

CoreDev API Manual



Revised on 2018. 04. 09.

본 제품 설명서에서 제공되는 정보는 코아로봇의 자산입니다.

코아로봇의 서면에 의한 동의 없이 전부 또는 일부를 무단 전재 및 재배포할 수 없으며,
제 3 자에게 제공되거나 다른 목적에 사용할 수 없습니다.

본 설명서는 성능 및 기능 개선을 위해 사전 예고 없이 변경될 수 있습니다.

Printed in the Republic Of Korea - 2016 년 6 월. 초판

CoreDev API Manual Revision History

[illegible]

목차

1. CNRobo 구조	7
2. CNRobo 의 관계도.....	8
3. CNRobo Header 및 Library.....	9
1) CNRobo 의 주요 header file :.....	9
2) CNRobo 의 library file : libcnrobo – Release 파일, libcnrobo – Debug 파일.....	9
4. CuiApi 구조	10
5. CuiApi 의 관계도.....	11
6. CuiApi Header 및 Library	12
3) CuiApi 의 주요 header file :.....	12
7. CoreCUI 에서 CNRobo Build.....	13
1) Desktop build : build 환경 조건 Qt version : Desktop Qt 5.5.1 32bit, Compiler: GCC.....	13
2) Build & run 설정 : QT 의 projects 의 build & run 을 아래와 그림과 같이 설정한다. 13	
3) QtARM build : build 환경 조건 Qt version : Qt 4.7.0 , Compiler : ARM-GCC 13	
4) Build & run 설정 : QT 의 projects 의 build & run 을 아래와 그림과 같이 설정한다. 13	
8. Main 화면	15
9. 초기화, Server 접속	16
1) CNRobo object 얻기	16
2) 초기화 작업과 Server 접속.....	16
10. Version 확인 및 Login	17
1) Version 확인.....	17

11.	Lock & Unlock	18
12.	Robot Status 확인	19
13.	Zeroing 과 ZeroOffset.....	21
1)	Zeroing 하기.....	21
2)	ZeroOffset 하기.....	21
3)	Encoder counter 얻기	22
14.	Jogging 화면 제작 (coreCUI base).....	23
1)	Servo on/off.....	23
2)	Enable get, set	23
3)	Inching.....	23
4)	Inching 값 설정.....	24
5)	Coordinate	24
6)	TR mode change	24
7)	Teach speed.....	24
8)	Error handling	25
9)	Warning handling.....	25
10)	Jog 동작.....	25
15.	Programs 리스트 관리	27
1)	Macro Program 전체 List 얻기	27
2)	Macro program 추가.....	27
3)	Macro program 삭제.....	27
4)	Macro program Step List 얻기	27
5)	Macro Step 추가, 저장	28
16.	Macro Program 실행.....	29
1)	load.....	29
2)	Start	29
3)	Hold.....	29

4)	Continue	29
5)	Abort	30
6)	Speed	30
7)	Step run	30
8)	Program run count 설정	30
9)	Reset.....	31
10)	Clear	31
11)	Step index 얻기	31
17.	Variables	32
1)	변수에 대한 공통 사항	32
2)	변수 생성	32
3)	변수 count 얻기.....	32
4)	변수 name 얻기	32
5)	변수 삭제	33
6)	변수 data 얻기	33
7)	변수 위치로 모터 이동(joint, trans).....	34
18.	Digital Input/Output.....	35
1)	getDI.....	35
2)	getDO	35
3)	setDI	35
4)	setDO	36
19.	Analog Input/Output	37
1)	getAI	37
2)	getAO.....	37
3)	setAO	37

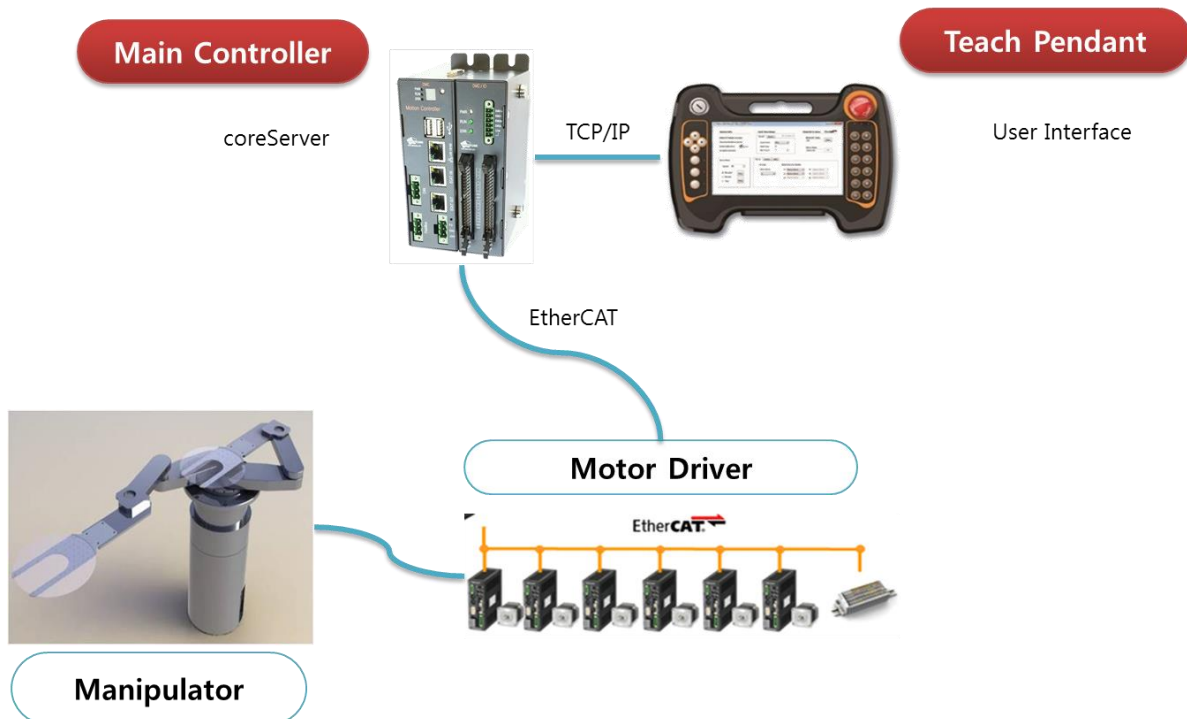
1. CNRobo 구조

로봇의 Main Controller 에는 로봇 제어 프로그램 및 로봇 언어 실행, 시스템 설정등의 로봇제어를 위한 소프트웨어가 내장되어 실행되어 있고, 사용자의 사용 환경은 Teach Pendant 및 PC, Tablet 등의 다양한 환경에서 제작되어 사용되어 질 수 있다.

이것은 Main Controller 와 User Interface 는 분리되어 네트워크를 통해 데이터를 주고 받기 때문이다. CNRobo 는 TCP/IP 를 이용한 로봇 API 의 집합으로, Main Controller 와 User Interface 사이의 네트워크 통신을 기반으로 제작되어 있다.

따라서, User Interface 개발자는 CNRobo 만을 이용하여, 로봇제어용 UI 프로그램을 제작할 수 있다.

- 장치 구조

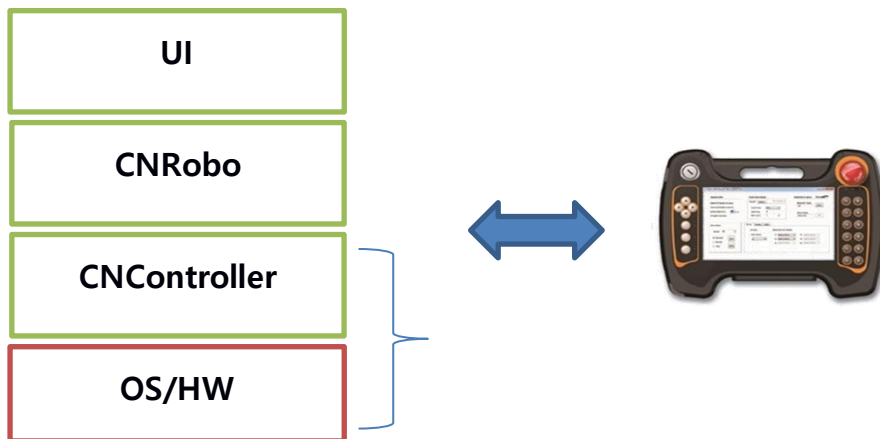


CNRobo 는 Qt 기반으로 제작되었기 때문에, 윈도우, 리눅스, 안드로이드 등의 다양한 OS 플랫폼아래 빌드된 라이브러리를 제공한다. CNRobo 는 자체의 중요 로봇 데이터의 복사본을 가지고 있게 되어, 개발자는 CNRobo 를 가상의 로봇으로 가정하여 명령을 실행하거나, 상태를 읽어 올 수 있게 된다. CNRobo 의 하부는

네트워크 및 OS 서비스를 이용하여, 원격의 메인 제어기인 coreServer 와 통신하는 모듈로 구성되어 있다.

2. CNRobo의 관계도

- UI, CNRobo, coreServer 관계.



3. CNRobo Header 및 Library

이후의 설명은 Linux 환경하에서의 개발을 전제로 설명을 할 것이며, Window, Mac 등도 유사한 과정을 통해 개발이 가능하다.

1) CNRobo의 주요 header file :

cnrobo.h : robot api 를 제공한다.

cntype.h : robot api 에서 사용하는 type 정의

cnerror.h : error code 정의

cnhelper.h : 각종 함수 제공.

2) CNRobo의 library file : libcnrobo – Release파일, libcnrobo – Debug파일

linux86 : libcnrobo.a, libcnrobo_d.a

linuxarm : libcnrobo.a, libcnrobo_d.a

mingw : libcnrobo.a, libcnrobo_d.a

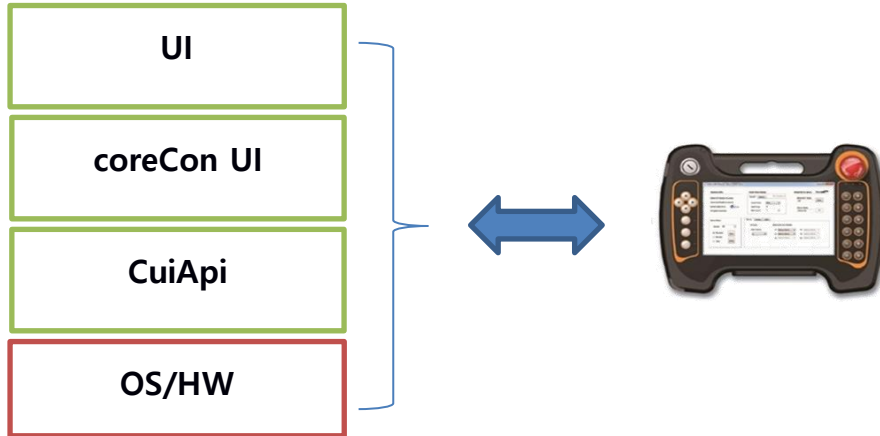
msvc : cnrobo.lib, cnrobo_d.lib

4. CuiApi 구조

로봇의 제어할 수 있는 Main Program 인 corecon 이 있고 corecon 에
내장되어있는 Robot Engine 을 제어할 수 있는 CUIAPI 가 있다. CUIAPI 를 통해
Custom UI 에서는 Robot 의 각종 기능들을 corecon ui 와는 별도로 제어할 수 있다.
Corecon 과 Custom UI 를 Plugin 하는 과정이 필요한데, Custom UI 를 So library
형태로 제작하여 corecon 에 Custom UI 에 대한 Widget Pointer 를 Return 하여
corecon 에서 Custom UI 의 Widget 를 Show 하여 사용한다.

5. CuiApi의 관계도

- UI, CuiApi, coreCon 관계.



6. CuiApi Header 및 Library

이후의 설명은 Linux 환경하에서의 개발을 전제로 설명할 것이다.

3) CuiApi 의 주요 header file :

cuiapi.h : robot api 를 제공한다.



cntype.h : robot api 에서 사용하는 type 정의

7. CoreCUI에서 CNRobo Build

CuiApi의 Build 및 제작은 Custom UI manual을 참고

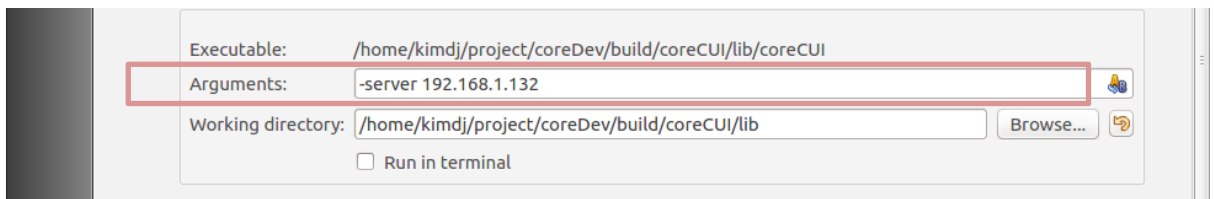
CuiApi의 경우 통신관련 설정은 불필요 하며 통신 관련 API는 제공하지 않는다.

- 1) Desktop build : build 환경 조건 QT version : Desktop Qt 5.5.1 32bit, Compiler: GCC
- 2) Build & run 설정 : QT의 projects의 build & run을 아래와 그림과 같이 설정한다.

 (Projects) Tap 에 있는  (Build& Run) Tap 으로 이동 Arguments 설정.

build 설정은 default 값을 사용하고 Run 만 설정한다.

- Run Tap-

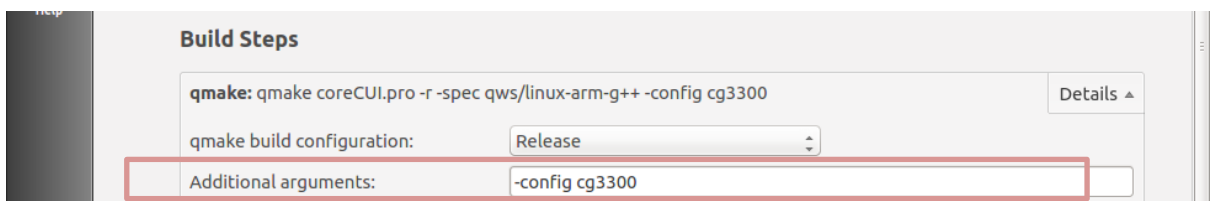


Arguments 내용 : -server [접속할 Device IP 주소]

ex) -server 192.168.1.132 <- Device의 ip가 192.168.1.132 일 경우 이와 같이 설정.

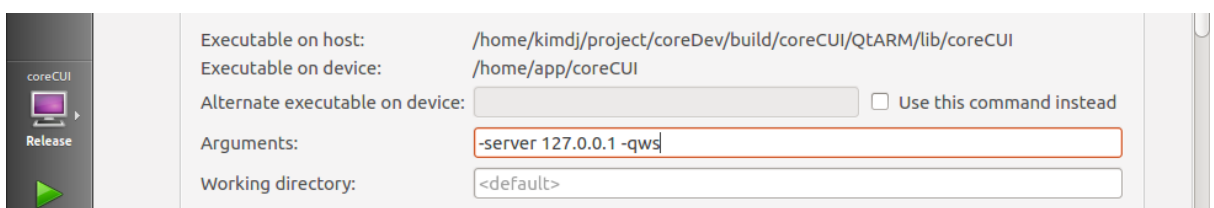
- 3) QtARM build : build 환경 조건 Qt version : Qt 4.7.0 , Compiler : ARM-GCC
- 4) Build & run 설정 : QT의 projects의 build & run을 아래와 그림과 같이 설정한다.

- Build Tap-



Arguments 내용 : -config cg3300

- Run Tap-

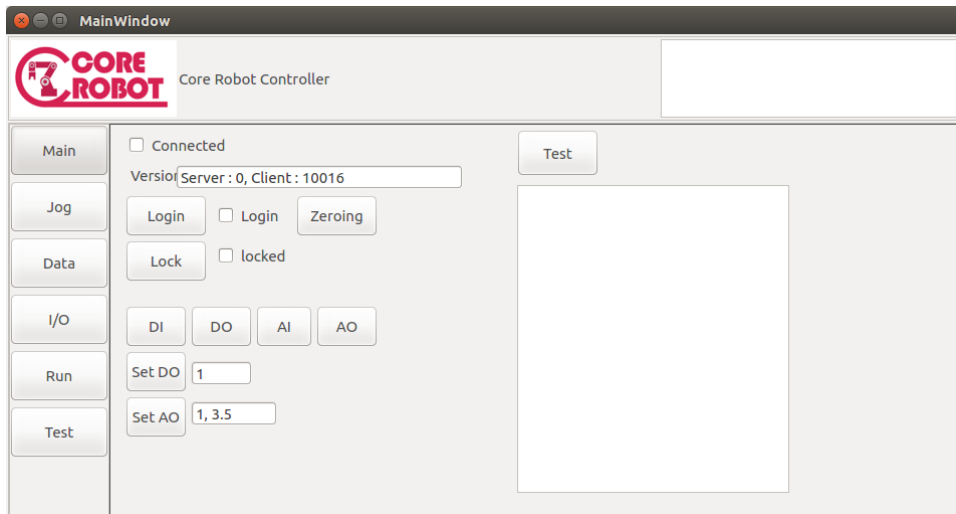


Argments 내용 : -server [로컬 IP 주소] -qws

프로그램이 Device 에서 실행되므로 로컬 IP 를 설정해주고 -qws 옵션을 사용

ex) -server 127.0.0.1 -qws

8. Main 화면 Example



메인 화면은 주로 server 와 관계된 기능과 status 들로 구성되었다.

9. 초기화, Server 접속

1) CNRobo object 얻기

CNRobo 는 singleton 방식을 채용하였다, 그러므로 CNRobo::getInstance()으로 Object 를 얻을 수 있다

ex) object 얻기

```
CNRobo* pCon = CNRobo::getInstance();
```

2) 초기화 작업과 Server 접속

초기화 작업 : ip address 설정, 서버 접속.

ex) IP 설정

```
CNRobo* pCon = CNRobo::getInstance();
```

```
QString serverAddress = "192.168.1.132";           //IP 입력
```

```
pCon->setServerAddress(serverAddress);             //IP 설정
```

ex) Server 접속

```
pCon->startService();                             //Service 시작.
```

서버와의 연결 상태는 pCon->isConnected()를 통해 확인 할 수 있다.

만일 서버와의 접속이 되지 않을 경우, 주기적으로 연결을 재개한다.

ex) Server 접속 해제

```
pCon->stopService();                               //Service 종료
```


10. Version 확인 및 Login

1) Version 확인

실행 중인 coreServer 의 Version 과 , CNRobo API 의 Version 이 같아야, API 의 사용이

가능하다, 그렇기 때문에 Version 확인을 하도록 한다.

ex) version 확인과 비교

```
CNRobo* pCon = CNRobo::getInstance();
unsigned int cnroboVer;
unsigned int serverVer;
cnroboVer = pCon->getVersion();           //cnrobo version 얻기
pCon->getServerVersion(&serverVer);       //server version 얻기
//view version
```

CNRobo 와 coreServer 버전이 다른 문제가 있을 때 기술 지원을 통해 업데이트를 받도록 한다.

ex) cnrobo, coreServer, Master, Kernel version 확인

```
CNRobo* pCon = CNRobo::getInstance();
QString strVer;
pCon->getVersion2(CNRobo::VT_CNROBO, strVer);
//view version
pCon->getVersion2(CNRobo::VT_CORESERVER, strVer);
//view version
pCon->getVersion2(CNRobo::VT_MASTER, strVer);
//view version
pCon->getVersion2(CNRobo::VT_KERNEL, strVer);
//view version
```

11. Lock & Unlock

하나의 coreServer 에 여러 개의 client 가 연결 될 수 있기 때문에, 제어권은 반드시 하나의 device 에만 제한하도록 되어 있다. 즉, 상태의 읽기는 연결에 성공한 모든 client device 가 가능하나, 로봇의 상태를 변경시키는 기능은 제어권을 획득한 device 만이 가능하다. 이러한 제어권을 획득하는 명령어가 lock()이며, 사용하고 난 후, 제어권을 반납하고, 모니터링만 할 경우, unlock()을 통해 제어권을 반납한다. 만일 통신이 끊어져, client 가 서버로부터 이탈된다면, 서버는 자동으로 이탈된 client 의 제어권을 반납하여, 다른 장치가 연결하여 사용할 수 있도록 하였다. getLockStatus(bool* block)을 통해 현 상태를 확인할 수 있다.

ex) Lock & unLock

```
CNrobo* pCon = CNRobo::getInstance();
```

```
bool bLock = false;
```

```
pCon->getLockStatus(bLock); //먼저 lock 상태를 확인 후 lock 을 시도한다.
```

```
if(!bLock) pCon->lock(10000); //lock 을 시작한다 10000msec 의 timelimit 을 설정한다.
```

```
Else
```

```
pCon->unlock(); //lock 을 해제한다.
```

12. Robot Status 확인

로봇에는 각종 Status 가 존재한다 Status 를 통해 현재 로봇의 상태를 모니터할 수 있다, status 를 통해 로봇이 특정 동작을 하거나 하지 못하도록 할 수 있다.

Robot 의 status 는 대부분 bool 값을 return 한다, 하지만 status 의 종류가 많은 경우

그 외에 다른 자료형으로 return 한다.

주로 사용하는 로봇 status 는 다음과 같다,

- 1) isMotorOn, getServoOn(중복)
 - Servo on 상태에서 true 를 리턴한다.
- 2) isTPEnable, getEnableOn(중복)
 - enable 상태에서 true 를 리턴한다, (데드맨 enable key 사용 시)
- 3) isMoving
 - robot 이 동작 중일 때 true 를 리턴한다.
- 4) isTPInching, getInchingOn
 - inching mode 에서 true 를 리턴한다.
- 5) isError
 - error 상태일 때 true 를 리턴한다.
- 6) isWarning
 - warning 상태일 때 true 를 리턴한다, warning 은 macro 에서 user 가 임의로 발생시킨다
- 7) isTeach, getTeachMode
 - teach mode 에서 true 를 리턴한다, false 은 repeat mode 이다.
- 8) getTaskStatus
 - task 동작 상태를 리턴한다. Macro Program 이 load 되지 않았을 때 0, 로드 된 후 running 상태가 아닐 때 2, 로드 된 후 running 상태일 때 1 을 리턴한다.
- 9) getHoldDoneStatus
 - macro program 이 로드 된 후 run 하기 전 혹은 running 중 hold 후 완전히 내부의 모든 시퀀스가 정지하였을 때 true 를 리턴한다.
- 10) getUseUserCoordFlag

-user coordinator 를 사용 시 true 리턴.

11) getCheckMode

- check mode 에서 true 리턴.

12) getHoldRun

- macro 를 동작시키지 않거나 동작 중 정지하여 robot 이 정지되었을 때 true 리턴.

13) getDebugFlag

- debug mode 에서 true 리턴

14) getDryRunMod.

-Dryrun mode 에서 ture 리턴

*이 외에 API 의 경우 help manual 을 참조

13. Zeroing과 ZeroOffset

Zeroing 은 로봇의 Joint 값을 초기화 하는 기능이다, 특정 위치를 로봇의 zero position 으로 설정할 수 있다, 만약 zeroing 이 특정 위치에서 이루어진다면 Zeroing 이 이루어진 위치는 로봇 좌표의 zero 점이 된다.

Zeroing 은 두 종류가 있다 로봇의 zero 점을 만드는 zeroing 과 zerooffset

1) Zeroing하기

axis mask 를 설정하여 원하는 모터만 zeroing 할 수 있다.

```
ex1) CNRobo* pCon = CNRobo::getInstance();
pCon->zeroing(-1);    //모든 axis 를 zeroing 한다
ex2) CNRobo* pCon = CNRobo::getInstance();
pCon->zeroing(0x03);  //1 축과 2 축을 zeroing 한다.
```

2) ZeroOffset하기

Position 배열값과 axis 의 개수를 보내어 원하는 axis 에 offset 값을 설정할 수 있다.
Point – 원하는 offset 값을 설정하기 위해 기존에 입력된 offset 값을 get 으로 먼저 받은 후 기존 data 의 값 기준으로 offset 값을 수정하여 set 한다, 만약 무조건 set 한다면 기존의 값이 바뀔 수 있다.

```
ex) CNRobo* pCon = CNRobo::getInstance();
float posArray[32];
memset(posArray, 0, sizeof(posArray));
pCon-> getZeroingOffset(posArray);          //offset 값 얻기
int naxis = pCon->getAxisCount();           //axis count 얻기
for(int i = 0; i < naxis; i++)
    ui->txtZeroingOffset->append(QString::number(posArray[i], 'g', 6));    //offset 값을
ui 에 쓰기
QTextDocument* txtDoc = ui->txtZeroingOffset->document();    //ui 값 얻기
QString strText = txtDoc->toPlainText();
QStringList strList = strText.split("\n", QString::SkipEmptyParts);
memset(posArray, 0, sizeof(posArray));
for(int i = 0; i < naxis; i++)
{
    str = strList[i];
    posArray[i] = str.toFloat();
}
```

```
pCon->setZeroingOffset(posArray, naxis);
```

```
//offset 값 설정하기.
```

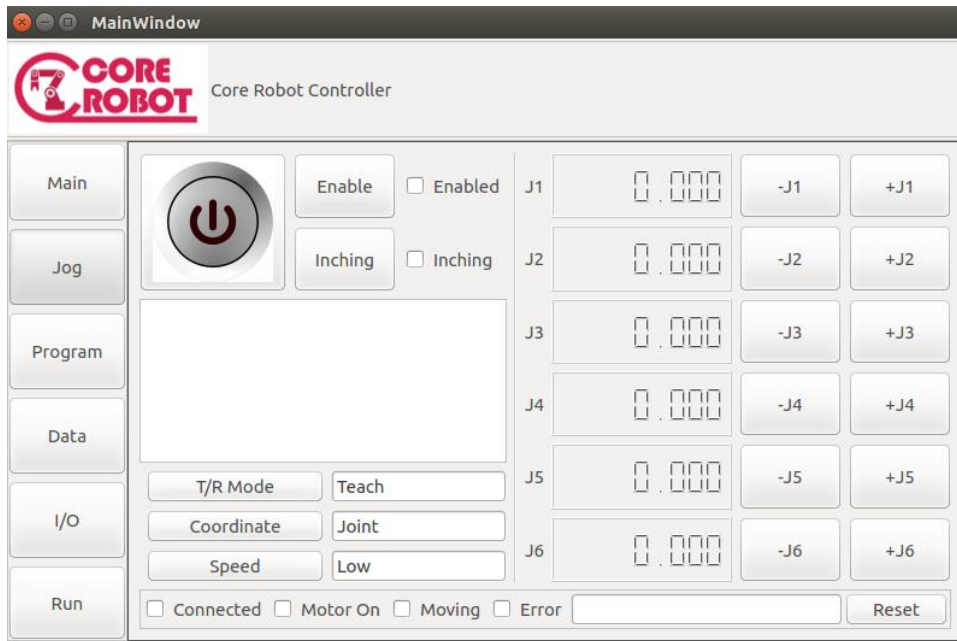
3) Encoder counter 얻기

Tp 와 연결되어있는 motor 의 encoder 값을 얻을 수 있다.

zeroingCount(long* penc)을 사용하여 penc 값을 받아 encoder 값을 확인할 수 있다.

```
ex) CNRobo* pCon = CNRobo::getInstance();  
long* enco;  
pCon->zeroingCount(enco);  
for(int i = 0; i < naxis; i++)  
{  
    Printf("encoder[%d] value = %d",i, enco[i]);  
}
```

14. Jogging 화면 제작 Example



1) Servo on/off

로봇 motor 의 전원을 담당하는 Servo on/off 는 `getServoOn()`을 통해 bool 형 리턴 값으로 현 상태를 파악한 뒤, `setServoOn(bool onoff)` 로 상태를 변경한다. 위 예시에서는 아이콘과 Motor on 체크박스를 통해 서버의 상태를 표시 해주고 있다. `isMotorOn()` 과 `getServoOn()`을 사용하여 상태를 확인할 수 있다.

2) Enable get, set

Jog 의 작동 잠금 장치 같은 역할로 `getEnableOn()`과 `setEnableOn(bool onoff)` 을 통해 상태를 변경할 수 있다. 위 예시에서는 Enable 체크 박스를 통해 표시 해주고 있다.

`isTPEnable()` 과 `getEnableOn()`을 사용하여 상태를 확인할 수 있다.

3) Inching

Inching 기능은 jog 를 동작 시킬 때 jog key 를 On/Off 했을 때 Motor 가 Speed 가 설정된 만큼 한 번만 움직이게하는 동작이다.

`getInchingOn()`과 `setInchingOn(bool onoff)`을 통해 상태를 변경하고 Inching 체크 박스로 표시 해주고 있다.

Inching 은 `setInchingOn(true);` 후 `setInchingOn(false);` 했을 때 on 되고 다시 한번 반복했을 때 off 된다.

isTPInching() 과 getInchingOn 을 사용하여 상태를 확인할 수 있다.

4) Inching 값 설정

인칭 값은 각각 mm 와 degree 로 설정하거나 얻을 수 있다.

인칭 level 은 3 단계로 되어있다, 0 : low, 1 : mid, 2 : high

인칭 가감속을 설정할 수 있다.

mm 단위의 inching 값을 level 별로 얻거나 설정한다.

getInchingStep(int level, float &step_mm); => [in] level, [out] step_mm

setInchingStep(int level, float step_mm); => [in] level, [in] step_mm

degree 단위의 inching 값을 level 별로 얻거나 설정한다.

getInchingStepAngular(int level, float& step_deg); => [in]level, [out] step_deg

setInchingStepAngular(int level, float step_deg); => [in]level, [in] step_deg

전체 인칭 level 의 가감속 값을 얻거나 설정한다. 단위는 'sec' 이다.

getInchingAccTime(float& acctime); => [out] acctime

setInchingAccTime(float acctime); => [in] acctime

5) Coordinate

Coordinate 는 getCoordinate() 와 setCoordinate(int co) 로 관리한다. 이때 int 값으로 사용되는 범위는 0~3 로 0 부터 차례대로 각각 joint coordinate, base coordinate, tool coordinate, user coordinate 를 의미한다.

getUserUserCoordFlag(bool* flag) 를 통해 user coordinate 의 존재 여부를 확인할 수 있다.

6) TR mode change

Teach 모드와 Repeat 모드를 전환하는 역할로 getTeachMode() 와 setTeachMode(bool bTeach)로 모드 전환할 수 있다. True 를 입력하면 Teach 모드로 전환되며, False 면 반대로 Repeat 모드가 된다.

isTeach () 과 getTeachMode 을 사용하여 상태를 확인할 수 있다.

7) Teach speed

Teach 과정에서 로봇이 움직이는 속도를 조절할 수 있다. getTeachSpeed()와 setTeachSpeed(int speed) 명령어를 사용하며 입력하는 int 값의 범위는 0~3 까지로 0 부터 차례대로 low , mid, high, low speed 를 의미한다.

MSPEED 에 설정 된 값에서 Low : 5% , mid : 30%, high : 100% 로 이동한다.

8) Error handling

에러는 두가지로 나뉜다, system 에서 발생하는 error 와 user 가 임의로 발생시키는 error 이 있다 user 가 발생시키는 error 은 비상정지 상황이거나 macro 명령은 RaiseError 을 사용하거나 Api 명령은 setEMGOn 를 사용했을 때 발생한다.

에러가 발생 시 프로그램이 멈추고 에러 상태가 되는데 이때 에러를 해결한 뒤, resetError(bool onoff)로 에러 상태를 해제시켜야 작동한다. pressed() 이벤트일 때, "true", released() 이벤트일 때, "false" 값을 넣어 사용한다.

isError 로 에러 상태를 확인하며, getErrorCode 로 에러 메시지를 확인한다.

getLastErrorInfo 로 main errorCode 와 error 가 발생한 axis 와 driver error code 를 확인한다.

EtherCat Error 는 resetEcatError(bool onoff)로 에러 상태를 해제시켜야 작동한다.

9) Warning handling

Warning 은 macro Program 에서 raiseWarning 명령을 통해 발생한다.

warning 시 단순히 warning flag 와 code 만 발생하기 때문에 warning 이후에 상황은

User 가 처리한다.

warning 이 발생했을 때에는 isWarning 이 true 가 되고 getWarningCode api 로 warning code 를 받을 수 있다, warning code 는 유저가 임의의 값을 정하여 발생킨다.

Warning 상태는 clearWarningCode 를 사용하여 초기화 할 수 있다.

10) Jog동작

Jog 를 동작시키기 위해서는 Motor On, Enable On, Teach mode 상태가 되어야한다.

Jog 의 on/off 는 setJogOn(unsigned int plusKey, unsigned int minusKey) 와 setJogOff(unsigned int plusKey, unsigned int minusKey) 명령어로 이루어지며 버튼이

pressed() 와 released() 상태일 때와 연동시켜 사용한다. setJogOn 과 setJogOff 의 unsigned int 형인 plusKey, minusKey 는 32 비트가 비트 하나당 Jog 하나에

대응된다.

최대 jog 8 개까지 가능하며 plusKey 와 minusKey 둘 중 한 값은 0 을 갖는다.

ex) JogOn 동작

```
unsigned int BITS[] = {  
    0x00000001,  
    0x00000002,  
    0x00000004,  
    0x00000008,  
    0x00000010,  
    0x00000020,  
    0x00000040,  
    0x00000080,  
    0x00000100,  
    0x00000200,  
    0x00000400,  
    0x00000800,  
    0x00001000,  
    0x00002000,  
    0x00004000,  
    0x00008000,  
    0x00010000,  
    0x00020000,  
    0x00040000,  
    0x00080000,  
    0x00100000,  
    0x00200000,  
    0x00400000,  
    0x00800000,  
    0x01000000,  
    0x02000000,  
    0x04000000,  
    0x08000000,  
    0x10000000,  
    0x20000000,  
    0x40000000,  
    0x80000000  
};  
  
void FormJog::onJogPressed()  
{  
    CNRobo* pCon = CNRobo::getInstance();  
    QPushButton* btn = (QPushButton*)sender();  
  
    for(int i = 0 ; i < m_jointMinus.size() ; i++)  
    {  
        if(m_jointMinus[i] == btn)  
        {  
            pCon->setJogOn(0, BITS[i]);  
            return;  
        }  
    }  
    for(int i = 0 ; i < m_jointPlus.size() ; i++)  
    {  
        if(m_jointPlus[i] == btn)  
        {  
            pCon->setJogOn(BITS[i], 0);  
            return;  
        }  
    }  
}
```

15. Programs 리스트 관리

존재하는 프로그램 리스트 정보를 관리하는 API 종류는 전체와 현재 등록된 프로그램

이름 출력, 프로그램의 추가 및 삭제, 프로그램 스텝 출력 및 추가와 같이 크게 3 가지

종류가 제공된다.

1) Macro Program 전체 List 얻기

CNRobo::getProgram(QStringList& pgms)를 사용하여 QStringList 로 받을 수 있다.

ex) program list 얻기

```
QStringList pgms;

probo->getPrograms(pgms);
for(int i = 0; i < pgms.size(); i++)
{
    ui->txtOut->append(pgms[i]);
}
```

2) Macro program 추가

createProgram(const QString& prgName)을 사용하여 생성하고자하는 macro program

이름을 prgName 인자 값으로 넣어 사용하면 macro 가 생성된다.

3) Macro program 삭제

deleteProgram(const QString& prgName) 를 사용하여 삭제하고자하는 macro program

이름을 prgName 에 인자 값으로 넣어 사용하면 macro 가 삭제된다.

4) Macro program Step List 얻기

getProgramStepsCount(const QString& prgName)을 통해 프로그램 이름에 해당하는

프로그램의 스텝 수를 리턴받을 수 있고, 직접적인 스텝 정보는

getProgramSteps(const

QString& prgName, QStringList& steps) 를 통해 얻을 수 있으며 프로그램 이름을 입력해주면 QStringList 에 스텝 정보가 저장되어 output 으로 제공된다.

ex) CNRobo* pCon = CNRobo::getInstance();

```

QString str = pCon->getCurprogramName();           //현재 load 된 macro program name 얻기
QStringList steps
pCon->getProgramSteps(str, steps);                 //macro program steps 얻기
for(int i = 0; i < steps.count(); i++)
{
    strTrim = steps[i];
    ui->txtPgm->appendPlainText(strTrim.trimmed());
}

```

5) Macro Step 추가, 저장

SetProgramSteps(const QString& prgName, QStringList& steps) 를 사용하여 원하는 macro program 에 Steps 을 저장한다, macro 가 존재하지 않을 때에는 macro 를 생성 후 Steps 를 저장한다.

ex) steps 삽입

```

QStringList steps;
steps.append("MoveJ #p1");
steps.append("MoveJ #p2");
QString prgName = "macro1";
probo->setProgramSteps(prgName, steps);

```

16. Macro Program 실행

서버에 프로그램 등록과 그 후 실행 및 관리하는 함수에 여러 종류가 제공된다.

1) load

setCurProgram(int taskid, const QString& prgName, int cycle)을 사용하여 현재 server 에 macro program 을 등록한다, 혹은 이미 등록된 프로그램을 교체할 수도 있다.

등록 된 macro program 의 이름은 getCurProgramName()으로 return 받을 수 있다
Macro program 을 등록한 후 등록이 잘 되었는지 getCurProgramName()으로 확인한다.

2) Start

프로그램을 작동시키는 명령어로는 executeProgram2(int taskid, const QString& prgName, int cycle) 가 있다, 최초 프로그램이 load 된 후 사용하거나 load 되지 않은 상태에서도 macro program 을 run 시킬 수 있다, task main 은 0, sub 는 1 이다.

program 을 load 한다면 executeProgram2 을 사용하여 동작시킬 수 있다.

```
ex) pCon->executeProgram2(0,"mymacro",-1); // -1 은 연속동작이다.
```

3) Hold

프로그램을 작동시킨 이후 holdProgram()을 통해 일시 정지가 가능하다. 현재 진행 중인 스텝에서 즉시 정지한다.

```
ex) pCon->holdProgram();
```

4) Continue

프로그램을 일시 정지 혹은 정지 시킨 상태에서 continueProgram()을 통해 다시 작동이 가능하다. 만약 setCurStep(int taskid, int istep) 을 사용 후 특정 step 부터 Macro 를 동작시키고 싶을 때 start 에 해당하는 함수를 사용하지 않고 continueProgram()을 사용하면 지정한 step 부터 동작시킬 수 있다.

```
ex) pCon->continueProgram();
```

5) Abort

프로그램 정지는 abortProgram(int taskid) 을 통해 가능하며 hold 와의 차이점은 동작 중인 Step 이 완료된 이후 정지하는 것이다.

```
ex) int task = 0;
pCon->abortProgram(task);
```

6) Speed

프로그램의 실행 속도는 setSpeed(float speed) 명령어를 사용하여 설정할 수 있으며, 실행 속도를 확인하고 싶은 경우, getOverrideSpeed() 명령어를 통해 현재 속도를 백분율로 확인할 수 있다.

```
ex) float speed = 10.0 // 0~100%로 speed 가 설정.
pCon->setSpeed(speed);
```

7) Step run

Step 동작을 실행시키기 위해 아래 api 를 사용한다.

setCurStep(int taskid, int istrp) 현재 step 위치를 이동한다.

setCurStep 후 runNextStep 시 현재 step 을 동작시킨다.

runNxtStep(int taskid) 현재 step 에서 다음 step 을 동작 시킨다.

runNxtStepOver(int tsakid) 현재 step 에서 다음 step 을 동작 시킨다, 단 call program 은 모든 step 동작을 완료한다.

runToNextMotionStep() 현재 step 에서 가장 가까운 motion 명령이 있는 다음 step 을 동작시킨다. Main taks 만 동작 된다.

runBackStep(int taskid) 현재 step 바로 전 step 을 동작시킨다.

runBackStepOver(int taskid) 현재 step 바로 전 step 을 동작시킨다, 단 call program 은 모든 step 동작을 완료한다.

8) Program run count 설정

setProgramRunCount(int taskid, int count)를 사용하여 현재 load 된 프로그램의 동작 횟수를 설정할 수 있다, 현재 설정 된 횟수는 getProgramRunCount(int taskid) 를 사용하여 얻는다, 동작 된 프로그램 cycle 은 getCurProgramCycle (int taskid)을 사용한다.

```
ex) int taskid = 0, count = -1;
```

```
setProgramRunCount(int taskid, int count)
```

```
//count 를 -1 로 설정 시 연속동작
```

9) Reset

resetCurProgram(int taskid)는 macro program 1step 으로 이동 한다.

ex)

```
int taskid = 0;
```

```
pCon-> resetCurProgram(taskid)
```

현재 step 을 가장 첫 번째 step 으로 이동 시킨다.

10) Clear

clearCurProgram(int taskid)는 loading 된 macro program 을 unloading 시킨다.

ex)

```
int taskid = 0;
```

```
pCon-> clearCurProgram(taskid)
```

현재 task 를 초기화 한다..

11) Step index 얻기

getRunningStepIndex(int taskid, int* movestep, int* planstep)로 현재 동작 중인 move step index 와 다음에 동작할 plan step index 를 얻을 수 있다.

getRunningMainStepIndex(int *planstep)로 call program 시 가장 main program 의 step 값을 얻을 수 있다.

17. Variables

변수의 타입은 Joint , Trans, Number, String 4 종류가 존재하며 각각 차례대로 int 값 1, 2, 3, 4 에 대응된다. 예를 들어, type 이 1 일 경우 Joint 변수를 의미한다. 이 변수들에 관해 개수, 이름, 정보, 리스트 형태의 4 가지 함수가 존재하며 내용은 다음과 같다.

1) 변수에 대한 공통 사항

Type : 1(joint), 2(trans), 3(number), 4(string)

Name : joint(#p1), trans(p1), number(no1), string(\$str) //변수명은 example

Cn_variant : value.val.joint[32], value.val.trans.p[3], value.val.f32, value.val.str[256]

createNonExist : true //true 변수가 존재하지 않으면 생성

2) 변수 생성

setVariableData(int type, QString &name, cn_variant &value, bool createNonExist)를

사용하여 원하는 타입의 변수를 생성할 수 있다.

```
ex) joint 생성
variant variValue;
variValue.val.joint.joint[0] = 20.1;
variValue.val.joint.joint[1] = 30.4;
QString varName = "#p1";
pCon->setVariableData(1, varName, variValue, true);
```

3) 변수 count 얻기

원하는 타입의 변수 총 개수를 얻기 위한 함수는 getVariableCount(int type, int* count)

가 있다. 원하는 타입을 1, 2, 3, 4 형태로 입력하여 개수를 count 에 받는다.

4) 변수 name 얻기

변수의 이름을 얻는 함수는 getVariableName(int type, int index, QString&

varname) 가 있다. 마찬가지로 타입과 함께 변수의 인덱스를 입력하여 이름을 varname 에 받는다, 통신으로 data 를 받기 때문에 대량의 data 취득 시 통신이 느려질 수 있으므로 getVariableList 를 사용하는 것을 권장.

```
ex)
//var info :
// name - #p1, type - joint, value - 90,0,0
// name - #p2, type - joint, value - 80,0,0
// name - tr1, type - trans, value - 90,0,0
QString varName;
pCon->getVariableName(1,0,varName);
ui->txtOut->append(varName); // #p1

pCon->getVariableName(1,1,varName);
ui->txtOut->append(varName); // #p2

pCon->getVariableName(2,0,varName);
ui->txtOut->append(varName); // tr1
```

5) 변수 삭제

deleteVariable(int type, QString name)에 type 과 변수의 name 을 입력하면 해당하는 변수는 삭제된다.

```
ex)
int type = 1;
QString varName = "#p1";
pCon->deleteVariable(type, varName);
```

6) 변수 data 얻기

변수의 정보를 얻는 함수는 getVariableData(int type, QString& name, cn_variant* value)가 있다. 타입과 함께 정보를 얻고자 하는 변수의 이름을 입력하여 value 에 변수 정보를 받는다.

```
ex)
//robot info : linear 3 axes
//var info : name - #p1, type - joint, value - 90,0,0
int type = 1;
QString varName = "#p1";
```

```

cn_variant varData;
pCon->getVariableData(type, varName, varData);
QString str = QString("%1 : %2,%3,%4").arg(varName).
                                arg(varData.joint.joint[0]).
                                arg(varData.joint.joint[1]).
                                arg(varData.joint.joint[2]);

ui->txtOut->append(str);

```

변수 List 얻기

변수 리스트를 얻는 함수는 `getVariableList(int type, QStringList& varnames, QVector<cn_variant> &vardata)` 가 있다. 이 함수는 입력된 타입의 변수 이름과 정보 전체를 얻는 함수다.

```

ex) joint 변수 list 얻기
//robot info : linear 3 axes
QStringList varNames;
QVector<cn_variant> varDataList;
pCon->getVariableList(1, varNames, varDataList);
int nVar = varNames.size();
for(int i = 0 ; i < nVar; i++)
{
    QString str = QString("%1 : %2,%3,%4").arg(varNames[i]).
                                arg(varDataList.joint.joint[0]).
                                arg(varDataList.joint.joint[1]).
                                arg(varDataList.joint.joint[2]);

    ui->txtOut->append(str);
}

```

7) 변수 위치로 모터 이동(joint, trans)

`moveToPosition(QString varName)`를 사용하여 Joint or Trans 변수의 위치로 이동할 수 있다.

```

ex)
QString varName = "#p1" // #p1 is joint variable , value : 90,0,0,0,0
int retcode = pCon->moveToPosition(varName); // joint 1 : 90, joint 2 ~ 6 : 0 위치로 이동

```

18. Digital Input/Output

Digital Input 은 getDI, Digital Output 은 getDO, setDO 를 통해 관리할 수 있다.

1) getDI

getDI(cn_ui32* vals, int length)는 Input 과 output 모두 cn_ui32(unsigned int)형의 배열과 길이를 이용하는데 총 32*길이 개수의 신호를 전달할 수 있게 하기 위함이다. 예를 들어, cn_ui32 din[4]가 있으면 din[0]은 1 번~32 번, din[1]은 33 번~64 번, din[2]는 65 번~96 번, din[3]은 97 번~128 번과 같은 방식이다.

```
ex) cn_ui32 din[4]
pCon->getDI(din, 4); // din[0] : 1~32, din[1] : 33 ~ 64, din[2] : 65 ~ 96, din[3] : 97 ~ 128
cn_ui32 din1 = din[0] & 0x01;
cn_ui32 din2 = din[0] & 0x02
cn_ui32 din3 = din[0] & 0x04
```

2) getDO

getDO(cn_ui32* vals, int length)로 현재 digital Output 의 상태 값을 얻을 수 있다. digital Output 의 vals 값은 하나의 배열 변수당 32bit 의 port 가 할당된다 할당할 수 있는 배열 변수는 최고 4 개 까지이다, Output 총 합은 32 * 4 하여 128 점 까지 할당할 수 있다

```
ex) cn_ui32 dout[4]
pCon->getDO(dout, 4); // dout[0] : 1~32, dout[1] : 33 ~ 64, dout[2] : 65 ~ 96, dout[3] : 97 ~ 128
cn_ui32 dout1 = dout[0] & 0x01;
cn_ui32 dout2 = dout[0] & 0x02
cn_ui32 dout3 = dout [0] & 0x04
```

3) setDI

setDI(int* di_list, int length) *on/off 할 output no.를 입력하고 배열 길이를 입력하면 output 을 출력할 수 있다. 설정을 원하는 di 값을 on 할 경우, 1 과 같이 플러스 값을, off 할 경우 -1 과 같이 마이너스 값을 di_list 에 설정한다. length 는 di_list 에 들어간 설정 개수로서 총 128 개까지 가능하다.

```
ex)
int signal = 1;    //DI index 1 bit on.
pCon->setDI(signal);
signal = -1;
pCon->setDI(signal);    //DI index 1 bit off.setDO
```

4) setDO

setDO(int* do_list, int length) 함수는 int 형의 signal 을 넘겨주어 사용하는데
범위는

+ - 128 이다. +는 on, -는 off 를 의미하며 숫자는 해당하는 Output 의 숫자다. 즉,
+34 이면 ,34 번째 Output 이 on 이 되는 것이다.

```
ex)
int sdo[5] = {1,2,-3,-4, 5};    // 1,2,5 on, 3,4 off
pCon->setDO(sdo, 5);            // output length
```

19. Analog Input/Output

Analog 는 input/output 1~128 범위의 channel 에 float 값을 설정하거나 설정되어있는 값을 취득하여 동작을 구현할 수 있으며 input 은 getAI, output 은 getAO, setAO 등을 이용한다.

1) getAI

getAI(cn_f32* vals, int length)는 Analog 입력 값을 취득

```
ex)
float val[8];
pCon->getAI(val, 8);
for(int i = 0 ; i < 8; i++ )
{
    ui->txtOut->append(QString::number(val[i]));
}
```

2) getAO

getAO(int channel, float* value, float* sigval)는 Analog 출력 값을 취득

```
ex)
float val[8];
pCon->getAO(val, 8);

for(int i = 0 ; i < 8; i++ )
{
    ui->txtOut->append(QString::number(val[i]));
}
```

3) setAO

setAO(int channel, float value)는 Analog 값을 출력한다.

```
ex)
int channel = 0
float value = 1
pCon->setAO(channel, value); //channel 0 print signal 1
```

```
channel = 1  
value = 1  
pCon->setAO(channel, value); //channel 1 print signal 1
```

*본 매뉴얼에서 제공하지 않는 API 의 경우 help 매뉴얼을 참조.