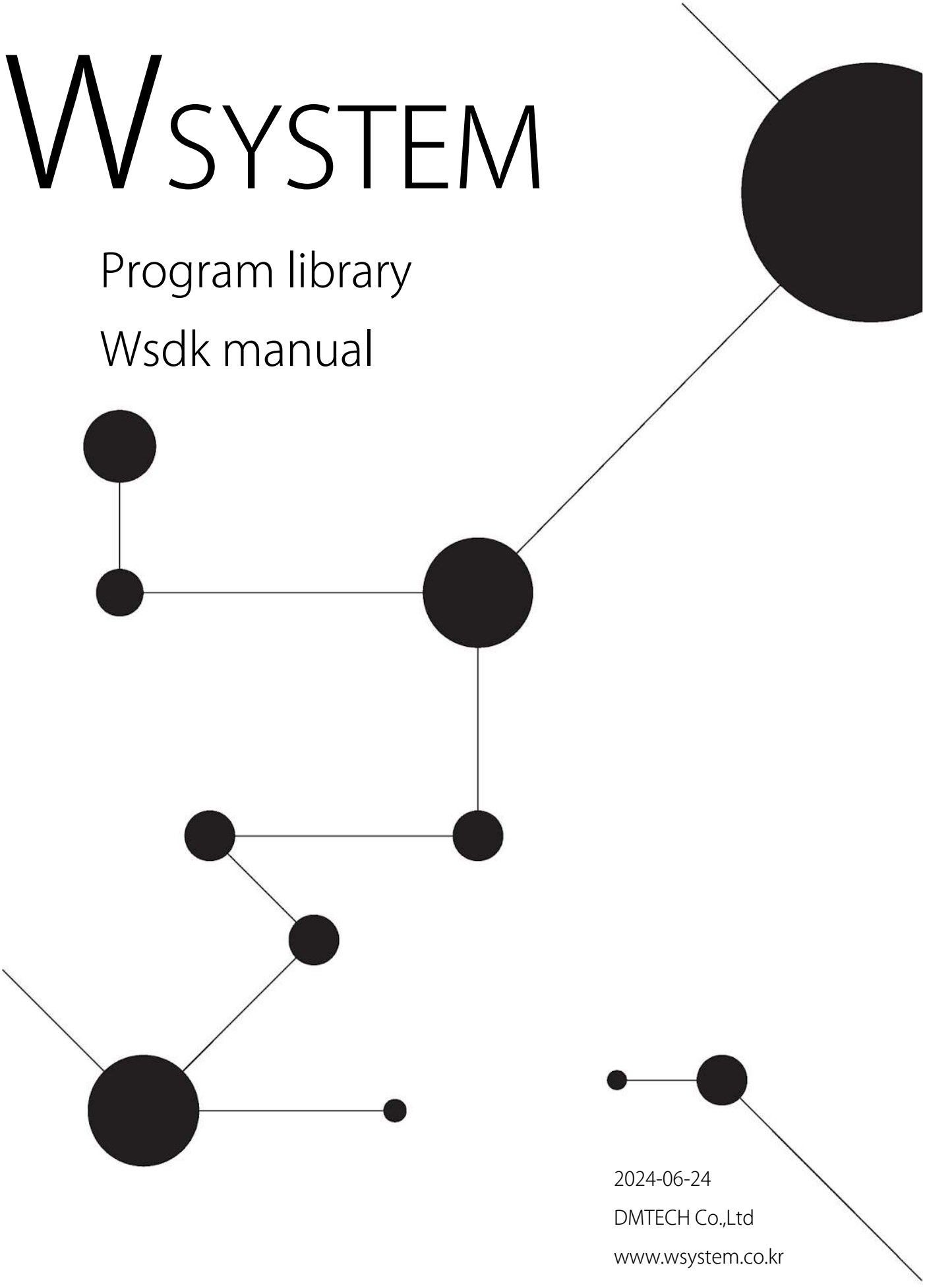
WSYSTEM

Program library

Wsdk manual



2024-06-24

DMTECH Co.,Ltd

www.wsystem.co.kr

목 차

1. PC 와 Wsystem network : Wnet 의 연결

USB 케이블(▶ USB to Micro B 5pin, 데이터 통신용)로 Wsystem 커플러인 TC720 과 PC 를 연결합니다.
드라이버는 Window10/11 에서 인터넷 연결시 자동으로 설치됩니다.
다운로드가 필요한 경우는 **http://sse2000.com/_/TC720_2.12.36.4.zip**

USB 케이블 연결이 정상적인 경우 TC720 의 적색 LED(PWR ISOLATED)가 ON 이며,
TC720 의 인식이 정상적인 경우 녹색 LED(USB EMULATED)가 ON 입니다. 따라서 정상적이면 둘다 ON 입니다.

장치관리자에서 시리얼포트번호를 확인합니다.
USB Serial Port 의 Properties -> Port Settings -> Advanced 에서 COM Port Number (▶ COM3 사용 권장)

2. Wsdk 라이브러리와 연결

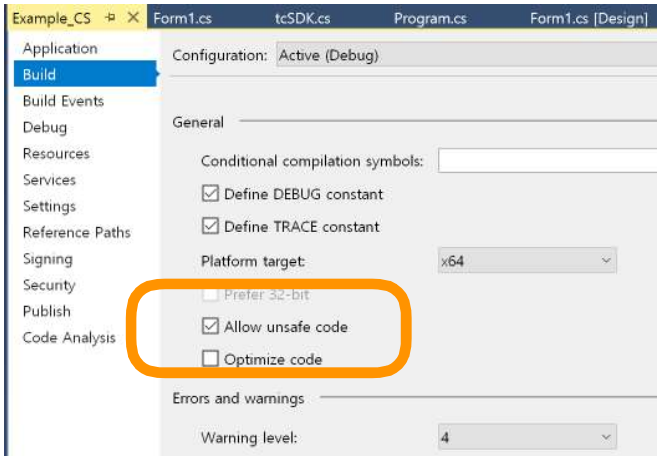
최신 라이브러리 파일을 다운로드합니다. http://sse2000.com/_/Wsdk.zip

헤더 파일 소스 파일	C++ (32/64 비트)	Wsdkdef_YYYYMMDD.h : Wsdk 의 상수 및 함수가 정의된 헤더 파일 Wsdk_YYYYMMDD.cpp
	C#	Wsdk_x64_YYYYMMDD.cs(64 비트) 또는 Wsdk_x86_YYYYMMDD.cs(32 비트)
	C	Wsdk_YYYYMMDD.c
DLL 파일	Wsdk_x64_YYYYMMDD.dll (64 비트 윈도우 10, 11) Wsdk_x86_YYYYMMDD.dll (32 비트 윈도우 10, 11) DLL 파일은 프로그램 Working Directory 에 다른 라이브러리와 함께 놓습니다.	
바이너리 파일	Wxxx_YYYYMMDD.bin (릴리즈된 Wdevice 펌웨어) 예) W520_YYYYMMDD.bin (W520 의 펌웨어), W430_YYYYMMDD.bin (W430 의 펌웨어)	

파일명의 YYYYMMDD 는 연월일로 표시되는 Wsdk 버전이며, 각 파일의 버전은 모두 동일해야 합니다.
프로그램 어플리케이션이 32 비트 또는 64 비트인지에 따라 x86, x64 파일을 구분해서 사용합니다.
이는 컴퓨터의 32/64 비트 구분이 아니라 프로그래밍 프로젝트 속성에서 컴파일 타켓이 32/64 비트용인지에 따릅니다.

▶ C++ 에서 라이브러리를 이용하는 경우 :

1) 제작할 애플리케이션 프로젝트에 Wsdk_YYYYMMDD.cpp 를 추가합니다.
2) 라이브러리를 구현할 소스 파일에다 #include "Wsdkdef_YYYYMMDD.h"를 선언합니다.
   Wsdkdef_x.h 에서는 Wsystem 프로그래밍을 위한 함수만이 아니라 편의를 위한 상수들도 정의되어 있습니다.
   이를 적절히 이용, 상위/하위 버전과 호환성을 유지할 수 있습니다.
3) Visual Studio 를 사용시에 다음 구성요소 설치를 권장합니다.
   - C++를 사용한 데스크톱 개발
   - 최신 vxxx 빌드 도구용 C++ MFC



▶ C#에서 라이브러리를 이용하는 경우 :

제작할 애플리케이션의 컴파일 타겟에 따라 Wsdk_x86_x.cs
또는 Wsdk_x64_x.cs 를 추가합니다.
함수들 중에 call by reference 방식으로 파라미터를 넘겨주어
결과값을 반환 받는 경우가 있기 때문에
unsafe 키워드를 허가해 주어야 하므로
프로젝트 속성에서 'Allow unsafe code'를 체크해 줍니다.

3. Log file 기능(Log file 이름 : W_yyyymmdd_hhmmss.log)

명령과 디바이스 상태가 실시간으로 기록되는 Log file 기능입니다.
Log file은 사용자의 디버깅 시간을 대폭 줄여주며 제조사의 기술 지원 또한 매우 용이하도록 도와줍니다.
따라서 Log file을 띄워놓고 실시간으로 확인하는 습관을 갖는다면 웬만한 문제들은 쉽게 파악되고 빨리 해결됩니다.

Log file 확인에 익숙해지면 문제 해결이 상당히 쉬워지므로 Wnet과 첫 연결시에 한번 Log file을 띄우게 되어있습니다.
Log file의 연결 프로그램으로 메모장 보다는 Visual Code를 사용하는게 좋으며, 코딩할 때에 Log file을 띄워놓고
클릭할 때마다 Log file이 갱신되어서 주고받은 데이터를 바로 확인이 가능하기 때문에 디버깅이 편합니다.
nowSet(0, W_IDX_LOG_LAUNCH, W_LOG_LAUNCH_DISABLE); 명령으로 Wnet 연결시에도 Log file를 띄우지 않게 되며,
nowSet(0, W_IDX_LOG_LAUNCH, W_LOG_LAUNCH_NOW); 명령을 하면 Log file을 즉시 화면에 띄웁니다.

설정 함수 : nowSet(0, W_IDX_LOG_LEVEL, W_LOG_LEVEL_x) 아래 5단계중 하나로 설정함

W_LOG_LEVEL_WSDK	프로그램의 시작과 끝 그리고 전역 알람만 기록합니다.
W_LOG_LEVEL_CMD_EXCEPT_DO	시작과 끝, 주요 명령, 전역 알람을 디지털 입출력은 제외하고 기록합니다.
W_LOG_LEVEL_CMD (기본 설정)	시작과 끝, 주요 명령, 전역 알람을 기록합니다.
W_LOG_LEVEL_CMD_AND_GUIDE	시작과 끝, 주요 명령, 전역 알람과 상태 에러 메시지도 함께 기록합니다.
W_LOG_LEVEL_ALL	모든 명령들을 전부 기록합니다. 예) nowState(axis) 명령도 기록합니다

※ 지정된 Log file 레벨에 따라서 Log 문구와 대상 함수의 종류가 다를 수 있습니다.
※ Log file이 지나치게 커지는걸 방지하기위해 상위 설정은 1시간이 지나면 자동으로 W_LOG_LEVEL_CMD으로 돌아갑니다.
하지만 장비 개발이 완료되고 장비가 출하될 때는 W_LOG_LEVEL_WSDK로 설정을 권장합니다.

Wnet과 연결되면 Log file은 매번 자동으로 생성하며 프로그램 수행 도중 이벤트 발생 즉시 파일에 저장합니다.
C:₩또는 D:₩, E:₩ 중에 "W_LOG" 폴더가 있다면 순서대로 검색하여 그 폴더에 Log file을 저장하며.
W_LOG 폴더가 없는 경우는 프로그램 Working Directory에 "W_년월일_시분초.log"라는 파일명으로 저장됩니다.
이 Log file은 텍스트 파일로 일반 에디터를 이용하여 열람할 수 있지만 Visual Code 사용을 추천합니다..

예1) 0000015.01 0x 8C0012 listMove(9, W_DIR_N, 10.500000*W_PPR) // Log file에 MOVE 명령 표시
(1) (2) (3)

(1) 7.2자리 숫자는 15.01초로 DLL의 로드 시간부터 초단위의 스탬프입니다.
하지만 MS윈도우 자체가 리얼타임이 아니기 때문에 TickCount는 절대적인 시간이 아닙니다.
시간이 같다고 동시에 실행된 것이 아니며 숫자의 차이만큼 아주 정확한 시간이 차이 나는 것은 아닙니다.
▶ 윈도우 자체가 실시간이 아닌 문제점에도 Wsytem을 사용하면 실시간으로 정확하고 빠른 제어가 가능합니다.

(2) 0x 8C0012 : 명령에 상태의 반환값 (에러값 xx + 통신 패킷수 xx + 사용자의 설정 숫자 xx + 리스트 총숫자 xx)
사용자가 보기 편하게하기 위해서 에러값이 00이면 표시하지 않습니다. (4. W_ERR 반환에서 상세 설명)

(3) listMove(9, W_DIR_N, 10.500000*W_PPR) : 함수명과 전달한 인자값입니다.
기본 설정이 1백만에 1회전이기 때문에 편의를 위해서 W_PPR(1,000,000)을 곱해서 10.5회전의 전달 값을 보여줍니다.

예2) 0000001.09 $ Wnet 8 Devices, 23 Axes Scaned
설명 : Wnet 연결시에 8개의 Wdevice와 총 23축이 스캔됨.
시간 스탬프 뒤에 $로 시작하는 문장은 Wsdk가 직접 표시하는 Log 문구입니다.

4. W_ERR 반환

Wsdk의 모든 함수는 실행 시에 4바이트의 unsigend int W_ERR[31 bit~0 bit]을 반환합니다.

1) 0x □□xxxxxx [31 bit ~ 24 bit] : 함수 호출 오류나 임시 에러 또는 전역 알람 상태

Wsdk 전체에 적용되는 전역 에러 값은 해당 에러가 해제되기 전까지는 다른 축이나 다른 함수를 호출하여도 공통으로 동일하게 나타납니다. 물론 전역 에러 발생 도중에 또 다른 전역 에러가 발생하면 에러 값은 변경될 수 있습니다. 이 때의 에러 내용을 모두 놓치지 않고 모니터링하려는 경우에는 Log 파일을 참조하십시오.
기본적으로 명령을 보낼 때마다 항상 W_ERR 값을 확인하시기 바랍니다.
▶ Log file에서는 에러값을 쉽게 보기위해 에러가 없는 00의 경우는 빈칸 두개로 표시됩니다.

2) 0x xx□□xxxx [23 bit ~ 16 bit] : Wsystem 디바이스들이 보내 온 모니터링 패킷 총숫자의 하위 8비트

사용자가 필요한 중요 정보는 아닙니다. nowGet(axis, W_IDX_FEED_NUM, &value)으로 32비트 전체값 확인도 가능합니다.

3) 0x xxxx□□xx [15 bit ~ 7 bit] : Wdevice 내부에서 사용자가 사용하는 UserValue 값의 하위 8비트

nowGet(axis, W_IDX_USER_VALUE, &value)로 32비트 전체값의 확인 가능합니다.
UserValue를 이용하여 필요한 값을 Log file에 남겨 유용하게 사용할 수도 있고, 다양한 인덱스로 사용할 수도 있습니다.
특히 listAction() 기능 사용시에는 유용하게 사용됩니다.

4) 0x xxxxxx□□ [7 bit ~ 0 bit] : LIST_NUM, 각축에 쌓인 Wait-Turn 함수들의 총숫자

Wait-Turn 함수는 호출된 순서대로 축의 LIST 대기열에 예약되어 차례대로 실행됩니다.(최대 256개)
LIST된 명령은 하드웨어적으로 딜레이 없이 순서대로 수행됨으로써 분산 제어의 최대 효율을 끌어 냅니다.
사용자는 축 별로 LIST되어있는 총숫자를 모니터링 함으로써 명령의 진행 상황과 완료를 확인할 수 있습니다.

처음 사용자의 경우에 각 명령 함수 호출 시에 명령이 완료되어야 반환하는 라이브러리가 사용하기 편할 수는 있으나 이는 전반적으로 시스템의 성능을 저하시키고 애플리케이션이 확장될수록 한계에 부딪히게 됩니다.
또한, 함수 호출 즉시 반환 된 후에 명령 종료시까지의 시간을 경험적으로 반영한 Delay를 이용한 제어 구조 역시 애플리케이션이 확장될수록 한계에 부딪히게 됩니다.
Thread를 이용한 명령/모니터링 루틴의 분리와 적절한 구간에 LIST 총숫자에 따른 대기상태를 구현한다면 시스템 성능도 유지하고 Wsystem의 분산 독립 제어의 장점을 유지할 수 있습니다.

예) listMove(100, ...); 함수에 반환되는 wErr값이 0x0B51002E 일때
0x0B(범위를 벗어난 축을 지정 에러) + 0x51(모니터링된 패킷수) + 0x00(UserValue 값) + 0x2E (리스트에 쌓인 총숫자는 46개)
Log file에 기록된 에러 메시지 : 0000008.62 0x0B51002E listMove(100, 1, 10000000) W_INVALID_AXIS

5. 에러 코드와 에러 내용

코드	에러 내용
상태 알림 : 에러값을 반환하지만 알람음을 내며 전체가 정지하는 일 없이 그 명령만 실행되지 않습니다.	
0x00	W_ERR_NONE : 에러 없음 (Log file 에는 편의를 위해서 에러가 없는 0x00 을 0x 빈칸 두개로 표시)
0xFF	W_ERR : 에러 목록 BitMask
0x01	W_NO_CONNECTED_WSDK : dll 라이브러리 로드 실패. 버전, dll 파일 경로 확인
0x02	W_NO_CONNECTED_COMM : 통신 연결 실패. USB / Modbus COM 포트 번호 확인
0x03	W_NO_CONNECTED_DEVICE : 디바이스 검색 실패. 전원, 통신연결, 디바이스 상태 확인
0x07	W_NEED_UPDATE : 펌웨어 업데이트 필요
0x08	W_NOT_FOUND_FIRMWARE : 펌웨어 파일 없음. bin 파일 경로 확인, 버전 확인, CategoryID 확인
0x09	W_INVALID_VERSION : 라이브러리 버전 오류. 개발사에 버전 및 CategoryID 확인
0x0A	W_INVALID_CALL : 함수 호출 에러. now/list, set/get 확인
0x0B	W_INVALID_AXIS : 축 지정 에러. 연결된 축 범위내에서 호출. 잘못된 ManualID 지정
0x0C	W_INVALID_PARAMETER : 함수 인자 조건 에러. 인자 조건 및 범위 확인
0x0D	W_INVALID_FUNCTION : 아직 구현되지 않은 기능이거나, 이미 제거된 기능)
0x0E	W_INVALID_PERMISSION : 권한 에러. 인증 파라미터 확인, 개발사에 문의
0x0F	W_INVALID_VIP : VIP 코드 관련 에러. 개발사와 협의
0x31	W_LIST_EMPTY : 리스트 없음
0x32	W_LIST_FULL : 축의 리스트가 꽉 참
0x33	W_SEED_BUSY : 명령 대기 버퍼가 중첩 됨. 사용자의 Thread 내 명령 함수 중첩 호출 확인
0x34	W_FEED_BUSY : 디바이스의 반환 요청이 이미 Wnet 버스를 독점
0x35	W_FEED_WAITING : 요청한 값에 대해 디바이스의 반환 대기
0x36	W_FEED_TIMEOVER : 요청한 값에 대해 디바이스의 반환이 지정 시간을 초과
0x37	W_HEED_BUSY : 디바이스 간 통신이 Wnet 버스를 독점
전역 알람 : 에러 메시지 반환되고 알람음을 계속 내며 정지해 있습니다.	
0x41	W_ALARM_NEED_CLEAR : 디바이스의 알람 상태 제거 필요
0x42	W_ALARM_ADC_ERROR : ADC 에러. 하드웨어 AS 요청
0x43	W_ALARM_ENC_ERROR : ENC 에러. 엔코더 불량, 모터 설정, 기구부 부하, 브레이크 설정 확인
0x44	W_ALARM_LOW_VOLTAGE : 구동 전압 에러. 모터 불량, 구동 전원 인가, 파워서플라이 용량 확인
0x45	W_ALARM_GAP_LIMIT : GAP 에러. 모터 불량, 모터 미연결. 척 상태 점검
0x46	W_ALARM_PHASE_BALANCE : 상 에러. 모터 불량, 하드웨어 AS 요청
0x47	W_ALARM_SENSITIVE_COIL : 코일 에러. 모터 불량, 디바이스 예열 필요
0x48	W_ALARM_FAULT_ORG : 원점 복귀 중 폴트. SERVO ON, ELN, ELP, ORG 설정 혹은 센서 감지 범위 확인
0x49	W_ALARM_FAULT_CNT : 카운트에 의한 폴트. 기구부 부하, 모터 토크, 전류 설정 등을 확인
0x4A	W_ALARM_FAULT_DIST : 거리값에 의한 폴트. 기구부 부하, 모터 토크, 전류 설정 등을 확인
0x4B	W_ALARM_UNDERSHOOT : 언더슈트. 기구부 관성, 모터 용량, ACC/PPS 설정 확인
0x4C	W_ALARM_OVERSHOOT : 오버슈트. 기구부 관성, 모터 용량, DEC/PPS 설정 확인
0x4D	W_ALARM_TEMPERATURE : 온도 초과. 보드 내 온도, 모터 온도 등 확인
0x4E	W_ALARM_DECEL_ADJUST : 개발사에 문의
0x51	W_ALARM_WNET_BUF_FULL : Wnet 버퍼 꽉 참. 명령중복 송신확인, W_IDX_FEED_MSEC 확인, 커플러 연결 확인
0x52	W_ALARM_WNET_CHECKSUM : Wnet 통신간의 체크섬 에러 발생. 통신케이블 확인, 디바이스 전원상태 확인
0x53	W_ALARM_WNET_HEED : Wnet 통신간의 디바이스 명령에 문제 발생
0x54	W_ALARM_ACT_BUF_FULL : Action 버퍼 꽉 참. 동시 설정 가능한 Action 기능 예약 수는 최대 8 개
0x55	W_ALARM_LIST_MISMATCH : 리스트 제어루틴 관련 에러. 북마크의 위치, 조건, 태그 확인
0x56	W_ALARM_OVERFLOW : 리스트, VIP, Action 코드 수행시, 인덱스 오버플로우, 변수범위 오버플로우
0xC9	W_ERR_TIMEOUT_TX : 송신 시간 제한 초과. USB/ Wnet/ Modbus/ EtherNET/ EtherCAT 통신케이블 확인, 전원 확인, 디바이스 상태 확인
0xCA	W_ERR_TIMEOUT_RX : 수신 시간 제한 초과. USB/ Wnet/ Modbus/ EtherNET/ EtherCAT 통신케이블 확인, 전원 확인, 디바이스 상태 확인
0xCB	W_ERR_ON_TX : 윈도우 내 버스 충돌. PC 사양 검토
0xCC	W_ERR_ON_RX : Wnet 통신 에러. Wnet 통신케이블 Twisted Pair + Shield Cable 사용 필수, PC 사양 검토
0xCD	W_ERR_ON_WFP : 파일 쓰기 스트림 에러. 개발사에 문의
0xCE	W_ERR_ON_RFP : 파일 읽기 스트림 에러. 개발사에 문의
0xCF	W_ERR_UNDER_SLOTH : PC 사양 검토(PC, BIOS, Windows), 스트림 속도 제어

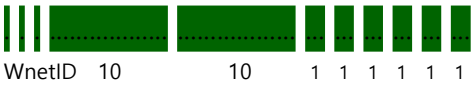
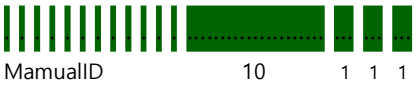
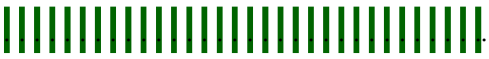
6. 알림 소리 설명

멜로디 코드 / 멜로디 묘사	멜로디 설명 / 발생 시점
W_BEEP_STOP	비프음 재생 중 강제 종료
	디바이스의 알람 상태를 알리는 알림 소리가 멈춘다고 알람이 제거되는 것은 아닙니다.
W_BEEP_SHORT	짧은 "삐" 소리. Device ID, WnetID 에 따른 음 높이
삐!	DIO 상태 변화시 이를 알립니다.
W_BEEP_LONG	긴 "삐" 소리. Device ID, WnetID 에 따라 음 높이가 높아짐
삐이~	명령 수행에 비정상적인 상태 발생시 이를 알립니다.
W_BEEP_DOUBLE	짧은 "삐" 소리 이후에 바로 긴 "삐"소리. Device ID, WnetID 에 따른 음 높이
삐!삐!	정상적인 시동시와 정상적인 설정 적용시에 울립니다.
W_BEEP_TOGGLE	삐익! 삐익! 계속 반복. Device ID, WnetID 에 따른 음 높이
삐이익~ 삐이익~	저전원 감지, 좌표 이탈, 오버플로우, 서보 알람 등 FAULT 상황에서 울립니다.
W_BEEP_AXISx	0 번~3 번 축 번호에 따라, 짧은 삐가 축 번호만큼 울린 후, 윅~하는 소리가 재생됩니다.
윅~, 삐윅~, 삐삐윅~, 삐삐삐윅~	디바이스내의 축번호를 삐소리의 갯수로 나타냅니다.
W_BEEP_INCx	축 번호에 따라 전체적인 음의 높이가 높아집니다.
상승하는 삐이익~	과부하 감지시 재생합니다.
W_BEEP_DECx	축 번호에 따라 전체적인 음의 높이가 낮아집니다.
하강하는 삐이익~	리미트 센서 감지시 재생합니다.
W_BEEP_CURVE	축 번호를 나타내는 W_BEEP_AXISx 와 조합로 재생합니다.
상승 후, 하강하는 삐이익~	이동하려는 시점에 목표 명령 위치값와 현재의 위치값이 같을 경우에 재생
W_BEEP_DOREMI	상승하는 도레미파솔라시도.
도레미파솔라시도	명령 인덱스 에러입니다. 저사양 환경, 통신선 단선, 제어 전원 불량등 확인합니다.
BEEP_STEP_ON	상승하는 도미솔~
도미솔~	엔코더를 사용하지 않는 Open 제어시에 모터를 ON 했을 경우 재생합니다.
BEEP_STEP_OFF	하강하는 솔미도~
솔미도~	정지 및 전류 차단 완료 후, 정상 STEP OFF 알립니다.
W_BEEP_FLYRUN_ON	상승하는 도도미미솔솔~
도도미미솔솔~	Flyrun 제어시에 모터를 ON 하고 Servo On 로직을 시작하면서 재생합니다.
BEEP_SERVO_ON	상승하는 도솔~
도솔~	서보앰프에 Servo On 명령을 보내면서 재생
BEEP_SERVO_ON	하강하는 솔도~
솔도~	서보 앰프에 Servo Off 명령을 보내면서 재생
BEEP_WOW	경쾌한 빠라바라바라밤
솔라시라솔라시~	Wnet 커플러 및 커플러에 연결된 디바이스의 스캔을 완료한 후에 이를 알립니다.
BEEP_ERR	하강하는 에러음.
도솔라파솔미파미레도	명령 체크썸 에러입니다. 노이즈, 통신선 단선, 노이즈, 전원 불량 등 확인합니다.
W_BEEP_ORG	'엘리제를 위하여' 멜로디 재생
미레# 미레# 미시레도라	원점 복귀 루틴이 동작 중일 때, 재생됩니다.
BEEP_BUTTERFLY	'나비야~ 나비야~ 이리 날아 오너라' 재생
솔미미~ 파레레~	Wsdk 나 Wdevice 의 펌웨어 버전에 문제나, 지정되지 않은 기능의 명령시에 재생합니다.
W_BEEP_SCHOOL	'학교종이 땡땡땡 어서모이자' 재생
솔솔라라솔솔미	Wdevice 내의 프로세스 독점 및 지연 발생시
W_BEEP_ONARA	'오나라 오나라 아주오나~' 재생
라시시 시라솔 미솔솔솔	EtherCAT 연결 성공시 재생합니다.
W_BEEP_ANIRAO	진도 아리랑 풍 재생 (송소희 LTE 광고 : "아니라오~ 아니라오~ 다되는건 아니라오~"의 의미)
시시라솔미솔	ModBus 연결 성공시 재생합니다.

7. Wdevice의 상태 표시 LED

Green LED, Red LED의 점등, 소등, 점멸로 운전 상태를 파악할 수 있습니다.

1) Green LED (WnetID와 ManualID를 계속 번갈아가며 표시, Wnet 미연결 표시, 펌웨어업데이트 표시)

상태	LED	점등 상태
Wnet과 미연결		2초 동안 매우 길게 켜졌다 꺼짐
WnetID를 표시		3번 매우 빠르게 깜박인 후에 표시되는 값 0.6초 동안 길게 켜지는 것은 10을 의미합니다. 0.1초 동안 짧게 켜지는 것은 1을 의미합니다. 예) 길게 2번 깜빡이고는 짧게 6번 깜빡인다면 이는 WnetID 26번임을 의미합니다.
ManualID를 표시		12번 매우 빠른게 깜박인 후에 표시되는 값 설정되지 않았으면 2초 동안 길게 깜박임 0.6초 동안 길게 켜지는 것은 10을 의미합니다. 0.1초 동안 짧게 켜지는 것은 1을 의미합니다. 예) 길게 1번 깜빡이고는 짧게 3번 깜빡인다면 이는 ManualID 13번임을 의미합니다. 만약, ManualID가 별도로 설정되지 않은 기본값이라면, 2초동안 매우 길게 켜집니다.
펌웨어 업데이트		업데이트중에는 계속 빠르게 깜박임 업데이트가 이뤄지지 않는다면, 통신선 연결이나 제품의 bin 파일이 있는지 등을 확인합니다.

2) Red LED (LED는 통신 패킷 송수신을 표시)

상태	LED	점등 상태
패킷이 수신되기 전		패킷이 1개라도 수신되기 전까지는 계속 켜져 있음 ※ Wnet 연결 실패시에 케이블 문제인지 확인하기 위해서 제품의 순서대로 Red LED를 확인합니다. Red LED가 계속 켜져 있는 제품이 보이면 그 제품의 수신부 케이블 점검하면 됩니다.
패킷 수신할 때		패킷 수신할 때만 10msec 동안 짧게 켜집니다. 수신된 패킷을 포워딩하던, 해당 디바이스에서 모니터링 데이터를 수신하던 간에 어떠한 패킷을 수신하게 되면, 10msec동안 짧게 켜집니다.

8. Wsdk 프로그래밍 사전 설명

1) WsdkDef_xxxxxxxx.h 헤더파일에는 상수와 함수가 정의 되어있고 간략한 설명이 있습니다.
프로그래밍 중에 필요한 간단한 내용들은 여기서 찾아보면 편합니다.

2) 제어에 사용되는 축 번호(Wnet ID)는 연결될 때의 Wdevice 연결 순서대로 자동으로 부여됩니다.
다른 ID을 별도로 설정(Manual ID, Nick ID)하고 영구 저장해서 사용하는 것도 가능합니다만 Wnet ID를 추천합니다.

3) 통신 제어가 고속이고 데이터 오류가 없기 때문에 아주 많은 축을 설정해도 순식간에 설정이 끝납니다.
따라서 시작할 때 마다 설정해 주는 것을 권장합니다. ※ 1초에 3000번 설정 가능
굳이 설정을 영구 저장해서 사용하려면 nowSet(axis,W_IDX_FLASH, ...); 함수를 사용합니다. (교환시에 문제될 소지 있음)

4) 모든 함수는 현재 상태인 W_ERR[31:0] 값을 반환합니다. 따라서, 함수 명령 마다 반환값을 모니터링 하는걸 권장합니다.
예) wErr = listSetAxis(1, W_AXIS_2S44A_057081); // 1번 축을 등록된 모터 사양으로 SERVO ON
전역 변수 wErr의 값을 모니터링 하면 문제가 생겼을 때, 즉시 알 수 있습니다.

5) nowState(axis);의 반환값은 모든 명령에서 반환되는 값과 같습니다. 또한 반환값은 설정된 Feed time 마다 돌아오는 모니터링 데이터를 메모리에서 읽는 거라서 많이 써도 통신 부담은 전혀 없습니다.
만일 listFeed(axis, ...); 함수를 사용하면, 필요한 모니터링 데이터만 즉시 업데이트 되므로, Feed time을 변경하여 모니터링 주기를 길게 하고, 그때 그때 필요한 데이터만 즉시 받아서 사용하는 방법도 있습니다.

6) 원점 복귀는 모션과 센서의 동기를 이용한 가장 빠르고 정확한 방식으로 한가지만 제공합니다.
하지만 listAction() 기능을 사용하면 상상하는 모든 방식의 원점 복귀 모드를 구성할 수 있습니다.

7) **Wsdk의 함수는 Wait-Turn(리스트 대기 실행 함수)과 Non-Wait(즉시 실행 함수)로 두 종류로 구분할 수 있습니다.**

▶ Wait-Turn Function : 리스트 대기 실행 함수
Wait-Turn 함수는 호출된 순서대로 그 축 번호의 list 대기열에 예약되어 차례대로 실행되며
최대 255개의 대기열 예약이 가능합니다.
listing된 함수 명령은 하드웨어적으로 딜레이 없이 순서대로 수행됨으로써 분산 제어의 최대 효율을 끌어냅니다.
다른 함수와 구분을 위해서 모든 Wait-Turn 함수명은 list로 시작됩니다.
nowState(axis)함수를 이용하여 list에 쌓여서 실행을 기다리는 함수의 총숫자를 모니터링 할 수 있습니다.
list 함수로 리스트의 순차적으로 수행되는 도중에 NonWait(now) 함수가 호출되면 리스트와 상관없이 즉시 실행됨으로써 의도치 않은 결과를 초래할 수 있으므로 주의가 필요합니다.

- listAction(axis, ...) 함수는 list 명령들을 제어 프로그래밍 언어처럼 사용할 수 있는 기능입니다.
이 기능은 각 축 마다 PLC가 하나씩 있는 것처럼, 다르게 말하면 각 축 마다 완전 독립적인 Thread가 동작중인 것처럼 동작이 가능하며, 이 것을 잘 이용하면 다축 제어가 손쉬워지고 딜레이 없는 궁극적 분산 제어도 실현할 수 있습니다.
또한 Mew 기능의 추가 명령어 하나면 프로그램 쓰던 것을 그대로 영구 저장하여 완전 독립형으로 사용도 가능합니다.
listAction()을 쉽게 사용하기 위해서는 몇가지 개념을 반드시 이해한 후에 사용해야 합니다만
저희가 다양한 예제를 제공하기 때문에 필요한 것을 가져다 쓰거나, 몇 번만 연습해보면 이해하게 됩니다.
자세한 설명은 9-16. listAction( ) 편을 확인해 주세요.
※ listAction(axis, ...) 복합 함수이기 때문에 list 쌓이는 숫자로 확인해 보면 1이 아니고, 함수 하나당 2씩 잡힙니다.

▶ Non-Wait Function : 즉시 실행 함수
NonWait 함수는 각 디바이스, 축의 list 대기열과는 상관없이 함수 호출 후에 디바이스에 명령이 전달 되자마자 실행되는 함수들입니다. 구분을 위해서 모든 NonWait 함수명은 now로 시작합니다.
list에 다른 함수가 대기중에도 바로 끼어들어서 실행이 되는 특성상 그 함수가 호출되는 위치와 타이밍을 정확히 이해하고 사용하는 것이 중요합니다.
- nowStop(axis); 등의 특정 함수들은 축의 list의 대기열을 모두 삭제합니다.
따라서, 프로그램 중간에 이런 NonWait 함수를 호출하는 경우에는 축의 list를 모두 삭제해도 문제없는 상태에서 호출해야 합니다.

9. Wait-Turn Function

9-1. listSetAxis(unsigned int axis, unsigned int axisType)

: 모터 설정 및 Servo(motor) ON/OFF, 브레이크 신호 출력 설정

예1) wErr = listSetAxis(1, W_AXIS_050W); // 1번 축을 아래 등록된 모터 사양으로 SERVO ON 함.
※ W_AXIS_050W = 0x08002403 (전류 2.5A, 토크 0.55Nm, 42각, 몸체 48mm+20mm, 축 5Φ, W_AXIS_2S403_042048)

예2) wErr = listSet(2, W_IDX_A_COIL_X100, 1.5 * 100); listSet(2, W_IDX_A_WORK_X100, 1.5 * 100);
wErr = listSet(2, W_IDX_A_BOOST_X100, 2 * 100); listSet(2, W_IDX_A_IDLE_X100, 0.3 * 100);
wErr = listSetAxis(2, W_AXIS_CUSTOM); // 2번 축을 사용자가 설정한 모터 사양으로 MOTOR ON 함.
CUSTOM 설정은 listSet()함수을 사용해서 Coil, Work, Boost, Idle의 4가지 전류를 먼저 설정하고 실행하면 적용됩니다.

예3) listSetAxis(9, W_AXIS_PD6) // W230(4축 펄스 제어기)의 전용 설정으로 9번축에서 드라이버로 Servo on 신호를 보냄.
※ W_AXIS_PD6은 W230의 첫번째와 두번째 축의 설정, W_AXIS_PD0은 세번째와 네번째 축의 설정입니다.

▶ W_AXIS_SERVO_OFF // listSetAxis(axis, W_AXIS_SERVO_OFF)인 경우 해당축을 Servo Off 함

▶ 표준 모델

```
W_AXIS_ETHERCAT = 0x00007B00,        // 이더캣 서보모터 제어
W_AXIS_050W     = 0x08002403,        // 2A5, 0Nm55, 042*(048+20), 5, W_AXIS_2S403_042048
W_AXIS_100W     = 0x08002404,        // 3A0, 1Nm00, 057*(051+20), 6.35, W_AXIS_2S404_057051
W_AXIS_200W     = 0x08002405,        // 4A0, 2Nm00, 057*(076+20), 6.35, W_AXIS_2S405_057076
W_AXIS_400W     = 0x08002406,        // 4A0, 3Nm00, 057*(112+20), 8, W_AXIS_2S406_057112
W_AXIS_750W     = 0x08002407,        // 5A6, 11Nm0, 086*(150+20), 14, W_AXIS_2S407_086150
W_AXIS_1k5W     = 0x08002408,        // 6A5, 21Nm0, 110*(155+30), 16, W_AXIS_2S408_110155
W_AXIS_200P     = 0x08003409,        // 5A7, 0Nm64, 060*(075+40), 14, PMSM, W_AXIS_3P409_060115
```

▶ 커스텀 모델 - 양단축, 중공축

```
W_AXIS_2S041_028051 = 0x00002041,    // 1A0, 0Nm12, 028*(051), 5
W_AXIS_2S042_035053 = 0x00002042,    // 2A0, 0Nm31, 035*(053), 5
W_AXIS_2S001_042048 = 0x00002001,    // 2A5, 0Nm55, 042*(048), 5
W_AXIS_2S043_042060 = 0x00002043,    // 2A0, 0Nm72, 042*(060), 12/9
W_AXIS_2S002_057076 = 0x00002002,    // 4A0, 2Nm00, 057*(076), 6.35
W_AXIS_2S045_057101 = 0x00002045,    // 5A0, 3Nm00, 057*(101), 15/12
```

▶ 커스텀 모델 - 코일조정, 브레이크, VCM, 3상 스텝서보, BLDC

```
W_AXIS_2S547_028051 = 0x08002547,    // 1A0, 0Nm12, 028*(051+12), 8000CPR, 5
W_AXIS_2S548_035053 = 0x08002548,    // 2A0, 0Nm31, 035*(053+12), 8000CPR, 5
W_AXIS_2S649_042048 = 0x08002649,    // 2A0, 0Nm48, 042*(048+19), 10000CPR, 5
W_AXIS_2S646_042060 = 0x08002646,    // 2A0, 0Nm72, 042*(060+19), 10000CPR, 8/6
W_AXIS_2S44A_057081 = 0x0800244A,    // 4A0, 2Nm20, 057*(081+22), 4000CPR, 6.35
W_AXIS_2S64B_057081 = 0x0800264B,    // 5A0, 2Nm20, 057*(081+22), 10000CPR, 6.35
W_AXIS_2S44C_057081 = 0x4800244C,    // 4A0, 2Nm20, 057*(081+22+39), 4000CPR, BRAKE, 6.35
W_AXIS_2S64D_057101 = 0x4800264D,    // 5A0, 3Nm00, 057*(101+22+39), 10000CPR, BRAKE, 12
W_AXIS_1V751_024050 = 0x08001751,    // 4A5, 19N, VCM
W_AXIS_1VB52_040080 = 0x08001B52,    // 3A4, 30N, VCM
W_AXIS_3S70A_110155 = 0x0800470A,    // 6A4, 16.0Nm, 110각*(155+30)mm, 3상 스텝서보, 19
W_AXIS_3B053_024136 = 0x00005053,    // 60,000rpm BLDC
```

▶ 기타 옵션

```
W_AXIS_CUSTOM    = 0x8000F000,
W_AXIS_BRAKE_DO4 = 0x50000000,
W_AXIS_BRAKE_DO3 = 0x40000000,
W_AXIS_ENC_FLIP  = 0x10000000,
W_AXIS_FLYRUN    = 0x08000000,
W_AXIS_FAULT     = 0x04000000,
W_AXIS_PUSH      = 0x02000000,
W_AXIS_WRENCH    = 0x01000000,
W_AXIS_DIO       = 0x0000D000,       // W_220, W_430x, DIO30
W_AXIS_PD6       = 0x00006700,       // W_230, Pulse-Direction Servo Amp, AB Encoder, DI2+DIO2
W_AXIS_PD0       = 0x00006000,       // W_230, Pulse-Direction Servo Amp, DI2+DIO2
W_AXIS_2S0       = 0x00002000,       // W_520, 2P STEP, DI3+DIO1+DO1
W_AXIS_2S7       = 0x00002700,       // W_520, 2P STEP, AB Encoder, DI3+DIO1+DO1
W_AXIS_2S8       = 0x00002800,       // W_520, 2P STEP, 485 Encoder, DI3+DIO1+DO1
W_AXIS_1V7       = 0x00001700,       // W_520, 1P VCM, AB Encoder, DI3+DIO1+DO1
W_AXIS_1VB       = 0x00001B00,       // W_520, 1P VCM, 485 P5 Encoder, DI3+DIO1+DO1
```

센서 타입 설정 파라미터
W_SENSOR_IGNOR : 기존 설정 삭제 W_SENSOR_A : A접점 입력, 감속 정지 W_SENSOR_B : B접범 입력, 감속 정지 W_SENSOR_A_STOP_WITHOUT_DECEL : A접점 입력, 즉시 정지 W_SENSOR_B_STOP_WITHOUT_DECEL : B접점 입력, 즉시 정지 W_SENSOR_2ND_A : 리미트 센서를 2중으로 사용할때 A접점 입력, 감속 정지 W_SENSOR_2ND_B : 리미트 센서를 2중으로 사용할때 B접점 입력, 감속 정지

9-2. listSetELN(unsigned int axis, unsigned int chDI, unsigned int sensorType)

: End Limit - 센서 설정

예) wErr = listSetELN(0, 0, W_SENSOR_A); // 0번축의 0번 입력을 -LIMIT로 설정, A접점 감속 정지

9-3. listSetELP(unsigned int axis, unsigned int chDI, unsigned int sensorType)

: End Limit + 센서 설정

예) wErr = listSetELP(10, 1, W_SENSOR_IGNORE); // 10번축의 1번 입력의 설정을 해제
wErr = listSetELP(1, 1, W_SENSOR_B); // 1번축의 1번 입력을 + LIMIT로 설정, B접점 감속 정지

9-4. listSetORG(unsigned int axis, unsigned int chDI, unsigned int sensorType)

: 홈 센서 설정 ※ 센서를 래치하기 때문에 W_SENSOR_x_STOP_WITHOUT_DECEL를 사용하면 에러가 납니다.

예) wErr = listSetORG(10, 1, W_SENSOR_IGNORE); // 10번축의 1번 입력의 설정을 해제
wErr = listSetORG(1, 2, W_SENSOR_A); // 1번축의 2번 입력을 홈센서로 설정, A접점 센서 래치

9-5. listSetACC(unsigned int axis, unsigned int accel)

: 가속도 설정, 설정 범위 20000pps(0.02rps) ~ W_PPR*4000pps (4000rps)
※ W_PPR=100,000이고 SlowP가 기본설정인 10일 때 1회전임. 12.5 * W_PPR이면 12.5 RPS(1초당 회전 속도)

예) wErr = listSetACC(0, 10 * W_PPR)); // 0번축 가속도를 1초에 10RPS(600RPM) 기울기로 증가하도록 설정

9-6. listSetDEC(unsigned int axis, unsigned int decel)

: 감속도 설정, 설정 범위 20000pps(0.02rps) ~ W_PPR*4000pps (4000rps
※ W_PPR=100,000이고 SlowP가 기본설정인 10일 때 1회전임. 12.5 * W_PPR이면 12.5 RPS(1초당 회전 속도)

예) listSetDEC(15, 350000); //15번 축의 감속도 설정, 3.5 RPS(1초당 회전 속도)임

9-7. listSetPPS(unsigned int axis, unsigned int pps)

: 속도 설정, 설정 범위 20000pps(0.02rps) ~ W_PPR*4000pps (4000rps) ※ W_PPR = 1,000,000
※ W_PPR=100,000이고 SlowP가 기본설정인 10일 때 1회전임. 12.5 * W_PPR이면 12.5 RPS(1초당 회전 속도)

예) listSetPPS (5, 50 * W_PPR); //5번 축의 속도 설정, 50 RPS(1초당 회전 속도) = 3000 RPM(1분당 속도)

회전 방향 설정	W_DIR_P (정방향) W_DIR_N (역방향) W_DIR_AUTO (절대 위치에서 자동 방향)

9-8. listMove(unsigned int axis, unsigned int dir, unsigned int distance)

: 상대 좌표로 이동

예) listMove (25, W_DIR_P, 2.5 * W_PPR)); // 25번 축이 + 방향으로 2.5회전 만큼 상대값으로 회전

9-9. listMoveTo(unsigned int axis, unsigned int dir, int position)

: 절대좌표 이동

예) listMoveTo (2, W_DIR_AUTO, -1 * W_PPR)); // 2번 축이 자동 방향으로 -1,000,000의 절대값으로 회전
W_DIR_AUTO가 아닌 W_DIR_P, W_DIR_N으로 절대값의 이동 방향 설정이 가능합니다.
만일 0위치에서 listMoveTo(2, W_DIR_P, -1 * W_PPR)); 명령이면 정회전으로 계속 회전해서
오버플로우가 난 후에도 계속 회전해서 -1 * W_PPR 위치까지 계속 돌면서 양방향으로 약 4200회전을 하게 됨.

입/출력 타입 설정	W_DIO_AXIS W_DIO_AMP W_DIO_MIO W_DIO_EDGE_RISE W_DIO_EDGE_FALL

9-10. listOnDO(unsigned int axis, unsigned int dioType, unsigned int bits)

: 0x1인 비트에 대해 DO ON

예) listOnDO(0, W_DIO_AXIS, 0x1 << 3); // 0번 축의 3번 DO(일반 접점)를 list의 순번이 오면 출력함.
W520의 경우 5점 (DI 0, DI 1, DI 2, DIO 3, DO 4 : 입력 3점, 입출력 공용 1점, 출력 1점이 있음)

9-11. listOffDO(unsigned int axis, unsigned int dioType, unsigned int bits)

: 0x1인 비트에 대해 DO OFF

예) listOffDO (2, W_DIO_AXIS, 0x1 << 4); // 2번 축의 4번 DO (일반 접점)를 list의 순번이 오면 OFF 함.

9-12. listSet(unsigned int axis, unsigned int idx, unsigned int value)

: W_IDX_x 파라미터에 따른 값 지정 ▶ '목차 11'의 21페이지에서 상세하게 설명합니다.

9-13. listSetPOS(unsigned int axis, int position)

: 위치값 설정

예) listSetPOS(3, -2.5 * W_PPR); // 3번 축의 현재 논리적 절대 위치값을 -250000으로 변경함.

9-14. listSetENC(unsigned int axis, int encoder)

: 엔코더값 설정

예) listSetENC (2, 8.5 * W_PPR); // 2번 축의 현재 엔코더 입력값을 850000으로 변경함.

9-15. listSetVAR(unsigned int axis, unsigned int varIdx, unsigned int value)

: Wdevice 내부의 VAR 변수의 값을 설정, VAR은 2048개의 인덱스에 각각 32비트값을 저장하는게 가능합니다.

예) for (int i=1; i<=6; i++) listSetVAR(0, i, I * W_PPR);
// 0번 축의 VAR 1번~6번 인텍스에 1회전~6회전 값을 list 실행 순서에 각각 저장합니다.

9-16. listFeed(unsigned int axis, feedType)

: 리스트에 이 순서가 오면 수동으로 즉시 해당 데이터를 피드백

feedType	
W_FEED_LIST_NUM	// 해당축 대기중인 함수 리스트 수
W_FEED_USER_VALUE	// UserValue값 반환 (listAction 함수 등에 의해 변경된 값 확인 등에 사용)
W_FEED_USER_VALUE_INC	// ++UserValue값 반환 (특정 루틴이 몇 회 수행되었는지 판단 등에 사용)
W_FEED_LATCH_PO	// 함수 실행 시점의 Position 위치값을 래치해서 반환
W_FEED_LATCH_ENC	// 함수 실행 시점의 Encoder 엔코더값을 래치해서 반환
W_FEED_LATCH_DIO	// 함수 실행 시점의 DIO 입출력상태를 래치해서 반환
W_FEED_DEVICE	//
W_FEED_520	//
W_FEED_230	//
W_FEED_130	//
W_FEED_220	//
W_FEED_030	//
W_FEED_520D	//

예) listFeed(2, W_FEED_LIST_NUM); // 2번 축에 리스트 순번이 오면 즉시 2번축의 W_ERR 상태를 반환함.
Wnet의 Feed Time 주기로 상태값이 매번 업데이트되지만 이 명령은 즉시 상태값이 반환되어 업데이트 되므로
Feed Time 주기를 기다리는 시간 지연을 방지함. 따라서 Feed Time 주기는 길게 가져가면서도
즉시 상태값을 반환해서 활용하는 코딩이 가능함.

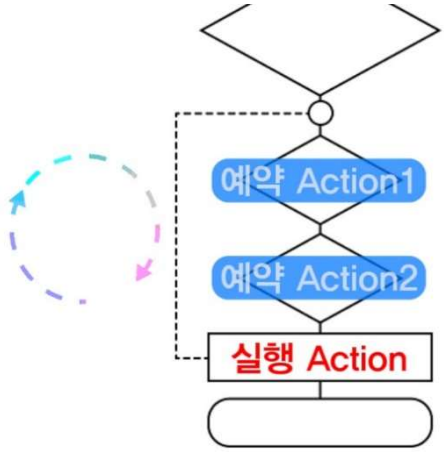
9-17. listAction( )의 사전 설명

1) listing 기능이 순서대로 쌓이고 먼저 온 명령을 순차적으로 실행하는 버퍼라면
Action 기능은 조건을 계속 감시하여 실행하고, Go, Stop 기능으로 대기와 진행, 다른 list로 점프하는 제어 구조입니다.

2) listing 기능으로도 분산 제어는 되지만 관리 프로그램의 구조가 좋아야 딜레이 없는 분산 제어가 가능합니다.
하지만 listAction은 관리까지도 Wdevice에게 넘김으로써 편리하게 궁국의 딜레이 없는 다축의 분산 제어가 가능합니다.
그래도 listing 기능만으로도 웬만한 장비의 제어에는 별문제가 없기 때문에 억지로 listAction를 사용하지 않아도 됩니다.

3) 혹시 listAction의 제어 구조에 대한 충분한 이해없이 코딩한다면 오동작의 위험이 있기 때문에, 오동작도 문제가 없는 환경에서 연습해주시기 바랍니다. 잘되던 코드에서 오동작이 발생하는 그런 문제가 아니고 연습 때의 실수의 문제입니다.
제공되는 예제를 따라하고 몇번의 연습 코딩을 해보면 이해가 되므로, 꼭 제공되는 프로그램 예제를 학습해주십시오.

4) 예약 Action들과 맨 아래의 실행 list는 한 덩어리의 블럭 함수로 간주합니다.
따라서 실행 list가 완료되어 삭제되면 예약 Action도 함께 삭제됩니다.
BACK/FORWORD의 명령으로 위로 점프시에 다시 예약됩니다.
만일 예약 Action만 있고 실행 Action이 없는 경우는 계속 살아있으며 따라서 예약 함수가 삭제되는 경우는 밑의 실행 list 함수가 완료되거나 예약 Action 이 조건을 만족하여 분기 때이며
아예 listAction(axis, x, x, x, W_ACT_CLEAR_ACTION, x, x); 명령으로 지금까지 예약을 전부 클리어할 때도 삭제되며 nowStop(); 에도 삭제됩니다.



5) 예약 Action 이 실행 Action 없이 사용되었을 경우는 리스트가 끝나도 살아서 계속 동작합니다.
이렇게 동작해야 코딩이 편한 경우들이 있기 때문에, 이러한 상황이 불필요하다면 listAction(axis, x, x, x, W_ACT_CLEAR_ACTION, x, x); 명령을 실행 합니다. (nowStop( )도 가능하지만 list 가 전부 클리어됩니다.)

6) 예약 Action 함수을 최대 8 개까지, 아래에 있는 실행 list 함수에 조건과 액션의 예약을 걸어놓습니다..
예약용 listAction 함수를, 실행 Action 함수로 사용하는 경우에는
스스로가 하나의 예약 Action 함수가 되므로 자신을 포함해서 8 개가 됩니다.

7) 루프를 돌게 해주는 W_ACT_GOSTOP_BACKWARD, W_ACT_GOSTOP_FORWARD 는
축이 이동 동작을 하는 중에, 이 함수의 실행 조건이 만족되서 실행을 해야 되는 난감한 상황에 빠지면,
W_ALARM_LIST_MISMATCH 알람이 발생합니다.

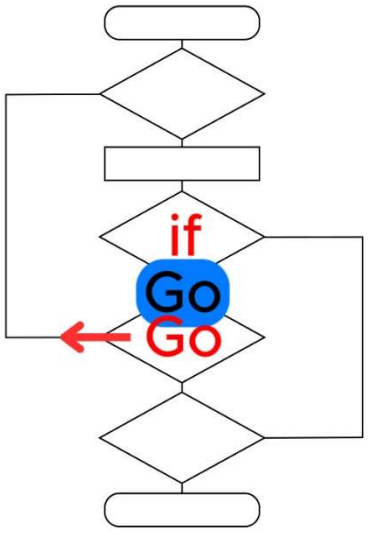
8) W_IF_DIO_ON/OFF 는 한번 실행되면 지워지지만, W_IF_DIO_RISING/FALLING 는 실행 Acton 이 실행된 이후에도 삭제되지 않고 계속 적용 됩니다.

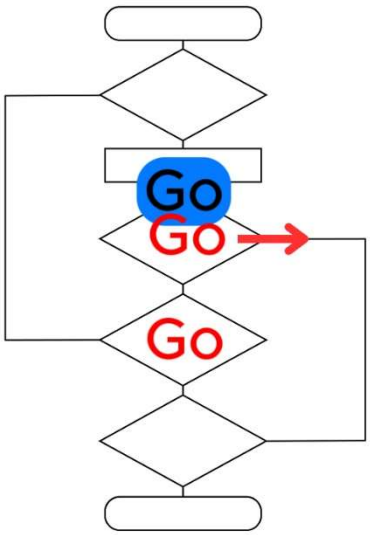
9) list 호출 시간을 고려해서 프로그래밍 해야합니다. (예제 2. VAR 2 축 NG 상황을 확인 바랍니다.)

※ 예제가 필요한 설명이 있어서 '9-20. 예제' 편을 반드시 읽어봐 주시기 바랍니다.

9-18. listAction()의 주요 3 가지 기능

1) Go 기능

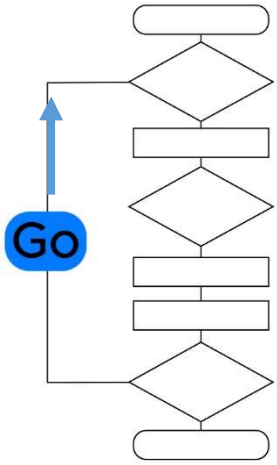


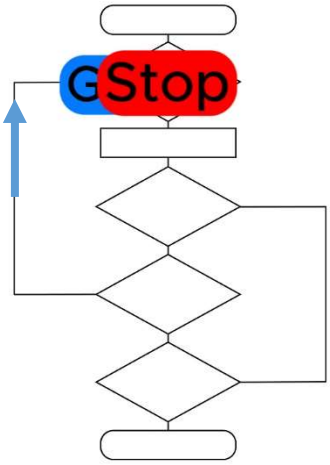


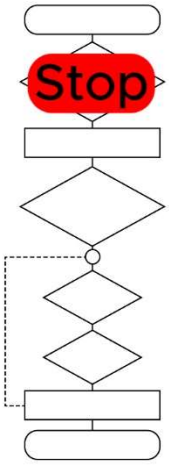
listing 되어 쌓여있는 함수들에서
되돌아가서 루프를 돌거나
앞으로 넘어가서 루프를 벗어나는 등의
흐름을 제어해주는 기능.

listSet( ) 명령으로 무조건 되돌아 가거나
listAction( ) 명령으로 조건 만족시에 앞 또는
뒤로 분기시킬 수 있으며
해당하는 GoStop Point 로 명령 실행이
이동하게 됩니다.

2) Stop 기능





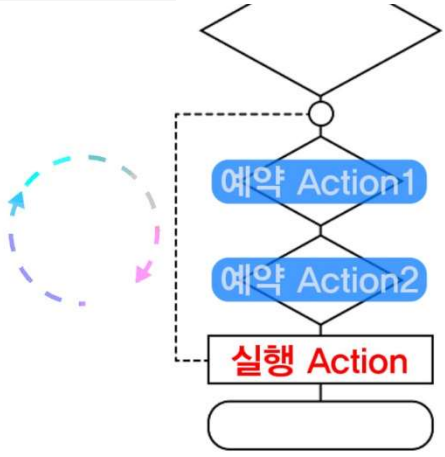


각 축에서 함수가 진행되는 것을
대기시키고 시작을 시점을 조정해서
축 사이의 동기를 맞춰주는 기능

각 축의 GoStop Value 값이
해당 GoStop Point 값과 같아야 통과되어
계속 진행되며 이걸로 각 축의 동기를
맞출 수 있습니다.

단, 해당 GoStop Point 값이 0~7 이면
무조건 통과됩니다.

3) Action 기능



실행 Action 의 완료까지 예약 Action 이 조건을 감시해서
예약 Action 의 조건에 부합하면 예약된 Action 을 실행하고
실행 Action 완료까지 실행되지 않으면 함께 삭제됩니다.

실행 Action 에 총 8 개의 예약 Action 을 걸어놓을수있습니다.
자신이 예약 listAction 함수인 경우는 자신도 포함해서 8 개입니다.

실행 Action 없이 예약 Action 이 걸려있는 경우에는
삭제되지 않고 계속 조건을 감시, 실행합니다.
이걸 삭제시키려면 listAction(axis, x, x, x, W_ACT_CLEAR_ACTION, x, x);
명령을 실행 합니다. (nowStop( )도 가능하지만 list 가 전부 클리어됩니다.)

※ 예제가 필요한 설명이 있어서 '9-20. 예제' 편을 반드시 읽어봐 주시기 바랍니다

9-19. listAction(axis, ifIdx, ifTarget, ifValue, actIdx, actTarget, actValue)의

: 특정 조건에서 지정 동작 수행

listAction 함수는 궁극의 분산 제어를 쉽게 가능하게 만드는 좋은 기능이지만,
이 함수에 대한 이해 없이 사용하면 실수에 의한 오작동이 발생할 수 있습니다.
이 기능을 사용할 때는 구조를 이해하고 예제들을 따라해본 후에 사용해 주시길 부탁드립니다.
listAction의 사전 설명, 개념도는 13페이지부터, 예제 설명은 17페이지에 있습니다.

ifIdx : 조건 지정 인덱스	listAction(axis, **ifIdx**, ifTarget, ifValue, actIdx, actTarget, actValue)
W_IF_FALSE	if조건이 항상 FALSE로 actIdx가 수행되지 않음 예) listAction(axis, W_IF_FALSE, 0, 0, W_ACT_ON_DO, 3, 0); // axis축 DO3이 ON 실행 안됨
W_IF_TRUE	if조건이 항상 true이므로, 예약 순서 상관없이, 즉시 actIdx 수행 예) listAction(axis, W_IF_TRUE, 0, 0, W_ACT_ON_DO, 3, 0); // 무조건 axis축 DO3이 ON
W_IF_MSEC	ifValue : msec 단위의 대기 시간, ifValue이 0일 때의 특별한 의미는 개념 편에서 설명 예) listAction(axis, W_IF_MSEC, 0, 2500, W_ACT_MOVETO_USER, 0, 0); // axis축이 2.5초 후에 userValue값으로 절대값 이동
W_IF_DIO_ON W_IF_DIO_OFF	ifTarget : 대상 DIO 채널번호 예) listAction(axis, W_IF_DIO_ON, 2, 0, W_ACT_MOVETO_USER, 0, 0); // axis축 DI2가 ON이면 userValue값으로 절대값 이동
W_IF_DIO_RISING W_IF_DIO_FALLING,	ifTarget : 대상 DIO 채널번호. ▶ **RISING, FALLING은 다른 함수 명령들과는 다르게,** **실행 후에도 삭제되지않고 계속 적용됩니다. 예제 편에서 상세 설명**
W_IF_LIST_EMPTY W_IF_LIST_NOT_EMPTY	ifTarget : 조건 비교할 대상 axis 번호. 잔여 리스트 없음, 있음 예) listAction(axis1, W_IF_LIST_EMPTY, axis2, 0, W_ACT_MOVETO_USER, 0, 0); // axis2축의 list가 없으면 userValue값으로 절대값 이동
W_IF_MOVE W_IF_ACCEL_END W_IF_DECEL_START W_IF_READY	ifTarget. 동작중, 가속완료, 감속시작, 서보 ON + 정지상태 + 알람 없음 예) listAction(axis1, W_IF_MOVE, axis2, 0, W_ACT_MOVETO_USER, 0, 0); // axis2축이 MOVE중이면 userValue값으로 절대값 이동
W_IF_POS_LT W_IF_POS_GT W_IF_ENC_LT W_IF_ENC_GT	ifTarget : 대상 axis 번호, ifValue : 비교 대상 위치/엔코더 값 예) listAction(axis1, W_IF_POS_GT, axis2, 1000000, W_ACT_MOVETO_USER, 0, 0); // axis2축의 논리 위치값이 1000000 보다 크면 userValue값으로 절대값 이동
W_IF_USER_CHANGED	ifTarget : UserValue 변경된 것을 감시할 대상 axis 번호
W_IF_USER_NE W_IF_USER_EQ W_IF_USER_LT W_IF_USER_GT	ifTarget : 대상 axis 번호, ifValue : 비교 대상 UserValue 값 예) listAction(axis1 W_IF_USER_EQ, axis2, 100, W_ACT_MOVETO_USER, 0, 0); // axis2축의 UserValue가 100과 같으면 userValue값으로 절대값 이동
W_IF_GOSTOP_CHANGED	ifTarget : GostopValue 변경된 것을 감시할 대상 axis 번호
W_IF_GOSTOP_NE W_IF_GOSTOP_EQ W_IF_GOSTOP_LT W_IF_GOSTOP_GT	ifTarget, ifValue. W_IF_GOSTOP_EQ + W_ACT_NEXT 인 경우, ifTarget 축의 GostopValue와 비교하는 GOSTOP_POINT로 지정 됨 예) listAction(axis1 W_IF_GOSTOP_EQ, axis2, 10, W_ACT_MOVETO_USER, 0, 0); // axis2축의 GoStopValue가 10과 같으면 userValue값으로 절대값 이동

actIdx : 지정 동작 인덱스	listAction(axis, ifIdx, ifTarget, ifValue, **actIdx**, actTarget, actValue)
W_ACT_NEXT	ifIdx 조건 만족시 다음 리스트 수행
W_ACT_ORG_N W_ACT_ORG_P	W_IF_DIO_ON/OFF 에 따라, A/B접점 원점 복귀 매크로 수행
W_ACT_MOVE_N_VAR, W_ACT_MOVE_P_VAR, W_ACT_MOVETO_VAR	actTarget를 인덱스로 하는 VAR 값으로 거리 이동 혹은 위치 이동. (0인 경우 UserValue, 1인경우 ++UserValue를 인덱스로 사용)
W_ACT_MOVETO_USER	UserValue값으로 위치 이동
이하 W_ACT_x 는, listAction 조건을 최대 8개까지 예약하고는 다음 리스트 동작 수행 (cf. 상기 W_ACT_x는 다음 리스트로 넘어가지 않음)	
W_ACT_GOSTOP_BACKWARD W_ACT_GOSTOP_FORWARD	actValue값으로 태그된 Gostop 위치로 돌아가거나, 건너뛰기 함 ▶ 축 이동 동작 중에 수행이 요청되면, W_ALARM_LIST_MISMATCH 발생
W_ACT_OVERRIDE_PPS	조건 만족시, actValue에 지정된 속도로 오버라이딩 ▶ 0이면 정지 시작이므로 정지 조건으로 사용됨
W_ACT_ADJUST_DISTANCE	actValue 만큼 이동 거리를 증가시키거나 차감 함 ▶ 잔여거리가 감속거리 이하면, 정지 시작
W_ACT_ON_DO W_ACT_OFF_DO	actTarget에 지정된 DIO 채널을 ON, OFF
W_ACT_ON_DO_VAR W_ACT_OFF_DO_VAR	actTarget을 인덱스로 하는 VAR 값 DO 채널을 ON, OFF ▶ 0인 경우 UserValue 값, 1인경우 ++UserValue를 인덱스로 사용
W_ACT_USER_VALUE W_ACT_USER_INC	actTarget에 지정된 축의 UserValue 를 설정 혹은 1 증가
W_ACT_GOSTOP_VALUE W_ACT_GOSTOP_INC	actTarget에 지정된 축의 GostopValue 를 actValue 로 설정 혹은 1 증가
W_ACT_LATCH_POS W_ACT_LATCH_ENC	actTarget에 지정된 축의 위치값, 엔코더값을 래치해서 반환
W_ACT_LATCH_DIO	해당 디바이스의 전체 축 DIO값을 래치해서 반환
W_ACT_BEEP_PLAY	actValue W_BEEP_x 지정하여, 특정 조건을 비프음 재생으로 확인
W_ACT_CLEAR_ACTION	예약된 Action listing 모두 클리어
단순 연산 인덱스	
W_ACT_USER_PLUS W_ACT_USER_MINUS	actTarget에 지정된 축의 UserValue에 actValue값을 더하거나 빼기
W_ACT_USER_PLUS_VAR W_ACT_USER_MINUS_VAR	UserValue에, actTarget을 인덱스로 하는 VAR 값을 더하거나 빼기
W_ACT_USER_FROM_VAR W_ACT_USER_TO_VAR	actTarget을 인덱스로 하는 VAR 값으로 UserValue를 설정하거나 그 반대
W_ACT_USER_FROM_GOSTOP W_ACT_USER_TO_GOSTOP	해당축의 UserValue를 actTarget축의 GostopValue값으로 설정하거나 그 반대
W_ACT_GOSTOP_FROM_USER W_ACT_GOSTOP_TO_USER	해당축의 GostopValue를 actTarget축의 UserValue값으로 설정하거나 그 반대

9-20. listAction( ) 예제

여기에 나오는 예제는 제공되는 프로그램의 버튼을 누르면 실행되는 예제들입니다.

1) GoStop 예제

```
// listAction(axis, ifIdx, ifTarget, ifValue, actIdx, actTarget, actValue)   //
// W_IF_GOSTOP_EQ + W_ACT_NEXT 인 경우, ifTarget 축의 Gostop Value와 비교하는 GOSTOP POINT로 지정됨 //

    nowSet(0, W_IDX_GOSTOP_VALUE, 0); // 0축 GV(Gostop Value)는 0 설정
    listSet(0, W_IDX_GOSTOP_POINT, 8); // 0축 GP(Gostop POINT) 8로 설정
                                       // GV은 0이므로 GP8을 통과 못함. 맨아래 nowSet()명령으로 GV가 8이되면서 통과
    listSet(0, W_IDX_GOSTOP_VALUE, 11); // 0축의 GV는 11
    listAction(0, W_IF_TRUE, 0, 0, W_ACT_GOSTOP_FORWARD, 0, 11); // 0축 GP 11로 이동

    listAction(0, W_IF_GOSTOP_EQ, 1, 15, W_ACT_NEXT, 0, 0); // 0축 GP 15 설정, 1축의 GV와 비교되지만, 0축의 GP 15지점임.
    listSet(0, W_IDX_GOSTOP_VALUE, 21); // 0축 GV = 21
    listAction(0, W_IF_TRUE, 0, 0, W_ACT_GOSTOP_FORWARD, 0, 21);

    listAction(0, W_IF_GOSTOP_EQ, 0, 11, W_ACT_NEXT, 0, 0); // 0축 GP 11로 설정됨.
    listSet(1, W_IDX_GOSTOP_VALUE, 15);   // 1축 GV는 15, 이건 1축의 리스트이므로 이미 실행되었습니다.
    listSet(0, W_IDX_GOSTOP_BACKWARD, 15) // 0축 GP 15로 이동;

    listAction(0, W_IF_TRUE, 0, 0, W_ACT_GOSTOP_FORWARD, 0, 8); // 실행되지 않음
    listSet(0, W_IDX_GOSTOP_BACKWARD, 8); // 실행되지 않음

    listSet(0, W_IDX_GOSTOP_POINT, 21); //GOSTOP #21
    listSet(0, W_IDX_BEEP_PLAY, W_BEEP_DOUBLE); // 알림음 발생

    listFeed(0, W_FEED_GOSTOP); //디버깅 용으로 Log file에 표시됩니다.

    listFeed(0, W_FEED_VAR | 3); // 디버깅 용으로 Log file에 표시됩니다.
    listFeed(1, W_FEED_VAR | 3); // 1번축 리스트이므로 먼저 실행되어 있습니다.

    Sleep(1000);
    nowSet(0, W_IDX_GOSTOP_VALUE, 8);
```

① 중요한 점은 0 축의 리스트에 쌓이고있는데, listSet(1, W_IDX_GOSTOP_VALUE, 15); listFeed(1, W_FEED_GOSTOP);
listFeed(1, W_FEED_VAR | 3); 은 1 축의 리스트에 쌓인다는 점입니다.
1 축 리스트는 GOSTOP POINT 가 없으므로 대기하고있던 0 번축과는 달리 벌써 실행됩니다. 따라서 Log file 에서도
0 번축의 반환값이 1 축보다 늦게 나옵니다. 이것은 예제이며, 실제는 반드시 각 축 별로 리스트를 따로 코딩해야 합니다.

② 마직막에 즉시 실행 nowSet(0, W_IDX_GOSTOP_VALUE, 8)을 실행하는 이유는, 리스트에 GoStop 의 FORWARD 가 있는 경우에 만의 하나 list 에 전부 쌓이기도 전에 FORWARD 가 실행되는 것을 방지하는 의미입니다.

③ nowSet(0, W_IDX_GOSTOP_VALUE, 8)은 즉시 실행, listSet(0, W_IDX_GOSTOP_VALUE, 21)는 쌓여있는 순서대로 실행합니다. 복사하여 붙여놓을 때 실수하기 쉬우니 주의바랍니다.

④ 프로그램 처음 부분에 위의 nowSet(0, W_IDX_GOSTOP_VALUE, 0)처럼 관련 값들을 초기화 해주시면 좋습니다.

2) VAR 2 축 NG 예제 설명 (실행 시간의 차이에 대해)

0번축	1번축

listSet(0, W_IDX_USER_VALUE, 0);
listSetAxis(0, W_AXIS_050W); // 0축 모터 설정
for (int j=1; j<=6; j++) nowSetVAR(0, j,
(j%2)?(int)(0.1*W_PPR*(0+1)):-(int)(0.1*W_PPR*(0+1))); //6개의 위치
 	
listSet(0, W_IDX_GOSTOP_POINT, 8); //GOSTOP #8
listSet(0, W_IDX_GOSTOP_VALUE, 7); //Gostop Value = 7
listAction(0, W_IF_MSEC, 0, 200, W_ACT_MOVETO_VAR, 1, 0);
listAction(0, W_IF_TRUE, 0, 0, W_ACT_GOSTOP_INC, 1, 0);
listFeed(0, W_FEED_BYTE_USER); //로그파일에 USER VALUE 기록
listFeed(0, W_FEED_BYTE_GOSTOP); //로그파일에 GP값 기록
listAction(0, W_IF_USER_LT, 0, 6, W_ACT_GOSTOP_BACKWARD, 0, 8);
listSet(0, W_IDX_BEEP_PLAY, W_BEEP_DOUBLE);
listAction(0, W_IF_TRUE, 0, 0, W_ACT_CLEAR_ACTION, 0, 0); //클리어
	listSet(1, W_IDX_USER_VALUE, 0);
	listSetAxis(1, W_AXIS_050W);
	for (int j=1; j<=6; j++) nowSetVAR(1, j,
	(j%2)?(int)(0.1*W_PPR*(1+1)):-(int)(0.1*W_PPR*(1+1))); //6개의 위치
	
	listSet(1, W_IDX_GOSTOP_POINT, 8); //GOSTOP #8
	listSet(1, W_IDX_GOSTOP_VALUE, 7);
	listAction(1, W_IF_MSEC, 0, 200, W_ACT_MOVETO_VAR, 1, 0);
	listAction(1, W_IF_TRUE, 0, 0, W_ACT_GOSTOP_INC, 0, 0);
	listFeed(1, W_FEED_BYTE_USER);
	listFeed(1, W_FEED_BYTE_GOSTOP);
	listAction(1, W_IF_USER_LT, 1, 6, W_ACT_GOSTOP_BACKWARD, 0, 8);
	listSet(1, W_IDX_BEEP_PLAY, W_BEEP_DOUBLE);
	listAction(1, W_IF_TRUE, 0, 0, W_ACT_CLEAR_ACTION, 0, 0);
	
	nowSet(W_ALL, W_IDX_GOSTOP_VALUE, 8);

① 이 예제에서 0축 코드와 1축 코드 순으로 실행되는데, 이들 중에 1개만 이미 SERVO ON된 상태라면 다른 한축은 SERVO ON까지 시간이 걸리므로 즉시 실행되는 nowSet(W_ALL, W_IDX_GOSTOP_VALUE, 8); 명령들 받지 못하게 되고, 따라서 GOSTOP #8을 지나갈수 없게 됩니다.

② 제공한 예제 코드들의 아래에 있는 **// VAR 2축 OK** 예제 처럼, 서보 ON 상태라면 서로가 다른 축의 GOSTOP VALUE를 1증가시키는 명령을 추가한다면 어떤 상태라도 두 축이 동시에 시작합니다.

③ for (int j=1; j<=6; j++) nowSetVAR(1, j, (j%2)?(int)(0.1*W_PPR*(1+1)):-(int)(0.1*W_PPR*(1+1)));
테스트를 위한 6개의 위치값을 만들어서 VAR값에 저장합니다. userValue값을 인덱스로 사용해서 하나씩 증가하면서 6개의 위치를 순차적으로 이동합니다.

④ listFeed(0, W_FEED_BYTE_USER), listFeed(0, W_FEED_BYTE_GOSTOP);으로 오동작시에 이 값을 확인해서 디버깅합니다.
로그 파일에 기록된 값들을 확인할 수 있습니다.

⑤ listAction(0, W_IF_USER_LT, 0, 6, W_ACT_GOSTOP_BACKWARD, 0, 8); 명령은 뒤에 실행 list 함수가 없으므로 진행 과정이 끝나도 예약 Action은 삭제되지 않고 계속 걸려있게 됩니다.
이런 명령 실행 구조가 코딩에 유리한 점이 많이 있습니다만 다음 과정에도 영향을 미치게 되므로 생각하지 못한 동작을 할수도있습니다.
따라서 다음 진행 과정이 계속될 때에 영향을 미치면 안되는 경우에는 listAction(....., W_ACT_CLEAR_ACTION, 0, 0); 명령으로 그때까지의 예약 Action들을 전부 클리어 합니다.
(nowStop() 명령도 전부 클리어 되지만 리스트에 놓을 수가 없고, 리스트 자체가 전부 클리어 되기 때문에 사용할 수 없습니다.)

10. Non-Wait Function

10-1. nowSDK(unsigned int port, unsigned int category)

: Wsdk dll 로드 및 Wnet, Modbus, EtherCAT 연결하며 일반 Wnet의 category의 값은 W_CATEGORY_NULL입니다.

예) wErr = nowWSDK(3, W_CATEGORY_NULL); // 선택된 COMPORT 3으로 Wnet을 연결합니다.

CategoryID 종류	설 명
W_CATEGORY_NULL	Wnet 연결
W_MODBUS_NULL	Modbus 연결
W_CAT_QC	QC 모드 연결

10-2. nowState(unsigned int axis)

: W_ERR[31:0] 반환, [4. W_ERR 반환]에 상세 설명

예) nowState(5); // 5번 축의 W_ERR[31 bit~0 bit] 값을 반환, 모든 명령에 대해 반환하는 값과 같습니다.

10-3. nowStop(unsigned int axis)

: 축 정지 및 축에 리스트된 함수들을 클리어

예) nowStop(11); // 11번 축의 감속 정지 및 축에 리스트된 함수들을 클리어

10-4. nowStopEMG(unsigned int axis)

: 감속없는 비상 정지(axis=W_ALL을 통해 모든 축 정지시, DO출력 또한 리셋)

예) nowStopEMG (1); // 1번 축의 즉시 정지 및 축에 리스트된 함수들을 클리어

10-5. nowOnDO(unsigned int axis, unsigned int dioType, unsigned int bits)

: 0x1인 비트에 대해 DO ON

예) nowOnDO(1, W_DIO_AXIS, 0x1 << 3); // 0번 축의 3번 DO (일반 접점 타입)를 즉시 ON 함.
W520의 경우라면 5점 (입력 3점 DI 0, DI 1, DI 2, 입/출력 겸용 1점 DIO 3, 출력 1점 DO 4) 중에 DIO3를 ON하게 됩니다. , 만일 0x1 << 4 라면 DO4를 ON하게 됩니다.

10-6. nowOffDO(unsigned int axis, unsigned int dioType, unsigned int bits)

: 0x1인 비트에 대해 DO OFF

예) nowOffDO(2, W_DIO_AXIS, 0x1 << 4); // 2번 축의 4번 DO (일반 접점 타입)를 즉시 OFF 함.

10-7. nowSet (unsigned int axis, unsigned int idx, unsigned int value)

: W_IDX_x 파라미터에 따른 설정값을 지정 ▶ '목차 11' 21페이지에서 상세 설명합니다.

10-8. nowSetVAR(unsigned int axis, unsigned int varIdx, unsigned int value)

: Wdevice 내부의 VAR 변수의 값을 설정, VAR은 2048개의 인덱스에 각각 32비트값을 저장하는게 가능합니다.

예) for (int i=1; i<=6; i++) nowSetVAR(0, i, I * W_PPR);
// 0번 축의 VAR 1번 ~ 6번 인덱스에 1회전 ~ 6회전 값을 각각 즉시 저장합니다.

10-9. nowGet(unsigned int axis, unsigned int idx, unsigned int* value)

: W_IDX_x 파라미터에 따른 설정값 반환 ▶ '목차 11' 21페이지에서 상세 설명합니다.

10-10. nowGetDIO(unsigned int axis, unsigned int dioType, unsigned int* bits)

: DIO 값 반환

예) nowGetDIO(9, W_DIO_AXIS, &bits); // 9번 축의 DIO 값 반환

10-11. nowGetPOS(unsigned int axis, int* position)

: Position, 위치값 반환

예) nowGetPOS(1, &position); // 0번 축의 절대 위치 값 반환

10-12. nowGetENC(unsigned int axis, int* encoder)

: Encoder, 엔코더값 반환, 반화값은 1회전으로 설정된 값입니다.

예) nowGetENC (2, &encoder); // 2번 축의 엔코더값 반환

10-13. nowGetVAR(unsigned int axis, unsigned int varIdx, unsigned int* value)

: Wdevice 내부의 VAR 변수의 값을 읽음, VAR은 2048개의 인덱스에 각각 32비트값을 저장하는게 가능합니다.

예) for (int i=1; i<=6; i++) nowGetVAR(0, i, &getVarValue);
// 0번 축의 VAR 1번 ~ 6번 인덱스의 저장 값을 읽어 옵니다..

11. listSet(axis, W_IDX_x, value),
nowSet(axis, W_IDX_x, value),
nowGet(axis, W_IDX_x, value)의 W_IDX_x 파라미터

11-1. Wsdk 라이브러리 연결, Log file 기록, 모니터링 설정	
W_IDX_x	설명과 예제
W_IDX_LOG_LEVEL	Log file 을 기록하는 Log 레벨의 설정 ▶ 목차 3. Log file 기능(3 페이지)에서 상세 설명. 총 5 단계로 구성됨 : W_LOG_LEVEL_WSDK(알람만), W_LOG_LEVEL_CMD_EXCEPT_DO, W_LOG_LEVEL_CMD, W_LOG_LEVEL_CMD_AND_GUIDE, W_LOG_LEVEL_ALL 예) nowtSet(1, W_IDX_LOG_LEVEL, W_LOG_LEVEL_ALL); //모든 함수를 Log file 에 기록함.
W_IDX_LOG_LAUNCH	W_LOG_LAUNCH_DISABLE : Wnet 연결시 Log file 의 자동 오픈 기능을 사용 안함 W_LOG_LAUNCH_NOW : 저장하던 Log file 을 지금 바로 화면에 열기 예) nowtSet(0, W_IDX_LOG_LAUNCH, W_LOG_LAUNCH_NOW); // Log file 이 바로 열림.
W_IDX_LOG_DUMP	VAR 데이터를 대량으로 전송하여 Log file 에 기록함. nowSet()은 이 함수 호출 시점이고 listSet() 이 함수 완료 시점에 덤프 데이터를 기록함. 로그 레벨이 W_LOG_LEVEL_CMD_AND_FEED 이상의 설정이 필수임. 예) listSet(1, W_IDX_LOG_DUMP, xx);
W_IDX_FEED_NUM	축 별로 모니터링 데이터가 수신된 총숫자 예) nowGet(2, W_IDX_FEED_NUM, &feedNum); //2 번축 모니터링 데이터 총숫자 반환

W_IDX_FEED_MSEC	디바이스에서 주기적으로 모니터링 데이터를 보내는 시간 간격이며 단위는 msec
	디바이스 별로 설정이 가능하며, 범위는 1~500 으로 그 이상을 입력하면 500 으로 설정됨.
	예) nowSet(1, W_IDX_ MSEC, 50); // 50msec 마다 첫번째 디바이스의 데이터 수신
	nowGet(3, W_IDX_ MSEC, &value); // 3 번측 디바이스의 Feed 간격 설정값을 반환함

W_IDX_FEED_RISING_MASK W_IDX_FEED_FALLING_MASK	특정 DIO 의 상태에 따라 모니터링 데이터를 송신하도록 설정 예) nowSet(1, W_IDX_FEED_RISING_MASK,); //
W_IDX_LATCH	listFeed(axis, W_FEED_LATCH_x) 에 의한 반환값 확인 예) listSet(1, W_IDX_ LATCH,); //
W_IDX_CLEAR_W_ERR	W_ERR 을 W_ERR_NONE 으로 리셋 ※ 에러 리셋이 되지 않고 전원 재투입 필수인 알람도 있습니다. (목차 4-1 참조) 예) nowSet(1, W_IDX_CLEAR_W_ERR, 0); // 1 번 축 에러 메시지 리셋
W_IDX_WNET	Wnet 연결을 해지하는데 사용 예) nowSet(1, W_IDX_WNET, 0); // Wnet 연결을 해지함
W_IDX_CATEGORY	W_IDX_x 장비 CataegoryID 를 설정 (VIP 코드가 들어간 제품의 설정)
W_IDX_MODBUS	Modbus 통신 관련 설정
W_IDX_ETHERNET	EtherNet 통신 관련 설정
W_IDX_ETHERCAT	EtherCat 통신 관련 설정

리스트 제어, Action 관련	
W_IDX_x	설명과 예제

W_IDX_USER_VALUE	UserValue 설정. 축 지정을 W_ALL로 해서 전축 지정 가능.
	listSet + W_LOG_LEVEL_CMD_AND_GUIDE 이상인 조건에서 로그 파일에 기록됨
	예) nowSet(10, W_IDX_USER_VALUE, 50); // 10번축 UserValue값을 50으로 즉시 설정
	listSet(1, W_IDX_USER_VALUE, 0); // list 순서에 1번축 UserValue값을 0으로 설정

W_IDX_GOSTOP_VALUE	GostopValue 설정. Gostop의 태크 값이 8 이상이고 GostopValue와 값이 같을 때만 Gostop 지점을 통과함. 예) listSet(0, W_IDX_GOSTOP_VALUE, 10); // 0번축의 GoStopValue값을 10으로 설정.
W_IDX_GOSTOP_POINT	축 리스트의 현재 위치에 지정된 태그값의 Gostop 추가 예) listSet(1, W_IDX_GOSTOP_POINT, 10); //list의 이 위치에 1번축의 GoStopPoint 10 설정
W_IDX_GOSTOP_BACKWARD	일치하는 태그의 Gostop 위치로 돌아감. Gostop 고급기능은 listAction()의 관련 파라미터 사용 예) listSet(1, W_IDX_GOSTOP_BACKWARD, 10); // 1번축의 GoStop Point가 10로 설정된 위치로 뒤로 돌아감.

사용자의 모터 사용 전류를 직접 설정 **제시된 모델명으로 설정하는 경우는 WORK = COIL, BOOST = COIL * 1.2, IDLE = COIL * 0.2 로 설정됨**	
W_IDX_x	설명과 예제
W_IDX_A_COIL_X100	모터 사양서의 전류값으로 설정함. Motor on 할 때 모터 테스트용으로 사용됨. 모터 보호를 위해 5.6A 까지만 설정 가능하지만 별도 설정으로 더 높이는 것도 가능 예) wErr = listSet(7, W_IDX_A_COIL_X100, 120); // 7 번축, 1.2A 로 Coil 전류를 설정
W_IDX_A_WORK_X100	모터 구동시 사용되는 전류값을 설정함. 모터 보호를 위해 5.6A 까지만 설정 가능하지만 별도 설정으로 더 높이는 것도 가능 예) wErr = listSet(1, W_IDX_A_WORK_X100, 213); // 0 번축, 2.13A 로 WORK 전류를 설정
W_IDX_A_BOOST_X100	모터 가/감속시에 사용되는 전류값을 설정함. 모터 보호를 위해 6.8A 까지만 설정 가능, 하지만 별도 설정으로 더 높이는 것도 가능 예) wErr = listSet(1, W_IDX_A_BOOST_X100, 550); // 1 번축, 5.5A 로 BOOST 전류를 설정
W_IDX_A_IDLE_X100,	모터의 과열을 막기위해 정지시 사용되는 전류값을 설정함. 모터 보호를 위해 5.6A 까지만 설정 가능, 하지만 별도 설정으로 더 높이는 것도 가능 예) wErr = listSet(2, W_IDX_A_IDLE_X100, 50); // 2 번축, 0.5A 로 정지시 전류를 설정함.

축별 설정 파라미터	
W_IDX_x	설명과 예제
W_IDX_MULTI_P	전자 기어비 : 설정된 숫자만큼 거리, 속도, 가감속을 곱해서 동작 예) nowSet(1, W_IDX_MULTI_P, -10); // 0 번 축의 회전 방향이 반대로 되며 10 배의 설정으로 동작함. 10 대 1 의 감속기를 썼을때 최종단에서 동일하게 동작할 수 있음.
W_IDX_SLOW_P	축의 위치, 속도, 가속도의 배수, 기본 설정은 10 임. (1 ~ 360 까지 가능) 예) nowSet(1, W_IDX_SLOW_P, 1); // 1 번축이 1,000,000(SlowP=10)에서 100,000(SlowP=1)에 1 회전으로 변경. nowSet(1, W_IDX_SLOW_P, 36); // 3,600,000(SlowP=36)에 1 회전으로 설정
W_IDX_FAULT_CNT	SERVO 제어에서 여기에 설정된 시간까지 목표 위치값을 계속 추종하지 못하면 알람 발생 예) wErr = nowSet(1, ,); // 0 번 축이
W_IDX_FAULT_DIST	SERVO 제어에서 여기에 설정된 거리까지 목표 위치값으로부터 거리가 벌어지면 알람 발생 예) wErr = nowSet(1, ,); // 0 번 축이
W_IDX_RISING_FILTER	DI 에서 스위치 등에 발생하는 채터링을 방지하기 위해 입력되는 RISING 부분이 일정 시간 이하면 필터링함.
W_IDX_FALLING_FILTER	DI 에서 스위치 등에 발생하는 채터링을 방지하기 위해 입력이 끊기는 FALLIING 부분이 일정 시간 이하면 필터링
W_IDX_ADC_RATIO	모터 전류와 회생 전류값 간의 관계
W_IDX_GAP_RATIO	Gate 와 회생 전류값 간의 관계
W_IDX_GAIN_P	PID 제어에서 Gain 인 P 값을 설정함. 높을수록 위치값을 빠르게 낮을수록 느리게 추종함. 일반적으로 원판이나 벨트 등의 관성부하일수록 Gain P 의 값이 높은게 유리함 하지만 이것은 모터의 소음, 진동 등의 원인이 되기때문에 적당한 값의 세팅이 필요함. 설정 범위는 0 부터 까지 (기본 설정으로 를 사용) 예) wErr = nowSet(2, W_IDX_GAIN_P, 15); // 2 번축, Gain P 를 15 로 설정
W_IDX_GAIN_I	PID 제어에서 Gain 인 I 값을 설정함.
W_IDX_GAIN_D	PID 제어에서 Gain 인 D 값을 설정함.
W_IDX_GAIN_P2	PID 제어에서 Gain 인 P2 값을 설정함.
W_IDX_GAIN_D2	PID 제어에서 Gain 인 D2 값을 설정함.

디바이스별 설정 파라미터	
W_IDX_x	설명과 예제
W_IDX_SERIAL_ID	Serial ID 를 설정, 또는 확인 예) wErr = nowGet(1, W_IDX_SERIAL_ID, &getSerialID); // 0 번축, 디바이스 Serial ID 확인
W_IDX_DEVICE_ID	Device ID 를 확인 DeviceID : W_DEVICE_NULL, W_VIP, W_400, W_120, W_430, W_030, W_220, W_130, W_230, W_520, W_ALL, W_NOTSET
W_IDX_SUB_ID	Sub ID 를 설정, 또는 확인 예) wErr = nowGet(1, W_IDX_SUB_ID, &getSublID); // 1 번축, 디바이스 Sub ID 확인
W_IDX_MANUAL_ID	Manual ID 를 설정, 또는 확인 예) nowGet(2, W_IDX_MANUAL_ID, &getManualID); // 2 번축, 디바이스 Manual ID 확인
W_IDX_VIP_ID	Vip ID 를 설정, 또는 확인 예) wErr = nowGet(3, W_IDX_VIP_ID, &getVipID); // 3 번축, 디바이스 Vip ID 확인
W_IDX_NICK_ID	Nick ID 를 설정, 또는 확인 예) wErr = nowGet(4, W_ W_IDX_NICK_ID, &getNickID); // 4 번축, 디바이스 Nick ID 확인
W_IDX_AXES_NUM	디바이스 마다 할당된 축수를 확인, W_ALL 로 전체 축수 확인도 가능 예) wErr = nowGet(W_ALL, W_IDX_AXES_NUM, &getValue); // 전체 축수를 확인
W_IDX_FEED_TYPE	W_FEED_x 를 설정하거나 확인
W_IDX_FIRMWARE	디바이스의 펌웨어를 현재 저장된 바이너리 펌웨어로 강제 업데이트 ※ 라이브러리 파일들과 바리너리 파일을 최신으로 바꾸면 Wnet 연결시 자동 업데이트 예) wErr = nowSet(W_ALL, W_IDX_FIRMWARE, 0x33322209); // W_ALL(전축) 강제 업데이트
W_IDX_REBOOT	디바이스를 강제 재부팅 예) wErr = nowSet(W_ALL, W_IDX_REBOOT, 0x33322209); // W_ALL(전축) 강제 리부팅
W_IDX_FLASH	디바이스내의 CFG 정보를 플래시 메모리에 기록, 설정의 영구 저장이 가능합니다. 예) wErr = nowSet(0, W_IDX_FLASH, x); // 0 번 축의 CFG 정보를 영구 저장
W_IDX_ACTIVATION	디바이스의 라이센스를 관리하는 파라미터

W_IDX_BEEP_LEVEL	디바이스의 알림 비프음의 레벨을 설정
	예) nowSet(W_ALL, W_IDX_BEEP_LEVEL, W_BEEP_LEVEL_ALL_INFORM); // 모든 알람 허용
	단계별로 W_BEEP_LEVEL_NEVER, W_BEEP_LEVEL_ONLY_ALARM,
	W_BEEP_LEVEL_INCLUDE_WARNING, W_BEEP_LEVEL_INCLUDE_IO,
	W_BEEP_LEVEL_ALL_INFORM(기본 설정) 까지 설정 가능

W_IDX_BEEP_MASK	W_IDX_BEEP_LEVEL_INCLUDE_IO 에서 대상이되는 DIO 채널의 비트 마스크

W_IDX_BEEP_PLAY	listSet(axis, W_IDX_BEEP_PLAY, W_BEEP_x)로 특정 비프음 수동 재생 예) wErr = nowSet(1, W_IDX_BEEP_PLAY, W_BEEP_SHORT); // 0 번 디바이스에 짧은 '삐'를 1 회 즉시 소리냄 알림 소리의 종류는 아래와 같습니다. W_BEEP_NULL : 소리 멈춤 W_BEEP_ON : 상승하는 "도미솔" W_BEEP_SHORT : 짧은 "삐" 소리 W_BEEP_LONG : 긴 "삐~" 소리 W_BEEP_DOUBLE : 짧은 "삐" 소리 후에 긴 "삐~" 소리 W_BEEP_TOGGLE : "삐~익! 삐~익!" 소리 반복 W_BEEP_REGENERATIVE : 상승하는 "삐요~~~~~~~~~" W_BEEP_AXIS0, W_BEEP_AXIS1, W_BEEP_AXIS2, W_BEEP_AXIS3 : 짧은 "삐"가 해당 축번호만큼 울린 후에 "윅~" 소리 W_BEEP_INC0, W_BEEP_INC1, W_BEEP_INC2, W_BEEP_INC3 : 상승하는 "삐이익~" 소리, 축번호가 높아지면 소리도 높아짐 W_BEEP_DEC0, W_BEEP_DEC1, W_BEEP_DEC2, W_BEEP_DEC3 : 하강하는 "삐이익~" 소리, 축번호가 높아지면 소리도 낮아짐 W_BEEP_CURVE : 상승후, 하강하는 "삐이익~" 소리 W_BEEP_DOREMI : "도레이파솔라시도" W_BEEP_STEP_ON : 상승하는 "도미솔~" 소리 W_BEEP_STEP_OFF : 하강하는 "솔미도~" 소리 W_BEEP_FLYRUN_ON : 상승하는 "도도미미솔솔~" 소리 W_BEEP_FLYRUN_OFF : 하강하는 "솔솔미미도도~" 소리 W_BEEP_SERVO_ON : 상승하는 "도솔" 소리 W_BEEP_SERVO_OFF : 하강하는 "솔도~" 소리 W_BEEP_WOW : 경쾌한 "빠라바라바라밤" 소리 W_BEEP_ERR : 하강하는 "도솔라파솔미파미레도" 소리 W_BEEP_PIPO : "삐~뽀~ 삐~뽀~" 소리 W_BEEP_BUTTERFLY : 나비야 나비야 연주 W_BEEP_SCHOOL : 학교종이 땡땡땡 연주 W_BEEP_ONARA : 오나라 오나라 연주 "라시시 시라솔 미솔솔" W_BEEP_ANIRAO : "시시라솔미솔~" W_BEEP_ORG : 엘리제를 위하여 연주