V(0.0)

Firmware/Error Management

	dsp_version	제어기의 firmware version을 반환.
	dsp_dll_version	Library 파일 version을 반환
	download_firmware_file	제어기의 firmware를 변경.
	verify_firmware_file	제어기의 firmware를 검증.
	error_msg	에러 메시지 반환.

SYNTAX	int32 dsp_version(int16 net_id);
	int16 dsp_dll_version(P_INT high, P_INT middle, P_INT low);
	int16 download_firmware_file(int16 net_id, char *file);
	int16 verify_firmware_file(int net_id, char *file);
	int16 error_msg(int16 code, char *dst);	

PROTOTYPE IN	eNetAxis2-link.h	
DESCRIPTION	**dsp_version(…)**은 제어기 firmware version을 32비트 정수 값으로 반환합니다. **dsp_dll_version(…)**은 PC Library version을 반환합니다. **download_firmware_file(…)**은 제어기의 Firmware를 Firmware File(*.BIN)로 변경합니다. 이 함수는 예약되어 있습니다. **verify_firmware_file(…)**은 제어기의 Firmware를 Firmware File(*.BIN)과 검증합니다. 이 함수는 예약되어 있습니다 **error_msg(…)**는 함수들에서 반환된 error code에 대한 메시지를 반환합니다. 이 함수는 예약되어 있습니다.	
RETURN VALUES	**Dsp_version(…)**은 Firmware Version을 반환합니다. DSP_OK(0), DSP_FUNCTION_NOT_AVAILABLE(22)	

Network Configuration

	net_init	Host PC의 Network 환경을 초기화.
	net_exit	구성된 Host PC의 Network Resource 들을 해제.
	net_resume	두절된 통신 재개.
	find_pci_dsp	네트워크에 연결된 제어기들의 국번을 반환
	check_communication_time	통신 시간을 반환.

SYNTAX	int16 net_init(int16 net_id, int16 port);
	int16 net_exit();
	int16 net_resume(int16 net_id);
	int16 find_pci_dsp(int16* board_found, unsigned16* pci_address);
	int16 check_communication_time(int16 net_id, P_INT32 interval);	

PROTOTYPE IN	eNetAxis2-link.h	
DESCRIPTION	net_init(…)는 제어기와 통신할 PC의 Network 환경을 초기화합니다. PC의 IP Address중 앞의 3자리는 "192.168.0"으로 설정해야 합니다. 마지막에 설정된 값은 0부터 10안의 값으로 설정해 주십시오. 마지막에 설정된 PC의 IP Address 값을 net_id에 설정합니다. 서브넷 마스크는 "255.255.255.0"을 설정하고, 기본 게이크웨이는 "192.168.0.16"을 설정합니다. port는 제어기와 통신할 PC의 port값을 설정합니다. 2004,2005번 포트가 사용자를 위해 예약되어 있어 이 포트번호들을 사용하는 것을 추천합니다. 제어기와 PC의 네트워크 환경이 구성되어야 다른 함수들을 정상적으로 호출할 수 잇기 때문에 이 함수는 프로그램의 초기에 호출되어야 합니다. net_exit()는 net_init 함수 호출로 구성된 PC의 Network 환경 Resource들을 해제하고 전체 Network 환경을 종료합니다. 프로그램 종료 시 호출되어야 합니다. 호출되지 않는다면, PC Resource의 누수가 발생됩니다. net_resume(…)은 제어기들과 정상적인 운전 중 네트워크 케이블 단선, 또는 제어기의 전원 Off 등에 의해 제어기와 통신 상태가 비정상적인 상태 (DISCONNECT)로 네트워크 운전 조건이 변경된 경우, 제어기와 네트워크 운영 재개 시 호출합니다. find_pci_dsp(…)은 네트워크 연결되어 있는 제어기를 검색하여 제어기들의 국번을 반환합니다. Board_found는 제어기 발견시 TRUE로 설정되고, pci_address는 제어기 국번이 저장됩니다. 두 변수는 256개의 어레이 변수로 선언되어야 합니다. check_communication_time(…)은 제어기와 PC간의 통신시간을 반환합니다. PC에서 제어기 정보를 1000번 읽기를 수행하고 수행된 시간을 msec 당위로 반환합니다.	
RETURN VALUES	DSP_OK(0), DSP_DISCONNECT(50), NET_NOT_FOUND (1001), NET_ILLEGAL_PARAMETER(1002)	

Initialization/Terminate

	dsp_init	특정 제어기와 통신 환경을 초기화.
	dsp_exit	특정 제어기와 통신 환경을 종료.
SYNTAX	int16 dsp_init(int16 net_id); int16 dsp_exit(int16 net_id);	
PROTOTYPE IN	eNetAxis2-link.h	
DESCRIPTION	**dsp_init(…)**는 네트워크상에 있는 특정 IP 번지를 가진 제어기와 통신 환경을 초기화합니다. Net_id에 통신 환경을 초기화할 제어기의 IP 국번을 입력합니다. 이 함수는 PC 네트워크 환경이 초기화된 후 실행할 수 있으며, 이 함수 실행 후 특정 IP 제어기에 관련된 함수들을 정상적으로 호출할 수 있습니다. ㈜ net_init(…) 함수를 먼저 실행해 주십시오. **Dsp_exit(…)**는 특정 IP 번지 제어기와 통신하기 위한 환경을 해제합니다. Net_id에 통신 환경을 종료할 제어기의 IP 국번을 입력합니다. 이 함수를 실행하면 통신 환경 종료 전에 제어기 리셋 명령이 먼저 실행됩니다.	
RETURN VALUES	DSP_OK(0), DSP_NOT_INITIALIZED(1), DSP_NOT_FOUND (2), DSP_INVALID_AXIS(3), DSP_TIMEOUT_ERROR(14), DSP_DISCONNECT(50), NET_NOT_INITIALIZED(1000)	

Configure Axis

	set_motor_direction 정 방향에 대한 모터 회전 방향 설정. get_motor_direction 정 방향에 대한 모터 회전 방향 설정 값 반환. set_encoder_phase 정 방향에 대한 encoder phase를 설정. get_encoder_phase 정 방향에 대한 encoder phase를 설정 값 반환.

SYNTAX	int16 set_motor_direction(int16 net_id, int16 axis, int16 direction);
	int16 get_motor_direction(int16 net_id, int16 axis, P_INT direction);
	int16 set_encoder_phase(int16 net_id, int16 axis, int16 phase);
	int16 get_encoder_phase(int16 net_id, int16 axis, P_INT phase);

PROTOTYPE IN	eNetAxis2-link.h
DESCRIPTION	set_motor_direction(…)은 정 방향에 대한 모터의 회전 방향을 설정합니다. TRUE 값을 설정하면 모터 CW(clockwise)가 정 방향이고, FALSE 값을 설정하면 CCW(counter-clockwise)가 정 방향입니다. 방향 전환을 위해 pulse 출력 방향이 변경됩니다. get_motor_direction(…)은 정 방향에 대한 모터의 회전 방향 설정 값을 반환합니다. set_encoder_phase(…)은 정 방향에 대한 encoder phase를 설정합니다. TRUE 값을 설정하면 A상이 미리 검출되면 정 방향 회전이고, FALSE 값을 설정하면 B상이 미리 검출되면 정 방향 회전 입니다. get_encoder_phase(…)은 정 방향에 대한 encoder phase를 설정 값을 반환합니다.
RETURN VALUES	DSP_OK(0), DSP_NOT_INITIALIZED(1), DSP_NOT_FOUND (2), DSP_INVALID_AXIS(3), DSP_TIMEOUT_ERROR(14), DSP_DISCONNECT(50)

Motion Status

in_motion	이송 중이면, TRUE를 반환.
in_position	In-position 설정 범위 안에 있으면, TRUE를 반환.
axis_done	대기 중이고 in_position 이 TRUE이면, TRUE를 반환.
all_done	제어기 모든 축의 axis_done 이 TRUE이면 TRUE를 반환.
axis_status	축의 상태를 가리키는 bit로 구성된 정수를 반환.
axis_source	축에 발생된 이벤트 원인을 반환.
axis_state	축에 발생된 이벤트 종류를 반환.
axis_status_all	모든 축 정보를 32비트 정수 값으로 반환.

SYNTAX

```
int16 in_motion(int16 net_id, int16 axis);
int16 in_position(int16 net_id, int16 axis);
Int16 axis_done(int16 net_id, int16 axis);
int16 all_done(int16 net_id);
int16 axis_status(int16 net_id, int16 axis);
int16 axis_source(int16 net_id, int16 axis);
int16 axis_state(int16 net_id, int16 axis);
int32 axis_status_all(int16 net_id, int16 axis);
```

PROTOTYPE IN

eNetAxis2-link.h

DESCRIPTION

in_motion(…)은 축 이송 중이면, TRUE를 반환합니다.

In_position(…)은 축의 위치 오차가 set_in_position(…)으로 설정된 범위내에 있으면 TRUE를 반환합니다.

Axis_done(…)은 축 상태를 반환하는 함수로 축이 대기 중이고, in-position(..)이 TRUE 이면, TRUE를 반환합니다.

All_done(…)은 제어기의 모든 축의 axis_done(…)이 TRUE 이면, TRUE를 반환합니다.

axis_status(…)은 축 상태를 bit로 표현하는 정수를 반환합니다.

#define	Hex	Description
IN_DONE	0x01	대기 중임.
IN_POSITION	0x10	In-position 설정 범위 안에 있음.
IN_MOTION	0x40	이송 지령 중임.

axis_source(…)은 축에 발생된 이벤트 원인을 반환합니다.

#define	Value	Description
ID_NONE	0	이벤트가 없음.
ID_HOME_SWITCH	1	Home 입력 이벤트 발생.
ID_POS_LIMIT	2	+ Limit 입력 이벤트 발생.
ID_NEG_LIMIT	3	- Limit 입력 이벤트 발생.
ID_AMP_FAULT	4	Servo Ready 입력 이벤트 발생.
ID_X_NEG_LIMIT	7	- Software Limit 발생.
ID_X_POS_LIMIT	8	+ Software Limit 발생.
ID_ERROR_LIMIT	9	허용 오차 범위 초과 발생.
ID_PC_COMMAND	10	프로그램 적으로 이벤트(stop, e-stop)를 발생

Axis_state(…)는 축에 발생된 이벤트 종류를 반환합니다.

#define	Value	Description
NO_EVENT	0	이벤트가 없음.
STOP_EVENT	8	stop rate로 감속 정지.
E_STOP_EVENT	10	E-stop rate로 감속 정지.
ABORT_EVENT	14	Amp Disable.

axis_status_all(…)은 축 상태 정보를 bit로 모두 표현하는 32비트 정수 값을 반환합니다.

#define	Hex	Description
IN_DONE	0x01	대기 중임.
IN_POSITION	0x10	In-position 설정 범위 안에 있음.
IN_MOTION	0x40	이송 지령 중임.
IN_DIRECTION	0x80	+ 방향 이송,

RETURN VALUES

DSP_OK(0), DSP_NOT_INITIALIZED(1), DSP_NOT_FOUND (2), DSP_INVALID_AXIS(3), DSP_TIMEOUT_ERROR(14), DSP_DISCONNECT(50)

Event Management

	controller_idle	축에 Abort 이벤트를 발생.
	set_e_stop	축에 비상 정지 이벤트를 발생
	set_stop	축에 정지 이벤트를 발생
	set_e_stop_rate	비상 정지 이벤트 시 감속도 값 설정.
	get_e_stop_rate	비상 정지 이벤트 시 감속도 설정 값 반환.
	set_stop_rate	정지 이벤트 시 감속도 값 설정.
	get_stop_rate	정지 이벤트 시 감속도 설정 값 반환.

SYNTAX	int16 controller_idle(int16 net_id, int16 axis);
	int16 set_e_stop(int16 net_id, int16 axis);
	int16 set_stop(int16 net_id, int16 axis);
	int16 set_e_stop_rate(int16 net_id, int16 axis, double rate);
	int16 get_e_stop_rate(int16 net_id, int16 axis, P_DOUBLE rate);
	int16 set_stop_rate(int16 net_id, int16 axis, double rate);i
	nt16 get_stop_rate(int16 net_id, int16 axis, P_DOUBLE rate);	

PROTOTYPE IN	eNetAxis2-link.h	
DESCRIPTION	**controller_idle(…)**는 축에 Abort 이벤트를 발생시킵니다. 명령 실행 후, axis_state(…)는 ABORT_EVENT를 반환합니다. 명령 실행 후, axis_source(…)는 ID_PC_COMMAND를 반환합니다. 명령 실행 후, Amp Disable 합니다. 이 기능은 예약 기능입니다. **Set_e_stop(…)**는 축에 비상정지 이벤트를 발생시킵니다. 명령 실행 후, axis_state(…)는 E_STOP_EVENT를 반환합니다. 명령 실행 후, axis_source(…)는 ID_PC_COMMAND를 반환합니다. 명령 실행 후, 축은 감속 정지 합니다. **Set_stop(…)**는 축에 정지 이벤트를 발생시킵니다. 명령 실행 후, axis_state(…)는 STOP_EVENT를 반환합니다. 명령 실행 후, axis_source(…)는 ID_PC_COMMAND를 반환합니다. 명령 실행 후, 축은 감속 정지 합니다. **set_e_stop_rate(…)**은 비상 정지 이벤트 발생시 적용될 감속도 값을 설정합니다. **get_e_stop_rate(…)**는 비상 정지 이벤트 발생시 적용될 감속도 설정 값을 반환합니다. set_stop_rate(…)은 정지 이벤트 발생시 적용될 감속도 값을 설정합니다. **get_stop_rate(…)**는 정지 이벤트 발생시 적용될 감속도 설정 값을 반환합니다.	
RETURN VALUES	DSP_OK(0), DSP_NOT_INITIALIZED(1), DSP_NOT_FOUND (2), DSP_INVALID_AXIS(3), DSP_TIMEOUT_ERROR(14), DSP_DISCONNECT(50)	

Event Recovery

	clear_status	Stop 또는 E-Stop 이벤트를 삭제.
	controller_run	Abort 이벤트 삭제.
SYNTAX	int16 clear_status(int16 net_id, int16 axis); int16 controller_run(int16 net_id, int16 axis);	
PROTOTYPE IN	eNetAxis2-link.h	
DESCRIPTION	clear_status(…)는 축에 발생된 Stop 또는 E-Stop 이벤트의 정보를 삭제하고 축 상태를 정상화 합니다. 명령 실행 후, axis_state(…)는 NO_EVENT를 반환합니다. 명령 실행 후, axis_source(…)는 ID_NONE를 반환합니다. controller_run(…)는 축에 발생된 Abort 이벤트 정보를 삭제하고 축 상태를 정상화 합니다. 명령 실행 후, 지령 위치가 실제(Actual) 위치로 설정됩니다. 명령 실행 후, axis_state(…)는 NO_EVENT를 반환합니다. 명령 실행 후, axis_source(…)는 ID_NONE를 반환합니다. 이 기능은 예약 기능입니다.	
RETURN VALUES	DSP_OK(0), DSP_NOT_INITIALIZED(1), DSP_NOT_FOUND (2), DSP_INVALID_AXIS(3), DSP_TIMEOUT_ERROR(14), DSP_DISCONNECT(50)	

Position Control

	set_position	실제(Actual) 위치를 설정.
	get_position	실제(Actual) 위치를 반환.
	set_command	지령 위치를 설정.
	get_command	지령 위치를 반환.
	get_error	위치 에러를 반환.
	set_position2	지령 위치와 실제(Actual) 위치를 설정.
	set_error_limit	위치 오차 검사 범위와 action 값 설정.
	get_error_limit	위치 오차 검사 범위와 action 설정 값을 반환.
	set_negative_sw_limit	- 방향 software 에러 검사 위치와 action을 설정.
	get_negative_sw_limit	- 방향 software 에러 검사 위치와 action 설정 값을 반환
	set_positive_sw_limit	+ 방향 software 에러 검사 위치와 action을 설정.
	get_negative_sw_limit	+ 방향 software 에러 검사 위치와 action 설정 값을 반환

SYNTAX

```
int16 set_position(int16 net_id, int16 axis, double pos);
int16 get_position(int16 net_id, int16 axis, P_DOUBLE position);
int16 set_command(int16 net_id, int16 axis, double pos);
int16 get_command(int16 net_id, int16 axis, P_DOUBLE position);
int16 get_error(int16 net_id, int16 axis, P_DOUBLE error);
int16 set_position2(int16 net_id, int16 axis, double pos);

int16 set_error_limit(int16 net_id, int16 axis, double limit, int16 action);
int16 get_error_limit(int16 net_id, int16 axis, P_DOUBLE limit, P_INT action);
int16 set_negative_sw_limit(int16 net_id, int16 axis, double pos, int16 action);
int16 get_negative_sw_limit(int16 net_id, int16 axis, P_DOUBLE pos, P_INT
      action);
int16 set_positive_sw_limit(int16 net_id, int16 axis, double pos, int16 action);i
nt16 get_positive_sw_limit(int16 net_id, int16 axis, P_DOUBLE pos, P_INT
action);
```

PROTOTYPE IN

eNetAxis2-link.h

DESCRIPTION

Set_position(…)는 축의 실제(Actual) 위치를 재설정 합니다. 실제 위치를 설정 시 지령 위치도 재설정됩니다. 지령 전 두 위치간의 차이만큼 지령 위치도 재설정 됩니다.

Get_position(…)은 encoder로부터 입력되는 실제(actual) 위치를 반환합니다.

Set_command(…)는 축의 지령 위치를 재설정 합니다. 지령 위치를 설정 시 실제(Actual) 위치도 재설정됩니다. 지령 전 두 위치간의 차이만큼 실제(Actual) 위치도 재설정 됩니다.

Get_command(…)는 축의 지령 위치를 반환합니다.

Get_error(…)은 지령 위치와 실제(actual) 위치와의 차이를 반환합니다.

set_position2(…)은 축의 지령 위치와 실제(actual)위치를 같은 위치 값으로 재설정 합니다.

set_error_limit(…)은 위치 오차 검사 범위와 오차 검출시 제어기 action에 대한 값을 설정합니다.

get_error_limit(…)은 위치 오차 검사 범위와 오차 검출 시 제어기 action에 대한 설정 값을 반환합니다.

#define	Value	Description
NO_EVENT	0	이벤트가 없음.
STOP_EVENT	8	stop rate로 감속 정지.
E_STOP_EVENT	10	E-stop rate로 감속 정지.
ABORT_EVENT	14	Amp Disable.

set_negative_sw_limit(…)은 - 방향 software limit 위치 값과 에러 검출 시 제어기 action을 설정합니다.

get_negative_sw_limit(…)은 - 방향 software limit 위치 값과 에러 검출 시 제어기 action에 대한 설정 값을 반환합니다.

get_positive_sw_limit(…)은 + 방향 software limit 위치 값과 에러 검출 시 제어기 action에 대한 설정 값을 반환합니다.

RETURN VALUES

DSP_OK(0), DSP_NOT_INITIALIZED(1), DSP_NOT_FOUND (2),
DSP_INVALID_AXIS(3), DSP_TIMEOUT_ERROR(14), DSP_DISCONNECT(50)

Trajectory Control

	get_velocity	지령 속도를 반환.
	get_accel	지령 가속도를 반환.
	dsp_feed_rate	전체 축의 feed rate를 변경.
	axis_feed_rate	한 축의 feed rate를 변경.

SYNTAX	int16 get_velocity(int16 net_id, int16 axis, P_DOUBLE velocity);
	int16 get_accel(int16 net_id, int16 axis, P_DOUBLE accel);
	int16 dsp_feed_rate(int16 net_id, double feed_rate);
	int16 axis_feed_rate(int16 net_id, int16 axis, double feed_rate);	

PROTOTYPE IN	eNetAxis2-link.h	
DESCRIPTION	**get_velocity(…)**은 축의 지령 속도를 반환합니다. **get_accel(…)**은 축의 지령 가속도를 반환합니다. **dsp_feed_rate(…)**은 전체 축의 feed rate를 변경합니다. 0.0부터 1.0까지 설정 가능합니다. 0.5 설정은 50% feed rate 설정을 의미하고, 1.0은 100% feed rate 설정을 의미 합니다. **axis_feed_rate(…)**은 한 축의 feed rate를 변경합니다. 0.0부터 1.0까지 설정 가능합니다. 0.5 설정은 50% feed rate 설정을 의미하고, 1.0은 100% feed rate 설정을 의미 합니다.	
RETURN VALUES	DSP_OK(0), DSP_NOT_INITIALIZED(1), DSP_NOT_FOUND (2), DSP_INVALID_AXIS(3), DSP_TIMEOUT_ERROR(14), DSP_DISCONNECT(50)	

Inputs/Outputs/Home

	disable_amplifier	Servo On 출력 신호를 Amp Disable 신호 level로 출력.
	enable_amplifier	Servo On 출력 신호를 Amp Enable 신호 level로 출력.
	get_amp_enable	Amp enable 출력 상태를 반환.
	set_amp_enable_level	Amp enable 출력 active 신호 level 설정.
	get_amp_enable_level	Amp enable 출력 active 신호 level 설정 값을 반환.
	set_boot_amp_enable	Boot시 Servo On 출력 상태 설정.
	get_boot_amp_enable	Boot시 Servo On 출력 상태 설정 값 반환.
	set_boot_amp_enable_level	Boot시 amp enable 출력 active 신호 level 설정.
	get_boot_amp_enable_level	Boot시 amp enable 출력 active 신호 level 설정 값 반환.
	set_negative_limit	- 방향 limit 감지 시 action 설정
	get_negative_limit	- 방향 limit 감지 시 action 설정 값 반환.
	set_negative_level	- 방향 limit 입력 active 신호 level 설정.
	get_negative_level	- 방향 limit 입력 active 신호 level 설정 값을 반환.
	set_positive_limit	+ 방향 limit 감지 시 action 설정.
	get_positive_limit	+ 방향 limit action 설정 값 반환.
	set_positive_level	+ 방향 limit 입력 active 신호 level 설정.
	get_positive_level	+ 방향 limit 입력 active 신호 level 설정 값을 반환.
	amp_fault_switch	Amp fault 입력 신호를 반환.
	set_amp_fault_level	amp fault 입력 신호의 active 신호 level 설정.
	get_amp_fault_level	amp fault 입력 신호의 active 신호 level 설정 값 반환
	set_amp_fault	amp fault 신호 감지 시 action 설정.
	get_amp_fault	amp fault 신호 감지 시 action 설정 값 반환.
	set_home_index_config	Home logic을 구성.
	set_home_leve	Home logic의 enable 상태를 설정.
	get_home_level	Home logic의 enable 설정 값을 반환.
	set_home	Home logic이 감지되었을 때 동작 설정.
	get_home	Home logic 감지 시 동작 설정 값 반환.
	home_switch	Home logic 입력 상태를 반환.
	set_home_level_inverse	Home logic 신호 level을 반전하여 설정하거나 반환하라는 Library 내부 상태 변수 설정.
	get_home_level_inverse	Home logic 신호 level을 반전하여 설정하거나 반환하라는 Library 내부 상태 설정 값 반환.
	reset_motor	모터 리셋 신호 출력.
	get_amp_reset_output	모터 리셋 출력 상태 반환.
	set_amp_reset_level	모터 리셋 출력 신호의 active 신호 level 설정.
	get_amp_reset_level	모터 리셋 출력 신호의 active 신호 level 설정 값을 반환.

SYNTAX	int16 disable_amplifier(int16 net_id, int16 axis);
	int16 enable_amplifier(int16 net_id, int16 axis);
	int16 get_amp_enable(int16 net_id, int16 axis, P_INT state);
	int16 set_amp_enable_level(int16 net_id, int16 axis, int16 level);
	int16 get_amp_enable_level(int16 net_id, int16 axis, P_INT level);
	int16 set_boot_amp_enable(int16 net_id, int16 axis, int16 state);

```
int16 get_boot_amp_enable(int16 net_id, int16 axis, P_INT state);
int16 set_boot_amp_enable_level(int16 net_id, int16 axis, int16 level);
int16 get_boot_amp_enable_level(int16 net_id, int16 axis, P_INT level);

int16 amp_fault_switch(int16 net_id, int16 axis);
int16 set_amp_fault_level(int16 net_id, int16 axis, int16 level);
nt16 get_amp_fault_level(int16 net_id, int16 axis, P_INT level);
int16 set_amp_fault(int16 net_id, int16 axis, int16 action);
int16 get_amp_fault(int16 net_id, int16 axis, P_INT action);

int16 set_negative_limit(int16 net_id, int16 axis, int16 action);
int16 get_negative_limit(int16 net_id, int16 axis, P_INT action);
int16 set_negative_level(int16 net_id, int16 axis, int16 level);
int16 get_negative_level(int16 net_id, int16 axis, P_INT level);
int16 set_positive_limit(int16 net_id, int16 axis, int16 action);
int16 get_positive_limit(int16 net_id, int16 axis, P_INT action);
int16 set_positive_level(int16 net_id, int16 axis, int16 level);
int16 get_positive_level(int16 net_id, int16 axis, P_INT level);

int16 set_home_index_config(int16 net_id, int16 axis, int16 config);
int16 set_home_level(int16 net_id, int16 axis, int16 level);
int16 get_home_level(int16 net_id, int16 axis, P_INT level);
int16 set_home(int16 net_id, int16 axis, int16 action);
int16 get_home(int16 net_id, int16 axis, P_INT action);
int16 home_switch(int16 net_id, int16 axis);
int16 set_home_level_inverse(int16 net_id, int16 axis, int16 inverse);
int16 get_home_level_inverse(int16 net_id, int16 axis, P_INT inverse);

int16 reset_motor(int16 net_id, int16 axis, int16 output);
int16 get_amp_reset_output(int16 net_id, int16 axis, P_INT output);
int16 set_amp_reset_level(int16 net_id, int16 axis, int16 level);
int16 get_amp_reset_level(int16 net_id, int16 axis, P_INT level);
```

PROTOTYPE IN

eNetAxis2-link.h

DESCRIPTION

disable_amplifier(…)는 Servo On 출력 신호를 set_amp_enable_level(…)로 설정한 신호 level 반대로 출력합니다. Step 드라이브 일체형인 경우에는 Step 드라이브를 free run 상태로 변경합니다.

enable_amplifier(…)는 Servo On 출력 신호를 set_amp_enable_level(…)로 설정한 신호 level로 출력합니다. Step 드라이브 일체형인 경우에는 Step 드라이브를 run 상태로 변경합니다.

get_amp_enable(…)는 amp enable 출력 상태를 반환하는 함수 입니다. Amp enable 이면 TRUE를 반환하고, amp disable이면 FALSE를 반환합니다.

set_amp_enable_level(…)은 amp enable 출력 신호의 active level 값을 설정합니다.

get_amp_enable_level(…)은 amp enable 출력 신호의 active level 설정 값을 반환합니다.

set_boot_amp_enable(…)은 boot시 amp enable 상태를 설정합니다.

get_boot_amp_enable(…)은 boot시 amp enable 상태를 설정 값을 반환합니다.

set_boot_amp_enable_level(…)은 boot시 amp enable 출력 신호의 active level 값을 설정합니다.

get_boot_amp_enable_level(…)은 boot시 amp enable 출력 신호의 active level 설정 값을 반환합니다.

set_negative_limit(…)은 – limit 가 감지되었을 때 동작을 설정합니다.

get_negative_limit(…)은 – limit 가 감지되었을 때 동작할 정보를 반환합니다.

#define	Value	Description
NO_EVENT	0	이벤트가 없음.
STOP_EVENT	8	Stop rate로 감속 정지.
E_STOP_EVENT	10	E-stop rate로 감속 정지.
ABORT_EVENT	14	Amp Disable.

set_negative_level(…)은 – limit 입력 신호의 active 신호 level을 설정합니다.

get_negative_level(…)은 – limit 입력 신호의 active 신호 level 설정 값을 반환합니다.

set_positive_limit(…)은 + limit가 감지되었을 때 동작을 설정합니다.

get_positive_limit(…)은 + limit 가 감지되었을 때 동작할 정보를 반환합니다.

#define	Value	Description
NO_EVENT	0	이벤트가 없음.
STOP_EVENT	8	Stop rate로 감속 정지.
E_STOP_EVENT	10	E-stop rate로 감속 정지.
ABORT_EVENT	14	Amp Disable.

set_positive_level(…) + limit 입력 신호의 active 신호 level을 설정합니다.

get_positive_level(…)은 + limit 입력 신호의 active level 설정 값을 반환합니다.

amp_fault_switch(…)은 Servo Ready 입력 신호를 반환합니다.
+24V 입력 시 TRUE를 반환하고, 0V 입력 시 FALSE를 반환합니다.

set_amp_fault_level(…)은 amp fault 입력 신호의 active 신호 level 값을 설정합니다.

get_amp_fault_level(…)은 amp fault 입력 신호의 active 신호 level 설정 값을 반환합니다.

set_amp_fault(…)은 amp fault 입력 신호가 감지 되었을 때 동작을 설정합니다.

#define	Value	Description
NO_EVENT	0	이벤트가 없음.
STOP_EVENT	8	Stop rate로 감속 정지.
E_STOP_EVENT	10	E-stop rate로 감속 정지.
ABORT_EVENT	14	Amp Disable.

get_amp_fault(…)은 amp fault 입력 신호가 감지 되었을 때 동작 설정 값을 반환합니다.

set_home_index_config (…)은 home logic의 구성 요소를 설정합니다. 이 기능은 예약 기능입니다.

#define	Value	Description

HOME_ONLY	0	Home 입력 접점만 감지.
LOW_HOME_AND_INEX	1	Home 입력이 Low이고, index(Z상) 감지.
INDEX_ONLY	2	Index(Z상)만 감지.
HIGH_HOME_AND_INDEX	3	Home 입력이 high이고, index(Z상) 감지.

set_home_leve(…)은 home 입력 접점의 active 상태를 설정합니다.

get_home_level(…)은 home 입력 접점의 active 구성 값을 반환합니다.

Set_home(…)은 home logic 입력이 감지되었을 때 동작을 설정한다. Action 인자를 통해 동작을 설정한다.

#define	Value	Description
NO_EVENT	0	이벤트가 없음.
STOP_EVENT	8	stop rate로 감속 정지.
E_STOP_EVENT	10	E-stop rate로 감속 정지.
ABORT_EVENT	14	Amp Disable.

Get_home(…)은 home logic 입력이 감지되었을 때 동작할 정보를 반환합니다.

home_switch(…)은 home logic 상태를 반환합니다.

set_home_level_inverse(…)은 home logic 신호 level의 설정 값을 반전하여 실행하라는 명령입니다. 이 함수는 Library 내부 플래그를 설정하는 함수로 home_switch(…), set_home_level(…), get_home_level(…)의 값을 반전하여 실행합니다. Library 내부 플래그임으로 Library 가 종료되면 기능이 유지되지 않습니다.

get_home_level_inverse(…)은 home logic 신호 level의 설정 값을 반전하여 실행되는 기능이 설정 되었는지 확인합니다. 이 기능은 Library 내에서 지원하는 기능입니다.

reset_motor(…)은 모터 리셋 신호를 출력합니다.

get_amp_reset_output(…)은 모터 리셋 신호 출력 상태를 반환합니다.

set_amp_reset_level(…)은 모터 리셋 출력 신호의 active 신호 level을 설정합니다..

get_amp_reset_level(…)은 모터 리셋 출력 신호의 active 신호 level 설정 값을 반환합니다.

RETURN VALUES

DSP_OK(0), DSP_NOT_INITIALIZED(1), DSP_NOT_FOUND (2), DSP_INVALID_AXIS(3), DSP_TIMEOUT_ERROR(14), DSP_DISCONNECT(50)

Point-to-Point/Velocity Motion

	start_move_all	2축 사다리꼴 가감속 운전 모드로 동시 이송.
	start_s_move	S자 가감속 운전 모드로 이송
	start_t_move	사다리꼴 가감속 운전 모드로 이송
	v_move	정속도 운전모드로 이송.
	wait_for_all	다중 축 이송이 완료될 때 까지 대기.

SYNTAX	int16 start_move_all(int16 net_id, int16 len, int16 *axis,
	double *position, double *velocity, double *accel);
	int16 start_s_move(int16 net_id, int16 axis, double position, double velocity,
	double acceleration, double jerk);
	int16 start_t_move(int16 net_id, int16 axis, double position, double velocity,
	double acceleration, double decel);
	int16 v_move(int16 net_id, int16 axis, double velocity, double acceleration);
	int16 wait_for_all(int16 net_id, int16 len, int16 *axis);	

PROTOTYPE IN	eNetAxis2-link.h	
DESCRIPTION	start_move_all(…)은 2축을 사다리꼴 가감속 운전 모드를 이용하여 동시에 이송합니다. start_s_move(…)은 S자 가감속 운전 모드를 이용하여 이송합니다. start_t_move(…)은 사다리꼴 가감속 운전 모드를 이용하여 이송합니다. v_move(…)은 정속도 이송합니다. 축은 지정속도로 무한 이송합니다. 이송 정지를 위해서는 '0'속도를 지령하거나, stop 명령을 지령합니다. wait_for_all(…)은 복수 축 이송이 완료될 때 까지 대기합니다.	
RETURN VALUES	DSP_OK(0), DSP_NOT_INITIALIZED(1), DSP_NOT_FOUND (2), DSP_INVALID_AXIS(3), DSP_TIMEOUT_ERROR(14), DSP_DISCONNECT(50)	

Interpolation Motion

	set_move_speed	보간 속도 설정.
	set_move_accel	보간 가속도 설정.
	set_move_ratio	좌표 변형 비율 설정.
	move_2	2축 직선 보간을 point list에 추가.
	arc_2	2축 원호 보간을 point list에 추가.
	set_arc_division	원호 보간의 원주 속도 설정.

SYNTAX	int16 set_move_speed(int16 net_id, double speed);
	int16 set_move_accel(int16 net_id, double acceleration);
	int16 set_move_ratio(int16 net_id, P_DOUBLE ratio);
	int16 move_2(int16 net_id, double x, double y);
	int16 arc_2(int16 net_id, double x_center, double y_center, double angle);
	int16 set_arc_division(int16 net_id, double degrees);	

PROTOTYPE IN	eNetAxis2-link.h	
DESCRIPTION	**set_move_speed(…)**은 보간 속도를 설정합니다. 이 기능은 예약 기능입니다. **set_move_accel(…)**은 보간 가속도를 설정합니다. 이 기능은 예약 기능입니다. **set_move_ratio(…)**은 보간 이송 시 좌표계 변형율을 설정합니다. 이 기능은 예약 기능입니다. **move_2(…)**은 2축 직선 보간을 point list에 추가합니다. 이 기능은 예약 기능입니다. **arc_2(…)**은 2축 원호 보간을 point list에 추가합니다. 이 기능은 예약 기능입니다. **set_arc_division(…)**은 원호 보간 시 원주 속도를 설정합니다. Degree의 단위는 degree/sec 입니다. 이 기능은 예약 기능입니다.	
RETURN VALUES	DSP_OK(0), DSP_NOT_INITIALIZED(1), DSP_NOT_FOUND (2), DSP_INVALID_AXIS(3), DSP_TIMEOUT_ERROR(14), DSP_DISCONNECT(50)	

Step Output Configuration

dsp_set_stepper	Pulse 출력 기능을 설정.
dsp_stepper	Pulse 출력 기능 설정 값을 반환.
dsp_set_step_config	Pulse 출력 모드를 설정.
dsp_step_config	Pulse 출력 모드 설정 값 반환.
dsp_set_step_speed	Pulse 출력 속도 설정.
dsp_step_speed	Pulse 출력 속도 설정 값 반환.

SYNTAX

```
int16 dsp_set_stepper(int16 net_id, int16 axis, int16 stepper);
int16 dsp_stepper(int16 net_id, int16 axis);
int16 dsp_set_step_config(int16 net_id, int16 axis, int16 mode);
int16 dsp_step_config(int16 net_id, int16 axis, P_INT mode);
int16 dsp_set_step_speed(int16 net_id, int16 axis, unsigned16 speed);
int16 dsp_step_speed(int16 net_id, int16 axis, P_UINT speed);
```

PROTOTYPE IN

eNetAxis2-link.h

DESCRIPTION

dsp_set_stepper(…)은 pulse 출력 기능을 설정합니다.

dsp_stepper(…)은 pulse 출력 기능 설정 값을 반환합니다.

dsp_set_step_config(…)는 pulse 출력 모드를 설정합니다.

#define	Value	Description
STEPDIR	0	Pule / direction 방식 출력.
CWCCW	1	CW / CCW 방식 출력.

dsp_step_config(…)은 pulse 출력 모드 설정 값을 반환합니다.

dsp_set_step_speed(…)는 pulse 출력 속도를 설정합니다. 이 기능은 예약 기능입니다.

dsp_step_speed(…)는 pulse 출력 속도 설정 값을 반환합니다. 이 기능은 예약 기능입니다.

RETURN VALUES

DSP_OK(0), DSP_NOT_INITIALIZED(1), DSP_NOT_FOUND (2),
DSP_INVALID_AXIS(3), DSP_TIMEOUT_ERROR(14), DSP_DISCONNECT(50)

Feedback Configuration

	get_dac_output	16비트 DAC 값을 반환합니다.
	disable_loopback	지령 펄스를 Encoder 포트로 반환하는 기능 취소.
	enable_loopback	지령 펄스를 Encoder 포트로 반환하는 기능.
	dsp_get_loopback	Loopback 설정 상태 반환.
	set_feedback	Feedback 장치를 설정.
	get_feedback	Feedback 장치 설정 값 반환.
	set_filter	PID 변수 값들을 설정.
	get_filter	PID 변수 설정 값들을 반환.
	set_integration	I 제어 적용 방법을 설정.
	get_integration	I 제어 적용 방법 설정 값을 반환.

SYNTAX

```
int16 get_dac_output(int16 net_id, int16 axis, P_INT32 frequency);
int16 disable_loopback(int16 net_id, int16 axis);
int16 enable_loopback(int16 net_id, int16 axis);
int16 dsp_get_loopback(int16 net_id, int16 axis, P_INT loopback);
int16 set_feedback(int16 net_id, int16 axis, int16 device);
int16 get_feedback(int16 net_id, int16 axis, P_INT device);
int16 set_filter(int16 net_id, int16 axis, P_INT32 filter);
int16 get_filter(int16 net_id, int16 axis, P_INT32 filter);
int16 set_integration(int16 net_id, int16 axis, int16 mode);
int16 get_integration(int16 net_id, int16 axis, P_INT mode);
```

PROTOTYPE IN

eNetAxis2-link.h

DESCRIPTION

get_dac_output(…)는 16비트 아날로그 출력 값(DAC)을 반환합니다. 이 함수는 예약 기능입니다.

disable_loopback(…)는 외부로 출력되는 이송 지령 펄스를 Encoder 포트로 내부 회귀하는 기능을 취소합니다.
명령 실행 후, Encoder 포트로 입력되는 펄스 수로 실제(Actual) 위치를 계산합니다.

enable_loopback(…)는 외부로 출력되는 이송 지령 펄스를 Encoder 포트로 내부 회귀하는 기능을 실행합니다.
명령 실행 후, 이송 지령 펄스가 Encoder 포트로 입력되어 실제(Actual) 위치를 계산합니다.

dsp_get_loopback(…)은 loopback 기능 설정 상태를 반환합니다. enable되어 있으면 TRUE를 반환합니다.

set_feedback(…)은 feedback 장치를 설정합니다. 이 기능은 예약 기능입니다.

get_feedback(…)은 feedback 장치 설정 값을 반환합니다. 이 기능은 예약 기능입니다.

Set_filter(…)은 PID 제어거 변수 값들을 설정합니다. 이 함수는 예약 기능입니다.

Get_filter(…)은 PID 제어 변수 설정 값들을 반환합니다. 이 함수는 예약 기능입니다.

set_integration(…)은 I 제어 적용 방법을 설정합니다.

#define	Value	Description
IM_STANDING	0	정지 시만 적용.

IM_ALWAYS	1	항상 적용.

get_integration(…)은 I 제어 적용 방법 설정 값을 반환합니다.

RETURN VALUES

DSP_OK(0), DSP_NOT_INITIALIZED(1), DSP_NOT_FOUND (2),
DSP_INVALID_AXIS(3), DSP_TIMEOUT_ERROR(14), DSP_DISCONNECT(50)

In Position

	set_in_position	in-position 검사 범위 값을 설정
	get_in_position	In-position 검사 범위 값을 반환.
SYNTAX	int16 set_in_position(int16 net_id, int16 axis, double limit); int16 get_in_position(int16 net_id, int16 axis, P_DOUBLE limit);	
PROTOTYPE IN	eNetAxis2-link.h	
DESCRIPTION	**set_in_position(…)**은 축에 적용할 in-position 검사 범위를 설정합니다. '0'을 설정하면 in-position 기능이 취소됩니다. **get_in_position(…)**은 축에 적용되고 있는 in-position 검사 범위를 반환합니다. In-position 검사 범위의 단위는 pulse 이고, 검사할 이송 오차 값을 설정합니다.	
RETURN VALUES	DSP_OK(0), DSP_NOT_INITIALIZED(1), DSP_NOT_FOUND (2), DSP_INVALID_AXIS(3), DSP_TIMEOUT_ERROR(14), DSP_DISCONNECT(50)	

Position Capture

set_capture_point	Capture를 위한 외부 신호 설정.
get_capture_point	Capture를 위한 외부 신호 설정 값 반환.
set_capture_encoder	Capture할 실제(actual) 위치 축을 설정.
get_capture_encoder	Capture할 실제(actual) 위치 축 설정 값 반환.
set_capture_mode	Capture 동작 모드 설정.
get_capture_mode	Capture 동작 모드 설정 값 반환.
set_capture_max_count	Capture 할 위치 개수 설정.
get_capture_max_count	Capture 할 위치 개수 설정 값 반환.
get_capture_run_count	Capture 된 위치 개수 값 반환.
start_capture	capture 기능 실행.
reset_capture	capture 기능 취소.
read_capture_data	capture 데이터 읽기.

SYNTAX

```
int16 set_capture_point(int16 net_id, int16 ch, int16 point);
int16 get_capture_point(int16 net_id, int16 ch, P_INT point);
int16 set_capture_encoder(int16 net_id, int16 ch, int16 encoder);
int16 get_capture_encoder(int16 net_id, int16 ch, P_INT encoder);
int16 set_capture_mode(int16 net_id, int16 ch, int16 mode);
int16 get_capture_mode(int16 net_id, int16 ch, P_INT mode);
int16 set_capture_max_count(int16 net_id, int16 ch, int32 max_cnt);
int16 get_capture_max_count(int16 net_id, int16 ch, P_INT32 max_cnt);
int16 get_capture_run_count(int16 net_id, int16 ch, P_INT32 run_count);
int16 start_capture(int16 net_id, int16 ch);
int16 reset_capture(int16 net_id, int16 ch);
int16 read_capture_data(int16 net_id, int16 ch, int16 cap_index, P_DOUBLE rising, P_DOUBLE falling);
```

PROTOTYPE IN

eNetAxis2-link.h

DESCRIPTION

Position capture 기능은 외부 입력 신호(입.출력 접점, Z 상)의 변화를 감지한 실제(actual) 위치를 저장하는 기능입니다. 2개의 채널로 구성되어 있고 2 개의 채널이 한 축을 위해 다 사용될 수 있습니다. 각 채널당 100개의 버퍼가 있어 외부 입력 신호 변화에 대하여 100개의 다른 위치를 저장할 수 있습니다.

set_capture_point(…)은 position capture를 위한 외부 신호를 설정합니다.

#define	Hex	Description
dCapReadyXPoint	0x1	X축 Servo Ready 접점.
dCapPlusLimitXPoint	0x2	X축 + Limit 접점.
dCapMinusLimitXPoint	0x4	X축 - Limit 접점.
dCapHomeXPoint	0x8	X축 Home 접점.
dCapReadyYPoint	0x10	Y축 Servo Ready 접점.
dCapPlusLimitYPoint	0x20	Y축 + Limit 접점.
dCapMinusLimitYPoint	0x40	Y축 - Limit 접점.
dCapHomeYPoint	0x80	Y축 Home 접점
dCapEnc_ZPhaseX	0x100	X축 Encoder Z상
dCapEnc_ZPhaseY	0x200	Y축 Encoder Z상

get_capture_point(…)은 position capture를 위한 외부 신호 설정 값을 반환합니다.

set_capture_encoder(…)은 Position Capture할 실제(actual) 위치 축을 설정합니다.

Value	Description
0	X축 실제(actual) 위치.
1	Y축 실제(actual) 위치.

get_capture_encoder(…)은 Position Capture할 실제(actual) 위치 축 설정값을 반환합니다.

set_capture_mode(…)은 Position Capture하는 방법을 설정합니다.

#define	Value	Description
dCapRisingEvent	1	신호가 '0'->'1'으로 변경 시 위치 capture.
dCapFallingEvent	2	신호가 '1'->'0'으로 변경 시 위치 capture.
dCapPulseEvent	3	신호가 '1'->'0' 그리고 '0'-> '1'로 변경 시 위치 capture.

get_capture_mode(…)은 Position Capture하는 방법 설정 값을 반환합니다.

set_capture_max_count(…)은 Capture 할 위치의 개수를 설정합니다. Position Capture 기능은 설정된 개수 만큼 수행 후 자동으로 종료됩니다. 최대 '100'개의 위치를 capture할 수 있습니다.

get_capture_max_count(…)은 Capture 할 위치의 개수 설정 값을 반환합니다.

get_capture_run_count(…)은 Capture 된 위치 개수 값을 반환합니다.

start_capture(…)은 capture 기능을 시작합니다. set_capture_max_count(…)에 설정된 횟수만큼 위치를 capture 하면 자동 정지됩니다.

reset_capture(…)은 capture 기능을 취소합니다.

read_capture_data(…)은 capture된 실제(actual) 데이터 값을 반환합니다. Cap_index를 통해 반환 할 buffer내에 있는 데이터를 가리킵니다. Rising 그리고 falling을 통해 데이터를 반환합니다.

RETURN VALUES

DSP_OK(0), DSP_NOT_INITIALIZED(1), DSP_NOT_FOUND (2), DSP_INVALID_AXIS(3), DSP_TIMEOUT_ERROR(14), DSP_DISCONNECT(50)

Trigger Output

set_trigger_point	Trigger 출력 접점 설정.
get_trigger_point	Trigger 출력 접점 설정 값 반환.
set_trigger_port	Trigger 기준 encoder 포트 설정.
get_trigger_port	Trigger 기준 encoder 포트 설정 값 반환.
set_trigger_pre_encoder_pulse	Trigger 시작 상대 위치 설정.
get_trigger_pre_encoder_pulse	Trigger 시작 상대 위치 설정 값 반환.
set_trigger_cyc_encoder_pulse	Trigger 주기 거리 값 설정.
get_trigger_cyc_encoder_pulse	Trigger 주기 거리 값 설정 값 반환.
set_trigger_pulse_width	출력 펄스 폭 설정.
get_trigger_pulse_width	출력 펄스 폭 설정 값 반환.
set_trigger_max_pulse	발생할 Pulse 개수를 설정.
get_trigger_max_pulse	발생할 Pulse 개수 설정 값 반환.
get_trigger_pulse	발생된 pulse 개수를 반환.
get_trigger_status	Trigger 설정 및 상태 값 반환.
start_trigger	펄스 열 출력 기능 실행.
reset_trigger	펄스 열 출력 기능 취소.

SYNTAX

```
int16 set_trigger_point(int16 net_id, int16 ch, int16 point);
int16 get_trigger_point(int16 net_id, int16 ch, P_INT point);
int16 set_trigger_port(int16 net_id, int16 ch, int16 port);
int16 get_trigger_port(int16 net_id, int16 ch, P_INT port);
int16 set_trigger_pre_encoder_pulse(int16 net_id, int16 ch, int32 pulse);
int16 get_trigger_pre_encoder_pulse(int16 net_id, int16 ch, P_INT32 pulse);
int16 set_trigger_cyc_encoder_pulse(int16 net_id, int16 ch, INT32 pulse);
int16 get_trigger_cyc_encoder_pulse(int16 net_id, int16 ch, P_INT32 pulse);
int16 set_trigger_pulse_width(int16 net_id, int16 ch, INT32 time);
int16 get_trigger_pulse_width(int16 net_id, int16 ch, P_INT32 time);
int16 set_trigger_max_pulse(int16 net_id, int16 ch, INT32 count);
int16 get_trigger_max_pulse(int16 net_id, int16 ch, P_INT32 count);
int16 get_trigger_pulse(int16 net_id, int16 ch, P_INT32 count);
int16 get_trigger_status(int16 net_id, int16 ch, P_INT32 status);
int16 start_trigger(int16 net_id, int16 ch);
int16 reset_trigger(int16 net_id, int16 ch);
```

PROTOTYPE IN

eNetAxis2-link.h

DESCRIPTION

지정된 실제(actual) 위치부터 출력 포트를 통해 pulse를 발생하는 기능입니다. 2 개의 채널로 구성되어 있고 2 개의 채널이 한 축을 위해 다 사용될 수 있습니다.

set_trigger_point(…)은 펄스 열을 발생할 출력 접점을 설정합니다.

#define	Hex	Description
dTriggerXOn	0x1	X축 Servo On.
dTriggerYOn	0x2	Y축 Servo On.
dTriggerXReset	0x4	X축 Servo Reset.
dTriggerYReset	0x8	Y축 Servo Reset.

get_trigger_point(…)은 펄스 열을 발생하기 위채 설정한 출력 접점 값을 반환합니다.

set_trigger_port(…)은 trigger를 위해 사용할 축을 설정합니다. Trigger 기능은 설정된 축의 실제(actual) 위치에 기준하여 동작됩니다.

Value	Description

0	X축 실제(actual) 위치.
1	Y축 실제(actual) 위치.

get_trigger_port(…)은 trigger를 위해 사용할 축 설정 값을 반환합니다

set_trigger_pre_encoder_pulse(…)은 현재 위치에서 펄스 열을 발생할 상대 위치 펄스 열 값을 설정합니다.

get_trigger_pre_encoder_pulse(…)은 현재 위치에서 펄스 열을 발생할 상대 위치 펄스 열 설정 값을 반환합니다.

set_trigger_cyc_encoder_pulse(…)은 주기적으로 펄스 열을 발생할 실제(actual) 거리 값을 설정합니다.

get_trigger_cyc_encoder_pulse(…)은 주기적으로 펄스 열을 발생할 실제(actual) 거리 설정 값을 반환합니다.

set_trigger_pulse_width(…)은 출력 펄스 폭을 시간(msec) 기준으로 설정합니다.

get_trigger_pulse_width(…)은 출력 펄스 폭을 시간(msec) 기준으로 설정한값을 반환합니다.

set_trigger_max_pulse(…)은 발생할 pulse의 개수를 설정합니다.

get_trigger_max_pulse(…)은 발생할 pulse 개수 설정 값을 반환합니다.

get_trigger_pulse(…)은 발생된 pulse 개수를 반환합니다.

get_trigger_status(…)은 Trigger 상태 및 설정 값을 반환합니다. 이 기능은 개발자용입니다.

start_trigger(…)은 펄스 열 출력을 위한 trigger 기능을 실행합니다.

reset_trigger(…)은 trigger 기능을 취소합니다.

RETURN VALUES

DSP_OK(0), DSP_NOT_INITIALIZED(1), DSP_NOT_FOUND (2), DSP_INVALID_AXIS(3), DSP_TIMEOUT_ERROR(14), DSP_DISCONNECT(50)

Input Digital Filter

	set_digital_input_filter 입력 접점 응답성 설정. get_digital_input_filter 입력 접점 응답성 설정 값 반환.
SYNTAX	int16 set_digital_input_filter(int16 net_id, int16 ch, int16 filter); int16 get_digital_input_filter(int16 net_id, int16 ch, P_INT filter);
PROTOTYPE IN	eNetAxis2-link.h
DESCRIPTION	입력 접점의 응답성을 설정하는 기능입니다. 2개의 입력 접점이 하나의 group으로 구성되어 응답성을 설정할 수 있습니다. set_digital_input_filter(…)은 입력 접점 응답성을 설정합니다. 1 usec ~ 65536 usec 시간을 설정할 수 있습니다. get_digital_input_filter(…)은 입력 접점 응답성 설정 값을 반환합니다.
RETURN VALUES	DSP_OK(0), DSP_NOT_INITIALIZED(1), DSP_NOT_FOUND (2), DSP_INVALID_AXIS(3), DSP_TIMEOUT_ERROR(14), DSP_DISCONNECT(50)

Configure Axis Parameters

	config_axis	struct tagAXISCONFIG 에 정의된 파라미터를 설정.

SYNTAX

int16 config_axis(int16 net_id, struct tagAXISCONFIG *pcfg);

PROTOTYPE IN

eNetAxis2-link.h

DESCRIPTION

config_axis (…)은 struct tagAXISCONFIG 에 정의된 모든 파라미터를 설정합니다.

```
struct tagAXISCONFIG
{
    //General parameters
    double StopRate;
    double EStopRate;
    double Inpostion;
    //Software limit config
    double ErrSWLim;
    double NegSWLim;
    double PosSWLim;
    short ErrSWLimAct;
    short NegSWLimAct;
    short PosSWLimAct;

    short AxisID;
    short StepMotor;
    short ClosedLoop;
    short StepSpeed;
    short Unipolar;
    short Feedback;
    short IntMode;        //Integration mode (always or standing)
    short AmpLevelOnBoot;
    long Filter[10]; //PID gain

    // IO 트리거시 레벨 설정
    // Hardware limit config (Switch)
    short NegSwt;          //Level
    short PosSwt;          //Level
    short HomeSwt;         //Level
    short AmpFaultSwt;   //Level


    // IO 트리거시 동작 설정
    short NegSwtAct;      //Action
    short PosSwtAct;      //Action
    short HomeSwtAct;     //Action
    short AmpFaultSwtAct; //Action
};
```

RETURN VALUES

DSP_OK(0), DSP_NOT_INITIALIZED(1), DSP_NOT_FOUND (2), DSP_INVALID_AXIS(3), DSP_TIMEOUT_ERROR(14), DSP_DISCONNECT(50)

Point List Management

	start_point_list	Point list를 초기화.
	end_point_list	Point list 작성을 종료.
	get_last_point	Point list 운전시 마지막 축 위치들을 반환.
	start_motion	Point list로 자동 운전 시작.
	stop_motion	자동 운전 정지.
	set_point	자동 운전 시작할 point 개수를 설정.

SYNTAX	int16 start_point_list(int16 net_id);
	int16 end_point_list(int16 net_id);
	void get_last_point(int16 net_id, P_DOUBLE x);
	int16 start_motion(int16 net_id);
	int16 stop_motion(int16 net_id);
	int16 set_point(int16 net_id, int16 points);	

PROTOTYPE IN	eNetAxis2-link.h	
DESCRIPTION	**start_point_list(…)**는 새로운 point list를 만들기 위해 제어기의 point list 버퍼를 초기화합니다. 이 함수는 예약 기능입니다. **end_point_list(…)**는 point list 작성을 종료합니다. Point list 운전시 end_point_list(…)가 실행된 point 내용까지 실행합니다. 이 함수는 예약 기능입니다. **get_last_point(…)**는 point list에 등록된 마지막 정보에 의해 축이 이송될 위치 값들을 반환합니다. 이 함수는 예약 기능입니다. **start_motion(…)**은 point list에 등록된 정보의 순서대로 자동 운전을 시작합니다. 이 함수는 예약 기능입니다. **stop_motion(…)**은 point list 자동 운전을 정지합니다. 이 함수는 예약 기능입니다. **Set_point(…)**는 자동 운전을 시작할 point 개수를 설정합니다. Point 개수를 설정한 후 point list에 등록된 point 개수가 설정 값과 동일하면 자동 운전을 시작합니다.	
RETURN VALUES	DSP_OK(0), DSP_NOT_INITIALIZED(1), DSP_NOT_FOUND (2), DSP_TIMEOUT_ERROR(14), DSP_DISCONNECT(50)	