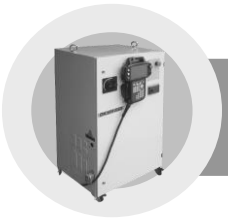
警告

应该由合格的安装人员进行安装、并且安装要符合所有国家法规和地方法规。



Hi5 控制器功能说明书

联机跟踪
(On-Line Tracking)

現代重工業

韩国语印刷 – 2012 年 8 月、第 1 版

地址:北京市丰台区卢沟桥南里 2 号
电话:010-83212588
传真:010-83212188
电子邮箱:robot_as@yahoo.com.cn
主页: http://www.hyundai-bj.com

現代重工業

目 录

图示目录

表格目录

1
概要

1.1. 序论

联机跟踪功能是采用 UDP/IP 通信在机器人控制器上同步反映用户需要的机器人指令、从而进行任意机器人动作的功能。用户利用外部电脑直接计划和新建机器人的位置增量命令后发送到机器人控制器上、机器人控制器在收到位置增量命令后反映到移动计划的同时把机器人当前的直角坐标位置发送到外部电脑以进行反馈。
使用该功能、用户在必要时可在外部电脑安装传感器以计划和反映特定作业用的任意 Motion、但外部 PC 的传感器 Interface 或 UDP/IP 通信则由用户按照用途进行设置。且机器人控制器在每 5msec 执行 UDP/IP 收发、在 5msec 内反映移动位置增量命令、外部 PC 应准确地在每 5msec 内计划 Motion、执行 UDP/IP 通信才能确保机器人的动作流畅。
UDP/IP 通信时机器人控制器起到 client、外部 PC 起到 server 的作用。用户需要时、用 C 语言编程的适用到外部 PC 上的 UDP/IP 通信程序在 3.2 节有相关说明、可参考、相关问题请咨询本公司 A/S 中心。

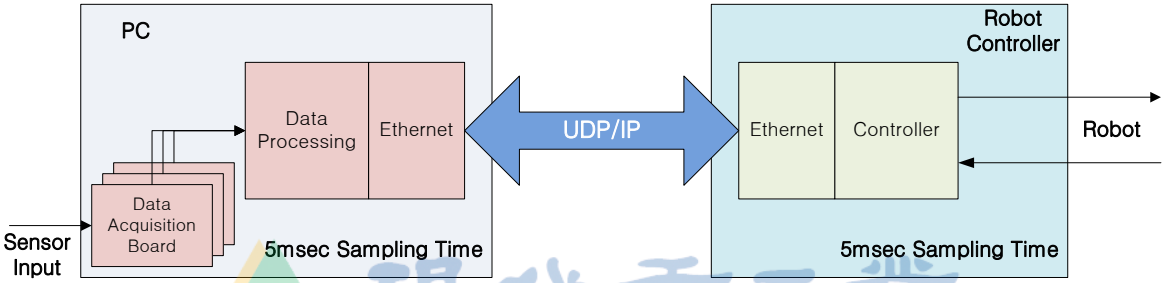


图 1.1 联机跟踪功能的系统组成例示

1.2. 功能概述

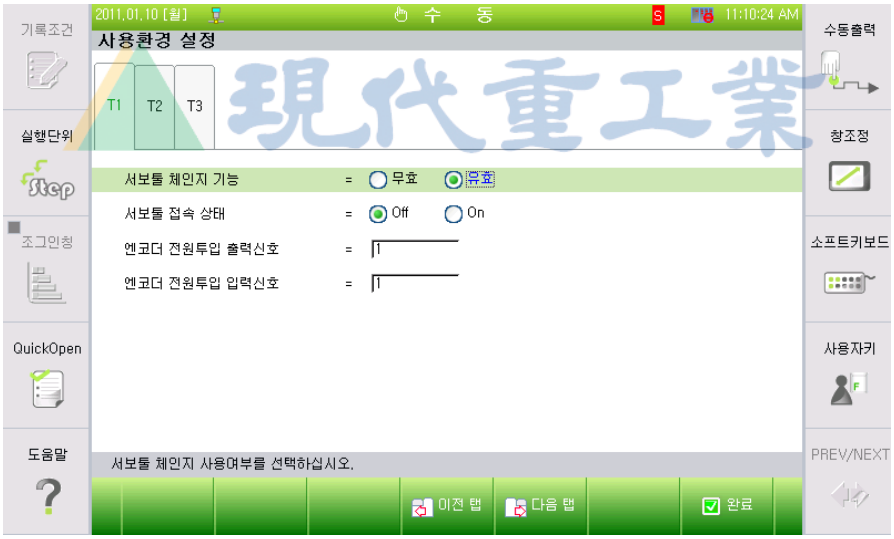
- 通信协议 : UDP/IP (64Bytes)
  - 机器人控制器 → PC : 机器人的 当前位置
  - PC → 机器人控制器 : 机器人的 增量命令
- 增量命令反映周期 : 5msec (200Hz)
- 支持适用增量命令 Filter
- 支持动作领域、速度限制设置功能

2
使用方法

2.1. 联机跟踪程序结构

为使用联机跟踪功能、通过 JOB 文件的 OnLTrack 命令激活本功能后对通信及位置增量命令 Filter 进行设置、必要时以 LIMIT 命令设置机器人的动作领域、速度限制项。最后采用 OnLTrack 命令关闭联机跟踪功能以退出本功能。

区分	说明	程序例示
功能开始、通信及 Filter 设置	启用联机跟踪功能的同时设置 UDP/IP 通信和增量命令的 Filter。	OnLTrack ON,IP=192.168.1.254, PORT=7127,CRD=1,Bypass,Fn=10
动作领域、速度限制的设置	启用联机跟踪功能时对机器人的动作领域和动作速度限制进行设置。 未进行设置时适用联机跟踪功能的 default 值。(参照下面的 LIMIT 指令)	LIMIT POS,+X=500,-X=500, +Y=500,-Y=500,+Z=500,-Z=500 LIMIT VEL,X=100,Y=100,Z=100, RX=50,RY=50,RZ=50
功能结束	关闭联机跟踪功能。	OnLTrack OFF



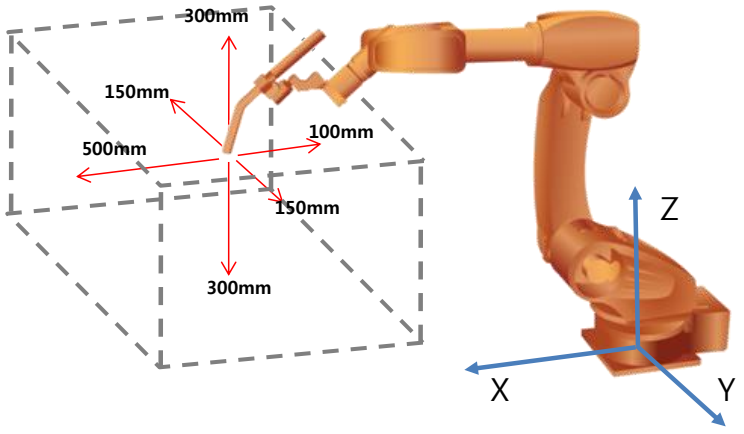
2.2. 联机跟踪相关的指令

联机跟踪功能使用的指令是 OnLTrack 和 LIMIT。
联机跟踪的激活/关闭和通信及 Filter 设置使用 OnLTrack 命令、动作领域和速度限制使用 LIMIT 命令。各指令相关说明如下。

2.2.1. OnLTrack 指令

说明	通过 UDP/IP 用以太网输入位置增量命令时、对此进行反映以启动机器人。	
输入方法	『[F6]: 输入命令』 → 『[F1]: Motion、I/O』 → 『PREV/NEXT』 → 『PREV/NEXT』 →『[F4]: OnLTrack』	
语法	OnLTrack <ON/OFF>、IP=<IP 地址>、PORT=<端口号>、CRD=<基准坐标系>、[<用户坐标系号]、[Bypass]、[Fn=<频率>]	
参数	ON/OFF	ON： 有效、OFF： 无效
	IP 地址	以太网通信用的 PC 的 IP 地址
	端口号	以太网通信用的 PC 端口号
	基准坐标系	计算公式。反映增量命令以启动机器人的基准坐标系 (0=Base、1=机器人、2=工具、3=U、4=Un).
	用户坐标系号	计算公式。基准坐标系为 U、Un 时的用户坐标系号
	Bypass	指令 Filter 的通过与否(ON=未通过、OFF=通过).、 设置为通过时适用控制器内的固有 Filter。 设置为未通过时适用其他 Filter。
	频率	Filter 未通过(Bypass ON)时适用的其他 Filter 的 cut-off 频率。
应用例	OnLTrack ON、IP=192.168.1.254、PORT=7127、CRD=1、Bypass、Fn=10 'OnLTrack 功能动作、以机器人坐标系动作、适用 cut-off 频率为 10Hz 的其他 Filter OnLTrack OFF 'OnLTrack 功能结束	
参考事项	▪ 控制器可在 OnLTrack ON 和 OnLTrack OFF 之间从 PC 接收位置增量命令。 ▪ 启用 OnLTrack 时、通过 Filter 设置、在控制器内对位置增量命令进行 Filtering 以反映到机器人动作。在这种情况下、以 Filtering 反映增量命令时有可能出现延时。 ▪ 启用 OnLTrack 时 'LIMIT POS,+X=300,-X=300,+Y=300,-Y=300,+Z=300,-Z=300'和 'LIMIT VEL,X=200,Y=200,Z=200,RX=100,RY=100,RZ=100'自动进行设置。	

2.2.2. LIMIT 指令

<table>
<tr><td>说明</td><td colspan="2">通过 OnLTrack 功能反映位置增量命令以启动机器人时、对最大动作领域和最大速度进行设置的功能。</td></tr>
<tr><td>输入方法</td><td colspan="2">『[F6]: 输入命令』 → 『[F1]: Motion、I/O』 → 『PREV/NEXT』→ 『PREV/NEXT』 → 『[F7]: LIMIT』</td></tr>
<tr><td>语法</td><td colspan="2">LIMIT POS、[+X=<+X 距离>]、[-X=<-X 距离>]、[+Y=<+Y 距离>]、[-Y=<-Y 距离>]、[+Z=<+Z 距离>]、[-Z=<-Z 距离>]
LIMIT VEL、[X=<X 速度>]、[Y=<Y 速度>]、[Z=<Z 速度>]、[RX=<RX 速度>]、[RY=<RY 速度>]、[RZ=<RZ 速度>]</td></tr>
<tr><td rowspan="3">参数</td><td>POS/VEL</td><td>限制项. POS : 距离、VEL : 速度</td></tr>
<tr><td>限制距离</td><td>计算公式。通过反映位置增量命令启动机器人时、以当前位置基准对机器人可移动的机器人坐标系上的最大领域进行设置。</td></tr>
<tr><td>限制速度</td><td>计算公式。通过反映位置增量命令启动机器人时、对机器人动作时机器人坐标系上的最大速度进行设置</td></tr>
<tr><td>应用例</td><td colspan="2">OnLTrack ON,IP=192.168.1.254,PORT=7127,CRD=1,Bypass,Fn=10
LIMIT POS,+X=500,-X=200,+Y=100,-Y=100,+Z=300,-Z=300
LIMIT VEL,X=200,Y=200,Z=200,RX=100,RY=100,RZ=100
WAIT DI10
LIMIT POS,+X=100,-X=100,+Y=100,-Y=100,+Z=100,-Z=100
LIMIT VEL,X=100,Y=100,Z=100,RX=50,RY=50,RZ=50
DELAY 10.0
OnLTrack OFF</td></tr>
<tr><td>参考事项</td><td colspan="2">▪ LIMIT POS 是以机器人的当前位置为基准设置最大动作领域。超出动作领域的位置增量命令将被忽略。
例) LIMIT POS,+X=500,-X=100,+Y=150,-Y=150,+Z=300,-Z=300
 

▪ 启用 OnLTrack 时自动设置'LIMIT POS,+X=300,-X=300,+Y=300,-Y=300,+Z=300,-Z=300'和'LIMIT VEL,X=200,Y=200,Z=200,RX=100,RY=100,RZ=100'。
▪ 启用 OnLTrack 前 LIMIT 命令将被忽略、如需另行设置、应在启用 OnLTrack 后进行设置。
▪ 启用 OnLTrack 后可多次使用 LIMIT 命令、但 LIMIT POS 在执行时应留意以机器人的当前位置为基准重新设置动作领域。</td></tr>
</table>

2.3. 机器人控制器-PC 之间的通信数据结构及通信方法

2.3.1. 通信方法及数据结构

启用联机跟踪功能时以 PC 为 server、机器人控制器为 client 进行 UDP/IP 通信、通信周期是 5msec。图 2.3 是以 C 语言表示采用 UDP/IP 通信进行 PC 与机器人控制器之间收发数据的结构。发送和接收的数据大小均为 64Bytes。(char 型 1byte、int 型 4bytes、double 型 8bytes)

```
struct{
        char    Command;
        char    char_dummy[3];
        int     State;
        int     Count;
        int     int_dummy;
        double  dData[6];
}
```

图 2.1 UDP/IP 通信的收发数据结构

图 2.3 的 Command 是表示 PC 和机器人控制器之间的连接开始、连接结束、位置增量命令和机器人位置传输的变量。为了 PC 和机器人控制器之间正确的通信连接和数据收发应按照表 2.1 所示、合理设置和传输 Command 值。联机跟踪功能的通信启动顺序请参见 2.3.2 节。

表 2.1 收发数据时 Command 变量内容

变量名		数据传输方向	
		机器人控制器 → PC	PC → 机器人控制器
Command	'S'	- 启用 OnLTrack 发送数据 - 通知联机跟踪功能开始	- 从控制器接收'S' 命令后把 'S'返送到控制器以通知连接 (未把'S'发送到控制器时控制器会忽略之后发送的数据。)
	'P'	- 接到位置增量命令时发送数据 (参照 State 变量) - 一同发送机器人的当前位置 (参照 dData 变量)	- 用来发送位置增量命令 (参照 dData 变量)
	'F'	- 关闭 OnLTrack 时发送数据 - 通知联机跟踪功能结束	- 在关闭控制器的 OnLTrack 之前通过 PC 结束联机跟踪功能时发送数据 - 通知联机跟踪功能结束

图 2.3 的 State 变量在从机器人控制器向 PC 发送数据时才有意义。State=1 表示开始连接、State=3 表示连接结束。因此 Command='S'时 State=1、Command='F'时 State=3。Command='P'时 State 为 -1 或 2。2 表示位置增量命令已反映到机器人 Motion 计划、-1 表示反映失败。(在机器人的异常位置(singular point)机器人的动作存在限制、有可能无法反映位置增量命令。)State 变量内容见表 2.2。

表 2.2 收发数据时 State 变量内容

变量名		数据传输方向	
		机器人控制器 → PC	PC → 机器人控制器
State	1	- 启用 OnLTrack 时发送数据 - 通知联机跟踪功能开始	- 不使用 (reserved)
	2	- 接收、反映位置增量命令时	
	-1	- 虽然接收了位置增量命令但未能反映时	
	3	- 关闭 OnLTrack 时发送数据 - 通知联机跟踪功能结束	

图 2.3 的 Count 是从 PC 向机器人控制器发送位置增量命令时显示是第几个位置增量命令的数字、是由用户设置的变量。从控制器向 PC 发送机器人的当前位置时、会如实发送之前从 PC 接收到的 Count 值。因此用户通过 Count 变量、每次发送数据增加一个 Count、同时与从控制器接收数据时发送 Count 的值进行比较、即可确认之前的数据是否发送完毕。Count 变量内容同表 2.3 所示。

表 2.3 收发数据时 Count 变量内容

变量名	数据传输方向	
	机器人控制器 → PC	PC → 机器人控制器
Count	- 把从 PC 接收的值和当前机器人位置一同返送到 PC	- 位置增量命令发送次数 - 在每次发送时用户可从 0 开始增加 1 次、以确认之前控制器的数据接收情况。

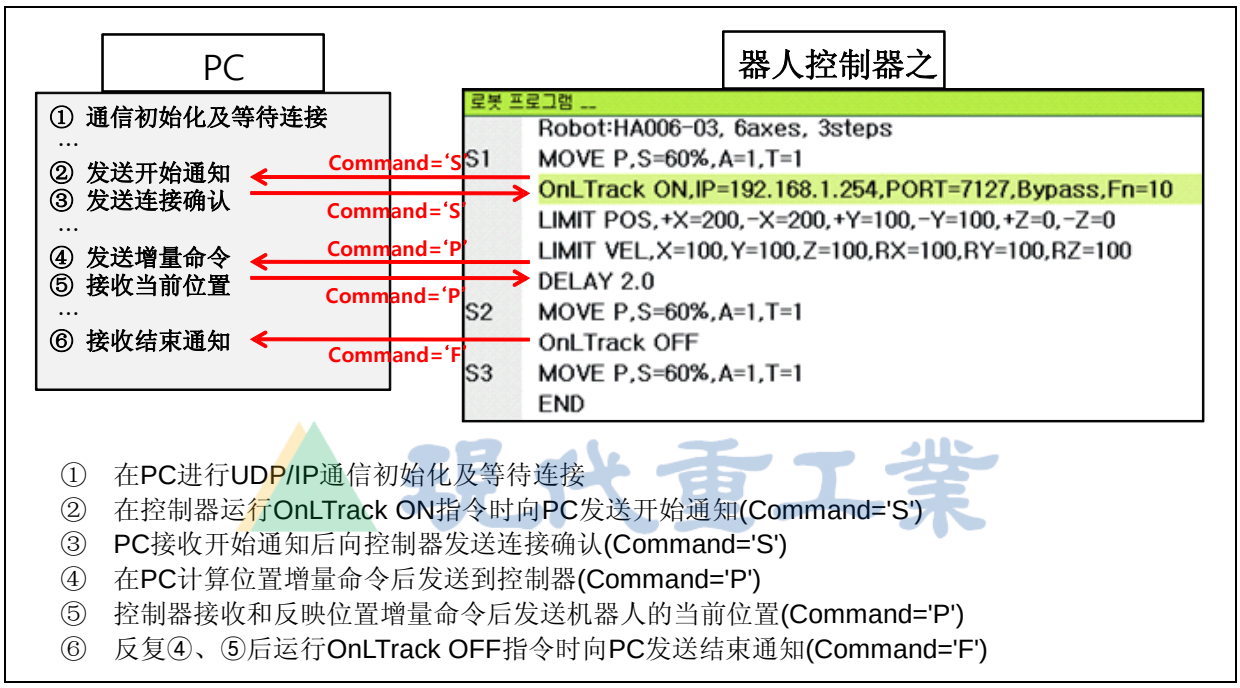
图 2.3 的 dData 根据发送方向具有不同的数据。从 PC 发送到机器人控制器时是指在 5msec 内机器人移动的位置增量命令、从机器人控制器发送到 PC 时是指机器人的当前直角坐标位置值。为了机器人的动作顺滑、应考虑加减速设置为每 5msec 实时发送位置增量命令。运行控制器的 OnLTrack ON 指令时按照 Filter 设置、位置增量命令被 Filtering 从而体现得非常柔和、但有可能发生延时。且位置增量命令会受到 LIMIT POS 和 LIMIT VEL 指令限制、需参看 2.2.2 节的命令文说明。机器人控制器发送到 PC 的机器人的当前直角坐标位置是指按照接收到的位置增量命令反映的机器人坐标系为基准的工具位置值。机器人的当前直角坐标位置值与可在 TP 上监控的机器人的直角坐标位置值相同。dData 变量内容如表 2.4。

表 2.4 收发数据时 dData 变量内容

变量名	数据传输方向	
	PC → 机器人控制器	机器人控制器 → PC
dData[0]~[5]	- 5msec 内机器人可移动的位置增量命令 (增量命令坐标系因 OnLTrack ON 命令文的设置而不同) dData[0] : X 方向 增量命令、m 单位 dData[1] : Y 方向 增量命令、m 单位 dData[2] : Z 方向 增量命令、m 单位 dData[3] : Rx 方向 增量命令、rad 单位 dData[4] : Ry 方向 增量命令、rad 单位 dData[5] : Rz 方向 增量命令、rad 单位	- 反映位置增量命令的机器人坐标系的当前工具位置值 dData[0] : X 方向 当前位置、m 单位 dData[1] : Y 方向 当前位置、m 单位 dData[2] : Z 方向 当前位置、m 单位 dData[3] : Rx 方向 当前位置、rad 单位 dData[4] : Ry 方向 当前位置、rad 单位 dData[5] : Rz 方向 当前位置、rad 单位

2.3.2. 通信方法

以 PC 为 server、机器人控制器为 client 进行 UDP/IP 通信、通信周期是 5msec。控制器的主板提供 3 个网络端口(EN0、EN1、EN2)、但 PC 要与在 EN2 设置的 IP 地址和端口号=6001 进行通信。(有关 EN 2 的 IP 地址等的设置请参照'Hi5 控制器操作说明书'的网络篇) PC 的 IP 地址和端口号应与 OnLTrack O N 命令上设置的值相同、PC 和控制器应交叉连接以太网线。为进行 PC 和控制器的数据收发应按照通信顺序收发 Command 变量。图 3.1 是显示 PC 和控制器之间的通信顺序和收发的 Command 变量。



① 在PC进行UDP/IP通信初始化及等待连接
② 在控制器运行OnLTrack ON指令时向PC发送开始通知(Command='S')
③ PC接收开始通知后向控制器发送连接确认(Command='S')
④ 在PC计算位置增量命令后发送到控制器(Command='P')
⑤ 控制器接收和反映位置增量命令后发送机器人的当前位置(Command='P')
⑥ 反复④、⑤后运行OnLTrack OFF指令时向PC发送结束通知(Command='F')

图 2.1 PC 和机器人控制器之间的通信顺序

以 PC 为 server、机器人控制器为 Client 进行 UDP/IP 通信。Client 即机器人控制器如图 3.1 所示、在运行机器人程序后根据 OnLTrack ON、OFF 指令自动与 server 即 PC 形成通信或结束通信、从而启动联机跟踪功能。且 PC 和机器人控制器之间的通信周期为 5msec、如果控制器在 5msec 内接收 2 个以上的数据时只适用最近接收的数据、会忽略之前接收到的数据、因此 PC 发送数据时应多加留意。

3
其他

3.1. 使用联机跟踪功能时的机器人动作

```
로봇 프로그램 --
     Robot:HA006-03, 6axes, 3steps
S1   MOVE P,S=60%,A=1,T=1
     OnLTrack ON,IP=192.168.1.254,PORT=7127,Bypass,Fn=10
     LIMIT POS,+X=200,-X=200,+Y=100,-Y=100,+Z=0,-Z=0
     LIMIT VEL,X=100,Y=100,Z=100,RX=100,RY=100,RZ=100
     DELAY 2.0
S2   MOVE L,S=60%,A=1,T=1
     OnLTrack OFF
S3   MOVE L,S=60%,A=1,T=1
     END
```

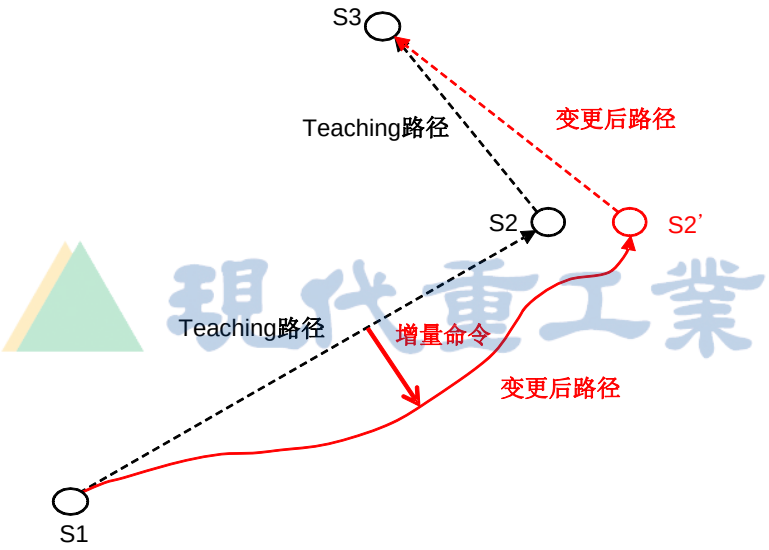


图 3.1 使用联机跟踪功能时的机器人路径变更

图 3.1 显示通过联机跟踪功能、机器人路径出现何种变化。使用联机跟踪功能时、达不到启用本功能的 OnLTrack ON 和 OnLTrack OFF 命令之间的 Step S2 地点、而达到任意一个 S2' 地点。位置增量命令反映到机器人的动作后加以修改、因此会超出计划的 Teaching 路径。但关闭 OnLTrack 后如图 3.1 所示、从 S2'位置出发达到 Step S3 位置。使用联机跟踪功能时 Teaching 路径由于增量命令可变更为任意路径、需加以留意。

3.2. 联机跟踪功能用 UDP/IP 程序例示

下面是使用联机跟踪功能时、在 PC 启用的 UDP/IP 通信程序例示、用键盘输入体现机器人动作程序 C 语言、程序相关问题请联系本公司 A/S 中心。

```
#include <tchar.h>
#include <winsock2.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>

#pragma comment(lib, "ws2_32.lib")

#define _PI          3.141592
#define Hi5_Ts      0.005

typedef struct{

    char     Command;
    char     char_dummy[3];
    int      State;
    int      Count;
    int      int_dummy;
    double dData[6];

} RECEIVE_INTERFACE;

typedef struct{

    char     Command;
    char     char_dummy[3];
    int      State;
    int      Count;
    int      int_dummy;
    double dData[6];

} SEND_INTERFACE;

unsigned long __stdcall Thread1( LPVOID lpParam );
void Init_Command_Data();
void Init_UDP(const char *PC_IP, unsigned short PC_Port, const char *ROBOT_IP, unsigned short
ROBOT_Port);

WSADATA wsaData;
SOCKET PC_Socket;
SOCKADDR_IN PC_Address, Hi5_Address;
int PC_AddressSize = sizeof(PC_Address);
char *IP_Hi5 = new char [16];
char *IP_PC = new char [16];
unsigned short Port_Hi5;
unsigned short Port_PC;
```

```
RECEIVE_INTERFACE     *pRECEIVE = new RECEIVE_INTERFACE;
SEND_INTERFACE     *pSEND = new SEND_INTERFACE;

int _tmain(int argc, _TCHAR* argv[])
{
    int     i, ReturnVal, ch;
    double       DeltaPosCmd[6];

    IP_Hi5="127.0.0.1";          // IP address of robot controller (Hi5 controller)
    Port_Hi5 = 6001;          // port number of robot controller (Hi5 controller)
    IP_PC="127.0.0.1";          // IP address of PC
    Port_PC = 7127;               // port number of PC

    Init_UDP(IP_PC, Port_PC, IP_Hi5, Port_Hi5);          // UDP/IP is initialized

    HANDLE     hThread1;
    DWORD     dwThreadID1;
    hThread1 = CreateThread( NULL, 0, Thread1, 0, 0, &dwThreadID1 );

    do
    {
        for(i=0; i<6; i++)
        {
            DeltaPosCmd[i] = 0.0;
        }
        ch = _getch();

        switch(ch)
        {
            // calculate incremental position command (DeltaPosCmd)
            case 'u':
            case 'U':
                DeltaPosCmd[0] = 150.0*Hi5_Ts;              // 150mm/sec * 0.005sec
                break;
            case 'j':
            case 'J':
                DeltaPosCmd[0] = -150.0*Hi5_Ts;
                break;
            case 'i':
            case 'I':
                DeltaPosCmd[1] = 150.0*Hi5_Ts;
                break;
            case 'k':
            case 'K':
                DeltaPosCmd[1] = -150.0*Hi5_Ts;
                break;
            case 'o':
            case 'O':
                DeltaPosCmd[2] = 150.0*Hi5_Ts;
                break;
            case 'l':
            case 'L':
                DeltaPosCmd[2] = -150.0*Hi5_Ts;
                break;
            case 'q':
            case 'Q':
```

```
                DeltaPosCmd[3] = 100.0*Hi5_Ts;              // 100deg/sec * 0.005sec
                break;
            case 'a':
            case 'A':
                DeltaPosCmd[3] = -100.0*Hi5_Ts;
                break;
            case 'W':
            case 'w':
                DeltaPosCmd[4] = 100.0*Hi5_Ts;
                break;
            case 's':
            case 'S':
                DeltaPosCmd[4] = -100.0*Hi5_Ts;
                break;
            case 'e':
            case 'E':
                DeltaPosCmd[5] = 100.0*Hi5_Ts;
                break;
            case 'd':
            case 'D':
                DeltaPosCmd[5] = -100.0*Hi5_Ts;
                break;
            default:
                break;
        }

        pSEND->Count++;
        pSEND->State = 2;
        pSEND->Command = 'P';
        // incremental position command must be expressed in terms of meter & radian
        for(i=0; i<3; i++)
        {
            pSEND->dData[i] = DeltaPosCmd[i] * 0.001;               // mm -> m
            pSEND->dData[i+3] = DeltaPosCmd[i+3] * _PI/180.0;      // deg -> rad
        }

        // send the data to Hi5 controller
        ReturnVal = sendto( PC_Socket,
            (char *)pSEND,
            sizeof(SEND_INTERFACE),
            0,
            (struct sockaddr *)&Hi5_Address,
            sizeof(Hi5_Address) );

    } while(ch!='x' && ch!='X');

    Sleep( 500 );
     closesocket( PC_Socket ); //关闭 Socket。
     WSACleanup();

    printf("Program is terminated.\n");

     return 0;
}

unsigned long __stdcall Thread1( LPVOID lpParam    )
```

```
{
    int     ReturnVal;
    bool Start_flag=false;
    WSANETWORKEVENTS event;

    WSAEVENT SockEvent = WSACreateEvent();
    printf( "Waiting for the beginning of On-line tracking by Hi5 controller...\n" );
    WSAEventSelect( PC_Socket, SockEvent, FD_READ );

    while(1)
    {
        WSAEnumNetworkEvents( PC_Socket, SockEvent, &event );
        if((event.INetworkEvents & FD_READ)==FD_READ)
        {
            // receive the data from Hi5 controller
            ReturnVal = recvfrom( PC_Socket,
                (char *)pRECEIVE,
                sizeof(RECEIVE_INTERFACE),
                0, (struct sockaddr *)&PC_Address,
                &PC_AddressSize );

            if(Start_flag==false)
            {
                if(pRECEIVE->Command == 'S')          // Start
                {
                    system( "cls" );
                    printf( "recv>> Command: %c,   Count: %d,   [X: %.3f, Y: %.3f, Z: %.3f, Rx: %.3f,
Ry: %.3f, Rz: %.3f] \n",
                        pRECEIVE->Command,
                        pRECEIVE->Count,
                        pRECEIVE->dData[0]*1000,          // m -> mm
                        pRECEIVE->dData[1]*1000,
                        pRECEIVE->dData[2]*1000,
                        pRECEIVE->dData[3]*180.0/_PI,     // rad -> deg
                        pRECEIVE->dData[4]*180.0/_PI,
                        pRECEIVE->dData[5]*180.0/_PI);

                    Start_flag = true;
                    Init_Command_Data();
                    pSEND->Command = 'S';
                    pSEND->Count = 0;
                    pSEND->State = 1;
                    ReturnVal = sendto( PC_Socket,
                        (char *)pSEND,
                        sizeof(SEND_INTERFACE),
                        0,
                        (struct sockaddr *)&Hi5_Address,
                        sizeof(Hi5_Address) );
                    printf("On-line tracking is started by Hi5 controller.\n");

                }
            }
            else
            {
                switch(pRECEIVE->Command)
                {
```

```
                case 'P':          // Play
                    system( "cls" );
                    printf( "recv>> Command: %c,   Count: %d,   [X: %.3f, Y: %.3f, Z: %.3f, Rx: %.3f,
Ry: %.3f, Rz: %.3f] \n",
                        pRECEIVE->Command,
                        pRECEIVE->Count,
                        pRECEIVE->dData[0]*1000,          // m -> mm
                        pRECEIVE->dData[1]*1000,
                        pRECEIVE->dData[2]*1000,
                        pRECEIVE->dData[3]*180.0/_PI,     // rad -> deg
                        pRECEIVE->dData[4]*180.0/_PI,
                        pRECEIVE->dData[5]*180.0/_PI);
                    break;
                case 'F':          // Finish
                    printf( "On-line tracking is finished by Hi5 controller.\n" );
                    Start_flag = false;
                    break;
                default:
                    break;
                }
            }
        }
    }

    WSACloseEvent( SockEvent );
    closesocket( PC_Socket );
    WSACleanup();
    exit( 0 );

    return 0;
}

void Init_Command_Data()
{
    int i, ReturnVal=0;;

    pSEND->Command = NULL; pSEND->State = 0; pSEND->Count = 0;
    pRECEIVE->Command = NULL; pRECEIVE->State = 0; pRECEIVE->Count = 0;
    for(i=0; i<6; i++)
    {
        pSEND->dData[i] = 0.0;
        pRECEIVE->dData[i] = 0.0;
    }

    return;
}

void Init_UDP(const char *PC_IP, unsigned short PC_Port, const char *ROBOT_IP, unsigned short
ROBOT_Port)
{
    PC_Address.sin_family = AF_INET;
    PC_Address.sin_addr.s_addr = inet_addr( PC_IP );
    PC_Address.sin_port = htons( PC_Port );
     Hi5_Address.sin_family = AF_INET;
    Hi5_Address.sin_addr.s_addr = inet_addr( ROBOT_IP );
    Hi5_Address.sin_port = htons( ROBOT_Port );
```

```
    // initiate use of WS2_32.DLL by a process
    if (WSAStartup(0x202,&wsaData) == SOCKET_ERROR)
    {
        printf( "Trouble occurs in WSAStartup setting.\n" );
        WSACleanup();
        exit( 0 );
    }
    // create PC socket for UDP
  PC_Socket = socket(AF_INET, SOCK_DGRAM,0); //

    if( PC_Socket == INVALID_SOCKET )
    {
        printf( "PC_Socket can't be created.\n" );
        WSACleanup();
        exit( 0 );
    }

  // associate PC address with PC socket
  if( bind(PC_Socket,(struct sockaddr*)&PC_Address,sizeof(PC_Address) ) == SOCKET_ERROR )
    {
        printf( "Socket can't be binded." );
        closesocket( PC_Socket );
        WSACleanup();
        exit( 0 );
    }

  return;
}
```

■ **Head Office**
1、Jeonha-dong、Dong-gu、Ulsan、Korea
TEL : 82-52-230-7901 / FAX : 82-52-230-7900

■ **BEIJING HYUNDAI**
JINGCHENG MACHINERY CO.、LTD.
NO.2NANLI、LUGOUQIAO、 FENGTAI DISTRICT、BEIJING
TEL : 86-010-8321-2588 / FAX : 86-010-8321-2188
E-Mail : robot_as@yahoo.com.cn
POST CODE : 100072

■ **韩国现代重工业本部**
蔚山市东区田下洞 1 番地
TEL : 82-52-230-7901 / FAX : 82-52-230-7900

■ **北京现代京城工程机械有限公司**
北京市丰台区卢沟桥南里 2 号
电话 ： 86-010-8321-2588 / 传真 ： 86-010-8321-2188
电子邮箱 ： robot_as@yahoo.com.cn
邮编 ：100072