axes perform linear follow-up motion, and the motion time is equal to that of the first three

axes.

3.11 Function of continuous interpolation motion

short smc_conti_set_lookahead_mode(WORD ConnectNo, WORD Crd, WORD mode, long LookaheadSegment, double PathError, double LookaheadAcc)

Function: Set interpolation mode and small segment preview parameters

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

Mode Interpolation mode: 0- lookbehind mode 0, 1-lookahead mode 1, 2-lookbehind mode 2

LookaheadSegment Lookahead segment number, i.e. the number of segments calculated internally during each running.

PathError Track error, unit: unit:

LookaheadAcc Turning acceleration, unit/s~2

Return value: Error code

Scope of application: SMC600 series controllers.

⚠Note: The number of lookahead segments, track error and turning acceleration are effective only when the enabling mode is lookahead motion.

short smc_conti_get_lookahead_mode(WORD ConnectNo, WORD Crd, WORD* mode, DWORD *LookaheadSegment, double* PathError, double* LookaheadAcc)

Function: Read interpolation mode and lookahead parameters of small segments

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

Mode Return interpolation mode: 0- lookbehind mode 0, 1-lookahead mode 1, 2-lookbehind mode 2.

LookaheadSegment Return the number of lookahead segments, i.e. the number of segments calculated internally during each running.

PathError Return track error

LookaheadAcc Return the turning acceleration; unit: unit/s~2.

Return value: Error code

Scope of application: SMC600 series controllers.

short smc_conti_set_blend(WORD ConnectNo, WORD Crd, WORD enable)

Function: Set the enabling status of Blend corner transition mode of continuous interpolation

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

enable Enable status, 0: disabled, 1: enabled.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note:

1) It works when interpolation mode is 0, and it is set by smc_conti_set_lookahead_mode.
2) When Blend corner smooth transition is enabled, the corners between the motion tracks will have smooth transition, thus obtaining a smoother speed curve.
3) When the computer executes this instruction, the function settings will be stored in the buffer zone, which will take effect when the next motion function of this function starts to move.
4) When corner smoothing is enabled, the corners will have smooth transition in the subsequent motion process unless the function is called again to prohibit corner smooth transition.

short smc_conti_get_blend(WORD ConnectNo, WORD Crd, WORD* enable)

Function: Read the enabling state setting of continuous interpolation Blend corner transition mode.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

enable Read the enable status settings, 0: disabled, 1: enabled.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_conti_open_list(WORD ConnectNo, WORD Crd, WORD AxisNum, WORD* AxisList)

Function: Open buffer zone of continuous interpolation

Parameter: ConnectNo Specified link number: 0-7.

Crd Coordinate system number, value range: 0~1

AxisNum Motion axis number, value range: 2~ max. axis number of controllers

AxisList Axis number list:

AxisList[0]: AxisX

AxisList[1]: AxisY

AxisList[2]: AxisZ

AxisList[3]: AxisU

AxisList[4]: AxisV

AxisList[5]: AxisW

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: 1) A maximum of 5, 000 instructions can be cached in continuous buffer zone.

2) XYZ is the driving axis and UVW is the auxiliary axis when executing circular arc or spiral motion. The main axis performs circular arc or spiral motion, while the auxiliary axis does not perform circular arc or spiral motion, but performs linear motion along with the main axis.

3) It will enter the continuous interpolation mode after opening the continuous interpolation buffer; at this time, the moving axis participating in the continuous interpolation can exit the continuous interpolation mode unless the instruction in the buffer is executed or the instruction smc_conti_stop_list is called.

short smc_conti_start_list(WORD ConnectNo, WORD Crd)

Function: Start continuous interpolation

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_conti_close_list(WORD ConnectNo, WORD Crd)

Function: Close the continuous interpolation buffer

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

Return value: Error code

short smc_conti_pause_list(WORD ConnectNo, WORD Crd)

Function: Pause continuous interpolation

Parameter: ConnectNo Specified link number: 0-7.

Crd Coordinate system number, value range: 0~1

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠ Note: After the continuous interpolation is suspended, the continuous interpolation motion will decelerate and stop. When smc_conti_start_list command is called again, the motion controller will continue to run the unfinished continuous interpolation track.

short smc_conti_stop_list(WORD ConnectNo, WORD Crd, WORD stop_mode)

Function: Stop continuous interpolation.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

stop_mode Stop mode, 0: deceleration stop, 1: immediate stop.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: When the continuous interpolation motion is being executed, this instruction can stop the continuous interpolation motion and make the moving axis which participates in the continuous interpolation to exit the continuous interpolation mode.

short smc_conti_change_speed_ratio(WORD ConnectNo, WORD Crd, double Percent)

Function: Adjust the speed ratio of continuous interpolation dynamically

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

Percent Speed ratio, value range: 0~2.0

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: When the computer executes this instruction, the motion controller will adjust the continuous interpolation speed ratio of the next track.

short smc_conti_delay(WORD ConnectNo, WORD Crd, double delay_time, long mark)

Function: Pause delay instruction in continuous interpolation

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

delay_time Delay time, unit: s

mark Label, designated freely, 0 means automatic numbering.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: 1) The delay time refers to the waiting time when the motion stops.

2) When the delay time is set as 0, the delay time will be infinitely long.

short smc_conti_line_unit(WORD ConnectNo, WORD Crd, WORD AxisNum, WORD* AxisList, double* Target_Pos, WORD posi_mode, long mark)

Function: Straight line interpolation instruction in continuous interpolation.

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	Crd	Coordinate system number, value range: 0~1
	AxisNum	Motion axis number, value range: 2- max. axis number of controllers
	AxisList	Axis number list
	Target_Pos	Target location array, unit: unit.
	posi_mode	Motion mode, 0: Relative coordinate mode, 1: Absolute coordinate mode
	mark	Label, designated freely, 0 means automatic numbering.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_conti_arc_move_center_unit(WORD ConnectNo, WORD crd, WORD AxisNum, WORD* AxisList, double* Target_Pos, double* Cen_Pos, WORD Arc_Dir, long Circle, WORD posi_mode, long mark)

Function: Spiral interpolation instruction based on circle center + end point arc extension in continuous interpolation (it can be used for two-axis arc interpolation).

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	Crd	Coordinate system number, value range: 0~1
	AxisNum	Axis number, value range: 2- max. axis number of controllers
	AxisList	Axis number list
	Target_Pos	Target location array, unit: unit.
	Cen_Pos	Circle center position array, unit: unit.
	Arc_Dir	Arc direction, 0: Clockwise, 1: Counterclockwise.
	Circle	Number of turns: Number of turns, value range: greater than or equal to 0. It means that spiral interpolation is performed at this time.

	The value means the number of turns of spiral. For example, 0 means 0 spiral interpolation, and 1 means 1 spiral interpolation …
posi_mode	Motion mode, 0: Relative coordinate mode, 1: Absolute coordinate mode
mark	Label, designated freely, 0 means automatic numbering.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: 1) The first three axes of the list must be a combination of XYZ axes; see the description of smc_conti_open_list function for the definition of Axis XYZUVW.

2) When the axis number is 2, the first two axes in the axis list are interpolated by plane spiral or concentric circles.

3) When the axis number is 3 and the motion track is spiral interpolation, the plane of the first two axes in the axis list is regarded as the base plane to carry out plane spiral interpolation. Meanwhile, the third axis of the axis list moves to a specified height, and the difference between the end and the starting position of the axis is the height of the spiral segment relative to the base plane.

4) When the axis number is greater than 3 and the motion track is spiral interpolation, the driving axis performs spiral interpolation and auxiliary axis moves linearly along with the driving axis, and the motion time is equal to the total motion time of driving axis. See the description of smc_conti_open_list function for the corresponding definitions of driving axis and auxiliary axis.

5) When the motion track is spiral interpolation:

It is a blooming spiral when the half distance from the starting point to the end point is greater than the distance from the starting point to the circle center on the base plane composed of the first two axises in axis list.

It is a convergent spiral when the half distance from the starting point to the end point is less than the distance from the starting point to the circle center on the base plane composed of the first two axes of axis list.

It is circular interpolation (it is cylindrical spiral if the number of interpolation axes is 3) when half of the distance from the starting point to the end point is equal to the distance from the starting point to the circle center on the base plane composed of the first two axes of the list

short smc_conti_arc_move_radius_unit(WORD ConnectNo, WORD crd, WORD AxisNum, WORD* AxisList, double* Target_Pos, double Arc_Radius, WORD Arc_Dir, long Circle, WORD posi_mode, long mark)

Function: Cylindrical spiral interpolation instruction based on radius + end point arc extended in continuous interpolation (it can be used for two-axis arc interpolation).

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	Crd	Coordinate system number, value range: 0~1
	AxisNum	Motion axis number, value range: 2- max. axis number of controllers
	AxisList	Axis number list
	Target_Pos	Target location array, unit: unit.
	Arc_Radius	Circular arc value, unit: unit:
	Arc_Dir	Arc direction, 0: Clockwise, 1: Counterclockwise.
	Circle	Number of turns, value range: higher than or equal to 0. The value means the number of turns of spiral. For example, 0 means 0 spiral interpolation, and 1 means 1 spiral interpolation …
	posi_mode	Motion mode, 0: Relative coordinate mode, 1: Absolute coordinate mode
	mark	Label, designated freely, 0 means automatic numbering.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: 1) The first three axes of the list must be a combination of XYZ axes; see the description of smc_conti_open_list function for the definition of Axis XYZUVW.

2) When the axis number is 2, the first two axes of axis list should be interpolated by plane arc.

3) When the axis number is 3, the plane of the first two axes in the axis list is regarded as the base plane to carry out plane arc interpolation; meanwhile, the third axis of axis list moves to the specified height; the difference between the end position and the starting position of the axis is the height of the cylindrical spiral line segment relative to the base plane.

4) When the axis number is greater than 3, the driving axis performs cylindrical spiral interpolation, the auxiliary axis moves linearly along with the driving axis, and the motion time is equal to the total motion time of the driving axis. See the description of function smc_conti_open_list for the corresponding definitions of driving axis and

auxiliary axis.

short smc_conti_arc_move_3points_unit(WORD ConnectNo, WORD Crd, WORD AxisNum, WORD* AxisList, double* Target_Pos, double* Mid_Pos, long Circle, WORD posi_mode, long mark)

Function: The cylindrical spiral interpolation instruction based on three-point arc extended in continuous interpolation (it can be used for two-axis and three-axis arc interpolation).

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

AxisNum Motion axis number, value range: 2- max. axis number of controllers

AxisList Axis number list

Target_Pos Target location array, unit: unit.

Mid_Pos Intermediate position array, unit: unit.

Circle Number of turns

Negative number: It means that spatial arc interpolation is performed at this time.

The absolute value of this value minus 1 means the turn number of space arc. For example, -1 means 0 space arcs, and -2 means 1 space arc …

Natural number: It means that cylindrical spiral interpolation is performed at this time.

The value means the number of turns of spiral. For example, 0 means 0 spiral interpolation, and 1 means 1 spiral interpolation …

posi_mode Motion mode, 0: Relative coordinate mode, 1: Absolute coordinate mode

mark Label, designated freely, 0 means automatic numbering.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: 1) The first three axes of the list must be a combination of XYZ axes; see the description of smc_conti_open_list function for the definition of Axis XYZUVW.

2) When the axis number is 2, the first two axes of axis list should be interpolated by plane arc.

3) When the axis number is 3 and the motion track is cylindrical spiral interpolation, the plane of the first two axes in axis list is regarded as the base plane to perform the plane arc interpolation. Meantime, the third axis of axis list moves to the specified height; the difference between the end and starting position of axis is the height of the cylindrical spiral line segment relative to the base plane.

4) When the axis number is greater than 3, the driving axis performs cylindrical spiral interpolation or spatial circular arc interpolation, while the auxiliary axis follows the driving axis for linear motion, and the motion time is equal to the total motion time of the driving axis. See the description of function smc_conti_open_list for the corresponding definitions of driving axis and auxiliary axis.

short smc_conti_pmove_unit(WORD ConnectNo, WORD Crd, WORD axis, double dist, WORD posi_mode, WORD mode, long imark)

Function: Control and specify other axis motion instructions in continuous interpolation.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

axis Specify the axis number, range 0- the max. axis number of controller - 1.

dist Target position, unit: unit

posi_mode Motion mode, 0: Relative coordinate mode, 1: Absolute coordinate mode

mode Mode:

0: Pause the start (execute this fixed-length motion when the last interpolation motion in the buffer is finished; the interpolation motion in the next section will be executed when the fixed-length motion in this section is finished.)

1: Start directly (execute this fixed-length motion when the last interpolation motion in the buffer is finished, and execute the next interpolation motion at the same time).

mark Label, designated freely, 0 means automatic numbering.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: 1) This instruction can control the designated axis to conduct fixed-length motion in

continuous interpolation motion.

2) This axis cannot be a motion axis engaged in continuous interpolation.

3) The function smc_set_profile_unit must be used to set the running speed of the axis before using this command to control the motion of the axis.

3.12 Function of interpolation status continuous detection

long smc_conti_remain_space(WORD ConnectNo, WORD Crd)

Function: Inquiry the remaining interpolation space of continuous interpolation buffer zone.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

Return value: The remaining size of continuous interpolation buffer zone

Scope of application: SMC300 and SMC600 series of controllers

long smc_conti_read_current_mark(WORD ConnectNo, WORD Crd)

Function: Read the current interpolation segment number of the continuous interpolation buffer zone

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

Return value: Number of current continuous interpolation segment

Scope of application: SMC300 and SMC600 series of controllers

short smc_conti_get_run_state(WORD ConnectNo, WORD crd)

Function: Read the motion state of continuous interpolation

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

Return value: Motion state, 0: in motion, 1: Pause D, 2: Normal stop, 3: Not started, 4: Idle, 5: Abnormal deceleration stop.

Scope of application: SMC300 and SMC600 series of controllers

short smc_check_done_multicoor(WORD ConnectNo, WORD Crd)

Function: Detect the running status of continuous interpolation.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

Return value: motion state, 0: Running, 1: Stop

Scope of application: SMC300 and SMC600 series of controllers

3.13 IO control function of continuous interpolation

short smc_conti_set_pause_output(WORD ConnectNo, WORD crd, WORD action, long mask, long state)

Function: Set IO output status when continuous interpolation is suspended or abnormally stopped.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

0: Keep the original status

1: Output set IO status when the continuous interpolation is suspended, and the IO state before suspension is not restored when the operation is resumed.

2: Output the set IO state when the continuous interpolation is suspended, and resume the IO state before the suspension when the operation continues.

3: Output the set IO status when continuous interpolation is suspended or stopped, or other abnormal stops are encountered (such as EMG signal).

mask Select the output port mark: bit0~bit31 stands for OutO~Out31, which will be output when the bit value is 1, or not output when the bit value is 0.

state Output level status: bit0~bit31 represents Out0~Out31; it will output high level when the bit value is 1, or output low level when the bit value is 0

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

Description of activating Mode 3: 1) The motion controller outputs the set IO state when continuous interpolation is suspended, and resumes the IO state before suspension when running is resumed.

2) The motion controller outputs the set IO state when continuous interpolation is stopped or other abnormal stops are encountered, but the previous IO state will not be restored when continuous interpolation is started again.

short smc_conti_get_pause_output(WORD ConnectNo, WORD crd, WORD* action, long

***mask, long *state)**

Function: Read IO output status settings when continuous interpolation is suspended or abnormally stopped

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

action Return to activation status settings

mask Return to the setting of output selection mark

state Return to setting of the output level state

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_conti_wait_input(WORD ConnectNo, WORD Crd, WORD bitno, WORD on_off, double TimeOut, long mark)

Function: Wait for IO input in continuous interpolation.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

bitno Enter port number, value range is 0~31.

on_off Level status, 0: low level, 1: high level.

TimeOut Timeout, unit: s.

mark Label, designated freely, 0 means automatic numbering.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: 1) When the timeout is set as 0, the motion controller will wait for the IO input signal for an infinite time.

2) If the timeout is not 0, the next track will be run immediately once there is a corresponding IO response within the timeout. If there is no IO response, the program will automatically run the next track after the waiting time is longer than the timeout.

short smc_conti_delay_outbit_to_start(WORD ConnectNo, WORD Crd, WORD bitno, WORD on_off, double delay_value, WORD delay_mode, double ReverseTime)

Function: IO lagging output relative to the starting point of track segment in continuous interpolation (be executed in segment).

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd	Coordinate system number, value range: 0~1
bitno	Output port number, value range: 0~31.
on_off	Level status, 0: low level, 1: high level.
delay_value	Lag value; unit: s (lag time mode) or unit (lag distance mode).
delay_mode	Lag mode, 0: lag time, 1: lag distance.
ReverseTime	Delay flip time after level output, unit: s.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: 1) Set the IO operation, it will work in the instruction of the next track.

2) When the ReverseTime parameter is set as 0, the corresponding IO port level will not be reversed and the value will remain unchanged.

3) When the lag mode is selected as lag distance, the position source is the instruction position counter.

short smc_conti_delay_outbit_to_stop(WORD ConnectNo, WORD Crd, WORD bitno, WORD on_off, double delay_time, double ReverseTime)

Function: IO lagging output relative to the end point of track segment in continuous interpolation.

Parameter: ConnectNo	Designated link No. 0-7, default value 0
Crd	Coordinate system number, value range: 0~1
bitno	Output port number, value range: 0~31.
on_off	Level status, 0: low level, 1: high level.
delay_time	Lag time, unit: s
ReverseTime	Reserved parameter, fixed value is 0

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: 1) Set the IO operation, it will work after the end of the next track of the instruction.

2) The instruction will not be executed if the smc_conti_clear_IO_action function is used to clear the unfinished IO actions in the segment,

short smc_conti_ahead_outbit_to_stop (WORD ConnectNo, WORD Crd, WORD bitno, WORD on_off, double ahead_value, WORD ahead_mode, double ReverseTime)

Function: Output IO in advance relative to the end point of track segment in continuous interpolation (be executed in segment).

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	Crd	Coordinate system number, value range: 0~1
	bitno	Output port number, value range: 0~31.
	on_off	Level status, 0: low level, 1: high level.
	ahead_value	Advance value; unit: s (advance time mode) or unit (advance distance mode).
	ahead_mode	Advance mode, 0: advance time, 1: advance distance.
	ReverseTime	Delay flip time after level output, unit: s.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: 1) Set the IO operation, it will work in the instruction of the next track.

2) When the ReverseTime parameter is set as 0, the corresponding IO port level will not be reversed, and the set value will remain unchanged.

3) When the lag mode is selected as lag distance, the position source is the instruction position counter.

short smc_conti_write_outbit(WORD ConnectNo, WORD Crd, WORD bitno, WORD on_off, double ReverseTime)

Function: Continuous interpolation buffer zone has IO output immediately.

Parameter:	ConnectNo	Specified link number: 0-7.
	Crd	Coordinate system number, value range: 0~1
	bitno	Output port number, value range: 0~31.
	on_off	Level status, 0: low level, 1: high level.
	ReverseTime	Delay flip time after level output, unit: s.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: 1) When being executed by computer, this instruction will be stored in the buffer zone. When the last motion instruction in the buffer zone is executed, the instruction will be executed.

2) After this instruction is called, the velocity curves of the previous track and the next track will be discontinuous, which means, the Blend smoothing mode will not work between these two tracks.

3) When the ReverseTime parameter is set as 0, the corresponding IO port level will not

be reversed, and the set value will remain unchanged.

short smc_conti_clear_io_action(WORD ConnectNo, WORD Crd, DWORD IoMask)

Function: Clear the unfinished IO actions in the segment.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Coordinate system number, value range: 0~1

IoMask Clear mark: bit0~bit31 respectively represent Out0~Out31 output ports; Bit: 1: Clear the unfinished actions in the corresponding output port segment, such as IO flip action after the flip time reaches; 0: no operation.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠ Note: This function works on SMC _ conti _ delay _ outbit _ to _ start, smc_conti_ahead_outbit_to_stop and smc_conti_delay_outbit_to_stop instructions.

3.14 Immediate output function of PWM

short smc_set_pwm_output(WORD ConnectNo, WORD pwm_no, double fDuty, double fFre)

Function: Set PWM to output parameters immediately.

Parameter: ConnectNo Designated link No. 0-7, default value 0

pwm_no PWM channel, value range: 0~1.

fDuty Duty cycle, value range: 0~1.

fFre The frequency ranges from 1hz to 500khz.

Return value: Error code

Scope of application: Controllers other than SMC100 series controllers.

short smc_get_pwm_output(WORD ConnectNo, WORD pwm_no, double* fDuty, double* fFre)

Function: Read current output parameters of PWM.

Parameter: ConnectNo Designated link No. 0-7, default value 0

pwm_no PWM channel, value range: 0~1.

fDuty Return duty cycle setting.

fFre Return frequency setting value, and value range is 1HZ-500KHZ.

Return value: Error code

Scope of application: Controllers other than SMC100 series controllers.

3.15 Function of general IO interface

short smC_read_inbit(WORD ConnectNo, WORD bitno)

Function: Read the level of certain input port of the specified controller.

Parameter: ConnectNo Designated link No. 0-7, default value 0

bitno Input the port number in the range of 0~ the number of local input ports of controller -1.

Return value: Specified input port level: 0: low level, 1: high level.

Application scope: Full series of controllers

short smc_write_outbit(WORD ConnectNo, WORD bitno, WORD on_off)

Function: Set the level of certain output port of the designated controller.

Parameter: ConnectNo Designated link No. 0-7, default value 0

bitno Output port number, value range: 0~ number of local output ports of the controller -1.

on_off Output level, 0: low level, 1: high level.

Return value: Error code

Application scope: Full series of controllers

short smc_read_outbit(WORD ConnectNo, WORD bitno)

Function: Read the level of certain output port of the specified controller.

Parameter: ConnectNo Designated link No. 0-7, default value 0

bitno Output port number, value range: 0~ number of local output ports of the controller -1.

Return value: Specify the level of the output port, 0: low level, 1: high level.

Application scope: Full series of controllers

DWORD smc_read_inport(WORD ConnectNo, WORD portno)

Function: Read the level of all input ports of the specified controller.

Parameter: ConnectNo Designated link No. 0-7, default value 0

portno IO group number, value range: smc604a: 0 ~ 1, other controllers are 0.

Return value: Bit value of each IO port.

Application scope: Full series of controllers

⚠Note: The PORT number in the IO PORT is 0 within 32 bits, and the port number beyond 32 bits is 1. The return value is decimal, and the bit value after conversion to binary corresponds to the input state value of each port.

DWORD smc_read_outport(WORD ConnectNo, WORD portno)

Function: Read the level of all output ports of the designated controller.

Parameter: ConnectNo Designated link No. 0-7, default value 0

portno IO group number, value range: smc604a: 0 ~ 1, other controllers are 0.

Return value: Bit value of each IO port.

Application scope: Full series of controllers

⚠ Note: The PORT number in the IO PORT is 0 within 32 bits, and the port number beyond 32 bits is 1. The return value is decimal, and the bit value after conversion to binary corresponds to the output state value of each port.

short smc_write_outport(WORD ConnectNo, WORD portno, DWORD port_value)

Function: Set the level of all output ports of the designated controller.

Parameter: ConnectNo Designated link No. 0-7, default value 0

portno IO group number, value range: smc604a: 0 ~ 1, other controllers are 0.

port_value Enter the port level value

Return value: Error code

Application scope: Full series of controllers

short smc_reverse_outbit(WORD ConnectNo, WORD bitno, double reverse_time)

Function: IO output time delay flip.

Parameter: ConnectNo Designated link No. 0-7, default value 0

bitno Output port number, value range: 0- number of local output ports of the controller -1.

reverse_time Delayed flip time, unit: s.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: 1) The function works on 0UT0- the value of each controller port.

2) When the delay flip time parameter is set as 0, the delay flip time will be infinite.

short smc_set_io_count_mode(WORD ConnectNo, WORD bitno, WORD mode, double filter_time)

Function: Set IO counting mode;

Parameter: ConnectNo Designated link No. 0-7, default value 0

bitno Enter the port number, and the value range is 0- the number of local input ports of the controller -1.

mode IO counting mode, 0: disabled, 1: rising edge counting, 2: falling edge counting.

filter_time Filtering time, unit: s.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_get_io_count_mode(WORD ConnectNo, WORD bitno, WORD* mode, double* filter_time)

Function: Read setting of IO counting mode;

Parameter: ConnectNo Designated link No. 0-7, default value 0

bitno Enter the port number, and the value range is 0- the number of local input ports of the controller.

mode Return IO counting mode, 0: disabled, 1: rising edge counting, 2: falling edge counting

filter_time Return filtering time; unit: s

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_set_io_count_value(WORD ConnectNo, WORD bitno, DWORD CountValue)

Function: Reset IO count value

Parameter: ConnectNo Designated link No. 0-7, default value 0

bitno Enter the port number, and the value range is 0- the number of local input ports of the controller -1.

CountValue IO count value

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_get_io_count_value(WORD ConnectNo, WORD bitno, DWORD* CountValue)

Function: Read IO count value;

Parameter: ConnectNo Designated link No. 0-7, default value 0

bitno Enter the port number, and the value range is 0- the number of local input ports of the controller -1.

CountValue Return the IO count value

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

3.16 Function of specific IO interface

short smc_set_inp_mode(WORD ConnectNo, WORD axis, WORD enable, WORD inp_logic)

Function: Set INP signal of specified axis

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

enable Enable INP signal, 0: disabled, 1: enabled.

inp_logic The effective level of INP signal, 0: low effective, 1: high effective.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: (1) When the INP signal function is enabled, the corresponding axis can move only when the INP signal is in an effective state, otherwise, the state of the detection axis is running at this time (which means, the motion of the axis is restricted), and the INPIO port should be mapped first if there is no hardware interface INP.

(2) When the axis number of this function is set as 255, INP signal parameters of all axes are set.

short smc_get_inp_mode(WORD ConnectNo, WORD axis, WORD* enable, WORD* inp_logic)

Function: Read INP signal setting of specified axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

enable Return to INP signal enable state.

inp_logic Set the effective level of INP signal.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_set_alm_mode(WORD ConnectNo, WORD axis, WORD enable, WORD alm_logic, WORD alm_action)

Function: Set ALM signal of specified axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

enableALM Signal enabling, 0: disabled, 1: enabled.

alm_logicALM The active level of signal, 0: low effective, 1: high effective.

alm_actionALM Signal braking mode, 0: immediate stop (only this mode is supported).

Return value: Error code

Application scope: Pulse type full series controller

⚠Note: When the function axis number is set as 255, all parameters of axis alarm signal are set.

short smc_get_alm_mode(WORD ConnectNo, WORD axis, WORD* enable, WORD* alm_logic, WORD* alm_action)

Function: Read ALM signal setting of specified axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

enable Return ALM signal enabling status.

alm_logic Return set effective level of ALM signal

alm_action Return braking mode of ALM signal.

Return value: Error code

Application scope: Pulse type full series controller

short smc_write_sevon_pin(WORD ConnectNo, WORD axis, WORD on_off)

Function: Control the output of servo enable port of specified axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

on_off Set the level of servo enable port, 0: low level, 1: high level.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_read_sevon_pin(WORD ConnectNo, WORD axis)

Function: Read the level of servo enable port of specified axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Return value: Servo enable port level, 0: low level, 1: high level.

Scope of application: SMC300 and SMC600 series of controllers

short smc_write_erc_pin(WORD ConnectNo, WORD axis, WORD sel)

Function: Control ERC signal output of specified axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

sel Output level, 0: low level, 1: high level.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_read_erc_pin(WORD ConnectNo, WORD axis)

Function: Read ERC port level of specified axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Return value: ERC port level, 0: low level, 1: high level.

Scope of application: Pulse type SMC300 and SMC600 series controllers.

short smc_read_alarm_pin(WORD ConnectNo, WORD axis)

Function: Read the ALARM port level of the specified axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Specify the axis number, and the value range is 0- the max. axis number of the controller.

Return value: ALARM port level, 0: low level, 1: high level.

Scope of application: Pulse type SMC300 and SMC600 series controllers.

short smc_read_inp_pin(WORD ConnectNo, WORD axis)

Function: Read the INP port level of the specified axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Return value: INP port level, 0: low level, 1: high level.

Scope of application: Pulse type SMC300 and SMC600 series controllers.

short smc_read_org_pin(WORD ConnectNo, WORD axis)

Function: Read the ORG port level of the specified axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Return value: ORG port level, 0: low level, 1: high level.

Scope of application: Pulse type SMC300 and SMC600 series controllers.

short smc_read_elp_pin(WORD ConnectNo, WORD axis)

Function: Read the positive hard limit port level of the specified axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Return value: EL+ port level, 0: low level, 1: high level.

Scope of application: Pulse type SMC300 and SMC600 series controllers.

short smc_read_eln_pin(WORD ConnectNo, WORD axis)

Function: Read the negative hard limit port level of the specified axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Return value: EL- port level, 0: low level, 1: high level.

Scope of application: Pulse type SMC300 and SMC600 series controllers.

short smc_read_emg_pin(WORD ConnectNo, WORD axis)

Function: Read the emergency stop port level of the specified axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Return value: EMG port level, 0: low level, 1: high level.

Scope of application: Pulse type SMC300 and SMC600 series controllers.

3.17 Sub-cam instruction

short smc_cam_table_unit(WORD ConnectNo, WORD MasterAxisNo, WORD SlaveAxisNo, DWORD Count, double* pMasterPos, double* pSlavePos, WORD SrcMode)

Function: Add electronic cam table.

Parameters: MasterAxisNo Master axis number

SlaveAxisNo Slave axis number

SrcMode Master axis position mode: 0- command position, 1- feedback position.

Count Number of data

Return value: Error code

Scope of application: SMC300 and 600 series controllers.

short smc_cam_move(WORD ConnectNo, WORD AxisNo)

Function: Start the electronic cam motion of slave axis.

Parameters: SlaveAxisNo Slave axis number

Return value: Error code

Scope of application: SMC300 and 600 series controllers.

3.18 Wheel function

short smc_handwheel_set_index(WORD ConnectNo, WORD AxisSelIndex, WORD RatioSelIndex)

Function: Select or replace the handwheel motion axis selection and ratio gear.

Parameter: ConnectNo Fixed link number: 0-7, the default value is 0.

AxisSelIndex Select gear, and the value range is SMC606:0-5.

RatioSelIndex Gear, value range: 0, 1, 2, which respectively correspond to the ratio of 1, 10, 100.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠ Note: This instruction applies to gear selection before starting the handwheel and gear change during the handwheel motion. The gear selection of one axis can correspond to multiple motion axes (smc_handwheel_set_axislist). The ratio value of each motion axis in the same axis gear is the same, but the ratio value can be modified (smc_handwheel_set_ratiolist).

short smc_handwheel_get_index(WORD ConnectNo, WORD* AxisSelIndex, WORD* RatioSelIndex)

Function: Read the motion axis selection and ratio gear of handwheel.

Parameter: ConnectNo Designated link No. 0-7, default value 0

AxisSelIndex Return the axis to select the gear, value range is SMC606:0-5.

RatioSelIndex Gear, value range: 0, 1, 2, which respectively correspond to the ratio of 1, 10, 100.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_handwheel_set_axislist(WORD ConnectNo, WORD AxisSelIndex, WORD AxisNum, WORD* AxisList)

Function: Set the specific motion axis under the same axis gear.

Parameter: ConnectNo Fixed link number: 0-7, the default value is 0.

AxisSelIndex Select gear, and value range is 0- the max. number of axises of the controller -1.

AxisNum Motion axis number, value range: 1~ max. axis number of controller -1.

AxisList Axis number list:

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_handwheel_get_axislist(WORD ConnectNo, WORD AxisSelIndex, WORD* AxisNum, WORD* AxisList)

Function: Read the specific motion axis under the same axis gear.

Parameter: ConnectNo Designated link No. 0-7, default value 0

AxisSelIndex Select gear, and value range is 0- the max. number of axises of the controller -1.

AxisNum Motion axis number, value range: 1~ max. axis number of controller -1.

AxisList Return to the list of axis numbers:

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_handwheel_set_ratiolist(WORD ConnectNo, WORD AxisSelIndex, WORD StartRatioIndex, WORD RatioSelNum, double* RatioList)

Function: Set the lower wheel ratio gear selected on the same axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

AxisSelIndex Select the gear of the axis, the value range is 0- the max. number of axises of the controller -1.

StartRatioIndex Select the starting index value, the value range is 0-2.

RatioSelNum Number of ratio values, range: 1-3.

RatioList Ratio list, value range: [1, 100].

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: The default parameters of this instruction are StartRatioIndex = 0, RatioSelNum = 3, and there are three arrays ratiolinist [3] corresponding to 1, 10 and 100 respectively. Which means, all three ratio values can be available when the same axis is selected.

short smc_handwheel_get_ratiolist(WORD ConnectNo, WORD AxisSellndex, WORD StartRatioIndex, WORD RatioSelNum, double* RatioList)

Function: Read the ratio gear of handwheel and ratio value of corresponding axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

AxisSelIndex Select the gear of the axis, the value range is 0- the max. number of axises of the controller -1.

StartRatioIndex Select the starting index value, the value range is 0-2.

RatioSelNum Number of ratio values.

RatioList Return to the ratio list; value range is [1, 100].

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_handwheel_set_mode(WORD ConnectNo, WORD InMode, WORD IfHardEnable)

Function: Set the handwheel motion mode, whether it is in hardware or software mode.

Parameter: ConnectNo Designated link No. 0-7, default value 0

InMode Input pulse mode, 0: pulse + direction, 1: Phase AB pulse.

IfHardEnable Motion mode, 0: software control, 1: hardware control.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: It is necessary to configure the hardware connection lines on the controller hen moving in hardware mode, but it can be skipped in software mode.

short smc_handwheel_get_mode(WORD ConnectNo, WORD* InMode, WORD* IfHardEnable)

Function: Read the handwheel motion mode, and where motion is in hardware or software mode.

Parameter: ConnectNo Designated link No. 0-7, default value 0

InMode Return to the input pulse mode, 0: pulse + direction, 1: Phase AB pulse.

IfHardEnable Return operation mode, 0: software control, 1: hardware control.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_handwheel_move(WORD ConnectNo, WORD ForceMove);

Function: Start handwheel motion.

Parameter: ConnectNo Designated link No. 0-7, default value 0

ForceMove Parameter is reserved, fixed as 0.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_handwheel_stop(WORD ConnectNo)

Function: Stop handwheel motion.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

3.19 Encoder function

short smc_set_counter_inmode(WORD ConnectNo, WORD axis, WORD mode)

Function: Set the counting mode of encoder.

Parameter: ConnectNo Fixed link number: 0-7, the default value is 0.

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

mode Encoder counting mode:

0: Non-A/B phase (pulse/direction).

1: 1×A/B

2: 2×A/B

3: 4×A/B

Return value: Error code

Application scope: Full series of controllers

short smc_get_counter_inmode(WORD ConnectNo, WORD axis, WORD* mode)

Function: Read the counting mode of encoder.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Specify the axis number, and the value range is 0- the max. axis number of the controller.

mode Return the counting mode of the encoder.

Return value: Error code

Application scope: Full series of controllers

short smc_set_encoder_unit(WORD ConnectNo, WORD axis, double encoder_value)

Function: Set the pulse count value of encoder of specified axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

encoder_value Code counter value, unit: unit:

Return value: Error code

Application scope: Full series of controllers

short smc_get_encoder_unit(WORD ConnectNo, WORD axis, double* pos)

Function: Read the pulse count value of encoder of specified axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

pos Return the encoder position value, unit: unit.

Return value: Error code

Application scope: Full series of controllers

short smc_set_ez_mode(WORD ConnectNo, WORD axis, WORD ez_logic, WORD ez_mode, double filter)

Function: Set EZ signal level of the specified axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

ez_logic EZ Signal active level, 0: low effective, 1: high effective.

ez_mode Reserved parameter, fixed value is 0

filter Reserved parameter, fixed value is 0

Return value: Error code

Application scope: Full series of controllers

⚠Note: When the function axis number is set as 255, all axis EZ signal parameters are set.

short smc_get_ez_mode(WORD ConnectNo, WORD axis, WORD* ez_logic, WORD* ez_mode, double* filter)

Function: Read EZ signal level of specified axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

ez_logic Set the EZ signal effective level.

ez_mode Reserved parameter

Filter Reserved parameter

Return value: Error code

Application scope: Full series of controllers

short smc_set_counter_reverse(WORD ConnectNo, WORD axis, WORD reverse);

Function: Set reverse phase of Phase AB count.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

reverse 0-A: Count before B, and 1-A Count before b-ultrasound.

	Normal		Take Negative Value	
Encoder	**Increase counting**	**Decrease counting**	**Decrease counting**	**Increase counting**
Phase A				
Phase B				

Return value: Error code

Application scope: Full series of controllers

⚠Note: The function is under mode 0 by default. When the hardware connection line of the encoder is unchanged, we will change mode 0 to 1, which can make its encoding count negative (negative).

short smc_get_counter_reverse(WORD ConnectNo, WORD axis, WORD* reverse);

Function: Read reverse phase value of Phase AB count.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

reverse 0-A: Count before B, and 1-A Count before b-ultrasound.

Return value: Error code

Application scope: Full series of controllers

3.20 Latch function of high-speed position

short smc_set_ltc_mode(WORD ConnectNo, WORD axis, WORD ltc_logic, WORD ltc_mode, double filter)

Function: Set LTC signal of specified axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

	ltc_logic	Trigger mode of latch signal: 0: falling edge latch, 1: rising edge latch.
	ltc_mode	Reserved value 0.
	filter	Reserved parameter, fixed value is 0

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_get_ltc_mode(WORD ConnectNo, WORD axis, WORD* ltc_logic, WORD* ltc_mode, double filter)

Function: Read LTC signal setting of specified axis.

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	axis	Designated axis No., value range: 0- Max. number of axises of the controller -1
	ltc_logic	Return the trigger mode of LTC signal, 0: falling edge latch, 1: rising edge latch.
	ltc_mode	Reserve parameter 0
	filter	Reserved parameter

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_set_latch_mode(WORD ConnectNo, WORD axis, WORD ltc_mode, WORD latch_source, WORD triger_chunnel)

Function: Set the latch mode.

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	axis	Designated axis No., value range: 0- Max. number of axises of the controller -1
	ltc_mode	Latch mode: 0: single latch, 2: continuous latch.
	latch_source	Latch source, 0: instruction position counter, 1: encoder counter.
	triger_chunnel	Reserve parameter 0

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_get_latch_mode(WORD ConnectNo, WORD axis, WORD* all_enable, WORD* latch_source, WORD* triger_chunnel)

Function: Read latch mode.

Parameter: ConnectNo Fixed link number: 0-7, the default value is 0.

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

ltc_mode Return latch mode: 0: single latch, 2: continuous latch.

latch_source Return latch source, 0: instruction position counter, 1: encoder counter.

triger_chunnel Reserve parameter 0

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_get_latch_value_unit(WORD ConnectNo, WORD axis, double* pos)

Function: Read the value of latch from the controller.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

pos Return the latched value.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: When continuous latch is selected, the latch times will be reduced by 1 and the latch value will be eliminated after successfully reading the latch value with this function.

short smc_get_latch_flag(WORD ConnectNo, WORD axis)

Function: Read the latch times of the specified axis from the controller.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Return value: Number of valid latches.

Scope of application: SMC300 and SMC600 series of controllers

short smc_reset_latch_flag(WORD ConnectNo, WORD axis)

Function: Reset the mark bit of latch.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: Make sure to firstly call this function to reset the mark bit of the latch before using the latch function

3.21 Latch function of original point

short smc_set_homelatch_mode(WORD ConnectNo, WORD axis, WORD enable, WORD logic, WORD source)

Function: Set the origin point latch mode.

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	axis	Designated axis No., value range: 0- Max. number of axises of the controller -1
	enable	Origin point latch enabling, 0: disabled, 1: enabled.
	logic	Trigger mode, 0: falling edge, 1: rising edge.
	source	Source selection, 0: instruction position counter, 1: encoder counter.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_get_homelatch_mode(WORD ConnectNo, WORD axis, WORD* enable, WORD* logic, WORD* source)

Function: Read the mode setting of the origin point latch.

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	axis	Designated axis No., value range: 0- Max. number of axises of the controller -1
	enable	Return to the origin latch enabling state.
	logic	Return trigger mode.
	source	Return location source selection.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_reset_homelatch_flag(WORD ConnectNo, WORD axis)

Function: Clear the origin point latch mark.

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	axis	Designated axis No., value range: 0- Max. number of axises of the

controller -1

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

long smc_get_homelatch_flag(WORD ConnectNo, WORD axis)

Function: Read the origin point latch mark.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Specify the axis number, and the value range is 0- the max. axis number of the controller.

Return value: origin latch mark, 0: not latched, 1: latched.

Scope of application: SMC300 and SMC600 series of controllers

short smc_get_homelatch_value_unit(WORD ConnectNo, WORD axis, double* pos)

Function: Read origin point latch value.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

pos Return the position value.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

3.22 EZ latch function

short smc_set_ezlatch_mode(WORD ConnectNo, WORD axis, WORD enable, WORD logic, WORD source)

Function: Set EZ latch mode.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

enable Ez latch enabling, 0: disabled, 1: enabled.

logic Trigger mode, 0: falling edge, 1: rising edge.

source Source selection, 0: instruction position counter, 1: encoder counter.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_get_ezlatch_mode(WORD ConnectNo, WORD axis, WORD* enable, WORD* logic, WORD* source)

Function: Read mode setting of EZ latch.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

enable Return Ez latch enabling status.

logic Return trigger mode.

source Return location source selection.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_reset_ezlatch_flag(WORD ConnectNo, WORD axis)

Function: Clear the origin point latch mark.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

long smc_get_ezlatch_flag(WORD ConnectNo, WORD axis)

Function: Read EZ latch mark.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Specify the axis number, and the value range is 0- the max. axis number of the controller.

Return value: origin latch mark, 0: not latched, 1: latched.

Scope of application: SMC300 and SMC600 series of controllers

short smc_get_ezlatch_value_unit(WORD ConnectNo, WORD axis, double* pos)

Function: Read EZ latch value.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

pos Return the position value.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

3.23 Position comparison function

short smc_compare_set_config(WORD ConnectNo, WORD axis, WORD enable, WORD cmp_source)

Function: Set one-dimensional position comparator.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

enable Compare function status, 0: disabled, 1: enabled.

cmp_source Comparison source, 0: instruction position counter, 1: encoder counter.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_compare_get_config(WORD ConnectNo, WORD axis, WORD* enable, WORD* cmp_source)

Function: Read settings of one-dimensional position comparator.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

enable Return comparison function status.

cmp_source Return the comparison source.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_compare_clear_points(WORD ConnectNo, WORD axis)

Function: Clear all added comparison points of one-dimensional position.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_compare_add_point_unit(WORD ConnectNo, WORD axis, double pos, WORD dir, WORD action, DWORD actpara)

Function: Add comparison point of one-dimensional position.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

pos Comparison position, unit: unit:

dir Comparison mode, 0: less than or equal to 1: greater than or equal to.

action Compare the number of point trigger function, see Table 3.5.

actpara See table 3.5 for comparison of point trigger function parameters.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

Table 3.5 Parameter values of smc_compare_add_point_unit function action and actpara.

Action	actpara	Function
1	IO number	Set IO as high level
2	IO number	Set IO as low level
3	IO number	Take reverse IO
5	IO number	Output 500us pulse.
6	IO number	Output 1ms pulse.
7	IO number	Output 10ms pulse.
8	IO number	Output 100ms pulse.
13	Axis number	Stop the specified axis

short smc_compare_get_current_point_unit(WORD ConnectNo, WORD axis, double* pos)

Function: Read the current comparison point position.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

pos Return the position of current comparison point; unit: unit.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_compare_get_points_runned(WORD ConnectNo, WORD axis, long *pointNum);

Function: Read the number of compared points.

Parameter: ConnectNo Specified link number: 0-7.

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

pointNum Return the number of compared points.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_compare_get_points_remained(WORD ConnectNo, WORD axis, long *pointNum)

Function: Read the number of compared points.

Parameter: ConnectNo Specified link number: 0-7.

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

pointNum Return the current number of inserted points, with a max. number of 256.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_compare_set_config_extern(WORD ConnectNo, WORD enable, WORD cmp_source)

Function: Set the two-dimensional position comparator.

Parameter: ConnectNo Designated link No. 0-7, default value 0

enable Status of two-dimensional position comparison function, 0: disabled, 1: enabled.

cmp_source Two-dimensional position comparison source, 0: instruction position counter, 1: encoder counter.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_compare_get_config_extern(WORD ConnectNo, WORD* enable, WORD* cmp_source)

Function: Read the settings of two-dimensional position comparator.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Enable Return comparison function status.

cmp_source Return the comparison source.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_compare_clear_points_extern(WORD ConnectNo)

Function: Clear all added comparison points of two-dimensional position.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_compare_add_point_extern_unit(WORD ConnectNo, WORD* axis, double* pos, WORD* dir, WORD action, DWORD actpara)

Function: Add comparison point of two-dimensional position.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Specify the axis list (two axises) which will have position comparison on the controller.

pos List of position comparison of two-dimensional position unit: unit

dir Comparison mode list, 0: less than or equal to, 1: greater than or equal to.

action See table 3.6 for the trigger function numbers of two-dimensional position comparison points.

actpara See table 3.6 for trigger function parameters of two-dimensional position comparison points.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

Table 3.6 Parameter values of SMC _ compare _ add _ point _ extern _ unit function action and actpara.

Action	actpara	Function
1	IO number	Set IO as high level
2	IO number	Set IO as low level
3	IO number	Take reverse IO
5	IO number	Output 500us pulse.
6	IO number	Output 1ms pulse.
7	IO number	Output 10ms pulse.

8	IO number	Output 100ms pulse.
13	Axis number	Stop the specified axis

short smc_compare_get_current_point_extern_unit(WORD ConnectNo, double* pos)

Function: Read the position of comparison point of current two-dimensional position.

Parameter: ConnectNo Designated link No. 0-7, default value 0

pos Return the position of the comparison point of the current two-dimensional position in unit.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_compare_get_points_runned_extern(WORD ConnectNo, long *PointNum)

Function: Inquiry the number of compared two-dimensional comparison points.

Parameter: ConnectNo Designated link No. 0-7, default value 0

PointNum Return the comparison points of compared two-dimensional positions.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_compare_get_points_remained_extern(WORD ConnectNo, long *PointNum)

Function: Inquiry the number of two-dimensional comparison points which can be added.

Parameter: ConnectNo Designated link No. 0-7, default value 0

PointNum Return the number of two-dimensional position comparison points which can be added.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

3.24 Comparison function of high-speed position

short smc_hcmp_set_mode(WORD ConnectNo, WORD hcmp, WORD cmp_mode)

Function: Set high-speed comparison mode.

Parameter: ConnectNo Designated link No. 0-7, default value 0

hcmp High speed comparator, value range: 0-1 (corresponding to the last two output ports of hardware).

cmp_mode Comparison mode:

0: disabled (default).

1: equals.

2: less than.

3: greater than.

4: Queue, comparison space of 127 points are provided, first adding and first comparison, add comparison points after comparison is done, or add multiple comparison points at one time.

5: Linearity, it provides initial comparison points, position increment and comparison times.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: 1) When mode 1 is selected, the CMP port outputs an active level only when the current position is equal to the comparison position.

2) When mode 2 is selected, the CMP port will always maintain an active level if the current position is smaller than the comparison position.

3) When mode 3 is selected, the CMP port will always maintain an active level if the current position is greater than the comparison position.

4) When mode 4 or 5 is selected, the time for the CMP port to output the active level is set by the time parameter (pulse width) of smc_hcmp_set_config function.

short smc_hcmp_get_mode(WORD ConnectNo, WORD hcmp, WORD* cmp_mode)

Function: Read the setting of high-speed comparison mode.

Parameter: ConnectNo Designated link No. 0-7, default value 0

hcmp High speed comparator, value range: 0-1 (corresponding to the last two output ports of hardware).

cmp_mode Return comparison mode settings.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_hcmp_set_config(WORD ConnectNo, WORD hcmp, WORD axis, WORD cmp_source, WORD cmp_logic, long time)

Function: Configure high-speed comparator.

Parameter: ConnectNo Designated link No. 0-7, default value 0

hcmp High speed comparator, value range: 0-1 (corresponding to the last two

output ports of hardware).

axis Associate axis number, value range: 0- max. axis number of controller -1.

cmp_source Compare position source: 0: instruction position counter, 1: encoder counter.

cmp_logic Active level: 0: low level, 1: high level.

time Pulse width, unit: us, value range: 1us~20s.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠ Note: The time parameter (pulse width) of this function works only in queue and linear comparison mode.

short smc_hcmp_get_config(WORD ConnectNo, WORD hcmp, WORD* axis, WORD* cmp_source, WORD* cmp_logic, long *time)

Function: Read the configuration of high-speed comparator.

Parameter: ConnectNo Designated link No. 0-7, default value 0

hcmp High speed comparator, value range: 0-1 (corresponding to the last two output ports of hardware).

axis Return the setting of the associated axis number, value range is 0- the max. axis number of controller -1.

cmp_source Return settings of comparison position source.

cmp_logic Return setting of active level.

time Return setting of pulse width.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_hcmp_clear_points(WORD ConnectNo, WORD hcmp)

Function: Clear all added comparison points of high-speed position.

Parameter: ConnectNo Designated link No. 0-7, default value 0

hcmp High speed comparator, value range: 0-1 (corresponding to the last two output ports of hardware).

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_hcmp_add_point_unit(WORD ConnectNo, WORD hcmp, double cmp_pos)

Function: Add/update high-speed comparison position.

Parameter: ConnectNo Designated link No. 0-7, default value 0

hcmp High speed comparator, value range: 0-1 (corresponding to the last two output ports of hardware).

cmp_pos In queue mode: add comparison position; unit: unit.

In linear mode: update the starting comparison position, unit: unit.

Other modes: update the comparison position; unit: unit.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠ Note: When executing linear comparison, this instruction is the starting command for setting the starting position of comparison, as well as the starting command, which needs to be called and executed after the comparator is configured.

short smc_hcmp_set_liner_unit(WORD ConnectNo, WORD hcmp, double Increment, long Count)

Function: Set parameters of high-speed comparison linear mode.

Parameter: ConnectNo Designated link No. 0-7, default value 0

hcmp High speed comparator, value range: 0-1 (corresponding to the last two output ports of hardware).

Increment Increment value of position, unit: unit (positive value means increasing position, negative value means decreasing position).

Count Comparison times, value range: 1~65535.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_hcmp_get_liner_unit(WORD ConnectNo, WORD hcmp, double* Increment, long *Count)

Function: Read the parameter settings of high-speed comparison linear mode.

Parameter: ConnectNo Designated link No. 0-7, default value 0

hcmp High speed comparator, value range: 0-1 (corresponding to the last two output ports of hardware).

Increment Return the setting of position increment value.

Count	Return the setting of comparison times.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_hcmp_get_current_state(WORD ConnectNo, WORD hcmp, long *remained_points, double* current_point, long *runned_points)

Function: Read high-speed comparison status.

Parameter: ConnectNo	Designated link No. 0-7, default value 0
hcmp	High speed comparator, value range: 0-1 (corresponding to the last two output ports of hardware).
remained_points	Return the number of comparison points which can be added.
current_point	Return the position of current comparison point; unit; unit.
runned_points	Return the compared points.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_write_cmp_pin(WORD ConnectNo, WORD hcmp, WORD on_off)

Function: control the output of the specified CMP port.

Parameter: ConnectNo	Designated link No. 0-7, default value 0
hcmp	High speed comparator, value range: 0-1 (corresponding to the last two output ports of hardware).
on_off	Set CMP port level, 0: low level, 1: high level.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: This function only works when the high-speed comparison function is disabled.

short smc_read_cmp_pin(WORD ConnectNo, WORD hcmp)

Function: Read the level of the specified CMP port.

Parameter: ConnectNo	Designated link No. 0-7, default value 0
hcmp	High speed comparator, value range: 0-1 (corresponding to the last two output ports of hardware).

Return value: CMP port level.

Scope of application: SMC300 and SMC600 series of controllers

3.25 Limit function of software/hardware

short smc_set_el_mode(WORD ConnectNo, WORD axis, WORD el_enable, WORD el_logic, WORD el_mode)

Function: Set EL limit signal.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Axis Designated axis No., value range: 0- Max. number of axises of the controller -1

el_enable Enabling state of EL signal:

0: Positive and negative limit is disabled.

1: Positive and negative limit is enabled.

2: Positive limit is disabled while negative limit is enabled.

3: Positive limit is enabled and negative limit is disabled.

el_logicEL Active level of signal:

0: Low level of positive and negative limit is valid.

1: High level of positive and negative limit is valid.

2: Positive limit is low and effective, while negative limit is high and effective

3: Positive limit is high and effective, while negative limit is low and effective.

el_modeEL Braking mode:

0: Positive and negative limit stops immediately.

1: Positive and negative limit have deceleration stop.

2: The positive limit stops immediately, and the negative limit has deceleration stop.

3: The positive limit has deceleration stop and the negative limit stops immediately.

Return value: Error code

Application scope: Pulse type full series controller

⚠Note: When the function axis number is set as 255, all parameters of axis limit signal are set.

short smc_get_el_mode(WORD ConnectNo, WORD axis, WORD* el_enable, WORD* el_logic, WORD* el_mode)

Function: Read EL limit signal setting.

Parameter: ConnectNo		Designated link No. 0-7, default value 0
	axis	Designated axis No., value range: 0- Max. number of axises of the controller -1
	el_enable	Set the enabling state of EL signal.
	el_logic	Set the effective level of EL signal.
	el_mode	Return EL braking mode.

Return value: Error code

Application scope: Pulse type full series controller

short smc_set_softlimit_unit(WORD ConnectNo, WORD axis, WORD enable, WORD source_sel, WORD SL_action, double N_limit, double P_limit)

Function: Set soft limit.

Parameter: ConnectNo		Designated link No. 0-7, default value 0
	Axis	Designated axis No., value range: 0- Max. number of axises of the controller -1
	Enable	Enable status, 0: disabled, 1: enabled.
	source_sel	Counter selection, 0: instruction position counter, 1: encoder counter.
	SLaction	Stop mode: 0: stop immediately, 1: deceleration stop.
	N_limit	Negative limit position, unit.
	P_limit	Positive limit position, unit.

Return value: Error code

Application scope: Pulse type full series controller

⚠Note: The positive and negative limit positions can be either positive or negative, but the positive limit position should be greater than the negative limit position.

short smc_get_softlimit_unit(WORD ConnectNo, WORD axis, WORD* enable, WORD* source_sel, WORD* SL_action, double* N_limit, double* P_limit)

Function: Read the soft limit setting.

Parameter: ConnectNo		Designated link No. 0-7, default value 0
	Axis	Designated axis No., value range: 0- Max. number of axises of the controller -1
	Enable	Return to the enabled state.
	source_sel	Return counter selection.

SL_action	Return to limit stop mode, 0: stop immediately, 1: deceleration stop.
N_limit	Return the number of negative limit pulses.
P_limit	Return the number of positive limit pulses.

Return value: Error code

Application scope: Pulse type full series controller

3.26 Function of motion abnormal stop

short smc_stop (WORD ConnectNo, WORD axis, WORD stop_mode)

Function: Stop motion of specified axis.

Parameter: ConnectNo	Designated link No. 0-7, default value 0
axis	Designated axis No., value range: 0- Max. number of axises of the controller -1
stop_mode	Braking mode, 0: deceleration stop, 1: immediate stop.

Return value: Error code

Application scope: Full series of controllers

⚠Note: The function is applied to uniaxial and PVT motion

short smc_stop_multicoor(WORD ConnectNo, WORD Crd, WORD stop_mode)

Function: Stop the motion of all axes in the coordinate system.

Parameter: ConnectNo	Designated link No. 0-7, default value 0
Crd	Specify the frame number of the controller (Value range: 0~1)
stop_mode	Braking mode, 0: deceleration stop, 1: immediate stop.

Return value: Error code

Application scope: Pulse type full series controller

⚠Note: This function applies to interpolation motion

short smc_emg_stop(WORD ConnectNo)

Function: Emergency stop of all axes.

Parameter: ConnectNo	Designated link No. 0-7, default value 0

Return value: Error code

Application scope: Full series of controllers

⚠Note: This function is applicable to all motion modes.

short smc_set_emg_mode(WORD ConnectNo, WORD axis, WORD enable, WORD emg_logic)

Function: Set EMG emergency stop signal.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Enable Enable/disable signal function, 0: disabled, 1: enabled.

emg_logicEMG Signal active level, 0: low effective, 1: high effective.

Return value: Error code

Application scope: Pulse type full series controller

short smc_get_emg_mode(WORD ConnectNo, WORD axis, WORD* enbale, WORD* logic)

Function: Read settings of EMG emergency stop signal.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Enable Return EMG signal function status.

Logic Set effective level of EMG signal.

Return value: Error code

Application scope: Pulse type full series controller

short smc_set_io_dstp_mode(WORD ConnectNo, WORD axis, WORD enable, WORD logic)

Function: Set level signal of IO to trigger deceleration stop;

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

enable Enable/disable hardware signal function, 0: disabled, 1: enabled.

logic Effective level of external deceleration stop signal, 0: low effective, 1: high effective.

Return value: Error code

Application scope: Pulse type full series controller

⚠Attention: (1) The deceleration time of deceleration stop signal (DSTP) is set by the function smc_set_dec_stop_time.

(2) When the function axis number is set as 255, all parameters of axis stop signal are

set.

short smc_get_io_dstp_mode(WORD ConnectNo, WORD axis, WORD* enable, WORD* logic)

Function: Read the level signal of IO-triggered deceleration stop.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

enable Return function status of DSTP hardware signal.

logic Return the set effective level of the external deceleration stop signal.

Return value: Error code

Application scope: Pulse type full series controller

short smc_set_dec_stop_time(WORD ConnectNo, WORD axis, double stop_time)

Function: Set the deceleration stop time of fixed-length motion;

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

stop_time Deceleration time, unit: s

Return value: Error code

Application scope: Pulse type full series controller

⚠Attention: For any abnormal stop, such as calling smc_stop function, triggering of limit signal (software and hardware), triggering of deceleration stop signal (DSTP), etc., the deceleration stop time will be the deceleration time set in the function of smc_set_dec_stop_time.

short smc_get_dec_stop_time(WORD ConnectNo, WORD axis, double* stop_time)

Function: read the setting of deceleration stop time of fixed-length motion;

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

stop_time Return set deceleration time, unit: s.

Return value: Error code

Application scope: Pulse type full series controller

short smc_set_vector_dec_stop_time(WORD ConnectNo, WORD Crd, double time)

Function: set the deceleration stop time of interpolation motion;

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Specify the interpolation system, and the value range is 0-1.

stop_time Deceleration time, unit: s

Return value: Error code

Application scope: Pulse type full series controller

⚠Attention: For any abnormal stop, such as calling smc_stop_multicoor function, triggering of limit signal (software and hardware), triggering of deceleration stop signal (DSTP), etc., the deceleration stop time will be the deceleration time set in the function of smc_set_vector_dec_stop_time.

short smc_get_vector_dec_stop_time(WORD ConnectNo, WORD Crd, double* time)

Function: Read the setting of interpolation motion deceleration stop time;

Parameter: ConnectNo Designated link No. 0-7, default value 0

Crd Specify the interpolation system, and the value range is 0-1.

stop_time Return set deceleration time, unit: s.

Return value: Error code

Application scope: Pulse type full series controller

3.27 Axis IO mapping function

short smc_set_axis_io_map(WORD ConnectNo, WORD Axis, WORD IoType, WORD MapIoType, WORD MapIoIndex, double filter_time)

Function: Set parameters of axis IO mapping.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Axis Designated axis No., value range: 0- Max. number of axises of the controller -1

IoType Specify the IO signal type of axis:

0: Positive limit signal, AxisIoInMsg_PEL

1: Negative limit signal, AxisIoInMsg_NEL

2: Origin signal, AxisIoInMsg_ORG

3: Emergency stop signa, AxisIoInMsg_EMG

4: Deceleration stop signal, AxisIoInMsg_DSTP(reserved)

5: Servo alarm signal, AxisIoInMsg_ALM

6: Servo preparation signal, AxisIoInMsg_RDY(reserved)

7: Servo in-place signal, AxisIoInMsg_INP

MapIoType Type of axis IO mapping:

0: Positive limit input port, AxisIoInPort_PEL

1: Negative limit input port, AxisIoInPort_NEL

2: Origin input port, AxisIoInPort_ORG

3: Servo alarm input port, AxisIoInPort_ALM

4: Servo preparation input port, AxisIoInPort_RDY

5: Servo in-place input port, AxisIoInPort_INP

6: Universal input port, AxisIoInPort_IO

MapIoIndex Index number of axis IO mapping:

1) When the IO mapping type is set as 6, this parameter can be set as an integer from 0 to 17, which means the specific general input port number corresponding to the mapping.
2) When the IO mapping type is set as 0-5, this parameter can be set as an integer from 0 to 7, which means the specific axis number corresponding to the mapping.

filter_time IO signal filtering time, unit: s.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

⚠Note: This function can realize free configuration of hardware input interface of dedicated IO signal.

short smc_get_axis_io_map(WORD ConnectNo, WORD Axis, WORD IoType, WORD* MapIoType, WORD* MapIoIndex, double* filter_time)

Function: Read parameters of axis IO mapping relation

Parameter: ConnectNo Designated link No. 0-7, default value 0

Axis Designated axis No., value range: 0- Max. number of axises of the controller -1

IoType Type of axis IO signal

MapIoType Return the type of axis IO mapping

MapIoIndex Return the index number of axis IO mapping.

filter_time Return the filtering time of axis IO signal, unit: s.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

3.28 Virtual IO mapping function

short smc_set_io_map_virtual(WORD ConnectNo, WORD bitno, WORD MapIoType, WORD MapIoIndex, double filter_time)

Function: Set parameters of virtual IO mapping.

Parameter: ConnectNo Designated link No. 0-7, default value 0

bitno Virtual IO port number, value range: 0- the max. number of input IO ports of the controller -1.

MapIoType Type of virtual IO mapping:

0: Positive limit signal, AxisIoInPort_PEL

1: Negative limit signal, AxisIoInPort_NEL

2: Origin signal, AxisIoInPort_ORG

3: Servo alarm input port, AxisIoInPort_ALM

4: Servo preparation input port, AxisIoInPort_RDY

5: Servo in-place input port, AxisIoInPort_INP

6: Universal input port, AxisIoInPort_IO

MapIoIndex Index number of IO mapping:

1) When the type of virtual IO mapping is set as 6, this parameter can be set as an integer from 0 to 17, which means the specific general input port number corresponding to the mapping.
2) When the type of virtual IO mapping is set as 0-5, this parameter can be set as an integer from 0 to 7, which means the specific axis number corresponding to the mapping.

filter_time Filtering time of virtual IO signal, unit: s.

Return value: Error code

Scope of application: Controllers other than SMC100 series controllers.

⚠Note: This function can realize the filtering function of special universal IO input interface.

short smc_get_io_map_virtual(WORD ConnectNo, WORD bitno, WORD* MapIoType, WORD* MapIoIndex, double* filter_time)

Function: Read parameters of virtual IO mapping.

Parameter: ConnectNo Designated link No. 0-7, default value 0

	bitno	Virtual IO port number, value range: 0- the max. number of input IO ports of the controller -1.
	MapIoType	Return the type of virtual IO mapping.
	MapIoIndex	Return the index number of virtual IO mapping.
	filter_time	Return the filtering time of virtual IO signal, unit: s.

Return value: Error code

Scope of application: Controllers other than SMC100 series controllers.

short smc_read_inbit_virtual(WORD ConnectNo, WORD bitno)

Function: Read the level status of filtered virtual IO port;

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	bitno	Virtual IO port number, value range: 0- the max. number of input IO ports of the controller -1.

Return value: Level of designated virtual IO port: 0: low level, 1: high level.

Scope of application: Controllers other than SMC100 series controllers.

⚠Note: 1) This function needs to be used in conjunction with the virtual IO mapping function.

2) The difference between function smc_read_inbit_virtual and smc_read_inbit function: smc_read_inbit can directly read the level state of hardware port without filtering, and smc_read_inbit_virtual can read the filtered level state of corresponding port after virtual IO mapping.

3.29 Password management function

short smc_write_sn(WORD ConnectNo, uint64sn)

Function: Write serial number.

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	Sn	Serial number

Return value: Error code

Application scope: Full series of controllers

short smc_read_sn(WORD ConnectNo, uint64* sn)

Function: Write serial number.

Parameter:	ConnectNo	Designated link No. 0-7, default value 0
	Sn	Read serial number.

Return value: Error code

Application scope: Full series of controllers

short smc_write_password(WORD ConnectNo, const char* new_sn)

Function: Encryption.

Parameter: ConnectNo Designated link No. 0-7, default value 0

new_sn Password, the password length is not more than 256 characters.

Return value: Error code

Application scope: Full series of controllers

short smc_check_password(WORD ConnectNo, const char* check_sn)

Function: Password verification.

Parameter: ConnectNo Designated link No. 0-7, default value 0

check_sn Old password, the length is not more than 256 characters.

Return value: Verification status, 0: failed, 1: succeeded.

Application scope: Full series of controllers

⚠Note: 1) Password cannot be verified again after failing for 3 times.

2) Users can add password verification when the system software is turned on, to encrypt the system software.

short smc_enter_password(WORD ConnectNo, const char* str_pass)

Function: Login password.

Parameter: ConnectNo Designated link No. 0-7, default value 0

str_pass Password, the password length is no more than 16 characters.

Return value: Error code

Application scope: Full series of controllers

short smc_modify_password(WORD ConnectNo, const char* spassold, const char* spass)

Function: Modify login password.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Spassold Old password, password length is no more than 16 characters.

str_pass New password, password length is no more than 16 characters.

Return value: Error code

Application scope: Full series of controllers

⚠ Attention: smc_enter_password and smc_modify_password are mainly used as internal protections such as uploading files and initializing parameters.

3.30 Document management function

short smc_download_file(WORD ConnectNo, const char* pfilename, const char* pfilenameinControl, WORD filetype)

Function: Download local files to FLASH.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Pfilename Local file name, including full path.

pfilenameinControl File name in controller.

filetype Type of file: 0-Basic, 1-Gcode, 2- parameter, 3- firmware.

Return value: Error code

Application scope: Full series of controllers

short smc_download_memfile(WORD ConnectNo, const char* pbuffer, uint32 buffsize, const char* pfilenameinControl, WORD filetype)

Function: Download memory files to FLASH.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Pbuffer Memory file buffer zone.

Buffsize Memory file size

pfilenameinControl File name in controller.

filetype Type of file: 0-Basic, 1-Gcode, 2- parameter, 3- firmware.

Return value: Error code

short smc_upload_file(WORD ConnectNo, 0~7const char* pfilename, const char* pfilenameinControl, WORD filetype)

Function: Upload FLASH file to local file.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Pfilename Local file name, including full path.

pfilenameinControl File name in controller.

filetype Type of file: 0-Basic, 1-Gcode, 2- parameter, 3- firmware.

Return value: Error code

Application scope: Full series of controllers

short smc_upload_memfile(WORD ConnectNo, char* pbuffer, uint32 buffsize, const char* pfilenameinControl, uint32* puifilesize, WORD filetype)

Function: Upload FLASH file to memory file.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Pbuffer Memory file buffer zone.

buffsize Size of memory file buffer zone.

pfilenameinControl File name in controller.

puifilesize File size in controller.

filetype Type of file: 0-Basic, 1-Gcode, 2- parameter, 3- firmware.

Return value: Error code

short smc_download_file_to_ram(WORD ConnectNo, const char* pfilename, WORD filetype)

Function: Download the local file to RAM, not saved after power failure.

Parameter: ConnectNo Designated link No. 0-7, default value 0

pfilename Local file name, including full path.

filetype Type of file: 0-Basic, 1_Gcode, 2- parameter, 3- firmware.

Return value: Error code

Application scope: Full series of controllers

short smc_download_memfile_to_ram(WORD ConnectNo, const char* pbuffer, uint32 buffsize, WORD filetype)

Function: Download the local file to RAM, not saved after power failure.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Pbuffer Memory file buffer zone.

Buffsize Memory file size

filetype Type of file: 0-Basic, 1-Gcode, 2- parameter, 3- firmware.

Return value: Error code

Application scope: Full series of controllers

short smc_get_progress(WORD ConnectNo, float* process)

Function: File download progress

Parameter: ConnectNo Designated link No. 0-7, default value 0

Process Return the file download progress.

Return value: Error code

Application scope: Full series of controllers

short smc_format_flash(WORD ConnectNo)

Function: format FLASH.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Return value: Error code

Application scope: Full series of controllers

3.31 Register operation

short smc_set_modbus_0x(WORD ConnectNo, WORD start, WORD inum, char* pdata)

Function: AND bit register.

Parameter: ConnectNo Designated link No. 0-7, default value 0

start First address of register: 0~9999.

inum Number of registers

pdata An array of bytes, one byte including 8-bit registers.

Return value: Error code

Application scope: Full series of controllers

⚠ Note: pdata, byte array, a byte is an 8-bit register. If pdata = 5, the corresponding bit values of bit registers 0, 1 and 2 are 1, 0 and 1, respectively. When the number of registers is more than 8, and the bit values of Register 9 and Register 10 are set as 1 and 1, then set Pdata[1] = 3.

short smc_get_modbus_0x(WORD ConnectNo, WORD start, WORD inum, char* pdata)

Function: Read bit register.

Parameter: ConnectNo Designated link No. 0-7, default value 0

start First address of register: 0~9999.

inum Number of registers

pdata Return an array of bytes, one byte includes 8 bit registers.

Return value: Error code

Application scope: Full series of controllers

short smc_set_modbus_4x(WORD ConnectNo, WORD start, WORD inum, WORD* pdata)

Function: Write word register.

Parameter: ConnectNo Designated link No. 0-7, default value 0

start First address of register: 0~9999.

inum Number of registers

pdata Register value array.

Return value: Error code

Application scope: Full series of controllers

short smc_get_modbus_4x(WORD ConnectNo, WORD start, WORD inum, WORD* pdata)

Function: Read word register.

Parameter: ConnectNo Designated link No. 0-7, default value 0

start First address of register: 0~9999.

inum Number of registers

pdata Return an array of word register values.

Return value: Error code

Application scope: Full series of controllers

short smc_set_persistent_reg(WORD ConnectNo, DWORD start, DWORD inum, const char* pdata);

Function: Set the value of power-down save register.

Parameter: ConnectNo Designated link No. 0-7, default value 0

start First address of register: 0~4096.

inum Number of registers, the max. number of writing is 1024 bytes.

pdata Register value array

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_get_persistent_reg(WORD ConnectNo, DWORD start, DWORD inum, char* pdata) Function: Read the value of the power-down save register.

Parameter: ConnectNo Designated link No. 0-7, default value 0

start First address of register: 0~4096.

inum Number of registers, the max. reading time is 1024 bytes.

pdata Read register value array.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

3.32 Operation function of analog quantity

double smc_get_ain (WORD ConnectNo, WORD channel)

Function: Read analog voltage value.

Parameter: ConnectNo Designated link No. 0-7, default value 0

channel Channel number: 0, 1.

Return value: analog voltage: 0-5V.

Scope of application: SMC104.

short smc_set_ain_action(WORD ConnectNo, WORD channel, WORD mode, double fvoltage, WORD action, double actpara)

Function: Set analog parameters.

Parameter: ConnectNo Designated link No. 0-7, default value 0

channel Channel number: 0, 1.

mode Trigger mode: 0- no trigger, 1- less than or equal to, 2- greater than or equal to.

fvoltage Voltage value: 0-5V.

action Trigger action, see the table below.

actpara See the following table for trigger parameters.

Return value: Error code

Scope of application: SMC104.

Action	Action	Actpara
Uniaxial deceleration stops	0	Axis number, SMC104(0-3).
Uniaxial immediate stop.	1	Axis number, SMC104(0-3).
Multi-axis deceleration stops.	2	Axis number mark
Multi-axis stops immediately.	3	Axis number mark

short smc_get_ain_action(WORD ConnectNo, WORD channel, WORD* mode, double* fvoltage, WORD* action, double* actpara)

Function: Read the analog setting parameter value.

Parameter: ConnectNo Designated link No. 0-7, default value 0

channel Channel number: 0, 1.

mode Read trigger mode: 0- no trigger, 1- less than or equal to, 2- greater than or equal to.

fvoltage Read voltage value: 0-5V.

action Read trigger action, see the table below.

actpara Read the trigger action parameters, see the table below.

Return value: Error code

Scope of application: SMC104.

Action	Action	Actpara
Uniaxial deceleration stops	0	Axis number, SMC104(0-3).
Uniaxial immediate stop.	1	Axis number, SMC104(0-3).
Multi-axis deceleration stops.	2	Axis number mark
Multi-axis stops immediately.	3	Axis number mark

short smc_set_ain_state(WORD ConnectNo, WORD channel)

Function: Set the trigger state of analog input.

Parameter: ConnectNo Designated link No. 0-7, default value 0

channel Channel number: 0, 1.

Return value: Error code

Scope of application: SMC104.

short smc_get_ain_state(WORD ConnectNo, WORD channel)

Function: Read the trigger status of analog input.

Parameter: ConnectNo Designated link No. 0-7, default value 0

channel Channel number: 0, 1.

Return value: 0- not triggered, 1- triggered.

Scope of application: SMC104.

⚠Note: Presently, this function is not supported by some controllers, such as SMC606 and SMC604. SMC104 has this function.

3.33 BASIC-related function

short smc_write_array(WORD ConnectNo, const char* name, uint32 startindex, int64*var, int32num)

Function: Write array values in batches, starting from the startindex index number of the array, and counting num array values continuously.

Parameter: ConnectNo Designated link No. 0-7, default value 0

name The array name, such as array.

startindex Start index number, for example, revelation index number is 5.

var Array value list array, such as {array[5], array[6]}, each value = actual

value *4294967296 (i.e., the actual value moves 32 bits to the right).

num Number of array values

Return value: Error code

Application scope: Full series of controllers

short smc_write_array_ex(WORD ConnectNo, const char* name, uint32 startindex, double* var, int32num)

Function: Write array values in batches, starting from the startindex index number of the array, and counting num array values continuously.

Parameter: ConnectNo Designated link No. 0-7, default value 0

name The name of the array, such as "array".

startindex The starting index number, such as the starting index number is 5.

var An array of values, such as {array[5], array[6]}

num Number of array values, for example, the number of array values is 2.

Return value: Error code

Application scope: Full series of controllers

short smc_read_array(WORD ConnectNo, const char* name, uint32 index, int64* var, int32* num)

Function: Read array values by index.

Parameter: ConnectNo Designated link No. 0-7, default value 0

name Return the array name, with commas between multiple numbers, such as "array0, array1".

index Index number, such as index number is 1.

var Return an array of array values corresponding to the index number, such as {array0[1], array1[1]}, where the actual value = each value /4294967296 (or each value moves 32 bits to the left) num. For example, the number of array values is 2.

Return value: Error code

Application scope: Full series of controllers

short smc_read_array_ex(WORD ConnectNo, const char* name, uint32 index, double* var, int32*num)

Function: Read array values by index.

Parameter: ConnectNo		Designated link No. 0-7, default value 0
	name	Return the array name, with commas between multiple numbers, such as "array0, array1".
	index	Index number, such as index number is 1.
	var	Return the array of array values corresponding to the index number, such as {array0[1], array1[1]}
	num	Return the number of array values, for example, the number of array values is 2.

Return value: Error code

Application scope: Full series of controllers

short smc_modify_array(WORD ConnectNo, const char* name, uint32 index, int64* var, int32 num)

Function: Modify array values by index.

Parameter: Name		Array names, with commas between them, such as "array0, array1".
	index	Index number, such as index number is 1.
	var	The index number corresponds to an array of array values, such as {array0[1], array1[1]}, and each value = actual value *4294967296 (or the actual value is shifted to the right by 32 bits).
	num	Number of arrays, for example, the number of arrays is 2.

Return value: Error code

Application scope: Full series of controllers

short smc_modify_array_ex(WORD ConnectNo, const char* name, uint32 index, double* var, int32num)

Function: Modify array values by index.

Parameter: ConnectNo		Designated link No. 0-7, default value 0
	name	Array names, with commas between them, such as "array0, array1".
	index	Index number, such as index number is 1.
	var	The index number corresponds to an array of array values, such as {array0[1], array1[1]}
	num	Number of arrays, for example, the number of arrays is 2.

Return value: Error code

Application scope: Full series of controllers

short smc_read_var(WORD ConnectNo, const char* name, int64* var, int32* num)

Function: Read variable value.

Parameter: ConnectNo Designated link No. 0-7, default value 0

name Variable names with commas between them, such as "var0, var1".

var Returns an array of variable values, such as {var0, var1}, where the actual value = each value /4294967296 (or each value moves 32 bits to the left).

num Return the number of variables, for example, the number of variables is 2.

Return value: Error code

Application scope: Full series of controllers

short smc_read_var_ex(WORD ConnectNo, const char* name, double* var, int32* num)

Function: Read variable value.

Parameter: ConnectNo Designated link No. 0-7, default value 0

name Variable names with commas between them, such as "var0, var1".

var Return an array of variable values, such as {var0, var1}

num Return the number of variables, for example, the number of variables is 2.

Return value: Error code

Application scope: Full series of controllers

short smc_modify_var(WORD ConnectNo, const char* name, int64* var, int32 num)

Function: Modify variable value.

Parameter: ConnectNo Designated link No. 0-7, default value 0

name Variable names, with commas between them.

var Variable value

num Number of variables

Return value: Error code

Application scope: Full series of controllers

short smc_modify_var_ex(WORD ConnectNo, const char* name, double* var, int32 num)

Function: Modify variable value.

Parameter: ConnectNo Designated link No. 0-7, default value 0

name Variable names with commas between them, such as "var0, varr".

var Array of variable values, such as {var0, var1}

num Number of variables, such as 2.

Return value: Error code

Application scope: Full series of controllers

short sm_get_stringtype(WORD ConnectNo, const char* varstring, int32* m_Type, int32* num)

Function: Read variable type.

Parameter: ConnectNo Specify link number: 0-7, default value: 0varstring variable name.

m_Typ Types, see string_types enumeration for details.

num Array length

Return value: Error code

Application scope: Full series of controllers

enumstring_types

{

STRING_USERSUB = 1, //SUB

STRING_VARIABLE = 2, //global variable.

STRING_ARRAY = 3, //global array.

STRING_PARA = 4, //parameter.

STRING_CMD = 5, //command.

STRING_KEYWORD = 6. //Reserve keywords.

STRING_VARIABLE_LOCAL = 7, //local variable.

STRING_ARRAY_LOCAL = 8, //local array.

STRING_UNKOWN = 10. //Unknown variable.

};

short smc_basic_delete_file(WORD ConnectNo)

Function: Delete BASIC program.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Return value: Error code

Application scope: Full series of controllers

short smc_basic_run(WORD ConnectNo)

Function: Run.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Return value: Error code

Application scope: Full series of controllers

short smc_basic_stop(WORD ConnectNo)

Function: Stop.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Return value: Error code

Application scope: Full series of controllers

short smc_basic_pause(WORD ConnectNo)

Function: Pause

Parameter: ConnectNo Link number: 0~7.

Return value: Error code

short smc_basic_step_run(WORD ConnectNo)

Function: Single step operation.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Return value: Error code

Application scope: Full series of controllers

short smc_basic_step_over(WORD ConnectNo)

Function: Run to the next breakpoint.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Return value: Error code

Application scope: Full series of controllers

short smc_basic_continue_run(WORD ConnectNo)

Function: Continue to run.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Return value: Error code

Application scope: Full series of controllers

short smc_basic_state(WORD ConnectNo, WORD* State)

Function: Current status.

Parameter: ConnectNo Designated link No. 0-7, default value 0

State Running status: 1- running, 2- suspended, 3- stopped, 100- abnormal.

Return value: Error code

Application scope: Full series of controllers

short smc_basic_current_line(WORD ConnectNo, uint32* line)

Function: Current execution line.

Parameter: ConnectNo Designated link No. 0-7, default value 0

line Run line number

Return value: Error code

Application scope: Full series of controllers

short smc_basic_break_info(WORD ConnectNo, uint32* line, uint32 linenum)

Function: Breakpoint information.

Parameter: ConnectNo Designated link No. 0-7, default value 0

line Breakpoint row list array.

linenum Breakpoint rows

Return value: Error code

Application scope: Full series of controllers

short smc_basic_message(WORD ConnectNo, char* pbuff, uint32 uimax, uint32* puiread)

Function: Read output information.

Parameter: ConnectNo Designated link No. 0-7, default value 0

pbuff Output information buffer zone.

uimax Buffer zone size

puiread Return the size of output information.

Return value: Error code

Application scope: Full series of controllers

short smc_basic_command(WORD ConnectNo, const char* pszCommand, char* psResponse, uint32 uiResponseLength)

Function: Online command.

Parameter: ConnectNo Designated link No. 0-7, default value 0

pszCommand Command string

psResponse Return character string

uiResponseLength Return the string length.

Return value: Error code

Application scope: Full series of controllers

3.34 G code-related function

short smc_gcode_check_file(WORD ConnectNo, const char* pfilenameinControl, uint8* pbIfExist, uint32* pFileSize)

Function: Check whether the file exists.

Parameter: ConnectNo Designated link No. 0-7, default value 0

pfilenameinControl Controller file name

pbIfExist Existence: 0- not exist, 1- exists.

pFileSize File size, return does not exist (uint32)-1.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_gcode_delete_file(WORD ConnectNo, const char* pfilenameinControl)

Function: Delete files.

Parameter: ConnectNo Designated link No. 0-7, default value 0

pfilenameinControl Controller file name

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_gcode_clear_file(WORD ConnectNo)

Function: Delete all files.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_gcode_get_first_file(WORD ConnectNo, char* pfilenameinControl, uint32* pFileSize)

Function: Read the first file name.

Parameter: ConnectNo Designated link No. 0-7, default value 0

pfilenameinControl Controller file name

pFileSize File size, return does not exist (uint32)-1.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_gcode_get_next_file(WORD ConnectNo, char* pfilenameinControl, uint32* pFileSize)

Function: Read the next file name.

Parameter: ConnectNo Designated link No. 0-7, default value 0

pfilenameinControl Controller file name

pFileSize File size, return does not exist (uint32)-1.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_gcode_start(WORD ConnectNo)

Function: Start.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_gcode_stop(WORD ConnectNo);

Function: Stop.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_gcode_pause(WORD ConnectNo)

Function: Pause

Parameter: ConnectNo Designated link No. 0-7, default value 0

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_gcode_state(WORD ConnectNo, WORD* State)

Function: Read the current status.

Parameter: ConnectNo Designated link No. 0-7, default value 0

State Current running status of G code: 1- running, 2- suspended, 3- stopped, 4- abnormal.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_gcode_set_current_file(WORD ConnectNo, const char* pfilenameinControl)

Function: set the current file.

Parameter: ConnectNo Designated link No. 0-7, default value 0

const char* pfilenameinControl Controller file name

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_gcode_get_current_file(WORD ConnectNo, char* pfilenameinControl, WORD* fileid)

Function: Read the current file name.

Parameters: WORD ConnectNo Link number: 0~7.

pfilenameinControl Return controller file name.

fileid Return the current file number.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_gcode_current_line(WORD ConnectNo, uint32* line, char* pCurLine)

Function: Read the current running line.

Parameter: ConnectNo Designated link No. 0-7, default value 0

uint32* line Line numbers

char* pCurLine Line string

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

short smc_gcode_get_file_profile(WORD ConnectNo, uint32* maxfilenum, uint32* maxfilesize, uint32* savedfilenum);

Function: Read attributes of G file.

Parameter: ConnectNo Designated link No. 0-7, default value 0

uint32* maxfilenum Max. number of files

uint32* maxfilesize Max.file capacity.

uint32* savedfilenum Number of saved files.

Return value: Error code

Scope of application: SMC300 and SMC600 series of controllers

3.35 Busbar-related function

3.35.1 Bus configuration function

short nmcs_reset_canopen(WORD ConnectNo)

Function: Reset CANopen bus.

Parameter: ConnectNo Designated link No. 0-7, default value 0

short nmcs_stop_etc(WORD ConnectNo, WORD* ETCState)

Function: Stop EtherCAT bus running.

Parameter: ConnectNo Designated link No. 0-7, default value 0

ETCState 0: Stop EtherCAT bus succeeded, 1: Stop EtherCAT bus failed.

Return value: Error code

short nmcs_axis_io_status(WORD ConnectNo, WORD axis)

Function: Acquire axis IO information.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Axis Axis number

short nmcs_get_LostHeartbeat_Nodes(WORD ConnectNo, WORD PortNo, WORD* NodeID, WORD* NodeNum)

Function: Acquire heartbeat message information, and support CANopen and EtherCat.

Parameter: ConnectNo Designated link No. 0-7, default value 0

PortNo The port number is fixed at 2 (where 0 and 1 are CANopen ports).

NodeID Node number

NodeNum Number of nodes

short nmcs_get_EmergeneyMessege_Nodes(WORD ConnectNo, WORD PortNo, DWORD* NodeMsg, WORD* MsgNum)

Function: Acquire emergency message information, and support CANopen and EtherCAT.

Parameter: ConnectNo Designated link No. 0-7, default value 0

PortNo The port number is fixed at 2 (where 0 and 1 are CANopen ports).

NodeMsg

MsgNum

short nmcs_set_node_od(WORD ConnectNo, WORD PortNum, WORD NodeNum, intIndex, intSubIndex, intValLength, intValue)

Function: Set the slave object dictionary.

Parameter: ConnectNo Designated link No. 0-7, default value 0

NodeNum Node number

PortNum Port number (0-3)

Index Index

SubIndex Child index

ValLength Length (this parameter has only three values: 8, 16 and 32).

Value Slave station value

Return value: Error code

Application scope: Full series of controllers

short nmcs_get_node_od(WORD ConnectNo, WORD PortNum, WORD NodeNum, intIndex, intSubIndex, int*ValLength, int*Value)

Function: Acquire the slave object dictionary.

Parameter: ConnectNo Designated link No. 0-7, default value 0

PortNum Port number (0-3)

NodeNum Node number

Index Index

SubIndex Child index

ValLength Length (this parameter has only three values: 8, 16 and 32).

Value Return the slave station value.

Return value: Error code

Application scope: Full series of controllers

short nmcs_SendNmtCommand(WORD ConnectNo, WORD PortNum, WORD NodeID, WORD NmtCommand)

Function: Send NMT management message.

Parameter: ConnectNo Designated link No. 0-7, default value 0

PortNum Port number:

0(CANBUS_0)

1(CANBUS_1)

NodeID Node number

NmtCommand NmtCommandNMT command, this parameter has five values:

0x01- Start the remote node.

0x02- Stop the remote node.

0x80- Remote node enters pre-operation mode.

0x81- Reset the remote node.

0x82- Reset communication of remote node

Return value: Error code

Application scope: Full series of controllers

short nmcs_get_cycletime(WORD ConnectNo, WORD PortNum, DWORD* CycleTime)

Function: Read EtherCAT bus cycle.

Parameter: ConnectNo Designated link No. 0-7, default value 0

PortNum EtherCAT port number, fixed at 2 (where 0 and 1 are CANopen ports).

CycleTime EtherCAT bus cycle, unit: us.

Return value: Error code

short nmcs_write_rxpdo_extra(WORD ConnectNo, WORD PortNo, WORD address, WORD DataLen, DWORD Value)

Function: Write extended rxpdo.

Parameter: ConnectNo Link number: 0-7, the default value is 0.

PortNum: Port number, 0, 1 means CANOpen, 2, 3 means EtherCAT port.

Address: The first address of the extended PDO.

DataLen: Data length, calculated as 16bit, with a max. value of 2 (representing 32bit of data).

Value: Data value.

short nmcs_read_rxpdo_extra(WORD ConnectNo, WORD PortNo, WORD address, WORD DataLen, DWORD* Value)

Function: Read extended rxpdo.

Parameter: ConnectNo Link number: 0-7, the default value is 0.

PortNum: Port number, 0, 1 means CANOpen, 2, 3 means EtherCAT port.

Address: The first address of the extended PDO.

DataLen: Data length, calculated as 16bit, with a max. value of 2 (representing 32bit of data).

Value: Data value.

Short nmcs_read_txpdo_extra(WORD ConnectNo, WORD PortNo, WORD address, WORD DataLen, DWORD* Value)

Function: Read extended txpdo.

Parameter: ConnectNo Link number: 0-7, the default value is 0.

PortNum: Port number, 0, 1 means CANOpen, 2, 3 means EtherCAT port.

Address: The first address of the extended PDO.

DataLen: Data length, calculated as 16bit, with a max. value of 2 (representing 32bit of data).

Value: Data value.

3.35.2 Bus IO and axis control function

short nmcs_get_axis_type(WORD ConnectNo, WORD axis, WORD* Axis_Type)

Function: Read axis type.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Axis number

Axis_Type Type of axis, 0: Virtual axis, 1: EtherCAT axis, 2: CANopen axis, 3: Pulse axis, 4: Unknown type axis.

Return value: Error code

short nmcs_set_axis_enable(WORD ConnectNo, WORD axis)

Function: Enable EtherCAT bus driver.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Axis number of EtherCAT bus axis, 255 means to enable all EtherCAT axes.

Return value: Error code

short nmcs_set_axis_disable(WORD ConnectNo, WORD axis)

Function: Disable EtherCAT bus driver

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Axis number of EtherCAT bus axis, 255 means all EtherCAT axes are disabled.

Return value: Error code

short nmcs_syn_move(WORD ConnectNo, WORD AxisNum, WORD* AxisList, int*Position, WORD* PosiMode)

Function: Synchronous motion.

Parameter: ConnectNo Designated link No. 0-7, default value 0

AxisNum Axis number (value range 1-32, max. 32 axes).

AxisList Axis list

Position Destination location list

PosiMode List of location modes:

0- Relative mode.

1- Absolute mode.

Return value: Error code

Application scope: Full series of controllers

⚠Note: The motion speed of this motion is set by smc_set_profile_unit.

short nmcs_get_axis_state_machine(WORD ConnectNo, WORD axis, WORD* Axis_StateMachine);

Function: Read EtherCAT bus axis state machine.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Axis number of EtherCAT bus axis.

Axis_StateMachine EtherCAT bus axis state machine.

0: Not started.

1: Start disabled state.

2: Ready to start state.

3: Startup status.

4: Operation enabled status.

5: Stop status.

6: Error trigger status.

7: Error status.

Return value: Error code

short nmcs_get_axis_controlmode(WORD ConnectNo, WORD axis, long *contrlmode);

Function: Read control mode of EtherCAT bus axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Axis number of EtherCAT bus axis.

contrlmode EtherCAT bus axis control mode, 6: Return to zero mode, 8: CSP mode.

Return value: Error code

short nmcs_set_home_profile(WORD ConnectNo, WORD PortNum, WORD axis, WORD home_mode, double High_Vel, double Low_Vel, double Tacc, double Tdec, double offsetpos)

Function: Set homing parameter of EtherCAT bus axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

PortNum EtherCAT bus port number, fixed at 2 (where 0 and 1 are CANopen ports).

axis Axis number of EtherCAT bus axis.

home_mode EtherCAT bus axis homing mode.

High_Vel EtherCAT bus axis homing at high speed.

Low_Vel EtherCAT bus axis homing at low speed.

Tacc EtherCAT bus axis homing acceleration.

Tdec EtherCAT bus axis homing deceleration.

offsetpos EtherCAT bus axis homing offset.

Return value: Error code

short nmcs_get_total_axes(WORD ConnectNo, DWORD* TotalAxis)

Function: Read the number of EtherCAT bus axes and virtual axes.

Parameter: ConnectNo Designated link No. 0-7, default value 0

TotalAxis Number of EtherCAT bus axis and virtual axis.

Return value: Error code

short nmcs_get_total_adcnum(WORD ConnectNo, WORD* TotalIn, WORD* TotalOut)

Function: Read the number of AD/DA input/output ports of EtherCAT bus.

Parameter: ConnectNo Designated link No. 0-7, default value 0

TotalIn EtherCAT bus AD input number.

TotalOut EtherCAT bus DA output number.

Return value: Error code

short nmcs_get_total_ionum(WORD ConnectNo, WORD* TotalIn, WORD* TotalOut)

Function: Read the number of I/O ports of EtherCAT bus.

Parameter: ConnectNo Designated link No. 0-7, default value 0

TotalIn EtherCAT bus IO inputs number.

TotalOut EtherCAT bus IO output number.

Return value: Error code

short nmcs_get_total_adcnum(WORD CardNo, WORD* TotalIn, WORD* TotalOut)

Function: Acquire the analog I/O points of EtherCAT bus.

Parameter: CardNo Control card number

TotalIn Return the number of analog input ports.

TotalOut Return the number of analog output ports.

Return value: Error code

Application scope: Full series of controllers

short nmcs_get_total_slaves(WORD ConnectNo, WORD PortNum, WORD* TotalSlaves)

Function: Acquire the total number of EtherCAT slave stations.

Parameter: ConnectNo Designated link No. 0-7, default value 0

PortNum EtherCAT port number, fixed at 2 (where 0 and 1 are CANopen ports).

TotalSlaves Total EtherCAT slave stations.

Return value: Error code

short nmcs_get_axis_io_out(WORD ConnectNo, WORD axis, DWORD* iostate)

Function: Acquire the digital output IO status of EtherCAT axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis EtherCAT axis number

iostate EtherCAT axis digital output IO status.

Return value: Error code

short nmcs_set_axis_io_out(WORD ConnectNo, WORD axis, DWORD iostate)

Function: Set the digital output IO status of EtherCAT axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis EtherCAT axis number

iostate EtherCAT axis digital output IO status.

Return value: Error code

short nmcs_get_axis_io_in(WORD ConnectNo, WORD axis, DWORD* iostate)

Function: Acquire the digital input IO status of EtherCAT axis.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis EtherCAT axis number

iostate EtherCAT axis digital quantity input IO status.

Return value: Error code

short nmcs_write_outbit(WORD ConnectNo, WORD NoteID, WORD IoBit, WORD IoValue)

Function: Set the level of an output port of the designated controller or expansion module.

Parameter: ConnectNo Designated link No. 0-7, default value 0

NodeID Node number

IoBit Output port number

IoValue Output level, 0: low level, 1: high level.

Return value: Error code

Application scope: Full series of controllers

short nmcs_read_outbit(WORD ConnectNo, WORD NoteID, WORD IoBit, WORD* IoValue)

Function: Read the level of an output port of a specified controller or expansion module.

Parameter: ConnectNo Designated link No. 0-7, default value 0

NodeID Node number

IoBit Output port number

IoValue Return output level, 0: low level, 1: high level.

Return value: Error code

Application scope: Full series of controllers

short nmcs_read_inbit(WORD ConnectNo, WORD NoteID, WORD IoBit, DWORD* IoValue)

Function: Read the level of an input port of a specified controller or expansion module.

Parameter: ConnectNo Designated link No. 0-7, default value 0

NodeID Node number

IoBit Enter port number

IoValue Input port level, 0: low level, 1: high level.

Return value: Error code

Application scope: Full series of controllers

short nmcs_write_outport(WORD ConnectNo, WORD NodeID, WORD PortNo, WORD IoValue)

Function: Set the level of all output ports of specified IO group number.

Parameter: ConnectNo Designated link No. 0-7, default value 0

NodeID Node number

PortNo IO group number, min. value is 0.

IoValue Input port level, 0: low level, 1: high level.

Return value: Error code

Application scope: Full series of controllers

⚠Note: The min. value of IO group number is 0, and IO is set every 32 points; if the IO number is over 32, the group number will be increased by 1.

short nmcs_read_outport(WORD ConnectNo, WORD NodeID, WORD PortNo, DWORD* IoValue)

Function: Read the level of all output ports of specified IO group number.

Parameter: ConnectNo Designated link No. 0-7, default value 0

NodeID Node number

PortNo IO group number, min. value is 0.

IoValue Input port level, 0: low level, 1: high level.

Return value: Error code

Application scope: Full series of controllers

⚠Note: The min. value of IO group number is 0, and IO is set every 32 points; if the IO number is over 32, the group number will be increased by 1.

short nmcs_read_inport(WORD ConnectNo, WORD NodeID, WORD PortNo, DWORD* IoValue)

Function: Read the level of all input ports of specified IO group number.

Parameter: ConnectNo Designated link No. 0-7, default value 0

NodeID Node number

PortNo IO group number, min. value is 0.

IoValue Input port level, 0: low level, 1: high level.

Return value: Error code

Application scope: Full series of controllers

⚠Note: The min. value of IO group number is 0, and IO is set every 32 points; if the IO number is over 32, the group number will be increased by 1.

3.35.3 Bus error code function

short nmcs_set_alarm_clear(WORD ConnectNo, WORD PortNo, WORD NodeNo)

Function: Clear the alarm signal, and support CANopen and EtherCAT.

Parameter: ConnectNo Designated link No. 0-7, default value 0

PortNo EtherCAT bus port number, fixed at 2 (where 0 and 1 are CANopen ports).

NodeNo The default value is 0.

short nmcs_get_errcode(WORD ConnectNo, WORD PortNum, WORD* errcode)

Function: Read EtherCAT bus status.

Parameter: ConnectNo Designated link No. 0-7, default value 0

PortNum EtherCAT bus port number, fixed at 2 (where 0 and 1 are CANopen ports).

errcode EtherCAT bus status, 0 means normal.

Return value: Error code

short nmcs_clear_errcode(WORD ConnectNo, WORD PortNo)

Function: Clear bus error.

Parameter: ConnectNo Designated link No. 0-7, default value 0

PortNum EtherCAT bus port number, fixed at 2 (where 0 and 1 are CANopen ports).

Return value: Error code

short nmcs_get_card_errcode(WORD ConnectNo, WORD* Errcode);

Function: Acquire bus error code of specified link number.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Errcode Bus error code.

Return value: Error code

Application scope: Full series of controllers

short nmcs_clear_card_errcode(WORD ConnectNo);

Function: Clear the bus error code of the specified link number.

Parameter: ConnectNo Designated link No. 0-7, default value 0

Return value: Error code

Application scope: Full series of controllers

short nmcs_get_axis_errcode(WORD ConnectNo, WORD axis, WORD* Errcode);

Function: Acquire the specified axis bus error code.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Errcode Bus error code.

Return value: Error code

Application scope: Full series of controllers

short nmcs_clear_axis_errcode(WORD ConnectNo, WORD axis);

Function: Clear the bus error code of the specified axis number.

Parameter: ConnectNo Designated link No. 0-7, default value 0

axis Designated axis No., value range: 0- Max. number of axises of the controller -1

Return value: Error code

Application scope: Full series of controllers

Table 1: List of API Function

The following table lists the PC-side functions used and their descriptions.

Function	Name	Function introduction
Controller configuration function	smc_board_init	The controller links initialization functions and allocates system resources.
	smc_board_init_ex	Advanced links initialization function of controller, allocates system resources.
	smc_board_close	The controller closes the function and releases the system resources.
	smc_set_connect_timeout	Network link timeout.
	smc_get_release_version	Read the release version number.
	smc_get_card_version	Acquire the controller hardware version number.
	smc_get_card_soft_version	Acquire the controller firmware version number.
	smc_get_card_lib_version	Acquire the version number of the controller DLL file.
	smc_get_total_axes	Acquire the axis number of the current controller.
	smc_set_debug_mode	Call printout settings through function
	smc_get_debug_mode	Read function and call printout settings.
	smc_format_flash	Format FLASH
	smc_set_ipaddr	Set new IP address of controller.
	smc_get_ipaddr	Read controller IP address.
	smc_set_com	Set the parameters of controller COM port.
	smc_get_com	Read the parameters of controller COM port.
Serial number	smc_write_sn	Write serial number
	smc_read_sn	Read serial number.
Pulse mode setting.	smc_set_pulse_outmode	Set the pulse output mode of designated axis
	smc_get_pulse_outmode	Specifies the setting of pulse output mode of specified axis.
Setting of pulse equivalent	smc_set_equiv	Set pulse equivalent
	smc_get_equiv	Return the setting of pulse equivalent value.
Backlash setting	smc_set_backlash_unit	Set reverse clearance value
	smc_get_backlash_unit	Read the setting of reverse gap value.
Condition monitoring of	smc_check_done	Detect the motion state of the specified axis.
	smc_check_done_multicoor	Detecting the motion state of coordinate system.
	smc_axis_io_enable_status	Read the enabling status of the special signal of the specified axis.

uniaxial speed parameters.	smc_axis_io_status	Read the state of the motion signal about the specified axis.
	smc_get_axis_run_mode	Read axis motion mode.
	smc_set_position_unit	Set the value of current instruction position counter.
	smc_get_position_unit	Read the value of current instruction position counter.
	smc_read_current_speed_unit	Read current speed of axis.
	smc_get_stop_reason	Read axis stop reason.
	smc_clear_stop_reason	Clear the axis stop reason.
	smc_get_target_position_unit	Read the current target location.
	smc_set_workpos_unit	Set the origin point of current workpiece.
	smc_get_workpos_unit	Read the origin point of current workpiece.
	smc_set_profile_unit	Set speed curve of single-axis motion
	smc_get_profile_unit	Read speed curve of single-axis motion
	smc_set_s_profile	Set the parameter value of S section of uniaxial velocity curve.
	smc_get_s_profile	Read the parameter value of S section of uniaxial velocity curve.
Uniaxial motion	smc_pmove_unit	Fixed-length motion
	smc_vmove	Specify the axis to move continuously.
	smc_reset_target_position_unit	Change the current target position of the specified axis online.
	smc_update_target_position_unit	Change the current target position of the specified axis by force.
	smc_change_speed_unit	Change the current speed of the specified axis online.
Homing motion	smc_set_home_pin_logic	Set origin point signal of ORG.
	smc_get_home_pin_logic	Read the setting of 0RG origin point signal.
	smc_set_homemode	Set homing mode
	smc_get_homemode	Read homing mode
	smc_set_ez_count	Set homing EZ number
	smc_get_ez_count	Read homing EZ number
	smc_set_home_position_unit	Set deviation position value after homing
	smc_get_home_position_unit	Read deviation position value after homing
	smc_set_home_profile_unit	Set homing speed parameter
	smc_get_home_profile_unit	Read homing speed parameter
	smc_set_el_home	Original point switch function at limit position
	smc_home_move	Homing motion

	smc_get_home_result	Read the origin point motion state.
PVT motion function	smc_ptt_table_unit	Transmit data to the specified data sheet in PTT mode.
	smc_pts_table_unit	Transmit data to the specified data sheet in PTS mode.
	smc_pvt_table_unit	Transmit data to the specified data sheet in PVT mode.
	smc_pvts_table_unit	Transmit data to the specified data sheet in PVTS mode.
	smc_pvt_move	Start PVT motion
Electronic cam	smc_cam_table_unit	Set cam sheet
	Smc_cam_move	Start the cam motion.
Interpolation motion parameters	smc_set_vector_profile_unit	Set interpolation motion speed parameter.
	smc_get_vector_profile_unit	Read interpolation motion speed parameter.
	smc_set_vector_s_profile	Set the smoothing time of speed curve of interpolated motion
	smc_get_vector_s_profile	Read the set smoothing time of interpolation velocity curve.
Single segment interpolation motion	smc_line_unit	Straight-line interpolated motion
	smc_arc_move_center_unit	Spiral interpolation motion extended based on circle center + arc end point mode (it can be used for two-axis arc interpolation)
	smc_arc_move_radius_unit	Spiral interpolation motion extended by radius + arc end point mode (it can be used for two-axis arc interpolation)
	smc_arc_move_3points_unit	Spiral interpolation motion extended by three-point circular arc mode (it can be used for spatial circular arc interpolation of two axes and three axes).
Continuous insertion Complementary motion	smc_conti_open_list	Open buffer zone of continuous interpolation
	smc_conti_set_lookahead_mode	Set the lookahead mode and parameters of continuous interpolation
	smc_conti_get_lookahead_mode	Read lookahead mode and parameter of continuous interpolation
	smc_conti_start_list	Start continuous interpolation
	smc_conti_close_list	Close buffer zone of continuous interpolation
	smc_conti_pause_list	Pause continuous interpolation
	smc_conti_stop_list	Stop continuous interpolation
	smc_set_arc_limit	Set arc speed limit function.
	smc_get_arc_limit	Read the arc speed limit function.
	smc_conti_set_blend	Set enabling status of Blend corner transition mode during continuous interpolation
	smc_conti_get_blend	Read the enabling status setting of Blend corner transition mode in continuous interpolation.

	smc_conti_change_speed_ratio	Dynamic adjustment of speed ratio of continuous interpolation
	smc_conti_delay	Pause delay command during continuous interpolation.
	smc_conti_line_unit	Straight-line interpolation command in continuous interpolation
	smc_conti_arc_move_center_unit	Spiral interpolation instruction based on circle center + end point arc extension in continuous interpolation (it can be used for two-axis arc interpolation).
	smc_conti_arc_move_radius_unit	Cylindrical spiral interpolation instruction based on radius + end point arc extended in continuous interpolation (it can be used for two-axis arc interpolation).
	mc_conti_arc_move_3points_unit	The cylindrical spiral interpolation instruction based on three-point arc extended in continuous interpolation (it can be used for two-axis and three-axis arc interpolation).
	smc_conti_pmove_unit	Control and specify the motion instruction of the outer axis in continuous interpolation.
Detection of continuous interpolation state.	smc_conti_remain_space	Inquiry the remaining interpolation space of continuous interpolation buffer zone.
	smc_conti_read_current_mark	Read the current interpolation segment number of the continuous interpolation buffer zone
	smc_conti_get_run_state	Read the status of continuous interpolation motion
	smc_check_done_multicoor	Detect the status of continuous interpolation motion
Continuous interpolation of IO	smc_conti_set_pause_output	Set IO output status when continuous interpolation is paused or stopped due to a fault
	smc_conti_get_pause_output	Read IO output status settings when continuous interpolation is suspended or abnormally stopped
	smc_conti_wait_input	Wait for IO input in continuous interpolation
	smc_conti_delay_outbit_to_start	IO lagging output relative to the starting point of track segment in continuous interpolation (be executed in segment).
	smc_conti_delay_outbit_to_stop	IO lag output relative to track end point during continuous interpolation
	smc_conti_ahead_outbit_to_stop	Output IO in advance relative to the end point of track segment in continuous interpolation (be executed in segment).
	smc_conti_write_outbit	Instant IO output in buffer zone during continuous interpolation
	smc_conti_clear_io_action	Clear the unexecuted IO action in section
PWM control output.	smc_set_pwm_output	Set PWM to output parameters immediately.
	smc_get_pwm_output	Read current output parameters of PWM.
Universal IO	smc_read_inbit	Read the level of certain input port of the specified controller.

operation	smc_write_outbit	Set the level of certain output port of the designated controller.
	smc_read_outbit	Read the level of certain output port of the specified controller.
	smc_read_inport	Read the level of all input ports of the specified controller.
	smc_read_outport	Read the level of all output ports of the designated controller.
	smc_write_outport	Set the level of all output ports of the designated controller.
	smc_reverse_outbit	Delayed rotation of IO output
	smc_set_io_count_mode	Set IO counting mode
	smc_get_io_count_mode	Read setting of IO counting mode;
	smc_set_io_count_value	Reset IO counting
	smc_get_io_count_value	Read IO counting
Specific IO operation	smc_set_inp_mode	Sets the INP signal for the specified axis
	smc_get_inp_mode	Read the INP signal settings for the specified axis
	smc_set_alm_mode	Sets the ALM signal of specified axis
	smc_get_alm_mode	Read the ALM signal setting of the specified axis
	smc_write_sevon_pin	Control the output of servo enable port of the specified axis
	smc_read_sevon_pin	Read the level of servo enable port of the specified axis
	smc_write_erc_pin	Control the ERC signal output of the specified axis
	smc_read_erc_pin	Read the ERC port level status of the specified axis
	smc_read_alarm_pin	Read the ALARM port level of the specified axis
	smc_read_inp_pin	Read the INP port level of the specified axis
	smc_read_org_pin	Read the 0RG port level of the specified axis
	smc_read_elp_pin	Read the ELP port level of the specified axis
	smc_read_eln_pin	Read the ELN port level of the specified axis
	smc_read_emg_pin	Read the EMG port level of the specified axis
Handwheel	smc_handwheel_set_axislist	Set the specific motion axis under the selected level of the same axis
	smc_handwheel_get_axislist	Read the specific motion axis under the selected level of the same axis
	smc_handwheel_set_ratiolist	Set the rate level of selected handwheel of the same axis
	smc_handwheel_get_ratiolist	Read the rate level of selected handwheel of the same axis
	smc_handwheel_set_mode	Set the handwheel motion mode, whether it is in hardware or software mode.

	smc_handwheel_get_mode	Read the handwheel motion mode, and where motion is in hardware or software mode.
	smc_handwheel_set_index	Select or replace the handwheel motion axis selection and ratio gear.
	smc_handwheel_get_index	Select or replace the gear selection and ratio of the handwheel motion axis.
	smc_handwheel_move	Start handwheel motion
	smc_handwheel_stop	Stop handwheel motion
Encoder	smc_set_counter_inmode	Set the counting mode of encoder
	smc_get_counter_inmode	Read the counting mode of encoder
	smc_set_encoder_unit	Set the encoder pulse counting of designated axis
	smc_get_encoder_unit	Read the encoder pulse counting of designated axis
	smc_set_ez_mode	Set the EZ signal level of designated axis
	smc_get_ez_mode	Read the EZ signal level of designated axis
	smc_set_counter_reverse	Set the reverse phase of Phase AB count value
	smc_get_counter_reverse	Read the reverse phase mode of Phase AB count value
High speed position latch.	smc_set_ltc_mode	Set the LTC signal of designated axis
	smc_get_ltc_mode	Read the LTC signal setting of designated axis
	smc_set_latch_mode	Set the latch mode.
	smc_get_latch_mode	Read latch mode
	smc_get_latch_value_unit	Read the value of latch from the controller.
	smc_get_latch_flag	Read the latch times of the specified axis from the controller.
	smc_reset_latch_flag	Reset the mark bit of latch.
Origin point latch	smc_set_homelatch_mode	Set latch mode of original point
	smc_get_homelatch_mode	Read latch mode setting of original point
	smc_reset_homelatch_flag	Clear latch mark of original point
	smc_get_homelatch_flag	Read latch mark of original point
	smc_get_homelatch_value_unit	Read latch value of original point
EZ latch	smc_set_ezlatch_mode	Set EZ latch mode.
	smc_get_ezlatch_mode	Read mode setting of EZ latch.
	smc_reset_ezlatch_flag	Clear EZ latch mark.
	smc_get_ezlatch_flag	Read EZ latch mark.
	smc_get_ezlatch_value_unit	Read EZ latch value.
Uniaxial position comparis	smc_compare_set_config	Set one-dimensional position comparator
	smc_compare_get_config	Read settings of one-dimensional position comparator

on	smc_compare_clear_points	Clear all added comparison points of one-dimensional position.
	smc_compare_add_point_unit	Add one-dimensional position comparison points
	smc_compare_get_current_point_u nit	Read the current comparison point position.
	smc_compare_get_points_runned	Read the number of compared points.
	smc_compare_get_points_remained	Read the number of compared points.
Two-dimensional position comparison	smc_compare_set_config_extern	Set two-dimensional position comparator
	smc_compare_get_config_extern	Read the settings of two-dimensional position comparator.
	smc_compare_clear_points_extern	Clear all added comparison points of two-dimensional position.
	smc_compare_add_point_extern_un it	Add comparison points of two-dimensional position
	smc_compare_get_current_point_extern_unit	Read that position of the comparison point of the current two-dimensional position
	smc_compare_get_points_runned extern	Inquiry the number of two-dimensional comparison points which have been compared.
	smc_compare_get_points_remained _extern	Inquiry the number of two-dimensional comparison points which can be added
High speed position comparison.	smc_hcmp_set_mode	Set the high-speed comparison mode
	smc_hcmp_get_mode	Read the settings of high-speed comparison mode
	smc_hcmp_set_config	Configure the high-speed comparator
	smc_hcmp_get_config	Read the configuration of high-speed comparator
	smc_hcmp_clear_points	Clear all added comparison points of high-speed position
	smc_hcmp_add_point_unit	Add/update high-speed comparison position
	smc_hcmp_set_liner_unit	Set the parameters of high-speed comparison linear mode
	smc_hcmp_get_liner_unit	Read the parameter settings of high-speed comparison linear mode.
	smc_hcmp_get_current_state	Read high-speed comparison status
	smc_write_cmp_pin	Control the output of designated CMP port
	smc_read_cmp_pin	Read the level of designated CMP port
Hardware limit	smc_set_el_mode	Set EL limit signal.
	smc_get_el_mode	Read EL limit signal setting.
	smc_set_softlimit_unit	Set soft limit.

	smc_get_softlimit_unit	Read the soft limit setting.
Function of motion abnormal stop	smc_stop	Stop of designated axis
	smc_stop_multicoor	Stop the motion of all axes in the coordinate system.
	smc_emg_stop	Emergency stop of all axes.
	smc_set_emg_mode	Set EMG emergency stop signal
	smc_get_emg_mode	Read setting of EMG emergency stop signal
	smc_set_io_dstp_mode	Set level signal of IO to trigger deceleration stop;
	smc_get_io_dstp_mode	Read IO trigger parameter of deceleration stop.
	smc_set_dec_stop_time	Set deceleration stop time.
	smc_get_dec_stop_time	Read the setting of deceleration stop time.
Axis IO mapping	smc_set_axis_io_map	Set parameters of axis IO mapping.
	smc_get_axis_io_map	Read parameters of axis IO mapping.
Virtual mapping of axis	smc_set_io_map_virtual	Set parameters of virtual IO mapping.
	smc_get_io_map_virtual	Read parameters of virtual IO mapping.
	smc_read_inbit_virtual	Read the level status of virtual IO port after filtering
Password management	smc_write_password	Encrypt
	smc_check_password	Password verification
	smc_enter_password	Login password
	smc_modify_password	Change login password.
File management	smc_download_file	Download local files to FLASH.
	smc_download_memfile	Download memory files to FLASH.
	smc_upload_file	Upload FLASH file to local file.
	smc_upload_memfile	Upload FLASH file to memory file.
	smc_download_file_to_ram	Download the local file to RAM, not saved after power failure.
	smc_download_memfile_to_ram	Download the memory file to RAM, not saved after power failure.
	smc_get_progress	File download progress
Register operation	smc_set_modbus_0x	Write bit register
	smc_get_modbus_0x	Read bit register
	smc_set_modbus_4x	Write word register
	smc_get_modbus_4x	Read word register
Analog operation	smc_get_ain	Read analog voltage value.
	smc_set_ain_action	Set analog parameters.

	smc_get_ain_action	Read the analog parameter values.
	smc_get_ain_state	Read the trigger status value of analog input.
	smc_set_ain_state	Set the trigger state of analog input.
BASIC-related function	smc_write_array	Write array values by index.
	smc_read_array	Read array values by index
	smc_modify_array	Modify array values by index
	smc_read_var	Read variable value
	smc_modify_var	Modify variable value
	smc_get_stringtype	Read variable type
	smc_basic_delete_file	Delete BASIC program.
	smc_basic_run	Run
	smc_basic_stop	Stop
	smc_basic_pause	Pause
	smc_basic_step_run	Single-step run
	smc_basic_step_over	Run to the next breakpoint
	smc_basic_continue_run	Continue running
	smc_basic_state	Current state
	smc_basic_current_line	Current execution line
	smc_basic_break_info	Breakpoint information
	smc_basic_message	Read output information
	smc_basic_command	Online command
G code-related function	smc_gcode_check_file	Check whether the file exists
	smc_gcode_delete_file	Delete a file
	smc_gcode_clear_file	Delete all files
	smc_gcode_get_first_file	Read the first file name
	smc_gcode_get_next_file	Read the next file name
	smc_gcode_start	Start
	smc_gcode_stop	Stop
	smc_gcode_pause	Pause
	smc_gcode_state	Read current status
	smc_gcode_set_current_file	Set current file
	smc_gcode_get_current_file	Read current file name
	smc_gcode_current_line	Read the current running line
	smc_gcode_get_file_profile	Read attributes of G file.

Busbar-related function	nmcs_reset_canopen	Reset CANopen bus.
	nmcs_stop_etc	Stop EtherCAT bus running.
	nmcs_axis_io_status	Acquire axis IO status.
	nmcs_get_LostHeartbeat_Nodes	Acquire heartbeat message information, and support CANopen and EtherCat.
	nmcs_get_EmergeneyMessege_Nodes	Acquire emergency message information, and support CANopen and EtherCAT.
Busbar-related function	nmcs_set_node_od	Set the slave object dictionary.
	nmcs_get_node_od	Acquire the slave object dictionary.
	nmcs_SendNmtCommand	Send NMT management message.
	nmcs_get_cycletime	Read EtherCAT bus cycle.
	nmcs_write_rxpdo_extra	
	nmcs_read_rxpdo_extra	
	nmcs_read_txpdo_extra	
	nmcs_get_axis_type	Read bus axis type.
	nmcs_set_axis_enable	Enable EtherCAT bus driver.
	nmcs_set_axis_disable	Disable EtherCAT bus driver
	nmcs_syn_move	Synchronous motion
	nmcs_get_axis_state_machine	Read EtherCAT bus axis state machine.
	nmcs_get_total_axes	Read the number of EtherCAT bus axes and virtual axes.
	nmcs_get_total_adcnum	Read the number of AD/DA input/output ports of EtherCAT bus.
	nmcs_get_total_ionum	Read the number of I/O ports of EtherCAT bus.
	nmcs_get_total_adcnum	Acquire the analog I/O points of EtherCAT bus.
	nmcs_get_total_slaves	Acquire the total number of EtherCAT slave stations.
	nmcs_get_axis_io_out	Acquire the digital output IO status of EtherCAT axis.
	nmcs_set_axis_io_out	Set the digital output IO status of EtherCAT axis.
	nmcs_get_axis_io_in	Acquire the digital input IO status of EtherCAT axis.
	nmcs_write_outbit	Set the level of an output port of the designated controller or expansion module.
	nmcs_read_outbit	Read the level of an output port of a specified controller or expansion module.
	nmcs_read_inbit	Read the level of an input port of a specified controller or expansion module.
	nmcs_write_outport	Set the level of all output ports of specified IO

		group number.
	nmcs_read_outport	Read the level of all output ports of specified IO group number.
	nmcs_read_inport	Read the level of all input ports of specified IO group number.
	nmcs_set_alarm_clear	Clear the alarm signal
	nmcs_get_errcode	Read EtherCAT bus status.
	nmcs_clear_errcode	Clear bus error.
	nmcs_get_card_errcode	Acquire bus error code of specified link number.
	nmcs_clear_card_errcode	Clear the bus error code of the specified link number.
	nmcs_get_axis_errcode	Acquire the specified axis bus error code.
	nmcs_clear_axis_errcode	Clear the bus error code of the specified axis number.

Table 2: List of Instruction Operation Error

When the BASIC program has syntax errors or parameter range errors, the BASIC program will be stopped, and the wrong position and information will be output at the same time. See the details in the figure below.

Error code	Error number	Possible reasons of error
Success	0	Successful operation
Unknown Error	1	Unknown error
Parameter Error	2	Parameter error
Operate Timeout	3	Operation timeout
State Busy	4	Status busy
Too Many Connections	5	Too many links.
Interpolation Error	6	Interpolation error
Connection Failure	7	Connection failed.
Cannot be connected	8	Unable to connect
Axis number is out of range	9	Card number or axis number is out of range.
Transport error	10	Data transmission error.
Firmwarefil eerror	12	Bad firmware file.
The firmware does not match	14	The firmware files do not match.
Firmware parameters error	20	Wrong firmware parameters.
The current state of firmware is not all owed to operate	22	Operation is not allowed in the current state of firmware.
The feature is not supported	24	The function is not supported.
Password error	25	Wrong password
The number of password inputs is limited	26	Password input times are limited.
ERR_AXIS_SEL_ERR	30	The axis gear selection of handwheel pulse is out of range (software control mode).
ERR_HAND_AXIS_NUM_ERR	31	Axis mapping quantity of handwheel pulse is out of range.
ERR_AXIS_RATIO_ERR	32	The ratio gear selection of handwheel pulse is out of range (software control mode).
ERR_HANDWH_START	33	Handwheel pulse mode has been entered, but software and hardware control modes cannot be switched.
ERR_AXIS_BUSY_STATE	34	The axis is in motion, fail to switch to handwheel mode.
ERR_LIST_NUM_ERR	50	The LIST number is out of range.

ERR_LIST_NOT_OEPN	51	The LIST is not initialized.
ERR_PARA_NOT_VALID	52	Parameter is not in valid range.
ERR_LIST_HAS_OPEN	53	The LIST has been opened.
ERR_MAIN_LIST_NOT_OPEN	54	The LIST is not initialized.
ERR_AXIS_NUM_ERR	55	Axis number is not in valid range.
ERR_AXIS_MAP_ARRAY_ERR	56	Axis mapping table is empty.
ERR_MAP_AXIS_ERR	57	Mapping axis error.
ERR_MAP_AXIS_BUSY	58	Mapping axis busy.
ERR_PARA_SET_FORBIT	59	Parameter changes are not allowed in motion.
ERR_FIFO_FULL	60	Buffer is full.
ERR_RADIUS_ERR	61	The radius is 0 or less than half of the distance between two points.
ERR_MAINLIST_HAS_START	62	The LIST has started.
ERR_ACC_TIME_ZERO	63	The acceleration and deceleration time are zero.
ERR_MAINLIST_NOT_START	64	The main LIST is not started.
ERR_POINT_SAME_ON_RADIUS	67	The starting and end point of arc interpolation cannot coincide in radius mode.
MCVP_SMOOTH_TIME_SET_ERROR	80	S time setting error (less than or equal to 0)
MCVP_START_VEL_SET_ERROR	81	Error in setting the absolute value of starting speed (less than 0).
MCVP_STEADY_VEL_SET_ERROR	82	Error in setting the absolute value of max. speed (less than or equal to 0).
MCVP_END_VEL_SET_ERROR	83	The absolute value of stop speed is set incorrectly (less than 0).
MCVP_TOTAL_LENGTH_SET_ERROR	84	Motion distance is 0, unable to move.
ERR_AXIS_INDEX	101	Selected axis exceeds the max. value.
ERR_SET_WHILE_MOVING	102	Axis is in motion, fails to set parameters.
ERR_ENTER_WHILE_MOVING	103	Axis is moving, fails to enter this mode.
ERR_PEL_STATE	104	The axis is in positive limit and fails to move forward.
ERR_NEL_STATE	105	The axis is in negative limit and fails to move in negative direction.
ERR_SOFT_PEL_STATE	106	The axis is in a soft positive limit and fails to move forward.
ERR_SOFT_NEL_STATE	107	The axis is in soft negative limit and fails to move in negative direction.
ERR_FORCE_IN_OTHER_MODE	108	Axis is in non-point position mode, and fails to be forcibly displaced.

ERR_MAX_VEL_ZERO	109	Error in setting the max. speed. It cannot be 0.
ERR_EQU_ZERO	110	Error in setting axis equivalent, cannot be 0.
ERR_BACKLASH_NEG	111	Error in setting the reverse clearance, which cannot be a negative value.
ERR_MAX_PULSE	112	Wrong setting position, which has exceeded the allowable range.
ERR_CMP_EXCEED_MAX_AXISES	121	Selected comparison axis is out of range.
ERR_CMP_EXCEED_MAX_INDEX	122	Comparison points are full, and fails to continue adding.
ERR_CMP_EXCEED_MAX_IO	123	Compared IO is out of range.
ERR_CMP_EXCEED_MAX_CHAN	124	Selected high-speed comparison IO is out of range.
ERR_MAP_AXISIO_MAX_AXISES	130	The mapped axis is out of range.
PVT_ERROR_AXISES_OVER_RANGE	140	Selected axis is out of range.
PVT_ERROR_INDEX_OVER_RANGE	141	Control point is full, and fails to continue adding.
PVT_ERROR_INDEX_EXCEED	142	Control point is full, and fails to continue adding.
PVT_ERROR_TIME_EROOR	143	The insertion time is 0 or negative.
HOME_ERROR_AXISES_OVER_RANGE	200	Selected axis exceeds the max. value.
HOME_ERROR_MAX_VEL	202	Set the max. speed of to 0.
HOME_ERROR_MAX_ACC	203	The set acceleration is less than or equal to 0.
HOME_ERROR_BOTH_LIMIT	207	Being in positive and negative limit at the same time, fail to start the motion back to zero.
ERROR_ZSHELL_PARAERR	1000	Parameter error
ERROR_ZSHELL_STOP_USER	1040	Stopped manually by user
ERROR_ZSHELL_STOP_OTHERTASK	1041	Other tasks are linked to stop.
ERROR_ZSHELL_PARA_CANNOT_MODIFIY	1042	A few parameters are not allowed to be modified, and the SET extension returns.
ERROR_ZSHELL_ARRAY_OVER	1043	Several groups are out of boundary
ERROR_ZSHELL_VAR_TOOMUCH	1044	Number of variables exceeds
ERROR_ZSHELL_ARRAY_TOOMUCH	1045	Number of arrays exceeds
ERROR_ZSHELL_ARRAY_NOSPACE	1046	Array has no space.
ERROR_ZSHELL_SUB_TOOMUCH	1047	Too many SUB
ERROR_ZSHELL_LABEL_NAMEERR	1048	Wrong identifier naming.
ERROR_ZSHELL_LABEL_TOOMANYCHARES	1049	Identifier naming is too long.

ERROR_ZSHELL_NO_RIGHTBRACKET	1050	There are no right brackets.
ERROR_ZSHELL_UNKOWNCHAR	1051	Unrecognized characters.
ERROR_ZSHELL_UNKOWN_LABEL	1052	Unrecognized name, encountered in expression.
ERROR_ZSHELL_STOP_INVALIDCMD	1053	Unidentified command
ERROR_ZSHELL_STOP_OVERSTACK	1054	Stack level exceeded.
ERROR_ZSHELL_OVER_RECURSION	1055	Excessive recursion
ERROR_ZSHELL_NO_QUOTEEND	1056	Quotes are not ended
ERROR_ZSHELL_CMD_CANNOTREAD	1057	Fail to be read, fail to be used in expressions.
ERROR_ZSHELL_CMD_CANNOTRUN	1058	Functions and the like fail to appear at the beginning of a line and can only be used in expressions.
ERROR_ZSHELL_LINE_MUST_END	1059	Ended at expected line, some special instructions are needed.
ERROR_ZSHELL_ARRAY_NEEDINDEX	1060	Arrays need to be numbered, and parameters are also used.
ERROR_ZSHELL_NOTBRACKET_AFTERVAR	1061	No number is required after the variable.
ERROR_ZSHELL_DIM_CONFLICT	1062	Conflicting array redefinition, inconsistent length.
ERROR_ZSHELL_DIM_ARRAYLENGTHERR	1063	Wrong array length.
ERROR_ZSHELL_DIM_LABEL_SUB	1064	Definition, name SUB
ERROR_ZSHELL_DIM_LABEL_PARA	1065	Definition, name PARA.
ERROR_ZSHELL_DIM_LABEL_RESERVE	1066	Definition, name reservation.
ERROR_ZSHELL_UNWANT_CHAR	1067	Characters not allowed to appear.
ERROR_ZSHELL_STACK_NOPUSH	1068	There is no stack in POP.
ERROR_ZSHELL_FORMAT_ERR	1070	Formal error
ERROR_ZSHELL_SET_OVER	1071	Parameter overflow, para(10)10 is too large.
ERROR_ZSHELL_PARA_RANGEERR	1072	Some functions and commands have the wrong parameter range.
ERROR_ZSHELL_PARA_TOOMANY	1073	Some functions and commands have too many parameters.
ERROR_ZSHELL_PARA_TOOFEW	1074	Some functions and commands have too few parameters.
ERROR_ZSHELL_NO_EXPR	1075	Unable to read expression.
ERROR_ZSHELL_OPERNOPARA	1076	Operator has no parameters.
ERROR_ZSHELL_NOPARA_BEFOREOPER	1077	There is no parameter before the operator.
ERROR_ZSHELL_SIGNAL_CANNOTINEXPR	1078	Symbols cannot be used in expressions.

ERROR_ZSHELL_NEED_OPER	1079	Binocular operator is required.
ERROR_ZSHELL_SUB_NOTSUB	1080	CALL is not SUB.
ERROR_ZSHELL_NO_AUTO	1081	No AUTO, fail to start.
ERROR_ZSHELL_EQ_WANTED	1082	The assignment statement has no equal sign, and variables or parameters can be assignment statements only.
ERROR_ZSHELL_FILE_VAIN	1083	Program file is empty.
ERROR_ZSHELL_SUB_RENAME	1084	SUB has duplicate names, including duplicate names with other names.
ERROR_ZSHELL_TASK_RUNNING	1085	Task is already running.
ERROR_ZSHELL_NEED_COMMA_BETWEEN_PARA	1086	Commas are required between operands.
ERROR_ZSHELL_NO_LEFTBRACKET	1087	There are no right brackets.
ERROR_ZSHELL_TOOMANY_IFNESTED	1088	IF is nested too much.
ERROR_ZSHELL_TOOMANY_LOOPNESTED	1089	LOOP is nested too much.
ERROR_ZSHELL_MOVEAXISES_FEW	1090	Too few interpolation axes.
ERROR_ZSHELL_CONST _VAR	1091	Variable fails to be modified.
ERROR_ZSHELL_NOT_ONLINECMD	1092	Not used as an online command.
ERROR_ZSHELL_AXIS_OVER	1093	Axis number exceeded.
ERROR_ZSHELL_CRD_OVER	1094	Interpolation system exceeded
ERROR_ZSHELL_STOP_UNKNOWN	1099	Unknown errors, which are not likely to occur, and caused by internal errors in general.
ERROR_ZSHELL_DIVISION_BY_ZERO	1200	Error in dividing by zero, please check whether the divisor is 0.