API version : GL010106-beta

Doosan Robot

M0609 I M0617 I M1013 I M1509
A0509 I A0509s I A0912 I A0912s

API Manual

문서 제.개정 이력

개정번호	제/개정 페이지 및 내용	개정 일자	수정자
1.0	최초 작성 및 배포	2018-06-29	이정우
1.1	신규 기능 추가	2020-05-13	공진혁
1.11	함수 이름 변경(DRL Style)	2020-05-28	공진혁
1.12	헤더 분할(CDRFLEx 추가)	2020-06-08	공진혁

1. 소개

본 API는 두산 로봇 제어기를 T/P 어플리케이션(로봇 제어기에 탑제된 사용자 GUI 프로그램)이 아닌 별도의 사용자 어플리케이션에서 직접 제어하기 위한 함수(기능)으로 구성되어 있으며, C/C++ 프로그래밍 언어로 개발되어 있다.

1.1 설치 가이드

1.1.1 라이브러리 구성

본 API는 3개의 C/C++ 헤더 파일과, 이와 관련된 라이브러리 파일 및 기타 지원(3rdparty) 라이브러 파일로 구성되어 있습니다.

종류	파일명	설명	비고
헤더파일	DRFL.h	라이브러리 함수 정의 파일	
	DRFS.h	라이브러리 관련 구조체 정의 파일	
	DRFC.h	라이브러리 관련 상수 정의 파일	
라이브러리 파일	libDRFL.a	리눅스 계열 라이브러러 파일	
	DRFLWin32.lib DRFLWin32.dll	윈도우 계열 라이브러리 파일	
기타 지원파일	libPocoFoundation.so libPocoFoundation.so.16 libPocoNet.so libPocoNet.so.16	리눅스 계열 종속성 라이브러러 파일	ldConfig 함수를 이용하여 라이브러리 등록
	PocoFoundation.lib PocoFoundation.dll PocoNet.lib PocoNet.dll	윈도우 계열 종속성 라이브러 파일	Microsoft Visual C++ 2010(x86) 재배포 가능 패키지 필요

1.1.2 라이브러리 링크하기

- **리눅스 라이브러리**

/etc/ld.so.conf.d/ 디렉토리 아래에 .conf 확장자(예, DRFLib.conf)를 가진 파일 생성하고, 파알 안에 *.so 파일의 경로를 추가한 후 ldconfig 명령어를 실행하여 해당 라이브러리를 설정한다.

- **윈도우 라이브러리**

윈도우 라이브러리는 Microsoft Visual C++ 2010(x86) 재배포 가능 패키지가 설치되어 있지 않은 경우, 정상적으로 동작하지 않을 수 있습니다.

C++ 프로젝트에서 본 API를 사용하기 위한 프로젝트 설정은 다음과 같다.
가) 헤더파일(include) 경로 설정
[구성 속성]->[C/C++]->[일반]->[추가 포함 디렉터리]
나) 라이브러리(lib) 경로 설정
[구성 속성]->[링커]->[일반]->[추가 라이브러리 디렉터리]

1.1.3 헤더파일 사용하기

라이브러리 함수 관련 헤더 파일(DRFL.h)을 #include 시 컴파일 옵션이 C++ 언어로 설정된 경우에는 CDRFLEx 클래스를 사용하여 프로그래밍 가능하나, C언어인 경우에는 "_함수명" 형식의 전역 함수를 사용하여 프로그래밍 해야 한다.

1.1.4 권장 운용 사양

본 라이브러리가 지원하는 권장 운용 사항은 다음과 같다.

구분	명칭	권장 사양	비고
윈도우	Operating System	Windows 7	
	CPU Acrchietecture	X86(64bi & 32-bit)	
	Compliler	Mircrosoft Visual C++ 2010	

구분	명칭	권장 사양	비고
리눅스	Distros	Ubuntu	
	Kernel and Standard Libraries	Linux 3.x kernel Any GLIBC since 2.0	
	CPU Acrchietecture	X86(64bi & 32-bit)	
	Compliler	GNU GCC 4.x or higher	

1.2 프로그래밍 유의사항

1.2.1 로봇 연결/해제

로봇 제어기와 본 API는 TCP/IP 통신을 통해서 연결됨으로 반드시 연결 수립 과정이 필요하며, 내부 연결 과정 중에 인증 절차가 포함되어 있음으로, 다른 기타 API나 소켓 관련 함수로 연결 시도시 정상적인 연결이 이루어지지 않음으로 반드시 본 API의 연결 관련 함수를 사용해야 한다.

또한 하나의 네트워크에 2개 이상의 로봇 제어기를 사용할 경우에는 T/P 어플리케이션에서 각각의 로봇제어기의 IP 주소를 겹치지 않게 변경하고, 각각의 로봇제어기에 연결 과정을 수행하면 정상적으로 제어할 수 있다.

그리고 본 API는 TCP/IIP 통신을 사용함으로 사용자 어플리케이션이 수행되는 컴퓨터의 성능이나, 네트워크의 부하에 따라 성능이 저하나 기능 오류가 발생할 수 있다.

1.2.2 로봇 초기화

로봇 제어기는 로봇 구동에 필요한 각종 정보를 T/P 어플리케이션의 초기화 과정을 통해 전달받아 처리함으로, 본 API 사용하여 구성된 사용자 어플리케이션은 로봇 운용 상태 정보에서 초기화 완료 여부를 반드시 확인 후 로봇 제어를 시작해야 한다. 초기화 완료 여부는 로봇 연결 관련 함수를 사용하여 정상 접속시 콜백 함수인 TOnMonitoringStateCB 함수나 사용자 호출 함수인 get_robot_state 함수를 통해서 확인할 수 있다.

1.2.3 제어권 관리

로봇 제어기는 안전상 하나의 어플리케이션에만 제어할 수 있도록 구성되어 있음으로, 로봇 초기화 완료 후 제어권 관련 함수인 ManageAccessControl 함수와 TOnChangingAccessControlCB 콜백 함수를 사용하여 제어권 획득 및 이양에 관련된 로직을 사용자 어플리케이션에 구현해야 하며, 제어권 소유시만에 제어 명령을 전달해야 한다. 제어권 없이 제어 명령을 전달할 경우, 모든 제어 명령은 무시되어 처리돼지 않는다.

1.2.4 로봇 운용 모드

로봇 제어기 운용모드는 자동모드와 수동모드 2개가 지원되며, Set/get_robot_mode 함수를 통해서 설정 및 모드를 확인할 수 있다. 자동모드는 당사가 제공하는 로봇 프로그래밍 언어인 DRL로 구성된 프로그램을 자동으로 실행하기 위한 모드이며, 수동모드는 단일 동작(예, 조그 동작)을 수행하기 위한 모드로 안전을 위하여 로봇의 끝단의 TCP 속도가 250mm/sec 제한되는 모드이다. 이와 관련하여 조인트 공간에 대한 로봇 움직임 제어 함수인 movej 명령어를 수동모드로 설정하고 최대 속도로 제어하고자 하는 경우에는, 속도 초과 오류가 발생하고, 로봇의 동작이 정지될 수 있음으로 운용 모드 설정에 유의해야 한다.

1.2.5 로봇 운용 상태

로봇 제어기의 운용 상태 정보는 다음과 같이 총 15가지 상태가 있으며, 예약 사용(11번~14번)을 제외한 모든 상태는 OnMonitoringState 콜백함수나 사용자 호출 함수인 get_robot_state 함수를 통해서 확인할 수 있다

순번	로봇 운용 상태	설명
0	STATE_INITIALIZING	T/P 어플리케이션에 의해서 자동으로 진입하는 상태로 각종 파라미터 설정을 위한 초기화 상태임. 초기화 완료시 자동으로 지령대기 상태로 전환됨.
1	STATE_STANDBY	운용 가능한 기본 상태로, 지령 대기 상태임
2	STATE_MOVING	지령 대기 상태에서 지령 수신 후 로봇 동작시 자동으로 전환되는 지령 동작 상태임. 동작 완료시 자동 지령 대기 상태로 전환됨.
3	STATE_SAFE_OFF	기능 및 동작 오류로 인한 로봇 정지 모드로, 서보 오프 상태(제어 정지 후 모터 및 브레이크 전원을 차단한 상태)임
4	STATE_TEACHING	직접교시 상태
5	STATE_SAFE_STOP	기능 및 동작 오류로 인한 로봇 정지 모드로, 안전 정지 상태(제어 정지만 수행한 상태, 자동모드인 경우 프로그램 일시 정지 상태)
6	STATE_EMERGENCY_STOP:	비상 정지 상태

순번	로봇 운용 상태	설명
7	STATE_HOMMING	홈밍 모드 상태. (로봇을 하드웨어적으로 정렬하는 상태)
8	STATE_RECOVERY	로봇 구동 범위 초과 등과 같은 오류로 인한 로봇 정지시, 구동 범위 이내로 로봇을 이동시키기 위한 복구 모드 상태
9	eSTATE_SAFE_STOP2	eSTATE_SAFE_STOP 상태와 동일하나, 로봇 구동 범위 초과로 인해 복구 모드로 전환해야 하는 상태
10	STATE_SAFE_OFF2	eSTATE_SAFE_OFF 상태와 동일하나, 로봇 구동 범위 초과로 인해 복구 모드로 전환해야 하는 상태
11	STATE_RESERVED1	예약 사용
12	STATE_RESERVED2	예약 사용
13	STATE_RESERVED3	예약 사용
14	STATE_RESERVED4	예약 사용
15	STATE_NOT_READY	로봇제어기 부트업이후 초기화를 위한 대기 상태임. T/P 어플리케이션에 의해 초기화 상태로 전환됨.

1.2.6 로봇 상태 천이

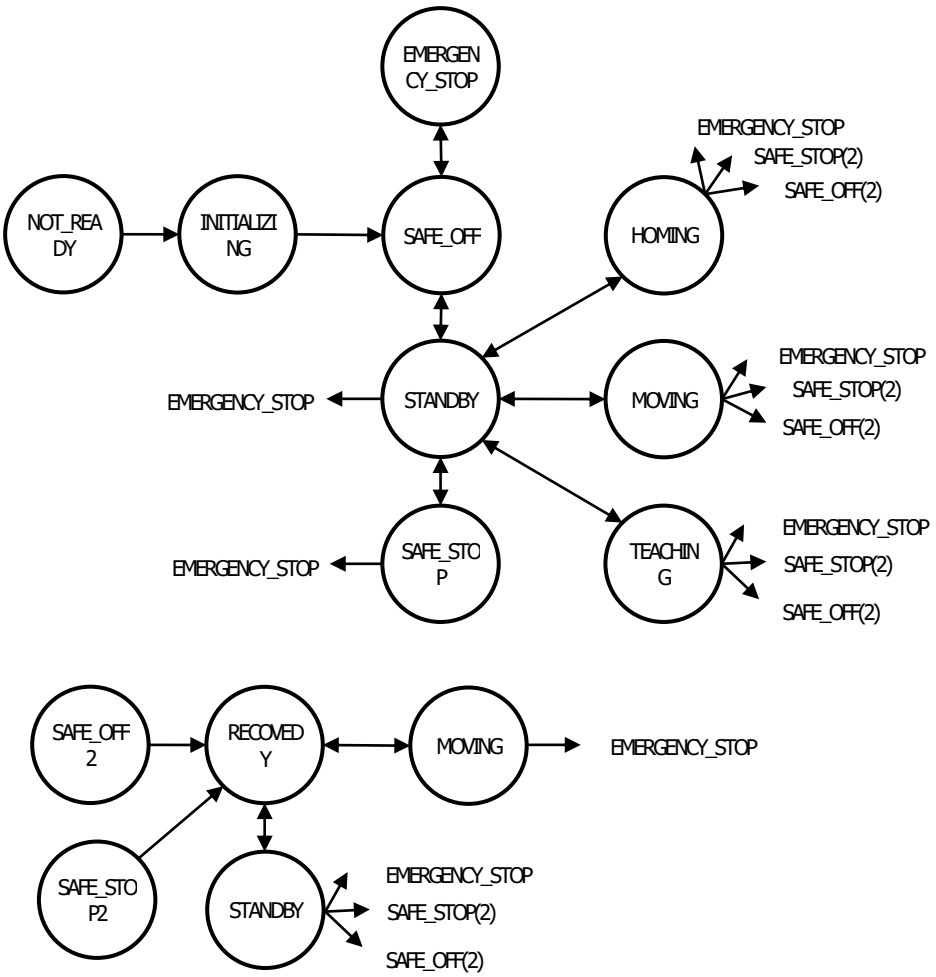
지령 대기 상태(STATE_STANDBY)가 로봇 제어을 위한 기본 준비 상태이며, 사용자로부터 제어 명령 수신시 자동으로 STATE_HOMING 상태나, STATE_MOVING 상태 혹은 STATE_TEACHING 상태로 전환하면서 동작을 수행하고, 오류없이 동작이 완료되면 다시 STATE_STANDBY 상태로 전환하여 사용자 명령을 대기하게 된다.

EMERGENCY_STOP 상태는 초기화(STATE_INITIALIZING) 상태를 제외한 어느 상태에서도 E/M 버튼에 의해서 전환되어 로봇이 정지하게 되며, 로봇 제어기 내부적으로 기능이나 동작 오류 발생시에도 SAFE_OFF 상태(모터 및 브레이크 전원 차단)나, SAFE_STOP 상태(제어 정지)로 자동 전환되어 로봇이 정지하게 된다.

이외에 안전상 사용자에 의해서 직접 상태가 전환되어야 하는 기능은 다음과 같으며, SetRobotControl 함수에 의해서 이 기능을 수행할 수 있다.

순번	로봇 상태 제어 명령	설명
0	CONTROL_INIT_CONFIG	STATE_NOT_READY 상태에서 STATE_INITIALIZING 상태로 전환하는 기능을 수행하며, T/P 어플리케이션만이 이 기능을 수행함.
1	CONTROL_ENABLE_OPERATION	STATE_INITIALIZING상태에서 STATE_STANDBY 상내로 전환하는 기능을 수행하며 T/P 어플리케이션만이 이 기능을 수행함
2	CONTROL_RESET_SAFET_STOP	STATE_SAFE_STOP 상태에서 STATE_STANDBY 상태로 전환하는 기능을 수행함. 자동모드 일 경우, 프로그램 재시작 여부를 설정할 수 있음..
3	CONTROL_RESET_SAFET_OFF	STATE_SAFE_OFF 상태에서 STATE_STANDBY 상태로 전환하는 기능을 수행함,
4	CONTROL_RECOVERY_SAFE_STOP	STATE_SAFE_STOP2 상태에서 S/W 적으로 STATE_RECOVERY 상태로 전환하는 기능을 수행함.
5	CONTROL_RECOVERY_SAFE_OFF	STATE_SAFE_OFF2 상태에서 S/W 적으로 STATE_ RECOVERY 상태로 전환하는 기능을 수행함.
6	CONTROL_RECOVERY_BACKDRIVE	STATE_SAFE_OFF2 상태에서 H/W 적으로 STATE_ RECOVERY 상태로 전환하는 기능을 수행함. STATE_STANDBY 상태로 전환할 수 없고 로봇 제어기 전원을 재부팅해야 함.
7	CONTROL_RESET_RECOVERY	STATE_ RECOVERY 상태에서 STATE_STANDBY 상태로 전환하는 기능을 수행함.

SAFE_OFF상태는 일반적으로 서보온에 해당하는 RESET_SAFE_OFF 명령에 의해 지령대기 상태(STATE_STANDBY)로 전환되며, SAFE_STOP 상태는 RESET_SAFE_STOP 사용자 명령에 의해 지령대기 상태(STATE_STANDBY)로 전환된다. 또한 로봇의 제한 한계치를 넘어서는 오류가 발생할 경우, SAFE_OFF2 상태(모터 및 브레이크 전원 차단)나, SAFE_STOP2 상태(제어 정지)로 전환되는데, 이경우에는 RECOVERY 모드로 전환하여 제한 한계치안으로 로봇을 이동 후 RESET_RECOVERY 명령으로 지령대기 상태(STATE_STANDBY)로 전환해야 오류 발생 없이 정상적으로 로봇 제어를 수행할 수 있다.



1.2.7 프로그램 실행 및 종료

프로그램이 내/외부 오류로 인한 종료 및 정상 종료인 경우에도, 반드시 프로그램 정지(drl_stop) 명령을 수행해야 하며, 프로그램이 내부적으로 완전히 종료 및 정리되는데 다소 시간이 필요함으로, 프로그램 종료에 관한 콜백 함수(TOnProgramStoppedCB)에서 프로그램의 종료 여부를 반드시 확인한 후 필요시 프로그램을 재시작해야 한다.

또한, 프로그램을 가상 로봇 시스템으로 설정하고 프로그램 실행 후 종료하면 자동적으로 실제 로봇 시스템으로 변경됨으로 이에 유의해야 한다.

1.2.8 로봇 안전 설정 기능 제한

로봇 동작(TOOL, TCP 등) 및 안전(안전영역, 안전 정지, 안전 I/O 등)과 관련된 설정 기능은 본 API에서 제공하지 않으며, T/P 어플리케이션에서 직접 설정해야 한다. 로봇 제어기 부팅시 초기화 과정이 T/P 어플리케이션에서만 수행되는 것으로 제한되어 있음으로 로봇 안전 설정은 반드시 T/P 어플리케이션에서 설정해야 한다.

예외적으로, 모션 제어 명령에 사용되는 TOOL 이나 TCP 설정은 본 API의 기능으로 제공하고 있으나, 이는 T/P 어플리케이션과 연동되지 않음으로, 로봇 제어기 재부팅시나 프로그램 재기동시 반드시 다시 재등록 후 사용해야 함으로, T/P 어플리케이션에서 설정 후 사용하는 것을 권장한다.

이와 관련하여 로봇 설정은 다음과 같은 로봇 정지 상태(STATE_INITIALIZING, STATE_STANDBY, STATE_SAFE_OFF, STATE_EMERGERNCY)에서만 가능하며, 이외에 상태에서 설정을 시도하면 오류가 발생하여 로봇 동작이 정지하면서 그에 관한 로그 및 알람 메시지가 콜백함수(TOnLogAlarmCB)에서 생성된다.

1.2.9 힘제어 기능 제한

힘제어 기능은 당사 로봇 프로그래밍 언어인 DRL을 사용(별도 DRL 프로그래밍 가이드 문서 참조)하여 제공되며, 본 API 기능으로는 직접적으로 제공하지 않는다. 본 API는 TCP/IP 통신 특성상 지연이 발생할 수 있어 사용자가 의도한 대로 힘제어 로직이 적용되지 않을 수 있어 인적 물적 재산 보호를 위하여 안전상 이를 제공하지 않는다.

만약 힘제어 기능을 사용하고자 하는 경우에는 DRL 프로그래밍 작성후 본 API가 제공하고 있는 프로그래밍 제어 함수를 사용하여 자동모드로 실행하여야 한다.

2. 정의(Definition)

2.1 상수 및 열거형 정의

2.1.1 enum.ROBOT_STATE

로봇 제어기의 운용 상태 정보를 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	STATE_INITIALIZING	T/P 어플리케이션에 의해서 자동으로 진입하는 상태로 각종 파라미터 설정을 위한 초기화 상태임.
1	STATE_STANDBY	운용 가능한 기본 상태 지령 대기 상태임
2	STATE_MOVING	지령 대기 상태에서 지령 수신 후 동작시 자동으로 전환되는 지령 동작 상태임. 동작 완료시 자동 지령 대기 상태로 전환됨.
3	STATE_SAFE_OFF	기능 및 동작 오류로 인한 로봇 정지 모드로, 서보 오프 상태(제어 정지 후 모터 및 브레이크 전원을 차단한 상태)임
4	STATE_TEACHING	직접교시 상태
5	STATE_SAFE_STOP	기능 및 동작 오류로 인한 로봇 정지 모드로, 안전 정지 상태(제어 정지만 수행한 상태, 자동모드인 경우 프로그램 일시 정지 상태)
6	STATE_EMERGENCY_STOP:	비상 정지 상태
7	STATE_HOMMING	홈밍 모드 상태(로봇을 하드웨어적으로 정렬하는 상태).
8	STATE_RECOVERY	로봇 구동 범위 초과 등과 같은 오류로 인한 로봇 정지시, 구동 범위 이내로 이동시키기 위한 복구 모드 상태임.
9	eSTATE_SAFE_STOP2	eSTATE_SAFE_STOP 상태와 동일하나, 로봇 구동 범위 초과로 인해 복구 모드로 전환해야 하는 상태
10	STATE_SAFE_OFF2	eSTATE_SAFE_OFF 상태와 동일하나, 로봇 구동 범위 초과로 인해 복구 모드로 전환해야 하는 상태
11	STATE_RESERVED1	예약 사용
12	STATE_RESERVED2	예약 사용

순번	상수명	설명
13	STATE_RESERVED3	예약 사용
14	STATE_RESERVED4	예약 사용
15	STATE_NOT_READY	로봇제어기 부트업이후 초기화를 위한 대기 상태임. T/P 어플리케이션에 의해 초기화 상태로 전환됨.

2.1.2 enum.ROBOT_CONTROL

로봇 제어기의 운용 상태를 전환 및 변경시킬 수 있는 열거형 상수로, 다음과 같이 정의되어 있다

순번	상수명	설명
0	CONTROL_INIT_CONFIG	STATE_NOT_READY 상태에서 STATE_INITIALIZING 상태로 전환하는 기능을 수행하며, T/P 어플리케이션만이 이 기능을 수행함.
1	CONTROL_ENABLE_OPERATION	STATE_INITIALIZING상태에서 STATE_STANDBY 상내로 전환하는 기능을 수행하며 T/P 어플리케이션만이 이 기능을 수행함
2	CONTROL_RESET_SAFET_STOP	STATE_SAFE_STOP 상태에서 STATE_STANDBY 상태로 전환하는 기능을 수행함. 자동모드 일 경우, 프로그램 재시작 여부를 설정할 수 있음..
3	CONTROL_RESET_SAFET_OFF	STATE_SAFE_OFF 상태에서 STATE_STANDBY 상태로 전환하는 기능을 수행함,
4	CONTROL_RECOVERY_SAFE_STOP	STATE_SAFE_STOP2 상태에서 S/W 적으로 STATE_RECOVERY 상태로 전환하는 기능을 수행함.
5	CONTROL_RECOVERY_SAFE_OFF	STATE_SAFE_OFF2 상태에서 S/W 적으로 STATE_ RECOVERY 상태로 전환하는 기능을 수행함.
6	CONTROL_RECOVERY_BACKDRIVE	STATE_SAFE_OFF2 상태에서 H/W 적으로 STATE_ RECOVERY 상태로 전환하는 기능을 수행함. STATE_STANDBY 상태로 전환할 수 없고 로봇 제어기 전원을 재부팅해야 함.
7	CONTROL_RESET_RECOVERY	STATE_ RECOVERY 상태에서 STATE_STANDBY 상태로 전환하는 기능을 수행함.

2.1.3 enum.MONITORING_SPEED

로봇 제어기의 속도 모드를 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	SPEED_NORMAL_MODE	정상 속도 모드.
1	SPEED_REDUCED_MODE	감속 속도 모드

2.1.4 enum.SPEED_MODE

MONITORING_SPEED 열거형 상수와 동일하게 정의되어 있다.

2.1.5 enum.ROBOT_SYSTEM

로봇 제어기의 로봇 운용 시스템을 의미하는 열거형 상수로 다음과 같이 정의되어 있다.

순번	상수명	설명
0	ROBOT_SYSTEM_REAL	실제 로봇 시스템
1	ROBOT_SYSTEM_VIRTUAL	가상 로봇 시스템

2.1.6 enum.ROBOT_MODE

로봇 제어기의 로봇 운용 모드를 의미하는 열거형 상수로 다음과 같이 정의되어 있다.

순번	상수명	설명
0	ROBOT_MODE_MANUAL	수동 모드
1	ROBOT_MODE_AUTONOMOUS	자동 모드
2	ROBOT_MODE_MEASURE	측정 모드(현재는 지원하지 않음)

2.1.7 enum.ROBOT_SPACE

로봇 제어기에서 로봇을 제어하는 좌표 공간을 의미는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	ROBOT_SPACE_JOINT	관절 공간
1	ROBOT_SPACE_TASK	작업 공간

2.1.8 enum.SAFE_STOP_RESET_TYPE

로봇 제어기의 운용 상태가 STATE_SAFE_STOP일 경우, 이를 해제하고, 이후 일련의 동작을 정의하기 위한 열거형 상수로 다음과 같이 정의되어 있다.

순번	상수명	설명
0	SAFE_STOP_RESET_TYPE_DEFAULT	단순 상태 해제(수동모드)
	SAFE_STOP_RESET_TYPE_PROGRAM_STOP	프로그램 종료(자동모드)
1	SAFE_STOP_RESET_TYPE_PROGRAM_RESUME	프로그램 재시작(자동모드)

2.1.9 enum.MANAGE_ACCESS_CONTROL

로봇 제어기의 제어권을 획득 및 변경할 수 열거형 상수로 다음과 같이 정의되어 있다.

순번	상수명	설명
0	MANAGE_ACCESS_CONTROL_FORCE_REQUEEST	제어권 강제 회수 메시지 송신
1	MANAGE_ACCESS_CONTROL_REQUEST,	제어권 이양 요청 메시지 송신
2	MANAGE_ACCESS_CONTROL_RESPONSE_YES	제어권 이양 승락 메시지 송신
3	MANAGE_ACCESS_CONTROL_RESPONSE_NO	제어권 이양 거절 메시지 송신

2.1.10 enum.MONITORING_ACCESS_CONTROL

로봇 제어기에서 제어권 변경시 이를 확인할 수 있는 열거형 상수로 다음과 같이 정의되어 있다.

순번	상수명	설명
0	MONITORING_ACCESS_CONTROL_REQUEST,	제어권 이양 요청 메시지 수신
1	MONITORING_ACCESS_CONTROL_DENY	제어권 이양 거절 메시지 수신
2	MONITORING_ACCESS_CONTROL_GRANT	제어권 획득 메시지 수신
3	MONITORING_ACCESS_CONTROL_LOSS	제어권 손실 메시지 수신

2.1.11 enum.JOG_AXIS

로봇 제어기에서 조그 제어를 수행할 각 축을 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	JOG_AXIS_JOINT_1	로봇의 1번 관절 혹은 축
1	JOG_AXIS_JOINT_2	로봇의 2번 관절 혹은 축
2	JOG_AXIS_JOINT_3	로봇의 3번 관절 혹은 축
3	JOG_AXIS_JOINT_4	로봇의 4번 관절 혹은 축
4	JOG_AXIS_JOINT_5	로봇의 5번 관절 혹은 축
5	JOG_AXIS_JOINT_6	로봇의 6번 관절 혹은 축
6	JOG_AXIS_TASK_X	로봇 TCP의 X 축
7	JOG_AXIS_TASK_Y	로봇 TCP의 Y 축
8	JOG_AXIS_TASK_Z	로봇 TCP의 Z 축
9	JOG_AXIS_TASK_RX	로봇 TCP의 RX 축
10	JOG_AXIS_TASK_RY	로봇 TCP의 RY 축
11	JOG_AXIS_TASK_RZ	로봇 TCP의 RZ 축

2.1.12 enum.JOINT_AXIS

로봇 제어기에서 관절 공간 좌표계 기준으로 로봇의 각 축을 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	JOINT_AXIS_1	로봇의 1번 관절 혹은 축
1	JOINT_AXIS _2	로봇의 2번 관절 혹은 축
2	JOINT_AXIS _3	로봇의 3번 관절 혹은 축
3	JOINT_AXIS _4	로봇의 4번 관절 혹은 축
4	JOINT_AXIS _5	로봇의 5번 관절 혹은 축
5	JOINT_AXIS _6	로봇의 6번 관절 혹은 축

2.1.13 enum.TASK_AXIS

로봇 제어기에서 작업 공간 좌표계 기준으로 로봇의 각 축을 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	TASK_AXIS_X	로봇 TCP의 X 축
1	TASK_AXIS_X	로봇 TCP의 Y 축
2	TASK_AXIS_X	로봇 TCP의 Z축

2.1.14 enum.FORCE_AXIS

로봇 제어기에서 힘 제어를 수행할 경우, 좌표 참조 기준에 대한 정의 기준을 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	FORCE_AXIS_X	x축

순번	상수명	설명
1	FORCE_AXIS_Y	y축
2	FORCE_AXIS_Z	z축
10	FORCE_AXIS_A	x축 회전
11	FORCE_AXIS_B	y축 회전
12	FORCE_AXIS_C	z축 회전

2.1.15 enum.MOVE_REFERENCE

로봇제어기에서 작업 공간 기준으로 모션 제어를 수행할 경우, 이동할 위치에 대한 정의 기준을 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	MOVE_REFERENCE_BASE	로봇 베이스 기준
1	MOVE_REFERENCE_TOOL	로봇 TCP 기준

2.1.16 enum.MOVE_MODE

로봇 제어기에서 모션 제어를 수행할 경우, 이동할 위치에 대한 표시 방식을 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	MOVE_MODE_ABSOLUTE	절대 좌표
1	MOVE_MODE_RELATIVE	상대 좌표

2.1.17 enum.FORCE_MODE

로봇 제어기에서 힘 제어를 수행할 경우, 힘 제어를 수행할 방법에 대한 표시 방식을 의마하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	FORCE_MODE_ABSOLUTE	절대 좌표
1	FORCE_MODE_RELATIVE	상대 좌표

2.1.18 enum.BLENDING_SPEED_TYPE

로봇 제어기에서 모션 제어 수행시, 각 경유지점에 대한 블렌딩 속도 타입을 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	BLENDING_SPEED_TYPE_DUPLICATE	선행 모션의 속도와 후행 모션의 속도를 중첩하여 처리
1	BLENDING_SPEED_TYPE_OVERRIDE	선행 모션의 속도를 후행 모션의 속도로 오버라이딩 처리

2.1.19 enum.STOP_TYPE

로봇 제어기에서 모션 제어 수행중 이를 정지시킬 수 있는 모션 정지 타입을 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	STOP_TYPE_QUICK_STO,	내부 예약 사용
1	STOP_TYPE_QUICK	빠른 정지(모션 궤적 유지)
2	STOP_TYPE_SLOW	느린 정지(모션 궤적 유지)
3	STOP_TYPE_HOLD	긴급 정지
	STOP_TYPE_EMERGENCY	긴급 정지

2.1.20 enum.MOVEB_BLENDING_TYPE

로봇 제어기에서 MoveB 모션 제어 수행시, 각 경유지점에 대한 블렌딩 모션 타입을 의미하는 열

거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	MOVEB_BLENDING_TYPE_LINE	직선
1	MOVEB_BLENDING_TYPE_CIRLCE	원호

2.1.21 enum.SPLINE_VELOCITY_OPTION

로봇 제어기에서 Spline 모션 제어 수행시, 각 경유 지점의 속도 제어 옵션를 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	SPLINE_VELOCITY_OPTION_DEFAULT	변속 모션
1	SPLINE_VELOCITY_OPTION_CONST	등속 모션

2.1.22 enum.GPIO_CTRLBOX_DIGITAL_INDEX

로봇 제어기의 컨트롤 박스에 장착된 GPIO 디지털 입출력 단자를 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	GPIO_CTRLBOX_DIGITAL_INDEX_1	컨트롤 박스 GPIO 1번 입출력 포트
1	GPIO_CTRLBOX_DIGITAL_INDEX_2	컨트롤 박스 GPIO 2번 입출력 포트
2	GPIO_CTRLBOX_DIGITAL_INDEX_3	컨트롤 박스 GPIO 3번 입출력 포트
3	GPIO_CTRLBOX_DIGITAL_INDEX_4	컨트롤 박스 GPIO 4번 입출력 포트
4	GPIO_CTRLBOX_DIGITAL_INDEX_5	컨트롤 박스 GPIO 5번 입출력 포트
5	GPIO_CTRLBOX_DIGITAL_INDEX_6	컨트롤 박스 GPIO 6번 입출력 포트
6	GPIO_CTRLBOX_DIGITAL_INDEX_7	컨트롤 박스 GPIO 7번 입출력 포트
7	GPIO_CTRLBOX_DIGITAL_INDEX_8	컨트롤 박스 GPIO 8번 입출력 포트
8	GPIO_CTRLBOX_DIGITAL_INDEX_9	컨트롤 박스 GPIO 9번 입출력 포트

순번	상수명	설명
9	GPIO_CTRLBOX_DIGITAL_INDEX_10	컨트롤 박스 GPIO 10번 입출력 포트
10	GPIO_CTRLBOX_DIGITAL_INDEX_11	컨트롤 박스 GPIO 11번 입출력 포트
11	GPIO_CTRLBOX_DIGITAL_INDEX_12	컨트롤 박스 GPIO 12번 입출력 포트
12	GPIO_CTRLBOX_DIGITAL_INDEX_13	컨트롤 박스 GPIO 13번 입출력 포트
13	GPIO_CTRLBOX_DIGITAL_INDEX_14	컨트롤 박스 GPIO 14번 입출력 포트
14	GPIO_CTRLBOX_DIGITAL_INDEX_15	컨트롤 박스 GPIO 15번 입출력 포트
15	GPIO_CTRLBOX_DIGITAL_INDEX_16	컨트롤 박스 GPIO 16번 입출력 포트

2.1.23 enum.GPIO_CTRLBOX_ANALOG_INDEX

로봇 제어기의 컨트롤 박스에 장착된 GPIO 아날로그 입출력 단자를 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	GPIO_CTRLBOX_ANALOG_INDEX_1	컨트롤 박스 GPIO 1번 입출력 포트
1	GPIO_CTRLBOX_ANALOG_INDEX_2	컨트롤 박스 GPIO 2번 입출력 포트

2.1.24 enum.GPIO_ANALOG_TYPE

로봇 제어기의 컨트롤 박스에 장착된 GPIO 아날로그 입출력 단자의 입출력 타입을 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	GPIO_ANALOG_TYPE_CURRENT	전류 입출력
1	GPIO_ANALOG_TYPE_VOLTAGE	전압 입출력

2.1.25 enum.GPIO_TOOL_DIGITAL_INDEX

로봇 끝단에 장착된 GPIO 디지털 입출력 단자를 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	GPIO_TOOL_DIGITAL_INDEX_1	로봇 끝단의 GPIO 1번 입출력 포트
1	GPIO_TOOL_DIGITAL_INDEX_2	로봇 끝단의 GPIO 2번 입출력 포트
2	GPIO_TOOL_DIGITAL_INDEX_3	로봇 끝단의 GPIO 3번 입출력 포트
3	GPIO_TOOL_DIGITAL_INDEX_4	로봇 끝단의 GPIO 4번 입출력 포트
4	GPIO_TOOL_DIGITAL_INDEX_5	로봇 끝단의 GPIO 5번 입출력 포트
5	GPIO_TOOL_DIGITAL_INDEX_6	로봇 끝단의 GPIO 6번 입출력 포트

2.1.26 enum.MODBUS_REGISTER_TYPE

로봇 제어기에서 등록할 수 있는 모드버스 레지스터 타입에 관한 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	MODBUS_REGISTER_TYPE_DISCRETE_INPUTS	Dircrete Input
1	MODBUS_REGISTER_TYPE_COILS	Coils
2	MODBUS_REGISTER_TYPE_INPUT_REGISTER	Input Register
3	MODBUS_REGISTER_TYPE_HOLDING_REGISTER	Holding Register

2.1.27 enum.DRL_PROGRAM_STATE

로봇 제어기에서 프로그램의 실행 상태를 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	DRL_PROGRAM_STATE_PLAY	프로그램 실행 상태
1	DRL_PROGRAM_STATE_STOP	프로그램 정지 상태
2	DRL_PROGRAM_STATE_HOLD	프로그램 일시 정지 상태

2.1.28 enum.PROGRAM_STOP_CAUSE

로봇 제어기에서 프로그램 종료시 종료 이유를 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	PROGRAM_STOP_CAUSE_NORMAL	정상 프로그램 종료
1	PROGRAM_STOP_CAUSE_FORCE	강제 프로그램 종료
2	PROGRAM_STOP_CAUSE_ERROR	내/외부 오류로 인한 프로그램 종료

2.1.29 enum.PATH_MODE

경로 수정 모드 사용 시 경로 모드를 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	PATH_MODE_DPOS	누적량
1	PATH_MODE_DVEL	증분량

2.1.30 enum.CONTRL_MODE

로봇 제어기 제어 모드를 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
3	CONTROL_MODE_POSITION	위치 제어 모드
4	CONTROL_MODE_TORQUE	토크 제어 모드

2.1.31 enum.DATA_TYPE

로봇 제어기에서 모니터링할 변수의 자료형을 의미하는 열거형 상수로, 다음과 같이 정의되어 있

다

순번	상수명	설명
0	DATA_TYPE_BOOL	boolean
1	DATA _TYPE_INT	integer
2	DATA _TYPE_FLOAT	float
3	DATA _TYPE_STRING	string
4	DATA _TYPE_POSJ	posj
5	DATA _TYPE_POSX	posx
6	DATA _TYPE_UNKNOWN	알 수 없는 타입

2.1.32 enum.VARIABLE_TYPE

로봇 제어기에서 모니터링할 변수의 타입을 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	VARIABLE_TYPE_INSTALL	설치 변수
1	VARIABLE_TYPE_GLOBAL	일반 변수

2.1.33 enum.SUB_PROGRAM

로봇 제어기에서 서브 프로그램의 작업을 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	SUB_PROGRAM_DELETE	서브 프로그램 삭제
1	SUB_PROGRAM_SAVE	서브 프로그램 저장

2.1.34 enum.SINGULARITY_AVOIDANCE

특이점 회피 방법을 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	SINGULARITY_AVOIDANCE_AVOID	자동 회피 모드
1	SINGULARITY_AVOIDANCE_STOP	감속 / warning / task 종료
2	SINGULARITY_AVOIDANCE_VEL	속도 가변

2.1.35 enum.MESSAGE_LEVEL

사용자에게 제공할 메시지의 종류를 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	MESSAGE_LEVEL_INFO	정보 알림
1	MESSAGE_LEVEL_WARN	경고 알림
2	MESSAGE_LEVEL_ALARM	에러 알림

2.1.36 enum.POPUP_RESPONSE

팝업 메시지에 대한 사용자 상호작용을 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	POPUP_RESPONSE_STOP	현재 구동 중인 Task 종료
1	POPUP_RESPONSE_RESUME	현재 구동 중인 Task 계속해서 실행

2.1.37 enum.MOVE_HOME

Homming 시 참조할 좌표계에 관한 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	MOVE_HOME_MECHANIC	기계적 홈 위치[0,0,0,0,0,0] 기준
1	MOVE_HOME_USER	사용자 지정 홈 위치 기준

2.2 구조체 정의

2.2.1 struct.SYSTEM_VERSION

로봇 제어기에서 상세 버전 정보를 확인하기 위한 구조체 정보로 다음과 같은 필드로 구성되어 있다.

BYTE#	필드명	데이터 형	값	비고
0	버전 정보	char	-	상위제어기 정보 (32byte)
32	버전 정보	char	-	하위제어기 정보 (32byte)
64	버전 정보	char	-	DRL 버전 번호 (32bytes)
96	버전 정보	char	-	인버터 정보 (32byte)
128	버전 정보	char	-	Safety Baord 정보(32byte)
160	버전 정보	char	-	로봇 시리얼 번호 (32byte)
192	버전 정보	char	-	로봇 모델 번호 (32byte)
224	버전 정보	char		JTS Board번호 (32byte)
256	버전 정보	char		Flange Board 번호 (32byte)

2.2.2 struct.MONITORING_DATA

로봇 제어기에서 로봇 운용 데이터 정보를 확인하기 위한 구조체 정보로 다음과 같은 5개의 운용 정보로 구성되어 있다

BYTE#	필드명	데이터 형	값	비고
0	운용 정보(#1)	-	-	제어관련 정보
2	운용 정보(#2)	-	-	관절공간(Joint Space) 정보
146	운용 정보(#3)	-	-	작업공간(Task Space) 정보
327	운용 정보(#4)	-	-	토크 및 외력 관련 정보
423	운용 정보(#5)	-	-	기타 로봇 관련 입력 정보

운용 정보(#1)은 다음과 같이 현재 로봇의 움직임에 관한 제어 모드 정보로 구성된다.

BYTE#	필드명	데이터 형	값	비고
0	제어 모드	uchar	0x00~0x01	위치제어: 0 토크제어: 1
1	제어 공간	uchar	0x00~0x01	관절각도 공간: 0 작업좌표 공간: 1

운용 정보(#2)는 다음과 같이 관절공간(로봇의 조인트 기반)에서 로봇의 움직임에 관한 제어 입출력 정보로 구성된다.

BYTE#	필드명	데이터 형	값	비고
0	위치 정보	float	-	6개의 현재 조인트 각도 정보

BYTE#	필드명	데이터 형	값	비고
24	위치 정보	float	-	6개의 현재 조인트 각도 정보 (Absolute Encorder)
48	속도 정보	float	-	6개의 현재 조인트 속도 정보
72	오차 정보	float	-	6개의 현재 조인트 오차 정보
96	위치 정보	float	-	6개의 타켓 조인트 위치 정보
120	위치 정보	float	-	6개의 타켓 조인트 속도 정보

운용 정보(#3)은 다음과 같이 작업공간(로봇의 Flange에 장착된Tool 기반)에서 로봇의 움직에 관한 제어 입출력 정보로 구성된다.

BYTE#	필드명	데이터 형	값	비고
0	위치 정보 (Tool)	float	-	6개의 현재 Tool 위치 정보
24	위치 정보 (Flange)	float	-	6개의 현재 Flange위치 정보
48	속도 정보	float	-	6개의 현재 Tool 속도 정보
72	오차 정보	float	-	6개의 현재 Tool 오차 정보
96	위치 정보	float	-	6개의 타겟 Tool 위치 정보
120	속도 정보	float	-	6개의 타겟 Tool 속도 정보
144	자세 정보	uchar	0x00~0x07	로봇의 자세 정보

✓ 자세 정보는 로봇이 한 지점을 가리킬 수 있는 8가지 자세 정보중 하나이다.
✓ 운용 정보(#3)의 속도 정보는 X, Y, Z, Rx, Ry, Rz의 현재 및 타겟 속도 정보이다.

운용정보(#4)는 다음과 같이 토크제어 모드에서 로봇의 움직임에 관한 토크 및 외력 등의 제어 입출력 정보로 구성된다.

BYTE#	필드명	데이터 형	값	비고
0	토크정보	float	-	6개의 현재 계산된 다이나믹 토크 정보
24	토크정보	float	-	6개의 현재 측정된 JTS 센서 정보
48	외력정보	float	-	6개의 현재 Joint 축별 외력 정보
72	외력정보	float	-	6개의 현재 Tool 기반 외력 정보

운용정보(#5)은 다음과 같이 기타 로봇 관련 입출력 정보로 구성된다.

BYTE#	필드명	데이터 형	값	비고
0	시간 정보	double	-	내부 클락 카운터 정보
8	I/O 정보(#1)	uchar	-	6개의 Digital On/Off정보
14	I/O정보(#2)	uchar		6개의 Digital On/Off정보
20	브레이크 정보	uchar	-	6개의 브레이크 상태 정보
26	버튼 정보	uint		5개의 로봇 버튼 정보

BYTE#	필드명	데이터 형	값	비고
46	전류 정보	float	-	6개 모터의 전류 소모량 정보
70	온도 정보	float		6개 인버터 온도 정보

✓ 브레이크 정보는 0이면 풀린 상태이며, 1이면 잠긴 상태로 로봇을 운용할 수 없는 상태이다.(현재는 지원되지 않는다)
✓ I/O 정보에 대한 설명은 다음과 같다.

I/O 정보	설명
I/O 정보(#1)	로봇 끝단에 부착되어 있는 6의 Digital Input 정보
I/O 정보(#2)	로봇 끝단에 부착되어 있는 6의 Digital Output 정보

✓ 버튼 정보는 6축에 부착되어 있는 5개의 푸쉬 버튼의 On/Off 상태를 의미한다.

2.2.3 struct.MONITORING_DATA_EX

로봇 제어기에서 로봇 운용 데이터 정보를 확인하기 위한 구조체 정보로 다음과 같은 7개의 운용 정보로 구성되어 있다

BYTE#	필드명	데이터 형	값	비고
0	운용 정보(#1)	-	-	제어관련 정보
2	운용 정보(#2)	-	-	관절공간(Joint Space) 정보
146	운용 정보(#3)	-	-	작업공간(Task Space) 정보
327	운용 정보(#4)	-	-	토크 및 외력 관련 정보

BYTE#	필드명	데이터 형	값	비고
439	운용 정보(#5)	-	-	월드 좌표 관련 정보
643	운용 정보(#6)	-	-	유저 좌표 관련 정보
825	운용 정보(#7)	-	-	기타 로봇 관련 입력 정보
919	예약 공간(#1)	-	-	120 바이트 예약 공간
1039	예약 공간(#2)	-	-	120 바이트 예약 공간

운용 정보(#1)은 다음과 같이 현재 로봇의 움직임에 관한 제어 모드 정보로 구성된다.

BYTE#	필드명	데이터 형	값	비고
0	제어 모드	uchar	0x00~0x01	위치제어: 0 토크제어: 1
1	제어 공간	uchar	0x00~0x01	관절각도 공간: 0 작업좌표 공간: 1

운용 정보(#2)는 다음과 같이 관절공간(로봇의 조인트 기반)에서 로봇의 움직임에 관한 제어 입출력 정보로 구성된다.

BYTE#	필드명	데이터 형	값	비고
0	위치 정보	float	-	6개의 현재 조인트 각도 정보
24	위치 정보	float	-	6개의 현재 조인트 각도 정보 (Absolute Encorder)
48	속도 정보	float	-	6개의 현재 조인트 속도 정보

BYTE#	필드명	데이터 형	값	비고
72	오차 정보	float	-	6개의 현재 조인트 오차 정보
96	위치 정보	float	-	6개의 타켓 조인트 위치 정보
120	위치 정보	float	-	6개의 타켓 조인트 속도 정보

운용 정보(#3)은 다음과 같이 작업공간(로봇의 Flange에 장착된Tool 기반)에서 로봇의 움직에 관한 제어 입출력 정보로 구성된다.

BYTE#	필드명	데이터 형	값	비고
0	위치 정보 (Tool)	float	-	6개의 현재 Tool 위치 정보
24	위치 정보 (Flange)	float	-	6개의 현재 Flange위치 정보
48	속도 정보	float	-	6개의 현재 Tool 속도 정보
72	오차 정보	float	-	6개의 현재 Tool 오차 정보
96	위치 정보	float	-	6개의 타겟 Tool 위치 정보
120	속도 정보	float	-	6개의 타겟 Tool 속도 정보
144	자세 정보	uchar	0x00~0x07	로봇의 자세 정보
145	회전 행렬	float	-	3x3 행렬 정보

✓ 자세 정보는 로봇이 한 지점을 가리킬 수 있는 8가지 자세 정보중 하나이다.
✓ 운용 정보(#3)의 속도 정보는 X, Y, Z, Rx, Ry, Rz의 현재 및 타겟 속도 정보이다.

운용정보(#4)는 다음과 같이 토크제어 모드에서 로봇의 움직임에 관한 토크 및 외력 등의 제어 입출력 정보로 구성된다.

BYTE#	필드명	데이터 형	값	비고
0	토크정보	float	-	6개의 현재 계산된 다이나믹 토크 정보
24	토크정보	float	-	6개의 현재 측정된 JTS 센서 정보
48	외력정보	float	-	6개의 현재 Joint 축별 외력 정보
72	외력정보	float	-	6개의 현재 Tool 기반 외력 정보

운용정보(#5)는 다음과 같이 월드 좌표계 공간에서 로봇의 움직임에 관한 제어 입출력 정보로 구성된다.

BYTE#	필드명	데이터 형	값	비고
0	World 좌표계와 Base 좌표계간 관계	float	-	6개의 위치 편차 정보
24	위치 정보 (Tool)	float	-	6개의 현재 Tool 위치 정보
48	위치 정보 (Flange)	float	-	6개의 현재 Flange 위치 정보
72	속도 정보	float	-	6개의 현재 Tool 속도 정보
96	외력 정보	float	-	6개의 현재 Tool 기반 외력 정보
120	위치 정보	float	-	6개의 타겟 Tool 위치 정보
144	속도 정보	float	-	6개의 타겟 Tool 속도 정보

BYTE#	필드명	데이터 형	값	비고
168	회전 행렬	float	-	3x3 행렬 정보

운용정보(#6)은 다음과 같이 유저 좌표계 공간에서 로봇의 움직에 관한 제어 입출력 정보로 구성된다.

BYTE#	필드명	데이터 형	값	비고
0	User 좌표계 ID	float	-	좌표계 ID 정보(101~200)
1	User 좌표계의 Parent 좌표계	float	-	Base 좌표: 0 World 좌표: 2
2	위치 정보 (Tool)	float	-	6개의 현재 Tool 위치 정보
26	위치 정보 (Flange)	float	-	6개의 현재 Flange위치 정보
50	속도 정보	float	-	6개의 현재 Tool 속도 정보
74	외력 정보	float	-	6개의 현재 Tool 기반 외력 정보
98	위치 정보	float	-	6개의 타겟 Tool 위치 정보
122	속도 정보	float	-	6개의 타겟 Tool 속도 정보
146	회전 행렬	float	-	3x3 행렬 정보

운용정보(#7)은 다음과 같이 기타 로봇 관련 입출력 정보로 구성된다.

BYTE#	필드명	데이터 형	값	비고
0	시간 정보	double	-	내부 클락 카운터 정보
8	I/O 정보(#1)	uchar	-	6개의 Digital On/Off정보
14	I/O정보(#2)	uchar		6개의 Digital On/Off정보
20	브레이크 정보	uchar	-	6개의 브레이크 상태 정보
26	버튼 정보	uint		5개의 로봇 버튼 정보
46	전류 정보	float	-	6개 모터의 전류 소모량 정보
70	온도 정보	float		6개 인버터 온도 정보

예약 공간 (#1)에는 운용정보의 확장을 위한 예비 공간이다. 버튼 정보는 운용정보(#7)의 버튼 정보와 동일하나, 자료형 및 버튼 크기가 변경된 정보를 포함하고 있다.

BYTE#	필드명	데이터 형	값	비고
0	버튼 정보	uchar	-	6개의 로봇 버튼 정보

버튼 정보는 운용정보(#7)의 버튼 정보와 동일하나, 자료형 및 버튼 크기가 변경된 정보를 포함하고 있다.

예약 공간(#2)에는 소형 모델 관련하여 다음과 같은 입출력 정보가 제공된다.

BYTE#	필드명	데이터 형	값	비고
0	토크 정보	float	-	6개의 현재 측정된 FTS 센서 정보

BYTE#	필드명	데이터 형	값	비고
24	토크 정보	float	-	6개의 현재 측정된 CS 센서 정보
48	속도 정보	float	-	3개의 현재 측정된 가속도 센서 정보
60	특이점 정보	uchar	-	Minimum Singular Value

✓ 브레이크 정보는 0이면 풀린 상태이며, 1이면 잠긴 상태로 로봇을 운용할 수 없는 상태이다.(현재는 지원되지 않는다)

✓ I/O 정보에 대한 설명은 다음과 같다.

I/O 정보	설명
I/O 정보(#1)	로봇 끝단에 부착되어 있는 6의 Digital Input 정보
I/O 정보(#2)	로봇 끝단에 부착되어 있는 6의 Digital Output 정보

✓ 버튼 정보는 6축에 부착되어 있는 5개의 푸쉬 버튼의 On/Off 상태를 의미한다.

2.2.4 struct.MONITORING_CTRLIO

로봇 제어기에서 컨트롤 박스에 장착되어 있는 I/O의 현재 상태 정보를 확인하기 위한 구조체 정보는 다음과 같은 필드로 구성되어 있다.

BYTE#	필드명	데이터 형	값	비고
0	I/O 정보(#1)	uchar	-	16개의 Digital On/Off정보
16	I/O 정보(#2)	float		2개의 Analog 수치 정보
24	I/O 정보(#3)	uchar	-	3개의 Switch On/Off 정보
27	I/O 정보(#4)	uchar		2개의 Safety On/Off 정보
29	I/O 정보(#5)	uchar		2개의 Encorder On/Off 정보
31	I/O 정보(#6)	uint		2개의 Encorder 수치 정보
39	I/O 정보(#7)	uchar	-	16개의 Digital On/Off정보
55	I/O 정보(#8)	float		2개의 Analog 수치 정보

I/O 정보에 대한 설명은 다음과 같다.

I/O 정보	설명
I/O 정보(#1)	컨트롤 박스에 부착되어 있는 16개의 Digital Input 정보
I/O 정보(#2)	컨트롤 박스에 부착되어 있는 2개의 Analog Input 정보
I/O 정보(#3)	직접교시 버튼 등과 같은 컨트롤 박스 및 T/P에 부착되어 있는 3개의 스위치 상태 정보
I/O 정보(#4)	컨트롤 박스에 부착되어 있는 Safety 관련2개의 Input 정보
I/O 정보(#5)	컨트롤 박스에 부착되어 있는 Encorder 관련2개의 Input 정보

I/O 정보(#6)	컨트롤 박스에 부착되어 있는 Encorder 관련2개의 raw data 정보
I/O 정보(#7)	컨트롤 박스에 부착되어 있는 16개의 Digital Output 정보
I/O 정보(#8)	컨트롤 박스에 부착되어 있는 2개의 Analog Output 정보

2.2.5 struct.MONITORING_CTRLIO_EX

로봇 제어기에서 컨트롤 박스에 장착되어 있는 I/O의 현재 상태 정보를 확인하기 위한 구조체 정보는 다음과 같은 필드로 구성되어 있다.

BYTE#	필드명	데이터 형	값	비고
0	I/O 정보(#1)	-	-	I/O 신호 입력 정보
31	I/O 정보(#2)	-		I/O 신호 출력 정보
57	I/O 정보(#3)	-	-	Encoder 신호 데이터 정보
69	예약 공간	-	-	24바이트 예약 공간

I/O 정보(#1)은 다음과 같이 컨트롤 박스 내 Safety B'd에 부착된 I/O 입력 정보로 구성된다.

BYTE#	필드명	데이터 형	값	비고
0	Digital 신호	uchar	0x00~0x01	16개의 Digital On/Off 정보
16	Analog 신호	float	-	2개의 Analog 수치 정보
24	Switch 신호	uchar	0x00~0x01	3개의 Switch On/Off 정보
27	Safety 신호	uchar	0x00~0x01	2개의 Safety On/Off 정보

BYTE#	필드명	데이터 형	값	비고
29	Analog 모드	uchar	0x00~0x01	2개의 Analog 모드 정보 전류: 0 전압: 1

Switch 신호 정보 직접교시 버튼 등과 같은 컨트롤 박스 및 T/P에 부착되어 있는 3개의 스위치 상태 정보이며, Safety 신호는 컨트롤 박스에 부착되어 있는 Safety Emergency입력 신호와 Protective-Stop 신호 2개의 입력 상태 정보이다.
운용 정보(#2)는 컨트롤 박스 내 Safety B'd에 부착된 I/O 출력 정보로 구성된다.

BYTE#	필드명	데이터 형	값	비고
0	Digital 신호	uchar	0x00~0x01	16개의 Digital On/Off 정보
16	Analog 신호	float	-	2개의 Analog 수치 정보
24	Analog 모드	uchar	0x00~0x01	2개의 Analog 모드 정보 전류: 0 전압: 1

운용 정보(#3)은 다음과 같이 컨트롤 박스 내 Safety B'd에 부착된 Encorder 데이터 정보로 구성된다.

BYTE#	필드명	데이터 형	값	비고
0	Strobe 신호 정보	uchar	0x00~0x01	2개의 Encorder On/Off 정보
2	RAW 데이터 정보	uint	-	2개의 Encorder 수치 정보
10	Reset 신호 정보	uchar	0x00~0x01	2개의 Encorder Reset 정보

예약 공간에는 소형 모델 관련하여 다음과 같은 입출력 정보가 제공된다.

BYTE#	필드명	데이터 형	값	비고
0	프로세스 버튼의 Digital 입력 신호	uchar	0x00~0x01	4개의 Digital On/Off 정보

2.2.6 struct.MONITORING_MODBUS

로봇 제어기에서 등록된 모드버스 I/O 정보가 있을 경우, 모드버스 I/O의 현재 상태 정보를 확인하기 위한 구조체로, 다음과 같은 필드로 구성되어 있다.

BYTE#	필드명	데이터 형	값	비고
0	I/O 개수	ushort	-	Modbus I/O 신호 개수
2	I/O 정보(#1)	-	-	Modbus I/O 신호 정보
…	…	..	..	Modbus I/O 신호 정보
2+(100*34)	I/O 정보(#100)	-	-	Modbus I/O 신호 정보

✓ I/O 정보(#N)는 모드버스 I/O 정보를 식별하기 위한 다음과 같이 구성된다.

BYTE#	필드명	데이터 형	값	비고
0	I/O 이름	char	-	Modbus I/O 신호 이름 (32bytes)
2	I/O 상태	ushort	-	Modbus I/O 신호 값

2.2.7 struct.LOG_ALARM

로봇 제어기에서 내/외부적 요인에 의해서 로봇 동작에 관한 기능 및 동작오류 발생시

이를 확인하기 위한 구조체로, 다음과 같은 필드로 구성되어 있다.

BYTE#	필드명	데이터 형	값	비고
0	로그 레벨	uchar	-	Info: 0 Warnning: 1 Alarm: 2
1	분류 번호	uchar	-	분류 코드
2	로그 번호	uint	-	메시지 번호
6	파라미터(#1)	char	-	예약 공간: 256
262	파라미터(#2)	char	-	예약 공간: 256
518	파라미터(#3)	char	-	예약 공간: 256

분류 및 오류 메시지는 사전에 정의된 내용을 번호를 통해서 전달하며, 필요 시 관련 파라미터를 함께 송부하며 자세한 설명은 로그 및 알람 정의 부분 참조 요망.

2.2.8 struct.MOVE_POSB

로봇 제어기에서 MoveB 모션 제어 수행시, 경유점 정보를 설정하기 구조체로 다음과 같은 필드로 구성되어 있다.

BYTE#	필드명	데이터 형	값	비고
0	위치 정보(#1)	float	-	6개의 Task Space 정보
24	위치 정보(#2)	float	-	6개의 Task Space 정보
48	모션 종류	uchar	0x00~0x03	1st모션과 2nd 모션의 조합 라인: 0 원호: 1
49	곡선 곡률	float		반지름 정보(mm 단위)

✓ MoveB의 경우 라인모션과 원호 모션의 조합으로 이루어진다. 라인 모션의 경우 시작점을 제외한 1개의 경유 지점이, 원호 모션의 경우, 시작점을 제외한 2개의 경유 지점이 필요하다. 즉, 라인 모션인 경우 위치 정보(#2)는 무시된다.

✓ 참조 기준은 베이스 프레임 기준이면 Base 좌표계를, 좌표계 기준이면 Tool을 선택한다.

2.2.9 struct.ROBOT_POSE

로봇 제어기에서 get_current_pose 명령어에 대한 현재 위치정보를 나타내기 위한 구조체 정보로 다음과 같은 필드로 구성되어 있다.

BYTE#	필드명	데이터 형	값	비고
위치 정보	위치 정보	float	6개의 위치 정보	위치 정보

2.2.10 struct.USER_COORDINATE

로봇 제어기에서 get_user_cart_coord 명령어에 대한 현재 사용자 좌표계 정보를 나타내기 위한 구조체 정보로 다음과 같은 필드로 구성되어 있다.

BYTE#	필드명	데이터 형	값	비고
0	요청 ID	uchar	101~120	지정 ID: 101~120
1	참조 기준	uchar	0x00, 0x02	Base 기준 : 0 World 기준 : 2
2	좌표계 정보	float	-	6개의 Task Space 정보

2.2.11 struct.MESSAGE_POPUP

로봇 제어기에서 유형화 어플리케이션을 통해 만들어진 프로그램 실행 시, 유형화 메시지(Popup) 관련 정보를 제공하기 위한 구조체 정보로 다음과 같은 필드로 구성되어 있다.

BYTE#	필드명	데이터 형	값	비고
0	메시지 정보	char	-	256 바이트 문자열
256	메시지 레벨	uchar	-	Message : 0 Warning : 1 Alarm : 2
257	버튼 타입	uchar	-	Resume&Stop 표시 : 0 Stop 표시 : 0

2.2.12 struct.MESSAGE_INPUT

로봇 제어기에서 DRL 프로그램 실행 시 사용자 입력을 받아 처리해야 할 경우, 사용자

입력 관련 정보를 제공하기 위한 구조체 정보로 다음과 같은 필드로 구성되어 있다.

BYTE#	필드명	데이터 형	값	비고
0	메시지 정보	char	-	256 바이트 문자열
256	자료형	uchar	-	int : 0 float : 1 string : 2 bool : 3

2.3 로그 및 알람 정의

2.3.1 LOG_LEVEL

로봇 제어기에서 알람 레벨을 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	LOG_LEVEL_RESERVED	내부 예약 상태
1	LOG_LEVEL_SYSINFO	단순 기능 및 동작 오류에 대한 정보용 메시지
2	LOG_LEVEL_SYSWARN	단순 기능 및 동작 오류로 인한 로봇이 정지된 상태
3	LOG_LEVEL_SYSERROR	안전 이슈나 장치 오류로 인한 로봇이 정지된 상태

2.3.2 LOG_GROUP

로봇 제어기에서 알람 그룹벨을 의미하는 열거형 상수로, 다음과 같이 정의되어 있다.

순번	상수명	설명
0	LOG_GROUP_RESERVED	
1	LOG_GROUP_SYSTEMFMK	하위 제어기(프레임워크)
2	eLOG_GROUP_MOTIONLIB,	하위 제어기(알고리즘)
3	LOG_GROUP_SMARTTP	상위 제어기 프로그램(GUI)
4	LOG_GROUP_INVERTER	로봇 인버터 보드
5	LOG_GROUP_SAFETYCONTROLLER	안전 보드(Safety Controller)

2.3.3 LOG_CODE

로그 및 알람 코드는 DRSC.h 파일에 enum 변수로 정의되어 있으며, 이에 대한 설명은 별도의 로그 및 알람 코드 정의서 문서 참조 요망.

2.5 콜백 함수 정의

2.5.1 TOnMonitoringStateCB

- **기능**

로봇제어기에서 운용 상태 정보 변경시 이를 확인하기 위한 콜백함수이다. 콜백 함수는 특정 이벤트 발생시 자동으로 실행되므로, 콜백 함수 내에서 과도한 실행 시간(50msec 이내)을 요구하는 코드를 작성해서는 안 된다.

- **인수**

인수명	자료형	기본값	설명
eState	enum.ROBOT_STATE	-	상수 및 열거형 정의 참조

- **리턴**

없음

- **예제**

```
void OnMonitoringStateCB(const ROBOT_STATE eState)
{
    switch((unsigned char)eState)
    {
        case STATE_SAFE_OFF:
            // 로봇 제어기 서보온
            drfl.SetRobotControl(CONTROL_RESET_SAFET_OFF);
        break;
        default:
        break;
    }
}

int main()
{
    drfl.SetOnMonitoringState(OnMonitoringStateCB);
}
```

2.5.2 TOnMonitoringDataCB

- **기능**

로봇 제어기의 현재 위치와 같은 로봇 운용 데이터를 확인하기 위한 콜백함수이다. 콜백 함수는 특정 이벤트 발생시 자동으로 실행되므로, 콜백 함수 내에서 과도한 실행 시간(50msec 이내)을 요구하는 코드를 작성해서는 안 된다.

- **인수**

인수명	자료형	기본값	설명
*pData	struct. MONITORING_DATA	-	구조체 정의 참조

- **리턴**

없음

- **예제**

```
void OnMonitoringDataCB(const LPMONITORING_DATA pData)
{
    // 조인트 공간 로봇 위치 데이터 표시
    cout << "joint data "
        << pData->_tCtrl._tJoint._fActualPos[0]
        << pData->_tCtrl._tJoint._fActualPos[1]
        << pData->_tCtrl._tJoint._fActualPos[2]
        << pData->_tCtrl._tJoint._fActualPos[3]
        << pData->_tCtrl._tJoint._fActualPos[4]
        << pData->_tCtrl._tJoint._fActualPos[5] << endl;
}

int main()
{
    drfl.SetOnMonitoringData(OnMonitoringDataCB);
}
```

2.5.3 TOnMonitoringDataExCB

- **기능**

로봇 제어기의 현재 위치와 같은 로봇 운용 확장 데이터를 확인하기 위한 콜백함수이다. 콜백 함수는 특정 이벤트 발생시 자동으로 실행되므로, 콜백 함수 내에서 과도한 실행 시간(100msec 이내)을 요구하는 코드를 작성해서는 안 된다.

- **인수**

인수명	자료형	기본값	설명
*pData	struct. MONITORING_DATA_EX	-	구조체 정의 참조

- **리턴**

없음

- **예제**

```
void OnMonitoringDataExCB(const LPMONITORING_DATA_EX pData)
{
    // 조인트 공간 로봇 위치 데이터 표시
    cout << "joint data "
        << pData->_tCtrl._tJoint._fActualPos[0]
        << pData->_tCtrl._tJoint._fActualPos[1]
        << pData->_tCtrl._tJoint._fActualPos[2]
        << pData->_tCtrl._tJoint._fActualPos[3]
        << pData->_tCtrl._tJoint._fActualPos[4]
        << pData->_tCtrl._tJoint._fActualPos[5] << endl;
}

int main()
{
    Drfl.SetOnMonitoringExData(OnMonitoringDataCB);
}
```

2.5.4 TOnMonitoringCtrlIOCB

- **기능**

로봇 제어기의 Control Box에 설치되어 있는 I/O 상태 데이터를 확인하기 위한 콜백함수이다. 콜백 함수는 특정 이벤트 발생시 자동으로 실행되므로, 콜백 함수 내에서 과도한 실행 시간(50msec 이내)을 요구하는 코드를 작성해서는 안 된다.

- **인수**

인수명	자료형	기본값	설명
*pCtrlIO	struct. MONITORING_CTRLIO	-	구조체 정의 참조

- **리턴**

없음.

- **예제**

```
void OnMonitoringCtrlIOCB(const LPMONITORING_CTRLIO pCtrlIO)
{
    // 디지털 입력 GPIO 상태 데이터 표시
    cout << "gpio data" << endl;
   for (int i = 0; i < NUM_DIGITAL; i++)
        cout << "DI#"<< i << ": " << pCtrlIO->_tInput._iActualDI[i] << endl;
}

int main()
{
    Drfl.SetOnMonitoringCtrlIO(OnMonitoringCtrlIOCB)
}
```

2.5.5 TOnMonitoringCtrlIOExCB

- **기능**

로봇 제어기의 Control Box에 설치되어 있는 I/O 상태 데이터를 확인하기 위한 콜백함수이다. 콜백 함수는 특정 이벤트 발생시 자동으로 실행되므로, 콜백 함수 내에서 과도한 실행 시간(50msec 이내)을 요구하는 코드를 작성해서는 안 된다.

- **인수**

인수명	자료형	기본값	설명
*pCtrlIO	struct. MONITORING_CTRLIO_EX	-	구조체 정의 참조

- **리턴**

없음.

- **예제**

```
void OnMonitoringCtrlIOExCB(const LPMONITORING_CTRLIO_EX pCtrlIO)
{
    // 디지털 입력 GPIO 상태 데이터 표시
    cout << "gpio data" << endl;
   for (int i = 0; i < NUM_DIGITAL; i++)
        cout << "DI#"<< i << ": " << pCtrlIO->_tInput._iActualDI[i] << endl;
}

int main()
{
    Drfl.SetOnMonitoringCtrlIOEx(OnMonitoringCtrlIOExCB)
}
```

2.5.6 TOnMonitoringModbusCB

- **기능**

로봇 제어기에 등록된 모드버스 I/O 상태 데이터를 확인하기 위한 콜백함수이다. 콜백 함수는 특정 이벤트 발생시 자동으로 실행되므로, 콜백 함수 내에서 과도한 실행 시간(50msec 이내)을 요구하는 코드를 작성해서는 안 된다.

- **인수**

인수명	자료형	기본값	설명
*pModbus	struct. MONITORING_MODBUS	-	구조체 정의 참조

- **리턴**

없음.

- **예제**

```
void OnMonitoringModbusCB(const LPMONITORING_MODBUS pModbus)
{
    // 모드버스 IO 상태 데이터 표시
   cout << "modbus data" << endl;
   for (int i = 0; i < pModbus->_iRegCount; i++)
        cout << pModbus->_tRegister[i]._szSymbol <<": "
             << pModbus->_tRegister[i]._iValue<< endl;
}

int main()
{
    Drfl.SetOnMonitoringModbus(OnMonitoringModbusCB)
}
```

2.5.7 TOnLogAlarmCB

- **기능**

로봇 제어기에서 발생하는 모든 알람 및 로그 정보 데이터를 확인하기 위한 콜백함수이다. 콜백 함수는 특정 이벤트 발생시 자동으로 실행되므로, 콜백 함수 내에서 과도한 실행 시간(50msec 이내)을 요구하는 코드를 작성해서는 안 된다.

- **인수**

인수명	자료형	기본값	설명
* pLogAlarm	struct.LOG_ALARM	-	구조체 정의 참조

- **리턴**

없음.

- **예제**

```
void OnLogAlarm(LPLOG_ALARM pLogAlarm)
{
    switch(pLogAlarm->_iGroup)
    {
    case LOG_GROUP_SYSTEMFMK:
        switch(pLogAlarm->_iLevel)
        {
        case LOG_LEVEL_SYSINFO:
            cout << "index(" << pLogAlarm->_iIndex << "), ";
            cout << "param(" << pLogAlarm->_szParam[0]<< ", ";
            cout << "param(" << pLogAlarm->_szParam[1]<< ", ";
            cout << "param(" << pLogAlarm->_szParam[2]<< ")" << endl;
            break;
        default:
            break;
        }
        break;
    default:
        break;
    }
}

int main()
{
    Drfl.SetOnLogAlarm(OnLogAlarmCB)
}
```

2.5.8 TOnMonitoringAccessControlCB

- **기능**

로봇 제어기의 제어권 상태(요청/승락/거절) 변경시 이를 확인하기 위한 콜백함수이다. 콜백 함수는 특정 이벤트 발생시 자동으로 실행되므로, 콜백 함수 내에서 과도한 실행 시간(50msec 이내)을 요구하는 코드를 작성해서는 안 된다.

- **인수**

인수명	자료형	기본값	설명
eAccCtrl	enum. MONITORING_ACCESS_CONTROL	-	상수 및 열거형 정의 참조

- **리턴**

없음

- **예제**

```
void OnMonitoringAccessControlCB(const MONITORING_ACCESS_CONTROL eAccCtrl)
{
   // 제어권 이양 메시지 수신
   case MONITORING_ACCESS_CONTROL_REQUEST:
       // 제어권 이양 거부
      drfl.ManageAccessControl(MANAGE_ACCESS_CONTROL_RESPONSE_NO);
   break;
   defatul:
   break;
}

int main()
{
    Drfl.SetOnMonitoringAccessControl (OnMonitoringAccessControlCB);
}
```

2.5.9 TOnHommingCompletedCB

- **기능**

로봇 제어기가 홈밍 제어 모드에 있을 경우, 홈밍 완료 여부를 확인하기 위한 콜백함수이다. 콜백 함수는 특정 이벤트 발생시 자동으로 실행되므로, 콜백 함수 내에서 과도한 실행 시간(50msec 이내)을 요구하는 코드를 작성해서는 안 된다.

- **인수**

없음.

- **리턴**

없음.

- **예제**

```
void OnHommingCompletedCB()
{
   // 홈밍 완료 메시지 생성
   cout << "homming completed" << endl;
   drfl.Homme(False)

}

int main()
{
    Drfl.SetOnHommingCompleted(OnHommingCompletedCB);
}
```

2.5.10 TOnTpInitializingCompletedCB

- **기능**

로봇 제어기 부팅후, T/P 어플리케이션에 의해 로봇 제어기가 초기화 과정을 수행할 경우, 초기화 완료 여부 위한 콜백함수이다. 콜백 함수는 특정 이벤트 발생시 자동으로 실행되므로, 콜백 함수 내에서 과도한 실행 시간(50msec 이내)을 요구하는 코드를 작성해서는 안 된다.

- **인수**

없음

- **리턴**

없음

- **예제**

```
void OnTpInitializingCompletedCB()
{
    // Tp 초기화 여부 확인후 제어권 요청.
    cout << "tp initializing completed" << endl;
    drfl.ManageAccessControl(MANAGE_ACCESS_CONTROL_REQUEST);
 }

int main()
{
    Drfl.SetOnTpInitializingCompleted(OnTpInitializingCompletedCB);
}
```

2.5.11 TOnMonitoringSpeedModeCB

- **기능**

로봇 제어기의 현재 속도 모드를 확인하기 위한 콜백함수이다. 콜백 함수는 특정 이벤트 발생시 자동으로 실행되므로, 콜백 함수 내에서 과도한 실행 시간(50msec 이내)을 요구하는 코드를 작성해서는 안 된다.

- **인수**

인수명	자료형	기본값	설명
eSpeedMode	enum.MONITORING_SPEED	-	상수 및 열거형 정의 참조

- **리턴**

없음.

- **예제**

```
void OnMonitoringSpeedModeCB(const MONITORING_SPEED eSpdMode)
{
   // 속도 모드 표시
   cout << "speed mode: " << (int)eSpeedMode <<endl;
}

int main()
{
    Drfl.SetOnMonitoringSpeedMode(OnMonitoringSpeedModeCB);
}
```

2.5.12 TOnMasteringNeedCB

- **기능**

로봇 제어기에서 외부 충격으로 인해 로봇의 축이 틀어졌을 경우, 이를 확인하기 위한 콜백함수이다. Home 명령을 통해 로봇의 축을 다시 정렬할 수 있으며, SetOnHommingCompleted 콜백 함수를 통해서 축 정렬이 완료되었음을 확인할 수 있다. 콜백 함수는 특정 이벤트 발생시 자동으로 실행되므로, 콜백 함수 내에서 과도한 실행 시간(50msec 이내)을 요구하는 코드를 작성해서는 안 된다.

- **인수**

없음

- **리턴**

없음.

- **예제**

```
void OnMasteringNeedCB()
{
    //로봇 축 정렬을 위한 홈밍 모드 시작
    drfl.Homme(True)
}

int main()
{
    Drfl.SetOnMasteringNeedCB(TOnMasteringNeedCB);
}
```

2.5.13 TOnProgramStoppedCB

- **기능**

로봇 제어기에서 자동모드로 프로그래밍 실행 도중 오류나 사용자 명령에 의해서 종료되었을 경우, 프로그램 실행이 완전히 종료되었음을 확인하기 위한 콜백함수이다. 콜백함수는 특정 이벤트 발생시 자동으로 실행되므로, 콜백 함수 내에서 과도한 실행 시간(50msec 이내)을 요구하는 코드를 작성해서는 안 된다.

- **인수**

인수명	자료형	기본값	설명
eStopCause	enum. PROGRAM_STOP_CAUSE	-	상수 및 열거형 정의 참조

- **리턴**

없음.

- **예제**

```
void OnProgramStoppedCB(const PROGRAM_STOP_CAUSE eStopCause)
{
    //프로그램을 재시작 가능 상태 표시
}

int main()
{
    Drfl.SetOnProgramStopped(OnProgramStoppedCB);
}
```

2.5.14 TOnDisconnectedCB

- **기능**

로봇 제어기와 연결이 외부 요인이나 혹은 사용자에 의해서 종료되었을 경우, 이를 확인하기 위한 콜백함수이다. 콜백 함수는 특정 이벤트 발생시 자동으로 실행되므로, 콜백 함수 내에서 과도한 실행 시간(50msec 이내)을 요구하는 코드를 작성해서는 안 된다.

- **인수**

없음

- **리턴**

없음.

- **예제**

```
void OnDisconnectedCB()
{
    //로봇 제어기 재접속 및 오류 처리 필요
}

int main()
{
    Drfl.SetOnDisconnected (OnDisconnectedCB);
}
```

2.5.15 TOnTpPopupCB

- **기능**

로봇 제어기에서 사용자 팝업 기능을 사용하였을 때 호출되는 콜백함수이다. 콜백 함수는 특정 이벤트 발생시 자동으로 실행되므로, 콜백 함수 내에서 과도한 실행 시간(50msec 이내)을 요구하는 코드를 작성해서는 안 된다.

- **인수**

인수명	자료형	기본값	설명
tPopup	struct.MESSAGE_POPUP	-	구조체 정의 참조

- **리턴**

없음.

- **예제**

```
void OnTpPopupCB(LPMESSAGE_POPUP tPopup)
{
    //사용자 팝업 생성 시 수행
}

int main()
{
    Drfl.SetOnTpPopup(OnTpPopupCB);
}
```

2.5.16 TOnTpLogCB

- **기능**

로봇 제어기에서 사용자 로그 기능을 사용하였을 때 호출되는 콜백함수이다. 콜백 함수는 특정 이벤트 발생시 자동으로 실행되므로, 콜백 함수 내에서 과도한 실행 시간(50msec 이내)을 요구하는 코드를 작성해서는 안 된다.

- **인수**

인수명	자료형	기본값	설명
strLog	문자열	-	256바이트 문자열

- **리턴**

없음.

- **예제**

```
void OnTpLogCB(const char* strLog)
{
    //사용자 로그 출력 시 수행
}

int main()
{
    Drfl.SetOnTpLog(OnTpLogCB);
}
```

2.5.17 TOnTpGetUserInputCB

- **기능**

로봇 제어기에서 사용자 입력 기능을 사용하였을 때 호출되는 콜백함수이다. 콜백 함수는 특정 이벤트 발생시 자동으로 실행되므로, 콜백 함수 내에서 과도한 실행 시간(50msec 이내)을 요구하는 코드를 작성해서는 안 된다.

- **인수**

인수명	자료형	기본값	설명
tInput	struct.MESSAGE_INPUT	-	구조체 정의 참조

- **리턴**

없음.

- **예제**

```
void OnTpGetUserInputCB(LPMESSAGE_INPUT tInput)
{
    //사용자 입력 받을 시 수행
}

int main()
{
    Drfl.SetOnTpGetUserInput(OnTpGetUserInputCB);
}
```

2.5.18 TOnTpProgressCB

- **기능**

로봇 제어기에서 실행 단계의 정보를 출력하였을 때 호출되는 콜백함수이다. 콜백 함수는 특정 이벤트 발생시 자동으로 실행되므로, 콜백 함수 내에서 과도한 실행 시간(50msec 이내)을 요구하는 코드를 작성해서는 안 된다.

- **인수**

인수명	자료형	기본값	설명
tProgress	struct.MESSAGE_PROGRESS	-	구조체 정의 참조

- **리턴**

  없음.

- **예제**

```
void OnTpProgressCB(LPMESSAGE_PROGRESS tProgress)
{
    //실행 단계 정보 출력 시 수행
}

int main()
{
    Drfl.SetOnTpProgress(OnTpProgressCB);
}
```

3. 함수(Function)

3.1 로봇 연결 함수

3.1.1 CDRFLEx.open_connection

- **기능**

로봇 제어기와 TCP/IP 통신을 사용하여 연결을 수립하기 위한 함수이다. TCP/IP 포트는 내부 고정임으로 별도로 지정할 필요는 없으며, 2개의 이상의 로봇 제어기를 사용할 경우, T/P 어플리케이션에서 IP 주소를 변경해야 한다..

- **인수**

인수명	자료형	기본값	설명
strIpAddr	string	“192.168.137.100”	제어기 IP

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
CDRFLEx drfl;
bool bConnected = drfl.open_connection("192.168.137.100");
if (bConnected) {
    SYSTEM_VERSION tSysVerion = {'\0', };
    Drfl.get_system_version(&tSysVerion)
    cout << "System version: " << tSysVerion._szController << endl;
}
```

3.1.2 CDRFLEx.close_connection

- **기능**

  로봇제어기와 통신 연결을 해제하기 위한 함수이다..

- **인수**

  없음

- **리턴**

  없음.

- **예제**

```
Drfl.close_connection();
```

3.2 로봇 속성 함수

3.2.1 CDRFLEx.get_system_version

- **기능**

로봇 제어기를 구성하고 있는 각각의 서브 시스템에 대한 버전 정보을 확인하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
*pVersion	struct.SYSTEM_VERSION	-	구조체 정의 참조

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
SYSTEM_VERSION tSysVerion = {'\0', };
Drfl.get_system_version(&tSysVerion);
cout << "version: " << tSysVerion._szController << endl;
```

3.2.2 CDRFLEx.get_library_version

- **기능**

  본 API의 버전 정보를 확인하기 위한 함수이다.

- **인수**

  없음

- **리턴**

값	설명
문자열(최대 32바이트)	버전 정보(예, GL:010105)

- **예제**

```
char *lpszLibVersion = get_library_version();
cout << "LibVersion: " << lpszLibVersion << endl;
```

3.2.3 CDRFLEx.get_robot_mode

- **기능**

로봇 제어기의 현재 운용 모드 정보를 확인하기 위한 함수이다. 자동모드는 일련의 순서로 구성된 동작(프로그램)을 자동으로 수행하기 위한 모드이며, 수동모드는 조그와 같은 단일 동작을 수행하기 위한 모드이다.

- **인수**

없음

- **리턴**

값	설명
enum.ROBOT_MODE	상수 및 열거형 정의 참조

- **예제**

```
string strDrlProgram = "\r\n\
loop = 0\r\n\
while loop < 3:\r\n\
 movej(posj(10,10.10,10,10.10), vel=60, acc=60)\r\n\
 movej(posj(00,00.00,00,00.00), vel=60, acc=60)\r\n\
 loop+=1\r\n\
movej(posj(10,10.10,10,10.10), vel=60, acc=60)\r\n";

if (drfl.get_robot_state() == eSTATE_STANDBY) {

   if (drfl.get_robot_mode() == ROBOT_MODE_MANUAL) {
       // 수동 모드
       drfl.jog(JOG_AXIS_JOINT_3, MOVE_REFERENCE_BASE, 60.f);
       sleep(2);
       drfl.jog(JOG_AXIS_JOINT_3, MOVE_REFERENCE_BASE, 0.f);
   }
   else {
      // 자동모드
      ROBOT_SYSTEM eTargetSystem = ROBOT_SYSTEM_VIRTUAL;
      Drfl.drl_start(eTargetSystem, strDrlProgram)
   }
}
```

3.2.4 CDRFLEx.set_robot_mode

- **기능**

로봇 제어기의 현재 운용 모드 정보를 설정하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
eMode	enum.ROBOT_MODE	-	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	성공
1	오류

- **예제**

```
if (Drfl.get_robot_mode() == ROBOT_MODE_MANUAL) {
    // 자동모드 전환
    Drfl.set_robot_mode(ROBOT_MODE_AUTONOMOUS);
}
```

3.2.5 CDRFLEx.get_robot_state

- **기능**

TOnMonitoringStateCB 콜백 함수와 더불어 로봇 제어기의 현재 운용 상태 정보를 확인하기 위한 함수로, 안전상 운용 상태에 따라 사용자가 set_robot_control 함수를 이용하여 운용 상태를 전환해야 한다.

- **인수**

없음

- **리턴**

값	설명
enum.ROBOT_STATE	상수 및 열거형 정의 참조

- **예제**

```
if (Drfl.get_robot_state() == STATE_STANDBY) {
    if (Drfl.get_robot_mode() == ROBOT_MODE_MANUAL) {
       // 수동모드
       Drfl.jog (JOG_AXIS_JOINT_3, MOVE_REFERENCE_BASE, 60.f);
       sleep(2);
       Drfl.jog(JOG_AXIS_JOINT_3, MOVE_REFERENCE_BASE, 0.f);
    }
}
```

3.2.6 CDRFLEx.set_robot_control

- **기능**

로봇 제어기에서 현재 운용 상태를 사용자가 직접 설정 및 전환할 수 있는 함수이다.

- **인수**

인수명	자료형	기본값	설명
eControl	enum.ROBOT_CONTROL	-	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
if (Drfl.get_robot_state () == STATE_SAFE_OFF) {
    // 서보온
    Drfl.set_robot_control(CONTROL_RESET_SAFET_OFF);
}
else if (Drfl.get_robot_state() == STATE_SAFE_OFF2) {
    // 복구 모드 진입
    Drfl.set_robot_control(CONTROL_RECOVERY_SAFE_OFF);
}
```

3.2.7 CDRFLEx.get_robot_system

- **기능**

로봇 제어기에서 현재 운용 로봇 시스템(가상 로봇, 실제 로봇) 정보를 확인하기 위한 함수이다.

- **인수**

없음.

- **리턴**

값	설명
enum.ROBOT_SYSTEM	상수 및 열거형 정의 참조

- **예제**

```
// 현재 로봇 시스템 확인
ROBOT_SYSTEM eRobotSystem = Drfl.get_robot_system();

if(eRobotSystem != ROBOT_SYSTEM_REAL) {
    Drfl.set_robot_system(ROBOT_SYSTEM_REAL);
}
else {
  //do somting …
}
```

3.2.8 CDRFLEx.set_robot_system

- **기능**

로봇 제어기에서 현재 운용 로봇 시스템을 설정 및 변경하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
eRobotSystem	enum.ROBOT_SYSTEM	-	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
ROBOT_SYSTEM eRobotSystem = Drfl.GetRobotSystem();
if(eRobotSystem != ROBOT_SYSTEM_REAL) {
    //자동모드 전환
    Drfl.set_robot_system(ROBOT_SYSTEM_REAL);
}
else {
  //do somting …
}
```

3.2.9 CDRFLEx.get_robot_speed_mode

- **기능**

TOnMonitoringSpeedModeCB 콜백 함수와 더불어 로봇 제어기에서 현재 속도 모드(정상 모드, 감속 모드) 정보를 확인하기 위한 함수이다.

- **인수**

없음

- **리턴**

값	설명
enum.SPEED_MODE	상수형 및 열거형 정의 참조

- **예제**

```
if (Drfl.get_robot_speed_mode()== SPEED_REDUCED_MODE){
   //감속모드 이면 정속 속도 모드로 변경
   Drfl.set_robot_speed_mode(SPEED_NORMAL_MODE);
}
```

3.2.10 CDRFLEx.set_robot_speed_mode

- **기능**

로보 제어기에서 현재 운용 중인 속도 모드를 설정 및 변경하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
eSpeedMode	enum.SPEED_MODE	-	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
if (Drfl.get_robot_speed_mode()== SPEED_REDUCED_MODE){
    //감속모드 이면 정속 속도 모드로 변경
    Drfl.set_robot_speed_mode(SPEED_NORMAL_MODE);
}
```

3.2.11 CDRFLEx.get_robot_state

- **기능**

로봇 제어기에서 현재 자동모드로 실행중인 프로그램 실행 상태 정보를 확인하기 위한 함수이다.

- **인수**

없음

- **리턴**

값	설명
enum.DRL_PROGRAM_STATE	상수 및 열거형 정의 참조

- **예제**

```
if (Drfl.get_robot_state() == DRL_PROGRAM_STATE_PLAY){
   // 프로그램 실행 중지
   Drfl.drl_stop(STOP_TYPE_SLOW)
}
else if (Drfl.get_robot_state() == DRL_PROGRAM_STATE_HOLD) {
   // 프로그램 실행 재개
   Drfl.drl_resume()
}
```

3.2.12 CDRFLEx.set_safe_stop_reset_type

- **기능**

로봇 제어기의 운용 상태 정보가 SAFE_STOP 일 경우, set_robot_control 함수를 이용하여 상태 전환 후 이후 자동으로 실행되는 일련의 동작을 정의하기 위한 함수이다. 로봇 운용 모드가 자동일 경우, 프로그램의 재실행 여부를 정의 및 설정할 수 있으며, 수동 모드일 경우에는 이 설정은 무시된다.

- **인수**

인수명	자료형	기본값	설명
eResetType	enum.SAFE_STOP_RESET_TYPE	SAFE_STOP_RESET_TYPE_DEFAULT	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	성공
1	오류

- **예제**

```
void OnMonitoringStateCB(const ROBOT_STATE eState)
{
    switch((unsigned char)eState)
    {
        case eSTATE_SAFE_STOP:
          if (Drfl.get_robot_mode(ROBOT_MODE_AUTONOMOUS) {
              // 자동모드 이면 상태 전환 후 프로그램 재실행
              Drfl.set_safe_stop_reset_type(SAFE_STOP_PROGRAM_RESUME);
              Drfl.set_robot_control(eCONTROL_RESET_SAFET_STOP);
          }
          else {
              // 수동모드이면 상태만 STATE_STANDBY로 전환
              Drfl.set_robot_control(eCONTROL_RESET_SAFET_STOP);
          }
        break;
        //...
```

3.2.13 CDRFLEx.get_current_pose

- **기능**

로봇 제어기에서 좌표계(관절 공간 또는 작업공간)에 따른 로봇의 각 축별 현재 위치 정보를 확인하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
eSpaceType	enum.ROBOT_SPACE	ROBOT_SPACE_JOINT	상수 및 열거형 정의 참조

- **리턴**

값	설명
struct.ROBOT_POSE	구조체 정의 참조

- **예제**

```
LPROBOT_POSE lpPose = drfl.get_current_pose(ROBOT_SPACE_JOINT);
// 조인트 공간의 현재 로봇 위치 표시
for (int k = 0; k < NUM_JOINT; k++ )
    cout << lpPose->_fPosition[k] << endl;
```

3.2.14 CDRFLEx.get_current_solution_space

- **기능**

로봇 제어기에서 로봇의 자세 정보를 확인하기 위한 함수이다.

- **인수**

없음

- **리턴**

값	설명
unsigned char(0~7)	로봇 자세 정보

- **예제**

```
unsigned char iSolutionSpace = Drfl.get_current_solution_space();
// 로봇의 현재 자세를 유지하면 MoveJ이동
float point[6] = { 10, 10, 10, 10, 10, 10 };
Drfl.movejx(point, iSolutionSpace, 30, 30);
```

3.2.15 CDRFLEx.get_last_alarm

- **기능**

로봇 제어기에서 가장 최근에 발생한 로그 및 알람 코드를 확인하기 위한 로 함수이다.

- **인수**

없음

- **리턴**

값	설명
Struct.LOG_ALARM*	구조체 정의 참조

- **예제**

```
LPLOG_ALARM pLogAlarm= Drfl.get_last_alarm();
switch(pLogAlarm->_iGroup)
{
case LOG_GROUP_SYSTEMFMK:
    switch(pLogAlarm->_iLevel)
    {
    case LOG_LEVEL_SYSINFO:
        cout << "index(" << pLogAlarm->_iIndex << "), ";
        cout << "param(" << pLogAlarm->_szParam[0]<< ", ";
        cout << "param(" << pLogAlarm->_szParam[1]<< ", ";
        cout << "param(" << pLogAlarm->_szParam[2]<< ")" << endl;
        break;
    default:
        break;
    }
    break;
default:
    break;
}
```

3.2.16 CDRFLEx.get_current_posx

- **기능**

현재 태스크 좌표계의 자세와 solution space를 반환한다. 이 때 자세는 eTargetRef를 기준으로 한다.

- **인수**

인수명	자료형	기본값	설명
eTargetRef	enum.COORDINATE_SYSTEM	COORDINATE_SYSTEM_BASE	상수 및 열거형 정의 참조

- **리턴**

값	설명
struct.ROBOT_TASK_POSE	구조체 정의 참조

- **예제**

```
ROBOT_TASK_POSE* result = Drfl.get_current_posx();
float* pos = new float[NUM_TASK];
int sol = result->_iTargetSol;
memcpy(pos, result->_fTargetPos, sizeof(float) * NUM_TASK);
```

3.2.17 CDRFLEx.get_desired_posx

- **기능**

현재 툴의 목표(target) 자세를 반환한다. 이 때 자세는 eTargetRef를 기준으로 한다.

- **인수**

인수명	자료형	기본값	설명
eTargetRef	enum.COORDINATE_SYSTEM	COORDINATE_SYSTEM_BASE	상수 및 열거형 정의 참조

- **리턴**

값	설명
enum.ROBOT_POSE	상수 및 열거형 정의 참조

- **예제**

```
ROBOT_POSE* result = Drfl.get_desired_posx();
for(int i = 0; i < NUM_TASK; i++)
{
    cout << result->_fPosition[i] << endl;
}
```

3.2.18 CDRFLEx.get_solution_space

- **기능**

  solution space의 값을 구한다.

- **인수**

인수명	자료형	기본값	설명
fTargetPos	float[6]	-	6개의 Joint Space 정보

- **리턴**

값	설명
unsigned char(0~7)	자세 정보

- **예제**

```
float p1[6] = {0, 0, 0, 0, 0, 0};
int sol = Drfl.get_solution_space(p1)
cout << sol << endl;
```

3.2.19 CDRFLEx.get_orientation_error

- **기능**

축 방향 eTaskAxis에 대한 임의의 pose fPosition1과 fPosition2 사이의 Orientation error 값을 반환한다.

- **인수**

인수명	자료형	기본값	설명
fPosition1	float[6]	-	6개의 Task Space 정보
fPosition2	float[6]		6개의 Task Space 정보
eTaskAxis	enum.TASK_AXIS		상수 및 열거형 정의 참조

- **리턴**

값	설명
float	Orientation Error 값

- **예제**

```
float x1[6] = {0, 0, 0, 0, 0, 0};
float x2[6] = {10, 20, 30, 40, 50, 60};
float diff = Drfl.get_orientation_error(x1, x2, TASK_AXIS_X);
cout << diff << endl;
```

3.2.20 CDRFLEx.get_control_mode

- **기능**

  현재 제어 모드를 반환한다.

- **인수**

  없음

- **리턴**

값	설명
CONTROL_MODE	상수 및 열거형 정의 참조

- **예제**

```
CONTROL_MODE mode =Drfl.get_control_mode();
cout << mode << endl;
```

3.2.21 CDRFLEx.get_current_rotm

- **기능**

입력된 기준좌표계(eTargetRef)에 해당하는 현재 툴의 회전 행렬을 반환한다.

- **인수**

인수명	자료형	기본값	설명
eTargetRef	enum.COORDINATE_SYSTEM	CORODINATE_SYSTEM_BASE	상수 및 열거형 정의 참조

- **리턴**

값	설명
float[3][3]	Rotation Matrix

- **예제**

```
float(*result)[3] = Drfl.get_current_rotm();
for (int i=0; i<3; i++)
{
    for (int j=0; j<3; j++)
    {
        cout << result[i][j] ;
    }
    cout << endl;
}
```

3.3 콜백함수 등록 함수

3.3.1 CDRFLEx.set_on_monitoring_state

- **기능**

로봇제어기의 운용 상태 정보 변경시 이를 자동으로 확인하기 위한 콜백함수를 등록하기 위한 함수이다. 데이터 변경시 자동으로 실행되어야 하는 기능을 작성할 때 유용하다. .

- **인수**

인수명	자료형	기본값	설명
pCallbackFunc	TOnMonitoringStateCB	-	콜백 함수 정의 참조

- **리턴**

없음

- **예제**

```
void OnMonitoringStateCB(const ROBOT_STATE eState)
{
    switch((unsigned char)eState)
    {
        case STATE_SAFE_OFF:
            // 로봇 제어기 서보온
            drfl.set_robot_control(CONTROL_RESET_SAFET_OFF);
        break;
        default:
        break;
    }
}

int main()
{
    drfl.set_on_monitoring_state(OnMonitoringStateCB);
 }
```

3.3.2 CDRFLEx.set_on_monitoring_data

- **기능**

로봇 제어기의 현재 위치와 같은 로봇 운용 데이터 정보를 자동으로 확인하기 위한 콜백함수를 등록하기 위한 함수이다. 데이터 변경시 자동으로 실행되어야 하는 기능을 작성할 때 유용하다. .

- **인수**

인수명	자료형	기본값	설명
pCallbackFunc	TOnMonitoringDataCB	-	콜백 함수 정의 참조

- **리턴**

없음

- **예제**

```
void OnMonitoringDataCB(const LPMONITORING_DATA pData)
{
    // 조인트 공간 로봇 위치 데이터 표시
    cout << "joint data "
        << pData->_tCtrl._tJoint._fActualPos[0]
        << pData->_tCtrl._tJoint._fActualPos[1]
        << pData->_tCtrl._tJoint._fActualPos[2]
        << pData->_tCtrl._tJoint._fActualPos[3]
        << pData->_tCtrl._tJoint._fActualPos[4]
        << pData->_tCtrl._tJoint._fActualPos[5] << endl;
}

int main()
{
    drfl.set_on_monitoring_data(OnMonitoringDataCB);
}
```

3.3.3 CDRFLEx.set_on_monitoring_data_ex

- **기능**

로봇 제어기의 현재 위치와 같은 로봇 운용 데이터 확장 정보를 자동으로 확인하기 위한 콜백함수를 등록하기 위한 함수이다. 데이터 변경시 자동으로 실행되어야 하는 기능을 작성할 때 유용하다. .

setup_monitoring_version 함수를 이용해 모니터링 데이터 버전을 1로 변경하였을 때 활성화된다.

해당 함수는 M2.5 버전 이상에서만 사용 가능하다.

- **인수**

인수명	자료형	기본값	설명
pCallbackFunc	TOnMonitoringDataExCB	-	콜백 함수 정의 참조

- **리턴**

없음

- **예제**

```
void OnMonitoringDataExCB(const LPMONITORING_DATA_EX pData)
{
    // 사용자 좌표계 로봇 위치 데이터 표시
    cout << "joint data "
        << pData->_tCtrl._tUser._fActualPos[0][0]
        << pData->_tCtrl._tUser._fActualPos[0][1]
        << pData->_tCtrl._tUser._fActualPos[0][2]
        << pData->_tCtrl._tUser._fActualPos[0][3]
        << pData->_tCtrl._tUser._fActualPos[0][4]
        << pData->_tCtrl._tUser._fActualPos[0][5] << endl;
}

int main()
{
    Drfl.set_on_monitoring_data_ex(OnMonitoringDataExCB);
}
```

3.3.4 CDRFLEx.set_on_monitoring_ctrl_io

- **기능**

로봇 제어기의 Control Box에 장착되어 있는 I/O의 현재 상태 정보를 자동으로 확인하기 위한 콜백함수를 등록하기 위한 함수이다. 자동으로 실행되어야 하는 기능을 작성할 때 유용하다.

- **인수**

인수명	자료형	기본값	설명
pCallbackFunc	TOnMonitoringCtrlIOCB	-	콜백 함수 정의 참조

- **리턴**

없음.

- **예제**

```
void OnMonitoringCtrlIOCB(const LPMONITORING_CTRLIO pCtrlIO)
{
    // 디지털 입력 GPIO 상태 데이터 표시
    cout << "gpio data" << endl;
   for (int i = 0; i < NUM_DIGITAL; i++)
        cout << "DI#"<< i << ": " << pCtrlIO->_tInput._iActualDI[i] << endl;
}

int main()
{
    drfl.set_on_monitoring_ctrl_io(OnMonitoringCtrlIOCB)
}
```

3.3.5 CDRFLEx.set_on_monitoring_ctrl_io_ex

- **기능**

로봇 제어기의 Control Box에 장착되어 있는 I/O의 현재 상태 정보를 자동으로 확인하기 위한 콜백함수를 등록하기 위한 함수이다. 자동으로 실행되어야 하는 기능을 작성할 때 유용하다.

- **인수**

인수명	자료형	기본값	설명
pCallbackFunc	TOnMonitoringCtrlIOExCB	-	콜백 함수 정의 참조

- **리턴**

없음.

- **예제**

```
void OnMonitoringCtrlIOExCB(const LPMONITORING_CTRLIO_EX pCtrlIO)
{
    // 디지털 입력 GPIO 상태 데이터 표시
    cout << "gpio data" << endl;
   for (int i = 0; i < NUM_DIGITAL; i++)
        cout << "DI#"<< i << ": " << pCtrlIO->_tInput._iActualDI[i] << endl;
}

int main()
{
    drfl.set_on_monitoring_ctrl_io_ex(OnMonitoringCtrlIOCBEx)
}
```

3.3.6 CDRFLEx.set_on_monitoring_modbus

- **기능**

로봇 제어기에 등록되어 있는 모드버스 I/O의 현재 상태 정보를 자동으로 확인하기 위한 콜백함수를 등록하기 위한 함수이다. 자동으로 실행되어야 하는 기능을 작성할 때 유용하다.

- **인수**

인수명	자료형	기본값	설명
pCallbackFunc	TOnMonitoringModbusCB	-	콜백 함수 정의 참조

- **리턴**

없음.

- **예제**

```
void OnMonitoringModbusCB(const LPMONITORING_MODBUS pModbus)
{
    // 모드버스 IO 상태 데이터 표시
   cout << "modbus data" << endl;
   for (int i = 0; i < pModbus->_iRegCount; i++)
        cout << pModbus->_iRegCount[i]._szSymbol <<": "
             << pModbus->_iRegCount[i]._iValue<< endl;}

int main()
{
    drfl.set_on_monitoring_modbus(OnMonitoringModbusCB)
}
```

3.3.7 CDRFLEx.set_on_log_alarm

- **기능**

로봇 제어기에서 발생하는 모든 알람 및 로그 정보 데이터를 자동으로 확인하기 위한 콜백함수를 등록하기 위한 함수이다. 자동으로 실행되어야 하는 기능을 작성할 때 유용하다

- **인수**

인수명	자료형	기본값	설명
pCallbackFunc	TOnLogAlarmCB	-	콜백 함수 정의 참조

- **리턴**

없음.

- **예제**

```
void OnLogAlarm(LPLOG_ALARM pLogAlarm)
{
    switch(pLogAlarm->_iGroup)
    {
    case LOG_GROUP_SYSTEMFMK:
        switch(pLogAlarm->_iLevel)
        {
        case LOG_LEVEL_SYSINFO:
            cout << "index(" << pLogAlarm->_iIndex << "), ";
            cout << "param(" << pLogAlarm->_szParam[0]<< ", ";
            cout << "param(" << pLogAlarm->_szParam[1]<< ", ";
            cout << "param(" << pLogAlarm->_szParam[2]<< ")" << endl;
            break;
        default:
            break;
        }
        break;
    default:
        break;
    }
}

int main()
{
    drfl.set_on_log_alarm(OnLogAlarmCB)
}
```

3.3.8 CDRFLEx.set_on_tp_popup

- **기능**

DRL에서 tp_popup명령을 사용했을 경우, 팝업 메시지를 확인하기 위한 콜백함수를 등록하기 위한 함수이다. 자동으로 실행되어야 하는 기능을 작성할 때 유용하다

- **인수**

인수명	자료형	기본값	설명
pCallbackFunc	TOnTpPopupCB	-	콜백 함수 정의 참조

- **리턴**

없음.

- **예제**

```
void OnTpPopup(LPMESSAGE_POPUP tPopup)
{
    cout << "Popup Message: " << tPopup->_szText << endl;
    cout << "Message Level: " << tPopup->_iLevel << endl;
    cout << "Button Type: " << tPopup->_iBtnType << endl;
}
int main()
{
    drfl.set_on_tp_popup(OnTpPopupCB)
}
```

3.3.9 CDRFLEx.set_on_tp_log

- **기능**

DRL에서 tp_log명령을 사용했을 경우, 로그 메시지를 확인하기 위한 콜백함수를 등록하기 위한 함수이다. 자동으로 실행되어야 하는 기능을 작성할 때 유용하다

- **인수**

인수명	자료형	기본값	설명
pCallbackFunc	TOnTpLogCB	-	콜백 함수 정의 참조

- **리턴**

없음.

- **예제**

```
void OnTpLog(const char* strLog)
{
    cout << "Log Message: " << strLog << endl;
}
int main()
{
    drfl.set_on_tp_log(OnTpLogCB)
}
```

3.3.10 CDRFLEx.set_on_tp_progress

- **기능**

DRL에서 tp_progress명령을 사용했을 경우, 실행 단계의 정보를 확인하기 위한 콜백함수를 등록하기 위한 함수이다. 자동으로 실행되어야 하는 기능을 작성할 때 유용하다

- **인수**

인수명	자료형	기본값	설명
pCallbackFunc	TOnTpProgressCB	-	콜백 함수 정의 참조

- **리턴**

없음.

- **예제**

```
void OnTpProgress(LPMESSAGE_PROGRESS tProgress)
{
    cout << "Progress cnt : " << tProgress->_iCurrentCount << endl;
    cout << "Current cnt : " << tProgress->_iCurrentCount << endl;
}
int main()
{
    drfl.set_on_tp_progress(OnTpProgressCB)
}
```

3.3.11 CDRFLEx.set_on_tp_get_user_input

- **기능**

DRL에서 tp_get_user_input명령을 사용했을 경우, 사용자 입력을 확인하기 위한 콜백함수를 등록하기 위한 함수이다. 자동으로 실행되어야 하는 기능을 작성할 때 유용하다

- **인수**

인수명	자료형	기본값	설명
pCallbackFunc	TOnTpGetUserInputCB	-	콜백 함수 정의 참조

- **리턴**

없음.

- **예제**

```
void OnTpGetuserInput(LPMESSAGE_INPUT tInput)
{
    cout << "User Input : " << tInput->_szText << endl;
    cout << "Data Type : " << tInput->_iType << endl;
}
int main()
{
    drfl.set_on_tp_get_user_input(OnTpGetUserInputCB)
}
```

3.3.12 CDRFLEx.set_on_monitoring_access_control

- **기능**

로봇 제어기의 제어권 상태(요청/승락/거절 인한) 변경시 이를 자동으로 확인하기 위한 콜백함수를 등록하기 위한 함수이다. 자동으로 실행되어야 하는 기능을 작성할 때 유용하다

- **인수**

인수명	자료형	기본값	설명
pCallbackFunc	TOnMonitoringAccessControlCB	-	콜백 함수 정의 참조

- **리턴**

없음

- **예제**

```
void OnMonitoringAccessControlCB(const MONITORING_ACCESS_CONTROL eAccCtrl)
{
   // 제어권 이양 메시지 수신
   case MONITORING_ACCESS_CONTROL_REQUEST:
       // 제어권 이양 거부
      drfl.ManageAccessControl(MANAGE_ACCESS_CONTROL_RESPONSE_NO);
   break;
   defatul:
   break;
}

int main()
{
    drfl.set_on_monitoring_access_control(OnMonitoringAccessControlCB);
}
```

3.3.13 CDRFLEx.set_on_homming_completed

- **기능**

로봇 제어기가 홈밍 제어 모드에 있을 경우, 홈밍 완료 여부를 자동으로 확인하기 위한 콜백함수를 등록하기 위한 함수이다. 자동으로 실행되어야 하는 기능을 작성할 때 유용하다.

- **인수**

인수명	자료형	기본값	설명
pCallbackFunc	TOnHommingCompletedCB	-	콜백 함수 정의 참조

- **리턴**

없음.

- **예제**

```
void OnHommingCompletedCB()
{
   // 홈밍 완료 메시지 생성
   cout << "homming completed" << endl;
   drfl.Homme(False)

}

int main()
{
    drfl.set_on_homming_completed(OnHommingCompletedCB);
}
```

3.3.14 CDRFLEx.set_on_tp_initializing_completed

- **기능**

로봇 제어기 부팅후, T/P 어플리케이션에 의해 로봇 제어기가 초기화 과정을 수행할 경우, 초기화 완료 여부를 여부를 자동으로 확인하기 위한 콜백함수를 등록하기 위한 함수이다. 자동으로 실행되어야 하는 기능을 작성할 때 유용하다.

- **인수**

인수명	자료형	기본값	설명
pCallbackFunc	TOnTpInitializingCompletedCB	-	콜백 함수 정의 참조

- **리턴**

없음

- **예제**

```
void OnTpInitializingCompletedCB()
{
    // Tp 초기화 여부 확인후 제어권 요청.
    cout << "tp initializing completed" << endl;
    drfl.manage_access_control(MANAGE_ACCESS_CONTROL_REQUEST);
 }

int main()
{
    drfl.set_on_tp_initializing_completed(OnTpInitializingCompletedCB);
}
```

3.3.15 CDRFLEx.set_on_monitoring_speed_mode

- **기능**

로봇 제어기의 현재 속도 모드를 자동으로 확인하기 위한 콜백함수를 등록하기 위한 함수이다. 자동으로 실행되어야 하는 기능을 작성할 때 유용하다.

- **인수**

인수명	자료형	기본값	설명
pCallbackFunc	TOnMonitoringSpeedModeCB	-	콜백 함수 정의 참조

- **리턴**

없음.

- **예제**

```
void OnMonitoringSpeedModeCB(const MONITORING_SPEED eSpdMode)
{
   // 속도 모드 표시
   cout << "speed mode: " << (int)eSpeedMode <<endl;
}

int main()
{
    drfl.set_on_monitoring_speed_mode(OnMonitoringSpeedModeCB);
}
```

3.3.16 CDRFLEx.set_on_mastering_need

- **기능**

로봇 제어기에서 외부 충격으로 인해 로봇의 축이 틀어졌을 경우, 이를 자동으로 확인하기 위한 콜백함수를 등록하기 위한 함수이다. 자동으로 실행되어야 하는 기능을 작성할 때 유용하다

- **인수**

인수명	자료형	기본값	설명
pCallbackFunc	TOnMasteringNeedCB	-	콜백 함수 정의 참조

- **리턴**

없음.

- **예제**

```
void OnMasteringNeedCB()
{
    //로봇 축 정렬을 위한 홈밍 모드 시작
    drfl.Home(True)
}

int main()
{
    drfl.set_on_mastering_need(TOnMasteringNeedCB);
}
```

3.3.17 CDRFLEx.set_on_program_stopped

- **기능**

로봇 제어기에서 자동모드로 프로그래밍 실행 도중 오류나 사용자 명령에 의해서 종료되었을 경우, 이를 자동으로 확인하기 위한 콜백함수를 등록하기 위한 함수이다. 자동으로 실행되어야 하는 기능을 작성할 때 유용하다

- **인수**

인수명	자료형	기본값	설명
pCallbackFunc	TOnProgramStoppedCB	-	콜백 함수 정의 참조

- **리턴**

없음.

- **예제**

```
void OnProgramStoppedCB(const PROGRAM_STOP_CAUSE eStopCause)
{
    //프로그램을 재시작 가능 상태 표시
}

int main()
{
    drfl.set_on_program_stopped(OnProgramStoppedCB);
}
```

3.3.18 CDRFLEx.set_on_disconnected

- **기능**

로봇 제어기와 연결이 외부 요인이나 혹은 사용자에 의해서 종료되었을 경우, 이를 자동으로 확인하기 위한 콜백함수를 등록하기 위한 함수이다. 자동으로 실행되어야 하는 기능을 작성할 때 유용하다

- **인수**

인수명	자료형	기본값	설명
pCallbackFunc	TOnDisconnectedCB	-	콜백 함수 정의 참조

- **리턴**

없음.

- **예제**

```
void OnDisconnectedCB()
{
    //로봇 제어기 재접속 및 오류 처리 필요
}

int main()
{
    drfl.set_on_disconnected(OnDisconnectedCB);
}
```

3.4 제어권 관리 함수

3.4.1 CDRFLEx.manage_access_control

- **기능**

로봇 제어기의 제어권을 요청 메시지를 송신하거나 제어권 요청 메시지 수신시 이에 대한 사용자 응답을 처리하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
eAccessControl	enum.MANAGE_ACCESS_CONTROL	MANAGE_ACCESS_CONTROL_REQUEST	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
void OnMonitoringAccessControlCB(const MONITORING_ACCESS_CONTROL eAccCtrl)
{
   // 제어권 이양 거부 수신
   case MONITORING_ACCESS_CONTROL_DENY:
       // 제어권 없음 표시
   break;
   defatul:
   break;
}

int main()
{
    drfl.set_on_monitoring_access_control(OnMonitoringAccessControlCB);
    // 제어권 이양 요청
    drfl.manage_access_control(MANAGE_ACCESS_CONTROL_REQUEST);

}
```

3.5 기본 제어 함수

3.5.1 CDRFLEx.jog

- **기능**

로봇 제어기에서 로봇의 각 축에 대한 조그 움직임 제어를 수행하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
eJointAxis	enum.JOG_AXIS	-	상수 및 열거형 정의 참조
eMoveReference	enum.MOVE_REFERENCE	-	상수 및 열거형 정의 참조
fVelocity	float	-	조그 속도(%단위) +: 양의 방향 0: 정지 -: 음의 방향

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
// 로봇베이스를 기준으로 + 방향으로 60% 속도로 조그
drfl.jog(JOG_AXIS_JOINT_1, MOVE_REFERENCE_BASE, 60.f);
// 정지
drfl.jog(JOG_AXIS_JOINT_1, MOVE_REFERENCE_BASE, 0.f);
// 로봇베이스를 기준으로 - 방향으로 60% 속도로 조그
drfl.jog(JOG_AXIS_JOINT_1, MOVE_REFERENCE_BASE, -60.f);
// 정지
drfl.jog(JOG_AXIS_JOINT_1, MOVE_REFERENCE_BASE, 0.f);
```

3.5.2 CDRFLEx.move_home

- **기능**

로봇 제어기에서 내/외부 오류로 인하여 로봇의 각 축이 틀어졌을 경우, 로봇의 각 축을 정렬하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
eMode	MOVE_HOME	MOVE_HOME_MECHANIC	상수 및 열거형 정의 참조
bRun	unsigned char	1	0: 정지 1: 동작

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
// Homing 동작 수행
drfl.move_home()
// Homing 동작 정지
drfl.move_home(MOVE_HOME_MECHANIC, (unsigned char)0);
```

3.6 모션 제어 함수

3.6.1 CDRFLEx.movej

- **기능**

로봇제어기에서 로봇을 현재 관절위치에서 목표 관절위치까지 이동시키기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
fTargetPos	float[6]	-	6개 축에 대한 목표 관절 위치
fTargetVel	float	-	속도
fTargetAcc	float	-	가속도
fTargetTime	float	0.f	도달 시간 [sec]
eMoveMode	enum.MOVE_MODE	MOVE_MODE_ABSOLUTE	상수 및 열거형 정의 참조
fBlendingRadius	float	0.f	blending시 radius
eBlendingType	enum.BLENDING_SPEED_TYPE	BLENDING_SPEED_TYPE_DUPLICATE	상수 및 열거형 정의 참조

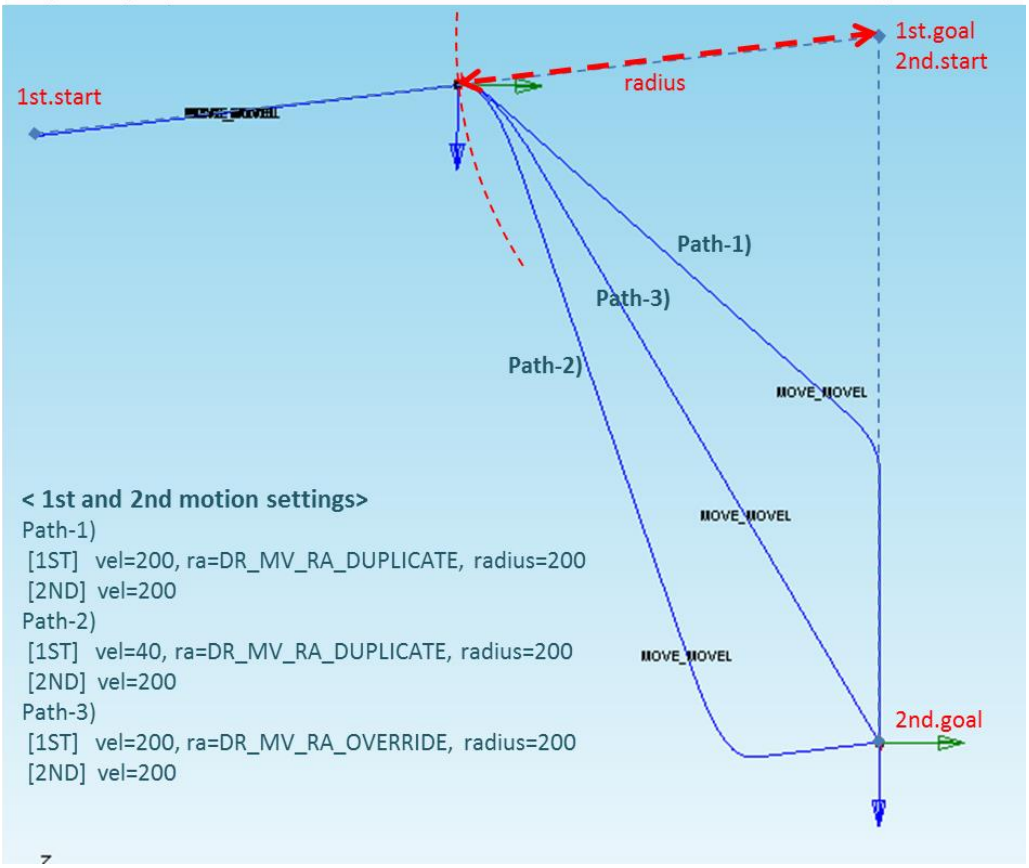
알아두기

・ fTargetTime 지정 시, fTargetVel, fTargetAcc 를 무시하고 fTargetTime 기준으로 처리됩니다.

주의

eBlendingType 이 BLENDING_SPEED_TYPE_DUPLICATE 이고 fBlendingRadius 가 0 보다 큰 조건으로 후속 모션이 블렌딩 될 경우 선행모션의 잔여거리, 속도, 가속도로 결정되는 잔여모션시간이 후행모션의 모션시간보다 큰 경우 후행모션이 먼저 종료된 후 선행모션이 종료될 수 있습니다. 관련한 사항은 아래 이미지를 참고하십시오.

< (Example) Path differences accord. to 1st and 2nd motion settings>



- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
// CASE 1
float q0[6] = { 0, 0, 90, 0, 90, 0 };
float jvel=10;
float jacc=20;
drfl.movej(q0, jvel, jacc);
# 속도 10(deg/sec), 가속도 20(deg/sec2)로 q0 관절각으로 이동
```

```
// CASE 2
float q0[6] = { 0, 0, 90, 0, 90, 0 };
float jTime=5;
drfl.movej(q0, 0, 0, jTime)
# q0 관절각까지 5초의 도착시간을 가지고 이동

// CASE 3
float q0[6] = { 0, 0, 90, 0, 90, 0 };
float q1[6] = {90, 0, 90, 0, 90, 0 };
float jvel=10;
float jacc=20;
float blending_radius=50;
drfl.movej(q0, jvel, jacc, 0, MOVE_MODE_ABSOLUTE, blending_radius);
// q0 관절각으로 이동하며 q0 관절각과 일치하는 위치로부터 50mm의 거리
// 가 될 때 다음 모션을 수행하도록 설정
drfl.movej(q1, jvel, jacc, 0, MOVE_MODE_ABSOLUTE, 0,
BLENDING_SPEED_TYPE_DUPLICATE));
      // 직전모션과 Blending되어 q1 관절각으로 이동
```

3.6.2 CDRFLEx.movel

- **기능**

로봇제어기에서 로봇을 작업 공간 안에서 목표 위치(pos)로 직선을 따라 이동시키기 위한 함수 이다.

- **인수**

인수명	자료형	기본값	설명
fTargetPos	float[6]	-	6개 축에 대한 목적 TCP 위치
fTargetVel	float[2]	-	선속도, 각속도
fTargetAcc	float[2]	-	선가속도, 각가속도
fTargetTime	float	0.f	도달 시간 [sec] * time 지정 시, vel, acc를 무시하고 time 기준으로 처리
eMoveMode	enum.MOVE_MODE	MOVE_MODE_ABSOLUTE	상수 및 열거형 정의 참조
eMoveReference	enum.MOVE_REFERENCE	MOVE_REFERENCE_BASE	상수 및 열거형 정의 참조
fBlendingRadius	float	0.f	blending시 radius
eBlendingType	enum.BLENDING_SPEED_TYPE	BLENDING_SPEED_TYPE_DUPLICATE	상수 및 열거형 정의 참조

알아두기

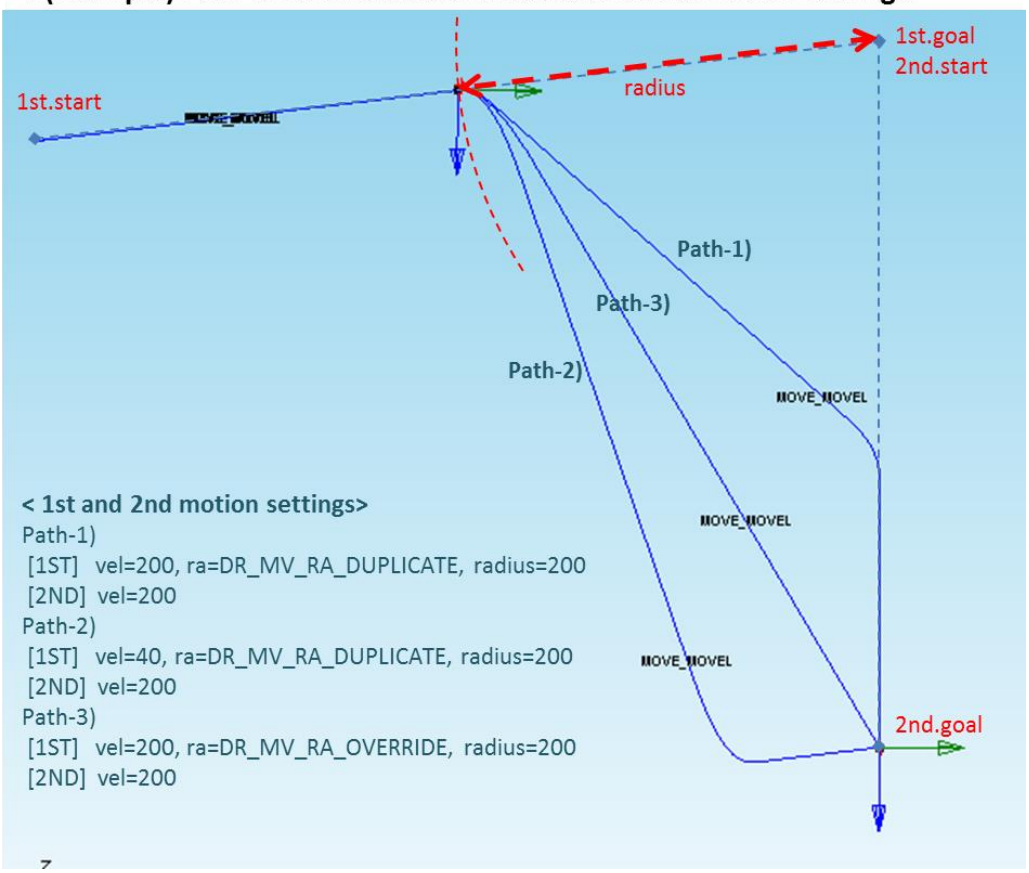
- fTargetVel 에 하나의 인자를 입력한 경우(예를들어, fTargetVel ={30, 0}) 입력된 인자는 모션의 선속도에 대응되며, 각속도는 선속도에 비례하여 결정됩니다.
- fTargetAcc 에 하나의 인자를 입력한 경우(예를들어, fTargetAcc ={60, 0}) 입력된 인자는 모션의 선가속도에 대응되며, 각가속도는 선가속도에 비례하여 결정됩니다.
- fTargetTime 지정 시, fTargetVel, fTargetAcc 를 무시하고 fTargetTime 기준으로 처리됩니다.

주의

eBlendingType 이 BLENDING_SPEED_TYPE_DUPLICATE 이고 fBlendingRadius 가 0 보다 큰 조건으로 후속 모션이 블렌딩 될 경우 선행모션의 잔여거리, 속도, 가속도로 결정되는

잔여모션시간이 후행모션의 모션시간보다 큰 경우 후행모션이 먼저 종료된 후 선행모션이 종료될 수 있습니다. 관련한 사항은 아래 이미지를 참고하십시오.

< (Example) Path differences accord. to 1st and 2nd motion settings>



- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
// CASE 1
float x1[6] = { 559, 434.5, 651.5, 0, 180, 0 };
float tvel = { 50, 50 };
float tacc = { 100, 100 };
drfl.movel(x1, tvel, tacc);
```

```
// 속도 50(mm/sec), 가속도 100(mm/sec2)로 x1 위치로 이동

// CASE 2
float x1[6] = { 559, 434.5, 651.5, 0, 180, 0 };
float tTime = 5;
drfl.movel(x1, 0, 0, tTime);
// x1 위치로 5초의 도착시간을 가지고 이동

// CASE 3
float x1[6] = { 559, 434.5, 651.5, 0, 180, 0 };
float tvel = 50;
float tacc = 100;
drfl.movel(x1, tvel, tacc, 0, MOVE_MODE_RELATIVE, MOVE_REFERENCE_TOOL);
// 시작위치에서 Tool 좌표계 기준으로 x1 만큼의 상대위치로 이동
float x1[6] = { 559, 434.5, 651.5, 0, 180, 0 };
float x2[6] = { 559, 434.5, 251.5, 0, 180, 0 };
float tvel = 50;
float tacc = 100;
float blending_radius = 100;
drfl.movel(x1, tvel, tacc, 0, MOVE_MODE_ABSOLUTE, MOVE_REFERENCE_BASE,
blending_radius);
```

3.6.3 CDRFLEx.movejx

- **기능**

로봇제어기에서 로봇을 관절 공간 안에서 목표 위치로 이동시키기 위한 함수 이다. 목표 위치는 작업공간 상의 위치임으로 movel과 동일하게 이동하지만 로봇의 모션은 관절공간에서 이루어지기 때문에 목표 위치까지 직선경로가 보장되지 않는다. 추가적으로 하나의 작업공간좌표에 대응하는 8가지의 관절조합형태(robot configuration)중 하나를 iSolutionSpace(solution space)에 지정해야 한다.

- **인수**

인수명	자료형	기본값	설명
fTargetPos	float[6]	-	6개 축에 대한 목적 TCP 위치
iSolutionSpace	unsigned char	-	관절조합형태(아래 설명 참조)
fTargetVel	float	-	속도
fTargetAcc	floa	-	가속도
fTargetTime	float	0.f	도달 시간 [sec]
eMoveMode	enum.MOVE_MODE	MOVE_MODE_ABSOLUTE	상수 및 열거형 정의 참조
eMoveReference	enum.MOVE_REFERENCE	MOVE_REFERENCE_BASE	상수 및 열거형 정의 참조
fBlendingRadius	Float	0.f	blending시 radius
eBlendingType	enum.BLENDING_SPEED_TYPE	BLENDING_SPEED_TYPE_DUPLICATE	상수 및 열거형 정의 참조

알아두기

- fTargetTime 을 지정할 경우 fTargetVel, fTargetAcc 를 무시하고 fTargetTime 기준으로 처리된다.
- 상대모션으로 입력하는 경우(eMoveMode = MOVE_MODE_ABSOLUTE), 선행모션에 블렌딩을 사용하는 경우 에러가 발생하므로 movej() 또는 MoveL()을 이용하여 블렌딩하는 것을 권장한다.
- 옵션 eBlendingType 및 fTargetVel / fTargetAcc 에 따른 블렌딩을 수행할 경우 movej(), MoveL() 설명을 참조하십시오.

- **Robot configuration (형태 vs. solution space)**

Solution space	Binary	Shoulder	Elbow	Wrist
0	000	Lefty	Below	No Flip
1	001	Lefty	Below	Flip
2	010	Lefty	Above	No Flip
3	011	Lefty	Above	Flip
4	100	Righty	Below	No Flip
5	101	Righty	Below	Flip
6	110	Righty	Above	No Flip
7	111	Righty	Above	Flip

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
// CASE 1
float x1[6] = { 559, 34.5, 651.5, 0, 180, 0 };
float sol=2;
float jvel=10;
float jacc=20;
drfl.movejx(x1, sol, jvel, jacc);
      // 속도 10(deg/sec), 가속도 20(deg/sec²)으로 x1 및 configuration과 일치하
      // 는 관절각으로 이동

// CASE 2
float x1[6] = { 559, 34.5, 651.5, 0, 180, 0 };
float sol=2;
float jTime=5;
drfl.movejx(x1, sol, 0, 0, jTime);
      // x1 및 configuration과 일치하는 관절각까지 5초의 도착시간을 가지고 이동

// CASE 3
float x1[6] = { 559, 34.5, 651.5, 0, 180, 0 };
float x2[6] = { 559, 434.5, 651.5, 0, 180, 0 };
```

```
float sol=2;
float jvel=10;
float jacc=20;
float blending_radius=50;
drfl.movejx(x1, sol, jvel, jacc, 0, MOVE_MODE_ABSOLUTE, blending_radius);
      // x1 및 configuation과 일치하는 관절각으로 이동하며 x1 위치로부터 50mm
      // 의 거리가 될 때 다음 모션을 수행하도록 설정
drfl.movejx(x2, sol, jvel, jacc, 0, MOVE_MODE_ABSOLUTE, 0,
BLENDING_SPEED_TYPE_DUPLICATE);
      // 직전모션과 Blending되어 x2 및 configuration과 일치하는 관절각으로 이동
```

3.6.4 CDRFLEx.movec

- **기능**

로봇 제어기에서 작업공간을 기준으로 로봇이 현재 위치에서 경유 지점를 지나 목표 위치까지 원호 또는 지정한 각도로 원호를 따라 이동시키기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
fTargetPos[0]	float[6]	-	경유 지점
fTargetPos[1]	float[6]		목표 위치
fTargetVel	float[2]	-	선속도, 각속도
fTargetAcc	float[2]	-	선가속도, 각가속도
fTargetTime	float	0.0	도달 시간 [sec]
eMoveMode	enum.MOVE_MODE	MOVE_MODE_ABSOLUTE	상수 및 열거형 정의 참조
eMoveReference	enum.MOVE_REFERENCE	MOVE_REFERENCE_BASE	상수 및 열거형 정의 참조
fTargetAngle1	float	0.0	angle1
fTargetAngle2	float	0.0	angle2
fBlendingRadius	float	0.0	blending시 radius
eBlendingType	enum.BLENDING_SPEED_TYPE	BLENDING_SPEED_TYPE_DUPLICATE	상수 및 열거형 정의 참조

알아두기

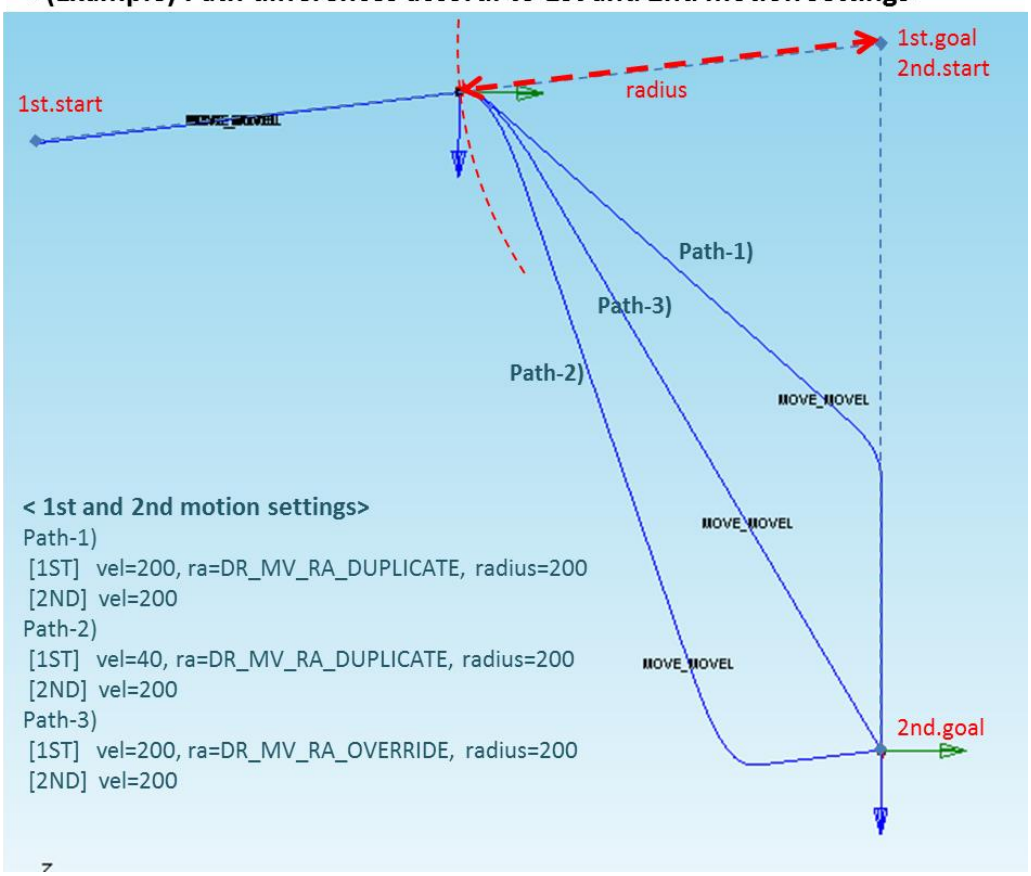
- fTargetVel 에 하나의 인자를 입력한 경우(예를들어, fTargetVel ={30, 0}) 입력된 인자는 모션의 선속도에 대응되며, 각속도는 선속도에 비례하여 결정됩니다.
- fTargetAcc 에 하나의 인자를 입력한 경우(예를들어, fTargetAcc ={60, 0}) 입력된 인자는 모션의 선가속도에 대응되며, 각가속도는 선가속도에 비례하여 결정됩니다.
- fTargetTime 지정 시, fTargetVel, fTargetAcc 를 무시하고 fTargetTime 기준으로 처리됩니다.
- eMoveMode 가 MOVE_MODE_RELATIVE 인 경우 fTargetPos[0] 과 fTargetPos[1] 는 각각 앞 선 위치값에 대한 상대좌표로 정의됩니다. (fTargetPos[0]은 시작점 대비 상대좌표, fTargetPos[1]는 fTargetPos[0]대비 상대좌표)
- fTargetAngle1 이 0 보다 크고, fTargetAngle2 이 0 인 경우 fTargetAngle1 은 Circular path 상의 총 회전각이 적용됩니다.

- fTargetAngle1 과 fTargetAngle2 가 0 보다 큰 경우, fTargetAngle1 은 circular path 상에서 정속으로 이동하는 총 회전각을, fTargetAngle2 는 가속과 감속을 위한 회전 구간의 회전각을 의미합니다. 이때 총 이동각은 fTargetAngle1+ 2 X fTargetAngle2 만큼 circular path 상을 움직입니다.

⚠ 주의

eBlendingType 이 BLENDING_SPEED_TYPE_DUPLICATE 이고 fBlendingRadius 가 0 보다 큰 조건으로 후속 모션이 블렌딩 될 경우 선행모션의 잔여거리, 속도, 가속도로 결정되는 잔여모션시간이 후행모션의 모션시간보다 큰 경우 후행모션이 먼저 종료된 후 선행모션이 종료될 수 있습니다. 관련한 사항은 아래 이미지를 잠고하십시오.

< (Example) Path differences accord. to 1st and 2nd motion settings>



- **리턴**

값	설명
0	오류

값	설명
1	성공

- **예제**

```
// CASE 1
float x1[2][6] = {{559,434.5,651.5,0,180,0}, {559,434.5,251.5,0,180,0}};
float tvel = {50,50}; # 태스크 속도를 50(mm/sec), 50(deg/sec)로 설정
float tacc = {100,100}; # 태스크 가속도를 100(mm/sec2), 100(deg/sec2)로 설정
drfl.movec(x1, tvel, tacc);
// 속도 50(mm/sec), 가속도 100(mm/sec2)로 x1[0]을 경유하여 x1[1]에 이르는
// 원호궤적을 따라 이동

// CASE 2
float x1[2][6] = {{559,434.5,651.5,0,180,0},{559,434.5,251.5,0,180,0}};
float tTime = 5;
drfl.movec(x1, 0, 0, tTime);
// x1[0]을 경유하여 x1[1]까지 5초의 도착시간을 가지고 원호궤적으로 이동

// CASE 3
float x1[2][6] = {{559,434.5,651.5,0,180,0},{559,434.5,251.5,0,180,0}};
float x2[2][6] = {{559,234.5,251.5,0,180,0 },{559,234.5,451.5,0,180,0}};
float tvel = {50,50};
float tacc = {100,100};
float blending_radius = 50;
drfl.movec(x1, tvel, tacc, 0, MOVE_MODE_ABSOLUTE, MOVE_REFERENCE_BASE, 0, 0,
blending_radius);
drfl.movec(x2, tvel, tacc, 0, MOVE_MODE_ABSOLUTE, MOVE_REFERENCE_BASE, 0, 0,
0, BLENDING_SPEED_TYPE_DUPLICATE);
```

3.6.5 CDRFLEx.movesj

- **기능**

로봇 제어기에서 로봇을 현재 위치에서 관절공간의 경유점들을 거쳐 목표위치(마지막 경유점)까지 연결되는 스플라인 곡선경로를 따라 이동시키기 위한 함수이다. 입력된 속도/가속도는 경로 중 최대 속도/가속도를 의미하며 입력되는 경유점의 위치에 따라 모션 중의 감속 또는 가속이 결정된다.

- **인수**

인수명	자료형	기본값	설명
fTargetPos	Float[MAX_SPLINE_POINT][6]	-	최대 100개까지의 경유점 리스트
nPosCount	unsigned char	-	유효 경유점 개수
fTargetVel	float	-	속도
fTargetAcc	float	-	가속도
fTargetTime	float	0.f	도달 시간 [sec]
eMoveMode	enum.MOVE_MODE	MOVE_MODE_ABSOLUTE	상수 및 열거형 정의 참조

알아두기

- fTargetTime 을 지정할 경우 fTargetVel, fTargetAcc 를 무시하고 fTargetTime 기준으로 처리됩니다.
- eMoveMode 가 MOVE_MODE_RELATIVE 인 경우 position list 의 각 pos 는 앞 선 pos 에 대한 상대좌표로 정의됩니다. (position list=[q1, q2, ...,q(n-1), q(n)]로 이루어질 때 q1 은 시작점 대비 상대각도, q(n)은 q(n-1) 대비 상대좌표)
- 선행모션과 후행모션에 대한 온라인 블렌딩은 지원하지 않습니다.

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
// CASE 1 : 절대각도 입력 (mod=MOVE_MODE_ABSOLUTE)
float jpos[4][6];
float jvel=10;
float jacc=10;
int jposNum = 4;
jpos[0][0]=0; jpos[0][1]=0; jpos[0][2]=-30; jpos[0][3]=0; jpos[0][4]=-30;
jpos[0][5]=0;
jpos[1][0]=90; jpos[1][1]=0; jpos[1][2]=0; jpos[1][3]=0; jpos[1][4]=0;
jpos[1][5]=0;
jpos[2][0]=0; jpos[2][1]=0; jpos[2][2]=-30; jpos[2][3]=0; jpos[2][4]=-30;
jpos[2][5]=0;
jpos[3][0]=-90; jpos[3][1]=0; jpos[3][2]=0; jpos[3][3]=0; jpos[3][4]=0;
jpos[3][5] = 0;
drfl.movesj(jpos, jposNum, jvel, jacc);
// jpos에 정의된 절대경유점 집합을 연결하는 스플라인 곡선을 최대속도
// 10(deg/sec), 최대가속도 10(deg/sec2)로 움직임

// CASE 2 : 상대각도 입력 (mod=MOVE_MODE_RELATIVE)
float jpos[4][6];
float jvel=10;
float jacc=10;
int jposNum = 4;
jpos[0][0]=0; jpos[0][1]=0; jpos[0][2]=-30; jpos[0][3]=0; jpos[0][4]=-30;
jpos[0][5]=0;
// 시작위치 + jpos[0]
jpos[1][0]=90;        jpos[1][1]=0; jpos[1][2]=30; jpos[1][3]=0;
jpos[1][4]=30; jpos[1][5]=0;
// 시작위치 + jpos[0] + jpos[1]
jpos[2][0]=-90; jpos[2][1]=0; jpos[2][2]=-30; jpos[2][3]=0; jpos[2][4]=-30;
jpos[2][5]=0;
// 시작위치 + jpos[0] + jpos[1] + jpos[2]
jpos [3][0]=-90; jpos [3][1]=0; jpos[3][2]=30; jpos[3][3]=0; jpos[3][4]=30;
jpos[3][5]=0;
// 시작위치 + jpos[0] + jpos[1] + jpos[2] + jpos[3]
drfl.movesj(jpos, jposNum, jvel, jacc, time, MOVE_MODE_RELATIVE);
// jpos에 정의된 상대경유점 집합을 연결하는 스플라인 곡선을 최대속도
// 10(deg/sec), 최대가속도 10(deg/sec2)로 움직임
```

3.6.6 CDRFLEx.movesx

- **기능**

로봇 제어기에서 로봇을 현재 위치에서 작업공간의 경유점들을 거쳐 목표위치(마지막 경유점)까지 연결되는 스플라인 곡선경로를 따라 이동시키기 위한 함수이다. 입력된 속도/가속도는 경로 중 최대 속도/가속도이며 정속모션 옵션을 선택할 경우 조건에 따라 입력한 속도로 정속도의 모션을 수행합니다.

- **인수**

인수명	자료형	기본값	설명
fTargetPos	float [MAX_SPLINE_POINT][6]	-	최대 100개까지의 경유점 정보
nPosCount	unsigned char	-	유효 경유점 개수
fTargetVel	float[2]	-	선속도, 각속도
fTargetAcc	float[2]	-	선가속도, 각가속도
fTargetTime	float	0.f	도달 시간 [sec]
eMoveMode	enum.MOVE_MODE	MOVE_MODE_ABSOLUTE	상수 및 열거형 정의 참조
eMove Reference	enum.MOVE_REFERENCE	MOVE_REFERENCE_BASE	상수 및 열거형 정의 참조
eVelOpt	enum.SPLINE_VELOCITY_OPTION	SPLINE_VELOCITY_OPTION_DEFAULT	상수 및 열거형 정의 참조

알아두기

- fTargetVel 에 하나의 인자를 입력한 경우(예를들어, fTargetVel ={30, 0}) 입력된 인자는 모션의 선속도에 대응되며, 각속도는 선속도에 비례하여 결정됩니다.
- fTargetAcc 에 하나의 인자를 입력한 경우(예를들어, fTargetAcc ={60, 0}) 입력된 인자는 모션의 선가속도에 대응되며, 각가속도는 선가속도에 비례하여 결정됩니다.
- fTargetTime 지정 시, fTargetVel, fTargetAcc 를 무시하고 fTargetTime 기준으로 처리됩니다
- eMoveMode 가 MOVE_MODE_RELATIVE 인 경우 position list 의 각 pos 는 앞 선 pos 에 대한 상대좌표로 정의됩니다. (position list=[q1, q2, ...,q(n-1), q(n)]로 이루어질 때 q1 은 시작점 대비 상대각도, q(n)은 q(n-1) 대비 상대좌표)
- 선행모션과 후행모션에 대한 온라인 블렌딩은 지원하지 않습니다.

⚠ **주의**

eVelOpt 을 SPLINE_VELOCITY_OPTION_CONST 옵션(등속모션)을 선택할 경우 입력된 경유점 간 거리와 속도 조건에 따라 등속모션을 사용할 수 없을 수 있으며, 이 경우에 변속모션 (eVelOpt =SPLINE_VELOCITY_OPTION_DEFAULT)으로 자동 전환됩니다.

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
// CASE 1 : 절대좌표계 기준 이동 (mod= MOVE_MODE_ABSOLUTE)
float xpos[4][6];
int xposNum = 4;
float tvel={ 50, 100 };
float tacc={ 50, 100 };
xpos[0][0]=559; xpos[0][1]=434.5; xpos[0][2]=651.5;
xpos[0][3]=0; xpos[0][4]=180; xpos[0][5]=0;
xpos[1][0]=559; xpos[1][1]=434.5; xpos[1][2]=251.5;
xpos[1][3]=0; xpos[1][4]=180; xpos[1][5]=0;
xpos[2][0]=559; xpos[2][1]=234.5; xpos[2][2]=251.5;
xpos[2][3]=0; xpos[2][4]=180; xpos[2][5]=0;
xpos[3][0]=559; xpos[3][1]= 234.5; xpos[3][2]=451.5;
xpos[3][3]=0; xpos[3][4]=180; xpos[3][5]=0;
drfl.movesx(xpos, xposNum, tvel, tacc, 0, MOVE_MODE_ABSOLUTE);
    // 현재위치에서 시작하여 xpos에 정의된 경유점 집합을 연결하는 스플라인
    // 곡선을 최대속도 50, 50(mm/sec, deg/sec), 최대가속도 100, 100(mm/sec2,
    // deg/sec2)로 움직임

// CASE 2 : 상대좌표계 기준 이동 (mod= MOVE_MODE_RELATIVE)
float xpos[4][6];
int xposNum = 4;
float tvel={ 50, 100 };
float tacc={ 50, 100 };
xpos[0][0]=0; xpos[0][1]=400; xpos[0][2]=0;
xpos[0][3]=0; xpos[0][4]=0; xpos[0][5]=0;
// 시작위치에서 상대좌표 xpos[0]만큼의 동차변환 → x0
xpos[1][0]=0; xpos[1][1]=0; xpos[1][2]=-400;
xpos[1][3]=0; xpos[1][4]=0; xpos[1][5]=0;
```

```
// x0에서 상대좌표 xpos[1]만큼의 동차변환 → x1
xpos[2][0]=0; xpos[2][1]=-200; xpos[2][2]=0;
xpos[2][3]=0; xpos[2][4]=0; xpos[2][5]=0;
// x1에서 상대좌표 xpos[2]만큼의 동차변환 → x2
xpos[3][0]=0; xpos[3][1]=0; xpos[3][2]=200;
xpos[3][3]=0; xpos[3][4]=0; xpos[3][5]=0;
// x3에서 상대좌표 xpos[3]만큼의 동차변환 → x3
drfl.movesx(xpos, xposNum, tvel, tacc, 0, MOVE_MODE_RELATIVE);
      // 현재위치에서 시작하여 xpos에 상대정의된 경유점 집합을 연결하는
// 스플라인 곡선을 최대속도 50, 50(mm/sec, deg/sec), 최대가속도 100,
      // 100(mm/sec2, deg/sec2)로 움직임
```

3.6.7 CDRFLEx.moveb

- **기능**

로봇 제어기에서 하나 이상의 라인 또는 원호 구성된 경로 정보을 받아 로봇을 경로 정보에 설정된 blending radius로 블렌딩하여 등속으로 이동시키기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
tTargetPos	struct MOVE_POSB [MAX_MOVEB_POINT]	-	최대 25개까지의 경로 정보
nPosCount	unsigned char	-	유효 경로 정보 개수
fTargetVel	float[2]	-	선속도, 각속도
fTargetAcc	float[2]	-	선가속도, 각가속도
fTargetTime	float	0.f	도달 시간 [sec]
eMoveMode	enum.MOVE_MODE	MOVE_MODE_ABSOLUTE	상수 및 열거형 상수 참조
eMoveReference	enum.MOVE_REFERENCE	MOVE_REFERENCE_BASE	상수 및 열거형 상수 참조

알아두기

- fTargetVel 에 하나의 인자를 입력한 경우(예를들어, fTargetVel ={30, 0}) 입력된 인자는 모션의 선속도에 대응되며, 각속도는 선속도에 비례하여 결정됩니다.
- fTargetAcc 에 하나의 인자를 입력한 경우(예를들어, fTargetAcc ={60, 0}) 입력된 인자는 모션의 선가속도에 대응되며, 각가속도는 선가속도에 비례하여 결정됩니다.
- fTargetTime 지정 시, fTargetVel, fTargetAcc 를 무시하고 fTargetTime 기준으로 처리됩니다
- eMoveMode 가 MOVE_MODE_RELATIVE 인 경우 posb list 의 각 pos 는 앞 선 pos 에 대한 상대좌표로 정의됩니다.

주의

- posb 에서 blending radius 가 0 인 경우, 사용자 입력 오류가 나타납니다.
- 연속된 Line-Line segment 가 같은 방향을 가질 경우 Line 의 중복입력으로 사용자 입력 오류가 나타납니다.

- 블렌딩 구간에서 조건에 따라 급격하게 방향전환이 발생하게 되는 경우 급가속을 방지하기 위해 사용자 입력오류가 나타납니다.
- 선행모션과 후행모션에 대한 온라인 블렌딩은 지원하지 않습니다.

▪ 리턴

값	설명
0	오류
1	성공

▪ 예제

```
MOVE_POSB xb[4];
memset(xb, 0x00, sizeof(xb));
int segNum = 4;
float tvel = { 50, 50 };
float tacc = { 100, 100 };
xb[0]._iBlendType = 0;        // line
xb[0]._fBlendRad  = 50;
xb[0]._fTargetPos[0][0] = 559;
xb[0]._fTargetPos[0][1] = 234.5;
xb[0]._fTargetPos[0][2] = 651.5;
xb[0]._fTargetPos[0][3] = 0;
xb[0]._fTargetPos[0][4] = 180;
xb[0]._fTargetPos[0][5] = 0;
xb[1]._iBlendType = 1;        // circle
xb[1]._fBlendRad  = 50;
xb[1]._fTargetPos[0][0] = 559;
xb[1]._fTargetPos[0][1] = 234.5;
xb[1]._fTargetPos[0][2] = 451.5;
xb[1]._fTargetPos[0][3] = 0;
xb[1]._fTargetPos[0][4] = 180;
xb[1]._fTargetPos[0][5] = 0;
xb[1]._fTargetPos[1][0] = 559;
xb[1]._fTargetPos[1][1] = 434.5;
xb[1]._fTargetPos[1][2] = 451.5;
xb[1]._fTargetPos[1][3] = 0;
xb[1]._fTargetPos[1][4] = 180;
xb[1]._fTargetPos[1][5] = 0;
xb[2]._iBlendType = 0;        // line
xb[2]._fBlendRad = 50;
xb[2]._fTargetPos[0][0] = 559;
xb[2]._fTargetPos[0][1] = 434.5;
xb[2]._fTargetPos[0][2] = 251.5;
xb[2]._fTargetPos[0][3] = 0;
xb[2]._fTargetPos[0][4] = 180;
xb[2]._fTargetPos[0][5] = 0;
xb[3]._iBlendType = 0;        // line
```

```
xb[3]._fBlendRad  = 50;
xb[3]._fTargetPos[0][0] = 559;
xb[3]._fTargetPos[0][1] = 234.5;
xb[3]._fTargetPos[0][2] = 251.5;
xb[3]._fTargetPos[0][3] = 0;
xb[3]._fTargetPos[0][4] = 180;
xb[3]._fTargetPos[0][5] = 0;
drfl.moveb(xb, segNum, tvel, tacc);
      // 현재위치에서 시작하여 LINE→CIRCLE→LINE→LINE으로 이루어진 궤적을
// 속도 50, 50(mm/sec, deg/sec), 가속도 100, 100(mm/sec2, deg/sec2)
// 유지하며 각 구간의 끝점에서 50mm 거리에 도달하면 다음 segment로
      // blending이 시작됨
```

3.6.8 CDRFLEx.move_spiral

- **기능**

로봇제어기에서 방사형 방향으로 반경이 증가하며 회전하는 Spiral motion과 축 방향으로 병행하면서 이동시키기 위한 함수이다. 현재 위치에서 eMoveReference 로 지정한 좌표계 상의 axis 방향에 수직인 평면에서의 나선궤적과 axis 방향으로의 직선궤적을 동시에 따라 이동한다.

- **인수**

인수명	자료형	기본값	범위	설명
eTaskAxis	enum.TASK_AXIS	-	-	상수 및 열거형 정의 참조
fRevolution	float	-	rev > 0	총 회전수 [revolution]
fMaximuRadius	float	-	rmax > 0	spiral 최종 반경 [mm]
fMaximumLength	float	-		axis 방향으로 이동하는 거리 [mm]
fTargetVel	float[2]	-		선속도, 각속도
fTargetAcc	float[2]	-		선가속도, 각가속도
fTargetTime	float	0.0	time ≥ 0	총 수행시간 <sec>
eMoveReference	enum.MOVE_REFERENCE	MOVE_REFERENCE_TOOL		상수 및 열거형 정의 참조

알아두기

- fRevolution 는 spiral 모션의 총 회전수를 의미합니다.
- fMaximuRadius 는 spiral 모션의 최대 반경을 의미합니다.
- fMaximumLength 는 모션 동안 axis 방향으로 병진하는 거리를 의미합니다. 단, 음수인 경우 –axis 방향 병진합니다.
- fTargetVel 은 spiral 모션의 이동 속도를 의미합니다.
- fTargetAcc 는 spiral 모션의 이동 가속도를 의미합니다.
- fTargetTime 을 지정할 경우 fTargetVel, fTargetAcc 를 무시하고 fTargetTime 기준으로 처리됩니다.
- eTaskAxis 는 Spiral 모션이 정의하는 평면에 수직인 축을 정의합니다.

- eMoveReference 는 spiral 모션이 정의하는 기준 좌표계를 의미합니다.
- 선행모션과 후행모션에 대한 온라인 블렌딩은 지원하지 않습니다.

⚠ 주의

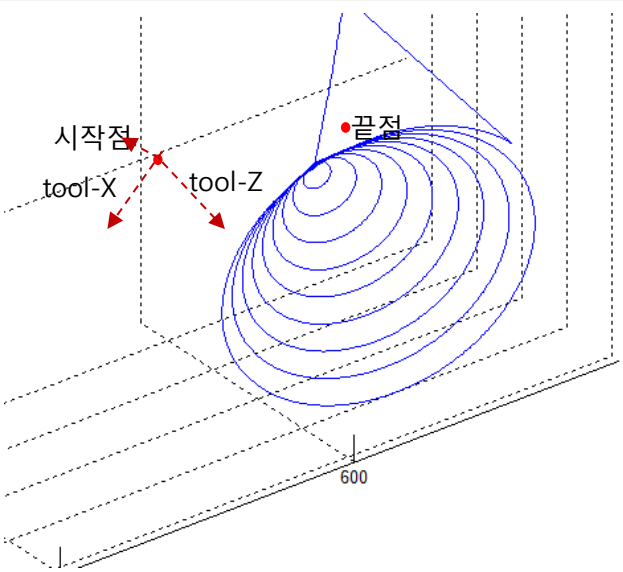
- 경로 생성 시 Spiral 경로에 의한 회전각 가속도를 연산하여 값이 큰 경우 안정적인 모션을 위하여 에러가 발생나타날 수 있습니다. 이 경우 fTargetVel, fTargetAcc 또는 fTargetTime 값을 작게 조정하는 것을 권장합니다.

▪ 리턴

값	설명
0	오류
1	성공

▪ 예제

```
float rev = 3;
float rmax = 50;
float lmax = 50;
float tvel = { 50, 50 };
float tacc = { 100, 100 };
Drfl.move_spiral(TASK_AXIS_Z, rev, rmax, lmax, tvel, tacc);
// 현재위치에서 시작하여 중심에서 최대 50mm 반경까지 나선형 궤적을
// 속도 50, 50(mm/sec, deg/sec), 가속도 100, 100(mm/sec2, deg/sec2)를
// 유지하며, 동시에 Tool z방향으로 50mm까지 이동
```



3.6.9 CDRFLEx.move_periodic

- **기능**

로봇제어기에서 현재 위치에서 시작하는 상대 모션으로 입력된 기준 좌표계(eMoveReference)의 각 축(병진 및 회전)에 대한 Sine 함수 기반으로 주기 모션을 수행합니다. 각 axis 별 모션의 특성은 fAmplitude와 fPeriodic 에 의해 결정되고, 가감속 시간과 총 모션 시간은 주기, 반복, 횟수에 의해 설정됩니다.

- **인수**

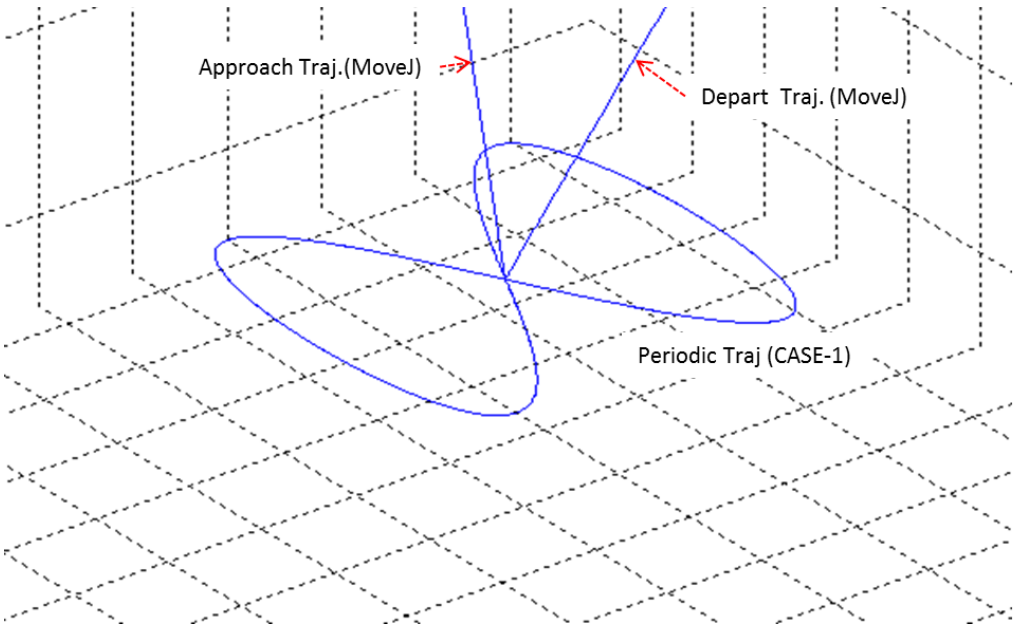
인수명	자료형	기본값	범위	설명
fAmplitude	float[6]	-	>=0	Amplitude(-amp에서 +amp사이 모션) [mm] or [deg]
fPeriodic	float[6]	-	>=0	period(1주기 소요 시간)[sec]
fAccelTime	float	-	>=0	Acc-, dec- time [sec]
nRepeat	unsigned char	-	> 0	반복 횟수
eMoveReference	enum	MOVE_REFERENCE_TOOL		상수 및 열거형 정의 참조

알아두기

- fAmplitude 는 진폭(amplitude)을 의미하며, 각 축(x, y, z, rx, ry, rz) 별로 amp 를 값으로 하는 6 개 원소의 list 형태로 입력해야 합니다. 단, 주기 모션을 진행하지 않는 축 방향은 amp 를 0 으로 입력해야 합니다.
- fPeriodic 는 해당 방향 모션의 1 회 반복 시간을 의미하며, 각 축(x, y, z, rx, ry, rz) 별 period 를 값으로 하는 총 6 개 원소의 list 형태로 입력하거나 대표값을 입력해야합니다.
- fAccelTime 은 주기모션의 시작과 끝의 가속 및 감속 시간을 의미합니다. 입력된 가감속시간과 최대주기*1/4 중 큰 값이 적용됩니다. 입력된 가감속 시간이 전체모션시간의 1/2 을 초과하는 경우 에러가 발생합니다.
- nRepeat 은 가장 큰 period 값을 가지는 축(기준 축)의 반복 횟수를 정의하며, 이에 따라 총 모션 시간이 결정됩니다. 나머지 축의 반복 횟수는 모션 시간에 따라 자동결정됩니다.모션이 정상 종료되는 경우 종료 위치가 시작 위치와 일치하게 하도록 나머지 축 모션은 기준 축 모션이 종료되기 전에 먼저 종료될 수 있습니다. 모든 축의 모션이 동시 종료되지 않는 경우 감속구간에서의 경로는 이전 경로에서 벗어나게 됩니다. 관련한 사항은 아래 이미지를 참조하십시오

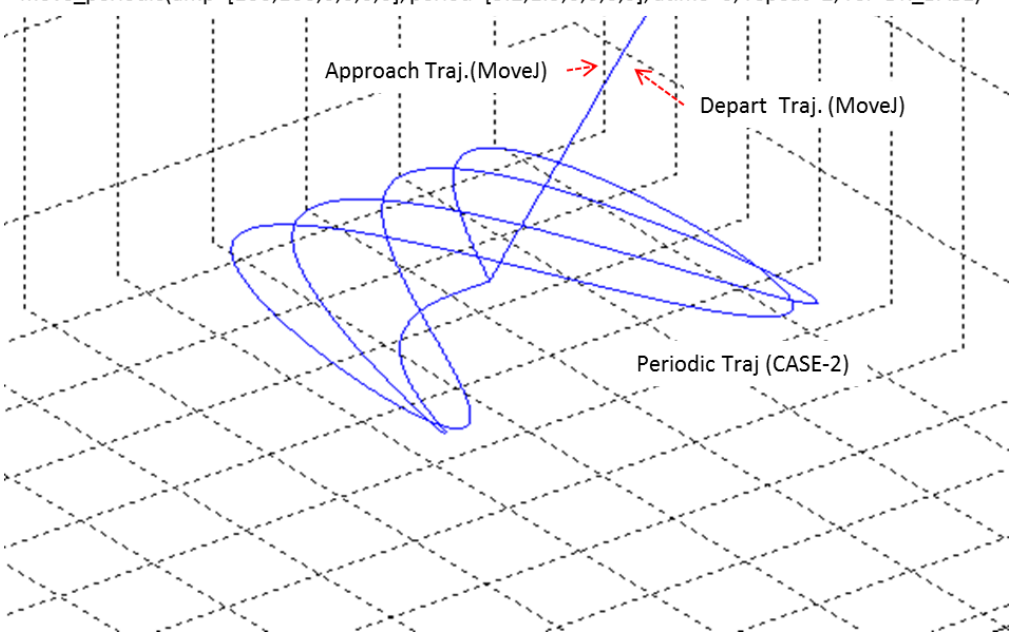
CASE-1) All-axis motions end at the same time

move_periodic(amp=[100,100,0,0,0,0], period=[3.2,1.6,0,0,0,0], atime=3.1, repeat=2, ref=DR_BASE)



CASE-2) Diff-axis motions end individually

move_periodic(amp=[100,100,0,0,0,0], period=[3.2,1.5,0,0,0,0], atime=0, repeat=2, ref=DR_BASE)



- eMoveReference 는 반복 모션의 기준 좌표계를 의미합니다.
- 모션명령 수행 시 최대속도 에러가 발생하는 경우 다음의 식을 참조하여 진폭 및 주기를 조정할 것을 제안합니다.
  최대속도=진폭(amp)*2*pi(3.14)/주기(period)
  (예, 진폭=10mm, 주기=1 초인 경우 최대속도=62.83mm/sec)
- 선행모션과 후행모션에 대한 온라인 블렌딩은 지원하지 않습니다.

▪ 리턴

값	설명
0	오류
1	성공

▪ 예제

```
<#1>
float fAmplitude[NUM_TASK] = {10,0,0,0,30,0};
float fPeriod[NUM_TASK] = { 1, 1, 1, 1, 1, 1};
drfl.move_periodic(fAmplitude, fPeriod, 0.2, 5, MOVE_REFERENCE_TOOL);
      # Tool 좌표계 x축(10mm 진폭, 1초 주기) 모션과 y회전축(진폭 30deg, 1초 주기)
      # 모션이 총 5회 반복 수행
<#2>
float fAmplitude[NUM_TASK] = {10,0,20,0,0.5,0};
float fPeriod[NUM_TASK] = { 1,0,1.5,0,0,0};
drfl.move_periodic (fAmplitude, fPeriod, 0.5, 3, MOVE_REFERENCE_BASE);
      # BASE 좌표계 x축(10mm 진폭, 1초 주기), z축(20mm 진폭, 1.5초 주기) 모션이
      # 총 3회 반복 수행됨, y회전축 모션은 period가 'zero(0)'이므로 미수행
      # z축 모션의 주기가 크므로 총 모션 시간은 약 5.5초(1.5초*3회 + 가감속 1초)
      # 이며, x축은 4.5회 반복 수행
```

3.6.10 CDRFLEx.amovej

- **기능**

비동기 방식의 movej로 블렌딩을 위한 fBlendingRadius 인자를 갖지 않는 점을 제외하고는 movej 함수와 동일하게 동작한다. 그러나 해당 명령어는 비동기 방식의 특성상 모션 종료를 기다리지 않고 모션 시작과 동시에 리턴하여 다음 라인을 실행한다.

- **인수**

인수명	자료형	기본값	설명
fTargetPos	float[6]	-	6개 축에 대한 목표 관절 위치
fTargetVel	float	-	속도
fTargetAcc	float	-	가속도
fTargetTime	float	0.f	도달 시간 [sec]
eMoveMode	enum.MOVE_MODE	MOVE_MODE_ABSOLUTE	상수 및 열거형 정의 참조
eBlendingType	enum.BLENDING_SPEED_TYPE	BLENDING_SPEED_TYPE_DUPLICATE	상수 및 열거형 정의 참조

알아두기

- fTargetTime 을 지정할 경우 fTargetVel, fTargetAcc 를 무시하고 fTargetTime 기준으로 처리됩니다.
- 옵션 eBlendingType 및 fTargetVel / fTargetAcc 에 따른 blending 시의 경로는 movej() 모션 설명을 참조할 것

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float q0[6] = { 0, 0, 90, 0, 90, 0 };
float q1[6] = { 90, 0, 90, 0, 90, 0 };
float q99[6] = { 0, 0, 0, 0, 0, 0 };
float jvel=10;
float jacc=20;
drfl.amovej(q0, jvel, jacc);
Sleep(3000);

Drfl.amovej (q1, jvel, jacc);
drfl.mwait(); // 모션이 종료할 때까지 프로그램 일시 중지

Drfl.movej(q99, jvel, jacc);
```

3.6.11 CDRFLEx.amovel

- **기능**

비동기 방식의 movel모션으로 블렌딩을 위한 fBlendingRadius 인자를 갖지 않는 점을 제외하고 movel와 동일하게 작동한다. 그러나 해당 명령어는 비동기 방식의 특성상 모션 종료를 기다리지 않고 모션 시작과 동시에 리턴하여 다음 라인을 실행한다.

- **인수**

인수명	자료형	기본값	설명
fTargetPos	float[6]	-	6개 축에 대한 목적 TCP 위치
fTargetVel	float[2]	-	선속도, 각속도
fTargetAcc	float[2]	-	선가속도, 각가속도
fTargetTime	float	-	도달 시간 [sec] * time 지정 시, vel, acc를 무시하고 time 기준으로 처리
eMoveMode	enum.MOVE_MODE	MOVE_MODE_ABSOLUTE	상수 및 열거형 정의 참조
eMoveReference	enum.MOVE_REFERENCE	MOVE_REFERENCE_BASE	상수 및 열거형 정의 참조
eBlendingType	enum.BLENDING_SPEED_TYPE	BLENDING_SPEED_TYPE_DUPLICATE	상수 및 열거형 정의 참조

알아두기

- fTargetVel 에 하나의 인자를 입력한 경우(예를들어, fTargetVel ={30, 0}) 입력된 인자는 모션의 선속도에 대응되며, 각속도는 선속도에 비례하여 결정됩니다.
- fTargetAcc 에 하나의 인자를 입력한 경우(예를들어, fTargetAcc ={60, 0}) 입력된 인자는 모션의 선가속도에 대응되며, 각가속도는 선가속도에 비례하여 결정됩니다.
- fTargetTime 지정 시, fTargetVel, fTargetAcc 를 무시하고 fTargetTime 기준으로 처리됩니다
- 옵션 eBlendingType 및 fTargetVel / fTargetAcc 에 따른 blending 시의 경로는 MoveL() 모션 설명을 참조할 것

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
// x1으로 모션시작 후 2초 후에 D-Out
float x1[6] = { 559, 434.5, 651.5, 0, 180, 0 };
float tvel = { 50, 50 };
float tacc = { 100, 100 };
drfl.amovel(x1, tvel, tacc);
Sleep(2000);
drfl.set_digital_output(GPIO_CTRLBOX_DIGITAL_INDEX_1, 1);
```

3.6.12 CDRFLEx.amovejx

- **기능**

비동기 방식의 movejx모션으로 블렌딩을 위한 fBlendingRadius 인자를 갖지 않는 점을 제외하고 movejx와 동일하게 작동한다. 그러나 해당 명령어는 비동기 방식의 특성상 모션 종료를 기다리지 않고 모션 시작과 동시에 리턴하여 다음 라인을 실행한다.

- **인수**

인수명	자료형	기본값	설명
fTargetPos	float[6]	-	6개 축에 대한 목적 TCP 위치
iSolutionSpace	unsigned char	-	관절조합형태 (아래 설명 참조)
fTargetVel	float	-	속도
fTargetAcc	float	-	가속도
fTargetTime	float	0.f	도달 시간 [sec]
eMoveMode	enum.MOVE_MODE	MOVE_MODE_ABSOLUTE	상수 및 열거형 정의 참조
eMoveReference	enum.MOVE_REFERENCE	MOVE_REFERENCE_BASE	상수 및 열거형 정의 참조
eBlendingType	enum.BLENDING_SPEED_TYPE	BLENDING_SPEED_TYPE_DUPLICATE	상수 및 열거형 정의 참조

알아두기

- fTargetTime 을 지정할 경우 fTargetVel, fTargetAcc 를 무시하고 fTargetTime 기준으로 처리됩니다.
- 옵션 eBlendingType 와 fTargetVel / fTargetAcc 에 따른 블렌딩 상태의 경로는 movej() 모션 설명을 참조하십시오.

주의

상대모션으로 입력하는 경우(eMoveMode = MOVE_MODE_RELATIVE), 진행중인 모션에 블렌딩 할 수 없으며 amovej() 또는 amovel()을 이용하여 블렌딩하는 것을 권장합니다.

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
// x1으로 조인트모션 시작 후 2초 후에 D-Out
float x1[6] = { 559, 34.5, 651.5, 0, 180, 0 };
float sol=2;
float jvel=10;
float jacc=20;
drfl.amovejx(x1, sol, jvel, jacc);
Sleep(2000);
drfl.set_digital_output(GPIO_CTRLBOX_DIGITAL_INDEX_1, 1);
```

3.6.13 CDRFLEx.amovec

- **기능**

비동기 방식의 movec모션으로 블렌딩을 위한 fBlendingRadius 인자를 갖지 않는 점을 제외하고 movec와 동일하게 작동한다. 그러나 해당 명령어는 비동기 방식의 특성상 모션 종료를 기다리지 않고 모션 시작과 동시에 리턴하여 다음 라인을 실행한다.

- **인수**

인수명	자료형	기본값	설명
fTargetPos[0]	float[6]	-	경유 지점
fTargetPos[1]	float[6]		목표 위치
fTargetVel	float[2]	-	선속도, 각속도
fTargetAcc	float[2]	-	선가속도, 각가속도
fTargetTime	float	0.f	도달 시간 [sec]
eMoveMode	enum.MOVE_MODE	MOVE_MODE_ABSOLUTE	상수 및 열거형 정의 참조
eMoveReference	enum.MOVE_REFERENCE	MOVE_REFERENCE_BASE	상수 및 열거형 정의 참조
fTargetAngle2	float	0.f	angle1
fTargetAngle2	float	0.f	angle2
eBlendingType	enum.BLENDING_SPEED_TYPE	BLENDING_SPEED_TYPE_DUPLICATE	상수 및 열거형 정의 참조

알아두기

- fTargetVel 에 하나의 인자를 입력한 경우(예를들어, fTargetVel ={30, 0}) 입력된 인자는 모션의 선속도에 대응되며, 각속도는 선속도에 비례하여 결정됩니다.
- fTargetAcc 에 하나의 인자를 입력한 경우(예를들어, fTargetAcc ={60, 0}) 입력된 인자는 모션의 선가속도에 대응되며, 각가속도는 선가속도에 비례하여 결정됩니다.
- fTargetTime 지정 시, fTargetVel, fTargetAcc 를 무시하고 fTargetTime 기준으로 처리됩니다.
- eMoveMode 가 MOVE_MODE_RELATIVE 인 경우 fTargetPos[0] 과 fTargetPos[1] 는 각각 앞 선 위치값에 대한 상대좌표로 정의됩니다. (fTargetPos[0]은 시작점 대비 상대좌표, fTargetPos[1]는 fTargetPos[0]대비 상대좌표)
- fTargetAngle1 이 0 보다 크고, fTargetAngle2 이 0 인 경우 fTargetAngle1 은 Circular path 상의 총 회전각이 적용됩니다.

- fTargetAngle1 과 fTargetAngle2 가 0 보다 큰 경우, fTargetAngle1 은 circular path 상에서 정속으로 이동하는 총 회전각을, fTargetAngle2 는 가속과 감속을 위한 회전 구간의 회전각을 의미합니다. 이때 총 이동각은 fTargetAngle1+ 2 X fTargetAngle2 만큼 circular path 상을 움직입니다.
- 옵션 eBlendingType 와 fTargetVel / fTargetAcc 에 따른 블렌딩 상태의 경로는 movej() 모션 설명을 참조하십시오

▪ 리턴

값	설명
0	오류
1	성공

▪ 예제

```
// x1의 두 점을 경유하는 원호모션 시작 후 3초 후에 D-Out
float x1[2][6] = { { 559, 434.5, 651.5, 0, 180, 0 }, { 559, 434.5, 251.5, 0, 180, 0 } };
float tvel = { 50, 50 }; // 태스크 속도를 50(mm/sec), 50(deg/sec)로 설정
float tacc = { 100, 100 }; // 태스크 가속도를 100(mm/sec2), 100(deg/sec2)로 설정
drfl.amovec(x1, tvel, tacc);
Sleep(3000);
drfl.set_digital_output(GPIO_CTRLBOX_DIGITAL_INDEX_1, 1);
```

3.6.14 CDRFLEx.amovesj

- **기능**

비동기 방식의 movesj모션으로 비동기 처리 방식외에는 movesj()와 동일하게 동작하며, 모션 종료를 기다리지 않고 모션 시작과 동시에 리턴하여 다음 라인을 실행한다. amovesj()에 의한 모션이 종료되기 전에 발생하는 새로운 모션 명령어는 안전상의 이유로 오류를 발생시킨다. 따라서 amovesj()와 이어지는 새로운 모션명령어 사이에는 mwait() 함수을 사용하여 amovesj()에 의한 모션이 종료된 것을 확인한 후 새로운 모션명령어가 시작되도록 해야한다.

- **인수**

인수명	자료형	기본값	설명
fTargetPos	Float[MAX_SPLINE_POINT] [6]	-	최대 100개까지의 경유점 리스트
nPosCount	unsigned char	-	유효 경유점 개수
fTargetVel	float	-	속도
fTargetAcc	float	-	가속도
fTargetTime	float	0.0	도달 시간 [sec]
eMoveMode	enum.MOVE_MODE	MOVE_MODE_ABSOLUTE	상수 및 열거형 정의 참조

알아두기

- fTargetTime 을 지정할 경우 fTargetVel, fTargetAcc 를 무시하고 fTargetTime 기준으로 처리됩니다.
- eMoveMode 가 MOVE_MODE_RELATIVE 인 경우 position list 의 각 pos 는 앞 선 pos 에 대한 상대좌표로 정의됩니다. (position list=[q1, q2, ...,q(n-1), q(n)]로 이루어질 때 q1 은 시작점 대비 상대각도, q(n)은 q(n-1) 대비 상대좌표)
- 선행모션과 후행모션에 대한 온라인 블랜딩을 지원하지 않습니다.

- **리턴**

값	설명
0	오류

값	설명
1	성공

- **예제**

```
// jpos의 모든 점을 경유하는 스플라인모션 시작 후 3초 후에 D-Out
float jpos[4][6];
float jvel=10;
float jacc=10;
int jposNum = 4;
jpos[0][0]=0; jpos[0][1]=0; jpos[0][2]=-30; jpos[0][3]=0; jpos[0][4]=-30;
jpos[0][5]=0;
jpos[1][0]=90; jpos[1][1]=0; jpos[1][2]=0; jpos[1][3]=0; jpos[1][4]=0;
jpos[1][5]=0;
jpos[2][0]=0; jpos[2][1]=0; jpos[2][2]=-30; jpos[2][3]=0; jpos[2][4]=-30;
jpos[2][5]=0;
jpos[3][0]=-90; jpos[3][1]=0; jpos[3][2]=0; jpos[3][3]=0; jpos[3][4]=0;
jpos[3][5] = 0;

drfl.movesj(jpos, jposNum, jvel, jacc);
Sleep(3000);
drfl.set_digital_output(GPIO_CTRLBOX_DIGITAL_INDEX_1, 1);
```

3.6.15 CDRFLEx.amovesx

- **기능**

비동기 방식의 movesx모션으로 비동기 처리 이외에는 movesx()와 동일하게 동작하며 모션 종료를 기다리지 않고 모션 시작과 동시에 리턴하여 다음 라인을 실행한다, amovesx()에 의한 모션이 종료되기 전에 발생하는 새로운 모션 명령어는 안전상의 이유로 오류를 발생시킨다. 따라서 amovesx 함수()와 이어지는 새로운 모션명령어 사이에는 mwait() 함수을 사용하여 amovesx()에 의한 모션이 종료된 것을 확인한 후 새로운 모션 명령어가 시작되도록 해야한다.

- **인수**

인수명	자료형	기본값	설명
fTargetPos	float [MAX_SPLINE_ POINT][6]	-	최대 100개까지의 경유점 정보
nPosCount	unsigned char	-	유효 경유점 개수
fTargetVel	float[2]	-	선속도, 각속도
fTargetAcc	float[2]	-	선가속도, 각가속도
fTargetTime	float	0.0	도달 시간 [sec]
eMoveMode	enum.MOVE_M ODE	MOVE_MODE_ ABSOLUTE	상수 및 열거형 정의 참조
eMove Reference	enum.MOVE_RE FERENCE	MOVE_REFERENCE_B ASE	상수 및 열거형 정의 참조
eVelOpt	enum.SPLINE_V ELOCITY_OPTIO N	SPLINE_VELOCITY_OP TION_DEFAULT	상수 및 열거형 정의 참조

알아두기

- fTargetVel 에 하나의 인자를 입력한 경우(예를들어, fTargetVel ={30, 0}) 입력된 인자는 모션의 선속도에 대응되며, 각속도는 선속도에 비례하여 결정됩니다.
- fTargetAcc 에 하나의 인자를 입력한 경우(예를들어, fTargetAcc ={60, 0}) 입력된 인자는 모션의 선가속도에 대응되며, 각가속도는 선가속도에 비례하여 결정됩니다.
- fTargetTime 지정 시, fTargetVel, fTargetAcc 를 무시하고 fTargetTime 기준으로 처리됩니다

- eMoveMode 가 MOVE_MODE_RELATIVE 인 경우 position list 의 각 pos 는 앞 선 pos 에 대한 상대좌표로 정의됩니다. (position list=[q1, q2, ...,q(n-1), q(n)]로 이루어질 때 q1 은 시작점 대비 상대각도, q(n)은 q(n-1) 대비 상대좌표)
- 선행모션과 후행모션에 대한 온라인 블렌딩은 지원하지 않습니다.
-

⚠ 주의

eVelOpt 을 SPLINE_VELOCITY_OPTION_CONST 옵션(등속모션)을 선택할 경우 입력된 경유점 간 거리와 속도 조건에 따라 등속모션을 사용할 수 없을 수 있으며, 이 경우에 변속모션 (eVelOpt =SPLINE_VELOCITY_OPTION_DEFAULT)으로 자동 전환됩니다.

▪ 리턴

값	설명
0	오류
1	성공

▪ 예제

```
// xpos의 모든 점을 경유하는 스플라인모션 시작 후 3초 후에 D-Out
float xpos[4][6];
int xposNum = 4;
float tvel={ 50, 100 };
float tacc={ 50, 100 };
xpos[0][0]=559; xpos[0][1]=434.5; xpos[0][2]=651.5;
xpos[0][3]=0; xpos[0][4]=180; xpos[0][5]=0;
xpos[1][0]=559; xpos[1][1]=434.5; xpos[1][2]=251.5;
xpos[1][3]=0; xpos[1][4]=180; xpos[1][5]=0;
xpos[2][0]=559; xpos[2][1]=234.5; xpos[2][2]=251.5;
xpos[2][3]=0; xpos[2][4]=180; xpos[2][5]=0;
xpos[3][0]=559; xpos[3][1]= 234.5; xpos[3][2]=451.5;
xpos[3][3]=0; xpos[3][4]=180; xpos[3][5]=0;
drfl.amovesx(xpos, xposNum, tvel, tacc, 0, MOVE_MODE_ABSOLUTE);
drfl.set_digital_output(GPIO_CTRLBOX_DIGITAL_INDEX_1, 1);
```

3.6.16 CDRFLEx.amoveb

- **기능**

비동기 방식의 moveb모션으로 비동기 처리 외에는 moveb() 함수와 동일하게 동작하며 모션 종료를 기다리지 않고 모션 시작과 동시에 리턴하여 다음 라인을 실행한다. amoveb() 함수에 의한 모션이 종료되기 전에 발생하는 새로운 모션지령은 안전상의 이유로 오류를 발생시킨다. 따라서 amoveb() 함수와 이어지는 새로운 모션명령어 사이에는 mwait() 함수를 사용하여 amoveb()에 의한 모션이 종료된 것을 확인한 후 새로운 모션 명령어가 시작되도록 해야한다

- **인수**

인수명	자료형	기본값	설명
tTargetPos	struct MOVE_POSB [MAX_MOVEB_POINT]	-	최대 25개까지의 경로 정보
nPosCount	unsigned char	-	유효 경로 정보 개수
fTargetVel	float[2]	-	선속도, 각속도
fTargetAcc	float[2]	-	선가속도, 각가속도
fTargetTime	float	0.f	도달 시간 [sec]
eMoveMode	enum.MOVE_MODE	MOVE_MODE_ABSOLUTE	상수 및 열거형 정의 참조
eMoveReference	enum.MOVE_REFERENCE	MOVE_REFERENCE_BASE	상수 및 열거형 정의 참조

알아두기

- fTargetTime 을 지정할 경우 fTargetVel, fTargetAcc 를 무시하고 fTargetTime 기준으로 처리됩니다.
- eMoveMode 가 MOVE_MODE_RELATIVE 인 경우 posb list 의 각 pos 는 앞 선 pos 에 대한 상대좌표로 정의됩니다.
- 선행모션과 후행모션에 대한 온라인 블렌딩은 지원하지 않습니다.

주의

- tTargetPos 에서 blending radius 가 0 인 경우, 사용자 입력 오류가 나타납니다.

- 연속된 Line-Line segment 가 같은 방향을 가질 경우 Line 의 중복입력으로 사용자 입력 오류가 나타납니다.
- 블렌딩 구간에서 조건에 따라 급격하게 방향전환이 발생하게 되는 경우 급가속을 방지하기 위해 사용자 입력오류가 나타납니다.
- 선행모션과 후행모션에 대한 온라인 블렌딩은 지원하지 않습니다.

▪ 리턴

값	설명
0	오류
1	성공

▪ 예제

```
// xb의 모든 경로를 따르는 모션 시작 후 3초 후에 D-Out
MOVE_POSB xb[4];
memset(xb, 0x00, sizeof(xb));
int segNum = 4;
float tvel = { 50, 50 };
float tacc = { 100, 100 };
xb[0]._iBlendType = 0;        // line
xb[0]._fBlendRad  = 50;
xb[0]._fTargetPos[0][0] = 559;
xb[0]._fTargetPos[0][1] = 234.5;
xb[0]._fTargetPos[0][2] = 651.5;
xb[0]._fTargetPos[0][3] = 0;
xb[0]._fTargetPos[0][4] = 180;
xb[0]._fTargetPos[0][5] = 0;
xb[1]._iBlendType = 1;        // circle
xb[1]._fBlendRad  = 50;
xb[1]._fTargetPos[0][0] = 559;
xb[1]._fTargetPos[0][1] = 234.5;
xb[1]._fTargetPos[0][2] = 451.5;
xb[1]._fTargetPos[0][3] = 0;
xb[1]._fTargetPos[0][4] = 180;
xb[1]._fTargetPos[0][5] = 0;
xb[1]._fTargetPos[1][0] = 559;
xb[1]._fTargetPos[1][1] = 434.5;
xb[1]._fTargetPos[1][2] = 451.5;
xb[1]._fTargetPos[1][3] = 0;
xb[1]._fTargetPos[1][4] = 180;
xb[1]._fTargetPos[1][5] = 0;
xb[2]._iBlendType = 0;        // line
xb[2]._fBlendRad = 50;
xb[2]._fTargetPos[0][0] = 559;
xb[2]._fTargetPos[0][1] = 434.5;
xb[2]._fTargetPos[0][2] = 251.5;
```

```
xb[2]._fTargetPos[0][3] = 0;
xb[2]._fTargetPos[0][4] = 180;
xb[2]._fTargetPos[0][5] = 0;
xb[3]._iBlendType = 0;        // line
xb[3]._fBlendRad  = 50;
xb[3]._fTargetPos[0][0] = 559;
xb[3]._fTargetPos[0][1] = 234.5;
xb[3]._fTargetPos[0][2] = 251.5;
xb[3]._fTargetPos[0][3] = 0;
xb[3]._fTargetPos[0][4] = 180;
xb[3]._fTargetPos[0][5] = 0;
drfl.amoveb(xb, segNum, tvel, tacc);
Sleep(3000);
drfl.set_digital_output(GPIO_CTRLBOX_DIGITAL_INDEX_1, 1);
```

3.6.17 CDRFLEx.amove_spiral

- **기능**

비동기 방식의 move_spiral모션으로 비동기 처리 외에는 move_spiral() 함수와 동일하게 동작하며 모션 종료를 기다리지 않고 모션 시작과 동시에 리턴하여 다음 라인을 실행한다.

amove_spiral() 함수에 의한 모션이 종료되기 전에 발생하는 새로운 모션지령은 안전상의 이유로 오류를 발생시킨다. 따라서 amove_spiral() 함수과 이어지는 새로운 모션명령어 사이에는 mwait() 함수를 사용하여 amove_spiral()에 의한 모션이 종료된 것를 확인한 후 새로운 모션명령어가 시작되도록 해야한다.

본 명령은 방사형 방향으로 반경이 증가하며 회전하는 Spiral motion과 축방향으로 병행하는 모션을 수행합니다. 현재위치에서 기준 좌표계(eMoveReference) 지정한 좌표계 상의 축 방향에 수직인 평면에서의 나선궤적과 축 방향으로의 직선궤적을 동시에 따라 이동한다.

- **인수**

인수명	자료형	기본값	범위	설명
eTaskAxis	enum.TASK_AXIS	-	-	상수 및 열거형 정의 참조
fRevolution	float	-	rev > 0	총 회전수 [revolution]
fMaximuRadius	float	-	rmax > 0	spiral 최종 반경 [mm]
fMaximumLength	float	-		axis 방향으로 이동하는 거리 [mm]
fTargetVel	float[2]	-		선속도, 각속도
fTargetAcc	float[2]	-		선가속도, 각가속도
fTargetTime	float	0.0	time ≥ 0	총 수행시간 <sec>
eMoveReference	enum.MOVE_REFERENCE	MOVE_REFERENCE_TOOL		상수 및 열거형 정의 참조

- **알아두기**

  - fRevolution 는 spiral 모션의 총 회전수를 의미합니다.
  - fMaximuRadius 는 spiral 모션의 최대 반경을 의미합니다.

- fMaximumLength 는 모션 동안 axis 방향으로 병진하는 거리를 의미합니다. 단, 음수인 경우 –axis 방향 병진합니다.
- fTargetVel 은 spiral 모션의 이동 속도를 의미합니다.
- fTargetAcc 는 spiral 모션의 이동 가속도를 의미합니다.
- fTargetTime 을 지정할 경우 fTargetVel, fTargetAcc 를 무시하고 fTargetTime 기준으로 처리됩니다.
- eTaskAxis 는 Spiral 모션이 정의하는 평면에 수직인 축을 정의합니다.
- eMoveReference 는 spiral 모션이 정의하는 기준 좌표계를 의미합니다.
- 선행모션과 후행모션에 대한 온라인 블렌딩은 지원하지 않습니다.

⚠ 주의

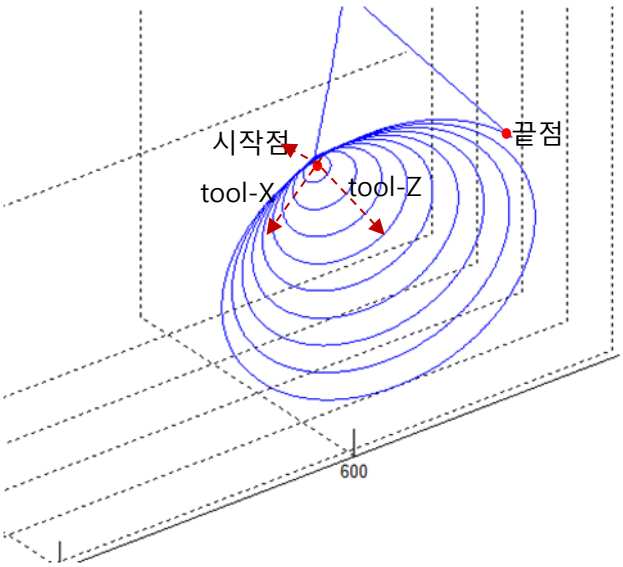
- 경로 생성 시 Spiral 경로에 의한 회전각 가속도를 연산하여 값이 큰 경우 안정적인 모션을 위하여 에러가 발생나타날 수 있습니다.
  이 경우 fTargetVel, fTargetAcc 또는 fTargetTime 값을 작게 조정하는 것을 권장합니다.

▪ 리턴

값	설명
0	오류
1	성공

▪ 예제

```
// 나선형 모션 시작 후 3초 후에 D-Out
float rev = 3;
float rmax = 50;
float lmax = 50;
float tvel = { 50, 50 };
float tacc = { 100, 100 };
Drfl.amove_spiral(TASK_AXIS_Z, rev, rmax, lmax, tvel, tacc);
Sleep(3000);
drfl.set_digital_output(GPIO_CTRLBOX_DIGITAL_INDEX_1, 1);
```


시작점
끝점
tool-X
tool-Z
600

3.6.18 CDRFLEx.amove_periodic

- **기능**

비동기 방식의 move_periodic모션으로 비동기 처리 외에 move_periodic() 함수와 동일하게 동작하며 모션 종료를 기다리지 않고 모션 시작과 동시에 리턴하여 다음 라인을 실행한다. move_periodic() 함수에 의한 모션이 종료되기 전에 발생하는 새로운 모션 명령어는 안전상의 이유로 오류를 발생시킨다. 따라서 amove_periodic() 함수와 이어지는 새로운 모션명령어 사이에는 mwait() 함수를 사용하여 amove_periodic()에 의한 모션이 종료된 것를 확인한 후 새로운 모션명령어가 시작되도록 해야한다.

이 명령어는 현재위치에서 시작하는 상대 모션어로 입력된 기준 좌표계(eMoveReferecne)의 각 축(병진 및 회전)에 대한 Sine함수 기반으로 주기모션을 수행합니다. 각 축 별 모션의 특성은 진폭과 주기에 의해 결정되고 가감속 시간과 총 모션 시간은 주기, 반복 및 횟수에 의해 설정된다.

- **인수**

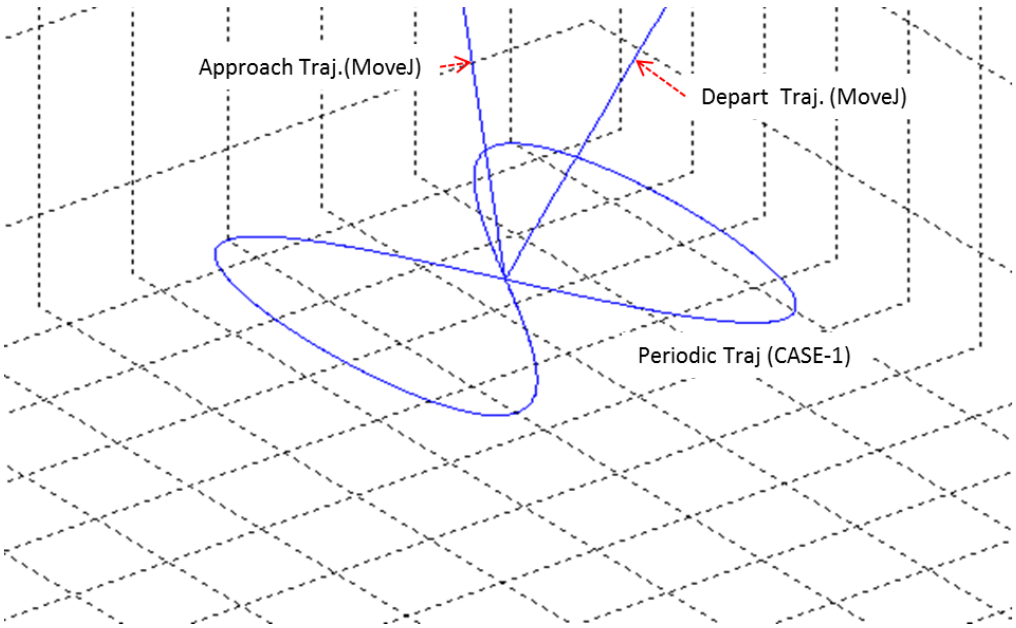
인수명	자료형	기본값	범위	설명
fAmplitude	float[6]	-	>=0	Amplitude(-fAmplitude 에서 + fAmplitude 사이 모션) [mm] or [deg]
fPeriodic	float[6]	-	>=0	period(1주기 소요 시간)[sec]
fAccelTime	float	-	>=0	Acc-, dec- time [sec]
nRepeat	unsigned char	-	> 0	반복 횟수
eMoveReference	enum.MOVE_REFERENCE	MOVE_REFERENCE_TOOL	-	상수 및 열거형 정의 참조

알아두기

- fAmplitude 는 진폭(amplitude)을 의미하며, 각 축(x, y, z, rx, ry, rz) 별로 6 개가 입력되어야 합니다. 단, 주기 모션을 진행하지 않는 축 방향은 fAmplitude 를 0 으로 입력해야 합니다.
- fPeriodic 는 해당 방향 모션의 1 회 반복 시간을 의미하며, 각 축(x, y, z, rx, ry, rz) 별 6 개가 입력되어야 합니다.
- fAccelTime 은 주기모션의 시작과 끝의 가속 및 감속 시간을 의미합니다. 입력된 가감속시간과 최대주기*1/4 중 큰 값이 적용됩니다. 입력된 가감속 시간이 전체모션시간의 1/2 을 초과하는 경우 에러가 발생합니다.

- nRepeat 은 가장 큰 period 값을 가지는 축(기준 축)의 반복 횟수를 정의하며, 이에 따라 총 모션 시간이 결정됩니다. 나머지 축의 반복 횟수는 모션 시간에 따라 자동 결정됩니다.
- 모션이 정상 종료되는 경우 종료 위치가 시작 위치와 일치하게 하도록 나머지 축 모션은 기준 축 모션이 종료되기 전에 먼저 종료될 수 있습니다. 모든 축의 모션이 동시 종료되지 않는 경우 감속구간에서의 경로는 이전 경로에서 벗어나게 됩니다. 관련한 사항은 아래 이미지를 참조하십시오

CASE-1) All-axis motions end at the same time
move_periodic(amp=[100,100,0,0,0,0], period=[3.2,1.6,0,0,0,0], atime=3.1, repeat=2, ref=DR_BASE)



- eMoveReference 는 반복 모션의 기준 좌표계를 의미합니다.
- 모션명령 수행 시 최대속도 에러가 발생하는 경우 다음의 식을 참조하여 진폭 및 주기를 조정할 것을 제안합니다.
  최대속도=진폭(fAmplitude)*2*pi(3.14)/주기(fPeriodic**)**
  (예, 진폭=10mm, 주기=1 초인 경우 최대속도=62.83mm/sec)
- 선행모션과 후행모션에 대한 온라인 블렌딩은 지원하지 않습니다.

▪ 리턴

값	설명
0	오류
1	성공

- **예제**

```
// Tool좌표계 x축(10mm진폭, 1초 주기)모션과 y회전축(진폭 0.5deg, 1초 주기)
// 모션을 총 5회 반복 수행
// periodic 모션을 시작하고 1초 후에 Digital_Output 채널1번을 SET(1) 한다.
float fAmplitude[NUM_TASK] = {10, 0, 0, 0, 0.5, 0};
float fPeriod[NUM_TASK] = { 1, 1, 1, 1, 1, 1};
Drfl.amove_periodic(fAmplitude, fPeriod, 0.5, 5, MOVE_REFERENCE_TOOL);
Sleep(1000);
Drfl.set_digital_output(GPIO_CTRLBOX_DIGITAL_INDEX_1, 1);
Drfl.mwait();
```

3.6.19 CDRFLEx.stop

- **기능**

로봇제어기에서 수행 중인 모션을 정지시키기 위한 함수이다. 인자로 받는 eStopType에 따라 다르게 정지하며 Estop을 제외한 모든 Stop 모드는 현재 수행하고 있는 구간의 모션을 정지한다

- **인수**

인수명	자료형	기본값	설명
eStopType	enum.STOP_TYPE	STOP_TYPE_QUICK	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
# x1으로 이동 시작 2초 후에 Soft Stop으로 모션 종료
float x1[NUM_TASK] = {784, 543, 970, 0, 180, 0};
drfl.movel(x1, 100, 200)              // x1으로 모션 및 즉시 다음 명령 수행
Sleep(2000)                            // 2초간 프로그램 일시 중지
drfl.stop(STOP_TYPE_SLOW)     // Soft Stop하여 모션 정지
```

3.6.20 CDRFLEx.move_pause

- **기능**

로봇제어기에서 현재 진행중인 로봇의 모션을 감속시켜 일시중지시키기 위한 함수이다. 진행중인 로봇 모션이 없는 경우에는 무시된다.

- **인수**

없음.

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float j00[NUM_JOINT] = {0, 0, 90, 0, 90,};
drfl.amovej(j00, 20, 40);  // 비동기모션 시작
while (1) {
    // 현재 관절각도 확인
    LPPOSITION pPose = drfl.get_current_pose(ROBOT_POSE_JOINT);
    if (pPose->_fTargetPos[2] >= 45) { // 3축 각도가 45도 이상이면,
        drfl.move_pause();           // 모션 일시정지
        Sleep(5000);                //5초간 기다림
        break;                     // while문 종료
    }
}
drfl.move_resume();          // 모션 재개
```

3.6.21 CDRFLEx.move_resume

- **기능**

로봇제어기에서 move_pause 함수로 일시 중지된 로봇의 모션을 재개하기 위한 함수이다. 진행중인 로봇 경로모션이 없는 경우에는 무시된다.

- **인수**

없음

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float j00[NUM_JOINT] = {0, 0, 90, 0, 90,};
drfl.amovej(j00, 20, 40);  // 비동기모션 시작
while (1) {
// 현재 관절각도 확인
LPPOSITION pPose = drfl.get_current_pose(ROBOT_POSE_JOINT);
    if (pPose->_fTargetPos[2] >= 45) { // 3축 각도가 45도 이상이면,
        drfl.move_pause();           // 모션 일시정지
        Sleep(5000);                //5초간 기다림
        break;                     // while문 종료
    }
}
drfl.move_resume();          // 모션 재개
```

3.6.22 CDRFLEx.mwait

- **기능**

로봇제어기에서 선행된 모션명령어의 동작이 종료되기를 기다리기 위한 함수이다. 비동기 모션 명령어와 본 함수를 결합하여 사용하면 동기 모션 명령어와 동일한 동작을 수행할 수 있다.

- **인수**

없음

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float point[6] = { 30, 30, 30, 30, 30, 30 };
drfl.amovej(point, 60, 120);
drfl.mwait();
```

3.6.23 CDRFLEx.trans

- **기능**

eSourceRef 좌표계 기준으로 정의된 Sourcepos를 동일 좌표계를 기준으로 offset만큼 이동/회전하여 eTargetRef 좌표계 기준으로 변환한 후 반환한다.

- **인수**

인수명	자료형	기본값	설명
fSourcePos	float[6]	-	6개의 Joint Space 정보
fOffset	float[6]	-	6개의 오프셋 정보
eSourceRef	enum.COORDINATE_SYSTEM	COORDINATE_SYSTEM_BASE	상수 및 열거형 정의 참조
eTargetRef	enum.COORDINATE_SYSTEM	COORDINATE_SYSTEM_BASE	상수 및 열거형 정의 참조

- **리턴**

값	설명
ROBOT_POSE	상수 및 열거형 정의 참조

- **예제**

```
float point[6] = { 30, 30, 30, 30, 30, 30 };
float offset[6] = { 100, 100, 100, 100, 100, 100};
LPROBOT_POSE res = Drfl.trans(point, offset);
for(int i=0; i<6; i++){
    cout << res->_fPosition[i] << endl;
}
```

3.6.24 CDRFLEx.fkin

- **기능**

조인트 공간에서 조인트 각도(fSourcePos)를 입력받아 eTargetRef좌표계 기준의 TCP의 포즈(태스크 공간의 위치 및 자세)를 반환한다.

- **인수**

인수명	자료형	기본값	설명
fSourcePos	float[6]	-	6개의 Joint Space 정보
eTargetRef	enum.COORDINATE_SYSTEM	COORDINATE_SYSTEM_BASE	상수 및 열거형 정의 참조

- **리턴**

값	설명
ROBOT_POSE	상수 및 열거형 정의 참조

- **예제**

```
float q1[6] = {0,0,90,0,90,0};
Drfl.movej(q1, 60, 30);
float q2[6] = {30, 0, 90, 0, 90, 0};
LPROBOT_POSE res = Drfl.fkin(q2, COORDINATE_SYSTEM_WORLD);
float vel[2] = {100, 100};
float acc[2] = {200, 200};
float x2[6] = {0,};
for(int i=0; i<6; i++){
    x2[i] = res->_fPosition[i];
}
Drfl.MoveL(x2, vel, acc);
```

3.6.25 CDRFLEx.ikin

- **기능**

작업 공간 내 기준 좌표계(eTargetRef)의 로봇 포즈에 상응하는 8개의 관절 형상 중 지정한 관절 형상 iSolutionSpace에 해당하는 관절 각도를 반환한다.

- **인수**

인수명	자료형	기본값	설명
fSourcePos	float[6]	-	6개의 Task Space 정보
iSolutionSpace	uchar	-	solution space
eTargetRef	enum.COORDINATE_SYSTEM	COORDINATE_SYSTEM_BASE	상수 및 열거형 정의 참조

- **리턴**

값	설명
ROBOT_POSE	상수 및 열거형 정의 참조

- **예제**

```
float x1[6] = {370.9, 719.7, 651.5, 90, -180, 0};
LPROBOT_POSE res = Drfl.ikin(x1, 2);
float q1[6] = {0,};
for(int i=0; i<6; i++){
    q1[i] = res->_fPosition[i];
}
Drfl.movej(q1, 60, 30);
```

3.6.26 CDRFLEx.set_ref_coord

- **기능**

  기준 좌표계를 설정한다.

- **인수**

인수명	자료형	기본값	설명
eTargetCoordSystem	enum.COORDINATE_SYSTEM	COORDINATE_SYSTEM_BASE	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float p1[6] = {0, 0, 90, 0, 90, 0};
Drfl.movej(p1, 60, 30);
float vec[2][3] = {{-1, 1, 1}, {1, 1, 0}};
float org[3] = {370.9, -419.7, 651.5};

int user = Drfl.set_user_cart_coord(vec, org);
Drfl.set_ref_coord((COORDINATE_SYSTEM)user);
```

3.6.27 CDRFLEx.check_motion

- **기능**

현재 진행 중인 모션의 상태를 확인한다.

- **인수**

없음

- **리턴**

값	설명
0	수행 중인 모션이 없음
1	모션 연산 중
2	모션이 수행 중

- **예제**

```
float q0[6] = {0, 0, 90, 0, 90, 0};
float q99[6] = {0, 0, 0, 0, 0, 0};
Drfl.amovej(q0, 60, 30); //q0로 모션 및 즉시 다음명령 수행
while(true)
{
    if(Drfl.check_motion() == 0) // 모션이 완료된 경우
    {
        Drfl.movej(q99, 60, 30); //q99로 조인트 모션
        break;
    }
}
```

3.6.28 CDRFLEx.enable_alter_motion

- **기능**

경로 수정 기능을 활성화 한다. 경로 생성의 단위 주기는 100msec이며 입력 인자 n을 설정하여 경로 생성 주기(n*100msec)를 변경 할 수 있다. 입력 인자 ePathMode를 통해 alter_motion의 입력값의 의미를 2가지 모드(누적량 모드, 증분량 모드) 중 하나로 선택하여 사용 할 수 있다. 누적량 모드의 경우 현재의 모션경로에 대한 절대적 증분위치/자세만큼 경로 수정량이 반영된다. 증분량 모드의 경우 바로 현재의 절대적 증분위치/자세에 입력된 증분위치/자세만큼 경로 수정량이 추가되어 반영된다. 입력 인자 eTargetRef를 통해 기준 좌표계를 설정할 수 있다. 입력 인자 fLimitDpos, fLimitDposPer를 통해 각 각 누적량, 증분량의 한계치를 설정 할 수 있다. 한계치를 벗어나는 위치 값의 한계치에 수렴한 값으로 경로 수정량이 재 조정된다.

해당 함수는 M2.4 버전 이상에서만 사용 가능하다.

- **인수**

인수명	자료형	기본값	설명
iCycleTime	int	-	경로 생성 주기
ePathMode	Int	-	경로 수정 모드
eTargetRef	int	-	상수 및 열거형 정의 참조
fLimitDpos	float[2]	-	첫번째 값 : 이동량 제한 값[mm] 두번째 값 : 회전량 제한 값[deg]
fLimitDposPer	float[2]	-	첫번째 값 : 이동량 제한 값[mm] 두번째 값 : 회전량 제한 값[deg]

알아두기

- alter_motion 은 사용자 thread 내에서만 동작한다.
- eTargetRef 가 None 인 경우, global coordinate 적용(global coordinate 초기값은 COORDINATE_SYSTEM_BASE 이며, set_ref_coord 명령에 의해 설정 가능)

- **리턴**

값	설명
0	오류

값	설명
1	성공

- **예제**

```
float limit_dPOS[2] = {50, 90};
float limit_dPOS_per[2] = {50, 50};
Drfl.enable_alter_motion(5, PATH_MODE_DPOS, COORDINATE_SYSTEM_BASE,
limit_dPOS, limit_dPOS_per);
```

3.6.29 CDRFLEx.alter_motion

- **기능**

입력 인자 pos에 해당하는 양만큼 경로 수정을 진행한다.

해당 함수는 M2.4 버전 이상에서만 사용 가능하다.

주의

- alter_motion 은 사용자 thread 내에서만 동작한다.

알아두기

- alter_motion 은 enable_alter_motion 을 통해 경로보정기능이 활성화 된 경우에만 유효하다.
- enable_alter_motion 의 설정 값 fLimitPos, fLimitPosPer 에 따라 경로 수정량은 조정될 수 있다.
- pos 의 방향값은 Fixed XYZ 로 설정하여야 한다.

- **인수**

인수명	자료형	기본값	설명
pos	float[6]	-	position list

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
DWORD WINAPI ThreadFunc(void *arg){
   while(true){
      float pos[6] = {10, 0, 0, 10, 0, 0};
      Drfl.alter_motion(pos);
   }
}
...
int _tmain (int argc, _TCHAR* argv[]){
```

```
    HANDLE hThread;
    DWORD dwThreadID;
    hThread = CreateThread(NULL, 0, ThreadFunc, NULL, 0, &dwThreadID);
    if (hThread == 0) {
        printf("Thread Error₩n");
        return 0;
    }
    float limit_dPOS[2] = {50, 90};
    float limit_dPOS_per[2] = {50, 50};
    Drfl.enable_alter_motion(5, PATH_MODE_DPOS, COORDINATE_SYSTEM_BASE,
limit_dPOS, limit_dPOS_per);
}
```

3.6.30 CDRFLEx.disable_alter_motion

- **기능**

  경로 수정 기능을 비활성화 한다.

  해당 함수는 M2.4 버전 이상에서만 사용 가능하다.

- **인수**

  없음

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
DWORD WINAPI ThreadFunc(void *arg){
   while(true){
      float pos[6] = {10, 0, 0, 10, 0, 0};
      Drfl.alter_motion(pos);
   }
}
...
int _tmain (int argc, _TCHAR* argv[]){
   HANDLE hThread;
   DWORD dwThreadID;
   hThread = CreateThread(NULL, 0, ThreadFunc, NULL, 0, &dwThreadID);
   if (hThread == 0) {
      printf("Thread Error₩n");
      return 0;
   }
   float limit_dPOS[2] = {50, 90};
   float limit_dPOS_per[2] = {50, 50};
   Drfl.enable_alter_motion(5, PATH_MODE_DPOS, COORDINATE_SYSTEM_BASE,
```

```
limit_dPOS, limit_dPOS_per);
    Drfl.disable_alter_motion();.
}
```

3.7 로봇 설정 함수

3.7.1 CDRFLEx.add_tool

- **기능**

로봇 끝단에 장착될 Tool 정보를 안전상 사전에 등록하여 사용하기 위한 함수이다, 본 함수를 이용하여 등록된 Tool 정보는 메모리에 저장됨으로 재부팅 후 다시 설정해야 하지만, T/P 어플리케이션에서 등록한 경우에는 초기화 과정에서 추가됨으로 재사용이 가능하다.

- **인수**

인수명	자료형	기본값	설명
strSymbol	string	-	Tool 이름
fCog	float[3]	-	무게 중심
fInertia	float[6]	-	관성 정보

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float fCog[3]  = {10, 10, 10};
float finertia[6] = { 0, 0, 0, 0, 0, 0 };
// Tool 등록
drfl.add_tool("tool#1", 5.3f, fCog, finertia);

// 현재 장착된 Tool 선택
drfl.set_tool("tool#1");

// Tool 등록 해제
drfl.del_tool("tool#1");
```

3.7.2 CDRFLEx.del_tool

- **기능**

로봇 제어기에 사전에 등록된 Tool 정보를 삭제하기 위한 함수이다

- **인수**

인수명	자료형	기본값	설명
strSymbol	string	-	Tool 이름

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float fCog[3]  = {10, 10, 10};
float finertia[6] = { 0, 0, 0, 0, 0, 0 };
// Tool 등록
drfl.add_tool("tool#1", 5.3f, fCog, finertia);

// 현재 Tool 선택
drfl.set_tool("tool#1");

// Tool 등록 해제
drfl.del_tool("tool#1");
```

3.7.3 CDRFLEx.set_tool

- **기능**

로봇 제어기에 사전에 등록되어 있는 Tool 정보 중 현재 장착된 Tool에 대한 정보를 설정하는 함수이다. 현재 장착된 Tool 이 없을 경우, 빈 문자열을 전달하면 현재 설정되어 있는 정보가 초기화된다.

- **인수**

인수명	자료형	기본값	설명
strSymbol	string	-	Tool 이름

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float fCog[3]  = {10, 10, 10};
float finertia[6] = { 0, 0, 0, 0, 0, 0 };
// Tool 등록
drfl.add_tool("tool#1", 5.3f, fCog, finertia);

// 현재 Tool 선택
drfl.set_tool("tool#1");

// Tool 등록 해제
drfl.del_tool("tool#1");
```

3.7.4 CDRFLEx.get_tool

- **기능**

로봇 제어기에서 현재 설정된 Tool 정보를 가져오는 함수이다. 설정된 Tool 정보가 없을 경우, 빈 문자열이 반환된다.

- **인수**

없음

- **리턴**

값	설명
string	Tool 이름

- **예제**

```
string strTool = drfl.get_tool();
// 현재 장착되어 있는 Tool 정보 확인
if (strTool != “tool#1) {
    drfl.set_tool(“tool#1);
}
```

3.7.5 CDRFLEx.add_tcp

- **기능**

로봇 TCP 정보를 안전상 사전에 등록하여 사용하기 위한 함수이다, 본 함수를 이용하여 등록된 TCP 정보는 메모리에 저장됨으로 재부팅 후 다시 설정해야 하지만, T/P 어플리케이션에서 등록한 경우에는 초기화 과정에서 추가됨으로 재사용이 가능하다.

- **인수**

인수명	자료형	기본값	설명
strSymbol	string	-	TCP 이름
fPosition	float[6]	-	TCP 정보

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float fTCP[6] = { 10, 10, 10, 0, 0, 0 };
// TCP 등록
drfl.add_tcp("tcp#1", fTCP);

// 현재 TCP 설정
drfl.set_tcp("tcpl#1");

// TCP 등록 해제
drfl.del_tcp(“tcp#1”);
```

3.7.6 CDRFLEx.del_tcp

- **기능**

로봇 제어기에 사전에 등록된 TCP 정보를 삭제하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
strSymbol	string	-	TCP 이름

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float fTCP[6] = { 10, 10, 10, 0, 0, 0 };
// TCP 등록
drfl.add_tcp("tcp#1", fTCP);

// 현재 TCP 설정
drfl.set_tcp("tcpl#1");

// TCP 등록 해제
drfl.ConfigDelete(“tcp#1);
```

3.7.7 CDRFLEx.set_tcp

- **기능**

로봇 제어기에 사전에 등록되어 있는 TCP 정보 중 현재 장착된 TCP에 대한 정보를 설정하는 함수이다. 현재 장착된 TCP가 없을 경우, 빈 문자열을 전달하면 현재 설정되어 있는 정보가 초기화된다.

- **인수**

인수명	자료형	기본값	설명
strSymbol	string	-	Tool 이름

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
drfl.set_tcp("tcp#1);

float point[6] = { 756.6f, 511.6f, 876.0f, 44.5f, 86.7f, 55.7f };
float vel[2] = {30, 0 };
float acc[2] = {30, 0 };

drfl.MoveL(point, vel, vel);
```

3.7.8 CDRFLEx.get_tcp

- **기능**

로봇 제어기에서 현재 설정된 TCP 정보를 가져오는 함수이다. 설정된 Tool 정보가 없을 경우, 빈 문자열이 반환된다.

- **인수**

없음

- **리턴**

값	설명
string	TCP 이름

- **예제**

```
string strTCP = drfl.get_tcp();
// 현재 장착되어 있는 TCP 정보 확인
if (strTCP != “tcp#1) {
      drfl.set_tcp(“tcp#1);
}
```

3.7.9 CDRFLEx.set_tool_shape

- 기능

로봇 제어기에 사전에 등록되어 있는 ToolShape 정보 중 현재 장착된 ToolShape에 대한 정보를 설정하는 함수이다. 현재 장착된 Tool이 없을 경우, 빈 문자열을 전달하면 현재 설정되어 있는 정보가 초기화된다.

해당 함수는 M2.4 버전 이상에서만 사용 가능하다.

- 인수

인수명	자료형	기본값	설명
strSymbol	string	-	ToolShape 이름

- 리턴

값	설명
0	오류
1	성공

- 예제

```
Drfl.set_tool_shape(“tool_shape1”) //TP에서 등록된 “tool_shape1”의 정보를 활성화 한다.
```

3.7.10 CDRFLEx.get_workpiece_weight

- **기능**

작업물의 무게를 측정하여 반환한다.

- **인수**

없음

- **리턴**

값	설명
float	측정 무게 값

- **예제**

```
float weight = Drfl.get_workpiece_weight();
```

3.7.11 CDRFLEx.reset_workpiece_weight

- **기능**

소재의 무게를 측정하기 전 알고리즘의 초기화를 위해 소재의 무게 정보를 초기화한다.

- **인수**

없음

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
Drfl.reset_workpiece_weight()
```

3.7.12 CDRFLEx.set_singularity_handling

- **기능**

task motion에서 특이점의 영향으로 path diaviation이 발생할 경우 대응 정책을 사용자가 선택할 수 있도록 하다. mode의 설정은 아래와 같은 설정이 가능하다.

- 자동회피 모드(Default) : SINGULARITY_AVOIDANCE_AVOID
- 경로 우선 : SINGULARITY_AVOIDANCE_STOP
- 속도 가변 : SINGULARITY_AVOIDANCE_VEL

기본 설정은 자동회피 모드이며, 이 설정의 경우 특이점으로 인한 불안정성을 감소시키지만 path tracking 정확도가 감소한다.

경로 우선 설정의 경우 singularity 의 영향으로 불안정성이 발생할 가능성이 있는 경우, 감속 후 warning 메시지를 출력하고 해당 Task를 종료한다.

속도 가변 설정의 경우 특이점으로 인한 불안정을 감소시키면서 path tracking 정확도를 높인다. 하지만 특이점 구간에서 TCP 속도 변경이 발생한다.

- **인수**

인수명	자료형	기본값	설명
eMode	SINGULARITY_AVOIDANCE	-	열거형 및 상수 정의 참조

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float p1[6] = {400, 500, 800, 0, 180, 0};
float p2[6] = {400, 500, 500, 0, 180, 0};
float p3[6] = {400, 500, 200, 0, 180, 0};
Drfl.set_singularity_handling(SINGULARITY_AVOIDANCE_AVOID); // 특이점 자동회피 모드
Drfl.MoveL(p1, 10, 20);
Drfl.set_singularity_handling(SINGULARITY_AVOIDANCE _STOP); // Task 모션 경로 우선
```

```
Drfl.MoveL(p2, 30, 60);
Drfl.set_singularity_handling(SINGULARITY_AVOIDANCE_VEL); // 특이점 속도 가변 모
드
```

3.7.13 CDRFLEx.setup_monitoring_version

- **기능**

로봇 제어기에서 전송되는 모니터링 정보의 버전 정보를 설정하기 위한 함수이다.

해당 함수는 M2.5 버전 이상에서만 사용 가능하다.

- **인수**

인수명	자료형	기본값	설명
iVersion	int	-	버전 정보 0 : 버전 v0 1 : 버전 v1

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
Drfl.setup_monitoring_version(1); //모니터링 데이터 버전 정보를 1로 설정
```

3.7.14 CDRFLEx.config_program_watch_variable

로봇 제어기에서 프로그램 실행 시 프로그램 내부 변수를 모니터링 하기 위하여 모니터링할 변수 명을 설정하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
eDivision	VARIABLE_TYPE	-	상수 및 열거형 정의 참조
eType	TYPE_TYPE	-	상수 및 열거형 정의 참조
strName	string	-	128바이트 변수명 문자열
strData	string	-	128바이트 데이터 문자열

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
Drfl.config_program_watch_variable(VARIABLE_TYPE_INSTALL, DATA_TYPE_FLOAT,
"var", "1.22"); // 설치변수 “var”을 float 1.22로 저장
```

3.8 I/O 제어 함수

3.8.1 CDRFLEx.set_tool_digital_output

- **기능**

로봇 제어기에서 로봇 끝단에 장착된 디지털 접점에 신호를 출력하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
eGpioIndex	enum.GPIO_TOOL_DIGITAL_INDEX	-	상수 및 열거형 정의 참조
bOnOff	bool	-	출력하고자 하는 데이터 • ON: 1 • OFF: 0

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
//로봇 암의 디지털 1번 출력 접점 출력
drfl.set_tool_digital_output(GPIO_TOOL_DIGITAL_INDEX_1, 1);
```

3.8.2 CDRFLEx.get_tool_digital_input

- **기능**

로봇 제어기에서 로봇 끝단에 장착된 디지털 접점의 신호를 확인하기 함수이다.

- **인수**

인수명	자료형	기본값	설명
eGpioIdex	enum.GPIO_TOOL_DIGITAL_INDEX	-	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	OFF
1	ON

- **예제**

```
//로봇 암의 디지털 1번 입력 접점 확인
bool bSignal = drfl.get_tool_digital_input(GPIO_TOOL_DIGITAL_INDEX_1);
if (bSignal == True) {
  // do something
}
```

3.8.3 CDRFLEx.get_tool_digital_output

- **기능**

로봇 제어기에서 로봇 끝단에 장착된 디지털 접점의 신호를 확인하기 함수이다.

- **인수**

인수명	자료형	기본값	설명
eGpioIdex	enum.GPIO_TOOL_DIGITAL_INDEX	-	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	OFF
1	ON

- **예제**

```
//로봇 암의 디지털 1번 출력 접점 확인
bool bSignal = drfl.get_tool_digital_output(GPIO_TOOL_DIGITAL_INDEX_1);
if (bSignal == True) {
  // do something
}
```

3.8.4 CDRFLEx.set_digital_output

- **기능**

로봇 제어기에서 컨트롤 박스에 장착된 디지털 접점에 신호를 출력하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
eGpioIdex	enum.GPIO_CTRLBOX_DIGITAL_INDEX	-	상수 및 열거형 정의 참조
bOnOff	bool	-	출력하고자 하는 데이터 • ON: 1 • OFF: 0

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
// 컨트롤박스 디지털 1번 출력 접점 출력
drfl.set_digital_output(GPIO_CTRLBOX_DIGITAL_INDEX_1, 1);
```

3.8.5 CDRFLEx.get_digital_input

- **기능**

로봇 제어기에서 컨트롤 박스에 장착된 디지털 접점의 신호를 확인하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
eGpioIdex	enum.GPIO_CTRLBOX_DIGITAL_INDEX	-	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	OFF
1	ON

- **예제**

```
//컨트롤 박스의 디지털 1번 입력 접점 체크
bool bSignal = drfl.get_digital_input(GPIO_CTRLBOX_DIGITAL_INDEX_1);
if (bSignal ==TRUE) {
  // do something
}
```

3.8.6 CRDFL.get_digital_output

- **기능**

로봇 제어기에서 컨트롤 박스에 장착된 디지털 접점의 신호를 확인하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
eGpioIdex	enum.GPIO_CTRLBOX_DIGITAL_INDEX	-	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	OFF
1	ON

- **예제**

```
//컨트롤 박스의 디지털 1번 출력 접점 체크
bool bSignal = drfl.get_digital_output(GPIO_CTRLBOX_DIGITAL_INDEX_1);
if (bSignal ==TRUE) {
  // do something
}
```

3.8.7 CDRFLEx.set_mode_analog_input

- **기능**

로봇 제어기에서 컨트롤 박스에 장착된 아날로그 입력 접점에 대한 채널 모드를 설정하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
eGpioIdex	enum.GPIO_CTRLBOX_ANALOG_INDEX_	-	상수 및 열거형 정의 참조
mod	enum.GPIO_ANALOG_TYPE	GPIO_ANALOG_TYPE_CURRENT	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
// 컨트롤 박스의 아날로그 1번 입력 접점을 전류 모드로 설정함
drfl.set_mode_analog_input(GPIO_CTRLBOX_ANALOG_INDEX_1,
GPIO_ANALOG_TYPE_CURRENT);

// 컨트롤 박스이 아놀로그 2번 입력 접점을 전압 모드로 설정함.
drfl.set_mode_analog_input(GPIO_CTRLBOX_ANALOG_INDEX_2,
GPIO_ANALOG_TYPE_VOLTAGE);
```

3.8.8 CDRFLEx.set_mode_analog_output

- **기능**

로봇 제어기에서 컨트롤 박스에 장착된 아날로그 출력 접점에 대한 채널 모드를 설정하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
eGpioIdex	enum.GPIO_CTRLBOX_ANALOG_INDEX_	-	상수 및 열거형 정의 참조
mod	enum.GPIO_ANALOG_TYPE	GPIO_ANALOG_TYPE_CURRENT	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
// 컨트롤박스의 아날로그 1번 출력 접점을 전류 모드로 설정함
drfl.set_mode_analog_output(pCtrl, GPIO_CTRLBOX_ANALOG_INDEX_1,
GPIO_ANALOG_TYPE_CURRENT);

// 컨트롤박스의 아날로그 2번 출력 접점을 전압 모드로 설정함
drfl.set_mode_analog_output(pCtrl, GPIO_CTRLBOX_ANALOG_INDEX_2,
GPIO_ANALOG_TYPE_VOLTAGE);
```

3.8.9 CDRFLEx.set_analog_output

- **기능**

로봇 제어기에서 컨트롤 박스에 장착된 아날로그 접점에 신호를 출력하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
eGpioIdex	enum.GPIO_CTRLBOX_ANALOG_INDEX_	-	상수 및 열거형 정의 참조
fValue	float	-	아날로그 신호 출력 • 전류 모드인 경우: 4.0~20.0 [mA] • 전압 모드인 경우: 0~10.0 [V]

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
// 컨트롤 박스의 아날로그 1번 출력 접점을 전류 모드로 설정
drfl.set_mode_analog_output(GPIO_CTRLBOX_ANALOG_INDEX_1,
GPIO_ANALOG_TYPE_CURRENT);

// 컨트롤 박스의 아날로그 1번 출력 접접에 5.2mA 출력
drfl.set_analog_output(GPIO_CTRLBOX_ANALOG_INDEX_1, 5.2);
```

3.8.10 CDRFLEx.get_analog_input

- **기능**

로봇 제어기에서 컨트롤 박스에 장착된 아날로그 접점의 신호를 확인하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
eGpioIdex	enum.GPIO_CTRLBOX_ANALOG_INDEX_	-	상수 및 열거형 정의 참조

- **리턴**

값	설명
아날로그 신호 입력	• 전류 모드인 경우: 4.0~20.0 [mA] • 전압 모드인 경우: 0~10.0 [V]

- **예제**

```
// 컨트롤 박스의 아날로그 1번 입력 점접을 전류 모드로 설정
drfl.set_mode_analog_output(GPIO_CTRLBOX_ANALOG_INDEX_1,
GPIO_ANALOG_TYPE_CURRENT);

// 컨트롤 박스의 아날로그 1번 입력 점접 전류값 확인
float fCurrent = drfl.get_analog_input(GPIO_CTRLBOX_ANALOG_INDEX_1);
```

3.8.11 CDRFLEx.add_modbus_signal

- **기능**

로봇 제어기에서 Modbus의 I/O 신호 사전에 등록하여 사용하기 위한 함수이다, 본 함수를 이용하여 등록된 Modbus I/O 신호 정보는 메모리에 저장됨으로 재부팅 후 다시 설정해야 하지만, T/P 어플리케이션에서 등록한 경우에는 초기화 과정에서 추가됨으로 재사용이 가능하다.

- **인수**

인수명	자료형	기본값	설명
strSymbol	string	-	modbus signal 이름
strIpAddress	string	-	modbus 모듈 ip 주소
nPort	unsigned short	-	modbus 모듈 port
eRegType	enum	-	상수 및 열거형 정의 참조
iRegIndex	unsigned short	-	Modbus signal의 index
nRegValue	unsigned short	0	type이 MODBUS_REGISTER_TYPE_COILS 또는 MODBUS_REGISTER_TYPE_HOLDING_REGISTER 일 때 출력값 (그 외 경우에는 무시됩니다.)
nSlaveId	unsigned short	255	ModbusTCP 모듈의 Slave ID(0 or 1-247 or 255)

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
/*
  Modbus IO를 연결하고 접점을 할당하는 예제
  Modbus IO IP: 192.168.127.254
  input 2점: “di1”,”di2”
*/

// set <modbus> input : “di1”, “di2”
```

```
drfl.add_modbus_signal("di1", "192.168.127.254" , 502,
MODBUS_REGISTER_TYPE_DISCRETE_INPUTS, 0, 0);
drfl.add_modbus_signal("di2", "192.168.127.254" , 502,
MODBUS_REGISTER_TYPE_DISCRETE_INPUTS, 0, 0);
```

3.8.12 CDRFLEx.del_modbus_signal

- **기능**

로봇 제어기에 사전에 등록된 Modbus I/O 신호 정보를 삭제하기 위한 함수이다

- **인수**

인수명	자료형	기본값	설명
strSymbol	string	-	등록된 modbus 신호의 이름

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
/*
 Modbus IO 신호가 “di1”, ”do1” 로 등록되어 있는데,
 이 신호 등록을 삭제하고자 하는 경우.
*/
drfl.del_modbus_signal("di1")      // “di1” 접점 등록 삭제
drfl.del_modbus_signal("do1")      // “do1” 접점 등록 삭제
```

3.8.13 CDRFLEx.set_modbus_output

- **기능**

로봇 제어기에서 Modbus I/O 신호 접점에 신호를 출력하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
strSymbol	string	-	modbus 이름
nValue	unsigned short	-	• Modbus digital I/O 인 경우: 0 or 1 • Modbus analog I/O 인 경우: 데이터

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
//Modbus digital I/O가 연결되어 있고, 신호가 “do1”, “do2” 로 등록되어 있는 경우
drfl.set_modbus_output("do1",1);
drfl.set_modbus_output("do2",0);
```

3.8.14 CDRFLEx.get_modbus_input

- **기능**

로봇 제어기에서 Modbus I/O 신호 접점의 신호를 확인하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
strSymbol	string	-	modbus 이름

- **리턴**

값	설명
unsigned short	• Modbus Digital I/O 인 경우: 0 or 1 • Modbus Analog 모듈인 경우: 데이터

- **예제**

```
//Modbus digital I/O가 연결되어 있고, 신호가 “di1”, “di2” 로 등록되어 있는 경우
unsigned short siganl1 = drfl.get_modbus_input("di1");
unsigned short signal2 = drfl.get_modbus_input("di2");
if ( signal1 == 1 && signal2 == 1) {

    // do something…
}
```

3.9 프로그램 제어 함수

3.9.1 CDRFLEx.drl_start

- **기능**

로봇 제어기에서 DRL 언어로 구성된 프로그램(태스크)을 실행하기 위한 함수이다.

- **인수**

인수명	자료형	기본값	설명
eRobotSystem	enum.ROBOT_SYSTEM		상수 및 열거형 정의 참조
strDrlProgram	string		실행 시킬 DRL 프로그램 문자열

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
string strDrlProgram = "loop = 0\nwhile loop < 3:\n
movej(posj(10,10.10,10,10.10), vel=60, acc=60)\n
movej(posj(00,00.00,00,00.00), vel=60, acc=60)\n  loop+=1\n
movej(posj(10,10.10,10,10.10), vel=60, acc=60)";
if (Drfl.get_robot_state() == STATE_STANDBY) {
    Drfl.set_robot_mode(ROBOT_MODE_AUTONOMOUS);
    if (Drfl.get_robot_mode() == ROBOT_MODE_AUTONOMOUS) {
    // 자동모드
    ROBOT_SYSTEM eTargetSystem = ROBOT_SYSTEM_VIRTUAL;
    Drfl.drl_start(eTargetSystem, strDrlProgram);
    }
}
```

알아두기

- 로봇 운용 상태가 지령 대기상태(STATE_STANDBY)이어야 하며, 로봇 모드가 자동모드일 때 사용해야 정상 동작한다.
- DRL 프로그램 작성은 별로 Programming Manual 문서을 참조해서 작성해야 합니다.

3.9.2 CDRFLEx.drl_stop

- **기능**

로봇 제어기에서 현재 실행중인 DRL 프로그램(태스크)을 정지하기 위한 함수입니다. 인자로 받는 eStopType에 따라 다르게 정지하며, 현재 수행하고 있는 구간의 모션을 정지합니다.

- **인수**

인수명	자료형	기본값	설명
eStopType	Enum. STOP_TYPE	STOP_TYPE_QUICK	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
DRL_PROGRAM_STATE eProgramState = drfl. get_program_state();
if ((eProgramState == DRL_PROGRAM_STATE_PLAY) ||
    (eProgramState == DRL_PROGRAM_STATE_HOLD)) {

    drfl.drl_stop(STOP_TYPE_QUICK);
    //…
}
```

알아두기

・ 프로그램이 정상 및 에러로 종료된 경우에도, 반드시 프로그램 정지 명령을 수행해야 한다.

3.9.3 CDRFLEx.drl_pause

- **기능**

로봇제어기에서 현재 실행 중인 DRL 프로그램(태스크)을 일시 정지하기 위한 함수이다.

- **인수**

없음

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
if (drfl.get_program_state() == DRL_PROGRAM_STATE_PLAY) {
    drfl.drl_pause();
}
```

3.9.4 CDRFLEx.drl_resume

- **기능**

로봇 제어기에서 현재 일시 정지된 DRL 프로그램(태스크)을 재개하기 위한 함수이다.

- **인수**

없음

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
if (drfl. get_program_state() == DRL_PROGRAM_STATE_HOLD) {
    bool bResult = drfl.drl_resume();
    //...
}
```

3.9.5 CDRFLEx.change_operation_speed

- **기능**

DRL로 작성된 프로그램의 작동 속도를 조절한다. 인수는 현재 설정된 속도에 대한 상대적인 비율을 백분율로 호나산한 값으로, 1에서 100까지의 값을 가진다.

- **인수**

인수명	자료형	기본값	설명
speed	float	-	operation speed(1~100)

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
Drfl.change_operation_speed(10);
string strDrlProgram = "loop = 0\nwhile loop < 3:\n
movej(posj(10,10.10,10,10.10), vel=60, acc=60)\n
movej(posj(00,00.00,00,00.00), vel=60, acc=60)\n  loop+=1\n
movej(posj(10,10.10,10,10.10), vel=60, acc=60)";

if (Drfl.get_robot_state() == STATE_STANDBY) {
    Drfl.set_robot_mode(ROBOT_MODE_AUTONOMOUS);
    if (Drfl.get_robot_mode() == ROBOT_MODE_AUTONOMOUS) {
    // 자동모드
    ROBOT_SYSTEM eTargetSystem = ROBOT_SYSTEM_VIRTUAL;
    Drfl.drl_start(eTargetSystem, strDrlProgram);
    }
}
```

3.9.6 CDRFLEx.save_sub_program

- **기능**

작성한 DRL언어로 구성된 프로그램(태스크)를 서브 프로그램으로 저장한다.

- **인수**

인수명	자료형	기본값	설명
iTargetSystem	SUB_PROGRAM	-	상수 및 열거형 정의 참조
strFileName	string	-	256바이트 파일 이름 정보
strDrlProgram	string	-	N개의 문자열 버퍼

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
string drl_string= "movej([0,0,0,0,0,0], 60, 30)";

Drfl.save_sub_program(SUB_PROGRAM_SAVE, "sub_test", drl_string.c_str());
```

3.9.7 tp_popup_response

- **기능**

DRL프로그램 동작 시 유형화 메시지(Popup) 출력 후 사용자 응답에 따른 다음 동작(일시 정지된 프로그램의 중지 및 재개)을 제어한다.

- **인수**

인수명	자료형	기본값	설명
eRes	POPUP_RESPONSE	-	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
// DRL 명령어 tp_popup 실행 시
Drfl.tp_popup_response(POPUP_RESPONSE_RESUME);
```

3.9.8 tp_get_user_input_response

- **기능**

DRL프로그램 동작 시 사용자 입력 요구 시 사용자 입력 정보를 전달한다.

- **인수**

인수명	자료형	기본값	설명
strUserInput	string	-	256바이트 사용자 입력 문자열

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
// DRL 명령어 tp_get_user_input 실행 시
Drfl.tp_get_user_input_response("tp get user input response");
```

3.10 힘/강성 제어 및 기타 사용자 편의 함수

3.10.1 CDRFLEx.parallel_axis

- **기능**

입력된 기준 좌표계(eSourceRef) 기준의 3개의 포즈(fTargetPos1, fTargetPos2, fTargetPos3)가 이루는 평면의 normal vector 방향에 Tool 좌표계의 지정축의 방향(eTaskAxis)을 일치시킨다. 이 때 로봇의 TCP 위치는 현재 위치를 유지한다.

- **인수**

인수명	자료형	기본값	설명
fTargetPos1	float[6]	-	6개의 Task Space 정보
fTargetPos2	float[6]	-	6개의 Task Space 정보
fTargetPos3	float[6]	-	6개의 Task Space 정보
eTaskAxis	TASK_AXIS	-	상수 및 열거형 정의 참조
eSourceRef	COORDINATE_SYSTEM	-	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float x0[6] = {0, 0, 90, 0, 90, 0};
Drfl.movej(x0, 60, 30);
float x1[6] = {0, 500, 700, 30, 0, 90};
float x2[6] = {500, 0, 700, 0, 0, 45};
float x3[6] = {300, 100, 500, 45, 0, 45};
Drfl.parallel_axis(x1, x2, x3, TASK_AXIS_X);
```

3.10.2 CDRFLEx.parallel_axis

- **기능**

입력된 기준 좌표계(eTargetRef) 기준의 벡터(fSourceVec) 방향에 Tool 좌표계의 지정축의 방향을 일치시킨다. 이 때 로봇의 TCP 위치는 현재 위치를 유지한다.

- **인수**

인수명	자료형	기본값	설명
eSourceVec	float[3]	-	3개의 XYZ 정보
eTaskAxis	TASK_AXIS	-	상수 및 열거형 정의 참조
eTargetRef	COORDINATE_SYSTEM	-	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float v[3] = {1000, 700, 300};

Drfl.parallel_axis(v, TASK_AXIS_X, COORDINATE_SYSTEM_BASE);
```

3.10.3 CDRFLEx.align_axis

입력된 기준 좌표계(eSourceRef) 기준의 3개의 포즈(fTargetPos1, fTargetPos2, fTargetPos3)가 이루는 평면의 normal vector 방향에 Tool 좌표계의 지정축의 방향(eTaskAxis)을 일치시킨다. 이 때 로봇의 TCP 위치는 현재 위치를 유지한다.

- **인수**

인수명	자료형	기본값	설명
fTargetPos1	float[6]	-	6개의 Task Space 정보
fTargetPos2	float[6]		6개의 Task Space 정보
fTargetPos3	float[6]		6개의 Task Space 정보
fSourceVec	float[3]		3개의 XYZ 정보
eTaskAxis	TASK_AXIS	-	상수 및 열거형 정의 참조
eTargetRef	COORDINATE_SYSTEM	-	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float x1[6] = {0, 500, 700, 30, 0, 0};
float x2[6] = {500, 0, 700, 0, 0, 0};
float x3[6] = {300, 100, 500, 0, 0, 0};
float pos[3] = {400, 400, 500};
Drfl.align_axis(x1, x2, x3, pos, TASK_AXIS_X, COORDINATE_SYSTEM_BASE);
```

3.10.4 CDRFLEx.align_axis

- **기능**

입력된 기준 좌표계(eTargetRef) 기준의 벡터(fSourceVec) 방향에 Tool 좌표계의 지정축의 방향을 일치시킨다. 이 때 로봇의 TCP 위치는 fTargetVec위치로 이동시킨다.

- **인수**

인수명	자료형	기본값	설명
eTargetVec	float[3]	-	3개의 XYZ 정보
eSourceVec	float[3]		3개의 XYZ 정보
eTaskAxis	TASK_AXIS	-	상수 및 열거형 정의 참조
eTargetRef	COORDINATE_SYSTEM	-	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float v1[3] = {400, 400, 500};
float v2[3] = {350, 37, 430};
Drfl.align_axis(v1, v2, TASK_AXIS_X, COORDINATE_SYSTEM_BASE);
```

3.10.5 CDRFLEx.is_done_bolt_tightening

- **기능**

  툴의 조임 토크를 모니터링하여 주어진 시간 내에 설정된 토크(m)에 도달한 경우 True를 반환하고, 주어진 시간을 초과한 경우 False를 반환한다.

- **인수**

인수명	자료형	기본값	설명
eForceAxis	FORCE_AXIS	-	상수 및 열거형 정의 참조
fTargetTor	float	0	Target torque
fTimeout	float	0	Monitoring duration [sec]

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float p0[6] = {0,0,90,0,90,0};
Drfl.movej(p0, 60, 30);
float stx[6] = {3000, 3000, 3000, 200, 200, 200};
Drfl.task_compliance_ctrl(stx);
float x1[6] = {559, 34.5, 651.5, 0, 180, 60};
float velx[2] = {50, 50};
float accx[2] = {50, 50};
Drfl.amovel(x1, velx, accx);
bool res = Drfl.is_done_bolt_tightening(FORCE_AXIS_Z, 10, 5);
int x = 0;
if(res){
    x = 1;
}
else{
    x = 2;
}
```

3.10.6 CDRFLEx.task_compliance_ctrl

- **기능**

기존에 설정한 기준 좌표계를 기준으로 태스크 Compliance control을 시작한다.

- **인수**

인수명	자료형	기본값	설명
fTargetStiffness	float[6]	[3000, 3000, 3000, 200, 200, 200]	3개의 Translational 강성 정보 3개의 회전 강성 정보
eForceReference	COORDINATE_SYSTEM	COORDINATE_SYSTEM_TOOL	상수 및 열거형 정의 참조
fTargetTime	float	0	강성 변화 시간 [sec] . 범위 0 ~ 1.0 . 주어진 시간 동안 linear transition

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float p0[6] = {0,0,90,0,90,0};
Drfl.movej(p0, 60, 30);
float stx[6] = {3000, 3000, 3000, 200, 200, 200};
Drfl.task_compliance_ctrl(stx);
```

3.10.7 CDRFLEx.release_compliance_ctrl

- **기능**

Compliance control을 종료하고 현재 위치에서 위치 제어를 시작한다.

- **인수**

없음

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float p0[6] = {0, 0, 90, 0, 90, 0};
Drfl.movej(p0, 60, 30);
float stx[6] = {3000, 3000, 3000, 200, 200, 200};
Drfl.task_compliance_ctrl(stx);
float stx2[6] = {1, 2, 3, 4, 5, 6};
Drfl.set_stiffnessx(stx2);
Drfl.release_compliance_ctrl();
```

3.10.8 CDRFLEx.set_stiffnessx

- **기능**

전역으로 설정된 좌표계(set_ref_coord 참조) 기준으로 강성값을 설정한다. 현재 강성 또는 기본값으로부터 STX로 주어진 time값 동안 linear transition한다. Translation 강성의 사용자 범위는 0 ~ 20000N/m, Rotational 강성의 사용자 범위는 0 ~ 400NM/rad이다.

- **인수**

인수명	자료형	기본값	설명
fTargetStiffness	float[6]	-	3개의 Translational 강성 정보 3개의 회전 강성 정보
eForceReference	COORDINATE_SYSTEM	COORDINATE_SYSTEM_TOOL	상수 및 열거형 정의 참조
fTargetTime	float	0	강성 변화 시간 [sec] . 범위 0 ~ 1.0 . 주어진 시간 동안 linear transition

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
float p0[6] = {0, 0, 90, 0, 90, 0};
Drfl.movej(p0, 60, 30);
float stx[6] = {3000, 3000, 3000, 200, 200, 200};
Drfl.task_compliance_ctrl(stx);
float stx2[6] = {1, 2, 3, 4, 5, 6};
Drfl.set_stiffnessx(stx2);
Drfl.release_compliance_ctrl();
```

3.10.9 CDRFLEx.calc_coord

- **기능**

지정한 좌표계(ref) 기준의 최대 4개의 입력점(x1~x4) 및 입력 모드(mod)를 기반으로 새로운 직교 좌표계를 계산한다. 여기서 입력 모드는 입력점의 개수가 2개인 경우에만 유효하다.
입력점의 개수가 1개인 경우, x1의 위치와 회전으로 좌표계가 계산된다.
입력점의 개수가 2개인 경우 입력모드가 0일 때, x1에서 x2로 향하는 벡터가 x방향, x축에 직교하는 평면에 투영된 현재의 Tool-z방향이 사용자 좌표계의 z방향으로 정의되며 x1의 위치가 원점이 되도록 좌표계가 계산된다.
입력점의 개수가 2개인 경우 입력모드가 1일 때, x1에서 x2로 향하는 벡터가 x방향, x축에 직교하는 평면에 투영된 x1의 z축방향이 사용자 좌표계의 z방향으로 정의되며 x1의 위치가 원점이 되도록 좌표계가 계산된다.
입력점의 개수가 3개인 경우, x1에서 x2로 향하는 벡터가 x방향으로 정의되며, x1에서 x3으로 향하는 벡터를 v라고 하였을 경우 z방향은 오른손법칙에 따라 x방향 벡터 곱하기 v로 정의되며, x1의 위치가 원점이 되도록 좌표계가 계산된다.
입력점의 개수가 4개인 경우, 입력점의 개수가 3개인 경우와 축의 방향은 동일하며 원점의 위치가 x4의 위치가 되도록 좌표계가 계산된다.
해당 함수는 M2.5 버전 이상에서만 사용 가능하다.

- **인수**

인수명	자료형	기본값	설명
nCnt	unsigned short	-	입력점 개수
nInputMode	unsigned short	-	입력 모드 (입력점개수가 2개인 경우에만 유효함) 0: 현재 Tool-z방향 기준으로 사용자 좌표계의 z방향 정의 1: x1의 z방향 기준으로 사용자 좌표계의 z방향 정의
eTargetRef	COORDINATE_SYSTEM	COORDINATE_SYSTEM_BASE	상수 및 열거형 정의 참조
fTargetPos1	float[6]		6개의 Task Space 정보

인수명	자료형	기본값	설명
fTargetPos2	float[6]		6개의 Task Space 정보
fTargetPos3	float[6]		6개의 Task Space 정보
fTargetPos4	float[6]		6개의 Task Space 정보

- **리턴**

값	설명
enum.ROBOT_POSE	상수 및 열거형 정의 참조

- **예제**

```
float pos1[6] = {500, 30, 500, 0, 0, 0};
float pos2[6] = {400, 30, 500, 0, 0, 0};
float pos3[6] = {500, 30, 600, 45, 180, 45};
float pos4[6] = {500, -30, 600, 0, 180, 0};
ROBOT_POSE* pose_user1 = Drfl.calc_coord(4, 0, COORDINATE_SYSTEM_BASE, pos1,
pos2, pos3, pos4);
for (int i=0; i<NUM_TASK; i++)
{
    cout << pose_user1->_fPosition[i] << endl;
}
```

3.10.10 CDRFLEx.set_user_cart_coord

- **기능**

기준 좌표계(eTargetRef) 기반의 새로운 사용자 좌표계를 설정할 수 있다. 총 20개의 사용자좌표계를 설정할 수 있으며, 20개가 넘어가면 새로운 직교 좌표계 설정이 불가하다.

- **인수**

인수명	자료형	기본값	설명
iReqId	int	-	식별자(ID) 0 : 자동 생성 101 ~ 120 : 지정 생성
fTargetPos	float[6]	-	6개의 TaskSpace 정보
eTargetRef	COORDINATE_SYSTEM	COORDINATE_SYSTEM_BASE	상수 및 열거형 정의 참조

- **리턴**

값	설명
int	Coordinate 설정 성공 설정된 Coordinate ID (101 ~ 120)

- **예제**

```
float pos1[6] = {500, 30, 500, 0, 0, 0};
int id = Drfl.set_user_cart_coord(0, pos1, COORDINATE_SYSTEM_BASE);
```

3.10.11 CDRFLEx.set_user_cart_coord

- **기능**

사용자가 입력 좌표계(eTargetRef) 기준의 포즈 fTargetPos[0], fTargetPos[1], fTargetPos[2]를 이용하여 새로운 직교 좌표계를 설정할 수 있다. 총 20개의 사용자좌표계를 설정할 수 있으며, 20개가 넘어가면 새로운 직교 좌표계 설정이 불가하다.

- **인수**

인수명	자료형	기본값	설명
fTargetPos	float[3][6]	-	6개의 Task Space 정보 #1 6개의 Task Space 정보 #2 6개의 Task Space 정보 #3
fTargetOrg	float[3]	-	사용자 원점 정보
eTargetRef	COORDINATE_SYSTEM	COORDINATE_SYSTEM_BASE	상수 및 열거형 정의 참조

- **리턴**

값	설명
int	Coordinate 설정 성공 설정된 Coordinate ID (101 ~ 120)

- **예제**

```
float x1[6] = {0,500,700,0,0,0};
float x2[6] = {500,0,700,0,0,0};
float x3[6] = {300,100,500,0,0,0};
float org[3] = {10, 20, 30};
float x[3][6] = {{0,500,700,0,0,0}, {500,0,700,0,0,0}, {300,100,500,0,0,0}};
Drfl.set_user_cart_coord(x, org, COORDINATE_SYSTEM_BASE);
```

3.10.12 CDRFLEx.set_user_cart_coord

- **기능**

입력좌표계 기준의 벡터 fTargetVec[0]와 fTargetVec[1]을 사용하여 새로운 직교 좌표계를 설정한다. 직교 좌표계의 원점은 입력 좌표계(eTargetRef) 기준의 fTargetOrg에 위치한다. 총 20개의 사용자좌표계를 설정할 수 있으며, 20개가 넘어가면 새로운 직교 좌표계 설정이 불가하다.

해당 함수는 M2.5 hotfix 버전 이상에서만 사용 가능하다.

- **인수**

인수명	자료형	기본값	설명
fTargetVec	float[2][3]	-	3개의 XYZ 정보 #1 3개의 XYZ 정보 #2
fTargetOrg	float[3]	-	사용자 원점 정보
eTargetRef	COORDINATE_SYSTEM	COORDINATE_SYSTEM_BASE	상수 및 열거형 정의 참조

- **리턴**

값	설명
int	설정된 Coordinate ID (101 ~ 120)

- **예제**

```
float vec[2][3] = {{-1, 1, 1}, {1, 1, 0}};
float org[3] = {370.9, -419.7, 651.5};

int user = Drfl.set_user_cart_coord(vec, org);
```

3.10.13 CDRFLEx.overwrite_user_cart_coord

- **기능**

요청하는 ID의 사용자 좌표계의 좌표 위치(fTargetPos), 기준 좌표계(eTargetRef) 정보를 변경한다.

해당 함수는 M2.5 버전 이상에서만 사용 가능하다.

- **인수**

인수명	자료형	기본값	설명
bTargetUpdate	bool	-	0 : 좌표계 미변경 1 : 좌표계 변경
iReqId	int	-	식별자
fTargetPos	float[6]	-	6개의 Task Space 정보
eTargetRef	COORDINATE_SYSTEM	COORDINATE_SYSTEM_BASE	상수 및 열거형 정의 참조

좌표계 변경(bTargetUpdate) 변수가 0일 경우에는 프로그램 실행시에만 유저 좌표계가 변경되어 유지되어야 하며, 1일 경우에는 상위제어기의 저장된 유저 좌표계 정보 자체가 변경되어야 한다.

- **리턴**

값	설명
int	변경된 Coordinate의 ID(101~120)

- **예제**

```
float pos1[6] = {30, 40, 50, 0, 0, 0};
int pose_user1 = Drfl.set_user_cart_coord(0, pos1, COORDINATE_SYSTEM_BASE);

float pos2[6] = {100, 150, 200, 45, 180, 0};
int result = Drfl.overwrite_user_cart_coord(0, pose_user1, pos2);

cout << result << endl;
```

3.10.14 CDRFLEx.get_user_cart_coord

- **기능**

해당하는 ID(iReqId)의 사용자 좌표계의 정보인 참조 기준 및 위치 정보를 조회한다.
해당 함수는 M2.5 버전 이상에서만 사용 가능하다.

- **인수**

인수명	자료형	기본값	설명
iReqId	int	-	식별자

- **리턴**

값	설명
USER_COORDINATE	변경된 Coordinate의 ID, 참조 기준 및 위치정보

- **예제**

```
float pos[6] = {10, 20, 30, 0, 0, 0};
int id = Drfl.set_user_cart_coord(0, pos);
USER_COORDINATE *temp = Drfl.get_user_cart_coord(id);
```

3.10.15 CDRFLEx.set_desired_force

- **기능**

전역으로 설정된 좌표계(set_ref_coord 참조) 기준으로 힘 제어 목표값, 힘 제어 방향, 힘 목표값, 외력참조모드를 설정한다.

- **인수**

인수명	자료형	기본값	설명
fTargetForce	float[6]	-	3개의 Translational 힘 성분 3개의 Rotational 모멘트 성분
iTargetDirection	unsigned char[6]	-	1이면 해당 방향 힘 제어 0이면 해당 방향 compliance 제어
eForceReference	COORDINATE_SYSTEM	COORDINATE_SYSTEM_TOOL	상수 및 열거형 정의 참조
fTargetTime	float	0	힘을 증가시키는 데 소요되는 시간[sec] 범위 : 0~1.0
eForceMode	FORCE_MODE	FORCE_MODE_ABSOLUTE	상수 및 열거형 정의 참조

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
Drfl.set_ref_coord(COORDINATE_SYSTEM_TOOL);
float x0[6] = {0, 0, 90, 0, 90, 0};
Drfl.movej(x0, 60, 30);
```

```
float stx[6] = {500, 500, 500, 100, 100, 100};
Drfl.task_compliance_ctrl(stx);
float fd[6] = {0, 0, 0, 0, 0, 10};
unsigned char fctrl_dir[6] = {0, 0 ,1, 0, 0, 1};
Drfl.set_desired_force(fd, fctrl_dir);
```

3.10.16 CDRFLEx.release_force

- **기능**

  힘 제어 목표값을 time 값 동안 0으로 줄이고 작업 공간을 순응 제어로 반환한다.

- **인수**

인수명	자료형	기본값	설명
fTargetTime	float	0	힘을 감소시키는데 소요되는 시간 범위 0~1.0

- **리턴**

값	설명
0	오류
1	성공

- **예제**

```
Drfl.set_ref_coord(COORDINATE_SYSTEM_TOOL);
float x0[6] = {0, 0, 90, 0, 90, 0};
Drfl.movej(x0, 60, 30);
float stx[6] = {500, 500, 500, 100, 100, 100};
Drfl.task_compliance_ctrl(stx);
float fd[6] = {0, 0, 0, 0, 0, 10};
unsigned char fctrl_dir[6] = {0, 0 ,1, 0, 0, 1};
Drfl.set_desired_force(fd, fctrl_dir);
float x1[6] = {0, 500, 700, 0, 180, 0};
float velx[2] = {60, 60};
float accx[2] = {30, 30};
Drfl.MoveL(x1, velx, accx);
Drfl.release_force(0.5);
Drfl.release_compliance_ctrl();
```

3.10.17 CDRFLEx.check_position_condition_abs

- **기능**

주어진 위치 상태를 확인한다. while 혹은 if 조건과 함께 사용하여 해당 조건을 반복 확인할 수 있다. fTargetPos는 입력좌표계(eTargetRef) 기준의 축 방향 및 포즈를 의미한다.

입력 좌표계가(eTargetRef)가 COORDINATE_SYSTEM_TOOL인 경우 입력 위치(fTargetPos)는 BASE 좌표계 기준의 값을 입력해야 한다.

➔ 상대 이동 기준은 check_position_condition_rel 함수를 참조

- **인수**

인수명	자료형	기본값	설명
eForceAxis	FORCE_AXIS	-	축 방향
fTargetMin	float	-	최솟값
fTargetMax	float	-	최댓값
eForceReference	COORDINATE_SYSTEM	COORDINATE_SYSTEM_TOOL	상수 및 열거형 정의 참조

- **리턴**

값	설명
1	조건이 참
0	조건이 거짓

- **예제**

```
bool CON1 = Drfl.check_position_condition_abs(FORCE_AXIS_X, -5, 0,
COORDINATE_SYSTEM_WORLD);
bool CON2 = Drfl.check_position_condition_abs(FORCE_AXIS_Y, -10000, 700);
bool CON3 = Drfl.check_position_condition_abs(FORCE_AXIS_Z, -10, -5);
bool CON4 = Drfl.check_position_condition_abs(FORCE_AXIS_Z, 30, -10000);
bool CON5 = Drfl.check_position_condition_abs(FORCE_AXIS_Z, -10, -5,
COORDINATE_SYSTEM_BASE);
bool CON6 = Drfl.check_position_condition_abs(FORCE_AXIS_Z, -10, -5);
```

3.10.18 CDRFLEx.check_position_condition_rel

- **기능**

주어진 위치 상태를 확인한다. while 혹은 if 조건과 함께 사용하여 해당 조건을 반복 확인할 수 있다. eTaskAxis, fTargetPos는 입력좌표계(eTargetRef) 기준의 축 방향 및 포즈를 의미한다.
입력 좌표계가(eTargetRef)가 COORDINATE_SYSTEM_TOOL인 경우 입력 위치(fTargetPos)는 BASE 좌표계 기준의 값을 입력해야 한다.
➔ 절대 이동 기준은 check_position_condition_abs 함수를 참조

- **인수**

인수명	자료형	기본값	설명
eForceAxis	FORCE_AXIS	-	축 방향
fTargetMin	float	-	최솟값
fTargetMax	float	-	최댓값
fTargetPos	float[6]	-	6개의 Task Space 정보
eForceReference	COORDINATE_SYSTEM	COORDINATE_SYSTEM_TOOL	상수 및 열거형 정의 참조

- **리턴**

값	설명
1	조건이 참
0	조건이 거짓

- **예제**

```
float posx1[6] = {400, 500, 800, 0, 180, 0};
bool CON7 = Drfl.check_position_condition_rel(FORCE_AXIS_Z, -10, -5, posx1,
COORDINATE_SYSTEM_TOOL);
```

3.10.19 CDRFLEx.check_position_condition

주어진 위치 상태를 확인한다. while 혹은 if 조건과 함께 사용하여 해당 조건을 반복 확인할 수 있다. eTaskAxis, fTargetPos는 입력좌표계(eTargetRef) 기준의 축 방향 및 포즈를 의미한다.
입력 좌표계가(eTargetRef)가 COORDINATE_SYSTEM_TOOL인 경우 입력 위치(fTargetPos)는 BASE 좌표계 기준의 값을 입력해야 한다.
eMode가 MOVE_MODE_RELATIVE일 경우 check_position_condition_rel을, MOVE_MODE_ABSOLUTE일 경우 check_position_condition_abs를 호출한다.

- **인수**

인수명	자료형	기본값	설명
eForceAxis	FORCE_AXIS	-	축 방향
fTargetMin	float	-	최솟값
fTargetMax	float	-	최댓값
fTargetPos	float[6]	-	6개의 Task Space 정보
eMode	MOVE_MODE	MOVE_MODE_ABSOLUTE	상수 및 열거형 정의 참조
eForceReference	COORDINATE_SYSTEM	COORDINATE_SYSTEM_TOOL	상수 및 열거형 정의 참조

- **리턴**

값	설명
1	조건이 참
0	조건이 거짓

- **예제**

```
float posx1[6] = {400, 500, 800, 0, 180, 0};
bool CON7 = Drfl.check_position_condition(FORCE_AXIS_Z, -10, -5, posx1,
MOVE_MODE_ABSOLUTE);
```

3.10.20 CDRFLEx.check_force_condition

- **기능**

주어진 힘 상태를 확인한다. 단, 힘의 방향은 고려하지 않고 크기로만 비교한다. while 또는 if 조건과 함께 사용하여 해당 조건을 반복 확인할 수 있다. 힘 측정 시 eForceAxis는 입력 좌표계(eTargetRef) 기준의 축 방향이고 모멘트 측정 시 eForceAxis는 툴 좌표계 기준의 축 방향이다.

- **인수**

인수명	자료형	기본값	설명
eForceAxis	FORCE_AXIS	-	상수 및 열거형 정의 참조
fTargetMin	float	-	최솟값
fTargetMax	float	-	최댓값
eForceReference	COORDINATE_SYSTEM	COORDINATE_SYSTEM_TOOL	상수 및 열거형 정의 참조

- **리턴**

값	설명
1	조건이 참
0	조건이 거짓

- **예제**

```
bool fcon1 = Drfl.check_force_condition(FORCE_AXIS_Z, 5, 10,
COORDINATE_SYSTEM_WORLD);
bool fcon2;
bool pcon1;
while(true){
    fcon2 = Drfl.check_force_condition(FORCE_AXIS_C, 30, -10000);
    pcon1 = Drfl.check_position_condition_abs(FORCE_AXIS_X, 0, 0.1);
    if(fcon2 && pcon1)
    {
        break;
    }
}
```

3.10.21 CDRFLEx.check_orientation_condition

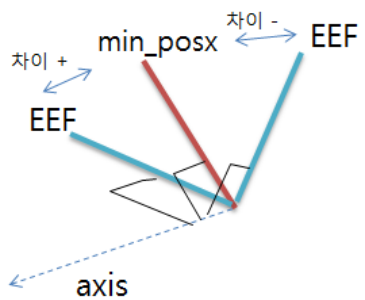
- **기능**

현재 로봇 엔드이펙터의 자세 정보와 주어진 위치 자세 간 차이의 상태를 확인한다. 현재 자세와 주어진 자세 간의 차이는 알고리즘 내부에서 회전행렬로 변환되어 “AngleAxis” 기법으로 차이 값(rad 단위)을 리턴한다. 차이가 + 값이면 true를, - 값이면 false를 리턴한다. 현재 자세를 기준으로, 주어진 position보다 차이가 +인지 –인지 확인할 때 사용한다. 사용 예로, 직접교시 position을 이용하여 현재 위치와 차이가 + 방향인지, - 방향인지를 판단한 후 orientation limit에 대한 조건을 만들 수 있다. 또한 while 또는 if 조건과 함께 사용하여 해당 조건을 반복 확인할 수 있다.

• Min만 설정 시: 차이가 +이면 True, -이면 False

• Min, Max 설정 시: min과의 차이가 -이고, max 차이가 +이면 True, 반대면 False

• Max만 설정 시: max 차이가 -이면 True, +이면 False



- **인수**

인수명	자료형	기본값	설명
eForceAxis	FORCE_AXIS	-	축 방향
fTargetMin	float[6]	-	6개의 최소값 정보
fTargetMax	float[6]	-	6개의 최대값 정보
eForceReference	COORDINATE_SYSTEM	COORDINATE_SYSTEM_TOOL	상수 및 열거형 정의 참조

- **리턴**

값	설명
1	조건이 참
0	조건이 거짓

- **예제**

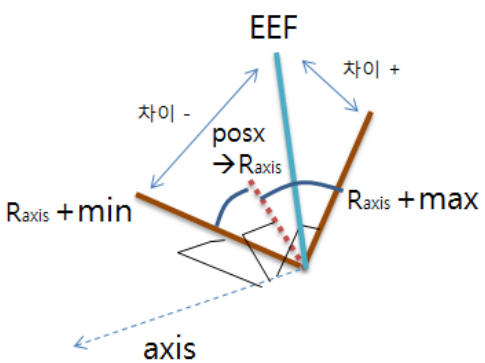
```
float posx1[6] = {400, 500, 800, 0, 180, 30};
float posx2[6] = {400, 500, 500, 0, 180, 60};
bool con1 = Drfl.check_orientation_condition(FORCE_AXIS_C, posx1, posx2);
```

3.10.22 CDRFLEx.check_orientation_condition

- **기능**

현재 로봇 엔드이펙터의 자세 정보와 회전각 범위 차이에 대한 상태를 확인한다. 현재 자세와 회전각 범위에 대한 차이는 알고리즘 내부에서 회전행렬로 변환되어 “AngleAxis” 기법으로 차이 값(rad 단위)을 리턴한다. 차이가 + 값이면 true를, -값이면 false를 리턴한다. 현재 자세를 기준으로, 주어진 position과 회전각 범위 차이가 +인지 –인지 확인할 때 사용한다. 사용 예로, 어떤 기준이 되는 position에서 min, max로 회전각 범위를 설정하여, 현재 위치와 차이가 + 방향인지, - 방향인지 판단한 후 orientation limit에 대한 조건을 만들 수 있다. 또한 while 또는 if 조건과 함께 사용하여 해당 조건을 반복 확인할 수 있다.

- Min만 설정 시: 차이가 +이면 True, -이면 False
- Min, Max 설정 시: min과의 차이가 -이고, max 차이가 +이면 True, 그 반대이면 False
- Max만 설정 시: max 차이가 -이면 True, +이면 False



알아두기

회전각 범위: 주어진 position에서 설정된 axis를 기준으로, 상대적인 각도 범위(min, max)를 말한다. 인자 ref에 따라 주어진 position의 기준 좌표계가 정해진다.

- **인수**

인수명	자료형	기본값	설명
eForceAxis	FORCE_AXIS	-	축 방향
fTargetMin	float	-	최소값
fTargetMax	float	-	최대값

인수명	자료형	기본값	설명
fTargetPos	float[6]	-	6개의 Task Space 정보
eForceReference	COORDINATE_SYSTEM	COORDINATE_SYSTEM_TOOL	상수 및 열거형 정의 참조

- **리턴**

값	설명
1	조건이 참
0	조건이 거짓

- **예제**

```
float posx1[6] = {400, 500, 800, 0, 180, 30};
bool con1 = Drfl.check_orientation_condition(FORCE_AXIS_C, 0, 5, posx1);
```

3.10.23 CDRFLEx.coord_transform

- **기능**

'eInCoordSystem' 기준 좌표계에서 표현되는 'fTargetPos' Task 좌표를 'eOutCoordSystem' 기준 좌표계에서 표현되는 Task 좌표로 변환하여, 출력한다. 아래의 경우에 대한 좌표변환 계산을 지원한다.

- (eInCoordSystem) 월드 기준 좌표계 → (eOutCoordSystem) 월드 기준 좌표계
- (eInCoordSystem) 월드 기준 좌표계 → (eOutCoordSystem) 베이스 기준 좌표계
- (eInCoordSystem) 월드 기준 좌표계 → (eOutCoordSystem) 툴 기준 좌표계
- (eInCoordSystem) 월드 기준 좌표계 → (eOutCoordSystem) 사용자 기준 좌표계
- (eInCoordSystem) 베이스 기준 좌표계 → (eOutCoordSystem) 월드 기준 좌표계
- (eInCoordSystem) 베이스 기준 좌표계 → (eOutCoordSystem) 베이스 기준 좌표계
- (eInCoordSystem) 베이스 기준 좌표계 → (eOutCoordSystem) 툴 기준 좌표계
- (eInCoordSystem) 베이스 기준 좌표계 → (eOutCoordSystem) 사용자 기준 좌표계
- (eInCoordSystem) 툴 기준 좌표계 → (eOutCoordSystem) 월드 기준 좌표계
- (eInCoordSystem) 툴 기준 좌표계 → (eOutCoordSystem) 베이스 기준 좌표계
- (eInCoordSystem) 툴 기준 좌표계 → (eOutCoordSystem) 툴 기준 좌표계
- (eInCoordSystem) 툴 기준 좌표계 → (eOutCoordSystem) 사용자 기준 좌표계
- (eInCoordSystem) 사용자 기준 좌표계 → (eOutCoordSystem) 월드 기준 좌표계
- (eInCoordSystem) 사용자 기준 좌표계 → (eOutCoordSystem) 베이스 기준 좌표계
- (eInCoordSystem) 사용자 기준 좌표계 → (eOutCoordSystem) 툴 기준 좌표계
- (eInCoordSystem) 사용자 기준 좌표계 → (eOutCoordSystem) 사용자 기준 좌표계

- **인수**

인수명	자료형	기본값	설명
fTargetpos	float[6]	-	6개의 Task Space 정보
eInCoordSystem	float	-	최솟값
eOutCoordSystem	float	-	최댓값

- **리턴**

값	설명
float[6]	6개의 Task Space 정보

- **예제**

```
float base_pos[6] = {400, 500, 800, 0, 180, 15};
ROBOT_POSE* tool_pos;
tool_pos = Drfl.coord_transform(base_pos, COORDINATE_SYSTEM_BASE,
COORDINATE_SYSTEM_TOOL);
```

API

Doosan Robotics

www.doosanrobotics.com