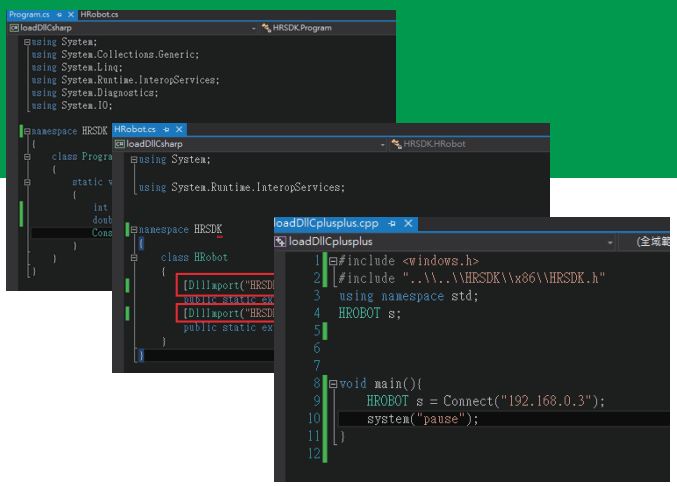
HIWIN®

Robot Software Development Kit HRSDK

For HRSS 3.3 Version
User Manual



Original Instruction

www.hiwin.tw

HIWIN® INDUSTRIE 4.0 Best Partner

Multi-Axis Robot

Pick-and-Place / Assembly /
Array and Packaging / Semiconductor /
Electro-Optical Industry /
Automotive Industry / Food Industry

- Articulated Robot
- Delta Robot
- SCARA Robot
- Wafer Robot
- Electric Gripper
- Integrated Electric Gripper
- Rotary Joint

Single-Axis Robot

Precision / Semiconductor /
Medical / FPD

- KK, SK
- KS, KA
- KU, KE, KC

Torque Motor Rotary Table

Medical / Automotive Industry /
Machine Tools / Machinery Industry

- RAB Series
- RAS Series
- RCV Series
- RCH Series

Ballscrew

Precision Ground / Rolled

- Super S Series
- Super T Series
- Mini Roller
- Ecological & Economical Lubrication Module E2
- Rotating Nut (R1)
- Energy-Saving & Thermal-Controlling (Cool Type)
- Heavy Load Series (RD)
- Ball Spline

Linear Guideway

Automation / Semiconductor / Medical

- Ball Type--HG, EG, WE, MG, CG
- Quiet Type--QH, QE, QW, QR
- Other--RG, E2, PG, SE, RC

Bearing

Machine Tools / Robot

- Crossed Roller Bearing
- Ballscrew Bearing
- Linear Bearing
- Support Unit

DATORKER® Strain Wave Gear

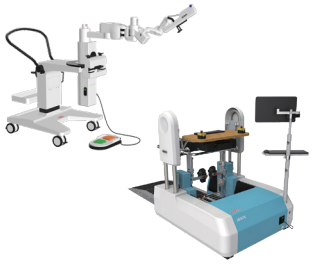
Robot / Automation Equipment /
Semiconductor Equipment / Machine Tools

- DSC-PO Type
- DSC-CO Type
- DSH-PH Type
- DSH-AH Type

AC Servo Motor & Drive

Semiconductor / Packaging Machine
/ SMT / Food Industry / LCD

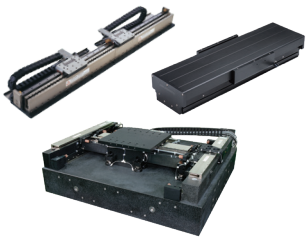
- Drives--D1, D2T/D2T-LM, E1
- Motors--50W~2000W



Medical Equipment

Hospital / Rehabilitation Centers /
Nursing Homes

- Robotic Gait Training System
- Robotic Endoscope Holder



Linear Motor Stage

Automated Transport /
AOI Application / Precision /
Semiconductor

- Iron-core Linear Motor
- Coreless Linear Motor
- Linear Turbo Motor LMT
- Planar Servo Motor
- Air Bearing Platform
- X-Y Stage • Gantry Systems
- Single-Axis Linear Motor Stage

Torque Motor & Direct Drive Motor

Machine Tools

- Torque Motor--
  TM-2/IM-2, TMRW Series

Inspection / Testing Equipment / Robot

- Direct Drive Motor--
  DMS, DMY, DMN, DMT Series

Table of Contents

1. Product Description

1.1. Function Description

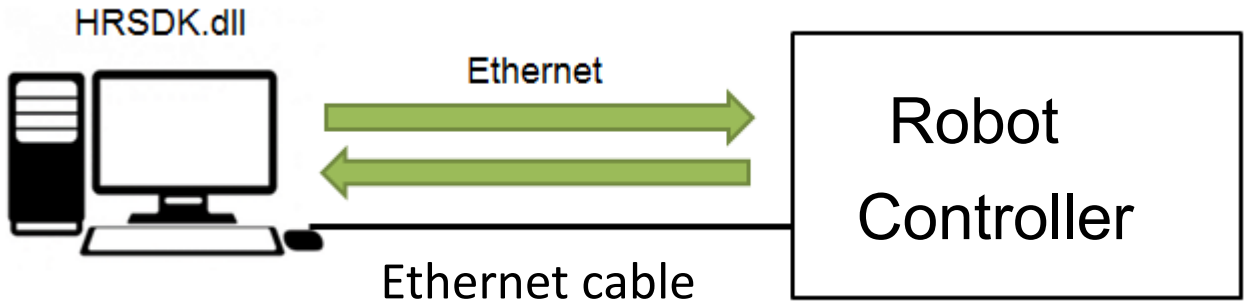


Figure 1-1 Simple system architecture diagram

HRSDK is stands for “HIWIN ROBOT Software Development Kit”. Programmers can develop applications for the autonomous control interface and processes to control the robot by using software database tools created by HIWIN, and can achieve the purpose of system integration.

- Applicable models include the six-axes GB and GC series controllers or the SCARA-LU series controllers; HRSDK databases and HRSS software version must be matched respectively as per the below table 1.3.

1.2. Hardware / Software Requirement

- Hardware Requirement:
  - ⇨ Hiwin Robot (or HRSS offline software)
  - ⇨ PC
  - ⇨ Ethernet Cable

- Software Requirement:
  - ⇨ Hiwin Robot Software System: HRSS V3.2.5 above & HRSDK function (robot controller)
  - ⇨ Program Integration Development Environment, support C++, C#, VB (user client computer)

1.3. Software version correspondence descriptions

Software version correspondence table

HRSDK version	Applicable HRSS software version	Release date		HRSDK version	Applicable HRSS software version	Release date
1.0.0	V2.1.23 and above V3.1.6 and above	2017/02/06		2.2.9	V3.3.11	2020/10/23
2.0.0	V3.2.0 V3.2.1	2017/07/13		2.2.10	V3.3.12	2020/11/23
2.1.1	V3.2.2	2017/09/11		2.2.11	V3.3.13	2020/12/25
2.1.2	V3.2.5	2018/01/05		2.2.12	V3.3.14	2021/01/28
2.1.4	V3.2.5	2018/02/14		2.2.13	V3.3.15	2021/02/26
2.1.5	V3.2.8	2018/03/09		3.0.1	V3.3.16	2021/03/26
2.1.6	V3.2.11	2018/07/18		3.0.2	V3.3.18	2021/5/30
2.1.7	V3.2.13	2018/09/27		3.0.3	V3.3.19	2021/6/30
2.1.8	V3.2.15 and above	2019/05/15		3.0.4	V3.3.20	2021/09/01
2.1.9	V3.3.1	2019/09/09		3.0.5	V3.3.22	2022/2/01
2.1.10	V3.3.1	2019/09/09				
2.1.11	V3.3.1	2019/09/09				
2.2.0	V3.3.2	2020/01/01				
2.2.2	V3.3.3	2020/02/27				
2.2.3	V3.3.3	2020/03/20				
2.2.4	V3.3.3	2020/04/20				
2.2.5	V3.3.7	2020/07/10				
2.2.6	V3.3.8	2020/07/22				
2.2.7	V3.3.9	2020/08/22				
2.2.8	V3.3.10	2020/09/22				

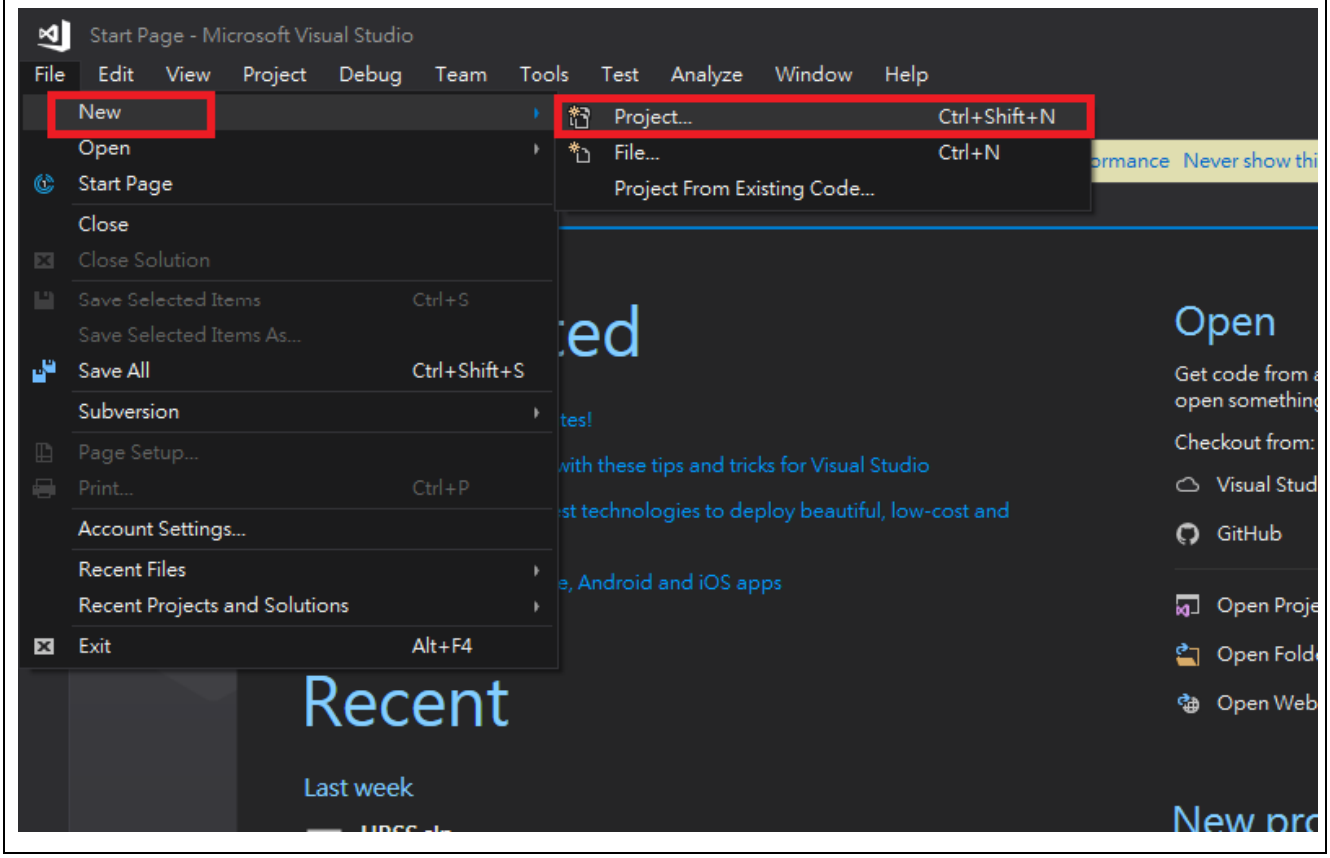
Supplementary description:

- The software architecture of HRSDK requires the DLL setting to correspond to the software version. The above table shows the corresponding versions; for example: HRSDK 2.2.11 must correspond to HRSS version 3.3.13, and so on.
- HRSDK version 3.0.1 and above supports connection compatibility with HRSS version 3.3.16 and above, meaning that the version is downwards compatible unless there are major revisions.
- If the manual is updated, there will be an updated version of correspondence descriptions; if the manual is not updated, consult the original manufacturer first for the corresponding version.

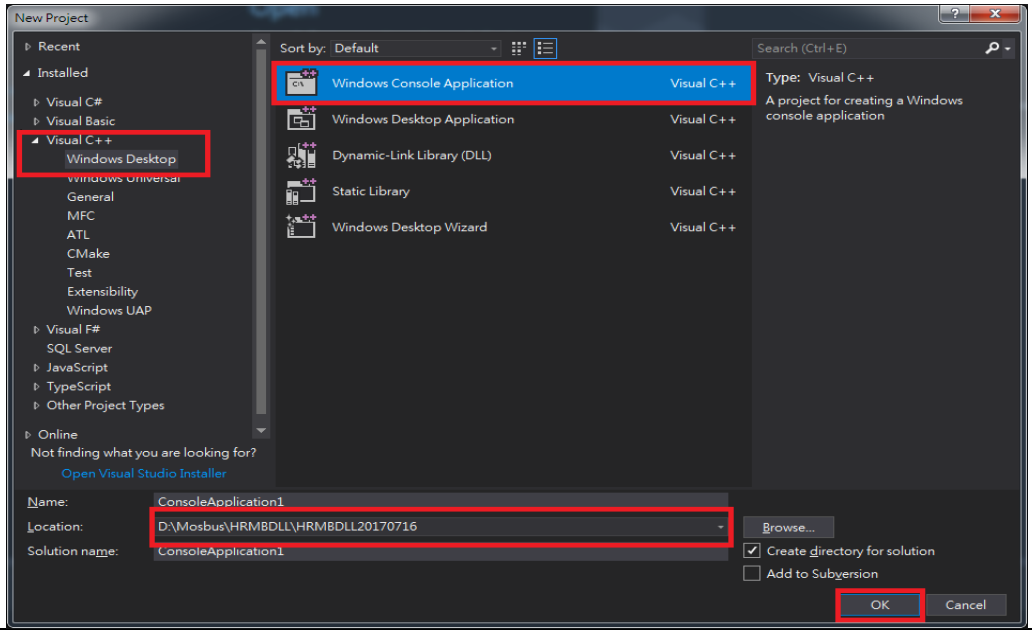
2. Preparations before importing database

2.1. C++ Solution Description

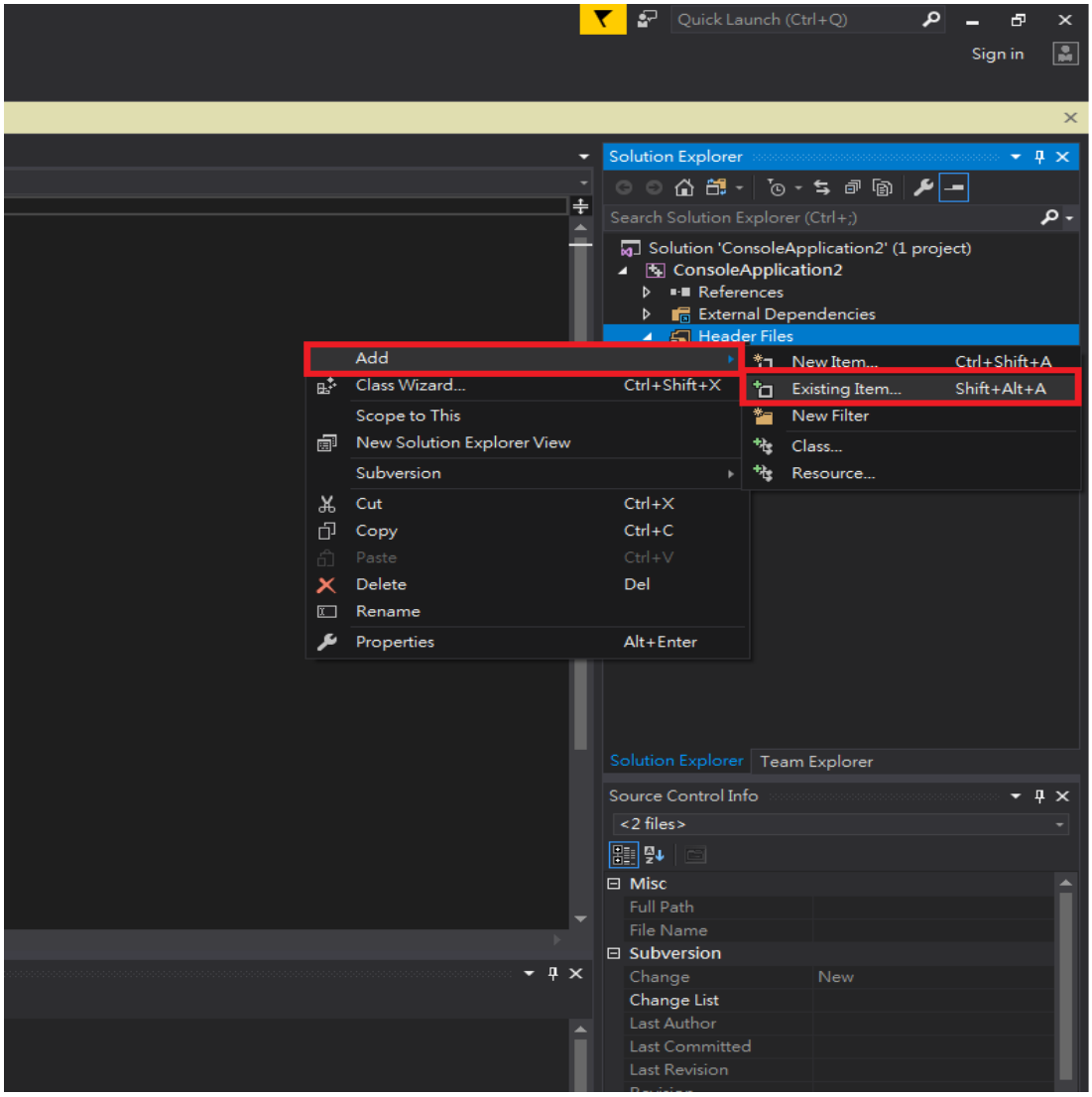
STEP 1: Add new project

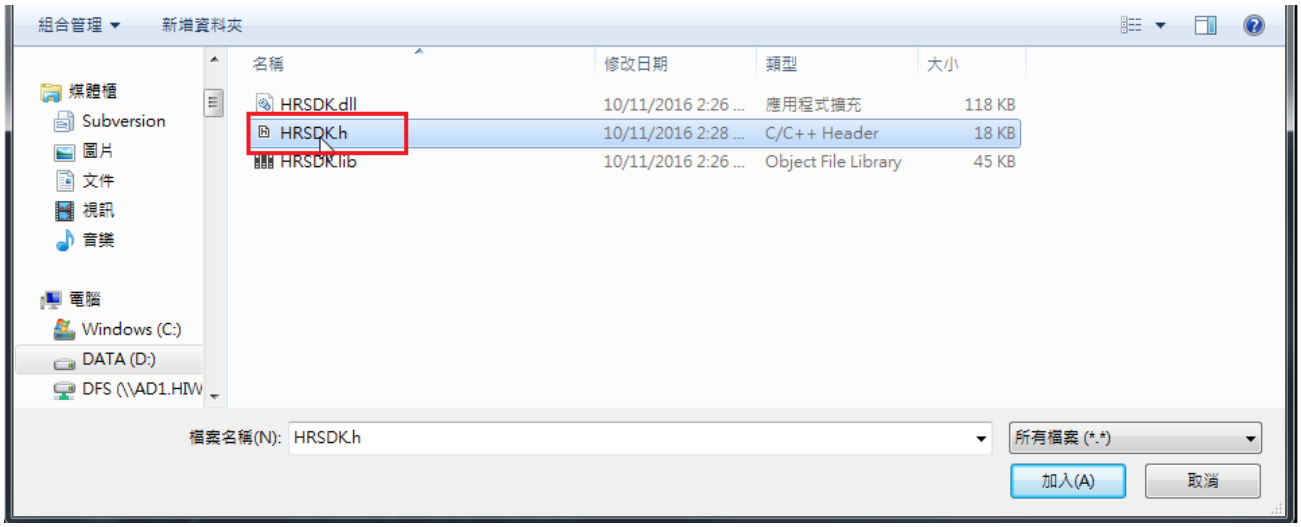


STEP 2: Select Visual C++ Win32 and set location

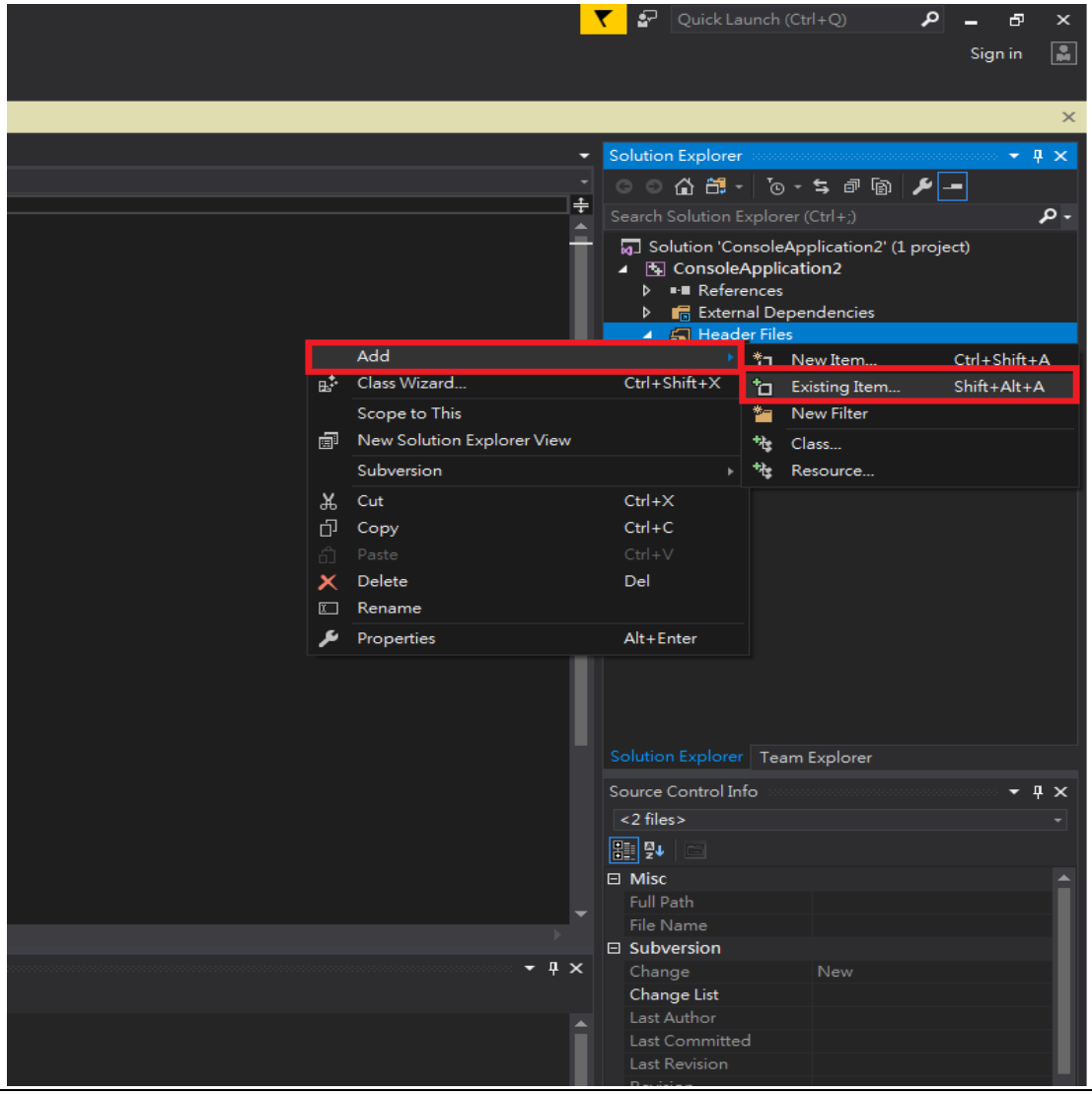


STEP 3: Add HRSDK.h into header file

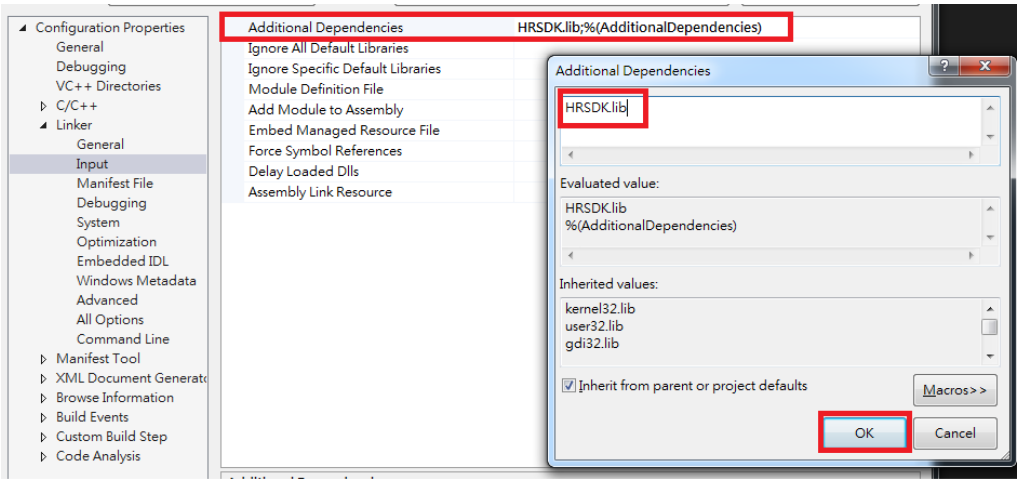




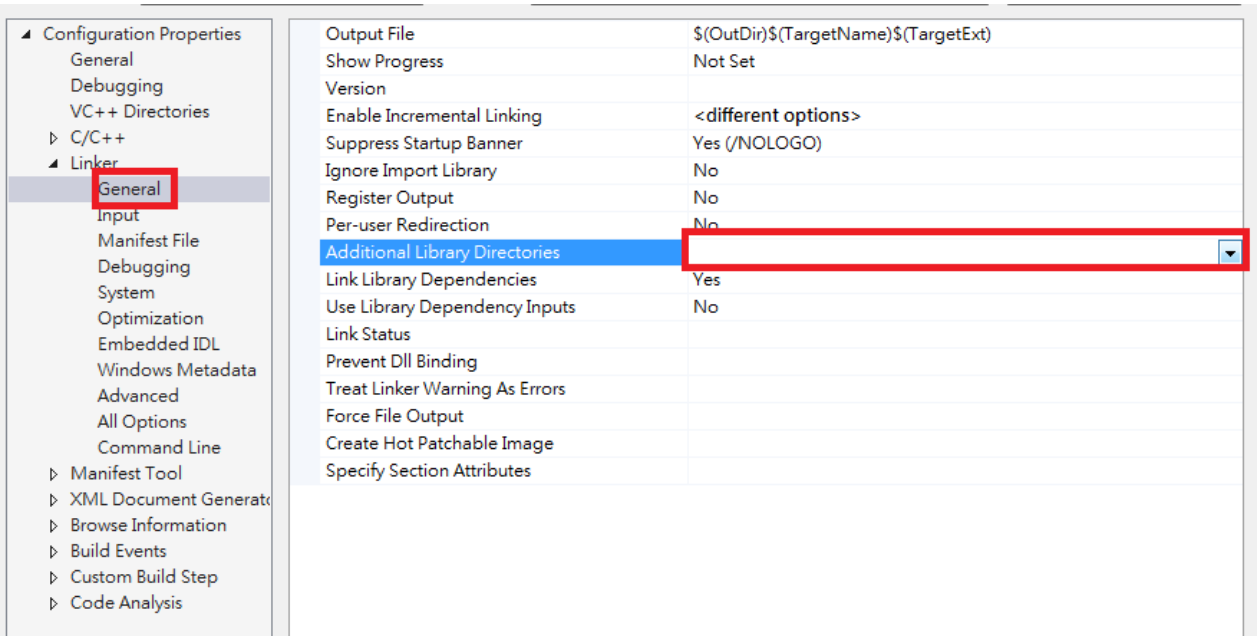
STEP 4: Right click project “Properties”



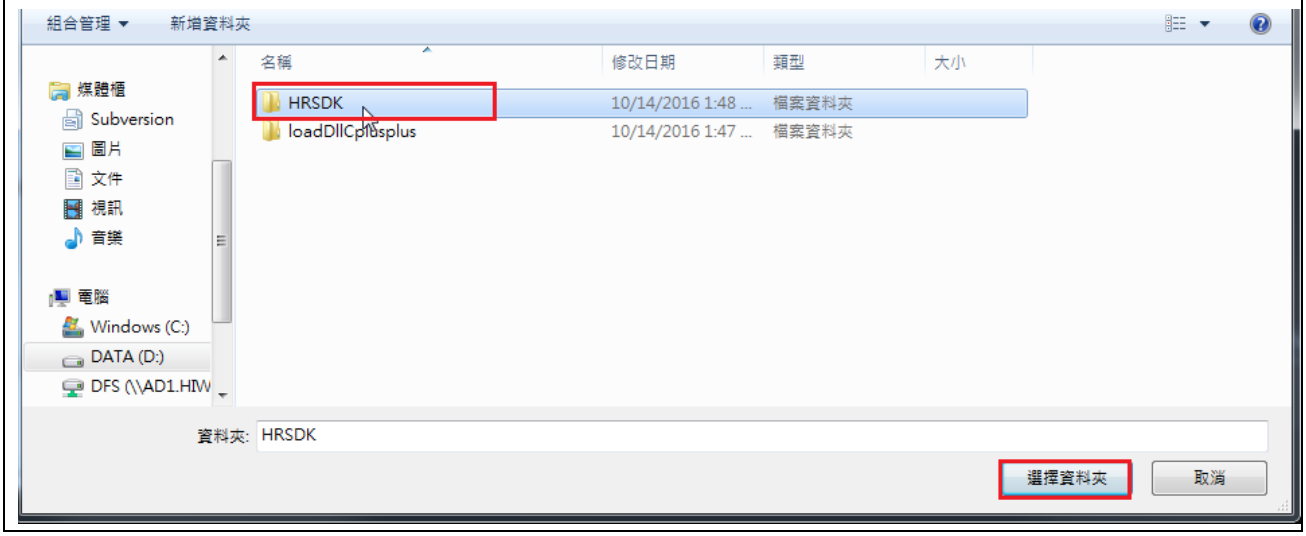
STEP 5: Select linker -> Input ->Additional Dependencies -> enter “HRSDK.lib”-> OK



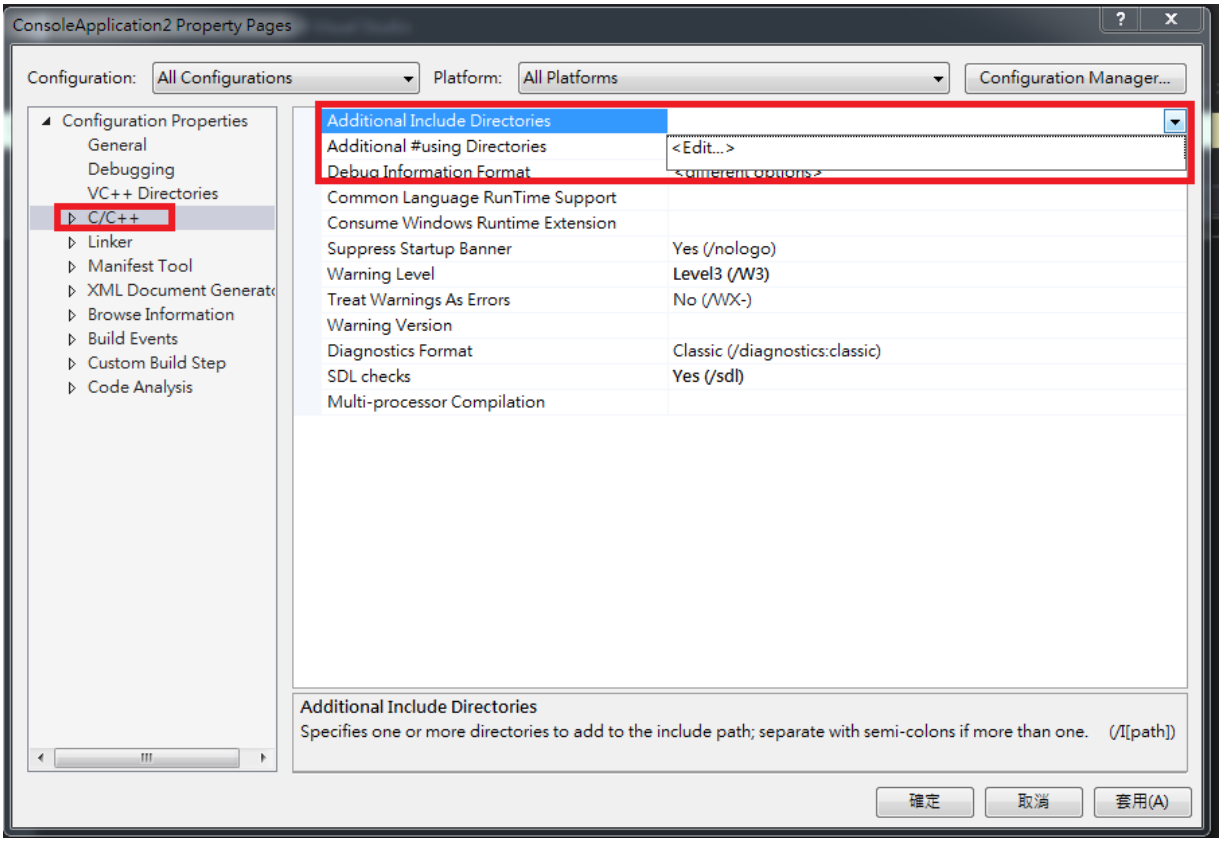
STEP 6: Linker -> General -> Additional Library Directories -> Add



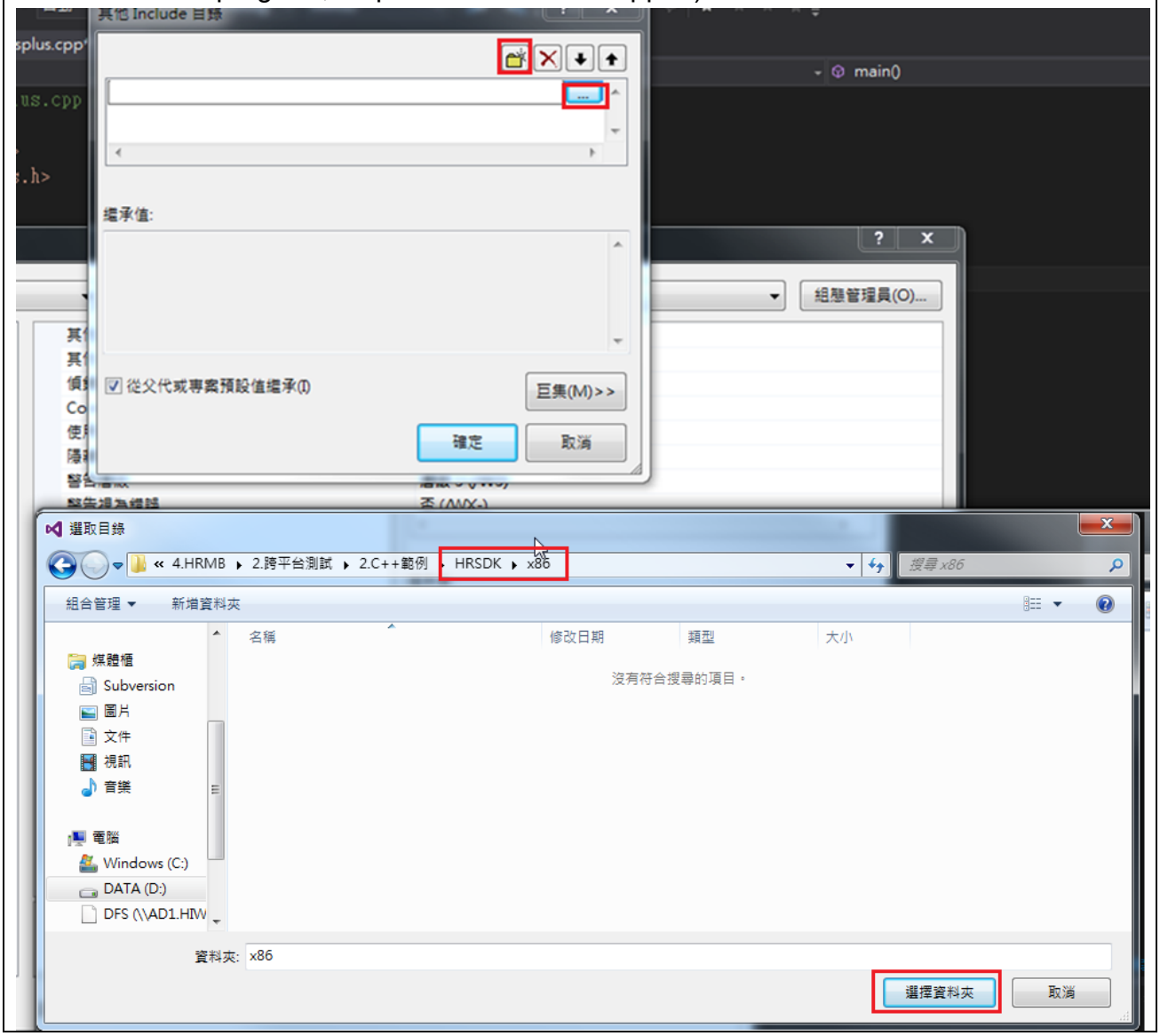
STEP 7: Select the file where dll is located (If dll is under the same root directory of the program, step 6 and 7 can be skipped)



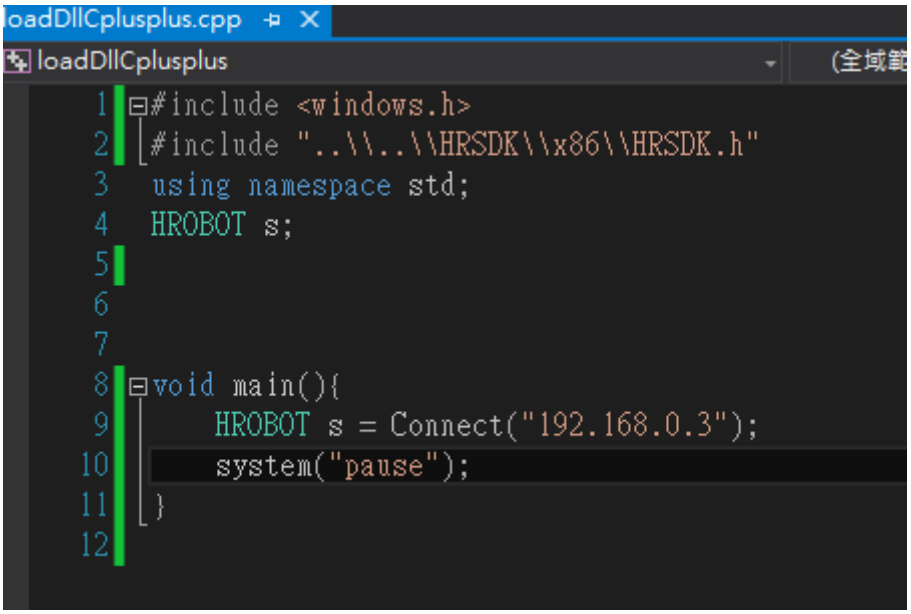
STEP 8: C/C++ -> Additional include directories -> Edit



STEP 9: Select directory ->Select HRSDK file (If .h file is under the same root directory of the program, step 8 and 9 can be skipped)



STEP 10: Include HRSDK.h header file -> start using dll



2.2. C# Solution Description

Follow the below 5 steps to start writing the Visual Studio C# program.

1. Download the latest HRSDK kit from the official website; it must include the HRSDK.dll and HRSDK.h files.
2. Open the Visual Studio C# program, and create a file <project name>.cs.
3. Copy HRSDK.dll into the project path, and also add the HRobot.cs file.
   The path is as follows: .../<project name> /bin/Debug
   (.../<project name> /bin/Release)
4. Make sure that the namespace is the same as the prefix of the added file; it must be changed if they are different.
5. Start using the dll database (HRobot.XXX).

■ Step 1: Go to the official website of HIWIN: http://www.hiwin.tw/, click on the robot model to download, and click on the robot software development kit (HRSDK) software data.

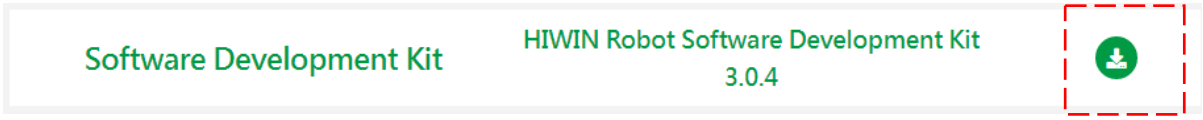


Figure 2-1 Data map for downloading the software development kit HRSDK

■ Step 2: (1). Open the Visual Studio C# program,
open file (F) -> New (N) -> Project (P).

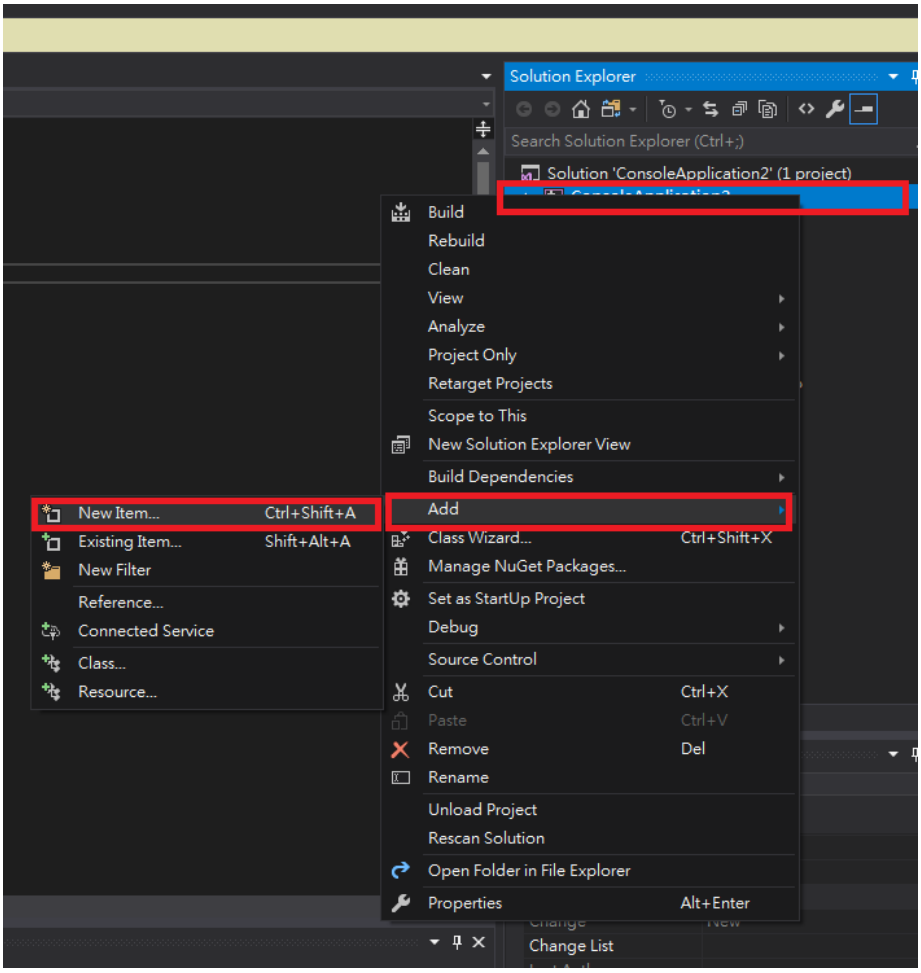


Figure 2-2 Open Visual Studio C# program project diagram

(2). Select the console application. (No interface, only text display; select Windows Forms App if you need the interface).

(3). Change the name and project name -> Press “OK” to create the file.

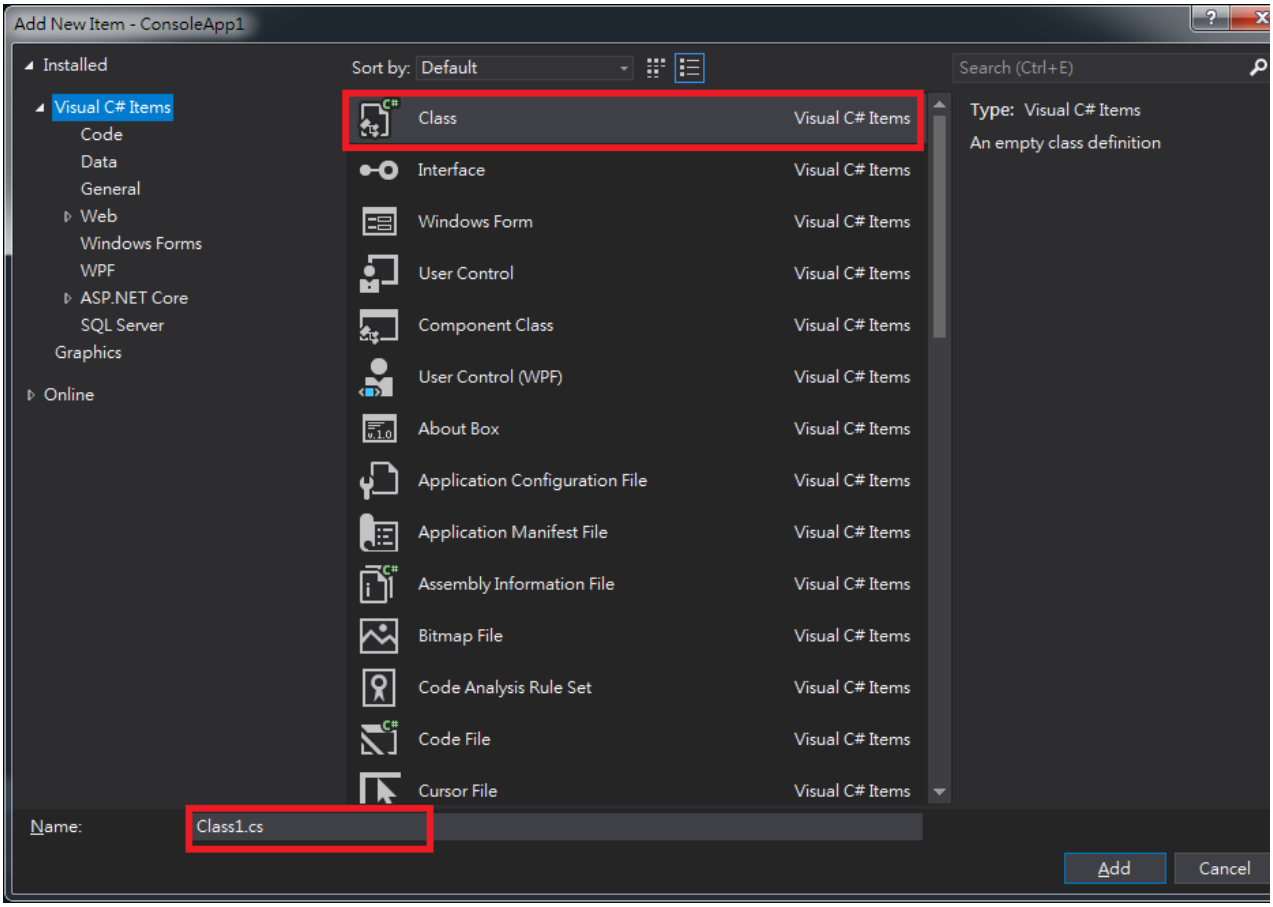


Figure 2-3 Console application program diagram

■ Step 3: Copy the HRSDK.dll and HRobot.cs files from the downloaded folder into the following paths:

.../<project name> /bin/Debug/

(.../<project name> /bin/Release/)

Figure 2-4 Copy the HRSDK.dll & HRobot.cs files into the specified folder

■ Step 4: (1) Project (P) -> Add existing item (G) -> Select the HRobot.cs file (.../<project name> /bin/Debug/)

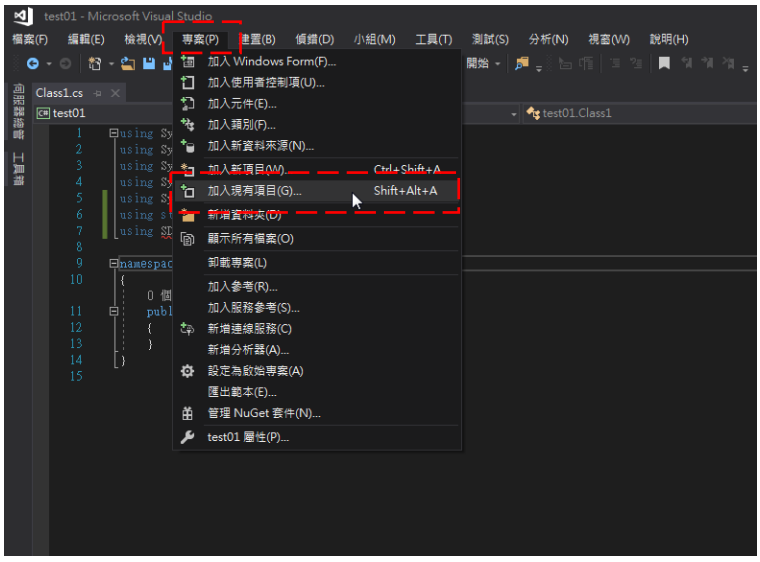


Figure 2-5 Add the HRobot.cs file

(2). Confirm that the HRobot.cs file exists from the project manager.

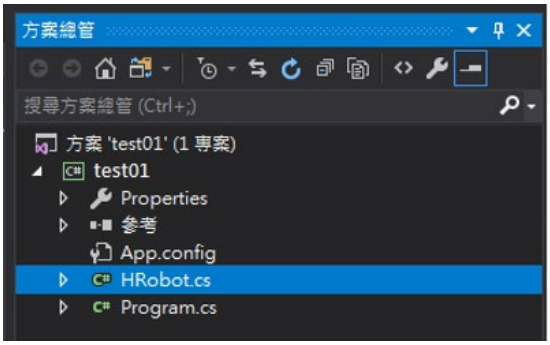


Figure 2-6 Confirm that the HRobot.cs file exists from the project manager

(3). Open the HRobot.cs file, and confirm that the prefix is SDKHrobot { ....}; (default name).

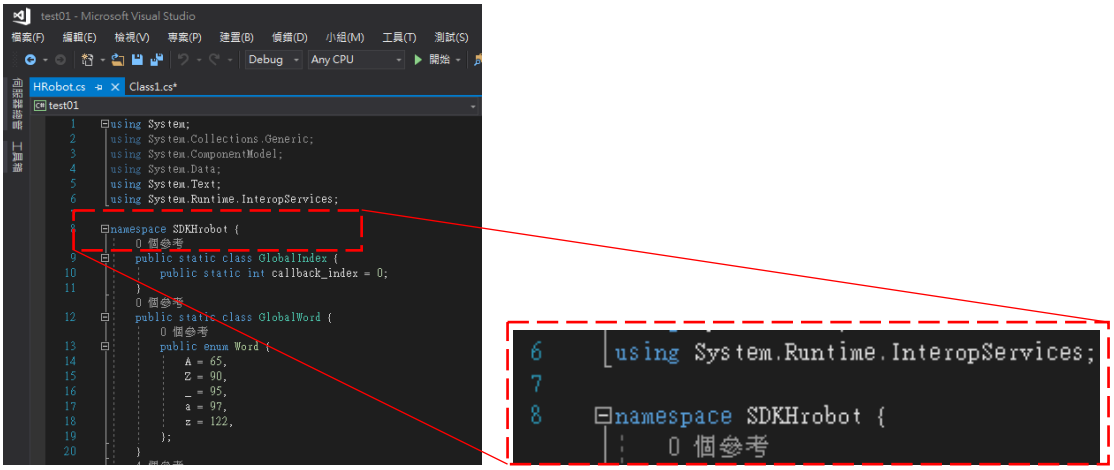


Figure 2-7 Confirm that the prefix is SDKHrobot

■ Step 5: (1). Add the header file to the Program.cs file in the program editor by using SDKHrobot.

(2). Type “HR” in the main program to test whether the HRobot database can be called; if it can, it means that it is successful and the HRSDK database can be used now.

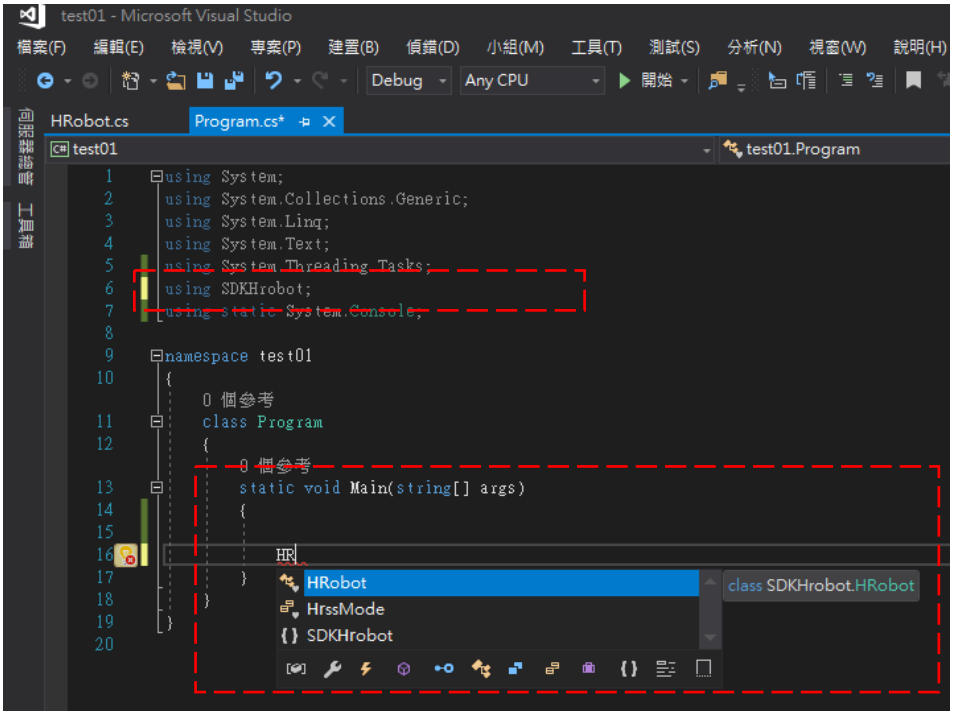


Figure 2-8 Adding the Program.cs file to header file

(3). Use the keyboard to enter a sample program and execute the program (F5) to test whether the sample program can run.

```
StringBuilder SDK_ver = new StringBuilder ();
SDKHrobot.HRobot.get_hrsdk_version (SDK_ver);
Console.WriteLine(" SDK_ver= " + SDK_ver);
Readykey ();                (Can omit)
```

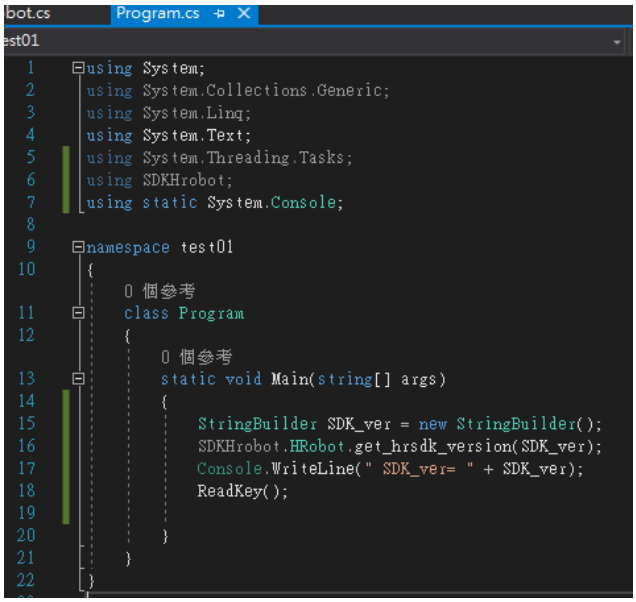


Figure 2-9 Enter sample program diagram

2.3. **Visual Basic (VB) Solution Description**

STEP 1: Put HRSDK.dll into project/bin/Debug (Release) route

STEP 2: Once function is declared, you can start using HRSDK.dll

```
Module1.vb
Module1
(宣告)
Imports System.Text
Imports System.Runtime.interopSvervice


Module Module1
    Public Declare Function StartRobot Lib "HRSDK.dll" Alias "Connect" (ByVal ip As String) As UInteger
    Public Declare Function set_motor_state Lib "HRSDK.dll" (ByVal robot As UInteger, ByVal state As Integer) As Integer
    Public Declare Function ptp_axis Lib "HRSDK.dll" (ByVal robot As UInteger, ByVal point As Double()) As Integer
    Public Declare Ansi Function set_mode Lib "HRSDK.dll" (ByVal robot As UInteger, ByVal p2 As Integer) As Integer

    Sub Main()
        Dim Name As String
        Dim stringBuilder As String
        stringBuilder = "127.0.0.1"


        Dim RobotID As UInteger
        RobotID = StartRobot(stringBuilder)
        Dim targetpoint As Double() = {0, 0, 0, 0, -90, 0}
        ptp_axis(RobotID, targetpoint)
        Console.Write(" Continue.....")
        'Console.ReadLine()
```

2.4. Connection class description

The connection class is divided into two identities: observer and operator.

- Observer: Only part of the application interface can be used; able to connect while HRSS is running and the number of connections is less than the maximum allowable number of 4 connections.
- Operator: Able to use all application interfaces; must connect when the operation mode of HRSS is set as EXT mode. Switching modes in HRSS will result in the connection class changing to the observer class.

2.5. Operation mode descriptions

There are two operation modes: manual mode and auto mode.

- **Manual mode function descriptions**

(1). Suitable for test runs, programming and teaching
(2). The linear motion speed is limited up to 250mm/s.
(3). The ratio of the point-to-point motion speed will be limited according to the robot model number
(4). Each time it is turned on or off, the overall robot speed is set as 10%, and the linear motion speed will be set as 250mm/s.
(5). The set_operation_mode instruction can be used to switch to manual mode.
(6). Jog operations are allowed

Precautions for manual mode:

Manual mode is used for debugging. Debugging includes setting, installing, adjusting, modifying or troubleshooting; the safety speed limit function should be enabled for the articulated robot. Precautions under the safety speed limit function:

(1). New or modified processes must be tested under environments with the safety function enabled.
(2). Tools, articulated robots or external axes are not allowed to touch or extend to the protective fences of the protective area.
(3). If workpieces, tools or any components are stuck or dropped, or if there are any mechanical failures or short-circuited appliances, activation of the articulated robot is not allowed.
(4). All debugging tasks must be performed when the people are outside the protected/fence/protection area.

- **Auto mode descriptions**

The auto mode function must comply with the following safety protection measures:

(1). All protective measures must be tested and installed.
(2). All suspended protective measures should have their functions resumed.
(3). Staff are not allowed to remain in the operation area.
(4). Standard operation rules must be followed.
(5). If the robot stops due to unknown reasons, the danger area can only be entered after the emergency stop button has been pressed.
(6). The disconnect function of the SDK can be used to allow auto mode to resume operation. Otherwise, the current motion will stop when there is abnormal disconnection.

3. Format command description and example references

All commands are divided into 12 classifications based on their different functions:

Classification item	Classification content	Corresponding chapter
1	Connection command class	3.1
2	Register command class	3.2
3	System variable command class	3.3
4	Input and output command class	3.4
5	Coordinate system command class	3.5
6	Task command class	3.6
7	File management command class	3.7
8	Controller command class	3.8
9	Jog command class	3.9
10	Motion command class	3.10
11	Articulated robot information command class	3.11
12	Communication setting command class	3.12

The following sections describe the instructions of commands classification and corresponding identities.

3.1. Connection Command Class

Reference function names and corresponding identities are shown in the below table:

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
open_connection	To connect with the HRSS system	O	O	O
disconnect	Disconnect with the HRSS system	O	O	O
set_connection_level	Set connection level	O	O	O
get_connection_level	Get connection level	O	O	O
get_hrsdk_version	Get HRSDK version number	O	O	O
get_hrss_sdkver	Get HRSS version number	O	O	O
get_hrsdk_sdkver	Get local HRSDK version number	O	O	O

The following section will describe the various function, instruction formats, parameter settings and example descriptions.

3.1.1. Open connection- open_connection

HROBOT open_connection (**const char*** address, **int** level, **callback_function** function)

Parameter	Data type	Description
address	const char*	Device IP address
level	int	Level of connection 0: Operator 1: Expert
function	void __stdcall FunName(uint16_t uint16_t uint16_t* int);	Event receive function, used to receive message return by controller
Return value	HROBOT	Success: Device ID (0-3 as valid device id) Fail: -1 connection is not set up -2 function is undefined -3: Cannot connect to robot: -4: Version no match

- **When the controller receives the first operator connection request, the controller will be in manual mode. All following connections will be observers.**
- Speed limit function
  - The speed of linear motion will be limited at 250mm/s
  - The speed ratio of point to point (PTP) motion will be limited by the type of model
  - Overall robot speed will be set at 10% each time when it is switched on or off, speed of linear motion will be set to 250mm/s
  - Speed_limit_off and speed_limit_on command can be used to switch off or on the speed limit function

3.1.2. Disconnect- disconnect

void disconnect (**HROBOT** robot)

Parameter	Data type	Description
robot	HROBOT	To interrupt the device ID

```
C++:
  void __stdcall FuncName (uint16_t uint16_t uint16_t* int);
  void main () {
    HROBOT robot;
    robot=open_connection ("192.168.0.3", 1, FuncName);
    //Do something
    disconnect (robot);
  }
  void __stdcall CallBackFun (uint16_t cmd, uint16_t rlt, uint16_t* msg, int len) {
    // process information from controller
  }
```

3.1.3. Set connection level- set_connection_level

int set_connection_level (HROBOT robot, int mode)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Connection mode 0: Observer 1: Operator
Return value	int	Success: 0 Fail: error code

```
C++:
  int mode = 1;
  set_connection_level (robot, mode);
```

3.1.4. Get connection level- get_connection_level

int get_connection_level (HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	0: Operator 1: Expert

```
C++:
  int level;
  level = get_connection_level(robot);
```

3.1.5. Get HRSDK version number- get_hrsdk_version

void get_hrsdk_version (**char*** **version**)

Parameter	Data type	Description
version	char*	HRSDK version number
Remark	➢ This version is a release version number. ➢ Version 3.0.X supports HRSS 3.3.16 and above connection is compatible. ➢ Please note that new instructions cannot be used on old versions of SDK. Please use the new functions with the new version of SDK	

C++:

```
char* sdk_version = new char [256];
get_hrsdk_version (sdk_version);
std:cout << "HRSDK Version: " << sdk_version << std:endl;
delete [] sdk_version;
```

3.1.6. Get HRSDK connection version- get_hrsdk_sdkver

void get_hrsdk_sdkver (int& large_ver, int& small_ver, int& revision)

Parameter	Data type	Description
large_ver	int	Large version connection number
small_ver	int	Small version connection number
revision	int	Version stamp
Remark	➢ Starting from SDK 3.0.1 and HRSS 3.3.16, the SDK connection will be paired with the robot connection based on this version number. The following situations will occur when the robot and SDK versions do not match. ➢ Connection cannot be made when the connection number of the large version is different; the SDK must be updated in order to be used. ➢ Connection can be made normally if the connection number of the small version is different; however, the new instructions cannot be used on the old versions of SDK. ➢ Version stamp; no special function.	

C++:

```
int client_L=0, server_L=0;
int client_s, server_s=0;
int rev;
get_hrsdk_sdkver (client_L, client_s, rev);
get_hrss_sdkver (robot, server_L, server_s, rev);

if (client_L != server_L){
    printf("Large version does not match and can't be connected.");
}else if(client_s != server_s){
    printf("small version does not match.");
}
```

3.1.7. Callback message descriptions- callback_function

void callback_function(uint16_t cmd, uint16_t rlt, uint16_t* msg, int len)

Parameter	Data type	Description
cmd	uint16_t	callback instruction command number
rlt	uint16_t	callback instruction's implementation process results
msg	uint16_t*	Callback instruction Strings
len	int	Callback string length
Remark	get_current_position must be executed first, for the mechanism to activate	

Command (Cmd)	Result (rlt)	Description	
0	4030	Alarm and error code occurred	
0	4031	Battery warning	
0	4032	Battery alarm	
0	4033	Battery normal	
0	4034	Network communication message	
0	4035	RS232 communication message	
0	4036	Modbus Monitor message	
0	4144	System Input value changed	
0	4145	System Output value changed	
0	4700	The row number of the program is currently executing	
0	4702	When the following information changes, the following information will be returned	
		hrss_mode	HRSS mode
		operation_mode	Operation modes
		override_ratio	Overall speed
		motor_state	Servo status
		exe_file_name	Execution file name
		FO	Function Output
		alarm_count	Number of alarms
		keep_alive	Stay connected
		motion_status	Motion status
		payload	Load value
		speed	T1 and T2 speed

Command (Cmd)	Result (rlt)	Description	
		The following additional information will be sent if the motion stopped	
		position coor joint encoder ext_pos ext_enc	Whether the coordinates changed Cartesian coordinates Axis coordinates Coded value External axis position External axis coded value
0	4703	Timer value changed	
0	4704	Counter value changed	
0	4705	Module Input value changed	
0	4706	Module Output value changed	
0	4707	FieldBus Input value changed	
0	4708	FieldBus Output value changed	
0	4710	PR changed, off_on, index, mode, position	
0	4711	Digital Input value changed	
0	4712	Digital Output value changed	
0	4713	Utilization history record information	
0	4714	Utilization start, when task_start starts executing	
0	4715	Utilization end, when task_start stops	
0	4716	Messages that HRSS sends to SDK	
0	4717	Which comments of PR have messages	
0	4718	External axis sensor limit light display	
1450	4028	Start clearing alarm	
1450	4029	End clearing alarm	
2162	4024	Delete HRSS backup file failed; file does not exist.	
2162	4025	Delete HRSS backup file successful	
2165	4044	Download Caterpillar failed	
4000	0	Execute ext_task_start	
4000	4013	ext_task_start already executing	
4001	2006	There already is motion executing before executing task_start	
4001	4011	task_start file open error	
4001	4012	task_start file name error	
4001	4013	There already is task executing before executing task_start	

Command (Cmd)	Result (rlt)	Description
4001	4014	task_start start executing
4004	4018	Stop task
4009	0	Download file
4009	201	Download file does not exist
4010	0	Upload file
4011	4020	HRSS upload file error
4011	4021	Update file send failed
4011	4022	HRSS update file failed
4011	4023	HRSS insufficient hardware capacity
4011	4026	HRSS start updating
4011	4027	HRSS update file sent successfully
4151	0	Articulated robot information
4202	0	TCP/IP communication connection successful
4202	9999	TCP/IP communication connection failed
4290	4047	External axis Mastering starts executing
4290	4048	External axis Mastering ended
4290	4049	External axis Mastering failed; there is an external axis currently moving.
4290	4050	External axis Mastering failed.
4701	1000	API instruction parsing failed
4709	501	Module IO saved successfully
63774	500	DIO set successfully

C++:

```
void __stdcall callBack(uint16_t cmd, uint16_t rlt, uint16_t* msg, int len) {
  switch (cmd)
  {
  case 0:
      if (rlt == 4030) {
          // HRSS_ALARM_NOTIFY
      } else if (rlt == 4031) {
          // HRSS_BATTERY_WARRING
      } else if (rlt == 4032) {
          // HRSS_BATTERY_ALARM
      } else if (rlt == 4033) {
```

```
                // HRSS_BATTERY_NORMAL
        }
        break;
        /* Clear Alarm */
case 1450:
        switch (rlt) {
        case 4028:
                // HRSS_START_CLEAR_ALARM
            break;
        case 4029:
                // HRSS_FINISH_CLEAR_ALARM
            break;
        default:
            break;
        }
        break;
        /* Task Start */
case 4001:
        switch (rlt) {
        case 4011:
                // ERROR_OPEN_FILE
            break;
        case 4014:
                // HRSS_TASK_START_FINISH
            break;
        default:
            break;
        }
        break;
        /* Update HRSS */
case 4011:
        switch (rlt) {
        case 4020:
                // HRSS_UPDATE_FILE_ERROR
            break;
        case 4021:
                // HRSS_UPDATE+FILE_TRANSFER_ERROR
            break;
```

```
        case 4022:
                // HRSS_UPDATE_FILE_UNARCHIVER_ERROR
            break;
        case 4023:
                // HRSS_HARD_DISK_CAPACITY_IS_NOT_ENOUGH
            break;
        case 4026:
                // HRSS_START_UPDATE
            break;
        case 4027:
                // HRSS_UPDATE_FILE_TRANSFER_SUCCESS
            break;
        default:
            break;
        }
        break;
    default:
        break;
    }
}
```

3.2. Register Command Class

Reference function names and corresponding identities are shown in the below table:

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
set_timer	Set robot's timer	O	O	×
get_timer	Get robot's timer	O	O	O
set_timer_start	Start robot's timer	O	O	×
set_timer_stop	Stop robot's timer	O	O	×
get_timer_status	Get robot's timer status	O	O	O
set_timer_name	Set robot's timer name	O	O	×
get_timer_name	Get robot's timer name	O	O	O
set_counter	Set robot's counter	O	O	O
get_counter	Get robot's counter	O	O	O
get_counter_name	Get robot's counter name	O	O	O
set_counter_name	Set robot's counter name	O	O	×
set_pr_type	Set position register coordinates type	O	O	×
get_pr_type	Get position register coordinates type	O	O	O
set_pr_coordinate	Set position register coordinates value	O	O	×
get_pr_coordinate	Get position register coordinates value	O	O	O
set_pr_tool_base	Set position register tool base number	O	O	×
get_pr_tool_base	Get position register tool base number	O	O	O
set_pr	Set position register value	O	O	×
get_pr	Get position register value	O	O	O

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
remove_pr	Clear position register value	O	O	×
set_pr_comment	Set position register comment	O	O	×
get_pr_comment	Get position register comment	O	O	O

The following section will describe the various function names, instruction formats, parameter settings and example descriptions.

3.2.1. Set robot's timer - set_timer

int set_timer(HROBOT robot, int index, int value)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Timer number (1-20)
value	int	value range(-2147483648~2147483647)
Return value	int	Success: 0 Fail: error code

C++:

```
set_timer(robot,1,100) //set Timer number 1 with value 100
```

3.2.2. Get robot's timer- get_timer

int get_timer(HROBOT robot, int index)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Counter number (1-20)
Return value	int	Success: value of Timer Fail: error code

C++:

```
set_timer(robot,1,100) //set Timer number 1 with value 100
get_timer(robot,1)   //get value from Timer number 1
```

3.2.3. Start robot's timer- set_timer_start

int set_timer_start (HROBOT robot, int index)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Counter number (1-20)
Return value	int	Success: value of Timer Fail: error code

C++:

```
int timer_index;
set_timer_start(robot, timer_index);
```

3.2.4. Stop robot's timer- set_timer_stop

int set_timer_stop(HROBOT robot, int index)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Timer number (1-20)
Return value	int	Success: value of Timer Fail: error code

C++:

```
int timer_index;
set_timer_stop(robot, timer_index);
```

3.2.5. Get robot's timer status- get_timer_status

int get_timer_status(HROBOT robot, int index)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Timer number (1-20)
Return value	int	Close: 0 Open: 1 Fail: error code

C++:

```
int timer_index = 1;
set_timer_status(robot, timer_index);
```

3.2.6. Set robot's timer name- set_timer_name

int set_timer_name(HROBOT robot, int index, wchar_t* name)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Timer number (1-20)
name	wchar_t*	Name of counter
Return value	int	Success: 0 Fail: error code

C++:

```
int timer_index = 1;
wchar_t* name = L"This is Timer1.";
set_timer_name(robot, timer_index, name);
```

3.2.7. Get robot's timer name- get_timer_name

int get_timer_name(HROBOT robot, int index, wchar_t* name, int array_size)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Timer number (1-20)
name	wchar_t*	➢ Get name of counter ➢ The first value is the comment length
array_size	int	Set it up to get the comment length
Return value	int	Success: 0 Fail: error code

C++:

```
int timer_index = 1;
wchar_t* name = L"This is Timer1.";
wchar_t re_name[20] = "";
int array_size = 10;
set_timer_name(robot, timer_index, name);
get_timer_name(robot, timer_index, re_name, array_size);
```

3.2.8. Set robot's counter- set_counter

int set_counter(HROBOT robot, int index, int value)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Counter number (1-20)
value	int	Range of value(-2147483648~2147483647)
Return value	int	Success: 0 Fail: error code

C++:

```
set_counter(robot,1,100) //set Counter number 1 with value 100
```

3.2.9. Get robot's counter- get_counter

int get_counter(HROBOT robot, int index)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Counter number (1-20)
Return value	int	Success: 0 Fail: error code

3.2.10. Set robot's counter name- set_counter_name

int set_counter_name(HROBOT robot, int index, wchar_t* name)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Counter number (1-20)
name	wchar_t*	The length of the counter name is within 200 characters
Return value	int	Success: 0 Fail: error code

3.2.11. Get robot's counter name- get_counter_name

int get_counter_name(HROBOT robot, int index, wchar_t* name, int array_size)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Counter number (1-20)
name	wchar_t*	➢ Get name of counter ➢ The first value is the comment length
array_size	int	Set it up to get the comment length
Return value	int	Success: 0 Fail: error code

C++:

```
int index = 5;
wchar_t* name = L"This is Counter5";
wchar_t str[20] = "";
int array_size = 10;
set_counter_name(robot, index, name);
get_counter_name(robot, index, str, array_size);
```

3.2.12. Set position register coordinates type- set_pr_type

int set_pr_type(HROBOT robot ,int pr_num, int type)

Parameter	Data type	Description
robot	HROBOT	Device ID
pr_num	int	Position register number(1-4000)
type	int	Cartesian coordinate system:0 Joint coordinate system:1
Return value	int	Success: 0 Fail: error code

3.2.13. Get position register coordinates type- get_pr_type

int get_pr_type(HROBOT robot ,int pr_num)

Parameter	Data type	Description
Robot	HROBOT	Device ID
pr_num	int	Position register number(1-4000)
Return value	int	Cartesian coordinate system:0 Joint coordinate system:1 Fail: error code

3.2.14. Set position register coordinates- set_pr_coordinate

int set_pr_coordinate(HROBOT robot ,int pr_num, double* coor)

Parameter	Data type	Description
robot	HROBOT	Device ID
pr_num	int	Position register number (1-4000)
coor	double [6]	Desired setting coordinate array: Cartesian coordinate system {X,Y,Z,A,B,C} Joint coordinate system {A1,A2,A3,A4,A5,A6}
Return value	int	Success: 0 Fail: error code

3.2.15. Get position register coordinates- get_pr_coordinate

int get_pr_coordinate(HROBOT robot ,int pr_num, double* coor)

Parameter	Data type	Description
robot	HROBOT	Device ID
pr_num	int	Position register number(1-4000)
coor	double [6]	Return coordinate array: Cartesian coordinate system{X,Y,Z,A,B,C} Joint coordinate system{A1,A2,A3,A4,A5,A6}
Return value	int	Success: 0 Fail: error code

3.2.16. Set position register tool base coordinates- set_pr_tool_base

int set_pr_tool_base(HROBOT robot ,int pr_num, int tool, int base)

Parameter	Data type	Description
robot	HROBOT	Device ID
pr_num	int	Position register number (1-4000)
tool	int	Tool number
base	int	Base number
Return value	int	Success: 0 Fail: error code

C++:

```
(1)
double coor [6]={0,0,0,0,-90,0};
set_pr(robot,1,1,coor,2,2); //set address register
                            //address register number:1
                            //coordinate type:joint
                            //coordinate:{0,0,0,0,-90,0}
                            //tool number:2
                            //base number:2
(2)
set_pr_type(robot,1,1);
set_pr_coordinate(robot,1,coor);
set_pr_tool_base(robot,1,2,2);

(3)
int prType=get_pr_type(robot,1,1);      //get pr type
double coor[6]
get_pr_coordinate(robot,1,coor);  //get coordinate from pr 1
int tool_base[2];
get_pr_tool_base(robot,1,tool_base);  //get tool and base from pr 1
                                      //tool:tool_base[0]
                                      //base: tool_base [1]
```

STEPS:

(1). Set pr information

(2). Same effect as (1)

(3). Obtain the information of position register

A. Type of coordinate

B. Joint coordinate system
C. Tool number
D. Base number

3.2.17. Get position register tool base coordinates- get_pr_tool_base

int get_pr_tool_base(HROBOT robot ,int pr_num, int* tool_base)

Parameter	Data type	Description
robot	HROBOT	Device ID
pr_num	Int	Position register number (1-4000)
tool_base	int[2]	int[0]:Tool number int[1]:Base number
Return value	int	Success: 0 Fail: error code

3.2.18. Set position register data- set_pr

int set_pr(HROBOT robot ,int pr_num, int coor_type, double *coor, double *ext_pos, int tool, int base)

Parameter	Data type	Description
robot	HROBOT	Device ID
pr_num	int	Position register number(1-4000)
coor_type	int	Type of coordinate system: Cartesian coordinate system:0 Joint coordinate system:1
coor	Double [6]	Desired position register coordinate Value1~Value6
ext_pos	Double [3]	Desired position register coordinate Value7~Value9
tool	int	Tool coordinate system number
base	int	Base coordinate system number
Return value	int	Success: 0 Fail: error code

3.2.19. Get position register value- get_pr

int get_pr(HROBOT robot ,int pr_num, int* coor_type, double* coor, double *ext_pos, int* tool, int* base)

Parameter	Data type	Description
robot	HROBOT	Device ID
pr_num	int	Position register number (1-4000)
coor_type	int	Type of coordinate system: Cartesian coordinate system: 0 Joint coordinate system: 1
coor	Double [6]	Position register coordinates to set Value1~Value6
ext_pos	Double [3]	Position register coordinates to set Value7~Value9
tool	int	Tool coordinate system number
base	int	Base coordinate system number
Return value	int	Success: 0 Fail: error code

3.2.20. Clear position register value- remove_pr

int remove_pr(HROBOT robot ,int pr_num)

Parameter	Data type	Description
robot	HROBOT	Device ID
pr_num	int	Position register number (1-4000)
Return value	int	Success: 0 Fail: error code
Remark	Clears comment and value simultaneously	

3.2.21. Get position register comment- get_pr_comment

int get_pr_comment (HROBOT robot ,int pr_num , wchar_t* comment, int arr_size)

Parameter	Data type	Description
robot	HROBOT	Device ID
pr_num	int	Position register number (1-4000)
comment	wchar_t*	➢ Get the comment text of the position register ➢ The first value is the comment length
arr_size	int	Set it up to get the comment length
Return value	int	Success: 0 Fail: error code

3.2.22. Set position register comment- set_pr_comment

int set_pr_comment (HROBOT robot ,int pr_num ,wchar_t* comment)

Parameter	Data type	Description
robot	HROBOT	Device ID
pr_num	int	Position register number (1-4000)
comment	wchar_t*	Command of position register
Return value	int	Success: 0 Fail: error code

C++:

```
(1)
double coor[6]={0,0,0,0,-90,0};
double ext_pos[3]={0,0,0}
set_pr(robot,2,1,coor,ext_pos,2,2); //set address register
                                    //address register number:2
                                    //coordinate type:joint
                                    //coordinate:{0,0,0,0,-90,0}
                                    //ext_pos:{0,0,0}
                                    //tool number:2
                                    //base number:2
int coor_type=-1;
coor[6]={0}
int tool = -1;
int base= -1;
int array_size = 10;
```

```
wchar_t* comment = L“This is a comment.”;
wchar_t str[20] = “”;
get_pr(robot,2 , &coor_type,coor,ext_pos,&tool,&base);
remove_pr(robot,2);
get_pr(robot,2 , &coor_type,coor,ext_pos,&tool,&base);
set_pr_comment (robot, 2, comment);
get_pr_comment (robot, 2, str, comment, array_size);
```

3.3. System Variable Command Class

Reference function names and corresponding identities are shown in the below table:

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
set_acc_dec_ratio	Set acceleration ratio	×	O	×
get_acc_dec_ratio	Get acceleration ratio	O	O	O
set_acc_time	Set acceleration time	×	O	×
get_acc_time	Get acceleration time	O	O	O
set_ptp_speed	Set PTP movement speed	×	O	×
get_ptp_speed	Get PTP movement speed	O	O	O
set_lin_speed	Set linear movement speed	×	O	×
get_lin_speed	Get linear movement speed	O	O	O
set_override_ratio	Set override ratio	O	O	×
get_override_ratio	Get override ratio	O	O	O
set_robot_id	Set the robot identification name	O	O	×
get_robot_id	Get the robot identification name	O	O	O
set_smooth_length	Set the motion smoothing radius	×	O	×
get_alarm_code	Get alarm code	O	O	O
set_digital_setting	Set digital signal setting	O	O	×
get_digital_setting	Get digital signal setting	O	O	O
set_language	Set language	O	O	O
get_controller_time	Get controller current time	O	O	O
set_user_alarm_setting_message	Set user's custom alarm message	O	O	×
get_user_alarm_setting_message	Get user's custom alarm message	O	O	O

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
get_ext_axis_setting	Get external axis normal setting	O	O	O
set_ext_axis_setting	Set external axis normal setting	O	O	×
get_ext_axis_setting_advanced	Get external axis advanced setting	O	O	O
set_ext_axis_setting_advanced	Set external axis advanced setting	O	O	×
ext_mastering	Zero-point calibration for the external axis	O	O	×
get_current_ext_pos	Get external axis coordinates	O	O	O
get_current_ext_mode	Get external axis synchronous and asynchronous mode	O	O	O

The following section will describe the various function names, instruction formats, parameter settings and example descriptions.

3.3.1. Set acceleration ratio- set_acc_dec_ratio

int set_acc_dec_ratio(HROBOT robot, int value)

Parameter	Data type	Description
robot	HROBOT	Device ID
value	int	Acceleration ratio 1-100(%)
Return value	int	Success: 0 Fail: error code

3.3.2. Get acceleration ratio- get_acc_dec_ratio

int get_acc_dec_ratio(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: acceleration ratio 1-100(%) Fail: error code

```
set_operation_mode (robot, 1);
set_acc_dec_ratio(robot,20);
acc=get_acc_dec_ratio (robot);
```

➢ The acceleration/deceleration ratio can only be set in auto mode

3.3.3. Set acceleration time- set_acc_time

int set_acc_time(HROBOT robot, double value)

Parameter	Data type	Description
robot	HROBOT	Device ID
Value	double	Acceleration time
Return value	int	Success: 0 Fail: error code

Warning

1. Using the SET_ACC instruction can allow the articulated robot to have a higher degree of operation. Still, a setting value that is too low may result in exceeding the articulated robot's load and causing error alarms to occur.
2. Please adjust the parameter appropriately based on the actual usage conditions to avoid causing damages to the equipment due to excessively high operating speeds.
3. Use of set_acc_dec_ratio is a recommended command.

3.3.4. Get acceleration time- get_acc_time

double get_acc_time(HROBOT robot)

Parameter	Data type	Description
Robot	HROBOT	Device ID
Return value	double	Success: acceleration time Fail: error code

3.3.5. Set point to point motion speed- set_ptp_speed

int set_ptp_speed(HROBOT robot, int value)

Parameter	Data type	Description
robot	HROBOT	Device ID
value	value	PTP speed ratio1-100(%)
Return value	int	Success: 0 Fail: error code

3.3.6. Get point to point motion speed- get_ptp_speed

int get_ptp_speed(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: speed ratio 1-100(%) Fail: error code

```
set_ptp_speed (robot,50);
vel = get_ptp_speed (robot);
```

3.3.7. Set straight line motion speed- set_lin_speed

int set_lin_speed(HROBOT robot, double value)

Parameter	Data type	Description
robot	HROBOT	Device ID
value	value	Speed of linear motion(mm/s) Upper limit depends on the robot model
Return value	int	Success: 0 Fail: error code

3.3.8. Get straight line motion speed- get_lin_speed

double get_lin_speed(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	double	Success: Speed of linear motion(mm/s) Fail: error code

```
set_lin_speed(robot);
vel =get_lin_speed(robot);
```

3.3.9. Set override ratio- set_override_ratio

int set_override_ratio(HROBOT robot, double value)

Parameter	Data type	Description
robot	HROBOT	Device ID
value	value	Override speed ratio 1-100(%)
Return value	int	Success: 0 Fail: error code

3.3.10. Get override ratio- get_override_ratio

int get_override_ratio(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: Override speed ratio 1-100(%) Fail: error code

```
set_ override_ratio(robot,80);
override=get_ override_ratio(robot);
```

3.3.11. Set robot ID- set_robot_id

int set_robot_id (HROBOT robot, char* robot_id)

Parameter	Data type	Description
robot	HROBOT	Device ID
robot_id	char*	Robot number
Return value	int	Success: robot number Fail: error code

```
char* robot_id = “robot1”
set_robot_id (robot, robot_id);
```

3.3.12. Get robot ID- set_robot_id

int set_robot_id (HROBOT robot, char* robot_id)

Parameter	Data type	Description
robot	HROBOT	Device ID
robot_id	char*	Robot number
Return value	int	Success: robot number Fail: error code

```
char* v = new char[256];
get_robot_id(robot, v);
delete[] v;
```

3.3.13. Set motion smooth radius- set_smooth_length

int set_smooth_length (HROBOT robot, int radius)

Parameter	Data type	Description
robot	HROBOT	Device ID
radius	int	Motion smooth radius must be greater than 100
Return value	int	Success: 0 Fail: error code

C++:

```
double radius = 200.0;
set_smooth_length (robot, radius);
```

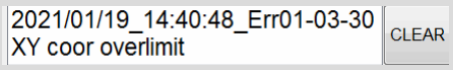
3.3.14. Get error code- get_alarm_code

int get_alarm_code(HROBOT robot, int& count uint64_t* alarm_code)

Parameter	Data type	Description
robot	HROBOT	Device ID
count	int&	Return alarm count
alarm_code	unsigned uint64_t[20]	Alarm code array ➢ A maximum of 20 alarm entries will be saved ➢ Calling clear_alarm will clear this array ➢ Please refer to the HRSS

		software manual for alarm code correspondence ➢ Convert to hexadecimal display to correspond to the software manual
Return value	int	Success: 0 Fail: error code

```
uint64_t alarm _code[20]={0};
int* count=0;
get_ alarm _code(robot,count,alarm_code);
```




```
// alarm_code: 0001033000000000
Disassembled into 0001 0330 0000 0000
0001: The last two digits correspond to the first classification of HRSS
0330: Corresponds to the second (03) and third (30) classifications
0000: Reserved
0000: Reserved
```

3.3.15. Set digital setting- set_digital_setting

int set_digital_setting (HROBOT robot, int* index char* text)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int[18]	Array with a length of 18 ➢ index [0]: Clear Error DI: 0 SI: 1 ➢ index[1]: I/O index value Disable: 0 DI: 1~48 SI: 1~256 ➢ index [2]: External Alarm DI: 0 SI: 1 ➢ index[3]: I/O index value Disable: 0

Parameter	Data type	Description
		DI: 1~48 SI: 1~256 ➢ index [4]: System Shutdown DI: 0 SI: 1 ➢ index[5]: I/O index value Disable: 0 DI: 1~48 SI: 1~256 ➢ index [6]: Motor Warning DO: 0 SO: 1 ➢ index[7]: I/O index value Disable: 0 DO: 1~48 SO: 1~256 ➢ index [8]: System StartUp DO: 0 SO: 1 ➢ index[9]: I/O index value Disable: 0 DO: 1~48 SO: 1~256 ➢ index [10]: Mode Output DO: 0 SO: 1 ➢ index[11]: I/O index value Disable: 0 DO: 1~48 SO: 1~256 ➢ index [12]: Reset Driver DO: 0 SO: 1 ➢ index[13]: I/O index value Disable: 0 DO: 1~48 ➢ index [14]: Emergency Output

Parameter	Data type	Description
		DO: 0 ➢ index[15]: I/O index value Disable: 0 DO: 1~48 ➢ index [16]: Reset Safety Relay DO: 0 ➢ index[17]: I/O index value Disable: 0 DO: 1~48
text	char	Sets the text to display when the DI/SI pin triggers the external alarm.
Return value	int	Success: 0 Fail: error code

3.3.16. Get digital setting- get_digital_setting

int get_digital_setting (HROBOT robot, int* index char* text)

Parameter	Data type	Description
robot	HROBOT	Device ID
count	int[19]	* The DIO/SIO is set to trigger the clear error and other instructions. ➢ index[0,2,4,6,8,10,12,1,16] 0: DIO 1:SIO ➢ index[1,3,5,7,9,11,13,15,17] 0: Disable this function DIO range 1~48 SIO range 1~256 ➢ index[18] text length
text	char	Get the text to display when the DI/SI pin triggers the external alarm.
Return value	int	Success: 0 Fail: error code

```
int DIO = 0;
int SIO = 1;
int index[18] = { DIO, 36, DIO, 37, SIO, 38, SIO, 39, DIO, 40, DIO, 41,
                  DIO, 42, DIO, 43, DIO, 44 };
int recv[19] = {0};
char* text = “External Alarm”;
set_digital_setting (robot, index, text);
get_digital_setting (robot, recv, text);
```

3.3.17. Set language- set_language

int set_language (HROBOT robot, int language)

Parameter	Data type	Description
robot	HROBOT	Device ID
language	int	0: English 1: Traditional Chinese 2: Simplified Chinese 3: Japanese 4: Korean
Return value	int	Success: 0 Fail: error code

```
int language = 0;
set_language (robot, language);
```

3.3.18. Get controller's current time- get_controller_time

int get_controller_time (HROBOT robot , int& year, int& month, int& day, int& hour, int& minute, int& second)

Parameter	Data type	Description
robot	HROBOT	Device ID
year	int&	Year
month	int&	Month
day	int&	Date
hour	int&	Hour
minute	int&	Minute
second	int&	Second
Return value	int	Success: 0 Fail: error code

```
int* year = 0;
int* month = 0;
int* day = 0
int* hour = 0;
int* minute = 0
int* second = 0;
get_controller_time (robot, year, month, day, hour, minute, second);
```

3.3.19. Set user's custom alarm message- set_user_alarm_setting_message

int set_user_alarm_setting_message(HROBOT robot , int num, char* message)

Parameter	Data type	Description
robot	HROBOT	Device ID
num	int	[0-9] number
message	char*	Set the message to display
Return value	int	Success: 0 Fail: error code

3.1.1.Get user's custom alarm message- get_user_alarm_setting_message

int get_user_alarm_setting_message(HROBOT robot , int num, char* message)

Parameter	Data type	Description
robot	HROBOT	Device ID
num	int	[0-9] number
message	char*	Display message
Return value	int	Success: 0 Fail: error code

```
int num= 0;
char* msg = "user alarm text";
get_user_alarm_setting_message(robot, num, msg);
get_user_alarm_setting_message(robot, num, msg);
```

3.3.20. Get external axis' general setting- get_ext_axis_setting

int get_ext_axis_setting(HROBOT robot , int index, bool& enable, int& mode, double& high_limit, double& low_limit)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	[0-2] index value
enable	bool&	0: Off 1: On
mode	int&	0: Sync 1: Async
high_limit	double&	Upper limit
low_limit	double&	Lower limit
Return value	int	Success: 0 Fail: error code

3.3.21. Set external axis' general setting- set_ext_axis_setting

int set_ext_axis_setting (HROBOT robot , int index, bool enable, int mode, double high_limit, double low_limit)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	[0-2] index value
enable	bool	0: Off 1: On
mode	int	0: Sync 1: Async
high_limit	double	Upper limit
low_limit	double	Lower limit
Return value	int	Success: 0 Parameter range error: -1 Synchronization/non-synchronization mode error: -2 Set synchronized rotation parameter failed: -3 Set synchronized linear parameter failed: -4 Fail: error code
Remark	➢ This API, set_ext_axis_setting_advanced, must be used for the setting to be saved. ➢ set_ext_axis_setting_advanced must be set before use. Otherwise, HRSS will initialize the default values and resulting in the setting of parameters failure.	

3.3.22. Get external axis' advanced setting- get_ext_axis_setting_advanced

int get_ext_axis_setting_advanced (HROBOT robot , int index, int& type, bool& math, bool& continuous, int* int_value, double* double_value)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	[0-2] index value
type	int&	0: Linear 1: Rotary
math	bool&	0: Off 1: On
continuous	bool&	0: Off 1: On
int_value	int[3]	This array includes three spaces ➢ int_value[0]: Motor direction 0: Positive 1: Negative ➢ int_value[1]: Maximum rotating speed of motor [rpm] ➢ int_value[2]: Motor resolution [pls/rev]
double_value	double[3]	This array includes three spaces ➢ double_value[0]: Acceleration time [ms] Range: [20-1000] ➢ double_value[1]: Reduction ratio of the reducer installed on the motor [mot-rev/axs-rev] ➢ double_value[2]: Pitch [mm/axs-rev]
Return value	int	Success: 0 Fail: error code

3.3.23. Set external axis' advanced setting- set_ext_axis_setting_advanced

int set_ext_axis_setting_advanced (HROBOT robot , int index, int type, bool math, bool continuous, int* int_value, double* double_value)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	[0-2] index value
type	int	0: Linear 1: Rotary
math	bool	0: Off 1: On
continuous	bool	0: Off 1: On
int_value	int[3]	This array includes three spaces ➢ int_value[0]: Motor direction 0: Positive 1: Negative ➢ int_value[1]: Maximum rotating speed of motor [rpm] ➢ int_value[2]: Motor resolution [pls/rev]
double_value	double[3]	This array includes three spaces ➢ double_value[0]: Acceleration time [ms] Range: [20-1000] ➢ double_value[1]: Reduction ratio of the reducer installed on the motor [mot-rev/axs-rev] ➢ double_value[2]: Pitch [mm/axs-rev]
Return value	int	Success: 0 Fail: error code
Remark	set_ext_axis_setting must be used in order to save the above settings.	

3.3.24. Zero point calibration for the external axis- ext_mastering

int ext_mastering (HROBOT robot , int index)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	[0-2] index value
Return value	int	Success: 0 Encoder is currently calibrating: 2 External axis not enabled: 3 Offline not supported: 100 Fail: error code

```
int index= 0;
bool enable = true;
int mode = 0;
double high_limit = 50;
double low_limit = -10;
int type = 0;
bool math = false;
bool continuous = false;
int value[3] = {5000, 500, 500};
double value[3] = {100, 100, 100};
set_ext_axis_setting_advanced (robot , int index, type, math, continuous, int_value,  double_value)

set_ext_axis_setting (robot , index, enable, mode, high_limit, low_limit);

ext_mastering(robot, index);
```

3.3.25. Get external axis' current position- get_current_ext_pos

int get_current_ext_pos (HROBOT robot , int index)

Parameter	Data type	Description
robot	HROBOT	Device ID
pos	double[3]	This array includes three spaces pos[0]: Point information of external axis 1 pos[1]: Point information of external axis 2 pos[2]: Point information of external axis 3
Return value	int	Success: 0 Fail: error code

3.3.26. Get external axis synchronous and asynchronous mode- get_current_ext_mode

int get_current_ext_mode (HROBOT robot , char* mode)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	char*	SYNC: Synchronized mode ASYNC: Asynchronized mode NULL: External axis not enabled Example: NULL_NULL_NULL Means that all three external axes are not enabled
Return value	int	Success: 0 Fail: error code

3.4. Input and Output Command Class

Reference function names and corresponding identities are shown in the below table:

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
get_digital_input	Get digital input	O	O	O
get_digital_output	Get digital output	O	O	O
set_digital_output	Set digital output	O	O	×
set_DI_simulation_Enable	Set simulation of digital input value	O	O	×
set_DI_simulation	Set simulation of digital input State	O	O	×
get_DI_simulation_Enable	Get simulation of digital input value	O	O	O
set_digital_input_comment	Set digital input comment	O	O	×
get_digital_input_comment	Get digital input comment	O	O	O
set_digital_output_comment	Set digital output comment	O	O	×
get_digital_output_comment	Get digital output comment	O	O	O
get_robot_input	Get robot input	O	O	O
get_robot_output	Get robot output	O	O	O
set_robot_output	Set robot output	O	O	×
get_valve_output	Set valve output	O	O	O
set_valve_output	Get valve output	O	O	×
get_function_input	Get function input	O	O	O
get_function_output	Get function output	O	O	O
get_module_input_config	Get module input configuration	O	O	O
get_module_output_config	Get module output configuration	O	O	O

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
set_module_input_simulation	Get module input simulation value	O	O	×
set_module_input_value	Get module input value	O	O	×
set_module_input_start	Get module input start number	O	O	×
set_module_input_end	Get module input end number	O	O	×
set_module_input_comment	Set module input command	O	O	×
set_module_output_value	Set module output value	O	O	×
set_module_output_start	Get module output start number	O	O	×
set_module_output_end	Get module output end number	O	O	×
set_module_output_comment	Set module output command	O	O	×
set_module_input_type	Set module input type	O	O	×
set_module_output_type	Set module output type	O	O	×
save_module_io_setting	Save module I/O setting	O	O	×
SyncOutput	Motion digital output open	O	O	×
get_DI_range	Get DI range value	O	O	O
get_DI_sim_range	Get DI value of simulation range	O	O	O
get_DI_comment_range	Get DI command of range	O	O	O
get_DO_range	Get DO commands of range	O	O	O
get_DO_comment_range	Get DO commands of range	O	O	O
get_FI_all	Get FI all value	O	O	O
get_FO_all	Get FO all value	O	O	O
get_timer_status_all	Get timer all status	O	O	O

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
get_timer_value_all	Get timer all values	O	O	O
get_timer_comment_range	Get timer commands's range	O	O	O
get_counter_value_all	Get all counter’s values	O	O	O
get_counter_comment_range	Get range of counter commands	O	O	O
get_fieldbus_rs_srw_range	Get Fieldbus Register SRW values of range	O	O	O
get_fieldbus_rs_srr_range	Get Fieldbus Register SRR values of range	O	O	O
get_fieldbus_rs_parameter_range	Get Fieldbus Register parameters range	O	O	O
get_fieldbus_rs_comment_range	Set Fieldbus Register parameters range	O	O	O
get_system_input_all	Get all system input values	O	O	O
get_system_output_all	Get all system output values	O	O	O
get_MI_config_all	Get all module input settings	O	O	O
get_MO_config_all	Get all module output settings	O	O	O
get_MI_comment_range	Get range of module input comment	O	O	O
get_MO_comment_range	Get range of module output comment	O	O	O
get_SI_range	Get SI range value	O	O	O
get_SI_sim_range	Get SI ranges simulation value	O	O	O
get_SO_range	Get SO range value	O	O	O
get_SI_comment_range	Get SI range comment	O	O	O
get_SO_comment_range	Get SO range comment	O	O	O
get_PR_comment_array	Get PR range comment	O	O	O
get_RI_all	Get all RI values	O	O	O
get_RO_all	Get all RO values	O	O	O
get_VO_all	Get all VO values	O	O	O
set_DI_array	Set multiple DI values	O	O	×

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
set_DI_sim_array	Set multiple DI simulation values	O	O	×
set_DO_array	Set multiple DO values	O	O	×
set_timer_value_array	Set multiple timer values	O	O	×
set_counter_array	Set multiple counter values	O	O	×
set_fieldbus_srw_array	Set multiple fieldbus SRW values	O	O	×
set_SI_array	Set multiple SI values	O	O	×
set_SI_sim_array	Set multiple SI simulation values	O	O	×
set_SO_array	Set multiple SO values	O	O	×
set_MO_array	Set multiple MO values	O	O	×
set_RO_array	Set multiple RO values	O	O	×
set_VO_array	Set multiple VO values	O	O	×

The following section will describe the various function names, instruction formats, parameter settings and example descriptions.

3.4.1. Get input status- get_digital_input

int get_digital_input (HROBOT robot ,int index)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index input value [1-48]
Return value	int	0:OFF 1:ON

```
int state;
state=get_digital_input (robot,1);
```

3.4.2. Set input simulation value- set_DI_simulation_Enable

int set_DI_simulation_Enable (HROBOT robot, int index, bool value)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index input value [1-48]
value	bool	true: Enable false: Disable
Return value	int	Successful: 0 Fail: error code

3.4.3. Set input status- set_DI_simulation

int set_DI_simulation (HROBOT robot, int index, bool value)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index input value [1-48]
value	bool	true: Enable false: Disable
Return value	int	Successful: 0 Fail: error code
Remark	simulation Input must first be enabled	

3.4.4. Get input simulation value- get_DI_simulation_Enable

int get_DI_simulation_Enable (HROBOT robot, int index)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index input value [1-48]
Return value	int	0: OFF 1: ON

```
int index = 1;
bool ON = true;
int recv = -1;
set_DI_simulation_Enable(robot, index, ON);
set_DI_simulation(robot, index, ON);
recv = get_DI_simulation_Enable(robot, index);
```

3.4.5. Get output status- get_digital_output

int get_digital_output (HROBOT robot ,Int Index)

Parameter	Data type	Description
robot	HROBOT	Device ID
Index	int	Index output value [1-48]
Return value	int	0: OFF 1: ON

```
int state;
state=get_digital_output (robot,1);
```

3.4.6. Set output status- set_digital_output

int set_digital_output (HROBOT robot ,Int Index,bool value)

Parameter	Data type	Description
robot	HROBOT	Device ID
Index	int	Index output value [1-48]
value	bool	true or false
Return value	int	Successful: 0 Fail: error code

```
int index;
set_digital_output (robot, index, true);
```

3.4.7. Set input comment- set_digital_input_comment

int set_digital_input_comment (HROBOT robot, int index, wchar_t* comment)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index input value [1-48]
comment	wchar_t*	Set comment text
Return value	int	Successful: 0 Fail: error code

3.4.8. Set output comment- set_digital_output_comment

int set_digital_output_comment (HROBOT robot, int index, wchar_t* comment)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index output value [1-48]
comment	wchar_t*	Set comment text
Return value	int	Successful: 0 Fail: error code

3.4.9. Get input comment- get_digital_input_comment

int get_digital_input_comment (HROBOT robot, int index, wchar_t* comment, int arr_size)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index value of input [1-48]
comment	wchar_t*	➢ Get comment text ➢ The first value is the comment length
arr_size	int	Set it up to get the comment length
Return value	int	Success: 0 Fail: error code

3.4.10. Get output comment- get_digital_output_comment

int get_digital_output_comment (HROBOT robot, int index, wchar_t* comment, int arr_size)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index output value [1-48]
comment	wchar_t*	➢ Get comment text ➢ The first value is the comment length
array_size	int	set it up to get the comment length
Return value	int	Success: 0 Fail: error code

```
int index = 1;
wchar_t* comment = L“Test comment”;
wchar_t str[20] = “”;
int array_size = 10;
set_digital_input_comment (robot, index, comment);
get_digital_input_comment(robot, index, str, array_size);
set_digital_output_comment (robot, index, comment);
get_digital_output_comment(robot, index, str, array_size);
```

3.4.11. Get robot input- get_robot_input

int get_robot_input (HROBOT robot ,Int Index)

Parameter	Data type	Description
robot	HROBOT	Device ID

Index	int	Index input value [1-48]
Return value	int	0: OFF 1: ON

```
int state;
state=get_robot_input (robot,1);
```

3.4.12. Get robot output- get_robot_output

int get_robot_output (HROBOT robot ,Int Index)

Parameter	Data type	Description
robot	HROBOT	Device ID
Index	int	Index value of output [1-8]
Return value	int	0: OFF 1: ON

```
int state;
state=get_robot_output (robot,1);
```

3.4.13. Set robot output- set_robot_output

int set_robot_output (HROBOT robot ,Int Index,bool value)

Parameter	Data type	Description
robot	HROBOT	Device ID
Index	int	Index output value [1-8]
value	bool	true or false
Return value	int	Successful: 0 Fail: error code

```
int state;
set_robot_output (robot,1,true);
```

3.4.14. Get electromagnetic valve output- get_valve_output

int get_valve_output (HROBOT robot ,Int Index)

Parameter	Data type	Description
robot	HROBOT	Device ID

Index	int	Index output value [1-3]
Return value	int	0: OFF 1: ON

```
int state;
state=get_valve_output (robot,1);
```

3.4.15. Set electromagnetic valve output- set_valve_output

int set_valve_output (HROBOT robot ,Int Index,bool value)

Parameter	Data type	Description
robot	HROBOT	Device ID
Index	int	Index output value [1-3]
value	bool	true or false
Return value	int	Successful: 0 Fail: error code

```
int state;
set_valve_output (robot,1,true);
```

3.4.16. Get function input status- get_function_input

int get_function_input (HROBOT robot ,Int Index)

Parameter	Data type	Description
robot	HROBOT	Device ID
Index	int	Index value of function input [1-8] 1: Start 2: Hold 3: Stop 4: Enable 5: RSR1 6: RSR2 7: RSR3 8: RSR4
Return value	int	0: OFF 1: ON

```
int state;
state=get_function_input (robot,1);
```

3.4.17. Get function output status- get_function_output

int get_function_output (HROBOT robot ,Int Index)

Parameter	Data type	Description
robot	HROBOT	Device ID
Index	int	Index value of function output [1-8] 1: Run 2: Held 3: Fault 4: Ready 5: ACK1 6: ACK2 7: ACK3 8: ACK4
Return value	int	0:OFF 1:ON

```
int state;
state=get_function_output (robot,1);
```

3.4.18. Get module input configuration- get_module_input_config

int get_module_input_config (HROBOT s, int index, bool& sim, bool& value, int& type, int& start, int& end, wchar_t* comment, int arr_size)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index value of module input number [1-32]
sim	bool	Simulation value 0:OFF 1:ON
value	bool	0:OFF 1:ON
type	int	0:DI 1:SI
start	int	0: Diseable [1-48]: Start index value of digital input number
end	int	0: Diseable [1-48]: End index value of digital input number (should not be smaller than starting number)
comment	wchar_t*	➢ Get comment text ➢ The first value is the comment length
arr_size	int	Set it up to get the comment length
Return value	int	Successful: 0 Fail: error code

3.4.19. Get module output configuration- arsine

int get_module_output_config (HROBOT s, int index, bool& sim, bool& value, int& type, int& start, int& end, wchar_t* comment, int arr_size)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index value of module output number [1-32]
sim	bool	Simulation value 0: OFF 1: ON
value	bool	0: OFF 1: ON
type	int	0: DO 1: SO
start	int	0: Diseable [1-48]: Start index value of digital output number
end	int	0: Diseable [1-48]: End index value of digital output number
comment	wchar_t*	➢ Get comment text ➢ The first value is a comment length
arr_size	int	Set it up to get the comment length
Return value	int	Successful: 0 Fail: error code

3.4.20. Set module input simulation value- set_module_input_config

int set_module_input_config (HROBOT s, int index, bool enable)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index value of module input number [1-32]
enable	bool	Simulation value 0: OFF 1: ON
Return value	int	Successful: 0 Fail: error code

3.4.21. Set module input- set_module_input_value

int set_module_input_value (HROBOT s, int index, bool enable)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index value of module input number [1-32]
enable	bool	0: OFF 1: ON ※Can only be set as ON when the sim is ON.
Return value	int	Successful: 0 Fail: error code

3.4.22. Set module input start number- set_module_input_start

int set_module_input_start(HROBOT s, int index, int start_number)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index value of module input number [1-32]
start_number	int	0: Diseable [1-48]: Start index value of digital input number
Return value	int	Successful: 0 Fail: error code

3.4.23. Set module input end number- set_module_input_end

int set_module_input_end(HROBOT s, int index, int end_number)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index value of module input number [1-32]
end_number	int	0: Diseable [1-48]: End index value of digital input number (must not be smaller than starting value)
Return value	int	Successful: 0 Fail: error code

3.4.24. Set module input comment- set_module_input_comment

int set_module_input_comment(HROBOT s, int index, wchar_t* comment)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index value of module input number [1-32]
comment	wchar_t*	Set comment for specified module input
Return value	int	Successful: 0 Fail: error code

3.4.25. Set module output- set_module_output_value

int set_module_output_value(HROBOT s, int index, bool enable)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index value of module output number [1-32]
enable	bool	0: OFF 1: ON
Return value	int	Successful: 0 Fail: error code

3.4.26. Set module output start number- set_module_output_start

int set_module_output_start(HROBOT s, int index, int start_number)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index value of module output number [1-32]
start_number	int	0: Diseable [1-48]: Start index value of digital output number
Return value	int	Successful: 0 Fail: error code

3.4.27. Set module output end number- set_module_output_end

int set_module_output_end(HROBOT s, int index, int end_number)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index value of module output number [1-32]
end_number	int	0: Diseable [1-48]: End index value of digital output number (must not be smaller than start)
Return value	int	Successful: 0 Fail: error code

3.4.28. Set module output comment- set_module_output_comment

int set_module_output_comment(HROBOT s, int index, wchar_t* comment)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index value of module output number [1-32]
comment	wchar_t*	Set comment text
Return value	int	Successful: 0 Fail: error code

3.4.29. Set module input type- set_module_input_type

int set_module_input_type(HROBOT s, int index, int type)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index value of module output number [1-32]
type	int	0: DI 1: SI
Return value	int	Successful: 0 Fail: error code

3.4.30. Set module output type- set_module_output_type

int set_module_output_type(HROBOT s, int index, int type)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Index value of module output number [1-32]
type	int	0: DO 1: SO
Return value	int	Successful: 0 Fail: error code

3.4.31. Save module setting- save_module_io_setting

int save_module_io_setting (HROBOT s)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Successful: 0 Fail: error code

```
bool sim,value;
int start,end, type;
wchar_t comment[100];
wchar_t* set_comment = L”module comment”;
int array_size = 10;
get_module_input_config (robot,0,sim,value,type,start,end,comment,
array_size);
get_module_output_config(robot,0,sim,value,type,start,end,comment,
array_size);
set_module_input_simulation(robot,0,true);
set_module_input_value(robot, 0,true);
set_module_input_start(robot, 0,2);
set_module_input_end(robot, 0,5);
set_module_input_type(robot, 0, type)
set_module_output_comment(robot, 0, set_comment);
set_module_output_value(robot,0, true);
set_module_output_start(robot, 0,10);
set_module_output_end(robot, 0, 10);
set_module_output_comment(robot, 0, set_comment);
```

```
set_module_output_type(robot, 0, type)
save_module_io_setting(robot);
```

3.4.32. Perform DO switch operation during motion- SyncOutput

int SyncOutput (HROBOT robot, int O_type, int O_id, int on_off, int synMode, int delay, int distance)

Parameter	Data type	Description
robot	HROBOT	Device ID
O_Type	int	0: DO digital output 1: RO robot output
O_id	int	Index value of [1-48] digital output Index value of [1-8] robot output
on_off	int	0: Output off 1: Output on
synMode	int	0: Start;The start motion position determines the delay time; if the delay is positive, change the output after N milliseconds; if it is negative, start motion will change the output. 1: End; The end position determines the delay time; if the delay is positive, output will be changed after reaching that point; if it is negative, output will be changed N milliseconds before the end position. 2: Path length determination; if the distance is positive, the start position distance + delay time = time to change output; if it is negative, the distance before end position + delay time = time to change the output.
delay	int	±1000ms delay time
distance	int	Only valid under Path mode ±2000mm delay distance
Return value	int	Success: 0 Failed: Parameter error
Remark	Supports a maximum of 8 SyncOutput instructions simultaneously.	

```
int type = 0;    //DO
int index = 3;
int ON = 1;
int Start = 0;
int End = 1;
int Path = 2;
int delay = 500;
int distance = 50;
double p1[6] = { 0, -200, 0, 0, 0, 0 };
double p2[6] = { 0, 150, 0, 0, 0, 0 };
lin_rel_pos(robot, 0, 0, p1 );
// 500ms after starting motion from p1, DO[3] changes to ON.
SyncOutput(robot, type, index, ON, Start, delay, distance);
// After starting motion from p1 and 1000ms before reaching the p2 point, DO[4] changes to ON.
SyncOutput(robot, type, 4, ON, End, -1000, distance);
// When the 50mm distance plus 500ms point is passed after starting motion from p1, DO[5] changes to ON.
SyncOutput(robot, type, 5, ON, Path, delay, distance);
// When the 50mm distance minus the 1000ms point is passed after starting motion from p1, DO[6] changes to ON; the output status of this option will change 1000ms earlier than // DO[5].
SyncOutput(robot, type, 6, ON, Path, -1000, distance);
lin_rel_pos( device_id, 0, 0, p2 );
```

3.4.33. Get DI range value- get_DI_range

int get_DI_range (HROBOT robot, int from_idx, int end_idx, int* value)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_idx	int	Start getting the value from the specified index
end_idx	int	Get value to a specified index
value	int*	Get the array value
Return value	int	Success: 0: Fail: error code

```
int from_idx = 1;
int end_idx = 48;
int values[48] = {0};
get_DI_range (robot, from_idx, end_idx, values);

// DI 1 = values[0]
// DI 2 = values[1]
// DI 3 = values[2]
…..
```

3.4.34. Get DI range simulation value- get_DI_sim_range

int get_DI_sim_range (HROBOT robot, int from_idx, int end_idx, int* value)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_idx	int	Start getting the value from the specified index
end_idx	int	Get value to a specified index
value	int*	Get the value array
Return value	int	Success: 0: Fail: error code

3.4.35. Get DI range comment- get_DI_comment_range

int get_DI_comment_range (HROBOT robot, int from_idx, int end_idx , wchar_t* comment, int& next_idx)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_idx	int	Start getting the comment from the specified index
end_idx	int	Get comment to a specified index
comment	wchar_t*	The first value of the array represents the overall length of this comment. The comment of each index uses “\t” as the separator.
next_idx	int&	When next_idx is 0, it means that the comment of the specified range has been fully retrieved; if next_idx is not 0, it means that comment needs to continue to be retrieved from that value (from_idx).
Return value	int	Success: 0: Fail: error code

```
int from_idx = 1;
int end_idx = 48;
int next_idx = 0;
wchar_t* comment[256] = {0};

for(int idx = from_idx; idx<=end_idx; idx++){
   get_DI_comment_range (robot, from_idx, end_idx, comment, next_idx);
   if(next_idx==0){
      break;
   }else{
      idx = next_idx;
   }
}

// comment[0] = comment length
// comment: 50DI1\tDI2\tDI3\t\t\t\t\tDI8\tDI9\tDI10\t
```

3.4.36. Get DO range value- get_DO_range

int get_DO_range (HROBOT robot, int from_idx, int end_idx, int* value)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_idx	int	Start getting the value from the specified index
end_idx	int	Get specified index value
value	int*	Get the array value
Return value	int	Success: 0: Fail: error code

3.4.37. Get DO range comment- get_DO_comment_range

int get_DO_comment_range (HROBOT robot, int from_idx, int end_idx , wchar_t* comment, int& next_idx)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_idx	int	Start getting the comment from the specified index
end_idx	int	Get comment to a specified index
comment	wchar_t*	The first value of the array represents the overall length of this comment. The comment of each index uses “\t” as the separator.
next_idx	int&	When next_idx is 0, it means that the comment of the specified range has been fully retrieved; if next_idx is not 0, it means that comment needs to continue to be retrieved from that value (from_idx).
Return value	int	Success: 0: Fail: error code

3.4.38. Get all FI values- get_FI_all

int get_FI_all (HROBOT robot, int* values)

Parameter	Data type	Description
robot	HROBOT	Device ID
values	int[8]	Get the array value
Return value	int	Success: 0: Fail: error code

3.4.39. Get all FO values-get_FO_all

int get_FO_all (HROBOT robot, int* values)

Parameter	Data type	Description
robot	HROBOT	Device ID
values	int[8]	Get the array value
Return value	int	Success: 0: Fail: error code

3.4.40. Get all timer statuses- get_timer_status_all

int get_timer_status_all (HROBOT robot, int* values)

Parameter	Data type	Description
robot	HROBOT	Device ID
values	int[20]	Get array's status
Return value	int	Success: 0: Fail: error code

3.4.41. Get all timer values- get_timer_value_all

int get_timer_value_all (HROBOT robot, int* values)

Parameter	Data type	Description
robot	HROBOT	Device ID
values	int[20]	Get the array value
Return value	int	Success: 0: Fail: error code

3.4.42. Get timer range comment- get_timer_comment_range

int get_timer_comment_range (HROBOT robot, int from_idx, int end_idx , wchar_t* comment, int& next_idx)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_idx	int	Start getting the comment from the specified index
end_idx	int	Get comment to a specified index
comment	wchar_t*	The first value of the array represents the overall length of this comment. The comment of each index uses “\t” as the separator.
next_idx	int&	When next_idx is 0, it means that the comment of the specified range has been fully retrieved; if next_idx is not 0, it means that comment needs to continue to be retrieved from that value (from_idx).
Return value	int	Success: 0: Fail: error code

3.4.43. Get all counter values- get_counter_value_all

int get_counter_value_all (HROBOT robot, int* values)

Parameter	Data type	Description
robot	HROBOT	Device ID
values	int[20]	Get the array value
Return value	int	Success: 0: Fail: error code

3.4.44. Get counter range comment- get_counter_comment_range

int get_counter_comment_range (HROBOT robot, int from_idx, int end_idx , wchar_t* comment, int& next_idx)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_idx	int	Start getting the comment from the specified index
end_idx	int	Get comment to a specified index
comment	wchar_t*	The first value of the array represents the overall length of this comment. The comment of each index uses “\t” as the separator.
next_idx	int&	When next_idx is 0, it means that the comment of the specified range has been fully retrieved; if next_idx is not 0, it means that comment needs to continue to be retrieved from that value (from_idx).
Return value	int	Success: 0: Fail: error code

3.4.45. Get Fieldbus Register SRW range value- get_fieldbus_rs_srw_range

int get_fieldbus_rs_srw_range (HROBOT robot, int from_idx, int end_idx , int* values)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_idx	int	Start getting the value from the specified index
end_idx	int	Get value to a specified index
values	int*	Get the array value
Return value	int	Success: 0: Fail: error code

3.4.46. Get Fieldbus Register SRR range value- get_fieldbus_rs_srr_range

int get_fieldbus_rs_srr_range (HROBOT robot, int from_idx, int end_idx , int* values)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_idx	int	Start getting the value from the specified index
end_idx	int	Get value to a specified index
values	int*	Get the array value
Return value	int	Success: 0: Fail: error code

3.4.47. Get Fieldbus Register range parameter- get_fieldbus_rs_parameter_range

int get_fieldbus_rs_parameter_range (HROBOT robot, int from_idx, int end_idx , wchar_t* para, int& next_idx)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_idx	int	Start getting the parameter from the specified index
end_idx	int	Get parameter to a specified index
comment	wchar_t*	The first value of the array represents the overall length of this parameter. The parameter of each index uses “\t” as the separator.
next_idx	int&	When next_idx is 0, it means that the parameter of the specified range has been fully retrieved; if next_idx is not 0, it means that parameter needs to continue to be retrieved from that value (from_idx).
Return value	int	Success: 0: Fail: error code

3.4.48. Get Fieldbus Register range comment- get_fieldbus_rs_comment_range

int get_fieldbus_rs_comment_range (HROBOT robot, int from_idx, int end_idx , wchar_t* comment, int& next_idx)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_idx	int	Start getting the comment from the specified index
end_idx	int	Get comment to a specified index
comment	wchar_t*	The first value of the array represents the overall length of this comment. The comment of each index uses “\t” as the separator.
next_idx	int&	When next_idx is 0, it means that the comment of the specified range has been fully retrieved; if next_idx is not 0, it means that comment needs to continue to be retrieved from that value (from_idx).
Return value	int	Success: 0: Fail: error code

3.4.49. Get all System input values- get_system_input_all

int get_system_input_all (HROBOT robot, int* values, , wchar_t* comment)

Parameter	Data type	Description
robot	HROBOT	Device ID
values	int*	Get the array value
comment	wchar_t*	The first value of the array represents the overall length of this comment. The comment of each index uses “\t” as the separator.
Return value	int	Success: 0: Fail: error code

3.4.50. Get all System output values- get_system_output_all

int get_system_output_all (HROBOT robot, int* values, , wchar_t* comment)

Parameter	Data type	Description
robot	HROBOT	Device ID
values	int*	Get the value array
comment	wchar_t*	The first value of the array represents the overall length of this comment. The comment of each index uses “\t” as the separator.
Return value	int	Success: 0: Fail: error code

3.4.51. Get all module input settings- get_MI_config_all

int get_MI_config_all (HROBOT robot, int* sim, int* values, int* types, int* starts, int* end)

Parameter	Data type	Description
robot	HROBOT	Device ID
sim	int[32]	Get simulated array
values	int[32]	Get the value array
types	int[32]	Get IO type array
starts	int[32]	Get the start array
end	int[32]	Get the end array
Return value	int	Success: 0: Fail: error code

3.4.52. Get all module output settings- get_MO_config_all

int get_MO_config_all (HROBOT robot, int* values, int* types, int* starts, int* end)

Parameter	Data type	Description
robot	HROBOT	Device ID
values	int[32]	Get the value array
types	int[32]	Get IO type array
starts	int[32]	Get the start array
end	int[32]	Get the end array
Return value	int	Success: 0: Fail: error code

3.4.53. Get the module input range comment- get_MI_comment_range

int get_MI_comment_range (HROBOT robot, int from_idx, int end_idx , wchar_t* comment, int& next_idx)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_idx	int	Start getting the comment from the specified index
end_idx	int	Get comment to a specified index
comment	wchar_t*	The first value of the array represents the overall length of this comment. The comment of each index uses “\t” as the separator.
next_idx	int&	When next_idx is 0, it means that the comment of the specified range has been fully retrieved; if next_idx is not 0, it means that comment needs to continue to be retrieved from that value (from_idx).
Return value	int	Success: 0: Fail: error code

3.4.54. Get the module output range comment- get_MO_comment_range

int get_MO_comment_range (HROBOT robot, int from_idx, int end_idx , wchar_t* comment, int& next_idx)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_idx	int	Start getting the comment from the specified index
end_idx	int	Get comment to a specified index
comment	wchar_t*	The first value of the array represents the overall length of this comment. The comment of each index uses “\t” as the separator.
next_idx	int&	When next_idx is 0, it means that the comment of the specified range has been fully retrieved; if next_idx is not 0, it means that comment needs to continue to be retrieved from that value (from_idx).
Return value	int	Success: 0: Fail: error code

3.4.55. Get SI range value- get_SI_range

int get_SI_range (HROBOT robot, int from_idx, int end_idx, int* value)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_idx	int	Start getting the value from the specified index
end_idx	int	Get value to a specified index
value	int*	Get the array value
Return value	int	Success: 0: Fail: error code

3.4.56. Get SI range simulation value- get_SI_sim_range

int get_SI_sim_range (HROBOT robot, int from_idx, int end_idx, int* value)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_idx	int	Start getting the value from the specified index
end_idx	int	Get value to a specified index
value	int*	Get the array value
Return value	int	Success: 0: Fail: error code

3.4.57. Get SO range value- get_SO_range

int get_SO_range (HROBOT robot, int from_idx, int end_idx, int* value)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_idx	int	Start getting the value from the specified index
end_idx	int	Get value to a specified index
value	int*	Get the array value
Return value	int	Success: 0: Fail: error code

3.4.58. Get SI range comment- get_SI_comment_range

int get_SI_comment_range (HROBOT robot, int from_idx, int end_idx , wchar_t* comment, int& next_idx)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_idx	int	Start getting the comments from the specified index
end_idx	int	Get comment to a specified index
comment	wchar_t*	The first value of the array represents the overall length of this comment. The comment of each index uses “\t” as the separator.
next_idx	int&	When next_idx is 0, it means that the comment of the specified range has been fully retrieved; if next_idx is not 0, it means that comment needs to continue to be retrieved from that value (from_idx).
Return value	int	Success: 0: Fail: error code

3.4.59. Get SO range comment- get_SO_comment_range

int get_SO_comment_range (HROBOT robot, int from_idx, int end_idx , wchar_t* comment, int& next_idx)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_idx	int	Start getting the comment from the specified index
end_idx	int	Get comment to a specified index
comment	wchar_t*	The first value of the array represents the overall length of this comment. The comment of each index uses “\t” as the separator.
next_idx	int&	When next_idx is 0, it means that the comment of the specified range has been fully retrieved; if next_idx is not 0, it means that comment needs to continue to be retrieved from that value (from_idx).
Return value	int	Success: 0: Fail: error code

3.4.60. Get PR range comment- get_PR_comment_array

int get_PR_comment_array (HROBOT robot, int* indexes, int from_idx, int len , wchar_t* comment, int& next_idx)

Parameter	Data type	Description
robot	HROBOT	Device ID
indexes	int*	This comment will get index value from the array. The comments will be captured from the index value carried over in order.
from_idx	int	Start getting the comment from the specified array index
len	int	To get indexes' total length
comment	wchar_t*	The first value of the array represents the overall length of this comment. The comment of each index uses "\t" as the separator.
next_idx	int&	When next_idx is 0, it means that the comment of the specified range has been fully retrieved; if next_idx is not 0, it means that comment needs to continue to be retrieved from that value (from_idx).
Return value	int	Success: 0: Fail: error code

3.4.61. Get all RI values- get_RI_all

int get_RI_all (HROBOT robot, int* values)

Parameter	Data type	Description
robot	HROBOT	Device ID
values	int[8]	Get the value array
Return value	int	Success: 0: Fail: error code

3.4.62. Get all RO values- get_RO_all

int get_RO_all (HROBOT robot, int* values)

Parameter	Data type	Description
robot	HROBOT	Device ID
values	int[8]	Get the array value
Return value	int	Success: 0: Fail: error code

3.4.63. Get all VO values- get_VO_all

int get_VO_all (HROBOT robot, int* values)

Parameter	Data type	Description
robot	HROBOT	Device ID
values	int[3]	Get the array value
Return value	int	Success: 0: Fail: error code

3.4.64. Set multiple DI values- set_DI_array

int set_DI_array (HROBOT robot, int* indexes, int* values, int len)

Parameter	Data type	Description
robot	HROBOT	Device ID
indexes	int*	Place the index value inorder to set into this array. The value will be set sequentially according to the index and value array of this array.
values	int*	Set the array value
len	int	Set it up array's length
Return value	int	Success: 0:` Fail: error code

```
int len = 48;
int indexes[48] = {0};
int values[48] = {0};
for(int idx=1; idx<=48;i++){
   indexes[idx-1] = idx;
   if(idx%2==0){
      values[idx-1] = 1;
   }else{
      values[idx-1] = 0;
   }
}
set_DI_array (robot, indexes, values, len);

// DI 1 = values[0]
// DI 2 = values[1]
// DI 3 = values[2]
…..
```

3.4.65. Set multiple DI simulation values- set_DI_sim_array

int set_DI_sim_array (HROBOT robot, int* indexes, int* values, int len)

Parameter	Data type	Description
robot	HROBOT	Device ID
indexes	int*	Place the index value inorder to set into this array. The value will be set sequentially according to the index and value array of this array.
values	int*	Set the array value
len	int	To set Length of array
Return value	int	Success: 0: Fail: error code

3.4.66. Set multiple DO values- set_DO_array

int set_DO_array (HROBOT robot, int* indexes, int* values, int len)

Parameter	Data type	Description
robot	HROBOT	Device ID
indexes	int*	Place the index value inorder to set into this array. The value will be set sequentially according to the index and value array of this array.
values	int*	Set the array value
len	int	To set Length of array
Return value	int	Success: 0: Fail: error code

3.4.67. Set multiple timer values- set_timer_value_array

int set_timer_value_array (HROBOT robot, int* indexes, int* values, int len)

Parameter	Data type	Description
robot	HROBOT	Device ID
indexes	int*	Place the index value inorder to set into this array. The value will be set sequentially according to the index and value array of this array.
values	int*	Set the value array
len	int	Length of array to set
Return value	int	Success: 0: Fail: error code

3.4.68. Set multiple counter values- set_counter_array

int set_counter_array (HROBOT robot, int* indexes, int* values, int len)

Parameter	Data type	Description
robot	HROBOT	Device ID
indexes	int*	Place the index value inorder to set into this array. The value will be set sequentially according to the index and value array of this array.
values	int*	Set the array value
len	int	Set the length of array
Return value	int	Success: 0: Fail: error code

3.4.69. Set multiple SI values- set_SI _array

int set_SI _array (HROBOT robot, int* indexes, int* values, int len)

Parameter	Data type	Description
robot	HROBOT	Device ID
indexes	int*	Place the index value inorder to set into this array. The value will be set sequentially according to the index and value array of this array.
values	int*	Set the array value
len	int	Set the length of array
Return value	int	Success: 0: Fail: error code

3.4.70. Set multiple SI simulation values- set_SI_sim _array

int set_SI_sim _array (HROBOT robot, int* indexes, int* values, int len)

Parameter	Data type	Description
robot	HROBOT	Device ID
indexes	int*	Place the index value inorder to set into this array. The value will be set sequentially according to the index and value array of this array.
values	int*	Set the array value
len	int	set the length of array
Return value	int	Success: 0: Fail: error code

3.4.71. Set multiple SO values- set_SO _array

int set_SO _array (HROBOT robot, int* indexes, int* values, int len)

Parameter	Data type	Description
robot	HROBOT	Device ID
indexes	int*	Place the index value inorder to set into this array. The value will be set sequentially according to the index and value array of this array.
values	int*	Set the array value
len	int	Set the length of array
Return value	int	Success: 0: Fail: error code

3.4.72. Set multiple Fieldbus SRW values- set_fieldbus_srw_array

int set_fieldbus_srw_array (HROBOT robot, int* indexes, int* values, int len)

Parameter	Data type	Description
robot	HROBOT	Device ID
indexes	int*	Place the index value inorder to set into this array. The value will be set sequentially according to the index and value array of this array.
values	int*	Set the array value
len	int	Set the length of array
Return value	int	Success: 0: Fail: error code

3.4.73. Set multiple MO values- set_MO _array

int set_MO _array (HROBOT robot, int* indexes, int* values, int len)

Parameter	Data type	Description
robot	HROBOT	Device ID
indexes	int*	Place the index value inorder to set into this array. The value will be set sequentially according to the index and value array of this array.
values	int*	Set the array value
len	int	Set the length of array
Return value	int	Success: 0: Fail: error code

3.4.74. Set multiple VO values- set_VO _array

int set_VO _array (HROBOT robot, int* indexes, int* values, int len)

Parameter	Data type	Description
robot	HROBOT	Device ID
indexes	int*	Place the index value inorder to set into this array. The value will be set sequentially according to the index and value array of this array.
values	int*	Set the array value
len	int	Set the length of array
Return value	int	Success: 0: Fail: error code

3.4.75. Set multiple RO values-set_RO _array

int set_RO _array (HROBOT robot, int* indexes, int* values, int len)

Parameter	Data type	Description
robot	HROBOT	Device ID
indexes	int*	Place the index value inorder to set into this array. The value will be set sequentially according to the index and value array of this array.
values	int*	Set the array value
len	int	Set the length of array
Return value	int	Success: 0: Fail: error code

3.5. Coordinate System Command Class

Reference function names and corresponding identities are shown in the below table:

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
set_base_number	Set base number	O	O	×
get_base_number	Get base number	O	O	O
define_base	Define base coordinates	O	×	×
get_base_data	Get base coordinates	O	O	O
set_tool_number	Set tool number	O	O	×
get_ tool _number	Get tool number	O	O	O
define_ tool	Define tool coordinates	O	×	×
get_ tool _data	Get tool coordinates	O	O	O
tool_calibration	Multi-point calibration of tool coordinates	O	O	×
base_calibration	Three-point calibration of base coordinates	O	O	×

The following section will describe the various function names, instruction formats, parameter settings and example descriptions.

3.5.1. Set base number- set_base_number

int set_base_number(HROBOT robot , int baseNum,int num)

Parameter	Data type	Description
robot	HROBOT	Device ID
baseNum	int	Base coordinate number
num	int	Selection of Base number (0-31)
Return value	int	Success: 0 Fail: error code

3.5.2. Get base number- get_base_number

int get_base_number(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: Base number (0-31) Fail: error code

```
set_base_number(robot,1);
int num=get_base_number(robot);
```

3.5.3. Define base coordinates- define_base

int define_base(HROBOT robot ,int baseNum ,double *coor)

Parameter	Data type	Description
robot	HROBOT	Device ID
baseNum	int	Base coordinate system number(1-31) Base number 0 cannot be defined Base number 1-30: Customizable coordinates
coor	double[6]	Coordinates {X,Y,Z,A,B,C} Range (2147418.112,- 2147418.112)
Return value	int	Successful: 0 Fail: error code

3.5.4. Get base coordinates- get_base_data

int get_base_data(HROBOT robot ,int num,double* coor)

Parameter	Data type	Description
robot	HROBOT	Device ID
num	int	To get a Base coordinate number (0-31)
coor	double[6]	Coordinates {X,Y,Z,A,B,C}
Return value	int	Success: 0 Fail: error code

3.5.5. Set tool number- set_tool_number

int set_tool_number(HROBOT robot ,int num)

Parameter	Data type	Description
robot	HROBOT	Device ID
num	int	Selection of Tool number (0-15)
Return value	int	Success: 0 Fail: error code

3.5.6. Get tool number- get_tool_number

int get_tool_number(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: Tool number (0-15) Fail: error code

```
set_tool_number(robot,10);          // set_tool_number
int vel=get_tool_number(robot);     // get_tool_number
```

3.5.7. Define tool coordinates- define_tool

int define_tool(HROBOT robot , int toolNum,double* coor)

Parameter	Data type	Description
robot	HROBOT	Device ID
toolNum	int	Define Tool number (1-15) Tool Coordinates number cannot be as; it can set from 1-15: Customizable coordinates
coor	double[6]	Coordinates {X,Y,Z,A,B,C} Range (2147418.112,- 2147418.112)
Return value	int	Success: 0 Fail: error code

3.5.8. Get tool coordinates- get_tool_data

int get_tool_data(HROBOT robot ,int num,double* coor)

Parameter	Data type	Description
robot	HROBOT	Device ID
num	int	Tool number to define (0-15)
coor	double [6]	Coordinates {X,Y,Z,A,B,C} Range of value (2147418.112,- 2147418.112)
Return value	int	Success: 0 Fail: error code

```
(1)
  double coor={0,50,300,0,0,0};
  define_tool(robot,2,coor);
  double ToolCoor[6];
  get_tool_data(robot,2,ToolCoor);
(2)
  double coor={0,0,100,0,0,0};
  define_base(robot,2,coor);
  double BaseCoor[6];
  get_base_data(robot,2,BaseCoor);
```

⇨ Define the tool coordinates and use get_tool_data to get information of the coordinates.

⇨ Define the base coordinates and use get_base_data to get information of the coordinates.

3.5.9. Multi-point calibration of tool coordinates- tool_calibration

Description:

Move & set the tool TCP from the 4 different directions & and their refereces. The robot control system calculates the tool center point (TCP) from different flange position. Use 4 different posture points to get the new tool base TCP point through software calculation (3-point calibration is used for the RS series), and that means to get the X, Y, Z, A, B and C values of the new tool coordinate system, in which the RS series do not have A and B values, and the C value (J4 rotation angle) is the first C value of the calibration point posture.

int tool_calibration (HROBOT robot, int type, double* p0_coord, double* p1_coord, double* p2_coord, double* p3_coord, double* result_coord)

Parameter	Data type	Description
robot	HROBOT	Device ID
type	int	0: RA and RT models 1: RS models
p0_coord	Double [6]	Measuring point 1
p1_coord	Double [6]	Measuring point 2
p2_coord	Double [6]	Measuring point 3
p3_coord	Double [6]	Measuring point 4; this point is not used as a reference for the RS series.
result_coord	Double [6]	New tool coordinate system value
Return value	int	Success: 0: Fail: error code
Remark	Move the TCP of the tool to be measured to a reference point from 3 different directions. The reference point can be randomly selected. The articulated robot control system calculates the TCP from different flange position values. Get the X, Y, Z, A, B and C values of the tool coordinate system after calibration	

C++ example file:

```
double p0_coord [6] = {0, 431, 26, 163, 2, 89};
double p1_coord [6] = {0, 470, 22, 179, -27, 91};
double p2_coord [6] = {0, 476, 18, -145, 0, 90};
double p3_coord [6] = {0, 486, -6,-176, 20, 89};
double result_coord [6] = {0};
int selected_tool = 3;

tool_calibration(robot, 0, p0_coord, p1_coord, p2_coord, p3_coord, result_coord);
define_tool(robot, selected_tool, result_coord);
```

3.5.10. Three-point calibration of base coordinates- base_calibration

Description: The user allocates a Cartesian Coordinate system (base coordinate system) to the work surface or workpiece in standard measurements. The origin of the base coordinate system is a point specified by the user. Use three points to get the new base coordinate system. During calibration, the information of the three points is needed to change the original robot coordinate system to the new user-specified coordinate system through software calculation.

1. The first point p0, use TCP to move to the origin of the new base coordinate system.
2. Second point p1, move TCP to a point on the positive X axis of the new base coordinate system.
3. Third point p2, move TCP to a point with a positive Y value on the XY plane.

int base_calibration (HROBOT robot, int type, double* p0_coord, double* p1_coord, double* p2_coord, double* result_coord)

Parameter	Data type	Description
robot	HROBOT	Device ID
type	int	0: ALL types of robot; Regardless of model number
p0_coord	Double [6]	Origin of the new base coordinate system
p1_coord	Double [6]	A point on the positive X axis of the new base coordinate system
p2_coord	Double [6]	A point on the new base coordinate system with a positive Y value
result_coord	Double [6]	New base coordinate system value
Return value	int	Success: 0: Fail: error code

C++ example file:

```
double p0_coord [6] = {0, 400, 0, 0, 0, 0};
double p1_coord [6] = {100, 400, 0, 0, 0, 0};
double p2_coord [6] = {0, 500, 0, 0, 0, 0};
double result_coord [6] = {0};
int selected_base = 3;

base_calibration(robot, 0, p0_coord, p1_coord, p2_coord, result_coord);
define_base(robot, selected_base, result_coord);
```

3.6. Task Command Class

Reference function names and corresponding identities are shown in the below table:

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
set_rsr	Set RSR	O	×	×
get_rsr_prog_name	Get RSR program name	O	O	O
remove_rsr	Remove RSR	O	×	×
ext_task_start	RSR/PNS Start external trigger task	O	O	×
task_ start	Start task	O	O	×
task_hold	Hold current task	O	O	×
task_continue	Continue current task	O	O	×
task_abort	Stop current task	O	O	×
get_execute_file_name	Get the currently executing program file name	O	O	O

The following section will describe the various function names, instruction formats, parameter settings and example descriptions.

3.6.1. Set RSR- set_rsr

int set_rsr(HROBOT robot , char* file_name,int index)

Parameter	Data type	Description
robot	HROBOT	Device ID
file_name	char*	Program file name to set
index	int	Task setting number RSR:(1-4)
Return value	int	Success: 0 Fail: error code

- Caution! If an operator class connection is used for SDK, the RSR of the controller will become invalid.

```
char* filename = “file.hrb”;
set_rsr(robot,filename, 1);
```

3.6.2. Get RSR program name- get_rsr_prog_name

int get_rsr_prog_name (HROBOT robot, int index, char* file_name)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Number of RSR to get RSR:(1-4)
file_name	char*	Get program file name
Return value	int	Success: 0 Fail: error code

```
char* filename;
get_rsr_prog_name(robot, 1, filename);
```

3.6.3. Remove RSR- remove_rsr

int remove_rsr (HROBOT robot , int index)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Task setting number RSR:(1-4)
Return value	int	Success: 0 Fail: error code

```
remove_rsr (robot, 1);
```

3.6.4. RSR/PNS enables external trigger tasks- ext_task_start

int ext_task_start(HROBOT robot , int mode,int select)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	External trigger mode 0: RSR mode 1: PNS mode
select	int	Task setting number RSR:(1-4) PNS:(1-2047)
Return value	int	Success: 0 Fail: error code

- RSR/PNS task number must be set before using this command
- If there are other motion commands when executing this command, this command will fail. The motion_abort instruction must be executed before executing to clear motion queues or wait for the execution of the motion command to complete.

3.6.5. Start task- task_ start

int task_ start(HROBOT robot ,char* file_name)

Parameter	Data type	Description
robot	HROBOT	Device ID
file_name	char*	Task name; the specified file name must include ".hrb" The task must exist in HRSS
Return value	int	Success: 0 Fail: error code

- If there are other motion commands when executing this command, this command will fail. The motion_abort instruction must be executed before executing to clear motion queues or wait for the execution of the motion command to complete.

```
double pos1[6] = { 50, 0, 0, 0, -90, 0 };
double pos2[6] = { -50, 0, 0, 0, -90, 0 };
set_motor_state(s, 1);    // servo on

// wait for servo on
while (!get_motor_state(s)) {
      Sleep(10);
}

// execute motion command
set_command_id(s, 20);
ptp_axis(s, pos1);    // this motion id will be 20
set_command_id(s, 21);
ptp_axis(s, pos2);    // this motion id will be 21

// task start will fail because there are motion command in motion queue
task_start(s, "program_Test.hrb"));    // fail

// clean command in motion queue
motion_abort(s);

// wait for command count = 0
while (get_command_count(s)) {
      Sleep(100);
}

// task start will succeed
task_start(s, "program_Test.hrb"); // successful

// task start will fail because there is a task exist
task_start(s, "program_Test.hrb"); // fail
```

3.6.6. Pause task- task_ hold

int task_ hold(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: 0 Fail: error code

3.6.7. Continue task- task_ continue

int task_ continue(HROBOT robot)

Parameter	Data type	Description
Robot	HROBOT	Device ID
Return value	int	Success: 0 Fail: error code

3.6.8. Stop task- task_ abort

int task_ abort(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: 0 Fail: error code

```
(1)
    ext_task_start(s, 1, 9);
    Sleep(3000);
    ext_task_start(s, 1, 94);
    task_hold(s);
    Sleep(1000);
    task_conti(s);
    Sleep(1000);
    task_abort(s);
    Sleep(100);
(2)
    ext_task_start(s, 0, 4);
    Sleep(3000);
    task_abort(s);
    Sleep(100);
(3)
    task_start(s, "task1.hrb");
    Sleep(3000);
    task_abort(s);
    Sleep(100);
```

(1). Start PNS mode task number 9
Run for three seconds and then start PNS task 94

- If execution of task 9 has been completed, task 94 will be executed sequentially
- If task 9 did not finish executing, task start will fail and the error code 4013 will be returned. Task 9 will continue executing Task will continue one second after the task is pausedContinue executing for one second and then stop the task

(2). Start RSR mode task number 4
Run for three seconds and then stop the task

(3). The start task name is task “task1”
Run for three seconds and then stop the task

3.6.9. Get name of task currently executing- get_execute_file_name

int get_execute_file_name (HROBOT robot , char* filename)

Parameter	Data type	Description
robot	HROBOT	Device ID
filename	cahr*	Current task name
Return value	int	Success: 0 Fail: error code

```
char* execute_file_name = new char[256];
get_execute_file_name(robo, execute_file_name);
std::cout << "Execute file name: " << execute_file_name << std::endl;
delete[] execute_file_name;
```

3.7. File management Command Class

Reference function names and corresponding identities are shown in the below table:

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
send_file	Upload HRB to controller	O	O	×
download_file	Download HRB to local terminal	O	O	O
delete_file	Delete robot motion file	O	O	×
delete_folder	Delete robot motion file folder	O	O	×
new_folder	New robot motion file folder	O	O	×
file_rename	Rename robot motion file	O	O	×
file_drag	Drag robot motion file	O	O	×
get_prog_number	Get file's number	O	O	O
get_prog_name	Get file name	O	O	O

The following section will describe the various function names, instruction formats, parameter settings and example descriptions.

3.7.1. Download HIWIM Robot language file- download_file

int download_file (HROBOT robot , char * from_path , char * to_path)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_path	char*	Path of hrb file to download.
to_path	char*	Path after hrb file was downloaded.
Return value	int	Success: 0 Fail: error code

```
HROBOT s;
char form_path[255] = "hrss.hrb";
char to_path [255] = " HRL_files\\hrss.hrb";
download_file(robot, form_path, to_path);
```

3.7.2. Upload HIWIN Robot language file- send_file

int send_file(HROBOT robot , char* from_path, char* to_path)

Parameter	Data type	Description
robot	HROBOT	Device ID
from_path	char*	The path of the hrb file to upload must be specified.
to_path	char*	Path of the hrb file to save; this file will be saved under the Program folder of HRS.
Return value	int	Success: 0 Fail: error code
Remark	If the file name is the same, it will be overwritten directly	

```
HROBOT robot;
send_file(robot , "folder1\\a123.hrb", "a123.hrb");
```

3.7.3. Delete robot motion file- delete_file

int delete_file(HROBOT robot , char* file_path)

Parameter	Data type	Description
robot	HROBOT	Device ID
file_path	char*	Delete the hrb file in the specified path; this file will be in the Program HRS folder.
Return value	int	Success: 0 Fail: error code

3.7.4. Delete robot motion file folder- delete_folder

int delete_folder (HROBOT robot , char* file_path)

Parameter	Data type	Description
robot	HROBOT	Device ID
file_path	char*	Delete the folder of the specified path; this folder will be in the HRS Program folder.
Return value	int	Success: 0 Fail: error code

3.7.5. New robot motion file folder- new_folder

int new_folder (HROBOT robot , char* name)

Parameter	Data type	Description
robot	HROBOT	Device ID
name	char*	New folder name; this folder will be in the HRS Program folder.
Return value	int	Success: 0 Fail: error code

3.7.6. Rename robot motion file- file_rename

int file_rename (HROBOT robot , char* old_file_name, char* new_file_name)

Parameter	Data type	Description
robot	HROBOT	Device ID
old_file_name	char*	The path of the hrb file to rename must be specified.
new_file_name	char*	Change to the specified file name.
Return value	int	Success: 0 Fail: error code

3.7.7. Drag robot motion file- file_drag

int file_drag (HROBOT robot , char* old_file_name, char* new_file_name)

Parameter	Data type	Description
Robot	HROBOT	Device ID
old_file_name	char*	The path of the hrb file to drag must be specified.
new_file_name	char*	Change to the specified file path.
Return value	int	Success: 0 Fail: error code

```
HROBOT robot;
new_folder(robot , "test_folder");
send_file(robot , "folder1/a123.hrb", "test_folder/a123.hrb");
file_drag(robot , "test_folder/a123.hrb", "b456.hrb");
file_rename (robot , "b456.hrb ", "c789.hrb");
delete_file (robot , "c789.hrb");
delete_folder(robot , "test_folder");
```

3.7.8. Get number of files- get_prog_number

int get_prog_number(HROBOT robot)

Parameter	Data type	Description
Robot	HROBOT	Device ID
Return value	int	Return the total number of files and folders in the Program folder

3.7.9. Get file name- get_prog_name

int get_prog_name(HROBOT robot , int file_index, char* file_name)

Parameter	Data type	Description
Robot	HROBOT	Device ID
file_index	int	Get the n^{th} file, including the folder
file_name	char*	➢ File name retrieved ➢ Sort according to file and folder names; folders are also considered single name
Return value	int	Success: 0 Fail: error code
Remark	The number of files must be retrieved first with get_prog_number before this API can be executed.	

```
char str[20] = “”;
int num = HRobot.get_prog_number( robot );
for ( int i = 0; i < num; i++ ) {
      get_prog_name( robot, i, str ) ;
      printf(“name: %s”, str);
}
```

3.8. Controller Command Class

Reference function names and corresponding identities are shown in the below table:

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
get_hrss_mode	Servo motor setting	O	O	O
set_motor_state	Set servo motor state	O	O	×
get_motor_state	Get servo motor state	O	O	O
set_operation_mode	Set operation mode	O	O	×
get_operation_mode	Get operation mode	O	O	O
clear_alarm	Clear alarm	O	×	×
update_hrss	Update HRSS version	O	×	×
get_robot_info	Get robot information	O	O	O

The following section will describe the various function names, instruction formats, parameter settings and example descriptions.

3.8.1. Get current mode of HRSS- get_hrss_mode

int get_hrss_mode(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	T1 mode: 0 AUTO mode: 1 T2 mode: 2 EXT mode: 3 Fail: error code

```
ret=get_hrss_mode (robot);
```

3.8.2. Servo setting- set_motor_state

int set_motor_state(HROBOT robot ,int state)

Parameter	Data type	Description
robot	HROBOT	Device ID
state	int	0: Servo off 1: Servo on Wait approximately 100ms after the servo has started; before other motion commands can be sent and executed
Return value	int	Successful: 0 Fail: error code

3.8.3. Get servo status- get_motor_state

int get_motor_state(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Servo off: 0 Servo on: 1 Fail: error code

```
if (get_motor_state(robot)==0){
    set_motor_state(robot,1);
 }
```

(1). Determine whether the motor is excited; excite the motor if it is not excited.

3.8.4. Set operation mode- set_operation_mode

int set_operation_mode (HROBOT robot, int mode)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Manual mode: 0 Auto mode: 1
Return value	int	Success: 0 Fail: error code

- **Manual mode**
  - Motion command speed is limited to 250mm/s
  - Jog instructions can be used
  - The overall speed will be set as 10%
  - Acceleration ratio cannot be set
  - Straight line speed and point to point speed ratio cannot be set
- **Auto mode**
  - Motion command speed limit varies according to model number
  - Jog instructions cannot be used
  - The overall speed will be set as 10%
  - When executing the motion program under this mode, the standard process must be followed to continue motion; while staying offline and use the disconnect command to disconnect.

3.8.5. Get operation mode- get_operation_mode

int get_operation_mode (HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Manual mode: 0 Auto mode: 1 Fail: error code

3.8.6. Error cleared- clear_alarm

int clear_alarm (HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: 0 Fail: error code

```
//if your arm get error
clear_alarm(robot);
```

3.8.7. Update HRSS- update_hrss

int update_hrss (HROBOT robot, char* path)

Parameter	Data type	Description
robot	HROBOT	Device ID
path	char*	Update file path
Return value	int	Success: 0 Fail: error code Update file error: 4020 File transfer failed: 4021 Update file decompression failed: 4022 Controller has insufficient space: 4023 Update file does not exist in the specified path: 4024

```
char path[255]=” C:/ HRSS 3.2.12.3940_update.exe”;
update_hrss(robot,path);
```

3.8.8. Get articulated robot information- get_robot_info

int get_robot_info (HROBOT robot, int page_sel, int tool_num, int base_num, char* info, bool is_ready)

Parameter	Data type	Description
robot	HROBOT	Device ID
page_sel	int	Select a page to get the callback data of the page. 0: Articulated robot coordinates position information 1: Digital input 2: Digital output 3: FieldBus I/O, 4: Function I/O, 5: Timer, 6: Counter 7: Communication TCP/IP 8: Communication RS232, 9: Origin setting 10: Electric gripper 11: Pick-on-the-fly visual configuration 12: Pick-on-the-fly visual object 13: System I/O 14: Module input 15: Module output 16: Register 17: Utilization rate 18: Calibration
tool_num	int	Get information of the selected tool in robot info
base_num	int	Get information of the selected base in robot info
info	char*	The information includes the following data, and each data is separated with “,”. 0: HRSS mode 1: Operation mode 2: Motion status 3: Acceleration time 4: Number of alarms 5: Function output

Parameter	Data type	Description
		6: Tool number 7: Base number 8: Tool coordinates, 6 values in total 14: Base coordinates, 6 values in total
bool	is_ready	0: Get latest articulated robot information 1: Use the previous articulated robot information automatically received
Return value	int	Success: 0 Fail: error code
Remark	➢ To get Callback Notifiction, the get_current_position command must be given first.	

3.9. Jog Command Class

Reference function names and corresponding identities are shown in the below table:

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
jog	Jog	O	×	×
jog_stop	Stop jog	O	×	×
jog_home	Jog return to home point	O	×	×

The following section will describe the various function names, instruction formats, parameter settings and example descriptions.

3.9.1. Jog- jog

int jog(HROBOT robot ,int space_type, int index, int dir)

Parameter	Data type	Description
robot	HROBOT	Device ID
space	int	Coordinate system type 0: Base coordinate system 1:Joint coordinate system 2:Tool coordinate system 3: External axis coordinate system
index	int	Jog object Cartesian coordinate system(X:0,Y:1,Z:2,A:3,B:4,C:5) Joint coordinate system (A1:0, A2:1, A3:2, A4:3, A5:4, A6:5) Tool coordinate system (Tx:0, Ty:1, Tz:2, RTx:3, RTy:4, RTz:5) External axis coordinate system (E1:0, E2:1, E3:2)
direction	int	Direction 1: Positive direction -1: Negative direction
Return value	int	Success: 0 Fail: error code

- Only valid under manual mode

- Jog unit description:
  (1).Cartesian coordinate system X, Y and Z axes are in millimeter (mm), A, B and C axis are in degree.
  (2). In the joint coordinate axis, the body of the articulated robot labelled as Joint1 (J1) which denoted as A1 in software, and physical robot Joint2 (J2) denoted as A2 in software, Joint3 (J3) means A3 in software and so on until Joint6.
  (3). Tool coordinate system's Tx, Ty and Tz axes are in millimeter (mm), RTx, RTy and RTz axes are in degree.
  (4). External axis coordinate system straight line unit is a millimeter (mm), and the rotation unit is a degree.

3.9.2. Jog reset- jog_home

int jog_home(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: 0 Fail: error code

- Only valid under safe line speed mode

3.9.3. Jog stop- jog_stop

int jog_stop(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: 0 Fail: error code

Only valid under safe line speed mode

3.10. Motion Command Class

Reference function names and corresponding identities are shown in the below table:

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
ptp_pos	Absolute coordinate position of PTP motion	O	O	×
ptp_axis	Absolute joint angle of PTP motion	O	O	×
ptp_rel_pos	Relative coordinate position of PTP motion	O	O	×
ptp_rel_axis	Relative joint angle of PTP motion	O	O	×
ptp_pr	Position register of PTP motion	O	O	×
lin_pos	Absolute coordinate position of linear motion	O	O	×
lin_axis	Absolute joint angle of linear motion	O	O	×
lin_rel_pos	Relative coordinate position of linear motion	O	O	×
lin_rel_axis	Relative joint angle of linear motion	O	O	×
lin_pr	Position register of linear motion	O	O	×
circ_axis	Absolute joint angle arc motion	O	O	×
circ_pos	Absolute coordinates position arc motion	O	O	×
circ_pr	Position register of circular motion	O	O	×
motion_hold	Hold motion	O	O	×
motion_continue	Continue motion	O	O	×

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
motion_abort	Stop motion	O	O	×
motion_delay	Delay motion	O	O	×
set_command_id	Set motion command number	O	O	×
get_command_id	Get current motion command number	O	O	O
get_command_count	Get current motion command from command queue	O	O	O
get_motion_state	Get current motion state	O	O	O
remove_command	Cancel unexecuted motion command	O	O	×
remove_command_tail	Cancel unexecuted motion command from the tail	O	O	×
ext_ptp_axis	External axis absolute joint angle point to point motion	O	O	×
ext_ptp_pos	External axis absolute coordinates position point to point motion	O	O	×
ext_lin_axis	External axis absolute joint angle straight line motion	O	O	×
ext_lin_pos	External axis absolute coordinates position straight line motion	O	O	×
ext_asyptp	External axis non-synchronized joint angle point to point motion	O	O	×

The following section will describe the various function names, instruction formats, parameter settings and example descriptions.

3.10.1. Absolute coordinates position point to point motion- ptp_pos

int ptp_pos(HROBOT robot, int mode, double* p)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Disable smooth function 1: Smooth according to the speed of the two-line segments
p	double[6]	Cartesian coordinate system{X,Y,Z,A,B,C} Range of value(2147418.112,- 2147418.112)
Return value	int	Success: 0 Fail: error code

3.10.2. Absolute joint angle point to point motion- ptp_axis

int ptp_axis(HROBOT robot, int mode, double* p)

Parameter	Data type	Description
robot	HROBOT	Device ID
p	Double [6]	Joint coordinate system{A1,A2,A3,A4,A5,A6} Range of value(2147418.112,- 2147418.112)
mode	int	Smooth mode 0: Disable smooth function 1: Smooth according to the speed of the two-line segments
Return value	int	Success: 0 Fail: error code

Remark: In the joint coordinate axis, the body of the articulated robot labelled as Joint1 (J1) which denoted as A1 in software, and physical robot Joint2 (J2) denoted as A2 in software, Joint3 (J3) means A3 in software and so on until Joint6.

3.10.3. Relative coordinates position point to point motion- ptp_rel_pos

int ptp_rel_pos(HROBOT robot, int mode, double* p)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Disable smooth function 1: Smooth according to the speed of the two line segments
p	Double [6]	Cartesian coordinate system{X,Y,Z,A,B,C} Range of value(2147418.112,- 2147418.112)
Return value	int	Success: 0 Fail: error code

3.10.4. Relative joint angle point to point motion- ptp_rel_axis

int ptp_rel_axis(HROBOT robot, int mode, double* p)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Disable smooth function 1: Smooth according to the speed of the two line segments
p	Double [6]	Joint coordinate system{A1,A2,A3,A4,A5,A6} Range of value(2147418.112,- 2147418.112)
Return value	int	Success: 0 Fail: error code

3.10.5. Point to point motion of position register- ptp_pr

int ptp_pr(HROBOT robot, int mode, int p)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Disable smooth function 1: Smooth according to the speed of the two line segments
p	int	Position register number (1-100)
Return value	int	Success: 0 Fail: error code

```
double AxishomePoint[6]={0,0,0,0,-90,0};
ptp_axis(s,0, AxishomePoint);    // RA robot home point

double TargetCartPoint[6]={217.5 ,58.5 ,601.659 ,60.8 ,35.5 ,95.7}
ptp_pos(s,0, TargetCartPoint);    // move to Point you want in Cart Space using PTP

double Xoffset[6]={-50,0,0,0,0,0};
ptp_rel_pos(s, 0, Xoffset); // new point will be {167.5,58.5,601.659,60.8,35.5,95.7}

set_pr(robot,100,1, AxishomePoint,0,0);
// set pr no.100 data with joint space{0,0,0,0,-90,0} in base 0 tool 0
ptp_pr(robot, 0,100);    //move to position register no.100 using PTP
```

3.10.6. Absolute coordinates position straight line motion- lin_pos

int lin_pos(HROBOT robot, int mode, double smooth_value, double* p)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Disable smooth function 1: Bezier curve smoothing percentage 2: Bezier curve smoothing radius 3: Smooth according to the speed of the two line segments
smooth_value	double	mode is 0: Invalid mode is 1: Smoothing percentage (1-100%) mode is 2: Smoothing radius (mm) mode is 3: Invalid
p	Double [6]	Cartesian coordinate system{X,Y,Z,A,B,C} Range of value(2147418.112,- 2147418.112)
Return value	int	Success: 0 Fail: error code

3.10.7. Absolute joint angle straight line motion- lin_axis

int lin_axis(HROBOT robot, int mode, double smooth_value, double* p)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Disable smooth function 1: Bezier curve smoothing percentage 2: Bezier curve smoothing radius 3: Smooth according to the speed of the two line segments
smooth_value	double	mode is 0: Invalid mode is 1: Smoothing percentage (1-100%) mode is 2: Smoothing radius (mm) mode is 3: Invalid
p	Double [6]	Joint coordinate system{A1,A2,A3,A4,A5,A6} Range of value(2147418.112,- 2147418.112)
Return value	int	Successful: 0 Fail: error code

Remark: In the joint coordinate axis, the body of the articulated robot labelled as Joint1 (J1) which denoted as A1 in software, and physical robot Joint2 (J2) denoted as A2 in software, Joint3 (J3) means A3 in software and so on until Joint6.

3.10.8. Relative coordinates position straight line motion- lin_rel_pos

int lin_rel_pos(HROBOT robot, int mode, double smooth_value, double* p)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Disable smooth function 1: Bezier curve smoothing percentage 2: Bezier curve smoothing radius 3: Smooth according to the speed of the two line segments
smooth_value	double	mode is 0: Invalid mode is 1: Smoothing percentage (1-100%) mode is 2: Smoothing radius (mm) mode is 3: Invalid
p	Double [6]	Cartesian coordinate system{X,Y,Z,A,B,C} Range of value(2147418.112,- 2147418.112)
Return value	int	Success: 0 Fail: error code

3.10.9. Relative joint angle straight line motion- lin_rel_axis

int lin_rel_axis(HROBOT robot , int mode, double smooth_value, double* p)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Disable smooth function 1: Bezier curve smoothing percentage 2: Bezier curve smoothing radius 3: Smooth according to the speed of the two-line segments
smooth_value	double	mode is 0: Invalid mode is 1: Smoothing percentage (1-100%) mode is 2: Smoothing radius (mm) mode is 3: Invalid
p	Double [6]	Joint coordinate system{A1,A2,A3,A4,A5,A6} Range of value(2147418.112,- 2147418.112)
Return value	int	Success: 0 Fail: error code

Remark: In the joint coordinate axis, the body of the articulated robot labelled as Joint1 (J1) which denoted as A1 in software, and physical robot Joint2 (J2) denoted as A2 in software, Joint3 (J3) means A3 in software and so on until Joint6.

3.10.10.Straight line motion of position register- lin_pr

int lin_pr(HROBOT robot, int mode, double smooth_value, int p)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Disable smooth function 1: Bezier curve smoothing percentage 2: Bezier curve smoothing radius 3: Smooth according to the speed of the two line segments
smooth_value	double	mode is 0: Invalid mode is 1: Smoothing percentage (1-100%) mode is 2: Smoothing radius (mm) mode is 3: Invalid
p	int	Position register number (1-100)
Return value	int	Success: 0 Fail: error code

```
double AxishomePoint[6]={0,0,0,0,-90,0};
lin_axis(s, 0, 10, AxishomePoint);    // RA robot home point

double TargetCartPoint[6]={217.5 ,58.5 ,601.659 ,60.8 ,35.5 ,95.7}
lin_pos(s, 0, 0, TargetCartPoint);

double Xoffset[6]={-50,0,0,0,0,0};
lin_rel_pos(s, 0, 0, Xoffset); // new point will be
{167.5,58.5,601.659,60.8,35.5,95.7}

set_pr(robot,100,1, AxishomePoint,0,0);
// set pr no.100 data with joint space{0,0,0,0,-90,0} in base 0 tool 0
lin_pr(robot, 0, 0, 100);    //move to position register no.100 using LIN
```

3.10.11. Absolute coordinates position arc motion- circ_pos

int circ_pos(HROBOT robot, int mode, double* p_aux,double* p_end)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Disable smooth function 1: Smooth according to the speed of the two-line segments
p_aux	Double [6]	Arc point of arc motion Cartesian coordinate system{X,Y,Z,A,B,C} Range of value(2147418.112,- 2147418.112)
p_end	Double [6]	End point of arc motion Cartesian coordinate system{X,Y,Z,A,B,C} Range of value(2147418.112,- 2147418.112)
Return value	int	Success: 0 Fail: error code

3.10.12. Joint coordinates position arc motion- circ_axis

int circ_axis(HROBOT robot, int mode, double* p_aux,double* p_end)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Disable smooth function 1: Smooth according to the speed of the two line segments
p_aux	Double [6]	Arc point of arc motion Joint coordinates {A1, A2, A3, A4, A5, A6} Range of value (2147418.112,- 2147418.112)
p_end	Double [6]	End point of arc motion Joint coordinates {A1, A2, A3, A4, A5, A6} Range of value(2147418.112,- 2147418.112)
Return value	int	Success: 0 Fail: error code

Remark: In the joint coordinate axis, the body of the articulated robot labelled as Joint1 (J1) which denoted as A1 in software, and physical robot Joint2 (J2) denoted as A2 in software, Joint3 (J3) means A3 in software and so on until Joint6.

```
  //Declare p1,p2
double aux_p[6] = { 174.5, 368, 164.7, -180, 0, 90 };
double end_p[6] = { 51, 368, -69.7, 180, 0, 90 };
double homeAxis[6] = { 0, 0, 0, 0, -90, 0};
ptp_axis(s, 0, homeAxis);     // ptp to home point
circ_pos(s, 0, aux_p, end_p);

double aux_p[6] = { -20, 0, -34, 0, -56, -20 };
double end_p[6] = { -13.5, 22.4, -28.4, 0, -96, -13.5 };
circ_axis(s, 0, aux_p, end_p);
```

3.10.13.Arc motion of position register- circ_pr

int circ_pr(HROBOT robot, int mode, int p1,int p2)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Disable smooth function 1: Smooth according to the speed of the two line segments
p1	int	Position register number (0-100)
p2	int	Position register number (0-100)
Return value	int	Success: 0 Fail: error code

```
//Set position register 1,2
double aux_p1[6] = { 174.5, 368, 164.7, -180, 0, 90 };
double end_p1[6] = { 51, 368, -69.7, 180, 0, 90 };
double aux_p2[6] = { -20, 0, -34, 0, -56, -20 };
double end_p2[6] = { -13.5, 22.4, -28.4, 0, -96, -13.5 };
set_pr_type(s, 1, 0);   // set_pr type to cart space
set_pr_type(s, 2, 0);
set_pr_type(s, 3, 1);   // set_pr type to joint space
set_pr_type(s, 4, 1);
set_pr_coordinate(s, 1, aux_p1);
set_pr_coordinate(s, 2, end_p1);
set_pr_coordinate(s, 3, aux_p2);
set_pr_coordinate(s, 4, end_p2);
circ_pr(robot, 0, 1,2);   // circle motion
```

3.10.14.Motion paused- motion_hold

int motion_hold(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: 0 Fail: error code

- **When tasks are executing, the instruction will fail.**

3.10.15.Motion continue- motion_continue

int motion_continue(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: 0 Fail: error code

➢ **When tasks are executing, the instruction will fail.**

3.10.16.Motion stop- motion_abort

int motion_abort(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: 0 Fail: error code

➢ **When tasks are executing, the instruction will fail.**

3.10.17.Motion delay- motion_delay

int motion_delay(HROBOT robot, int delay)

Parameter	Data type	Description
robot	HROBOT	Device ID
delay	int	Delay time, unit millisecond
Return value	int	Success: 0 Fail: error code

➢ **When tasks are executing, the instruction will fail.**

```
double pos1[6] = { 50, 0, 0, 0, -90, 0 };
double pos2[6] = { -50, 0, 0, 0, -90, 0 };
set_motor_state(s, 1);    // servo on

// wait for servo on
while (!get_motor_state(s)) {
      Sleep(10);
}

// execute motion command
set_command_id(s, 20);
ptp_axis(s, pos1);    // this motion id will be 20
set_command_id(s, 21);
ptp_axis(s, pos2);    // this motion id will be 21
motion_delay(s, 1000);
motion_hold(s);
motion_continue(s);
motion_abort(s);


// task start will fail because there are motion command in motion queue
task_start(s, "program_Test.hrb"));    // fail

// clean command in motion queue
motion_abort(s);

// wait for command count = 0
while (get_command_count(s)) {
      Sleep(100);
}

// task start will succeed
task_start(s, "program_Test.hrb"); // successful

// task start will fail because there is a task exist
task_start(s, "program_Test.hrb"); // fail
```

```
// motion planning command will fail
motion_delay(s, 1000); // fail
motion_hold(s); // fail
motion_continue(s); // fail
motion_abort(s); // fail
set_command_id(s, 20); // fail
// task stop
task_abort(s);
```

3.10.18. Set motion command number- set_command_id

int set_command_id(HROBOT robot ,int id)

Parameter	Data type	Description
Robot	HROBOT	Device ID
Id	int	Number of commands to set
Return value	int	Success: 0 Fail: error code

- **When tasks are executing, the instruction will fail.**

3.10.19. Get motion command number- get_command_id

int get_command_id(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: Motion command number Fail: error code

```
  set_command_id(robot,10);
  ptp_pos(robot,p1);

  set_command_id(robot,11);
  ptp_pos(robot,p2);

  set_command_id(robot,12);
  ptp_pos(robot,p3);

  set_command_id(robot,13);
  ptp_pos(robot,p4);

  set_command_id(robot,14);
  ptp_pos(robot,p5);

  while(get_motion_state(robot
)!=1){
    comId=get_command_id(robot);
  }
```

3.10.20. Get number of commands in the motion command queue- get_command_count

int get_command_count(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: Number of motion commands Fail: error code

3.10.21. Get current motion status- get_motion_state

int get_motion_state(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: Current motion status 0: De-excitation status 1: Wait, idle status 2: Running status 3: Paused status 4: Delay status 5: Moving status Failed: Error code

- **Idle status: No motion command**

3.10.22. Remove single command in the motion command queue- remove_command

int remove_command(HROBOT robot, int num)

Parameter	Data type	Description
Robot	HROBOT	Device ID
num	int	Command to remove
Return value	int	Success: 0 Fail: error code

3.10.23. Remove multiple latest commands in the motion command queue- remove_command_tail

int remove_command_tail(HROBOT robot, int num)

Parameter	Data type	Description
robot	HROBOT	Device ID
num	int	Number of latest commands removed
Return value	int	Success: 0 Fail: error code

3.10.24. External axis absolute joint angle point to point motion- ext_ptp_axis

int ext_ptp_axis (HROBOT robot, int mode, double* axis)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Disable smooth function 1: Smooth according to the speed of the two-line segments
axis	Double [9]	Joint coordinate system{A1, A2, A3, A4, A5, A6, E1, E2, E3} Range of value(2147418.112,- 2147418.112) Four-axis SCARA articulated robot, A5 and A6 values are set as 0, The external axis mode must be set as synchronized (Sync) to move; the external axis will not move if disabled or in asynchronized mode.
Return value	int	Success: 0 Fail: error code

3.10.25. External axis absolute coordinates position point to point motion- ext_ptp_pos

int ext_ptp_pos (HROBOT robot, int mode, double* axis)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Disable smooth function 1: Smooth according to the speed of the two-line segments
axis	Double [9]	Cartesian coordinate system{X, Y, Z, A, B, C, E1, E2, E3} Range of value(2147418.112,- 2147418.112) Four-axis SCARA articulated robot, A and B values are set as 0, The external axis mode must be set as synchronized (Sync) to move; the external axis will not move if disabled or in asynchronized mode.
Return value	int	Success: 0 Fail: error code

3.10.26. External axis absolute joint angle straight line motion- ext_lin_axis

int ext_lin_axis (HROBOT robot, int mode, double smooth_value, double* axis)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Disable smooth function 1: Bezier curve smoothing percentage 2: Bezier curve smoothing radius 3: Smooth according to the speed of the two-line segments
smooth_value	double	mode is 0: Invalid mode is 1: Smoothing percentage (1-100%) mode is 2: Smoothing radius (mm) mode is 3: Invalid
axis	Double [9]	Joint coordinate system{A1, A2, A3, A4, A5, A6, E1, E2, E3} Range of value(2147418.112,- 2147418.112) Four-axis SCARA articulated robot, A5 and A6 values are set as 0, The external axis mode must be set as synchronized (Sync) to move; the external axis will not move if disabled or in asynchronized mode.
Return value	int	Success: 0 Fail: error code

Remark: In the joint coordinate axis, the body of the articulated robot labelled as Joint1 (J1) which denoted as A1 in software, and physical robot Joint2 (J2) denoted as A2 in software, Joint3 (J3) means A3 in software and so on until Joint6.

3.10.27. External axis absolute coordinates position straight line motion- ext_lin_pos

int ext_lin_pos (HROBOT robot, int mode, double smooth_value, double* pos)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Disable smooth function 1: Bezier curve smoothing percentage 2: Bezier curve smoothing radius 3: Smooth according to the speed of the two line segments
smooth_value	double	mode is 0: Invalid mode is 1: Smoothing percentage (1-100%) mode is 2: Smoothing radius (mm) mode is 3: Invalid
pos	Double [9]	Cartesian coordinate system{X, Y, Z, A, B, C, E1, E2, E3} Range of value(2147418.112,- 2147418.112) Four-axis SCARA articulated robot, A and B values are set as 0, The external axis mode must be set as synchronized (Sync) to move; the external axis will not move if disabled or in asynchronized mode.
Return value	int	Success: 0 Fail: error code

3.10.28. External axis non-synchronized joint angle point to point motion- ext_asyptp

int ext_asyptp (HROBOT robot, int mode, double* axis)

Parameter	Data type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Disable smooth function 1: Smooth according to the speed of the two-line segments
axis	Double [3]	Joint coordinate system{ E1, E2, E3} Range of value(2147418.112,- 2147418.112) ➢ The external axis mode must be set as asynchronized (Async) to move; if set as synchronized, the [must set as asynchronized] alarm will be issued. ➢ If not moving the external axis, 0xffff can be carried over to prevent synchronized mode for issuing alarms. ➢ EX: Only enable the E2 axis double pos[3] = {0xffff, 10, 0xffff}
Return value	int	Success: 0 Fail: error code

```
double ext_axis[9] = {10, 20, 0, 0, -90, 0, 10.5, 20.5, 30.5};
ext_ptp_axis(robot, 0, ext_axis);
ext_lin_axis(robot, 0, 0, ext_axis);

double ext_pos[9]={ 217.5 ,58.5 ,601.659 ,60.8 ,35.5 ,95.7, -10.8, -20.6, -30.7};
ext_ptp_pos(robot, 0, ext_pos);
ext_lin_pos(robot, 0, 0, ext_pos);

double ext_asyaxis[3]={ -10.3, 20.4, 0xffff};
// E1, E2: Async;   E3: Sync
ext_asyptp(robot, 0, ext_asyaxis);
```

3.11. Manipulator Information Command Class

Reference function names and corresponding identities are shown in the below table:

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
get_encoder_count	Get current encoder value	O	O	O
get_current_joint	Get current joint coordinate	O	O	O
get_current_position	Get current absolute coordinate position	O	O	O
get_current_rpm	Get current shaft speed	O	O	O
get_device_born_date	Get device manufacture time	O	O	O
get_operation_time	Get controller booting time	O	O	O
get_mileage	Get motor mileage of each axis	O	O	O
get_total_mileage	Get accumulative motor mileage of each axis	O	O	O
get_utilization	Get accumulative utilization	O	O	O
get_utilization_ratio	Get utilization ratio	O	O	O
get_motor_torque	Get motor load percentage	O	O	O
get_hrss_version	Get HRSS version	O	O	O
get_robot_type	Get robot model	O	O	O
set_home_point	Setting home position	O	O	×
set_ext_home_point	Setting external home position	O	O	×
get_home_point	Get home position	O	O	O
get_ext_home_point	Get external home position	O	O	O
get_previous_pos	Get lastest position of previous power off	O	O	O
get_previous_extpos	Get lastest external position	O	O	O

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
	of previous power off			
confirm_home_point	Confirm to check home point	O	×	×
enable_joint_soft_limit	Enable joint coordinate software limit	O	O	×
enable_cart_soft_limit	Enable cartesian coordinate system software limit	O	O	×
set_joint_soft_limit	Set joint coordinate software limit	O	O	×
set_cart_soft_limit	Set cartesian coordinate system software limit	O	O	×
get_joint_soft_limit_config	Set joint coordinate software limit configuration	O	O	O
get_cart_soft_limit_config	Set cartesian coordinate system software limit configuration	O	O	O
set_payload_config	Set payload configuration	O	O	×
get_payload_config	Get payload configuration	O	O	O
set_payload_active	Set valid load	O	O	×
get_payload_active	Get valid load	O	O	O
get_home_warning_setting	Get point alarm configuration	O	O	O
set_home_warning_setting	Set point alarm configuration	O	O	×
get_ext_driver_limit	Get external axis driver limit	O	O	O
set_ext_driver_limit	Set external axis driver limit	O	O	×
get_ext_encoder	Get external axis encoder value	O	O	O
get_robot_dh	Get articulated robot DH value	O	O	×
get_gear_ratio	Get gear ratio	O	O	×

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
get_robot_data	Get robot information data	O	O	×

The following section will describe the various function names, instruction formats, parameter settings and example descriptions.

3.11.1. Get current encoder value- get_encoder_count

int get_encoder_count(HROBOT robot ,INT32* value)

Parameter	Data type	Description
robot	HROBOT	Device ID
value	Double [6]	Encoder value of each axis
Return value	int	Success: 0 Fail: error code

3.11.2. Get current joint coordinates- get_current_joint

int get_current_joint(HROBOT robot ,double* coor)

Parameter	Data type	Description
robot	HROBOT	Device ID
coor	Double [6]	Array used to save the joint coordinates, unit: degree.
Return value	int	Success: 0 Fail: error code

```
double[6] pos;
get_current_joint(robot,pos);    //get current point in joint coordinate
```

3.11.3. Get current absolute coordinates- get_current_position

int get_current_position(HROBOT robot ,double* coor)

Parameter	Data type	Description
robot	HROBOT	Device ID
coor	Double [6]	Array used to save the absolute coordinates, unit: millimeter (mm).
Return value	int	Success: 0 Fail: error code

```
double[6] pos;
get_current_position(robot,pos);   //get current point in Cartesian coordinate
```

3.11.4. Get current rotation speed- get_current_rpm

int get_current_rpm(HROBOT robot ,double* coor)

Parameter	Data type	Description
robot	HROBOT	Device ID
coor	Double [6]	Rotation speed of each joint motor
Return value	int	Success: 0 Fail: error code

```
double[6] rpm;
rpm=get_current_rpm(robot);
```

3.11.5. Get manufactured date of device- get_device_born_date

int get_device_born_date (HROBOT robot ,int* YMD)

Parameter	Data type	Description
robot	HROBOT	Device ID
YMD	int[3]	YMD[0]: Manufactured year YMD[1]: Manufactured month YMD[2]: Manufactured day
Return value	int	Success: 0 Fail: error code

3.11.6. Get controller's startup time- get_operation_time

int get_operation_time(HROBOT robot ,int* YMDHm)

Parameter	Data type	Description
robot	HROBOT	Device ID
YMDHm	int[5]	YMDHm[0]: Startup time year YMDHm[1]: Startup time month YMDHm[2]: Startup time day YMDHm[3]: Startup time hour YMDHm[4]: Startup time minute
Return value	int	Success: 0 Fail: error code

3.11.7. Get mileage of each axis motor- get_mileage

int get_mileage(HROBOT robot ,double* mil)

Parameter	Data type	Description
robot	HROBOT	Device ID
mil	double[6]	mil[0]: Rotation count of first axis motor mil[1]: Rotation count of second axis motor mil[2]: Rotation count of third axis motor mil[3]: Rotation count of fourth axis motor mil[4]: Rotation count of fifth axis motor mil[5]: Rotation count of sixth axis motor
Return value	int	Success: 0 Fail: error code

- **Motor rotation count can be reset through HRSS**

3.11.8. Get accumulated mileage of each axis motor- get_total_mileage

int get_total_mileage(HROBOT robot ,double* mil)

Parameter	Data type	Description
robot	HROBOT	Device ID
mil	Double [6]	mil[0]: Accumulated rotation count of first axis motor mil[1]: Accumulated rotation count of second axis motor mil[2]: Accumulated rotation count of third axis motor mil[3]: Accumulated rotation count of fourth axis motor mil[4]: Accumulated rotation count of fifth axis motor mil[5]: Accumulated rotation count of sixth axis motor
Return value	int	Success: 0 Fail: error code

- **Motor accumulated rotation count cannot be reset through HRSS**

3.11.9. Get accumulated utilization rate- get_utilization

int get_utilization (HROBOT robot ,int* ult)

Parameter	Data type	Description
robot	HROBOT	Device ID
ult	Double [6]	ult [0]: Utilization year ult [1]: Utilization month ult [2]: Utilization day ult [3]: Utilization hour ult [4]: Utilization minute ult [5]: Utilization second
Return value	Int	Success: 0 Fail: error code

3.11.10. Get utilization percentage- get_utilization_ratio

int get_utilization_ratio(HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	Int	Success: Utilization percentage Fail: error code

3.11.11. Get motor load percentage- get_motor_torque

int get_motor_torque(HROBOT robot ,double* cur)

Parameter	Data type	Description
robot	HROBOT	Device ID
cur	Double [6]	cur [0]: First axis torque percentage cur [1]: Second axis torque percentage cur [2]: Third axis torque percentage cur [3]: Fourth axis torque percentage cur [4]: Fifth axis torque percentage cur [5]: Sixth axis torque percentage
Return value	Int	Success: 0 Fail: error code

3.11.12. Get HRSS version number- get_hrss_version

int get_hrss_version (HROBOT robot ,char* ver)

Parameter	Data type	Description
robot	HROBOT	Device ID
ver	Double [6]	HRSS version number
Return value	Int	Success: 0 Fail: error code

```
char* HrssV = new char[256];
get_hrss_version(s, HrssV);
std::cout << "HRSS version:" << HrssV << std::endl;
delete[]HrssV;
```

3.11.13. Get HRSS connection version- get_hrss_sdkver

void get_hrss_sdkver (HROBOT robot, int& large_ver, int& small_ver, int& revision)

Parameter	Data type	Description
robot	int	Connection number
large_ver	int	Large version connection number
small_ver	int	Small version connection number
revision	int	Version stamp
Remark	➢ Starting from SDK3.0.1 and HRSS 3.3.16, the SDK connection will be paired with the robot connection based on this version number. The following situations will occur when the robot and SDK versions do not match. ➢ Connection cannot be made when the software version number have a big different; the SDK must be updated in order to be used. ➢ Connection can be made normally if the software version number have a small different; however, the new instructions might be missed and can not be used on the old versions of SDK. ➢ Version stamp; no special function.	

```
int client_L=0, server_L=0;
int client_s, server_s=0;
int rev;
get_hrsdk_sdkver (client_L, client_s, rev);
get_hrss_sdkver (robot, server_L, server_s, rev);

if(client_L != server_L){
    printf("Large version does not match and can't be connected.");
}else if(client_s != server_s){
    printf("small version does not match.");
}
```

3.11.14. Get robot model number- get_robot_type

int get_robot_type (HROBOT robot ,char* robType)

Parameter	Data type	Description
robot	HROBOT	Device ID
Robot Type	char*	Robot model number
Return value	Int	Success: 0 Fail: error code

```
char* v = new char[256];
get_robot_type(device_id, v);
std::cout << "ROBOT TYPE:\t" << v << std::endl;
delete[]v;
```

3.11.15. Set reset position- set_home_point

int set_home_point (HROBOT robot, double* joint)

Parameter	Data type	Description
robot	HROBOT	Device ID
joint	double[6]	Coordinates A1~A6 of each joint
Return value	int	Success: 0 Fail: error code

3.11.16. Set external axis reset position- set_ext_home_point

int set_ext_home_point (HROBOT robot, double* ext_pos)

Parameter	Data type	Description
robot	HROBOT	Device ID
ext_pos	Double [3]	Coordinates E1~E3 of each external axis
Return value	int	Success: 0 Fail: error code

3.11.17. Get reset position- get_home_point

int get_home_point (HROBOT robot, double* joint)

Parameter	Data type	Description
robot	HROBOT	Device ID
joint	Double [6]	Coordinates A1~A6 of each joint
Return value	int	Success: 0 Fail: error code

3.11.18. Get external axis reset position- get_ext_home_point

int get_ext_home_point (HROBOT robot, double* ext_pos)

Parameter	Data type	Description
robot	HROBOT	Device ID
ext_pos	Double [3]	Coordinates E1~E3 of each external axis
Return value	int	Success: 0 Fail: error code

3.11.19. Get previous shutdown position- get_previous_pos

int get_previous_pos (HROBOT robot, double* joint)

Parameter	Data type	Description
robot	HROBOT	Device ID
joint	Double [6]	Coordinates A1~A6 of each joint
Return value	int	Success: 0 Fail: error code

3.11.20. Get previous external axis shutdown position- get_previous_extpos

int get_previous_extpos (HROBOT robot, double* ext_pos)

Parameter	Data type	Description
robot	HROBOT	Device ID
ext_pos	Double [3]	Coordinates E1~E3 of each external axis
Return value	int	Success: 0 Fail: error code

```
double joint[6] = {20, 20, 0, 0, 0, 0};
double recv[6] = {0, 0, 0, 0, 0, 0};
double ext_pos = {10, 20 ,30};
double recv_ext = {0,0,0};

set_home_point (robot, joint);
```

```
set_ext_home_point (robot, ext_pos);
get_home_point(robot, recv);
get_ext_home_point(robot, ext_pos);
get_previous_pos(robot, recv);
get_previous_extpos(robot, recv_ext);
```

3.11.21. Point inspection- confirm_home_point

int confirm_home_point(HROBOT s)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: 0 Inspection failed: 2004 Fail: error code
Remark	When the positions before startup and shutdown are different (collided with the robot), Alarm 01-04-30 “Start pos delineation error” appears, and point inspection is required. 1. Move to Home point 2. Use this API to perform point inspection	

3.11.22. Enable joint coordinates software limit- enable_joint_soft_limit

int enable_joint_soft_limit (HROBOT robot, bool enable)

Parameter	Data type	Description
robot	HROBOT	Device ID
enable	bool	0: OFF 1: ON
Return value	int	Success: 0 Fail: error code

3.11.23. Enable cartesian coordinates software limit-enable_cart_soft_limit

int enable_cart_soft_limit (HROBOT robot, bool enable)

Parameter	Data type	Description
robot	HROBOT	Device ID
enable	bool	0: OFF 1: ON
Return value	int	Success: 0 Fail: error code

3.11.24. Set joint coordinates upper and lower limits-set_joint_soft_limit

int set_joint_soft_limit (HROBOT robot, double* low_limit, double* high_limit)

Parameter	Data type	Description
robot	HROBOT	Device ID
low_limit	double[6]	Lower limit of each joint coordinate
high_limit	double[6]	Upper limit of each joint coordinate
Return value	int	Success: 0 Fail: error code
Remark		➢ Upper and lower limits for the A1~A6 axes can be set for each axis limit of the six-axis robot; upper and lower limits for the A1~A4 axes can be set for each axis limit of the SCARA robot. ➢ When the number of movement commands sent to the robot exceeded the limit set, or when the robot exceeded the position of the limit set during the movement process, real-time alarms will be issued and the motion will stop.

3.11.25. Set cartesian coordinates upper and lower limits- set_cart_soft_limit

int set_cart_soft_limit (HROBOT robot, double* low_limit, double* high_limit)

Parameter	Data type	Description
robot	HROBOT	Device ID
low_limit	double[3]	Lower limit of each cartesian coordinate
high_limit	double[3]	Upper limit of each cartesian coordinate
Return value	int	Success: 0 Fail: error code
Remark	➢ The cartesian coordinate limit can set the upper and lower limits of X, Y and Z. ➢ When the number of movement commands sent to the robot exceeded the limit set, or when the robot exceeded the position of the limit set during the movement process, real-time alarms will be issued and the motion will stop. ➢ This limit range is set according to the TCP of the robot while the base coordinate is 0 (Base 0).	

3.11.26. Get joint coordinates software limit configuration- get_joint_soft_limit_config

int get_joint_soft_limit_config (HROBOT robot, bool& enable, double* low_limit, double* high_limit)

Parameter	Data type	Description
robot	HROBOT	Device ID
enable	bool&	0: OFF 1: ON
low_limit	double[6]	Lower limit of each joint coordinate
high_limit	double[6]	Upper limit of each joint coordinate
Return value	int	Success: 0 Fail: error code

3.11.27. Get cartesian coordinates software limit configuration- get_cart_soft_limit_config

int get_cart_soft_limit_config (HROBOT robot, bool& enable, double* low_limit, double* high_limit)

Parameter	Data type	Description
robot	HROBOT	Device ID
enable	bool&	0: OFF 1: ON
low_limit	Double [3]	Lower limit of each cartesian coordinate
high_limit	Double [3]	Upper limit of each cartesian coordinate
Return value	int	Success: 0 Fail: error code

```
double joint_low_limit[6] = { -20, -20, -35, -20, 0, 0 };
double joint_high_limit[6] = { 20, 20, 0, 0, 0, 0 };
double cart_low_limit[6] = { -100, 300, -100, 0, 0, 0 };
double cart_high_limit[6] = { 100, 450, -25, 0, 0, 0 };
bool re_bool = false;

set_joint_soft_limit(device_id, joint_low_limit, joint_high_limit);
set_cart_soft_limit(device_id, cart_low_limit, cart_high_limit);
enable_joint_soft_limit(device_id, true);
enable_cart_soft_limit(device_id, true);
get_joint_soft_limit_config (device_id, re_bool, joint_low_limit, joint_high_limit);
get_cart_soft_limit_config (device_id, re_bool, cart_low_limit, cart_high_limit);
```

3.11.28. Set load configuration- set_payload_config

int set_payload_config(HROBOT s, int index, double* value, char* comment)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Number range: [0-19]
value	Double [7]	➢ Array with a length of 7 0: Quality (kg) cannot be 0 Center (mm) 1: Xc 2: Yc 3: Zc Rotational inertia (kg*mm^2) 4: Ixx 5: Iyy 6: Izz
comment	char*	Set load comment
Return value	int	Success: 0 Exceeded allowable value of RS405: 2 Fail: error code

3.11.29. Get load configuration- get_payload_config

int get_payload_config (HROBOT s, int index, double* value, char* comment)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Number range: [0-19]
value	Double [7]	➢ Array with a length of 7 0: Quality (kg) 1: Xc 2: Yc 3: Zc 4: Ixx 5: Iyy 6: Izz
comment	char*	Get load comment
Return value	int	Success: 0 Fail: error code

3.11.30. Set valid load- set_payload_active

int set_payload_active(HROBOT s, int index)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	Number range: [0-19]
Return value	int	Success: 0 Fail: error code
Remark	Set the load of this number as valid; there can only be one enabled load	

3.11.31. Get valid load- get_payload_active

int get_payload_active(HROBOT s, int& index)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int&	Number range: [0-19]
Return value	int	Success: 0 Fail: error code

```
int value = -1;
int index = 5;
int& re_index = -1;
char* comment = “payload comment”;
double value[7] = {1,2,3,4,5,6,7 };

set_payload_config(robot, index, value, comment);
get_payload_config(robot, index, value, comment);
set_payload_active(robot, index);
get_payload_active(robot, re_index);
```

3.11.32. Get point alarm configuration- get_home_warning_setting

int get_home_warning_setting (HROBOT robot, double* allow_error, double* near_home)

Parameter	Data type	Description
robot	HROBOT	Device ID
allow_error	Double [9]	Allows only issuing the [Start pos delineation error], error code 01-04-30 alarm when the difference between the angles before shutdown and after startup is greater than the setting value. This array includes 9 spaces A1~A6 與 E1~E3 allow_error[0]: A1 allow_error[1]: A2 allow_error[2]: A3 allow_error[3]: A4 allow_error[4]: A5 allow_error[5]: A6 allow_error[6]: E1 allow_error[7]: E2 allow_error[8]: E3
near_home	Double [9]	When the point alarm [Start pos delineation error] is issued, it needs to jog back to the origin; if the difference between the position of the articulated robot and the home point is within the angle value set here, Confirm Home Point can be used to clear the alarm. This array includes 9 spaces A1~A6 and E1~E3
Return value	int	Success: 0 Fail: error code

Remark: In the joint coordinate axis, the body of the articulated robot labelled as Joint1 (J1) which denoted as A1 in software, and physical robot Joint2 (J2) denoted as A2 in software, Joint3 (J3) means A3 in software and so on until Joint6.

3.11.33. Set point alarm configuration- set_home_warning_setting

int set_home_warning_setting (HROBOT robot, double* allow_error, double* near_home)

Parameter	Data type	Description
robot	HROBOT	Device ID
allow_error	Double [9]	Allows only issuing the [Start pos delineation error], error code 01-04-30 alarm when the difference between the angles before shutdown and after startup is greater than the setting value. This array includes 9 spaces A1~A6 and E1~E3 allow_error[0]: A1 allow_error[1]: A2 allow_error[2]: A3 allow_error[3]: A4 allow_error[4]: A5 allow_error[5]: A6 allow_error[6]: E1 allow_error[7]: E2 allow_error[8]: E3
near_home	Double [9]	When the point alarm [Start pos delineation error] is issued, it needs to jog back to the origin; if the difference between the position of the articulated robot and the home point is within the angle value set here, Confirm Home Point can be used to clear the alarm. This array includes 9 spaces A1~A6 and E1~E3
Return value	int	Success: 0 Fail: error code

```
double allow_error [9] = {1, 1, 1, 1, 1, 1, 3, 3, 3};
double near_home [9] = {1, 1, 1, 1, 1, 1, 3, 3, 3};

set_home_warning_setting (robot, allow_error, near_home);
get_home_warning_setting (robot, allow_error, near_home);
```

3.11.34. Get external axis driver limit- get_ext_driver_limit

int get_ext_driver_limit (HROBOT robot, int index, bool& enable, bool& inverse, int& negative_num, int& positive_num, bool& N_light, bool& P_light)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	E1 - E3 number range: [0-2]
enable	bool&	0: Disable function 1: Enable function
inverse	bool&	This option will cause the positive and negative light signal to light up reversely, 0: Disable function, lights on when the limit is triggered 1: Enable function, lights on when not triggered, goes off when the limit is triggered
negative_num	int&	Get pin of the negative coding value monitoring driver Range [0-31]
positive_num	int&	Get pin of the positive coding value monitoring driver Range [0-31]
N_light	bool&	Lights on when negative driver limit is triggered; if [inverse] is enabled, the display method is reversed. 0: Light goes off 1: Light turns on
P_light	bool&	Lights on when the positive driver limit is triggered; if [inverse] is enabled, the display method is reversed. 0: Light goes off 1: Light turns on
Return value	int	Success: 0 Fail: error code

3.11.35. Set external axis driver limit- set_ext_driver_limit

int set_ext_driver_limit (HROBOT robot, int index, bool enable, bool inverse, int negative_num, int positive_num)

Parameter	Data type	Description
robot	HROBOT	Device ID
index	int	E1 - E3 number range: [0-2]
enable	bool	0: Disable function 1: Enable function
inverse	bool	This option will cause the positive and negative light signal to light up reversely, 0: Disable function, lights on when the limit is triggered 1: Enable function, lights on when not triggered, goes off when the limit is triggered
negative_num	int	Select pin of the negative coding value monitoring driver Range [0-31]
positive_num	int	Select pin of the positive coding value monitoring driver Range [0-31]
Return value	int	Success: 0 Fail: error code

3.11.36. Get external axis encoder value-get_ext_encoder

int get_ext_encoder (HROBOT robot, int32_t* EncCount)

Parameter	Data type	Description
robot	HROBOT	Device ID
EncCount	int32_t[3]	This array size is 3 spaces; get the encoder value of E1 to E3
Return value	int	Success: 0 Fail: error code

3.11.37. Get articulated robot DH value- get_robot_dh

Description: Use this instruction to get the D-H table information of the articulated robot.

int get_robot_dh(HROBOT s, int type, double dh_value[][6])

Parameter	Data type	Description
robot	HROBOT	Device ID
type	int	0: RA (joint articulated robot, 6-axis) 1: RS (SCARA articulated robot, SCARA) 2: RD (Parallel articulated robot, Delta)
dh_value	double[][6]	RA: Space needed [6][6] RS: Space needed [4][6] RD: Space needed [1][6]
Return value	int	Success: 0 Fail: error code

C++ example file:

```
double ra_dh[6][6] = {0};
double rs_dh[4][6] = {0};
double rd_dh[1][6] = {0};

get_robot_dh(robot, 0, ra_dh);
get_robot_dh(robot, 1, rs_dh);
get_robot_dh(robot, 2, rd_dh);
```

3.11.38.Get gear ratio- get_gear_ratio

Description: Use this instruction to get the gear ratio information of the articulated robot.

int get_gear_ratio(HROBOT s, double gear_ratio[6])

Parameter	Data type	Description
robot	HROBOT	Device ID
gear_ratio	Double [6]	Gear ratio (%)
Return value	int	Success: 0 Fail: error code

Get articulated robot information

C++ example file:

```
double gear_ratio [6] = {0};

get_gear_ratio(robot, gear_ratio);
```

3.11.39. Get articulated robot information- get_robot_data

Description: This instruction can be used to get information of the articulated robot, including system information, port information and the setting parameter information of each axis.

int get_robot_data (HROBOT robot, int* sys_info, int* port_info, double* axis_info)

Parameter	Data type	Description
robot	HROBOT	Device ID
sys_info	int[3]	System information. [0]：Max Speed (mm/s) [1]：Mode [2]：Axis Count
port_info	int[6]	Port information. [0]：Robot IO [1]：External [2]：Reserved [3]：Driver [4]：Touch Pancel [5]：Teach Pendant
axis_info	double[axis_count][7]	Information of each axis. axis_count: Number of articulated robot axis. [0]：Pos To Encoder Direction [1]：PPR [2]：RPM [3]：Pitch [4]：Gear Ratio [5]: High Limit (linear is mm; rotation is degree) [6]: Low Limit (linear is mm; rotation is degree)
Return value	int	Success: 0: Fail: error code

C++ example file:

```
int sys_info[3] = {0};
int port_info[6] = {0};
double axis_info[6][7] = {0};

get_robot_data(robot, sys_info, port_info, axis_info);
```

3.12. Communication Setting Command Class

Reference function names and corresponding identities are shown in the below table:

Function Name	Description	Operator		Observer
		Manual mode	Auto mode	
set_network_show_msg	Set display message status	O	O	×
get_network_show_msg	Get display message status	O	O	O
network_connect	Network connection	O	O	×
network_disconnect	Network disconnected	O	O	×
network_send_msg	Send network message	O	O	×
network_recieve_msg	Receive network message	O	O	O
set_network_config	Set network configuration	O	O	×
get_network_config	Get network configuration	O	O	O
network_change_ip	Change the network IP address	O	O	×
network_get_state	Get connection status	O	O	O

The following section will describe the various function names, instruction formats, parameter settings and example descriptions.

3.12.1. Set display message status- set_network_show_msg

int set_network_show_msg (HROBOT robot, int enable)

Parameter	Data type	Description
robot	HROBOT	Device ID
enable	int	0: OFF 1: ON
Return value	int	Success: 0 Fail: error code
Remark	Messages received, but cannot be displayed if this function is not set as ON.	

3.12.2. Get display message status- get_network_show_msg

int get_network_show_msg (HROBOT robot, int enable)

Parameter	Data type	Description
robot	HROBOT	Device ID
enable	int&	0: OFF 1: ON
Return value	int	Success: 0 Fail: error code

3.12.3. Network connection- network_connect

int network_connect (HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: 0 Fail: error code

3.12.4. Network disconnected- network_disconnect

int network_disconnect (HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Success: 0 Fail: error code

3.12.5. Send network message- network_send_msg

int network_send_msg (HROBOT robot, char* msg)

Parameter	Data type	Description
robot	HROBOT	Device ID
enable	char*	“%%%” must be added to the end of the string of the message to send EX: str = “{test msg%%%}”
Return value	int	Success: 0 Fail: error code

3.12.6. Receive network message- network_recieve_msg

int network_recieve_msg (HROBOT robot, char* msg)

Parameter	Data type	Description
robot	HROBOT	Device ID
enable	char*	Message to receive
Return value	int	Success: 0 Fail: error code
Remark		➢ This function is invalid; Callback notification must be used to receive network messages, and step no 7 from table 3.12.7 must be set on the get_robot_info page to receive callback. ➢ get_current_position must be called to enable the Callback Notification function in order to use Notification. ➢ The network_get_state function must be called continuously in order to receive messages. ➢ There will be “&&&” at the end ➢ Please refer to Sample Code 12.Network for detailed examples.

3.12.7. Set network configuration- set_network_config

int set_network_config (HROBOT robot, int connect_type, char* ip_addr, int port, int bracket_type, int separator_type, bool is_format)

Parameter	Data type	Description
robot	HROBOT	Device ID
connect_type	int	Set the connection status to server or client 0: Server 1: Client
ip_addr	char*	Target IP address of the connection
port	int	Connection gateway
bracket_type	int	➢ Use brackets to surround the data sent. ➢ Brackets same as HRSS must be used to surround the data. Otherwise, the data received cannot be displayed. 0: {} 1: [] 2: () 3: <> 4: “ ? ” in front
separator_type	int	➢ Select a symbol to separate data; when the HRB instruction CREAD of the

		articulated robot program is used to receive network messages, this symbol will be used to split the data content. 0: , 1: _
is_format	bool	0: OFF Data sent is displayed using ASCII graphics (alphanumeric characters) 1: ON Data sent is displayed using ASCII hexadecimal
Return value	int	Success: 0 Fail: error code

3.12.8. Get network configuration- get_network_config

int get_network_config (HROBOT robot, int& connect_type, char* ip_addr, int& port, int& bracket_type, int& separator_type, bool& is_format)

Parameter	Data type	Description
robot	HROBOT	Device ID
connect_type	int&	Get the status of the connection set, server or client 0: Server 1: Client
ip_addr	char*	Get the target IP address of the connection
port	int&	Get connection gateway
bracket_type	int&	Get selected bracket type 0: {} 1: [] 2: () 3: <> 4: ? in front
separator_type	int&	Get the delimiter 0: , 1: _
is_format	bool&	Get display format 0: OFF Data sent is displayed using ASCII graphics (alphanumeric characters) 1: ON Data sent is displayed using ASCII hexadecimal
Return value	int	Success: 0 Fail: error code

3.12.9. Change network IP address- network_change_ip

int network_change_ip (HROBOT robot, int lan_index, int ip_type, char*

ip_addr)

Parameter	Data type	Description
robot	HROBOT	Device ID
lan_index	int	Set the Nth network interface card; the quantity is the same as the number of network interface cards the user's computer has. 0: LAN1 1: LAN2
ip_type	int	0: Dynamic IP, the IP address is obtained automatically by the computer. 1: Static IP, users can set the IP address of the computer themselves.
ip_addr	char*	When static IP is used, settings will be based on this address
Return value	int	Success: 0 Fail: error code

3.12.10. Get connection status- network_get_state

int network_get_state (HROBOT robot)

Parameter	Data type	Description
robot	HROBOT	Device ID
Return value	int	Connected: 1 Disconnected: 0
Remark	➢ When the Notify mechanism is enabled, this function must be called continually for Notify to receive messages; the receiving frequency is that data will only be received when this function is called. ➢ To enable Notification, get_current_position must be executed first to enable it.	

```
(1)-------- set config
int server_type = 0; //   socket server
char* ip = "127.0.0.1";
int port = 5123;
int bracket = 0;   //   {}
int separator = 0;   //   ,
bool is_format = false;
int show_msg = -1;
```

```
set_network_config(robot _id, client_type, ip, port, bracket, separator, is_format);
network_connect(robot_id);
get_network_show_msg(robot_id, show_msg);
if (!show_msg) {
      set_network_show_msg(robot_id, true);
}
char* msg = “test msg%%%”;
network_send_msg (robot_id, msg)
network_disconnect (robot_id)

char set_ip = “192.168.0.25”;
// network_change_ip(0, 0, set_ip);

(2) --------- get config
int re_type = 0; //    socket server
char* re_ip = new char [20];
int re_port = 5123;
int re_bracket = 0;    //    {}
int re_separator = 0;    //    ,
bool re_is_format = false;
int show_msg = -1;
int connect_state = -1;
get_network_config(device_id, re_type, re_ip, re_port, re_bracket, re_separator,
re_is_format);
for(int i=0;i<3; i++){
    connect_state = network_get_state(robot_id)
    Sleep(200);
}
```

4. Program editor feedback error code description

Users can use the values returned by the variables to know the execution status of program instructions, use this to determine the problems, and determine the problem with the corresponding value. Here is a general error code description that can be used to find the direction of the problem. Please refer to the descriptions in the below table.

Summary	Description	Error Code
Failed	Command execution failed; please confirm whether the connection status is normal	-11
	Return mechanism creation failed, cannot connect to robot, version mismatch	-1~-4
Normal	Command completed normally	0000
No new password	The new password set for change password cannot be empty or special characters	0049
Password error	Wrong password entered	0050
Unauthorized	Authorization failed; please contact customer service personnel	0100
Disconnected	This connection was interrupted	0110
No file is executing	Articulated robot currently has no files executing	0150
File open failed	File open failed or file name error; please confirm whether the file path is correct	0200
File does not exist	File does not exist; please confirm whether the file path is correct	0201
File send error	Network communication abnormal	0202
File communication error	Transferred file is too big or file transfer channel abnormal	0206
Received file packet abnormal	File transfer channel abnormal	0207
File write error	Write error occurs when there is a problem with the file transfer	0208
File download timeout	Transferred file is too big or file transfer channel abnormal	0209
File name error	Please confirm whether the file name includes special characters	0210
File name too long	File name too long	0211
Command timeout	Transfer command timeout; connection abnormal or	0212

Summary	Description	Error Code
	extend command transfer intervals	
No alarms occurred	No alarms occurred, cannot clear alarm	0300
Cannot execute	Cannot execute requested command	2000
Parameter error	Command parameter error	2004
Command execution abnormal	Command execution became abnormal	2005
Cannot accept command	Does not accept the execution of commands based on the system status	2006
String too long	The length of the string entered is too long	2007
Mode error	Current status is not in EXT mode	2008
Command inconsistent	Number of send command and receive command is not consistent; please send again	2009
Controller already exists	Controller already exists	2100
Command transfer error	Problems occurred for the network communication, command receive/send error	3000
Command receive error	Problems occurred for network communication	3001
IO used	The IO set is already used	3100
Mode prohibition	The current mode does not accept the execution of commands	4000
Servo prohibition	This command cannot be executed under the non-excited status	4001
Motion register prohibition	When the number of motion register reaches 1000 entries, the requested command cannot be executed	4003
External axis not enabled	Instruction sent cannot be executed because the external axis was not enabled.	4005
File execution abnormal	RSR/PNS task setting abnormal	4010
	RSR/PNS task execution failed	4011
	Task name error or missing	4012
	There already is task executing	4013
Update file abnormal	Update file failed	4020
	Update file send failed	4021
	Update file receive failed	4022
	HRSS has insufficient hard drive space; please delete unnecessary files	4023

Summary	Description	Error Code
	HRSS update file does not exist; please confirm whether the file path is correct	4024
Function abnormal	This function is abnormal	9999

■ Example description:

Using connection (open_connection) as example:

```
int device_id = HRobot.open_connection("127.0.0.1", 1, callback);
Console.WriteLine("Return value device_id result: " + device_id);
```

If device_id returned 0, it means it is normal; if the value returned is -1~-4, it means that there are problems:

Return value result	Description
0	Command completed normally; connected successfully
-1	Command execution failed; please confirm whether the connection status is normal, connection failed
-2	Return mechanism creation failed
-3	Cannot connect to robot
-4	HRSDK and HRSS version mismatch

Index

Robot Software Development Kit (Original Instruction) User Manual

Publication Date：March 2022

1. HIWIN is a registered trademark of HIWIN Technologies Corp.. For your protection, avoid buying counterfeit products from unknown sources.
2. Actual products may differ from specifications and photos provided in this catalog. These differences may be the result of various factors including product improvements.
3. HIWIN website for patented product directory: http://www.hiwin.tw/Products/Products_patents.aspx
4. HIWIN will not sell or export products or processes restricted under the "Foreign Trade Act" or related regulations. Export of restricted products should be approved by proper authorities in accordance with relevant laws and shall not be used to manufacture or develop nuclear, biochemical, missiles or other weapons.

HIWIN®

Global Sales and Customer Service Site

HIWIN GmbH
OFFENBURG, GERMANY
www.hiwin.de
www.hiwin.eu
info@hiwin.de

HIWIN Schweiz GmbH
JONA, SWITZERLAND
www.hiwin.ch
info@hiwin.ch

HIWIN KOREA
SUWON · CHANGWON, KOREA
www.hiwin.kr
info@hiwin.kr

HIWIN JAPAN
KOBE · TOKYO · NAGOYA · NAGANO · TOHOKU · SHIZUOKA · HOKURIKU · HIROSHIMA · FUKUOKA · KUMAMOTO, JAPAN
www.hiwin.co.jp
info@hiwin.co.jp

HIWIN s.r.o.
BRNO, CZECH REPUBLIC
www.hiwin.cz
info@hiwin.cz

HIWIN CHINA
SUZHOU, CHINA
www.hiwin.cn
info@hiwin.cn

HIWIN USA
CHICAGO, U.S.A.
www.hiwin.us
info@hiwin.com

HIWIN FRANCE
STRASBOURG, FRANCE
www.hiwin.fr
info@hiwin.de

Mega-Fabs Motion Systems, Ltd.
HAIFA, ISRAEL
www.mega-fabs.com
info@mega-fabs.com

HIWIN Srl
BRUGHERIO, ITALY
www.hiwin.it
info@hiwin.it

HIWIN SINGAPORE
SINGAPORE
www.hiwin.sg
info@hiwin.sg

HIWIN TECHNOLOGIES CORP.
No. 7, Jingke Road,
Taichung Precision Machinery Park,
Taichung 40852, Taiwan
Tel: +886-4-23594510
Fax: +886-4-23594420
www.hiwin.tw
business@hiwin.tw (Sales)
robotservice@hiwin.tw (Customer Service)

The specifications in this catalog are subject to change without notification.

©2022 FORM C25UE601-2203 (PRINTED IN TAIWAN)