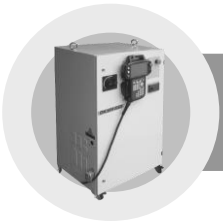
현대중공업

경고

모든 설치 작업은 반드시 자격있는
설치기사에 의해 수행되어야 하며
관련 법규 및 규정을 준수하여야 합니다.

Hyundai Robot

Hi50T120401FMKR1



Hi5 제어기 기능설명서

온라인 트래킹
(On-Line Tracking)

HYUNDAI
HEAVY INDUSTRIES CO.,LTD.

HYUNDAI
HEAVY INDUSTRIES CO.,LTD.

Printed in Korea - 2012년 4월. 1판

현대중공업

목 차

그림 목차

표 목차

1
개요

1.1. 서론

온라인 트래킹 기능은 UDP/IP 통신으로 로봇 제어기에 사용자가 원하는 로봇의 지령을 실시간으로 반영하여 임의의 로봇 동작을 가능하게 할 수 있는 기능입니다. 사용자는 외부 PC 에서 로봇의 위치 증분지령을 직접 계획, 생성하여 로봇 제어기로 송신하면 로봇 제어기는 위치 증분지령을 수신 후 모션계획에 반영하고 이와 동시에 외부 PC 에 로봇의 현재 직교좌표 위치를 송신하여 피드백 해줍니다.
본 기능을 사용하면 사용자는 필요한 경우 외부 PC 에 임의의 센서를 부착하고 특정 작업을 위한 임의의 모션을 계획하고 반영할 수 있지만, 외부 PC 의 센서 인터페이스나 UDP/IP 통신 구현은 사용자가 직접 용도에 맞게 구현하여야 합니다. 또한 로봇 제어기는 5msec 마다 UDP/IP 송수신을 수행하고 5msec 내에 움직일 위치 증분지령을 반영하기 때문에 외부 PC 는 정확히 5msec 마다 모션을 계획하고 UDP/IP 통신을 수행하여야만 로봇이 부드러운 동작을 수행할 수 있습니다.
UDP/IP 통신 시 로봇 제어기는 client, 외부 PC 는 server 역할을 담당합니다. 사용자가 원할 경우, 외부 PC 에 구현된 UDP/IP 통신 프로그램은 C 언어로 작성한 예제 프로그램을 3.2 절에서 제공하고 있으니 참고하시기 바라며, 관련 문의는 당사 A/S 에 연락 주시기 바랍니다.

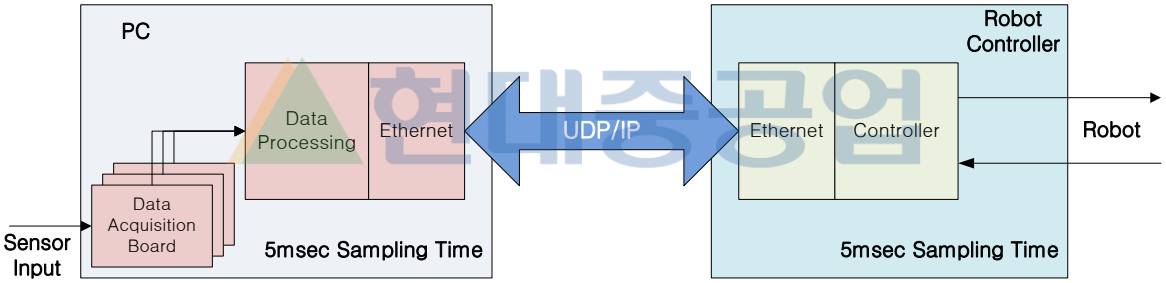


그림 1.1 온라인 트래킹 기능의 시스템 구성 예

1.2. 기능 요약

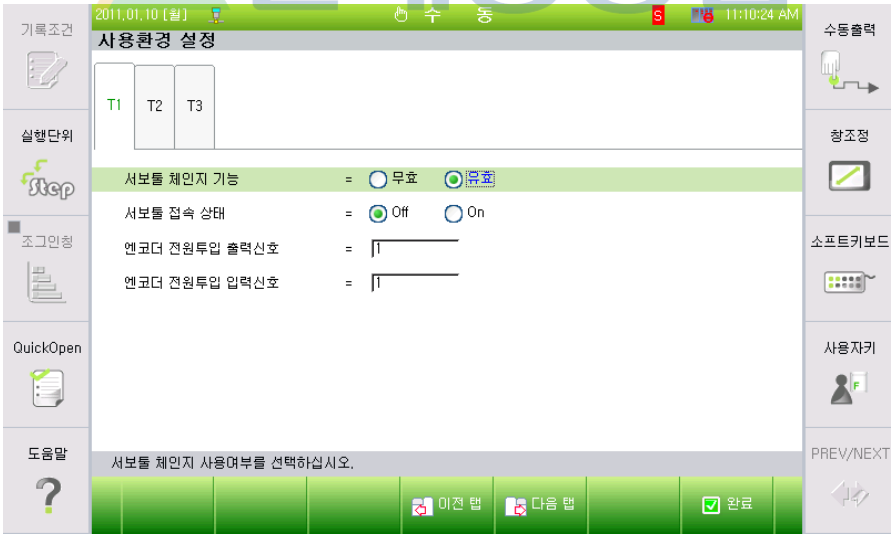
- 통신 프로토콜 : UDP/IP (64Bytes)
  - 로봇 제어기 → PC : 로봇의 현재위치
  - PC → 로봇 제어기 : 로봇의 증분지령
- 증분지령 반영주기 : 5msec (200Hz)
- 증분지령 필터적용 지원
- 동작영역, 속도제한 설정 기능 지원

2
사용 방법

2.1. 온라인 트래킹 프로그램 구조

온라인 트래킹 기능을 사용하기 위해서는 JOB 파일 상에 OnLTrack 명령으로 본 기능을 활성화 시키면서 통신 및 위치 증분지령 필터 설정을 하고, 필요한 경우 LIMIT 명령으로 로봇의 동작영역, 속도제한 설정을 해야 합니다. 마지막으로 OnLTrack 명령으로 온라인 트래킹 기능을 비활성화 하여 기능 종료를 수행하면 됩니다.

구분	설명	프로그램 예
기능시작, 통신 및 필터 설정	온라인 트래킹 기능을 On 하면서 UDP/IP 통신과 증분지령의 필터 설정을 합니다.	OnLTrack ON,IP=192.168.1.254, PORT=7127,CRD=1,Bypass,Fn=10
동작영역, 속도제한 설정	온라인 트래킹 기능 활성화 시 로봇의 동작영역 설정과 동작 속도제한 설정을 합니다. 설정이 없을 경우, 온라인 트래킹 기능의 default 값이 적용됩니다. (아래 LIMIT 명령문 참조)	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
기능종료	온라인 트래킹 기능을 off 합니다.	OnLTrack OFF



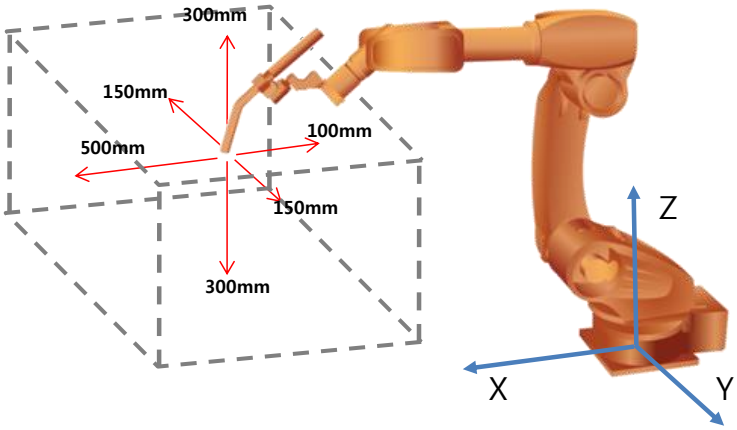
2.2. 온라인 트래킹 관련 명령문

온라인 트래킹 기능에 사용되는 명령문은 OnLTrack 과 LIMIT 입니다. 온라인 트래킹의 활성/비활성화와 통신 및 필터설정은 OnLTrack 이 사용되고, 동작영역과 속도제한은 LIMIT 문을 사용합니다. 각 명령문의 설명은 다음과 같습니다.

2.2.1. OnLTrack 문

<table>
<tr><td>설명</td><td colspan="2">UDP/IP 에 의해 이더넷으로 위치 증분지령이 입력될 때, 이를 반영하여 로봇을 동작</td></tr>
<tr><td>입력방법</td><td colspan="2">『[F6]: 명령입력』 → 『[F1]: 모션,I/O』 → 『PREV/NEXT』 → 『PREV/NEXT』 → 『[F4]: OnLTrack』</td></tr>
<tr><td>문법</td><td colspan="2">OnLTrack <ON/OFF>,IP=<IP 주소>,PORT=<포트번호>,CRD=<기준좌표계>,[<사용자좌표계번호>],[Bypass],[Fn=<주파수>]</td></tr>
<tr><td rowspan="7">파라미터</td><td>ON/OFF</td><td>ON : 유효, OFF : 무효</td></tr>
<tr><td>IP 주소</td><td>이더넷 통신을 위한 PC 의 IP 주소.</td></tr>
<tr><td>포트번호</td><td>이더넷 통신을 위한 PC 의 포트번호.</td></tr>
<tr><td>기준좌표계</td><td>산술식. 증분지령이 반영되어 로봇이 동작할 기준좌표계 (0=베이스,1=로봇,2=툴,3=U,4=Un).</td></tr>
<tr><td>사용자좌표계번호</td><td>산술식. 기준좌표계가 U, Un 인 경우 사용자좌표계 번호.</td></tr>
<tr><td>Bypass</td><td>지령필터의 통과 여부(ON=미통과, OFF=통과).,
통과 설정 시에는 제어기 내 고유필터 적용됨.
미통과 설정 시에는 별도의 필터가 적용됨.</td></tr>
<tr><td>주파수</td><td>필터 미통과(Bypass ON)시 적용될 별도 필터의 cut-off 주파수.</td></tr>
<tr><td>사용 예</td><td colspan="2">OnLTrack ON,IP=192.168.1.254,PORT=7127,CRD=1,Bypass,Fn=10
'OnLTrack 기능 동작, 로봇 좌표계로 동작, cut-off 주파수가 10Hz 인 별도 필터적용
OnLTrack OFF 'OnLTrack 기능 종료</td></tr>
<tr><td>참고사항</td><td colspan="2">▪ 제어기는 OnLTrack ON 과 OnLTrack OFF 사이에서 PC 로부터 위치 증분지령을 받을 수 있습니다.
▪ OnLTrack ON 수행 시 필터 설정에 의해 제어기 내에서 위치 증분지령을 필터링하여 로봇 동작에 반영합니다. 이 때 필터링에 의한 증분지령 반영 시 시간지연이 발생할 수 있습니다.
▪ OnLTrack ON 실행 시 'LIMIT POS,+X=300,-X=300,+Y=300,-Y=300,+Z=300,-Z=300'과 'LIMIT VEL,X=200,Y=200,Z=200,RX=100,RY=100,RZ=100'이 자동으로 설정됩니다.</td></tr>
</table>

2.2.2. LIMIT 문

<table>
<tr><td>설명</td><td colspan="2">OnLTrack 기능에 의해 위치 증분지령을 반영하여 로봇을 동작할 때, 최대 동작영역과 최대 속도를 설정하는 기능</td></tr>
<tr><td>입력방법</td><td colspan="2">『[F6]: 명령입력』 → 『[F1]: 모션,I/O』 → 『PREV/NEXT』→ 『PREV/NEXT』 → 『[F7]: LIMIT』</td></tr>
<tr><td>문법</td><td colspan="2">LIMIT POS,[+X=<+X 거리>],[-X=<-X 거리>],[+Y=<+Y 거리>],[-Y=<-Y 거리>],[+Z=<+Z 거리>],[-Z=<-Z 거리>]
LIMIT VEL,[X=<X 속도>],[Y=<Y 속도>],[Z=<Z 속도>],[RX=<RX 속도>],[RY=<RY 속도>],[RZ=<RZ 속도>]</td></tr>
<tr><td rowspan="3">파라미터</td><td>POS/VEL</td><td>제한항목. POS : 거리, VEL : 속도</td></tr>
<tr><td>제한거리</td><td>산술식. 위치 증분지령을 반영하여 로봇이 동작할 때, 로봇이 동작할 수 있는 로봇 좌표계 상 최대 영역을 현재 위치 기준으로 설정</td></tr>
<tr><td>제한속도</td><td>산술식. 위치 증분지령을 반영하여 로봇이 동작할 때, 로봇이 동작할 수 있는 로봇 좌표계 상 최대 속도를 설정</td></tr>
<tr><td>사용 예</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>참고사항</td><td colspan="2">▪ LIMIT POS 는 로봇의 현재 위치 기준으로 최대 동작영역이 설정됩니다. 동작영역을 벗어나는 위치 증분지령은 무시됩니다.
예) LIMIT POS,+X=500,-X=100,+Y=150,-Y=150,+Z=300,-Z=300
 

▪ OnLTrack ON 실행 시 'LIMIT POS,+X=300,-X=300,+Y=300,-Y=300,+Z=300,-Z=300'과 'LIMIT VEL,X=200,Y=200,Z=200,RX=100,RY=100,RZ=100'이 자동으로 설정됩니다.
▪ OnLTrack ON 실행 전 LIMIT 명령은 무시되므로 별도의 설정을 원할 경우 OnLTrack ON 이후 설정하여야 합니다.
▪ OnLTrack ON 실행 후 LIMIT 명령은 여러번 사용할 수 있으나 LIMIT POS 는 실행 시 로봇의 현재 위치 기준으로 동작영역을 재설정하는 것에 유의해야 합니다.</td></tr>
</table>

2.3. 로봇 제어기-PC 간 통신 데이터 구조 및 통신 방법

2.3.1. 통신 방법 및 데이터 구조

온라인 트래킹 기능에서 PC 는 server, 로봇 제어기는 client 로 UDP/IP 통신이 수행되고 통신 주기는 5msec 입니다. 그림 2.3 은 PC 에서 UDP/IP 통신으로 로봇 제어기와 송수신하는 데이터 구조를 C 언어로 표현한 것 입니다. 보낼 때와 받을 때 데이터 크기는 모두 64Bytes 입니다. (char 형 1byte, int 형 4bytes, double 형 8bytes)

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

그림 2.1 UDP/IP 통신의 송수신 데이터 구조

그림 2.3 의 Command 는 PC 와 로봇 제어기 간 연결시작, 연결종료, 위치 증분지령와 로봇 위치 전송을 나타내는 변수입니다. PC 와 로봇 제어기의 올바른 통신연결과 데이터 송수신을 위해서는 표 2.1 과 같이 Command 값을 적절히 설정, 전송해야 합니다. 온라인 트래킹 기능의 통신 동작순서는 2.3.2 절을 참조 하십시오.

표 2.1 송수신 시 Command 변수 내용

변수명		데이터 전송 방향	
		로봇 제어기 → PC	PC → 로봇 제어기
Command	'S'	- OnLTrack ON 실행 시 전송 - 온라인 트래킹 기능 시작을 알림	- 제어기로부터 'S' 명령이 수신된 후 'S'를 다시 전송하여 연결 확인을 알림 ('S'를 제어기로 전송하지 않으면 제어기는 이후 전송되는 데이터를 무시함)
	'P'	- 위치 증분지령이 수신되었을 때 전송 (State 변수 참조) - 로봇의 현재 위치를 같이 송신함 (dData 변수 참조)	- 위치 증분지령을 송신할 때 사용함 (dData 변수 참조)
	'F'	- OnLTrack OFF 실행 시 전송 - 온라인 트래킹 기능 종료를 알림	- 제어기의 OnLTrack OFF 실행되기 전 PC 에서 온라인 트래킹 기능 종료를 할 때 전송 - 온라인 트래킹 기능 종료를 알림

그림 2.3 의 State 변수는 로봇 제어기에서 PC 로 데이터를 송신할 때만 의미를 갖습니다. State=1 이면 연결시작을, State=3 이면 연결종료를 나타냅니다. 따라서 Command='S'일 때 State=1 이고, Command='F'일 때 State=3 입니다. Command='P'일 때는 State 는 -1 이나 2 입니다. 2 이면 위치 증분지령이 로봇 모션계획에 반영된 것을 나타내며 -1 이면 반영되지 못함을 뜻합니다. (로봇의 특이점 위치(singular point)에서는 로봇의 동작이 제한적이어서 위치 증분지령을 반영하지 못할 수 있습니다.) State 변수 내용을 정리하면 표 2.2 와 같습니다.

표 2.2 송수신 시 State 변수 내용

변수명		데이터 전송 방향	
		로봇 제어기 → PC	PC → 로봇 제어기
State	1	- OnLTrack ON 실행 시 전송 - 온라인 트래킹 기능 시작을 알림	- 사용하지 않음 (reserved)
	2	- 위치 증분지령이 수신, 반영되었을 때	
	-1	- 위치 증분지령이 수신되었으나 반영되지 못했을 때	
	3	- OnLTrack OFF 실행 시 전송 - 온라인 트래킹 기능 종료를 알림	

그림 2.3 의 Count 는 PC 에서 로봇 제어기로 위치 증분지령을 송신 시 몇번째 위치 증분지령 전송인지를 나타내는 숫자로 사용자가 설정하는 변수입니다. 제어기에서 PC 로 로봇 현재 위치를 송신할 때는 직전에 PC 로부터 받은 Count 값을 그대로 보냅니다. 따라서 사용자는 Count 변수를 이용하여 매 데이터 전송마다 Count 를 1 씩 증가시키면서 제어기로부터 데이터 수신 시 Count 를 송신한 값과 비교하면 직전의 데이터 전송이 수행되었는지 쉽게 확인할 수 있습니다. Count 변수 내용을 정리하면 표 2.3 과 같습니다.

표 2.3 송수신 시 Count 변수 내용

변수명	데이터 전송 방향	
	로봇 제어기 → PC	PC → 로봇 제어기
Count	- PC 로부터 수신된 값을 현재 로봇 위치와 함께 PC 로 재송신	- 위치 증분지령 송신 횟수값 - 사용자가 직접 매 송신마다 0 부터 시작하여 1 씩 증가시켜 이전 데이터의 제어기 수신을 확인할 수 있음

그림 2.3의 dData은 전송 방향에 따라 다른 데이터 내용을 갖습니다. PC에서 로봇 제어기로 송신 시에는 5msec 동안 로봇이 움직일 위치 증분지령이고 로봇 제어기에서 PC 로 송신 시에는 로봇의 현재 직교좌표 위치값입니다. 위치 증분지령은 부드러운 로봇의 동작을 위해 가감속을 고려하여 5msec 마다 실시간으로 송신을 할 것을 추천합니다. 제어기의 OnLTrack ON 문 수행 시 필터설정에 따라 위치 증분지령이 필터링되어 부드럽게 반영되나 시간지연이 발생할 수 있습니다. 또한 위치 증분지령은 LIMIT POS 와 LIMIT VEL 문으로 제한되므로 2.2.2 절의 명령문 설명을 참조하시기 바랍니다. 로봇 제어기로부터 PC 로 수신되는 로봇의 현재 직교좌표 위치는 수신된 위치 증분지령을 반영한 로봇 좌표계 기준의 툴 위치값를 의미합니다. 로봇의 현재 직교좌표 위치값은 TP 상에서 모니터링할 수 있는 로봇의 직교좌표 위치값과 동일합니다. dData 변수 내용을 정리하면 표 2.4 와 같습니다.

표 2.4 송수신 시 dData 변수 내용

변수명	데이터 전송 방향	
	PC → 로봇 제어기	로봇 제어기 → PC
dData[0]~[5]	- 5msec 동안 로봇이 움직일 위치 증분지령 (증분지령 좌표계는 OnLTrack ON 명령문의 설정을 따름) dData[0] : X 방향 증분지령, m 단위 dData[1] : Y 방향 증분지령, m 단위 dData[2] : Z 방향 증분지령, m 단위 dData[3] : Rx 방향 증분지령, rad 단위 dData[4] : Ry 방향 증분지령, rad 단위 dData[5] : Rz 방향 증분지령, rad 단위	- 위치 증분지령을 반영한 로봇 좌표계의 현재 툴 위치값 dData[0] : X 방향 현재위치, m 단위 dData[1] : Y 방향 현재위치, m 단위 dData[2] : Z 방향 현재위치, m 단위 dData[3] : Rx 방향 현재위치, rad 단위 dData[4] : Ry 방향 현재위치, rad 단위 dData[5] : Rz 방향 현재위치, rad 단위

2.3.2. 통신 방법

PC 는 server, 로봇 제어기는 client 로 UDP/IP 통신이 수행되고 통신 주기는 5msec 입니다. 제어기의 메인보드는 3 개의 네트워크 포트 (EN0, EN1, EN2)를 제공하나 PC 에서는 EN2 에 설정된 IP 주소와 포트 번호=6001 와 통신해야 합니다. (EN2 의 IP 주소 등 설정은 'Hi5 제어기 조작설명서'의 네트워크 편을 참조하십시오.) PC 의 IP 주소와 포트 번호는 OnLTrack ON 문에 설정된 값과 동일하여야 하며 PC 와 제어기는 크로스 이더넷 케이블로 연결되어야 합니다. PC 와 제어기의 데이터 송수신을 위해서는 통신 순서에 맞도록 Command 변수를 송수신해야 합니다. 그림 3.1 은 PC 와 제어기간 통신 순서와 송수신 되는 Command 변수를 나타낸 것입니다.

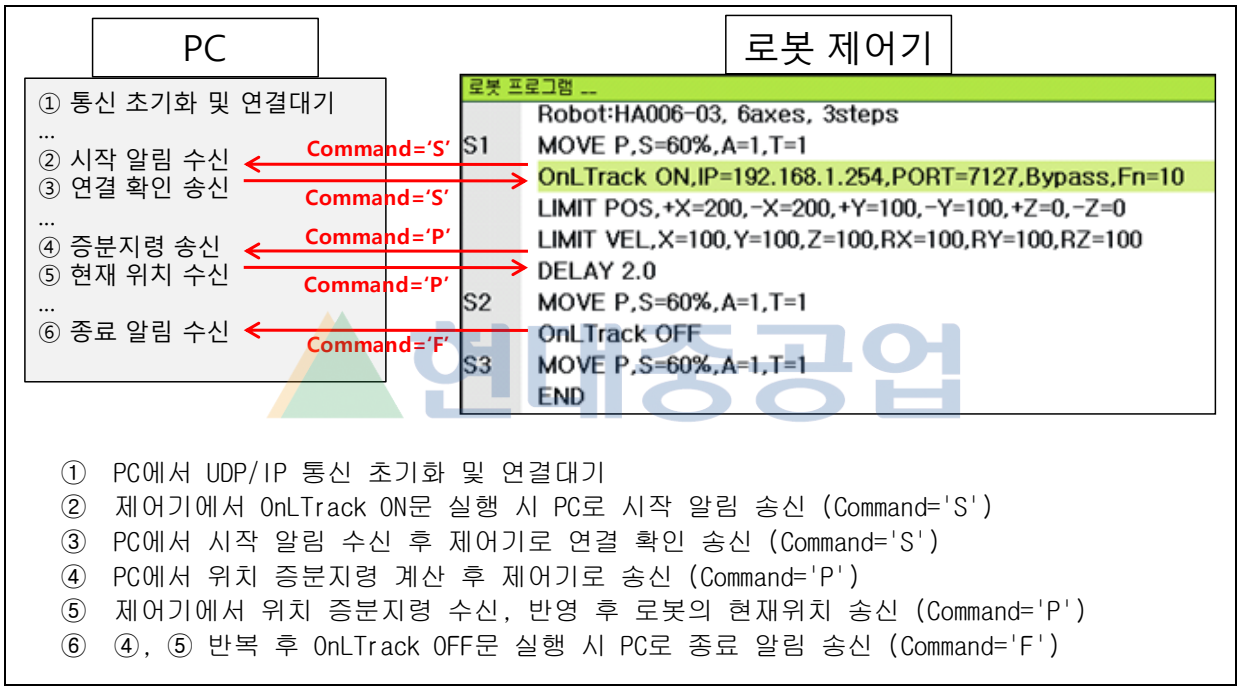


그림 2.2 PC 와 로봇 제어기간 통신 순서

PC 는 server, 로봇 제어기는 client 로 UDP/IP 통신이 수행되어야 합니다. client 인 로봇 제어기는 그림 3.1 과 같은 로봇 프로그램이 재생되면 OnLTrack ON, OFF 문에 따라 server 인 PC 에 자동으로 통신연결이 시작 및 종료가 되면서 온라인 트래킹 기능이 작동될 수 있습니다. 또한 PC 와 로봇 제어기 간 통신 주기는 5msec 입니다. 만약 제어기 5msec 이내에 2 개 이상의 데이터가 수신되면 최근 수신된 데이터만 적용되고 이전 데이터는 무시되므로 PC 에서 데이터 전송 시 유의하시기 바랍니다.

3
기타

3.1. 온라인 트래킹 기능 사용 시 로봇 동작

```
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
```

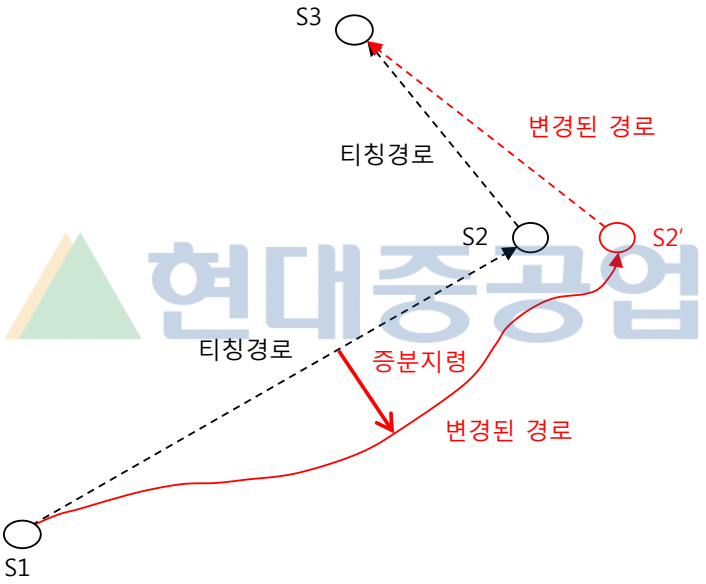


그림 3.1 온라인 트래킹 기능 사용 시 로봇의 경로 변경

그림 3.1 은 온라인 트래킹 기능에 의해 로봇의 경로가 어떻게 변하는지를 나타냅니다. 온라인 트래킹 기능을 사용하면 본 기능이 활성화 되는 OnLTrack ON 과 OnLTrack OFF 문 사이의 스텝 S2 지점에 도달하지 못하고 임의의 S2' 지점에 도달하게 됩니다. 위치 증분지령이 로봇의 동작에 반영되어 수정되기 때문에 계획된 티칭경로를 벗어나기 때문입니다. 하지만 OnLTrack OFF 이후에는 그림 3.1 에서 보듯이 S2' 지점에서 출발하여 스텝 S3 지점에 도달하게 됩니다. 이처럼 온라인 트래킹 기능을 사용할 때는 티칭경로가 증분지령에 의해 임의의 경로로 변경될 수 있으니 유의하시기 바랍니다.

3.2. 온라인 트래킹 기능 사용을 위한 UDP/IP 프로그램 예시

다음은 온라인 트래킹 기능 사용을 위한 PC 에서 구동할 UDP/IP 통신 프로그램 예입니다. 키보드 버튼입력으로 로봇을 동작하는 프로그램으로 C 언어로 구현되었으며, 프로그램 관련 문의는 당사 A/S 에 연락하시기 바랍니다.

```
#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 ); //소켓을닫습니다.
    WSACleanup();

   printf("Program is terminated.Wn");

    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...Wn" );
   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( "recv>> Command: %c,  Count: %d,  [X: %.3f, Y: %.3f, Z: %.3f, Rx: %.3f,
```

```
Ry: %.3f, Rz: %.3f] Wn",
                    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.Wn");

            }
         }
         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] Wn",
                    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.Wn" );
               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.Wn" );
        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;
}
```

현대중공업

HYUNDAI
HEAVY INDUSTRIES CO.,LTD.

- **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 1 번지

- 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 140-2 번지

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

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

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

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