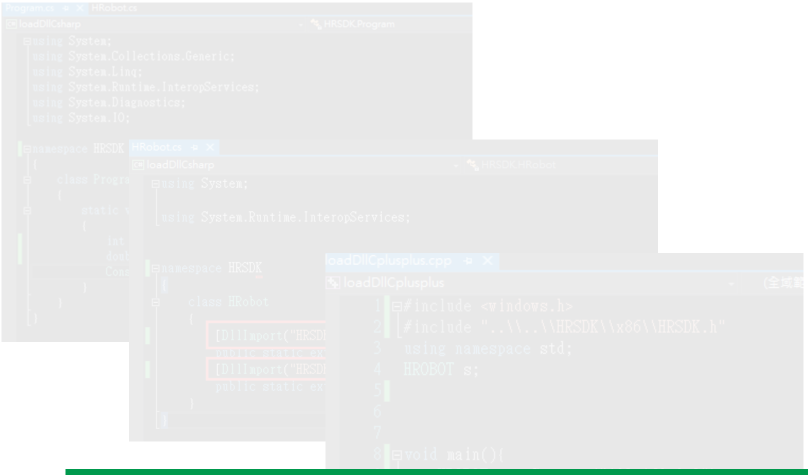
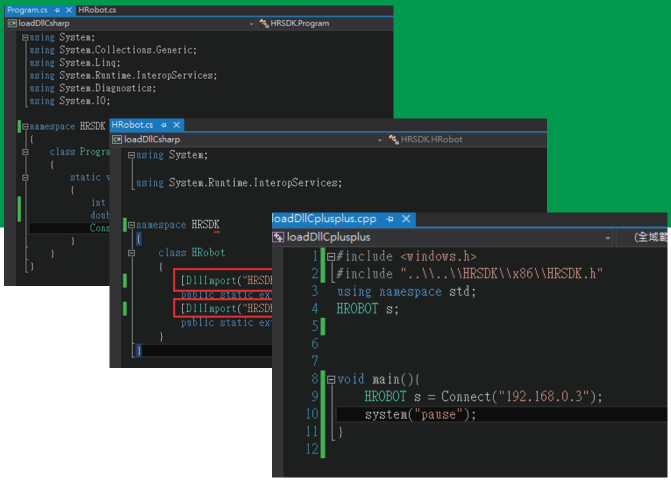


HIWIN Robot Software Development Kit



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.InteropServices;
using System.Diagnostics;
using System.IO;

namespace HRSDK
{
    class Progra
    {
        static v
        {
            int
            doub
            Cons
        }
    }
}
```

```
using System;

using System.Runtime.InteropServices;


namespace HRSDK
{
    class HRobot
    {
        [DllImport("HRSDI
        public static ex
        [DllImport("HRSDI
        public static ex
    }
}
```

```
#include <windows.h>
#include "..\\..\\HRSDK\\x86\\HRSDK.h"
using namespace std;
HROBOT s;



void main(){
    HROBOT s = Connect("192.168.0.3");
    system("pause");
}

```

User Manual

Original Instruction

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

Direct Drive Rotary Table

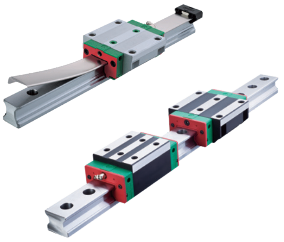
Aerospace / Medical / Automotive industry / Machine tools / Machinery industry

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

Ballscrew

Precision Ground / Rolled

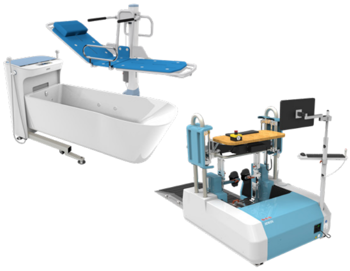
- Super S series
- Super T series
- Mini Roller
- Ecological & Economical lubrication Module E2
- Rotating Nut (R1)
- Energy-Saving & Thermal-Controlling (C1)
- Heavy Load Series (RD)
- Ball Spline



Linear Guideway

Automation / Semiconductor / Medical

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



Medical Equipment

Hospital / Rehabilitation centers / Nursing homes

- Robotic Gait Training System
- Hygiene System
- Robotic Endoscope Holder

Bearing

Machine tools / Robot

- Crossed Roller Bearings
- Ball Screw Bearings
- Linear Bearing
- Support Unit

AC Servo Motor & Drive

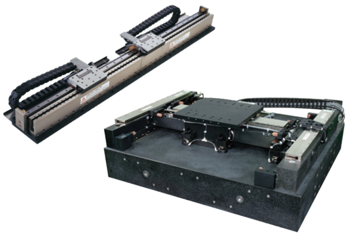
Semiconductor / Packaging machine /SMT / Food industry / LCD

- Drives-D1, D1-N, D2T
- Motors-50W~2000W

Driven Tool Holders

All kinds of turret

- VDI Systems
  Radial Series, Axial Series, MT
- BMT Systems
  DS, NM, GW, FO, MT, OM, MS



Linear Motor

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

Torque Motor (Direct Drive Motor)

Inspection / Testing equipment / Machine tools / Robot

- Rotary Tables-TMS,TMY,TMN
- TMRW Series
- TMRI Series

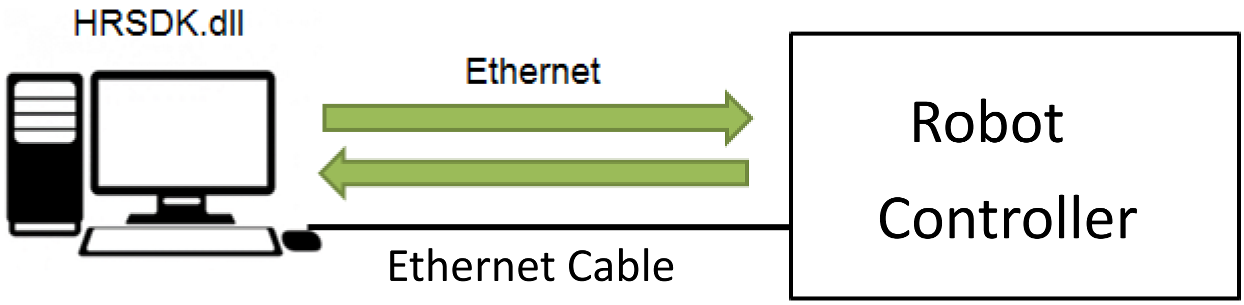
Content

Version Update

Edition	Date	Applicable Software	Applicable Range	Remark
1.0.0	2017/02/06	HRSS V2.1.23 above HRSS V3.1.6 above	All Articulated Series Robot All Delta Series Robot	Preliminary Issue
2.0.0	2017/07/13	HRSS V3.2.0 HRSS V3.2.1	All Articulated Series Robot All Delta Series Robot	Add Control Function
2.1.1	2017/09/11	HRSS V3.2.2	All Articulated Series Robot All Delta Series Robot	Add Certification
2.1.2	2018/01/05	HRSS V3.2.5	All Articulated Series Robot All Delta Series Robot	Add Product Description
2.1.4	2018/02/14	HRSS V3.2.5	All Articulated Series Robot All Delta Series Robot	Modified Example Code
2.1.5	2018/03/09	HRSS V3.2.8	All Articulated Series Robot All Delta Series Robot	Add Comment
2.1.6	2018/07/18	HRSS V3.2.11	All Articulated Series Robot All Delta Series Robot	Add Chp.1.3,1.4 & 2.6 Modified image of chapter 2.1, 2.2, 2.3
2.1.7	2018/10/24	HRSS V3.2.13	All Articulated Series Robot All Delta Series Robot	Add new command

1 Product Description

1.1 Function



User can develop the client program by different languages such as C++, C#, VB to control the robot remotely.

1.2 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 Connect to Controller

■ Description

Set IP address for robot controller.

Robot controller consist of two internet port, they are：Port 1 and Port 2.

User can choose to change IP address for either Port 1 or Port 2, using DHCP mode (automatic obtain IP address) or Static mode (specify specific IP address).

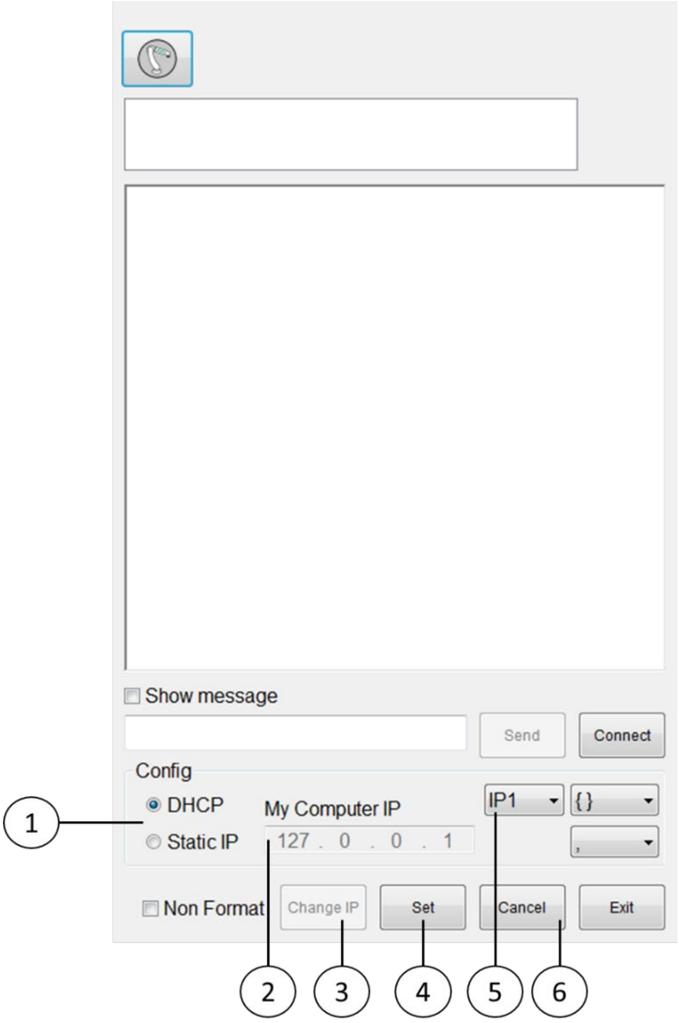


Figure 1 Change IP interface

No.	Description
1	DHCP / Static IP mode selection
2	Static IP, specific IP address
3	Enter Change IP interface
4	Confirm setting
5	Select to change Port 1/ Port2 IP address
6	Cancel setting

■ Operation Steps

Main menu >Start-up >Network Config>Change IP

- DHCP

1. Click [DHCP] option.
2. Press [Set] button.
3. Wait for the bar to finish loading, setting is completed.

- Static IP

1. Click [Static IP] option.
2. In [My Computer IP] column enter required IP address.
3. Press [Set] botton.
4. Wait for the bar to finish loading, setting is completed.

If setting failed message appeared, please check the internet connection to see if it is connected properly or there is a problem in IP setting.

1.4 Connect to Offline HRSS

1. Download offline HRSS from Hiwin website.

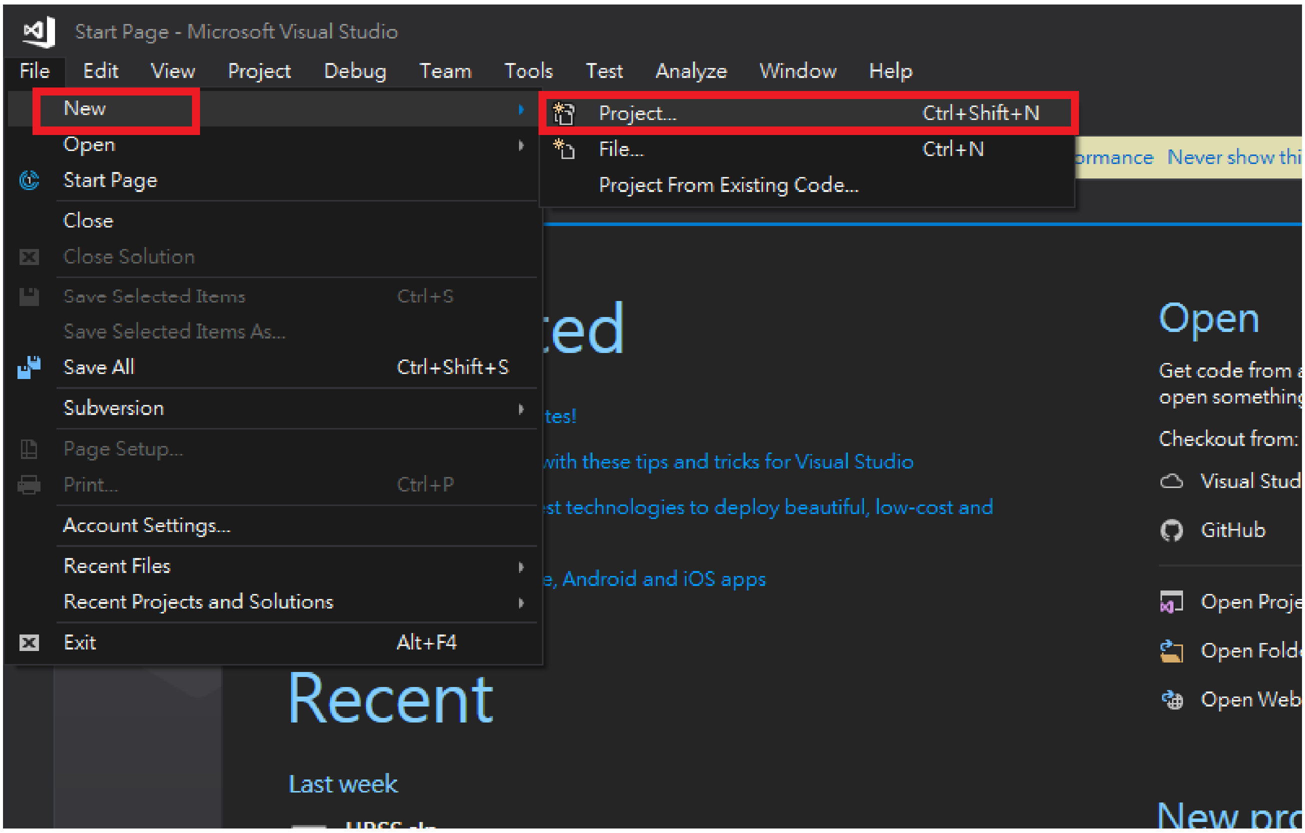
2. Launch offline HRSS on the same computer of client programs.

3. HRSS offline IP address: 127.0.0.1

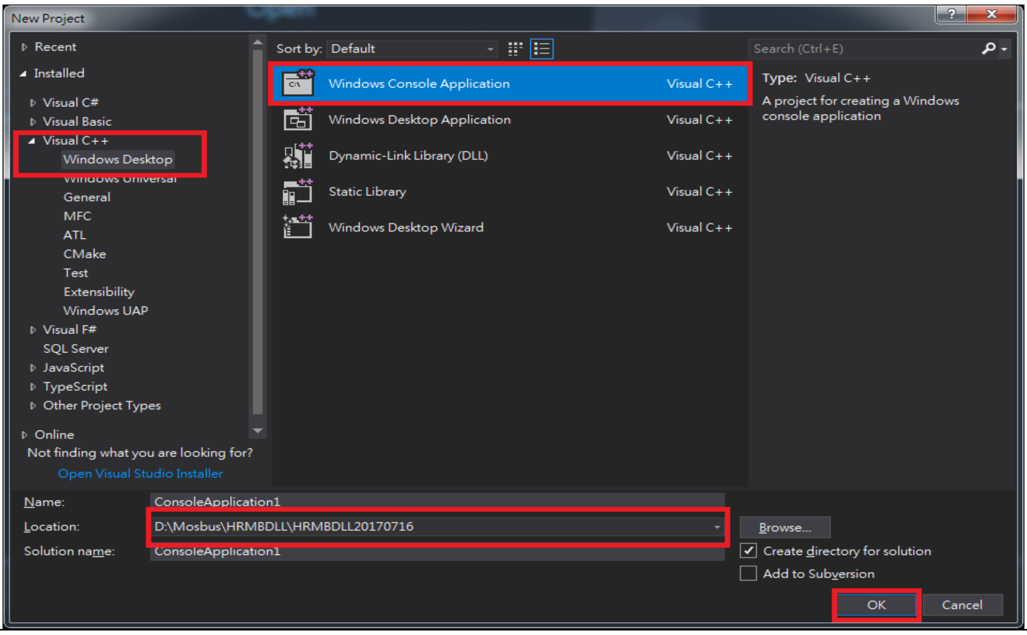
2 Instruction

2.1 C++

STEP 1 : Add new project



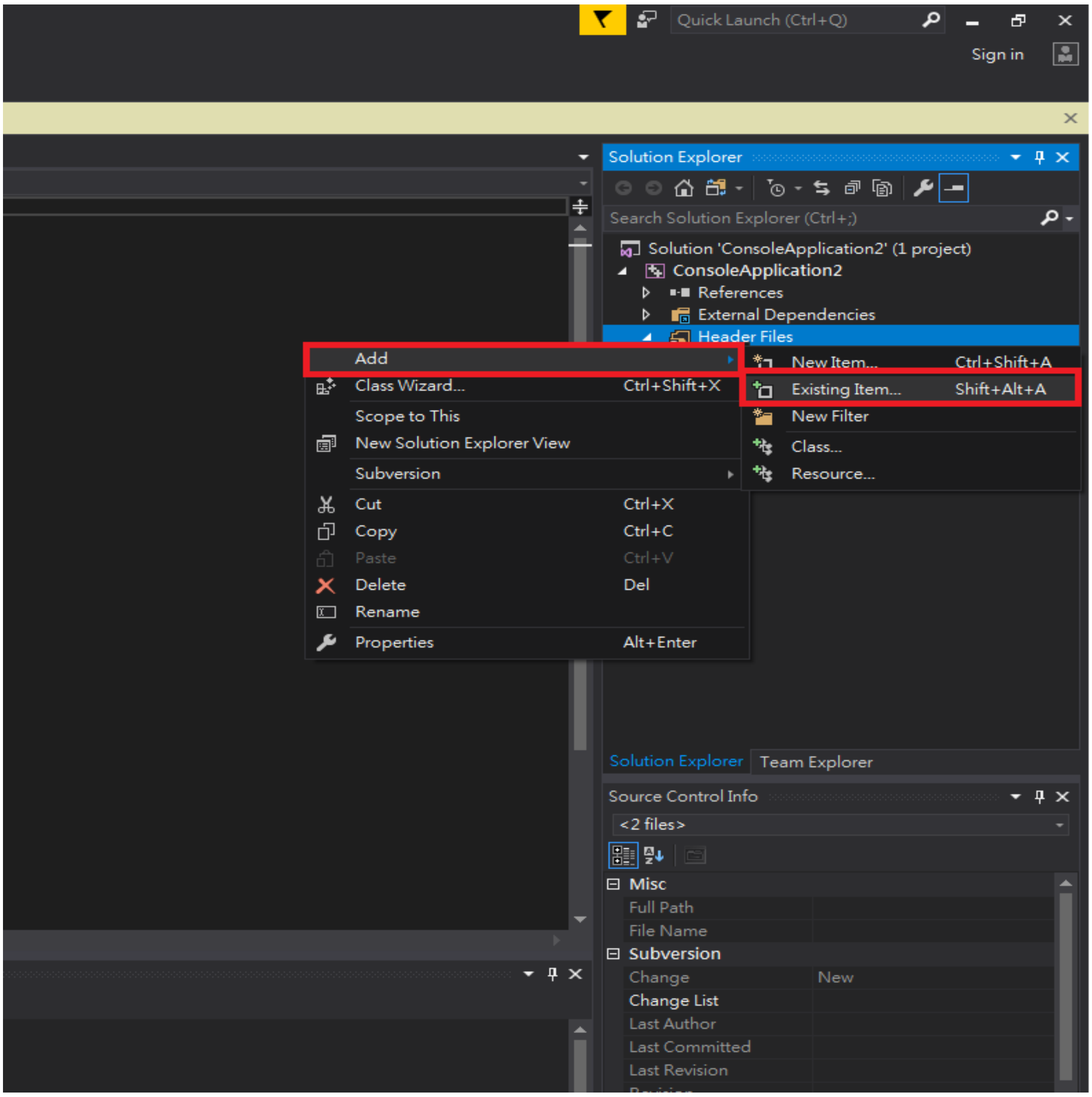
STEP 2 : Select Visual C++ Windows Console Application and set location

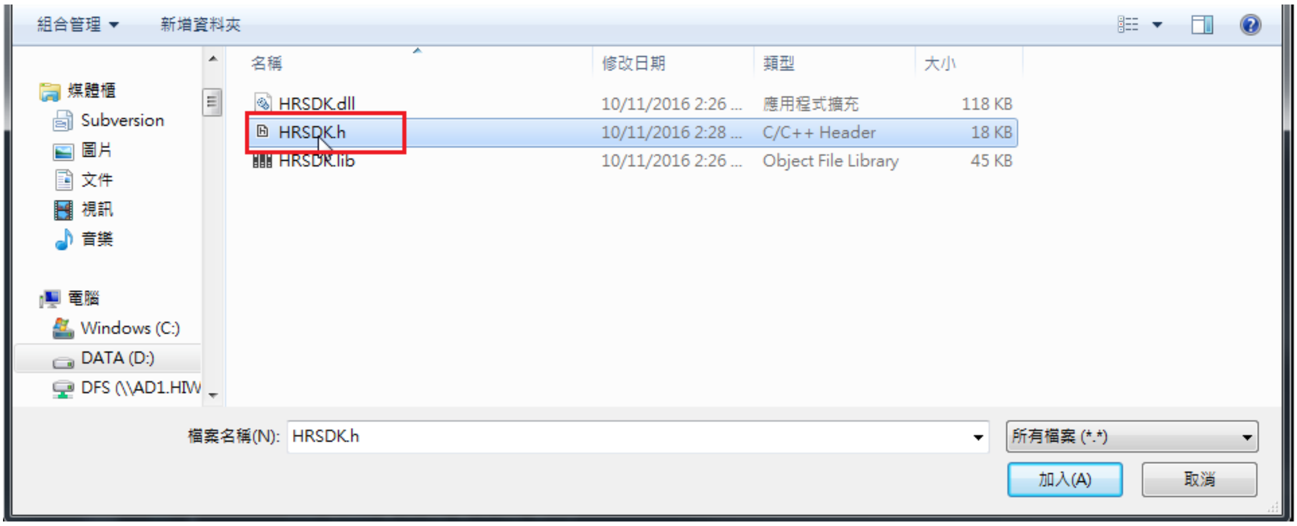


Type: Visual C++
A project for creating a Windows console application

Name: ConsoleApplication1
Location: D:\Mosbus\HRMBDLL\HRMBDLL20170716
Solution name: ConsoleApplication1
☑ Create directory for solution
☐ Add to Subversion

STEP 3 : Add HRSDK.h into header file

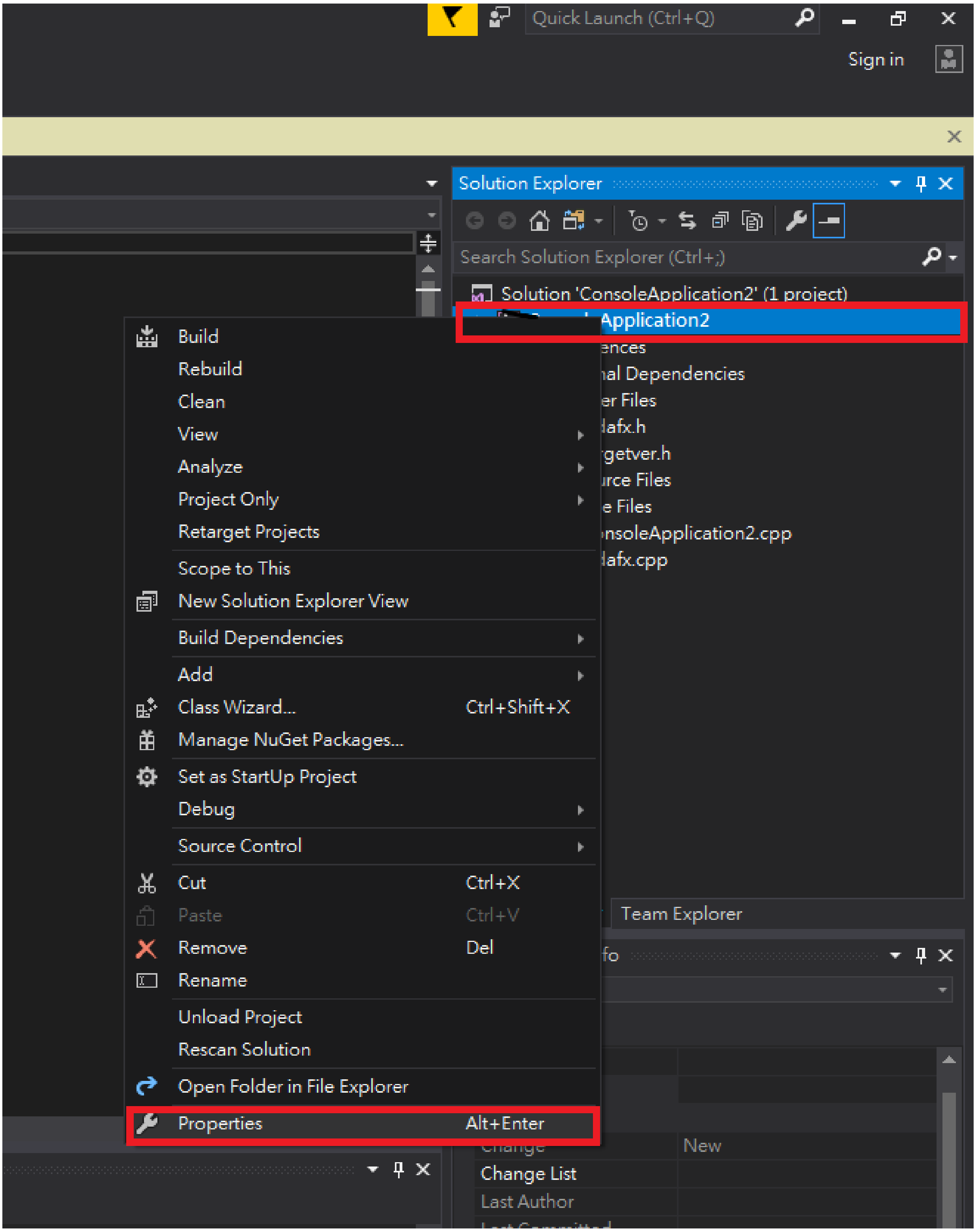




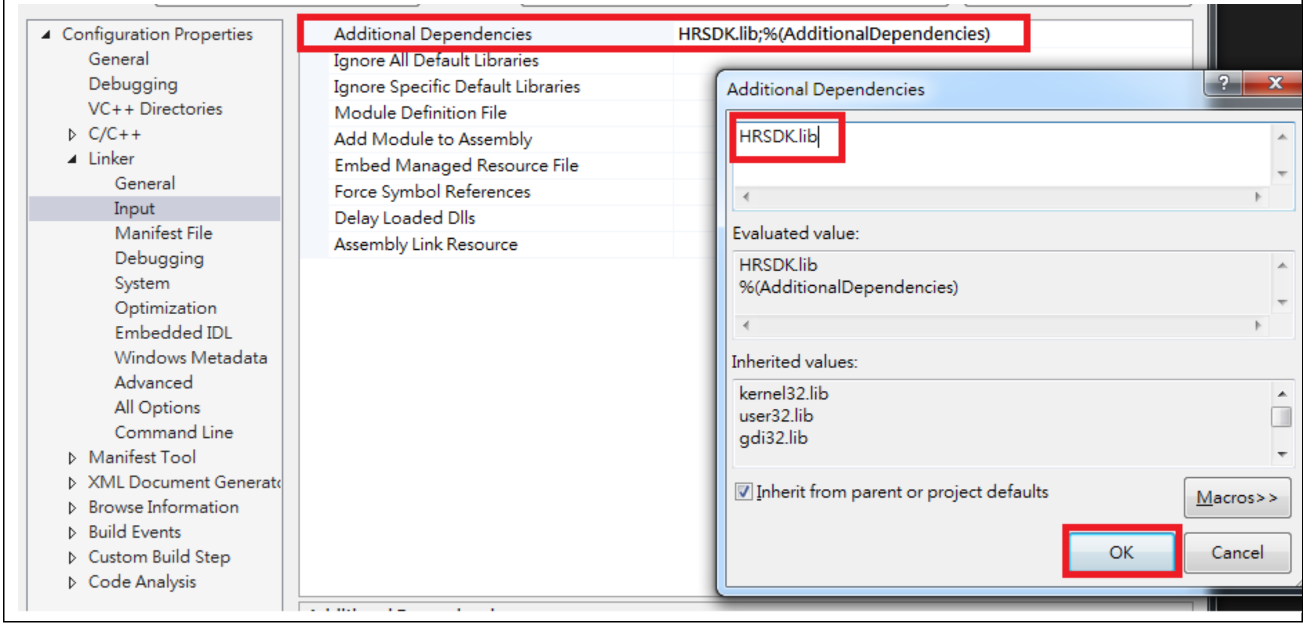
名稱	修改日期	類型	大小
HRSDK.dll	10/11/2016 2:26 ...	應用程式擴充	118 KB
HRSDK.h	10/11/2016 2:28 ...	C/C++ Header	18 KB
HRSDK.lib	10/11/2016 2:26 ...	Object File Library	45 KB

檔案名稱(N): HRSDK.h

STEP 4 : Right click project「Properties」



STEP 5 : Select linker -> Input ->Additional Dependencies -> enter「HRSDK.lib」-> OK



Additional Dependencies	HRSDK.lib;%(AdditionalDependencies)

Additional Dependencies

HRSDK.lib

Evaluated value:

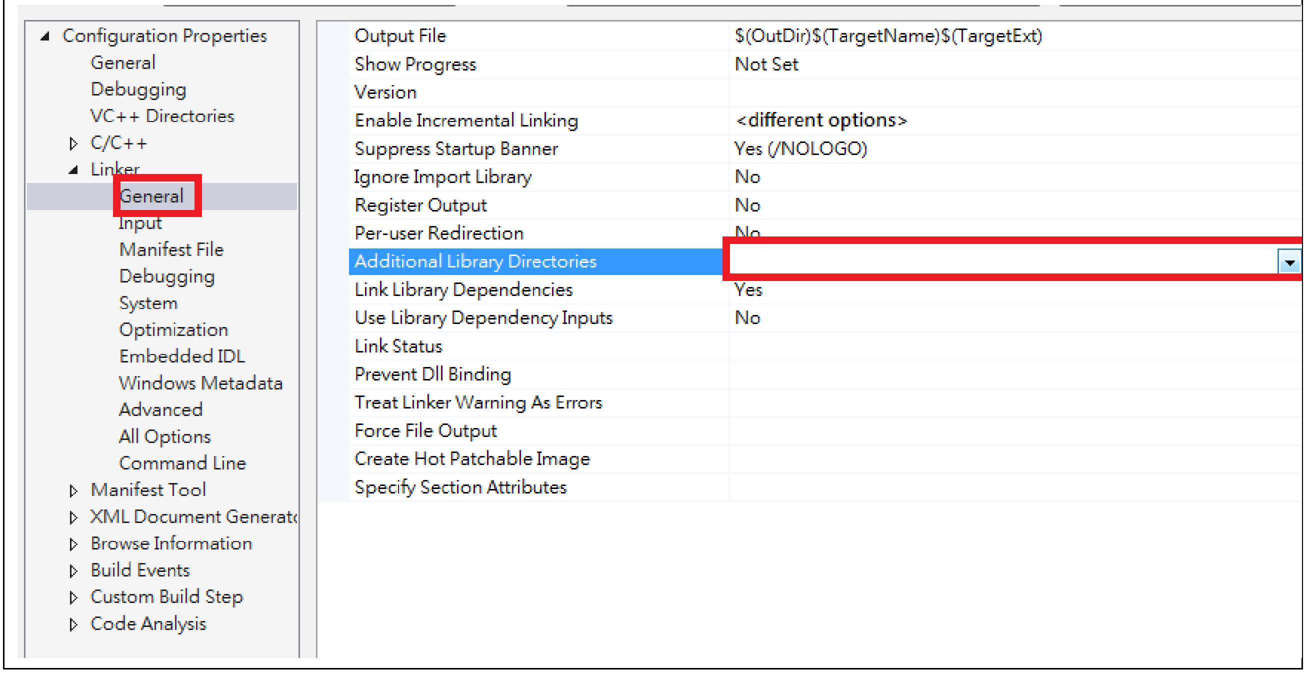
HRSDK.lib
%(AdditionalDependencies)

Inherited values:

kernel32.lib
user32.lib
gdi32.lib

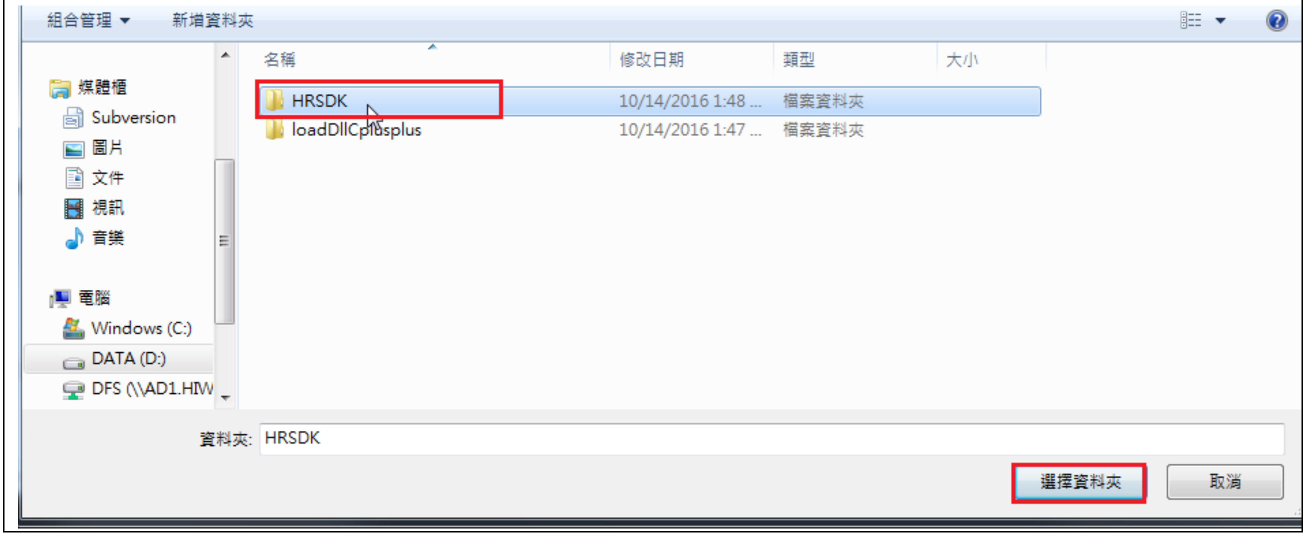
☑ Inherit from parent or project defaults

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



Output File	$(OutDir)$(TargetName)$(TargetExt)
Show Progress	Not Set
Version	
Enable Incremental Linking	<different options>
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Per-user Redirection	No
Additional Library Directories	
Link Library Dependencies	Yes
Use Library Dependency Inputs	No
Link Status	
Prevent Dll Binding	
Treat Linker Warning As Errors	
Force File Output	
Create Hot Patchable Image	
Specify Section Attributes	

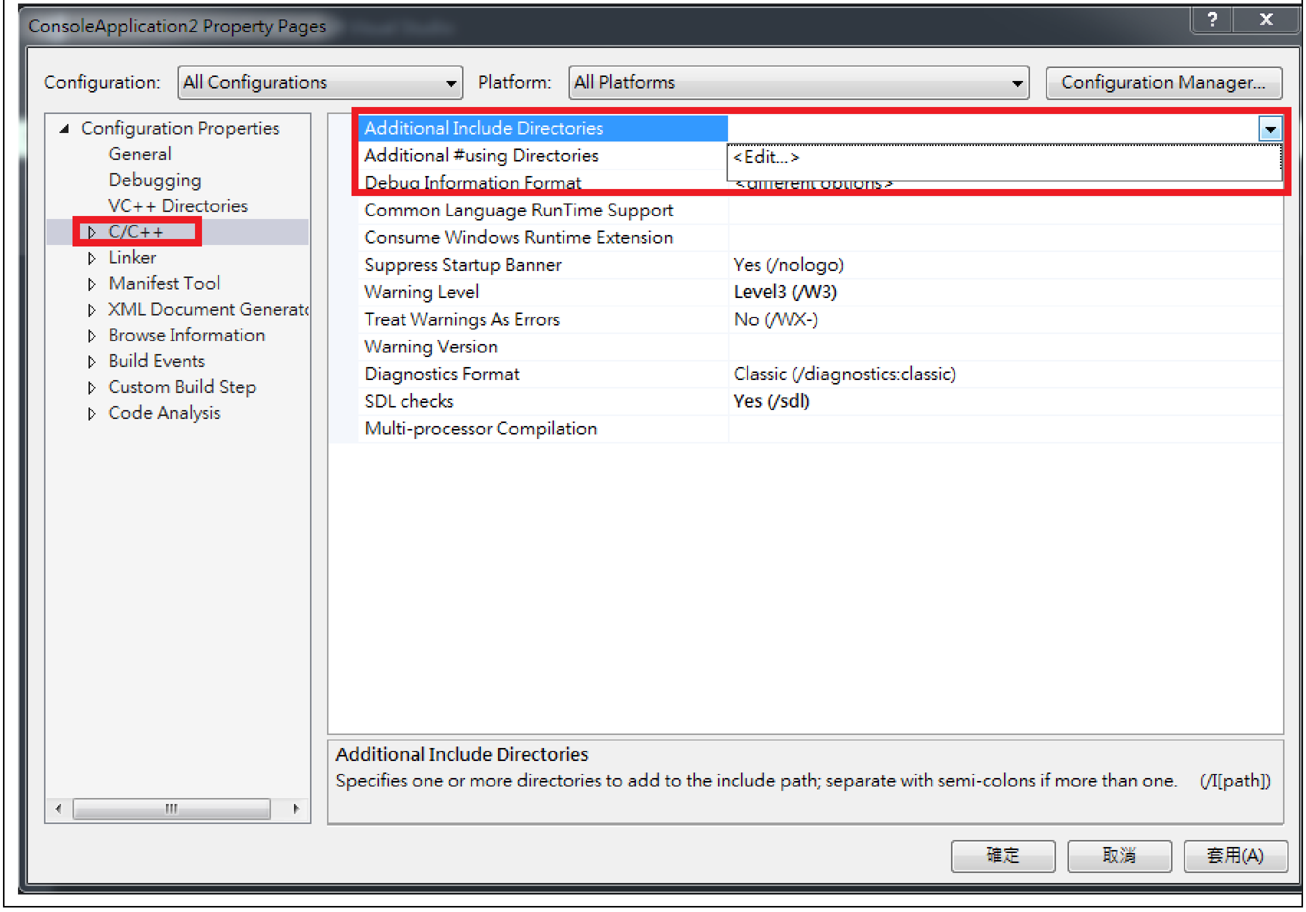
STEP 7 : Select the file where dll is located (If dll is under the same root directory of the program, step 6、7 can be skipped)



名稱	修改日期	類型	大小
HRSDK	10/14/2016 1:48 ...	檔案資料夾	
loadDllCplusplus	10/14/2016 1:47 ...	檔案資料夾	

資料夾: HRSDK

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



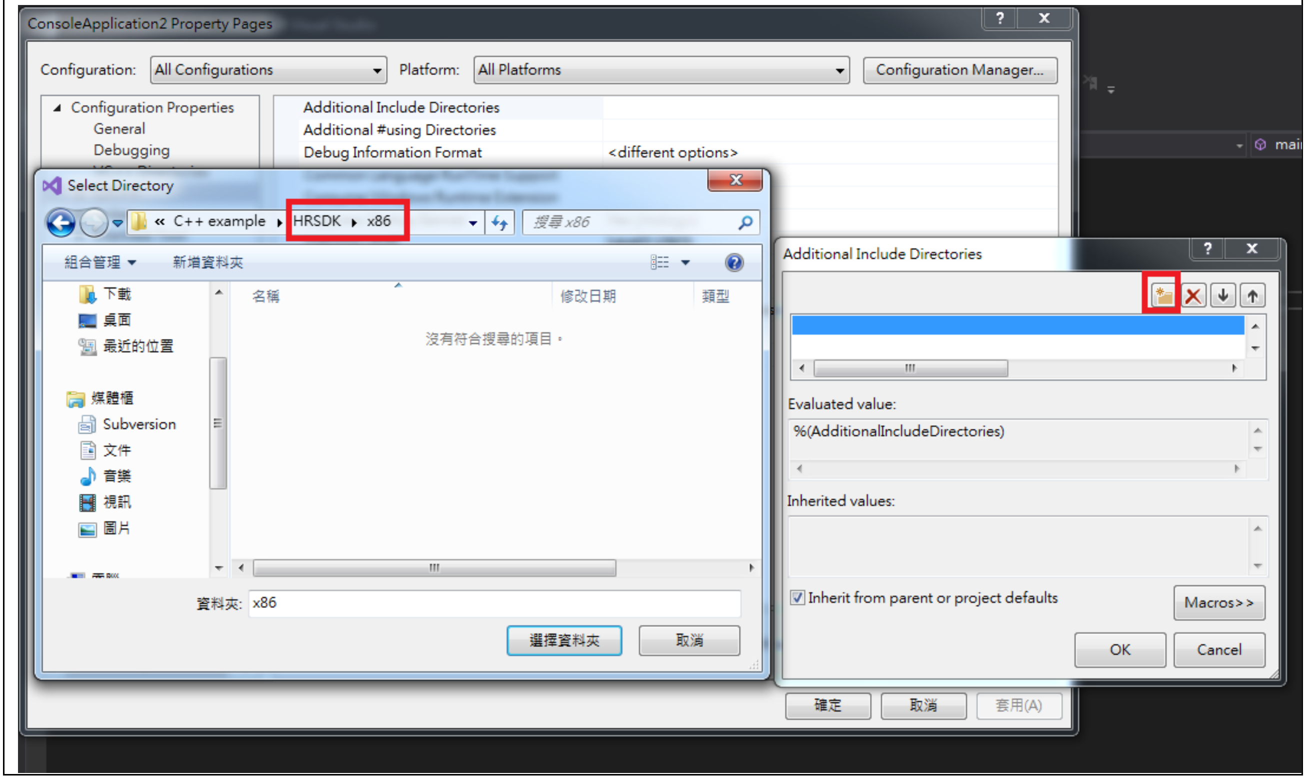
Configuration: All Configurations

Platform: All Platforms

Additional Include Directories	
Additional #using Directories	<Edit...>
Debug Information Format	<different options>
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	Yes (/nologo)
Warning Level	Level3 (/W3)
Treat Warnings As Errors	No (/WX-)
Warning Version	
Diagnostics Format	Classic (/diagnostics:classic)
SDL checks	Yes (/sdl)
Multi-processor Compilation	

Additional Include Directories
Specifies one or more directories to add to the include path; separate with semi-colons if more than one. (/I[path])

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



ConsoleApplication2 Property Pages

Configuration: All Configurations

Platform: All Platforms

Additional Include Directories

Additional #using Directories

Debug Information Format <different options>

Select Directory: « C++ example ▸ HRSDK ▸ x86

沒有符合搜尋的項目。

資料夾: x86

Additional Include Directories

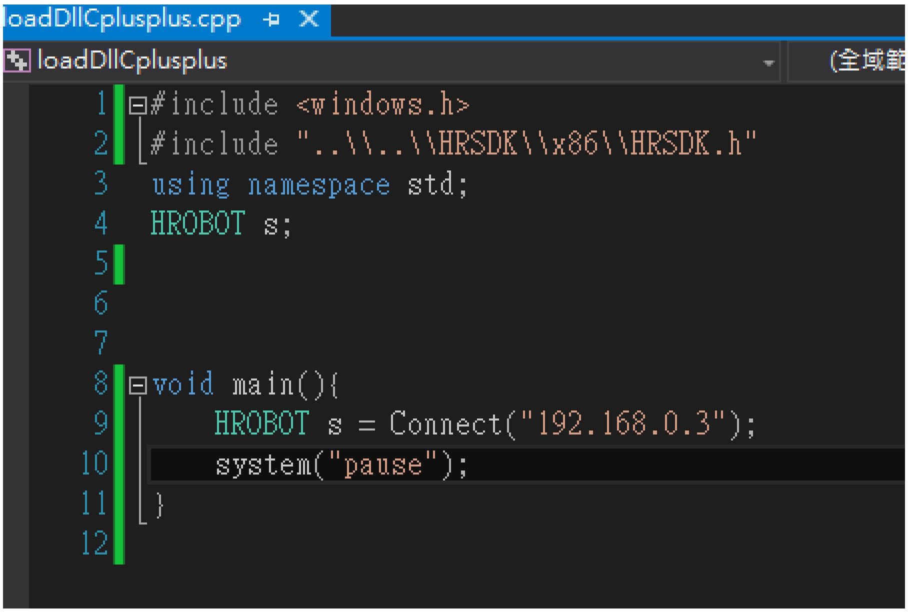
Evaluated value:

%(AdditionalIncludeDirectories)

Inherited values:

☑ Inherit from parent or project defaults

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



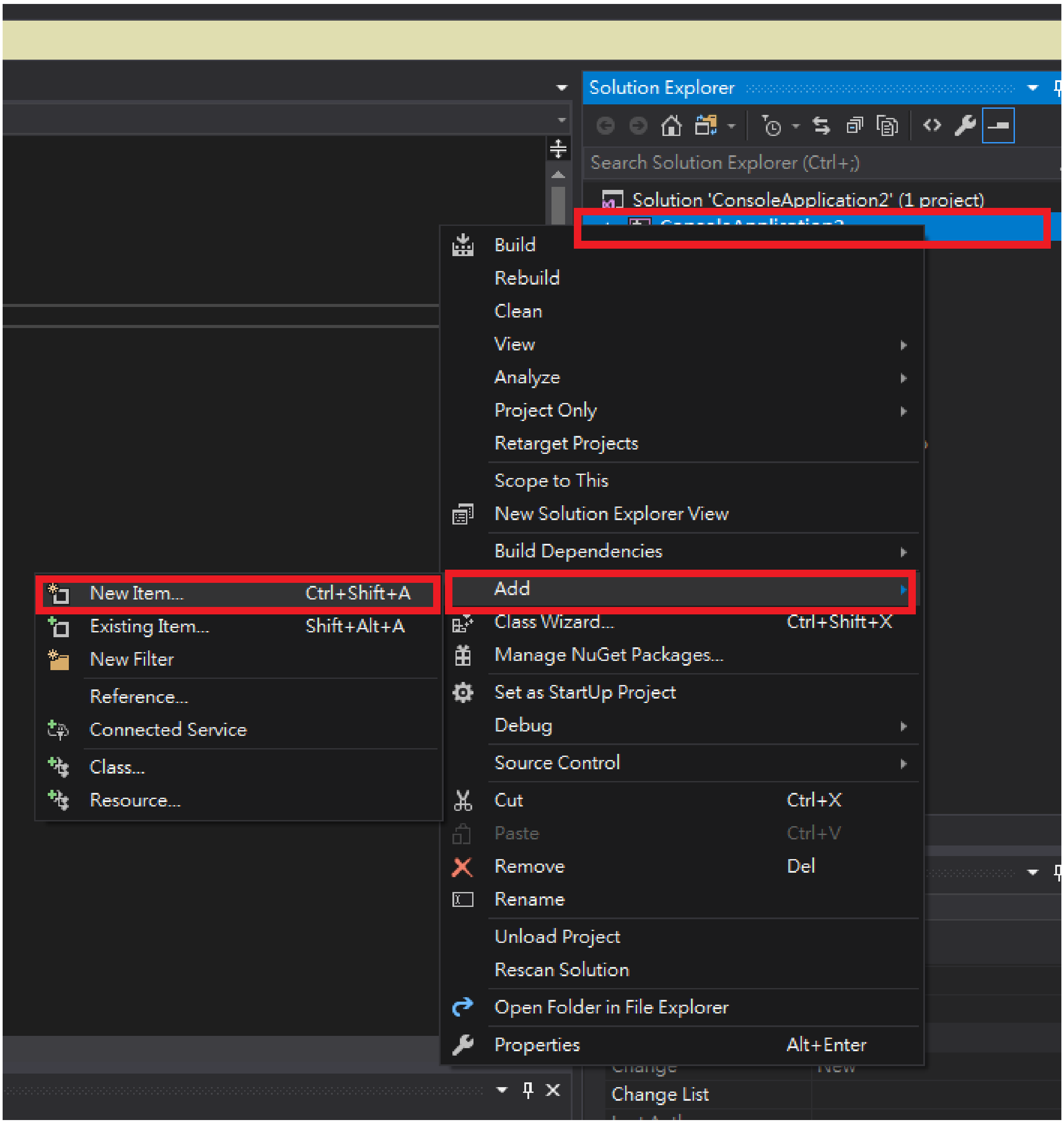
loadDllCplusplus.cpp

loadDllCplusplus

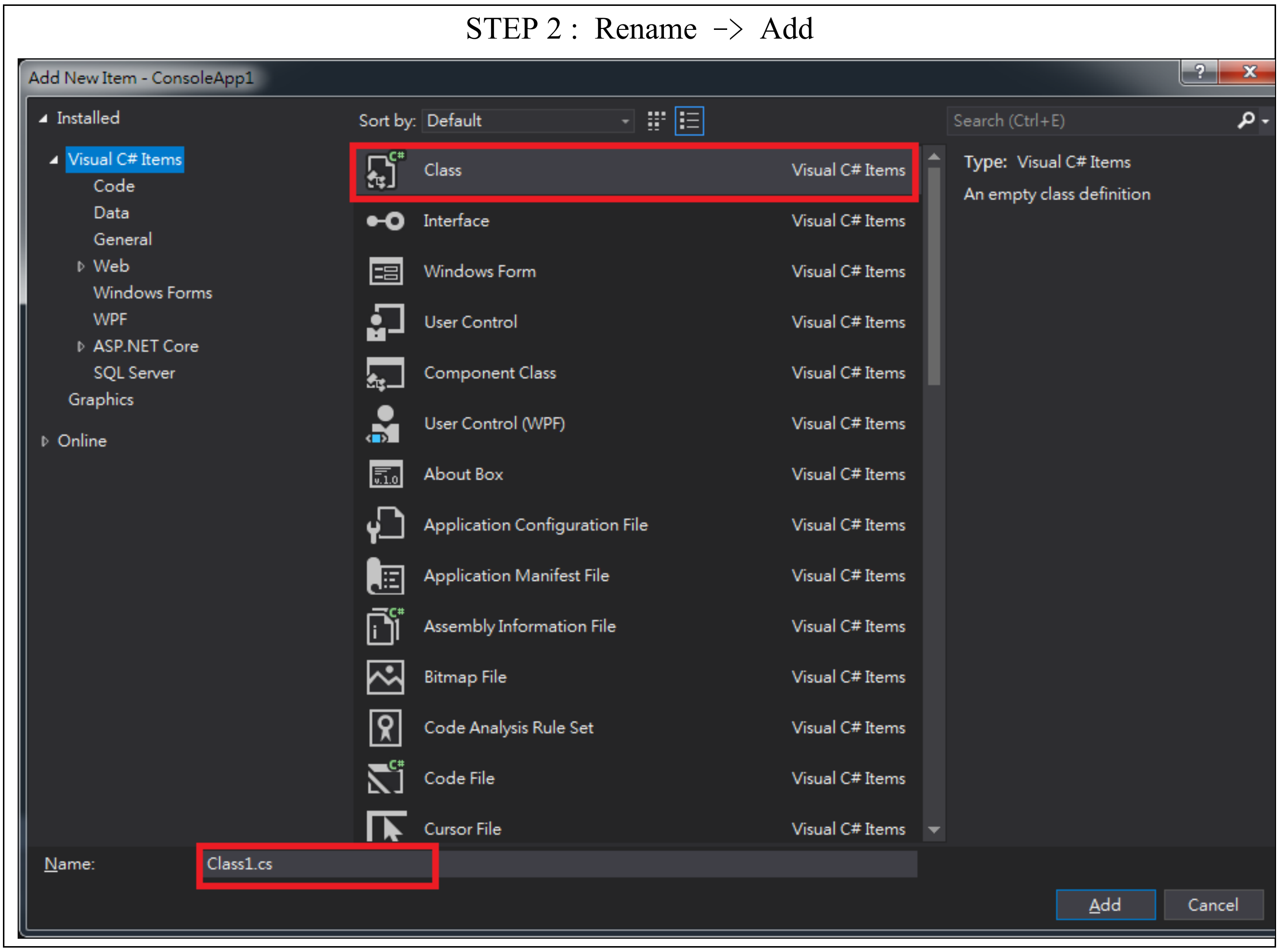
```
#include <windows.h>
#include "..\\..\\HRSDK\\x86\\HRSDK.h"
using namespace std;
HROBOT s;



void main(){
    HROBOT s = Connect("192.168.0.3");
    system("pause");
}
```

2.2 C#

STEP 1: Create C# progject -> Right click Project -> Add -> New



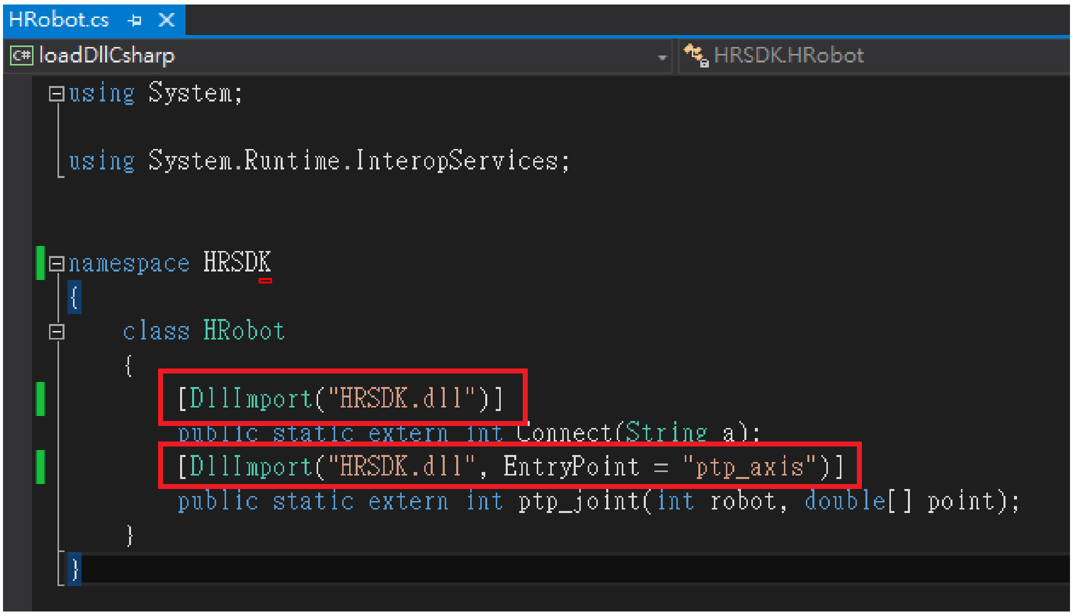
STEP 2： Rename -> Add


Add New Item - ConsoleApp1
Installed
Visual C# Items
Code
Data
General
Web
Windows Forms
WPF
ASP.NET Core
SQL Server
Graphics
Online
Sort by: Default
Search (Ctrl+E)
Class Visual C# Items
Interface Visual C# Items
Windows Form Visual C# Items
User Control Visual C# Items
Component Class Visual C# Items
User Control (WPF) Visual C# Items
About Box Visual C# Items
Application Configuration File Visual C# Items
Application Manifest File Visual C# Items
Assembly Information File Visual C# Items
Bitmap File Visual C# Items
Code Analysis Rule Set Visual C# Items
Code File Visual C# Items
Cursor File Visual C# Items
Type: Visual C# Items
An empty class definition
Add
Cancel

Name: Class1.cs

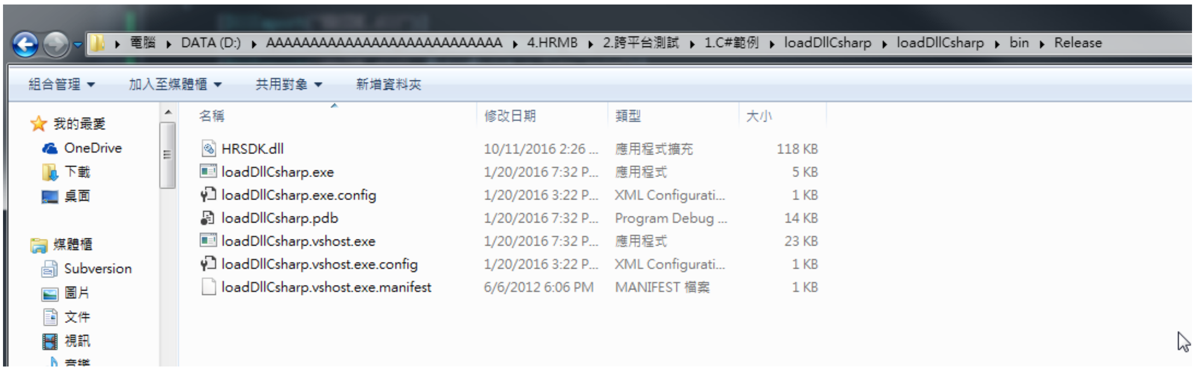
STEP 3： Enter required function. Name of the function can be changed.

EntryPoint name must match the name in dll.



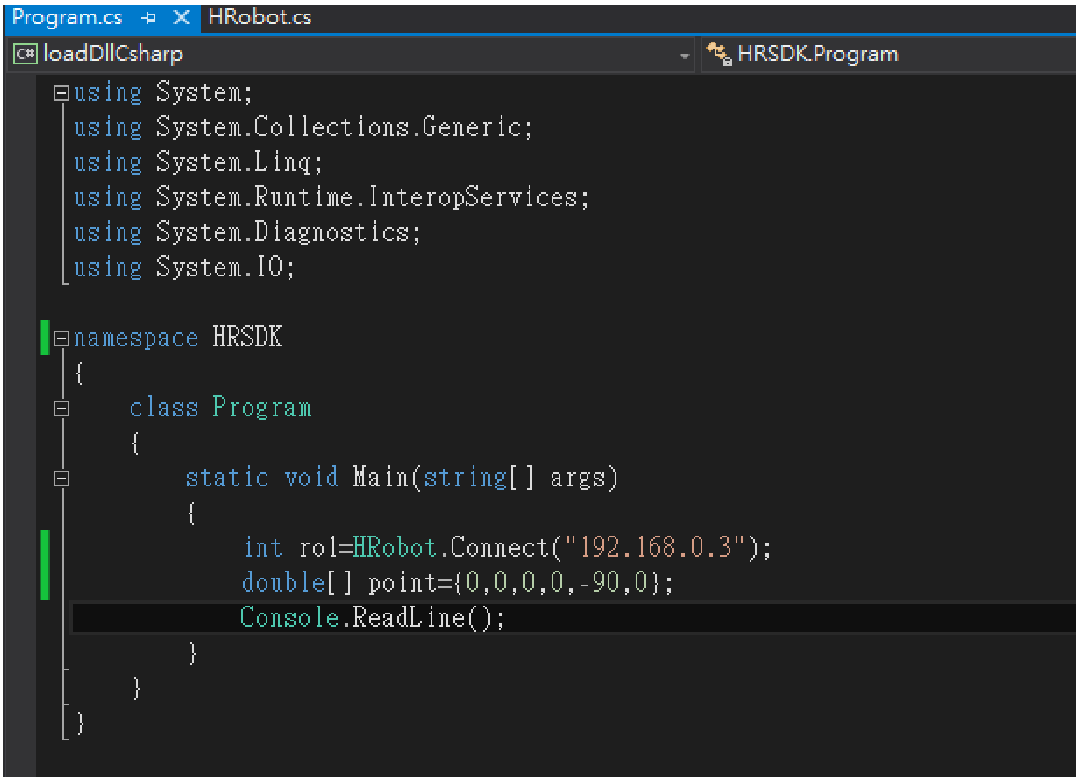
```
using System;

using System.Runtime.InteropServices;


namespace HRSDK
{
    class HRobot
    {
        [DllImport("HRSDK.dll")]
        public static extern int Connect(String a);
        [DllImport("HRSDK.dll", EntryPoint = "ptp_axis")]
        public static extern int ptp_joint(int robot, double[] point);
    }
}
```

STEP 4 : Put HRSDK.dll into 專案/bin/Debug(Release) route


電腦 ▸ DATA (D:) ▸ AAAAAAAAAAAAAAAAAAAAAAAAAAA ▸ 4.HRMB ▸ 2.跨平台測試 ▸ 1.C#範例 ▸ loadDllCsharp ▸ loadDllCsharp ▸ bin ▸ Release
組合管理
加入至媒體櫃
共用對象
新增資料夾
我的最愛
OneDrive
下載
桌面
媒體櫃
Subversion
圖片
文件
視訊
音樂

名稱	修改日期	類型	大小
HRSDK.dll	10/11/2016 2:26 ...	應用程式擴充	118 KB
loadDllCsharp.exe	1/20/2016 7:32 P...	應用程式	5 KB
loadDllCsharp.exe.config	1/20/2016 3:22 P...	XML Configurati...	1 KB
loadDllCsharp.pdb	1/20/2016 7:32 P...	Program Debug ...	14 KB
loadDllCsharp.vshost.exe	1/20/2016 7:32 P...	應用程式	23 KB
loadDllCsharp.vshost.exe.config	1/20/2016 3:22 P...	XML Configurati...	1 KB
loadDllCsharp.vshost.exe.manifest	6/6/2012 6:06 PM	MANIFEST 檔案	1 KB

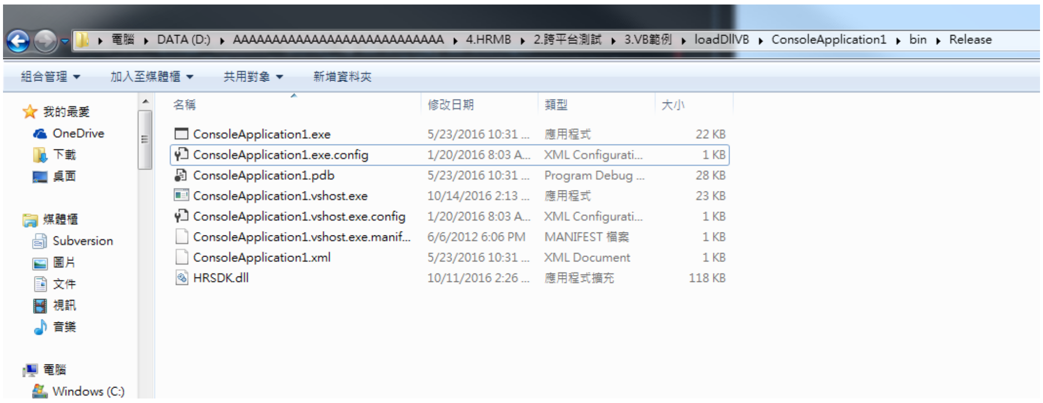
STEP 5 : Start using dll。



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.InteropServices;
using System.Diagnostics;
using System.IO;

namespace HRSDK
{
    class Program
    {
        static void Main(string[] args)
        {
            int rol=HRobot.Connect("192.168.0.3");
            double[] point={0,0,0,0,-90,0};
            Console.ReadLine();
        }
    }
}
```

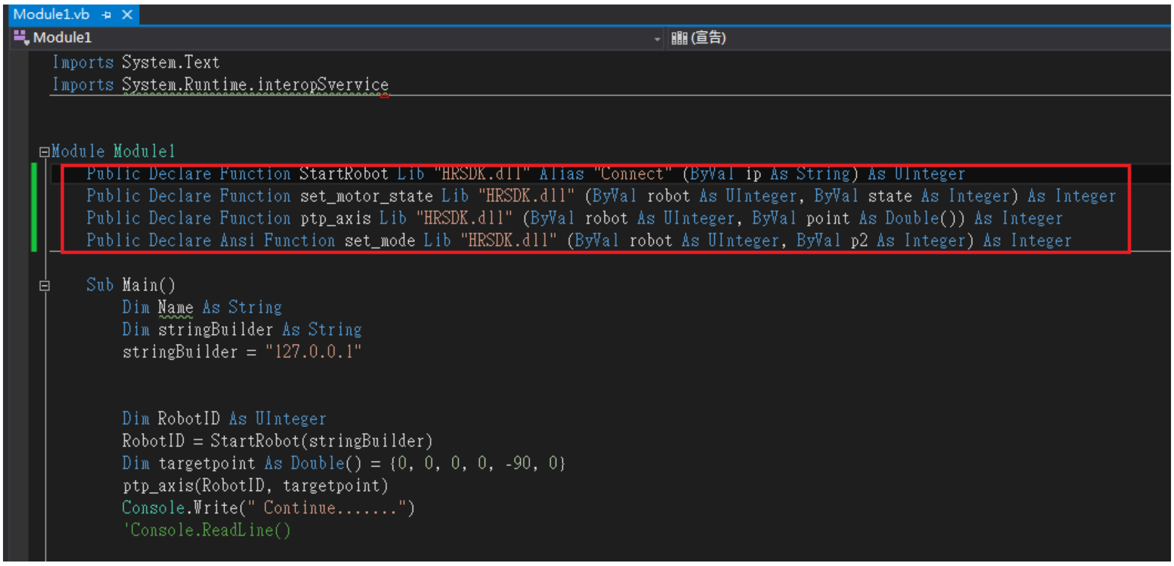
2.3 VB

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


電腦 ▸ DATA (D:) ▸ AAAAAAAAAAAAAAAAAAAAAAAAAAA ▸ 4.HRMB ▸ 2.跨平台測試 ▸ 3.VB範例 ▸ loadDllVB ▸ ConsoleApplication1 ▸ bin ▸ Release
組合管理
加入至媒體櫃
共用對象
新增資料夾
我的最愛
OneDrive
下載
桌面
媒體櫃
Subversion
圖片
文件
視訊
音樂
電腦
Windows (C:)

名稱	修改日期	類型	大小
ConsoleApplication1.exe	5/23/2016 10:31 ...	應用程式	22 KB
ConsoleApplication1.exe.config	1/20/2016 8:03 A...	XML Configurati...	1 KB
ConsoleApplication1.pdb	5/23/2016 10:31 ...	Program Debug ...	28 KB
ConsoleApplication1.vshost.exe	10/14/2016 2:13 ...	應用程式	23 KB
ConsoleApplication1.vshost.exe.config	1/20/2016 8:03 A...	XML Configurati...	1 KB
ConsoleApplication1.vshost.exe.manif...	6/6/2012 6:06 PM	MANIFEST 檔案	1 KB
ConsoleApplication1.xml	5/23/2016 10:31 ...	XML Document	1 KB
HRSDK.dll	10/11/2016 2:26 ...	應用程式擴充	118 KB

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



```
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 Safety Speed Limit Function Description

- Safety speed limit function

(1) Suitable for test running, programming and teaching

(2) Speed of linear motion will be limited at 250mm/s

(3) Speed ratio of point to point motion is set according to the model of robot

(4) Overall speed will be set at 10% for each start and shutdown, speed of linear motion is set at 250mm/s

(5) Speed limit function can be switched on or off with the command speed_limit_on and speed_limit_off

(6) Jog operation is allowed

- Switch on safety speed limit function

  Safety Speed limit function is used during debugging. Debugging includes setting, installing, adjusting, modifying or troubleshooting, the safety speed limit function of the robot shall be switched on. Under safety speed limit function, the following should be noticed:

  (1) New or modified program should only be allow to run when safety speed limit function is switched on

  (2) Tools, robot arm or external axis are forbidden to contact or extent to the fence in the safety zone.

  (3) If any workpiece, tool or component is stuck or dropped, any machine failure or short circuit, robot arm is forbidden to start up.

  (4) All debugging should be operated outside fence / safety zone

- Switch off safety speed limit function

  Switch off safety speed limit function must meet the safety protection measurement below:

  (1) All safety measurements should be tested and installed

  (2) All safety measurements for pausing should be able to resume their function

  (3) No operator should be remain within operation zone

  (4) MUST comply with standard operating specification

  (5) If robot stopped operation due to unknown reason, ONLY when emergency button is pressed then is allowed to enter danger zone.

- Safety speed limit function operation permission table:

Function	Safety speed limit function	Safety speed limit (OFF)
Speed increase setting	X	O
Point to point motion speed setting	O	O
Linear motion speed setting	O	O
Command setting	O	O
Tool, Base coordinate number setting	O	O
Definition of tool and base coordinate	O	X
Modify tool and base coordinate	O	X
Server on	O	O
Operation mode setting	O	O
Clear error	O	O

Address register setting	O	O
PTP, Linear, Arc motion command	O	O
Motion program: Pause, Continue, Stop, Delay	O	O
Jog	O	X
Achieve system parameters: Position, Speed, Error code	O	O

Function can be used under all mode

Function can be used only when safety speed limit function is switched ON

Function can be used only when safety speed limit function is switched OFF

2.5 Connection Level Description

- Connection LevelOperator: Allow to operate part of the application interface, when HRSS is running and the number of the connection is less than the maximum number of 10 connection.
  - Expert: Allow to operate all application interface, need to be connected when HRSS operation mode is set to EXT mode. Switching modes in the HRSS will cause the level of the connection to become the operator mode.
  - If more than one expert level is connected, user should self-allocate the order of usage.

2.6 Command Queue

The command below are the commands that will enter server side command queue:

```
HRSDK_API int __stdcall set_acc_dec_ratio(HROBOT s, int acc);
HRSDK_API int __stdcall set_ptp_speed(HROBOT s, int vel);
HRSDK_API int __stdcall set_lin_speed(HROBOT s, double vel);
HRSDK_API int stdcall ptp_pos(HROBOT s, int mode, double *p);
HRSDK_API int stdcall ptp_axis(HROBOT s, int mode, double *p);
HRSDK_API int stdcall ptp_rel_pos(HROBOT s, int mode, double *p);
HRSDK_API int stdcall ptp_rel_axis(HROBOT s, int mode, double *p);
HRSDK_API int stdcall ptp_pr(HROBOT s, int p);
HRSDK_API int stdcall lin_pos(HROBOT s, int mode, double smooth_value, double *p);
HRSDK_API int stdcall lin_axis(HROBOT s, int mode, double smooth_value, double *p);
HRSDK_API int stdcall lin_rel_pos(HROBOT s, int mode, double smooth_value, double *p);
HRSDK_API int stdcall lin_rel_axis(HROBOT s, int mode, double smooth_value, double *p);
HRSDK_API int stdcall lin_pr(HROBOT s, int mode, double smooth_value, int p);
HRSDK_API int stdcall circ_pos(HROBOT s, int mode, double* p_aux, double* p_end);
HRSDK_API int stdcall circ_axis(HROBOT s, int mode, double* p_aux, double* p_end);
HRSDK_API int __stdcall circ_pr(HROBOT s, int mode, int p1, int p2);
```

These command will be kept in server side command queue and execute one by one in FIFO (First In First Out) order.

3 Command List

3.1 Connection Command

Group	Function Name	Description	Operator
Connect	Connect	Connected with the system	O
	Close	Disconnect with the system	O
	get_HRSDK_version	Get HRSDK version number	O
	get_connection_level	Get connection level	O

3.2 Register Command

Group	Function Name	Description	Operator
	get_timer	Get robot's timer	O
	set_timer	Set robot's timer	╳
	get_counter	Get robot's counter	O
	set_counter	Set robot's counter	╳
	get_pr_type	Get position register coordinates type	O
	set_pr_type	Set position register coordinates type	╳
	get_pr_coordinate	Get position register coordinates value	O
	set_pr_coordinate	Set position register coordinates value	╳
	get_pr_tool_base	Get position register tool base number	O
	set_pr_tool_base	Set position register tool base number	╳
	set_pr	Set position register value	╳

3.3 System Variable Command

Group	Function Name	Description	Operator
	set_acc_dec_ratio	Set acceleration ratio	╳
	get_acc_dec_ratio	Get acceleration ratio	O
	set_ptp_speed_ratio	Set PTP movement speed ratio	╳
	get_ptp_speed_ratio	Get PTP movement speed ratio	O
	set_lin_speed	Set linear movement speed	╳
	get_lin_speed	Get linear movement speed	O
	set_override_ratio	Set override ratio	╳
	get_override_ratio	Get override ratio	O
	get_alarm_code	Get alarm code	O

	set_robot_id	Set the robot identification name	O
	get_robot_id	Get the robot identification name	O
	set_smooth_length	Set the motion smoothing radius	×

3.4 Input and Output Command

Group	Function Name	Description	Operator
	get_DI	Get input	O
	get_DO	Get output	O
	set_DO	Set output	O
	get_FI	Get function input	O
	get_FO	Get function output	O
	get_RI	Get robot input	O
	get_RO	Get robot output	O
	set_RO	Set robot output	O
	get_VO	Get solenoid valve output	O
	set_VO	Set solenoid valve output	O

3.5 Coordinate System Command

Group	Function Name	Description	Operator
Base Coordinates	set_base_number	Set base number	×
	get_base_number	Get base number	O
	define_base	Define base coordinates	×
	get_base_data	Get base coordinates	O
Tool Coordinates	set_tool_number	Set tool number	×
	get_ tool _number	Get tool number	O
	define_ tool	Define tool coordinates	×
	get_ tool _data	Get tool coordinates	O

3.6 Task Command

Group	Function Name	Description	Operator
	ext_task_start	RSR/PNS Start external trigger task	×
	task_ start	Start task	×
	task_hold	Hold current task	×
	task_ continue	Continue current task	×

	task_abort	Stop current task	×

3.7 Controller Setting Command

Group	Function Name	Description	Operator
	set_motor_state	Servo motor setting	×
	get_motor_state	Get servo motor state	O
	speed_limit_on	Turn on speed limit function	×
	speed_limit_off	Turn off speed limit function	×
	get_speed_limit_state	Get speed limit function state	O
	clear_alarm	Clear error	×

3.8 Jog

Group	Function Name	Description	Operator
	jog	Jog	×
	jog_stop	Stop jog	×
	jog_home	Jog return to home point	×

3.9 Motion Command

Group	Function Name	Description	Operator
PTP Motion	ptp_pos	Absolute coordinate position of PTP motion	×
	ptp_axis	Absolute joint angle of PTP motion	×
	ptp_rel_pos	Relative coordinate position of PTP motion	×
	ptp_rel_axis	Relative joint angle of PTP motion	×
	ptp_pr	Position register of PTP motion	×
Linear Motion	lin_pos	Absolute coordinate position of linear motion	×
	lin_axis	Absolute joint angle of linear motion	×
	lin_rel_pos	Relative coordinate position of linear motion	×

	lin_rel_axis	Relative joint angle of linear motion	╳
	lin_pr	Position register of linear motion	╳
CIRC Motion	circ_set_aux_pos	Set circular arc point of circular motion	╳
	circ_set_end_pos	Set end position of circular motion	╳
	circ_pos	Absolute coordinate position of circular motion	╳
	circ_pr	Position register of circular motion	╳
Motion Program	motion_hold	Hold motion	╳
	motion_continue	Continue motion	╳
	motion_abort	Stop motion	╳
	motion_delay	Delay motion	╳
	remove_command	Cancel unexecuted motion command	╳
	remove_command_tail	Cancel unexecuted motion command from the tail	╳
	set_command_id	Set motion command number	╳
	get_command_id	Get current motion command number	O
	get_command_count	Get current motion command from command queue	O
	get_motion_state	Get current motion state	O

3.10 Manipulator Information Command

Group	Function Name	Description	Operator
	get_encoder_count	Get current encoder value	O
	get_current_joint	Get current joint coordinate	O
	get_current_position	Get current absolute coordinate position	O
	get_current_rpm	Get current shaft speed	O
	get_device_born_date	Get device manufacture time	O
	get_operation_time	Get controller booting time	O
	get_mileage	Get motor mileage of each axis	O
	get_total_mileage	Get accumulative motor mileage of each axis	O

	get_utilization	Get accumulative utilization	O
	get_utilization_ratio	Get utilization ratio	O
	get_motor_torque	Get motor load percentage	O
	get_HRSS_version	Get HRSS version	O
	get_HRSS_version_v2	Get HRSS version	O
	get_robot_type	Get robot model	O
	get_robot_type_v2	Get robot model	O

3.11 Connection Command

Group	Function Name	Description	Operator
	update_hrss	Update controller HRSS software	✕
	send_file	Send hrb file	✕

4 Command Reference

4.1 Connection Command

4.1.1 Set Up Connection

HROBOT Connect(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 call back (uint16_t command uint16_t result uint16_t* message, int length);	Event receive function, used to receive message return by controller
return	HROBOT	Success: Device ID (0-65535 as valid device id) Fail: -1 connection is not set up -2 function is undefined

- When controller received first connection request from expert, speed limit function will be turned on. Follow-up expert/operator will not affect the setting of speed limit function when requesting for connection.
- 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 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

4.1.2 Disconnect Connection

void Close(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID

C++:

```
void __stdcall FuncName(uint16_t uint16_t uint16_t* int);

void main(){
   HROBOT robot;
   robot=Connect("192.168.0.3", 1 ,FuncName);
   //Do something
   Close(robot);
}

void __stdcall CallBackFun(uint16_t cmd, uint16_t rlt, uint16_t* msg, int len) {
   // process information from controller
}
```

4.1.3 Get HRSDK Version Number

void get_HRSDK_version(const char*& version)

Parameter	Data Type	Description
version	const char*	HRSDK version number

C++:

```
char* version;
get_HRSDK_version(version);
std::cout << version << std::endl;
```

4.1.4 Get Level of Connection

int get_connection_level(**HROBOT** **robot**)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	0: Operator 1: Expert

C++:

```
int state;
state=get_connection_level(robot);
```

4.2 Register Command

4.2.1 Get Robot Timer

int get_timer(HROBOT robot, int timer_num)

Parameter	Data Type	Description
robot	HROBOT	Device ID
timer_num	int	Timer number (1-20)
return	int	Success: Value of timer Fail: -1

C++:

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

4.2.2 Get Robot Timer

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

Parameter	Data Type	Description
robot	HROBOT	Device ID
timer_num	int	Timer number (1-20)

value	int	Range of value(-999999999-999999999)
return	int	Success: 0 Fail: -1

```
C++:
  set_timer(robot,1,100); //set Timer number 1 with value 100
```

4.2.3 Get Robot Counter

int get_counter(HROBOT robot, int counter_num)

Parameter	Data Type	Description
robot	HROBOT	Device ID
counter_num	int	Counter number (1-20)
return	int	Success: value of counter Fail: -1

```
C++:
  set_counter(robot,1,100); //set Counter number 1 with value 100
  ret=get_counter(robot,1);    //get value from Counter number 1
```

4.2.4 Set Robot Counter

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

Parameter	Data Type	Description
robot	HROBOT	Device ID
counter_num	int	Counter number (1-20)
value	int	Range of value(-999999999-999999999)
return	int	Success: 0 Fail: -1

```
C++:
  set_counter(robot,1,100); //set Counter number 1 with value 100
```

4.2.5 Get Position Register Coordinate System 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	int	Cartesian coordinate: 0 Joint coordinate: 1 Fail: -1

4.2.6 Set Position Register Coordinate System 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: 0 Joint: 1
return	int	Success: 0 Fail: Error code

4.2.7 Get Position Register 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{X,Y,Z,A,B,C}
		Joint coordinate{A1,A2,A3,A4,A5,A6}

return	int	Success: 0 Fail: -1

4.2.8 Set Position Register 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{X,Y,Z,A,B,C}
		Joint coordinate{A1,A2,A3,A4,A5,A6}

return	int	Success: 0 Fail: error code

4.2.9 Get Position Register of Tool, Base Coordinate

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	int	Success: 0 Fail: -1

4.2.10 Set Position Register of Tool, Base Coordinate

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(1-100)
tool	int	Tool number
base	int	Base number
return	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]
```

(1). Set pr information

(2). Same effect as (1)

(3). Obtain information of position register

A. Coordinate type

B. Joint coordinate

C. Tool number

D. Base number

4.2.11 Set Position Register Data

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

Parameter	Data Type	Description
robot	HROBOT	Device ID
pr_num	int	Position register number(1-4000)

coor_type	int	Coordinate system type Cartesian coordinate:0 Joint coordinate:1
coor	double[6]	Desired position register coordinate
tool	int	Tool coordinate number
base	int	Base coordinate number
return	int	Success: 0 Fail: -1

```
C++:
        (1)
        double coor[6]={0,0,0,0,-90,0};
        set_pr(robot,400,1,coor,2,2); //set address register
                                     //address register number:400
                                     //coordinate type:joint
                                     //coordinate:{0,0,0,0,-90,0}
                                     //tool number:2
                                     //base number:2
```

4.3 System Variable Command

4.3.1 Set Acceleration 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	int	Success: 0 Fail: Error code

4.3.2 Get Acceleration Ratio

int get_acc_dec_ratio(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID

return	int	Success: acceleration ratio 1-100(%) Fail: -1

C++:

```
speed_limit_off(robot);
set_acc_dec_ratio(robot,20);
acc=get_acc_dec_ratio (robot);
```

⚠ Accelerate/Decelerate ratio can be set only when safety speed limit function is switched off

4.3.3 Set speed ratio of point-to-point(PTP) motion

int set_ptp_speed(HROBOT robot, int value)

Parameter	Data Type	Description
robot	HROBOT	Device ID
value	int	PTP speed ratio1-100(%)
return	int	Success: 0 Fail: Error code

4.3.4 Get speed ratio of point-to-point(PTP) motion

int get_ptp_speed(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: speed ratio 1-100(%) Fail: -1

C++:

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

4.3.5 Set speed of linear motion

int set_lin_speed(HROBOT robot, double value)

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

4.3.6 Get speed of linear motion

double get_lin_speed(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	double	Success: Speed of linear motion(mm/s) Fail: -1

C++:

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

4.3.7 Set Override Speed Ratio

int set_override_ratio(HROBOT robot, int value)

Parameter	Data Type	Description
robot	HROBOT	Device ID
value	int	Override speed ratio 1-100(%)
return	int	Success: 0 Error: Error code

4.3.8 Get Override Speed Ratio

int get_override_ratio(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: Override speed ratio 1-100(%) Fail: -1

C++:

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

4.3.9 Get Error Code

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

Parameter	Data Type	Description
robot	HROBOT	Device ID
count	int&	Return number of alarm
alarm_code	unsigned uint64_t[20]	Alarm code array Store up to 20 alarms Call clear_alarm to clear the array For alarm code correspondence, please refer to HRSS software manual Convert to hexadecimal display with corresponding to software manual
return	int	Success: 0 Fail: -1

C++:

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

4.3.10 Set Robot Identification Name

int set_robot_id(HROBOT robot, char* robot_id)

Parameter	Data Type	Description
robot	HROBOT	Device ID
alarm_code	char*	Robot name character array: Up to 50 characters can be used Uppercase and lowercase letters, numbers, and various symbol characters can be used. No other language is allowed besides English
return	int	Success:0 Fail: -1

```
C++:
    char robot_id[]="RCA00000000 ";
    set_robot_id (robot, robot_id);
```

4.3.11 Get Robot Identification Name

int get_robot_id(HROBOT robot, char* robot_id)

Parameter	Data Type	Description
robot	HROBOT	Device ID
alarm_code	char*	Robot name character array: Up to 50 characters can be used Uppercase and lowercase letters, numbers, and various symbol characters can be used. No other language is allowed besides English
return	int	Success:0 Fail: -1

```
C++:
    char robot_id[50];
    get_robot_id (robot, robot_id);
```

4.3.12 Set Motion Smooth Radius

double set_smooth_length (HROBOT robot, int radius)

Parameter	Data Type	Description
robot	HROBOT	Device ID
radius	int	Motion smooth radius need to be greater than 100
return	double	Success: 0 Fail: Error code

```
C++:
      double radius = 200.0;
      set_smooth_length (robot, radius);
```

4.4 Output/Input Command

4.4.1 Get Input State

int get_DI(HROBOT robot ,int index)

Parameter	Data Type	Description
robot	HROBOT	Device ID
index	Int	Index of input [1-48]
return	Int	0:OFF 1:ON

```
C++:
    int state;
    state=get_DI(robot,1);
```

4.4.2 Get Output State

int get_DO(HROBOT robot ,int index)

Parameter	Data Type	Description
robot	HROBOT	Device ID
index	int	Index of output [1-48]
return	int	0:OFF 1:ON

```
C++:
    int state;
    state=get_DO(robot,1);
```

4.4.3 Set Output State

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

Parameter	Data Type	Description
robot	HROBOT	Device ID
index	int	Index of output [1-48]
value	bool	true or false
return	int	Success: 0 Fail:-1

```
C++:
    int state;
    set_DO(robot,1,true);
```

4.4.4 Get Functional Input State

int get_FI(HROBOT robot ,int index)

Parameter	Data Type	Description
robot	HROBOT	Device ID
index	int	Index of functional input 0: Start 1: Hold 2: Stop 3: Enable 4: RSR1

		5: RSR2 6: RSR3 7: RSR4
return	int	0: OFF 1: ON

```
C++:
   int state;
   state=get_FI(robot,1);
```

4.4.5 Get Functional Output State

int get_FO(HROBOT robot ,int index)

Parameter	Data Type	Description
robot	HROBOT	Device ID
index	int	Index of functional output[0-7] 0: Run 1: Held 2: Fault 3: Ready 4: ACK1 5: ACK2 6: ACK3 7: ACK4
return	Int	0:OFF 1:ON

```
C++:
   int state;
   state=get_FO(robot,1);
```

4.4.6 Get Robot Input

int get_RI(HROBOT robot , int index)

Parameter	Data Type	Description
robot	HROBOT	Device ID
index	int	Index of input [1-8]
return	int	0:OFF 1:ON

C++:

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

4.4.7 Get Robot Output

int get_RO(HROBOT robot , int index)

Parameter	Data Type	Description
robot	HROBOT	Device ID
index	int	Index of output [1-8]
return	int	0:OFF 1:ON

C++:

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

4.4.8 Set Robot Output

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

Parameter	Data Type	Description
robot	HROBOT	Device ID
index	int	Index of output [1-8]
value	bool	true or false
return	int	Success: 0 Fail: Error code

C++:

```
int state;
state=set_RO(robot,1,true);
```

4.4.9 Get Solenoid Valve Output

int get_VO(HROBOT robot , int index)

Parameter	Data Type	Description
robot	HROBOT	Device ID

index	int	Index of output[1-3]
return	int	0:OFF 1:ON

```
C++:
   int state;
   state=get_VO(robot,1);
```

4.4.10 Set Solenoid Valve Output

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

Parameter	Data Type	Description
robot	HROBOT	Device ID
index	int	Index of output[1-3]
value	bool	true or false
return	int	Success: 0 Fail: Error code

```
C++:
   int state;
   set_VO(robot,1,true);
```

4.5 Coordinate Command

4.5.1 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	Select desired base number (0-31)
return	int	Success: 0 Fail: Error code

4.5.2 Get Base Number

int get_base_number(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: base number(0-31) Fail: -1

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

4.5.3 Define Base Coordinate

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

Parameter	Data Type	Description
robot	HROBOT	Device ID
baseNum	int	Base coordinate number(1-31) Not allow to define base number(0) Base number(1-30): Allow customized coordinate

coor	double[6]	Coordinate{X,Y,Z,A,B,C}
		Range(2147418.112,-2147418.112)

return	int	Success: 0 Fail: Error code

4.5.4 Get Base Coordinate

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

Parameter	Data Type	Description
robot	HROBOT	Device ID
num	int	Obtain desired base coordinate number(0-31)
coor	double[6]	Coordinate{X,Y,Z,A,B,C}
return	int	Success: 0 Fail: Error code

4.5.5 Set Tool Number

int set_tool_number(HROBOT robot ,int num)

Parameter	Data Type	Description
robot	HROBOT	Device ID
num	int	Select desired tool number(0-15)
return	int	Success: 0 Fail: Error code

4.5.6 Get Tool Number

int get_tool_number(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: Tool number(0-15) Fail: Error code

```
C++:
    set_tool_number(robot,20);              // set_tool_number
    int vel=get_tool_number(robot);// get_tool_number
```

4.5.7 Define Tool Coordinate

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

Parameter	Data Type	Description
robot	HROBOT	Device ID
toolNum	int	Desired definition of tool number(1-15) Tool number 0 not allow to define coordinate Tool number 1-15: Allow to self-define coordinate

coor	double[6]	Coordinate{X,Y,Z,A,B,C}
		Range(2147418.112,-2147418.112)

return	int	Success: 0 Fail: Error code

4.5.8 Get Tool Coordinate

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

Parameter	Data Type	Description
robot	HROBOT	Device ID
num	int	Desired definition of tool number (0-15)
coor	double[6]	Coordinate{X,Y,Z,A,B,C} Numerical range (2147418.112,- 2147418.112)
return	int	Success: 0 Fail: Error code

```
C++:
(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);
```

(1) Define tool coordinate, obtain coordinate information using get_tool_data

(2) Define base coordinate, obtain coordinate information using get_base_data

4.6 Task Command

4.6.1 RSR/PNS Start an External Trigger Task

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	int	Success: 0 Fail: Error code

⚠ Set the RSR/PNS task number before using this command.

⚠ When executing this command, the command will fail if there are other motion command. Before execution, execute motion_abort is required to clear the motion queue or wait for the motion command to complete.

4.6.2 Task Start

int task_ start(HROBOT robot ,char* file_name)

Parameter	Data Type	Description
robot	HROBOT	Device ID
file_name	char*	Task name Task needed to be present in HRSS
return	int	Success: 0 Fail: Error code

⚠ When executing this command, the command will fail if there are other motion command. Before execution, execute motion_abort is required to clear the motion queue or wait for the motion command to complete.

C++:

```
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"));    // 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"); // successful

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

4.6.3 Task Hold

int task_ hold(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: 0 Fail: Error code

4.6.4 Task Continue

int task_ continue(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: 0 Fail: Error code

4.6.5 Task Abort

int task_ abort(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: 0 Fail: Error code

```
C++:
(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");
        Sleep(3000);
        task_abort(s);
        Sleep(100);
```

(1) Start PNS mode task number 9

Start the task of PNS task 94 after running for 3 seconds

- If task 9 has been executed, task 94 will be executed sequentially

- If execution of task 9 is not completed, the task fails to start and returns error code 4013. Task 9 will continue to execute.

Hold the task for 1 second then continue the task

Continue the task for 1 second then stop the task

(2) Start RSR mode task number 4

Stop the task after running for 3 seconds

(3) Start the task which the task name is “task1”

Stop the task after running for 3 seconds

4.7 Controller Command

4.7.1 Servo Setting

int set_motor_state(HROBOT robot ,int state)

Parameter	Data Type	Description
robot	HROBOT	Device ID
state	int	0:Servo shutdown 1:Servo start After the servo started, it required to take about 100ms to execute other motion commands
return	int	Success: 0 Fail: Error code

4.7.2 Get Servo State

int get_motor_state(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Deivce ID
return	int	Servo shutdown: 0 Servo start:1 Fail: -1

C++:

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

(1). Determine whether the motor is energized, if not energized the motor

4.7.3 Safety Speed Limitation Function ON

int speed_limit_on(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: 0 Fail: Error code

4.7.4 Safety Speed Limitation Function OFF

int speed _limit_off(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: 0 Fail: Error code

⚠ Switch ON Safety Speed Limit Function

- When new expert level is connected, the safety speed limit function will turn on automatically.
- Motion command speed limit at 250mm/s
- Jog command is allowed
- If set_lin_speed speed parameter is over 250, it will be set as 250
- Overall speed will be set as 10%
- PTP speed ratio will be set as 10%
- Accelerate ratio cannot be set

⚠ Switch OFF Safety Speed Limit Function

- When new connection is connected but not expert level, the safety speed limit function will not be turned on.
- Motion command speed limit depends on the model
- Jog command not allowed
- Overall speed will be set as 10%

4.7.5 Get Safety Speed Limit Function State

int get_ speed _limit_state(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	0: Function OFF 1: Function ON Fail: -1

4.7.6 Clear Error

int clear_alarm (HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: 0 Fail: Error code

C++:

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

4.8 Jog

4.8.1 Jog

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

Parameter	Data Type	Description
robot	HROBOT	Device ID
space_type	int	Coordinate type 0: Base coordinate (Cartesian coordinate) 1: Joint coordinate 2: Tool coordinate (Cartesian coordinate)
index	int (0~5)	Jog object Cartesian coordinate (X:0,Y:1,Z:2,A:3,B:4,C:5) Joint coordinate (A1:0, A2:1, A3:2, A4:3, A5:4, A6:5)
dir	int	Direction

		1: Positive direction -1:Negiatve direction
return	int	Success: 0 Fail: Error code

⚠ Only valid in safety speed limit mode

4.8.2 Stop Jog

int jog_stop(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: 0 Fail: Error code

⚠ Only valid in safety speed limit mode

4.8.3 Jog Back to Home Position

int jog_home(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: 0 Fail: Error code

⚠ Only valid in safety speed limit mode

4.9 Motion Command

4.9.1 Absolute Coordinate PTP Motion Position

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

Parameter	Data Type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Smoothing function OFF 1: Speed of smooth according to two line
p	double[6]	Cartesian coordinate{X,Y,Z,A,B,C} Value range (2147418.112,- 2147418.112)

return	int	Success: 0 Fail: Error code

4.9.2 Absolute Joint Angle PTP motion

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

Parameter	Data Type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Smoothing function OFF 1: Speed of smooth according to two line
p	double[6]	Joint coordinate{A1,A2,A3,A4,A5,A6} Value range (2147418.112,- 2147418.112)
return	int	Success: 0 Fail: Error code

4.9.3 Relative Coordinate Position PTP Motion

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

Parameter	Data Type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Smoothing function OFF 1: Speed of smooth according to two line
p	double[6]	Cartesian coordinate{X,Y,Z,A,B,C} Value range (2147418.112,- 2147418.112)
return	int	Success: 0 Fail: Error code

4.9.4 Relative Joint Angle PTP Motion

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

Parameter	Data Type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Smoothing function OFF 1: Speed of smooth according to two line
p	double[6]	Joint coordinate{A1,A2,A3,A4,A5,A6} Value range (2147418.112,- 2147418.112)
return	int	Success: 0 Fail: Error code

4.9.5 Position Register PTP Motion

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

Parameter	Data Type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Smoothing function OFF 1: Speed of smooth according to two line
p	int	Position register number(0-100)
return	int	Success: 0 Fail: Error code

```
C++:
    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, 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
```

4.9.6 Absolute Coordinate Position Linear Motion

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: Smoothing function OFF 1: Bezier curve smoothing percentage 2: Bezier curve smoothing radius 3: Speed of smooth according to two line
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{X,Y,Z,A,B,C} Value range (2147418.112,- 2147418.112)

return	int	Success: 0 Fail: Error code

4.9.7 Absolute Joint Angle Linear Motion

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: Smoothing function OFF 1: Bezier curve smoothing percentage 2: Bezier curve smoothing radius 3: Speed of smooth according to two line
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{X,Y,Z,A,B,C} Value range (2147418.112,- 2147418.112)
return	int	Success: 0 Fail: Error code

4.9.8 Relative Coordinate Position Linear Motion

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: Smoothing function OFF 1: Bezier curve smoothing percentage 2: Bezier curve smoothing radius

		3: Speed of smooth according to two line
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{X,Y,Z,A,B,C} Value range (2147418.112,- 2147418.112)
return	int	Success: 0 Fail: Error code

4.9.9 Relative Joint Angle Linear Motion

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: Smoothing function OFF 1: Bezier curve smoothing percentage 2: Bezier curve smoothing radius 3: Speed of smooth according to two line
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{A1,A2,A3,A4,A5,A6} Value range (2147418.112,- 2147418.112)
return	int	Success: 0 Fail: Error code

4.9.10 Positon Register Linear Motion

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: Smoothing function OFF 1: Bezier curve smoothing percentage 2: Bezier curve smoothing radius 3: Speed of smooth according to two line
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(0-100)
return	int	Success: 0 Fail: Error code

C++:

```
  double AxishomePoint[6]={0,0,0,0,-90,0};
  lin_axis(s, 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
```

4.9.11 Absolute Coordinate Position Circular Motion

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: Smoothing function OFF 1: Speed of smooth according to two line
p_aux	double[6]	Arc point of circular motion Cartesian coordinate{X,Y,Z,A,B,C} Value range (2147418.112,- 2147418.112)
p_end	double[6]	End of circular motion Cartesian coordinate{X,Y,Z,A,B,C} Value range (2147418.112,- 2147418.112)
return	int	Success: 0 Fail: Error code

4.9.12 Joint Coordinate Position Circular Motion

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: Smoothing function OFF 1: Speed of smooth according to two line
p_aux	double[6]	Arc point of circular motion Joint coordinate{A1, A2, A3, A4, A5, A6} Value range (2147418.112,- 2147418.112)
p_end	double[6]	End point of circular motion

		Joint coordinate{A1, A2, A3, A4, A5, A6} Value range (2147418.112,- 2147418.112)
return	int	Success: 0 Fail: Error code

C++:

```
  //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);
```

4.9.13 Position Register Circular Motion

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

Parameter	Data Type	Description
robot	HROBOT	Device ID
mode	int	Smooth mode 0: Smoothing function OFF 1: Speed of smooth according to two line
p1	int	Position register number(0-100)
p2	int	Position register number(0-100)
return	int	Success: 0 Fail: Error code

```
C++:
   //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
```

4.9.14 Hold Motion

int motion_hold(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: 0 Fail: Error code

⚠ The instruction will fail when there is a task executing.

4.9.15 Continue Motion

int motion_continue(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: 0 Fail: Error code

⚠ The instruction will fail when there is a task executing.

4.9.16 Abort Motion

int motion_abort(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: 0 Fail: Error code

⚠ The instruction will fail when there is a task executing.

4.9.17 Delay Motion

int motion_delay(HROBOT robot ,int delay)

Parameter	Data Type	Description
robot	HROBOT	Device ID
delay	int	Delay time, unit ms
return	int	Success: 0 Fail: Error code

⚠ The instruction will fail when there is a task executing.

C++:

```
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"));    // 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"); // successful

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

4.9.18 Set Motion Command Number

int set_command_id(HROBOT robot ,int id)

Parameter	Data Type	Description
robot	HROBOT	Device ID
id	int	Set desired command number
return	int	Success: 0 Fail: Error code

⚠ The instruction will fail when there is a task executing

4.9.19 Get Motion Command Number

int get_command_id(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: Motion command number Fail: -1

```
C++:
    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);
    }
```

4.9.20 Cancel Unexecuted Motion Command

int remove_command (HROBOT robot ,int num)

Parameter	Data Type	Description
robot	HROBOT	Device ID
num	int	Desired number of command preserved
return	int	Success: 0 Fail: Error code

C++:

```
for (size_t i = 0; i < 1000; i++)
{
        ptp_axis(device_id, 1, joint);
}
Sleep(100);
std::cout << get_command_count(device_id) << std::endl;
remove_command (device_id, 20);
Sleep(1);
std::cout << get_command_count(device_id) << std::endl;
```

4.9.21 Cancel Unexecuted Motion Command from Tail

int remove_command_tail (HROBOT robot ,int num)

Parameter	Data Type	Description
robot	HROBOT	Device ID
num	int	Desired number of data cancelled(cancelled from the last sent command)
return	int	Success: 0 Fail: Error code

C++:

```
for (size_t i = 0; i < 1000; i++)
{
        ptp_axis(device_id, 1, joint);
}
Sleep(100);
std::cout << get_command_count(device_id) << std::endl;
remove_command_tail(device_id, 20);
Sleep(1);
std::cout << get_command_count(device_id) << std::endl;
```

4.9.22 Get Number of Motion Command Queue

int get_command_count(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: Number of motion command Fail: -1

4.9.23 Get Current Motion State

int get_motion_state(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: Current motion state 1:Idle state 2:Motion state 3:Hold state 4:Delay state 5:Command waiting state Fail: -1

⚠ Idle state: No motion command

⚠ Command waiting state: Has motion command but unexecuted

4.10 Manipulator Information Command

4.10.1 Get Current Encoder Value

int get_encoder_count(HROBOT robot ,INT32*   value)

Parameter	Data Type	Description
robot	HROBOT	Device ID
value	double[6]	Each axis encoder value
return	int	Success: 0 Fail: -1

4.10.2 Get Current Joint Coordinate

int get_current_joint(HROBOT robot ,double* coor)

Parameter	Data Type	Description
robot	HROBOT	Device ID
coor	double[6]	Stored in joint coordinate array
return	int	Success: 0 Fail: -1

C++:

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

4.10.3 Get Current Absolute Coordinate

int get_current_position(HROBOT robot ,double* coor)

Parameter	Data Type	Description
robot	HROBOT	Device ID
coor	double[6]	Stored in absolute coordinate array
return	int	Success: 0 Fail: -1

C++:

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

4.10.4 Get Current RPM

int get_current_rpm(HROBOT robot ,double* coor)

Parameter	Data Type	Description
robot	HROBOT	Device ID
coor	double[6]	Each axis motor speed
return	int	Success: 0 Fail: -1

C++:

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

4.10.5 Get Device Manufacture Date

int get_device_born_date (HROBOT robot ,int *YMD)

Parameter	Data Type	Description
robot	HROBOT	Device ID
YMD	int[3]	YMD[0] : Manufacture year YMD[1] : Manufacture month YMD[2] : Manufacture day
return	int	Success: 0 Fail: -1

4.10.6 Get Controller Operation Time

int get_operation_time(HROBOT robot ,int *YMDHm)

Parameter	Data Type	Description
robot	HROBOT	Device ID
YMDHm	int[5]	YMDHm[0] : Operation year YMDHm[1] : Operation month YMDHm[2] : Operation day YMDHm[3] : Operation hour YMDHm[4] : Operation minute
return	int	Success: 0 Fail: -1

4.10.7 Get Mileage of Each Axis Motor

int get_mileage(HROBOT robot ,double *mil)

Parameter	Data Type	Description
robot	HROBOT	Device ID
mil	double[6]	mil[0]: 1st axis motor rotation number mil[1]: 2nd axis motor rotation number mil[2]: 3rd axis motor rotation number mil[3]: 4th axis motor rotation number mil[4]: 5th axis motor rotation number mil[5]: 6th axis motor rotation number
return	int	Success: 0 Fail: -1

⚠ Motor rotational speed can be cleared using HRSS.

4.10.8 Get Cumulative Mileage of Each Axis Motor

int get_total_mileage(HROBOT robot ,double *mil)

Parameter	Data Type	Description
robot	HROBOT	Device ID
mil	double[6]	mil[0]: 1st axis motor cumulative rotation number mil[1]: 2nd axis motor cumulative rotation number mil[2]: 3rd axis motor cumulative rotation number mil[3]: 4th axis motor cumulative rotation number mil[4]: 5th axis motor cumulative rotation number mil[5]: 6th axis motor cumulative rotation number
return	int	Success: 0 Fail: -1

⚠ Motor cumulative rotational speed cannot be cleared using HRSS.

4.10.9 Get Cumulative Utilization Rate

int get_utilization (HROBOT robot ,int *ult)

Parameter	Data Type	Description
robot	HROBOT	Device ID
ult	double[6]	ult [0]: Utilization rate (annual) ult [1]: Utilization rate (month) ult [2]: Utilization rate (day) ult [3]: Utilization rate (hour) ult [4]: Utilization rate (minute) ult [5]: Utilization rate (second)
return	int	Success: 0 Fail: -1

4.10.10 Get Percentage of Utilization

int get_utilization_ratio(HROBOT robot)

Parameter	Data Type	Description
robot	HROBOT	Device ID
return	int	Success: Percentage of utilization Fail: -1

4.10.11 Get Percentage of Motor Load

int get_motor_torque(HROBOT robot ,double *cur)

Parameter	Data Type	Description
robot	HROBOT	Device ID
cur	double[6]	cur [0]: 1staxis torque percentage cur [1]: 2ndaxis torque percentage cur [2]: 3rdaxis torque percentage cur [3]: 4thaxis torque percentage cur [4]: 5thaxis torque percentage cur [5]: 6thaxis torque percentage
return	int	Success: 0 Fail: -1

4.10.12 Get HRSS Version Number

int get_HRSS_version (HROBOT robot ,char *version)

Parameter	Data Type	Description
robot	HROBOT	Device ID
version	char	HRSS version number
return	int	Success: 0 Fail: -1

C++:

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

4.10.13 Get Robot Model Number

int get_HRSS_version_v2 (HROBOT robot ,const char*& ver)

Parameter	Data Type	Description
robot	HROBOT	Device ID
ver	const char*&	HRSS version
return	int	Success: 0 Fail: -1

```
C++:
        const char* hrss_version =NULL;
        get_HRSS_version_v2(s, hrss_version);
        std::cout << "HRSS version:" << hrss_version << std::endl;
```

4.10.14 Get Robot Model Number

int get_robot_type (HROBOT robot ,char *robType)

Parameter	Data Type	Description
robot	HROBOT	Device ID
robType	char	Robot model number
return	int	Success: 0 Fail: -1

```
C++:
    char* version;
    get_robot_type(s, v);
    std::cout << "ROBOT TYPE:\t" << v << std::endl;
```

4.10.15 Get Robot Model Number (No memory recycling required)

int get_robot_type_v2 (HROBOT robot , const char*& robType)

Parameter	Data Type	Description
robot	HROBOT	Device ID
robType	const char*&	Robot model number
return	int	Success: 0 Fail: -1

```
C++:
    const char* type=NULL;
    get_robot_type_v2(s, type);
    std::cout << "ROBOT TYPE:\t" << type << std::endl;
```

4.11 System File Command

4.11.1 Update HRSS:

Specified the file path, it will send the update file to the controller for update. After the transfer is completed and updated, the status will be returned by callback_function.

int update_hrss(HROBOT robot ,char *filePath)

Parameter	Data Type	Description
robot	HROBOT	Device ID
filePath	char*	HRSS update file path and name
return	int	Success: 0 Fail: -1 Update file file error:4020 File transfer failed:4021 Update file unzip failed:4022 Insufficient controller space:4023 The specified path update file does not exist:4024

C++:

```
HROBOT s;
char path[255] = "C:/ HRSS 3.2.12.3940_update.exe";

update_hrss(s, path);
```

4.11.2 Transfer hrb file

int send_file(HROBOT robot ,char * root_folder ,char * filename,int opt)

Parameter	Data Type	Description
robot	HROBOT	Device ID
root_folder	char*	The hrb file is located at the root directory path of the local end (equivalent to the Program folder on the controller side)
filename	char*	The file path of the hrb file is in this root directory.
opt	int	Write to file mode. 0: If there is already a file with the same name on the controller side, this file will be overwritten. 1: If there is already a file with the same name on the controller side, the content will be added to the end of the original content. If the controller does not have this file, a new file will be created regardless of the mode selected.
return	int	Success: 0 Fail: -1

C++:

```
HROBOT s;
HRobot.send_file(s , "HRL_files\\", "folder1\\a123.hrb", 0);
```

4.12 Error Code

Code	Summary	Description
0000	Normal	Command completed normally
0100	Unauthorized	Authorization failure, please contact customer service staff
2000	Unable to execute	Unable to execute requested command
2004	Parameter error	Command parameter error
2005	Abnormal command execution	Abnormal command execution occur
2006	Unable to accept command	Execution of commands is not accepted depending on system status
4000	Mode prohibited	Current mode does not accept execution of commands
4001	Servo prohibited	Command cannot be executed in a non-excited state
4003	Motion register prohibited	When number of motion register reached 1000, requested command could not be executed.
4010	Abnormal file execution	RSR/PNS task setting abnormal
4011		RSR/PNS task execution failed
4012		Task name has or lost
4013		Task execution in progress
9999	Abnormal function	The function is abnormal

HIWIN Robot Software Development Kit User Manual

Publication Date：October 2018, first edition

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 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.
4. HIWIN website for patented product directory: http://www.hiwin.tw/Products/Products_patents.aspx

Subsidiaries / Research Center

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

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

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

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

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

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

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

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

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

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

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

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