



WARNING

**THE INSTALLATION SHALL BE MADE
BY QUALIFIED INSTALLATION
PERSONNEL AND SHOULD CONFORM
TO ALL NATIONAL AND LOCAL CODES**



Hi5 Controller Function Manual

On-Line Tracking





The information presented in the manual is the property of HHI.
Any copy or even partial is not allowed without prior written authorization from HHI.
It may not be provided to the third party, nor used for any other purposes.

HHI reserves the right to modify without prior notification.

Printed in Korea – July. 2012. 1st Edition
Copyright © 2012 by Hyundai Heavy Industries Co., Ltd





Contents

1. Overview 1-1

- 1.1. Introduction 1-2
- 1.2. Summary on Functions 1-2

2. Use Methods 2-1

- 2.1. Structure of Online Tracking Program..... 2-2
- 2.2. Commands Related to Online Tracking..... 2-3
 - 2.2.1. OnLTrack Command 2-3
 - 2.2.2. LIMIT Command 2-4
- 2.3. Communication Data Structure & Methods Between Robot Controller and PC 2-5
 - 2.3.1. Communication Methods and Data Structure 2-5
 - 2.3.2. Communication Methods 2-8

3. Other Matters 3-1

- 3.1. Robot Motion During the Use of Online Tracking..... 3-2
- 3.2. Examples of UDP/IP Program for the Use of Online Tracking 3-3



Contents

Figures

Figure 1.1 Example of System Configuration of Online Tracking Function	1-2
Figure 2.1 Send and Received Data Structure of UDP/IP communications	2-5
Figure 2.2 Communication Order between PC and Robot controller	2-8
Figure 3.1 Robot's Path Change During the Use of Online Tracking	3-2

Tables

Table 2.1 Variables of Command During the Communications	2-5
Table 2.2 Variables of State During the Communications	2-6
Table 2.3 Variable of Count During the Communications	2-6
Table 2.4 Variables of dData During the Communications	2-7





HYUNDAI
HEAVY INDUSTRIES CO., LTD.

1

Overview



1. Overview

1.1. Introduction

The online tracking has a function of applying a user's command to a robot's controller in real time by means of the UDP/IP communications so as to enable any robot motions. If a user plans, creates and transmits an incremental command of a robot position from an external PC directly to a robot controller, the robot receives and applies the incremental command to a motion plan, and at the same time, feeds the robot's current position in the rectangular coordinates back to the external PC.

If using this function, the user may attach any sensors to the external PC if necessary and plan and apply any motions for a specific job. However, in order to realize the external PC's sensor interface or UDP/IP communications, the user shall match it for the use in person. In addition, the robot controller performs the UDP/IP communications exactly every 5 msec and applies incremental commands of positions to motions within 5 msec; therefore, the external PC shall plan motions exactly every 5 msec and perform the UDP/IP communications so as to make the robot move smoothly.

During the UDP/IP communications, the robot serves as a client; the external PC as a server. With respect to the UDP/IP communication program that is realized on the external PC, Chapter 3.2 provides the example of the program built in the C language on the user's request. So, please refer to the example of the program and contact our A/S center to ask relevant questions.

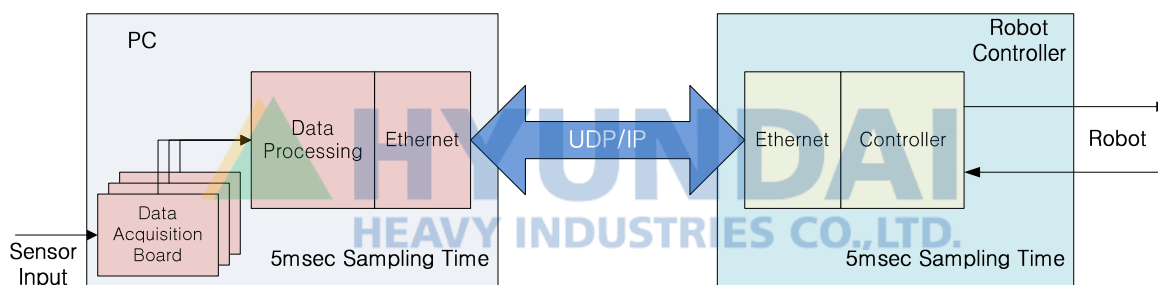


Figure 1.1 Example of System Configuration of Online Tracking Function

1.2. Summary on Functions

- Communication protocol: UDP/IP (64Bytes)
 - Robot controller → PC: robot's current position
 - PC → robot controller: robot's incremental command
- Application cycle of an incremental command: 5msec (200Hz)
- Support for the incremental command filter application
- Support for the function of setting motion fields and speed limits



HYUNDAI
HEAVY INDUSTRIES CO., LTD.

2

**Use
Methods**

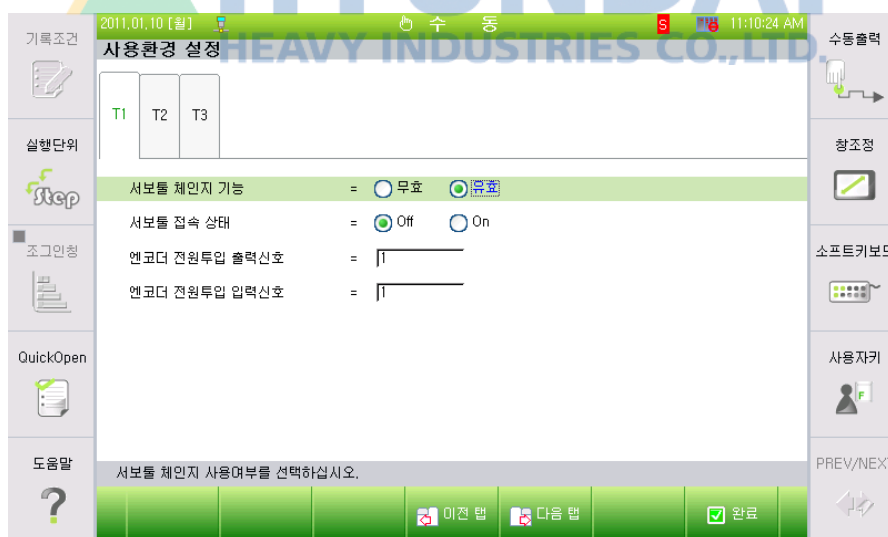


2. Use Methods

2.1. Structure of Online Tracking Program

In order to use the online tracking function, set the communications and the incremental command filter for positions while activating this function by means of the OnLTrack command on the JOB file. And set robot's motion fields and speed limits with the use of the LIMIT command, if necessary. Lastly, Terminate the function by inactivating the online tracking function with.

Division	Description	Examples of the program
Start the function and set the communications and filter	While switching on the online tracking function, set the UDP/IP communications and the incremental command filter.	OnLTrack ON,IP=192.168.1.254,PORT=7127,CRD=1,Bypass,Fn=10
Set motion fields and speed limits	When activating the online tracking function, create fields and speed limits of robot's motions. The default of the online tracking function is applied unless there are any settings. (Refer to the following the LIMIT command.)	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
Terminate the function	Terminate the online tracking function.	OnLTrack OFF



2.2. Commands Related to Online Tracking

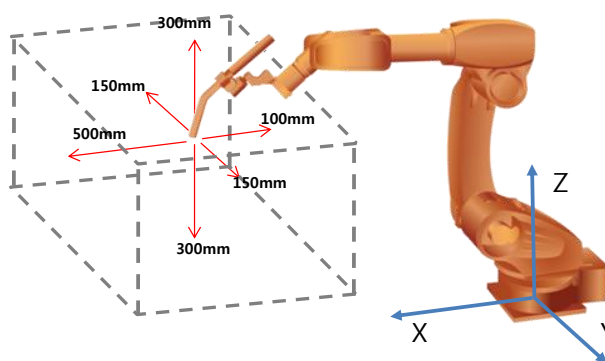
OnLTrack and LIMIT are the commands used for the online tracking. The OnLTrack is responsible for activating/inactivating the online tracking and for setting the filter; the LIMIT is responsible for setting motion fields and speed limits.

Description of each command is as follows.

2.2.1. OnLTrack Command

Description	When an incremental command of a position is input through Ethernet by means of UDP/IP, apply the command to a robot motion.	
Input methods	『[F6]: Input Command』 → 『[F1]: Motion, I/O』 → 『PREV/NEXT』 → 『PREV/NEXT』 → 『[F4]: OnLTrack』	
Sentence rule	OnLTrack <ON/OFF>,IP=<IP address>,PORT=<Port No.>,CRD=<Normal coordinate system>,<No. of user coordinate system>],[Bypass],[Fn=<Frequency>]	
Parameter	ON/OFF	ON : valid, OFF : invalid
	IP Address	PC's IP address for Ethernet communications
	Port No.	PC's port numbers for Ethernet communications
	Reference Coordinate System	Arithmetic expression. It determines a robot motion by applying the incremental command. (0=Base, 1=Robot, 2=Tool, 3=U, 4=Un).
	No. of User coordinate system	Arithmetic expression. When the reference coordinate system is U and Un, it means the numbers of the user coordinate system.
	Bypass	Determine whether to accept the command filter (On=rejected, Off=accepted) When 'OFF' is set, the controller's built-in filter applies. When 'ON' is set, the separate filter applies.
	Frequency	It is a cut-off frequency of the separate filter applied when the filter is rejected (Bypass ON).
Examples of use	OnLTrack ON,IP=192.168.1.254,PORT=7127,CRD=1,Bypass,Fn=10 'Motion by OnLTrack function, Motion by the robot coordinate system, Separate filter with a cut-off frequency of 10 Hz applied OnLTrack OFF 'Terminate the OnLTrack function	
Note	<ul style="list-style-type: none"> ▪ The controller may receive incremental commands of positions between OnLTrack ON and OnLTrack OFF from the PC. ▪ When the OnLTrack ON is run, the controller receives the incremental command by means of the filter settings and applies the command to a robot motion. During the process, a time delay may occur when the incremental command is applied by filtering. ▪ When the OnLTrack ON is run, 'LIMIT POS, +X=300, -X=300, +Y=300, -Y=300, +Z=300, -Z=300' and 'LIMIT VEL, X=200, Y=200, Z=200, RX=100, RY=100, RZ=100' are automatically created. 	

2.2.2. LIMIT Command

Description	It has a function of setting the maximum motion field and speed when a robot moves based on the application of an incremental command of a position by means of OnLTrack.	
Input methods	『[F6]: Input Command』 → 『[F1]: Motion, I/O』 → 『PREV/NEXT』 → 『PREV/NEXT』 → 『[F7]: LIMIT』	
Sentence rule	LIMIT POS,[+X=<+X Distance>],[-X=<-X Distance>],[+Y=<+Y Distance>],[-Y=<-Y Distance>],[+Z=<+Z Distance>],[-Z=<-Z Distance>] LIMIT VEL,[X=<X Speed>],[Y=<Y Speed>],[Z=<Z Speed>],[RX=<RX Speed>],[RY=<RY Speed>],[RZ=<RZ Speed>]	
Parameter	POS/VEL	Limit items. POS : Distance, VEL : Speed
	Limited Distance	Arithmetic expression. By applying the incremental commands of positions, set the maximum field in the robot coordinate system where a robot may move as a standard of the current position when the robot operates.
	Limited speed	Arithmetic expression. By applying the incremental commands of positions, set the maximum speed in the robot coordinate system where a robot may move when the robot operates.
Examples of use	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	
Note	<p>▪ LIMIT POS sets up the maximum motion field based on a robot's current position. The incremental command of the position that is outside the motion field is ignored. Examples) LIMIT POS,+X=500,-X=100,+Y=150,-Y=150,+Z=300,-Z=300</p>  <p>▪ When the OnLTrack ON is run, 'LIMIT POS, +X=300, -X=300, +Y=300, -Y=300, +Z=300, -Z=300' and 'LIMIT VEL, X=200, Y=200, Z=200, RX=100, RY=100, RZ=100' are automatically created.</p> <p>▪ The LIMIT command is ignored before the OnLTrack ON is run; therefore, if the user wants separate settings, the settings shall be undertaken after the OnLTrack ON is executed.</p> <p>▪ After the OnLTrack ON, the LIMIT command may be used several times. However, pay attention to the fact that when the LIMIT POS command is performed, the motion field is reset based on the robot's current position.</p>	

2.3. Communication Data Structure & Methods Between Robot Controller and PC

2.3.1. Communication Methods and Data Structure

In the online tracking function, the UDP/IP communications are performed between a PC as a server and a robots as a client with a communication cycle of 5 msec. Based on the C language, Figure 2.3 shows the structure of data which the PC transmits to and receives from the robot controller through the UDP/IP communications. The data size is 64 Bytes during the transmission and reception process. (char type is 1 byte, int type, 4 bytes and double type, 8 bytes.)

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

Figure 2.1 Send and Received Data Structure of UDP/IP communications

The Command of Figure 2.3 is the variable that shows the start and end of the connection, the incremental command of the position, and the transmission of the robot position between the PC and the robot. To correctly connect the PC to the robot controller and succeed in data communications, set and transmit appropriate command values as shown in the following Table 2.1. Refer to Chapter 2.3.2 with respect to the order of the online tracking communications, please.

Table 2.1 Variables of Command During the Communications

Variable name		Data transmission direction	
		Robot controller→ PC	PC → robot controller
Command	'S'	<ul style="list-style-type: none"> - Transmit 'S' when running OnLTrack ON. - Notify the start of online tracking 	<ul style="list-style-type: none"> - Notify the connection check by feeding the 'S' back after receiving the 'S' command from the controller. (If 'S' is not transmitted back to the controller, the controller ignores any data afterwards.)
	'P'	<ul style="list-style-type: none"> - Send 'P' when receiving an incremental command of a location (refer to the variables of State). - Transmit a robot's current position together. (Refer to the variables of dData.) 	<ul style="list-style-type: none"> - 'P' is used when an incremental command of a location is transmitted. (Refer to the variables of dData.)
	'F'	<ul style="list-style-type: none"> - Transmit 'F' when running OnLTrack OFF. - Notify the end of online tracking. 	<ul style="list-style-type: none"> - Transmit 'F' when the PC terminates the online tracking before the controller's OnLTrack OFF is run. - Notify the end of online tracking.

The variables of State of Figure 2.3 have meanings only when the robot controller transmits data to the PC. 'State=1' means the start of the connection, and 'State=3', the end of the connection. Therefore, when the Command is 'S', the State is 1; when the Command is 'F', the State is 3. When the Command is 'P', the State is -1 or 2. '2' means that an incremental command of a position has been applied to a robot's motion plan; otherwise, '-1' means no application of the incremental command. (At the robot's singular point, the robot's motion is limited, so the incremental command of the position may not apply.) The summary on the variables of State is as seen in Table 2.2.

Table 2.2 Variables of State During the Communications

Variable name		Data transmission direction	
		Robot controller → PC	PC → robot controller
State	1	- Send '1' when running OnLTrack ON. - Notify the start of online tracking.	- N/A (reserved)
	2	- When a incremental command of a position is received and applied	
	-1	-When the incremental command of the position was received but is not applied	
	3	- Send '3' when running OnLTrack OFF. - Notify the end of online tracking.	

The Count of Figure 2.3 is the user-set variable that shows how many incremental commands of positions have been transmitted from the PC to the robot controller. When the controller transmits the robot's current position to the PC, it sends the value of Count that has been received shortly before. Therefore, users increase the Count by one (1) every time they transmit data with use of the variable of Count; then compare the transmitted value of Count with the received Count from the controller; and may check easily whether the data has been transmitted shortly before. The summary on the variable Count is as seen in Table 2.3.

Table 2.3 Variable of Count During the Communications

Variable name		Data transmission direction	
		Robot controller → PC	PC → robot controller
Count		- Retransmit the value that has been received from the PC along with the robot's current position.	- The number of transmitted incremental commands of locations - The user directly increases the value from 0 by one (1) during every transmission and may check whether the controller has received the previous data.

The dData of Figure 2.3 has different data depending on the transmission directions. The data from a PC to a robot controller are the incremental commands of the positions where the robot will move within 5 msec; the data from the robot controller to the PC are the values of the rectangular coordinates showing the robot's current position. It is recommended that the incremental command of the position should be transmitted in real time every 5 msec so that the robot moves smoothly considering its variable speed. When the controller runs the OnLTrack ON command, the incremental commands of the positions applies smoothly because the filter settings filter the incremental commands of positions, but a time delay may occur. In addition, incremental commands of positions are limited to the LIMIT POS and LIMIT VEL commands, so please refer to Chapter 2.2.2 of the command description. The robot's current position in the rectangular coordinates received from the robot controller to the PC means the value of the tool position based on the robot coordinate system that applies the received incremental command of the position. The robot's current position in the rectangular coordinates is the same as that of the value of the robot's rectangular coordinates which may be monitored on TP. The summary on the variables of dData are as seen in Table 2.4.

Table 2.4 Variables of dData During the Communications

Variable name	Data transmission direction																									
	PC → robot controller	Robot controller → PC																								
dData[0]~[5]	<p>- Incremental commands of positions where a robot moves for 5 msec (the coordinate system of the incremental command follows the settings of the OnLTrack ON command)</p> <table><tr><td>dData[0]</td><td>: X-direction incremental command, m unit</td></tr><tr><td>dData[1]</td><td>: Y-direction incremental command, m unit</td></tr><tr><td>dData[2]</td><td>: Z-direction incremental command, m unit</td></tr><tr><td>dData[3]</td><td>: Rx-direction incremental command, rad unit</td></tr><tr><td>dData[4]</td><td>: Ry-direction incremental command, rad unit</td></tr><tr><td>dData[5]</td><td>: Rz-direction incremental command, rad Unit</td></tr></table>	dData[0]	: X-direction incremental command, m unit	dData[1]	: Y-direction incremental command, m unit	dData[2]	: Z-direction incremental command, m unit	dData[3]	: Rx-direction incremental command, rad unit	dData[4]	: Ry-direction incremental command, rad unit	dData[5]	: Rz-direction incremental command, rad Unit	<p>- Values of the current tool positions of the robot coordinate system that applies the incremental commands of positions</p> <table><tr><td>dData[0]</td><td>: X-direction current position, m unit</td></tr><tr><td>dData[1]</td><td>: Y-direction current position, m unit</td></tr><tr><td>dData[2]</td><td>: Z-direction current position, m unit</td></tr><tr><td>dData[3]</td><td>: Rx-direction current position, rad unit</td></tr><tr><td>dData[4]</td><td>: Ry-direction current position, rad unit</td></tr><tr><td>dData[5]</td><td>: Rz-direction current position, rad unit</td></tr></table>	dData[0]	: X-direction current position, m unit	dData[1]	: Y-direction current position, m unit	dData[2]	: Z-direction current position, m unit	dData[3]	: Rx-direction current position, rad unit	dData[4]	: Ry-direction current position, rad unit	dData[5]	: Rz-direction current position, rad unit
	dData[0]	: X-direction incremental command, m unit																								
dData[1]	: Y-direction incremental command, m unit																									
dData[2]	: Z-direction incremental command, m unit																									
dData[3]	: Rx-direction incremental command, rad unit																									
dData[4]	: Ry-direction incremental command, rad unit																									
dData[5]	: Rz-direction incremental command, rad Unit																									
dData[0]	: X-direction current position, m unit																									
dData[1]	: Y-direction current position, m unit																									
dData[2]	: Z-direction current position, m unit																									
dData[3]	: Rx-direction current position, rad unit																									
dData[4]	: Ry-direction current position, rad unit																									
dData[5]	: Rz-direction current position, rad unit																									

2.3.2. Communication Methods

A PC serves as a server and a robot controller serves as a client with the use of the UDP/IP communications and the communication cycle of 5 msec. The controller's main board provides 3 network ports (EN0, EN1, and EN2) but shall communicate with the EN2-set IP address and the port No. 6001 on the PC. (Refer to the network of the 'Hi5 Controller Instruction Manual with respect to the settings of the EN2 IP address.) The PC's IP address and the port number shall be the same as the values set by the OnLTrack ON command, and the PC and the controller shall be connected through the cross Ethernet cable. The Command variables shall be transmitted and received in accordance with the communication order for the communications between the PC and the controller. Figure 3.1 shows the communication order between the PC and the controller and the Command variables that are transmitted and received.

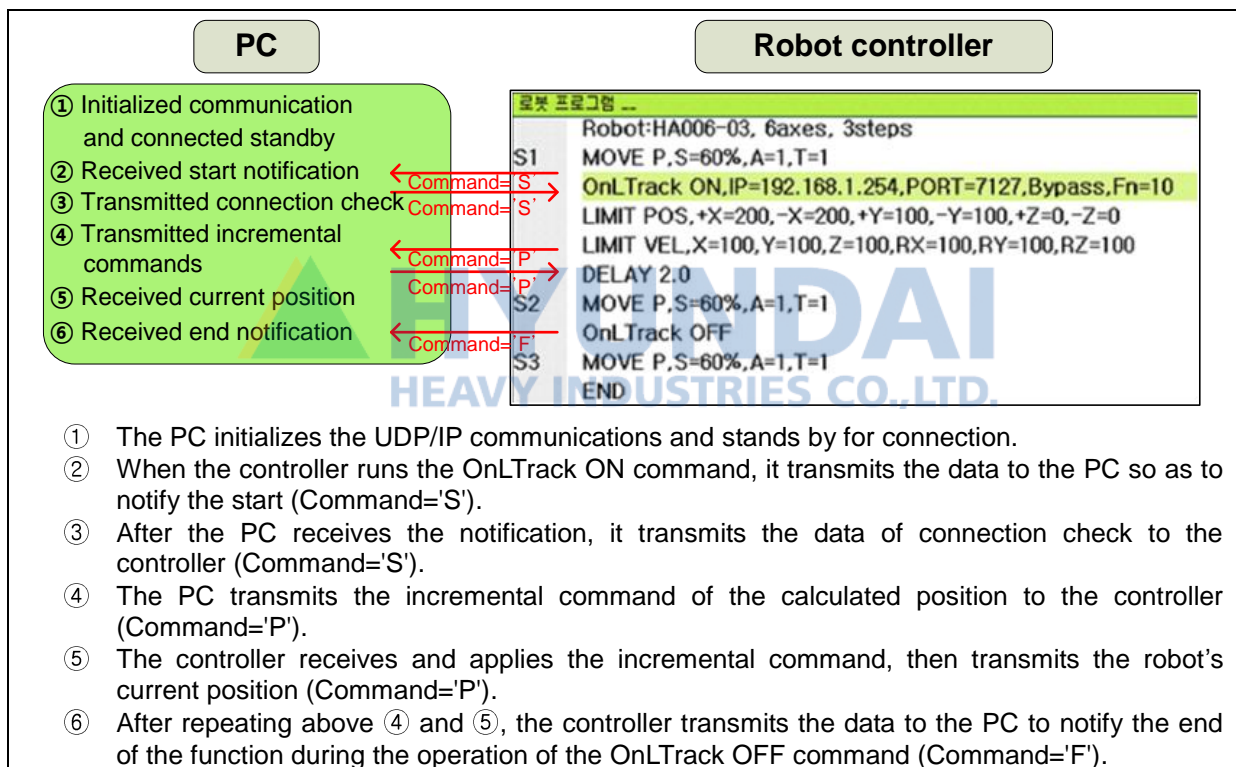


Figure 2.2 Communication Order between PC and Robot controller

A PC shall serve as a server and a robot controller as a client through the UDP/IP communications. When the robot program operates as seen in Figure 3.1, the robot controller, as a client automatically starts and terminates the communications with the server PC in accordance with the OnLTrack ON and OFF commands, which enable the online tracking to function. Furthermore, the communication cycle between the PC and the controller is 5 msec. If 2 and more data are received within 5 msec, the latest received data apply and the previous data are ignored; therefore, be careful when transmitting data from the PC, please.



HYUNDAI
HEAVY INDUSTRIES CO., LTD.

3

**Other
Matters**



3. Other Matters

3.1. Robot Motion During the Use of Online Tracking

```

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
  
```

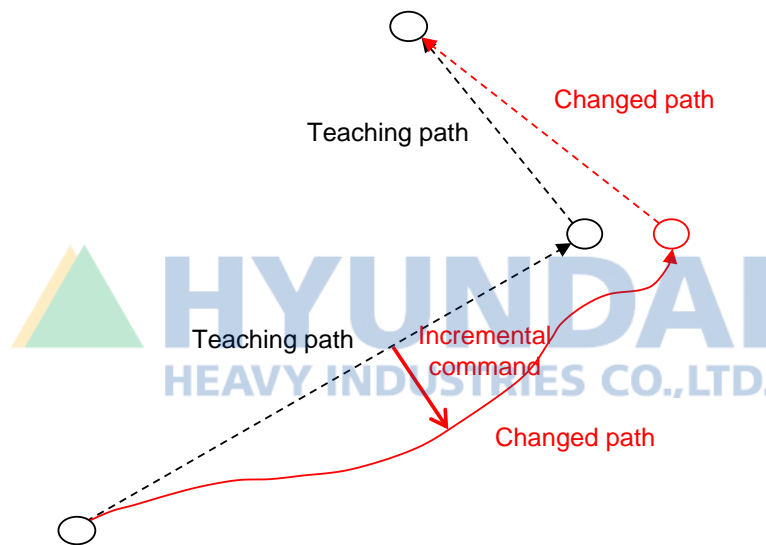


Figure 3.1 Robot's Path Change During the Use of Online Tracking

Figure 3.1 shows how the robot paths change in accordance with the online tracking. With the use of the online tracking, the robot gets to an S2' point, not reaching the step S2 between the OnLTrack ON and OnLTrack OFF commands that activate this function because the incremental command of the position is applied to the robot motion, which makes the robot out of the planned teaching path. However, after the OnLTrack OFF, it starts from the S2' and gets to the step S3 as seen in Figure 3.1. Like this, the use of the online tracking may cause teaching paths to be changed by the incremental command; therefore, be careful, please.

3.2. Examples of UDP/IP Program for the Use of Online Tracking

The following examples are the examples of the UDP/IP communication program which is driven on the PC for the use of the online tracking function. This program is built in the C language and may operate a robot by means of keyboard input. Please contact our A/S center with respect to relevant questions.

```
#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
RetVal = 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 ); // Close the 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.lNetworkEvents & 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( "rcv>> 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**

Tel. 82-52-202-7901 / Fax. 82-52-202-7900
1, Jeonha-dong, Dong-gu, Ulsan, Korea

- **A/S Center**

Tel. 82-52-202-5041 / Fax. 82-52-202-7960

- **Seoul Office**

Tel. 82-2-746-4711 / Fax. 82-2-746-4720
140-2, Gye-dong, Jongno-gu, Seoul, Korea

- **Ansan Office**

Tel. 82-31-409-4945 / Fax. 82-31-409-4946
1431-2, Sa-dong, Sangnok-gu, Ansan-si, Gyeonggi-do, Korea

- **Cheonan Office**

Tel. 82-41-576-4294 / Fax. 82-41-576-4296
355-15, Daga-dong, Cheonan-si, Chungcheongnam-do, Korea

- **Daegu Office**

Tel. 82-53-746-6232 / Fax. 82-53-746-6231
223-5, Beomeo 2-dong, Suseong-gu, Daegu, Korea

- **Gwangju Office**

Tel. 82-62-363-5272 / Fax. 82-62-363-5273
415-2, Nongseong-dong, Seo-gu, Gwangju, Korea

- **본사**

Tel. 052-202-7901 / Fax. 052-202-7900
울산광역시 동구 전하동 1번지

- **A/S 센터**

Tel. 82-52-202-5041 / Fax. 82-52-202-7960

- **서울 사무소**

Tel. 02-746-4711 / Fax. 02-746-4720
서울특별시 종로구 계동 140-2번지

- **안산 사무소**

Tel. 031-409-4959 / Fax. 031-409-4946
경기도 안산시 상록구 사동 1431-2번지

- **천안 사무소**

Tel. 041-576-4294 / Fax. 041-576-4296
충남 천안시 다가동 355-15번지

- **대구 사무소**

Tel. 053-746-6232 / Fax. 053-746-6231
대구광역시 수성구 범어 2동 223-5번지

- **광주 사무소**

Tel. 062-363-5272 / Fax. 062-363-5273
광주광역시 서구 농성동 415-2번지