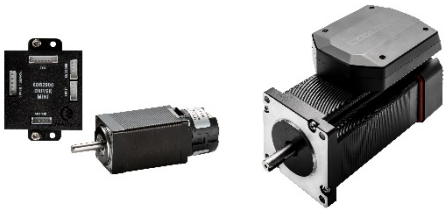
EDB 시리즈 펌웨어 매뉴얼



단축 스텝모터 드라이브/컨트롤러

Full Closed Loop

RS485 시리얼 통신

ERAETECH

1. EMCL과 EMCL-IDE의 개요

EDB시리즈의 소프트웨어는 마이크로 프로세서에서 작동하는 부트로더 (Boot Loader)와 펌웨어 두 부분으로 구성됩니다. 부트로더는 제조사가 입력한 이후 그대로 남아 있는 반면 펌웨어는 유저에 의해 언제든 업데이트 될 수 있습니다.

EDB 시리즈는 EMCL Direct 모드와 Standalone 기능의 EMCL 프로그램을 지원합니다. 최대 2048개의 명령어를 저장할 수 있습니다. Direct 모드와 대부분의 경우 RS485 통신으로 마스터/슬레이브 통신 관계를 따릅니다. 인터페이스 버스 마스터로 Host 컴퓨터 (예: PC/PLC)는 EDB시리즈로 명령을 보냅니다. EMCL은 이 명령을 해석하고 모션 컨트롤러를 시작, 그리고 입력을 읽고 출력을 쓰거나 특정된 명령에 따라 필요한 것들을 행합니다. 이 스텝이 끝난 후 모듈은 버스 마스터로 RS485를 통해 응답을 합니다. 이 후 마스터는 다음 명령을 전송합니다. 일반적으로 모듈은 전송모드로 바뀌고 버스에 응답을 실행합니다. 그렇지 않은 경우에는 수신모드로 남아 있습니다. 명령을 받지 않는다면 인터페이스를 통해 어떠한 데이터도 전송하지 않습니다. 이 방법에서 한 개의 버스에 둘 이상의 Node가 연결 되었을 시 어떠한 충돌도 발생하지 않습니다.

이레텍은 모션컨트롤 언어 "EMCL" 로 구성된 모션 컨트롤 명령어 세트를 제공합니다. 모든 모션제어 명령은 모듈에서 Standalone 모드를 작동시키는 프로그램을 작성하기 위해 호스트 컴퓨터에서 EMCL 연상기호로 작성 컴파일하여 HEX파일을 EMCL에서 EEPROM에 저장 시킬 수 있습니다. 이는 모션 제어 명령 뿐 아니라 프로그램 구조를 제어하는 명령 (파라미터) 도 가능하게 합니다 (조건부 점프, 비교/계산 등).

모든 명령은 이진 구조이며 어셈블리 형식으로 사용합니다. 어셈블리 형식으로 EMCL-IDE를 사용하는 Standalone 모드에서 손쉬운 사용을 위해 사용되는 반면 이진은 Direct 모드에서 호스트로부터 모듈에 명령을 보내기 위해 사용됩니다.

각각의 환경설정을 허용하는 Global 파라미터와 각 축을 위한 환경설정 변수 또한 사용이 가능합니다.

본 펌웨어 매뉴얼은 EDB-EXCEL/COMPACT/ALL/MINI 유저를 위한

펌웨어 매뉴얼입니다.

2. 명령의 형태

2.1 이진 명령 형태

모든 명령은 어셈블리 기호로 사용되고 이진 형태를 가집니다. 명령이 호스트에서 모듈로 전송될 때 이진 형태를 사용해야 합니다. 모든 명령은 1바이트 명령필드, 1바이트 타입필드, 1바이트 모터/뱅크 필드, 그리고 4바이트의 값 필드로 구성되어 있습니다. 따라서 명령의 이진 형태는 항상 7바이트를 가집니다. 명령이 RS485를 통해 전송 시 시작부분은 주소 바이트, 끝 부분은 체크섬 바이트로 둘러싸여 있어야 합니다. 이러한 경우 프로토콜은 총 9바이트로 구성됩니다.

바이트	설명
1	주소 (Address)
1	명령 넘버 (Instruction)
1	타입 넘버 (Type)
1	모터 혹은 뱅크 (Motor/Bank)
4	값 (Value :MSB 먼저)
1	체크섬 (Check-sum) 내부에서 계산되어 자동추가됨

표 2. 1 이진명령 형태 프로토콜

체크섬은 모든 바이트들을 덧셈한 후 하위 1byte 만을 취하여 명령어 열 끝에 추가 됨.

2.1.1 체크섬 계산

앞서 말한 것처럼 체크섬 (Check-Sum)은 8비트를 사용하는 모든 바이트 (주소 바이트를 포함)들을 더함으로써 계산됩니다. 이 방법에 대한 두 가지 예시입니다.

- in C:

```
unsigned char i, Checksum, Command[9];
//Set the "Command" array to the desired command
Checksum = Command[0]; for(i=1; i<8; i++)
Checksum+=Command[i];
  Command[8]=Checksum; //insert checksum as last byte of the command //Now, send it to
  the module
```

- in Delphi: var

```
i, Checksum: byte;
Command: array[0..8] of byte; //Set the "Command" array to the desired command
Checksum:=Command[0]; //Calculate the Checksum:
for i:=1 to 7 do Checksum:=Checksum+Command[i]; Command[8]:=Checksum;
//Now, send the "Command" array (9 bytes) to the module
```

2.2 형태

명령이 모듈로 전송될 때 모듈은 응답을 보냅니다. 응답 형태는 아래와 같습니다.

바이트	설명
1	응답 주소
1	모듈 주소
1	상태 코드 (100은 에러 없음을 의미)
1	명령 넘버
4	값 (Value :MSB 먼저)
1	체크섬 (Check-sum)

표 2. 2 응답형태의 프로토콜

그림

그림 2. 1 EMCL에서의 응답 형태

중요: 체크섬은 8 바이트를 사용하는 모든 바이트들을 더함으로 계산됩니다. 응답 전 다음명령을 전송하지 마십시오!

2.3 상태코드

응답은 상태 코드를 포함합니다. 상태 코드는 아래의 값 중 하나를 가질 수 있습니다.

코드	설명
100	성공적으로 실행, 에러 없음
101	EMCL 프로그램 EEPROM으로 명령 로드
1	틀린 체크 섬
2	유효하지 않은 명령
3	틀린 타입
4	유효하지 않은 값
5	잠겨진 환경 설정 EEPROM
6	불가능한 명령

표 2. 3 상태코드

그림

2.4 Stand-Alone 어플리케이션

모듈에는 EMCL 어플리케이션의 저장을 위해 EMCL 메모리를 장착하고 있습니다. 프로그램을 EEPROM에 다운로드 할 수 있고 이를 실행 할 수 있습니다. EMCL-IDE는 에디터와 어셈블리로 입력 가능한 EMCL 어셈블러를 포함합니다. 이는 자동적으로 이진법 형태로 모아집니다. 후에 이 코드는 실행을 위해 모듈로 다운로드 할 수 있습니다.

2.5 모터 제어 프로그램 DLL.

① Define 파라미터

```
#define EMC_REPLY_OK 0
    ; reply가 정상 상태
#define EMC_REPLY_NOT_READY 1
    ; reply 데이터들이 모두 수신되지 않는 상태
#define EMC_REPLY_CHECKSUM_ERROR 2
    ; reply 데이터의 check-sum 오류가 발생된 상태
#define EMC_REPLY_VERSION 3
    ; reply 데이터가 버전 정보일 때
#define EMC_REPLY_TIMEOUT 4
    ; reply 함수 호출 타임을 초과할 때
```

② Typedef

- 호스트 (PC)에서 모터로 전송되는 데이터 형태

바이트	설명
1	주소 (Address)
1	명령 넘버 (Instruction)
1	타입 넘버 (Type)
1	모터 혹은 뱅크 (Motor/Bank)
4	값 (MSB 먼저)
1	체크섬 (Check-sum)

표 2.4 이진 명령 형태

- 호스트 (PC)가 모터로부터 수신되는 데이터 형태

바이트	설명
1	응답 주소
1	모듈 주소
1	상태 코드 (100은 에러 없음을 의미)
1	명령 넘버
4	값 (Value :MSB 먼저)
1	체크섬 (Check-sum)

표 2.5 응답 형태

1) 기본 구조로 9 바이트 구성

```
typedef struct _typeBASE
{
    unsigned char Command;
    unsigned char Type;
    unsigned char Bank;
    int           Value;
}type_BASE, *ptype_BASE;
```

2) 커맨드가 없는 8 바이트 구성

```
typedef struct _typeSHORT
{
    unsigned char Type;
    unsigned char Bank;
    int           Value;
}type_SHORT, *ptype_SHORT;
```

3) BYTE 타입으로 구성 : Firmware 버전에 사용

```
typedef struct _typeONEBYTE
{
    unsigned char Command;
    unsigned char Type;
    unsigned char Bank;
    unsigned char Byte0;
    unsigned char Byte1;
    unsigned char Byte2;
    unsigned char Byte3;
}type_ONEBYTE, *ptype_ONEBYTE;
```

③ 함수 설명

OpenRS232(LPCTSTR ComName, DWORD BaudRate);

; RS232/RS485 통신포트 열기 함수

예) 핸들값 = OpenRS232(TEXT("COM2"), 9600); // 포트는 COM2이고 9600bps 일때

CloseRS232(HANDLE Handle);

; RS232/RS485 통신포트 닫기 함수

예) 핸들값으로 통신포트 닫기

CmdTX(HANDLE Handle, unsigned char MotorAddress, type_BASE tyBCF);

; 핸들값, 모터 serial address, 구조체(type_BASE) 데이터를 마스터(PC)에서 슬레이브(모터) 전송하는 함수

CmdTxA(HANDLE Handle, unsigned char uchAddress, unsigned char uchCommand, unsigned char uchType, unsigned char uchBank, int iValue, unsigned char *TxBuffer);

; 핸들값, 모터 serial address, 모터Command, 모터type, 모터bank, 모터 설정 값을 마스터(PC)에서 슬레이브(모터) 전송하는 함수

(Command는 데이터 시트 참조)

CmdTxB(HANDLE Handle, unsigned char uchAddress, type_BASE tyBASE, unsigned char *TxBuffer);

; 핸들값, 모터serial address, 구조체(type_BASE) 데이터, 마스터(PC)에서 슬레이브 (모터) 전송하는 함수

GetReplyA(HANDLE Handle, unsigned char *uchReplyAddr, unsigned char *uchStatus, int *iValue, unsigned char *RxBuffer, BOOL bVersion);

; 핸들값, 모터 reply address, status, 수신받을데이터, 수신데이터버퍼, 슬레이브(모터)로부터 마스터(PC)가 수신하는 함수

(여기서 "status"는 그림3과 같습니다.)

CmdReplyA(HANDLE Handle, unsigned char *uchReplyAddr, unsigned char *uchStatus, int *iValue, unsigned char *RxBuffer, BOOL bVersion,
int iTimeLimit, unsigned char *uchRETURN);

; 핸들값, 모터 reply address, status, 수신 받을 데이터, 수신데이터버퍼, 버전 정보 데이터를 구분, TimeOut에 대한 설정치[단위ms], 수신 상태 정보를 슬레이브(모터) 로부터 마스터(PC)가 수신하는 함수

코드	설명
100	성공적으로 실행, 에러 없음
101	EMCL 프로그램 EEPROM으로 명령 로드
1	틀린 체크 섬
2	유효하지 않은 명령
3	틀린 타입
4	유효하지 않은 값
5	잠겨진 환경 설정 EEPROM
6	불가능한 명령

표 2.6 상태코드

DLL 및 예제 소스는 이레텍 공식 웹 사이트에서 다운로드 하실 수 있습니다.

3. EMCL 명령어

명령어	숫자	파라미터	설명
ROR	1	<motor number>, <velocity>	특정 속도로 우측 회전
ROL	2	<motor number>, <velocity>	특정 속도로 좌측 회전
MST	3	<motor number>	모터 스톱
MVP	4	ABS|REL|COORD, <motor number>, <position|offset>	위치로 이동 (절대값 혹은 상대값)
SAP	5	<parameter>, <motor number>, <값>	축 파라미터 설정 (모터 제어 특정 세팅)
GAP	6	<parameter>, <motor number>	축 파라미터 읽기 (모션 제어 특정 세팅 읽기)
SGP	9	<parameter>, <bank number>, 값	글로벌 파라미터 세팅 (모듈 특정 세팅) 예) 통신 세팅이나 유저 변수등
GGP	10	<parameter>, <bank number>	글로벌 파라미터 읽기 (모듈 특정 세팅 읽기) 예) 통신 세팅이나 유저 변수등
SIO	14	<port number>, <bank number>, <값>	특정 값으로 디지털 출력 세팅
GIO	15	<port number>, <bank number>	아날로그/디지털 입력 값 읽기
CALC	19	<operation>, <값>	진행 누적 & 값
COMP	20	<값>	누적 < >값 비교
JC	21	<condition>, <jump address>	조건부 점프
JA	22	<jump address>	절대적 점프
CSUB	23	<subroutine address>	서브루틴 불러오기
RSUB	24		서브루틴에서 리턴
WAIT	27	<condition>, <motor number>, <ticks>	외의 프로그램 실행 기다리기
STOP	28		프로그램 실행 멈춤
SCO	30	<coordinate number>, <motor number>, <position>	좌표 세팅
GCO	31	<coordinate number>, <motor number>	좌표 읽기
CCO	32	<coordinate number>, <motor number>	좌표 캡쳐하기
CALCX	33	<operation>	누적 변구 & X-레지스터 진행
AAP	34	<parameter>, <motor number>	축 파라미터 누적 변수
AGP	35	<parameter>, <bank number>	글로벌 파라미터 누적 변수
ACO	39	<coordinate number>, <motor number>	좌표 누적 변수

표 3.1 EMCL 명령어

3.1 제어 서브젝트에 따른 명령 리스트

3.1.1 모션 명령

이러한 명령들은 모터의 모션을 제어합니다. 가장 중요한 명령이고 Direct 모드와 Stand-Alone 모드에서 실행 가능합니다.

연상 기호	명령 넘버	설명
ROL	2	왼쪽으로 이동
ROR	1	오른쪽으로 이동
MVP	4	특정 위치로 이동
MST	3	모터 정지
RFS	13	리퍼런스 서치
SCO	30	좌표 저장
CCO	32	좌표 캡쳐
GCO	31	좌표 얻기

표 3.2 모션 명령 파라미터

3.1.2 파라미터 명령

이러한 명령들은 축 파라미터 혹은 글로벌 파라미터들을 세팅하거나 읽기, 저장하기 등에 사용됩니다. 글로벌 파라미터가 모듈을 제어하는 반면 축 파라터는 각 축마다 독립적으로 세팅할 수 있습니다. 이러한 명령들은 Direct 모드나 Stand-Alone 모드에서 사용됩니다.

연상 기호	명령 넘버	설명
SAP	5	축 파라미터 세팅
GAP	6	축 파라미터 읽기
SGP	9	글로벌 파라미터 세팅
GGP	10	글로벌 파라미터 읽기

표 3.3 각 축 명령 파라미터

3.1.3 제어 명령

프로그램의 흐름 (루프, 조건, 점프 등)을 제어하기 위해 사용됩니다. (Direct 모드에서 사용 불가)

연상 기호	명령 넘버	설명
JA	22	항상 점프
JC	21	조건부 점프
COMP	20	누적변수와 비교
CSUB	23	서브루틴 불러오기
RSUB	24	서브루틴에서 리턴
WAIT	27	특정 이벤트 대기
STOP	28	EMCL 프로그램 종료

표 3.4 제어 명령 파라미터

3.1.4 I/O 포트 명령

외부 I/O 포트를 제어하고 Direct 모드와 Stand-Alone 모드에서 사용 가능합니다.

연상 기호	명령 넘버	설명
SIO	14	출력 설정
GIO	15	입력 읽기

표 3.5 I/O 명령 파라미터

3.1.5 계산 명령

EMCL 어플리케이션 안에서 계산 목적을 위해 사용됩니다.

연상 기호	명령 넘버	설명
CALC	19	누적 변수와 정수 값을 사용해 계산하기
CALCX	33	누적 변수와 X 레지스터를 사용해 계산하기
AAP	34	누적 변수를 축 파라미터로 복사하기
AGP	35	누적 변수를 글로벌 파라미터로 복사하기
ACO	39	누적 변수를 좌표로 복사하기

표 3.6 계산 명령 파라미터

4. 명령

ROR (우측으로 회전)

이 명령으로 모터는 특정 속도로 우측 방향 회전합니다 (위치 카운터 증가). 속도는 초당 펄스 (PPS) 단위로 주어집니다.

내부 기능: 첫째로 속도 모드가 선택됩니다. 그리고 속도 값이 축 파라미터 #0으로 옮겨집니다 (타겟 속도)
-16777216pps 부터 +16777216pps 사이의 속도 선택을 가능하게 합니다.

연관된 명령어 : ROL, MST, SAP, GAP

연상기호: ROR 0, <velocity> (pps 단위로 움직입니다)

이진법 표현:

지시 넘버	타입	모터/뱅크	값
1	(관계 없음)	0*	-16777216pps…… +16777216pps

***하나의 모터일 때 모터 넘버는 항상 0입니다.**

Direct 모드에서 응답:

상태	값
100 - OK	(관계 없음)

예시:

속도 값 10,000pps로 오른쪽으로 회전하라.
연상기호 : ROR 0, 10000

이진법:

Byte Index	0	1	2	3	4	5	6	7
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$01	$00	$00	$00	$00	$27	$10

ROL (좌측으로 회전)

이 명령으로 모터는 특정 속도로 좌측 방향 회전합니다 (ROR의 반대 방향, 위치 카운터 증가). 속도는 초당 펄스 (PPS) 단위로 주어집니다.

내부 기능: 첫째로 속도 모드가 선택됩니다. 그리고 속도 값이 축 파라미터 #0으로 옮겨집니다 (타겟 속도)
-16777216pps 부터 +16777216pps 사이의 속도 선택을 가능하게 합니다.

연관된 명령어 : ROR, MST, SAP, GAP

연상기호: ROL 0, <velocity> (pps 단위로 움직입니다)

이진법 표현:

지시 넘버	타입	모터/뱅크	값
2	(관계 없음)	0*	-16777216pps…… +16777216pps

***하나의 모터일 때 모터 넘버는 항상 0입니다.**

Direct 모드에서 응답:

상태	값
100 - OK	(관계 없음)

예시:

속도 값 20,000pps로 왼쪽으로 회전하라.
연상기호 : ROL 0, 20000

이진법:

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$02	$00	$00	$00	$00	$4E	$20

MST (모터 정지)

이 명령으로 모터는 부드럽게 정지하도록 지시 됩니다..

내부 기능: 축 파라미터 타겟 속도는 0으로 세팅됩니다.

연관된 명령어 : ROL, ROR, SAP, GAP

연상기호: MST 0

이진법 표현:

지시 넘버	타입	모터/뱅크	값
3	(관계 없음)	0*	(관계 없음)

***하나의 모터일 때 모터 넘버는 항상 0입니다.**

Direct 모드에서 응답:

상태	값
100 - OK	(관계 없음)

예시:

정지하라.

연상 기호: MST 0

이진법:

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte 3	Operand Byte 2	Operand Byte 1	Operand Byte 0
(hex)	$01	$03	$00	$00	$00	$00	$00	$00

MVP (특정 위치로 이동)

이 명령으로 모터는 특정한 상대적 혹은 절대적 위치로 이동하게 됩니다. 유닛에 프로그램 된 가/감속 램프와 위치 속도가 사용됩니다. 이 명령은 명령 해석과 모션 컨트롤러의 초기화 후 응답이 즉시 전송되는 논-블로킹 (Non-blocking) 이후의 명령은 모터가 마지막 위치에 도달하는 것을 기다리지 않고 진행됩니다. 최대 속도와 가속은 축 파라미터 #4와 #5에 의해 정의 됩니다.

MVP 명령의 범위는 사인된 32비트입니다 (-2.147.483.648 ... +2.147.483.647). 포지셔닝은 MST, ROL 혹은 ROR을 사용할 때 인터럽트 될 수 있습니다.

세가지 작동 타입이 활용 가능합니다:

- 범위 -2.147.483.648 ... +2.147.483.647 (-231 ... 231-1) 사이의 절대 위치로 이동
- Offset 의 도움으로 실제 위치로 상대적 이동을 시작, 이 경우 새로운 결과 위치 값이 언급된 제한을 초과해서는 안됩니다.
- 모터를 좌표 (SCO를 참조) 로 이동 (전에 저장되었던 좌표)

주의: 실제 위치와 새로운 위치 사이의 거리는 2.147.483.647 (231-1)을 초과해서는 안됩니다. 그렇지 않다면 모터는 짧은 거리를 택하기 위해 반대 방향으로 작동합니다.

내부 기능: 새로운 위치 값은 축 파라미터 #2 타겟 포지션으로 옮겨집니다.
연관된 명령어 : SAP, GAP, SCO, CCO, GCO, MST
연상기호: MVP <ABS/REL/COORD>, 0, <Position/Offset/Coordinate number>

이진법 표현:

지시 넘버	타입	모터/뱅크	값
4	0 ABS – 절대적	0*	<Position>
	1 REL – 상대적	0	<Offset>
	2 COORD – 좌표	0	<Coordinate number> (0 ... 20)

***하나의 모터일 때 모터 넘버는 항상 0입니다.**

Direct 모드에서 응답:

상태	값
100 - OK	(관계 없음)

예시:

(절대) 위치 90,000으로 이동
연상기호: MVP ABS, 0, 90000

이진법:

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$04	$00	$00	$00	$01	$5F	$90

예시:

현재 위치에서 10,000 마이크로 스텝 뒤로 (상대적 10,000) 모터를 움직이십시오.

연상기호: MVP REL, 0, -10000

이진법:

Byte Index	0	1	2	3	4	5	6	7
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$04	$01	$00	$ff	$ff	$d8	$f0

예시:

전에 저장된 자료 #8로 모터를 움직이십시오.

연상기호: MVP COORD, 0, 8

이진법:

Byte Index	0	1	2	3	4	5	6	7
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$04	$02	$00	$00	$00	$00	$08

좌표로 이동시 좌표는 SCO, CCO, 혹은 ACO 명령의 도움과 함께 반드시 미리 세팅 되어야 합니다.

SAP (축 파라미터 세팅)

이 명령으로 모듈의 모든 파라미터를 세팅 할 수 있습니다. 이 세팅은 SRAM에 저장 되며, 휘발성입니다 (파워가 꺼진 후 정보를 잃음을 의미합니다).

본 명령과 함께 사용될 수 있는 파라미터와 값을 가진 표는 축 파라미터 리스트를 참고하시기 바랍니다.

내부 기능: 파라미터 형태는 따라오는 0을 무시하도록 변환됩니다 (혹은 마이너스 값). 이 파라미터는 적절한 장치에서 정확한 위치로 옮겨집니다.

연관된 명령어 : GAP, AAP

연상기호: SAP <parameter number>, 0, <값>

이진법 표현:

지시 넘버	타입	모터/뱅크	값
5	<parameter number>	0*	< >

***하나의 모터일 때 모터 넘버는 항상 0입니다.**

Direct 모드에서 응답:

상태	값
100 - OK	(관계 없음)

예시:

모터의 최대 전류를 176으로 세팅하십시오.

연상기호 : SAP 6, 0, 176

이진법:

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$05	$06	$00	$00	$00	$00	$b0

GAP (축 파라미터 읽기)

EDB 시리즈의 대부분의 파라미터들은 축에 대하여 개별적 조정이 가능합니다. 이 파라미터를 사용하여 읽기가 가능합니다. Stand-Alone 모드에서 요청 값은 또한 추가적 프로세싱의 목적을 위하여 누적 변수 레지스터로 전달이 됩니다 (조건부 점프 같은 경우). Direct 모드에서 읽혀진 값은 reply의 값 필드에서만 출력됩니다 (누적변수에 영향 없음).

본 명령과 함께 사용될 수 있는 파라미터와 값을 가진 표는 축 파라미터 리스트를 참고하시기 바랍니다.

내부 기능: 파라미터 형태는 따라오는 0을 무시하도록 변환됩니다 (혹은 마이너스 값). 이 파라미터는 적절한 장치에서 정확한 위치로 옮겨집니다.

연관된 명령어 : SAP, AAP

연상기호: GAP <parameter number>, 0

이진법 표현:

지시 넘버	타입	모터/뱅크	값
6	<parameter number>	0*	(관계 없음)

***하나의 모터일 때 모터 넘버는 항상 0입니다.**

Direct 모드에서 응답:

상태	값
100 - OK	(관계 없음)

예시:

모터의 실제 위치를 읽어 오십시오.

연상기호 : GAP 1, 0

이진법:

Byte Index	0	1	2	3	4	5	6	7
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$06	$01	$00	$00	$00	$00	$00

응답:

Byte Index	0	1	2	3	4	5	6	7
기능	호스트 주소	타겟 주소	상태	지시 넘버	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$02	$01	$64	$06	$00	$00	$27	$10

상태 = 에러 없음, 위치 = 10,000

SGP (글로벌 파라미터 세팅)

이 명령으로 모션 컨트롤에 직접적인 연관이 없는 모듈 특정 파라미터들의 대부분을 특정화 할 수 있고 EMCL 유저 변수도 바꿀 수 있습니다. 글로벌 파라미터는 호스트 인터페이스, 주변부, 혹은 어플리케이션 특정 변수와 연관이 있습니다. 이러한 파라미터의 서로 다른 그룹들은 추후 제품의 더 큰 최종 넘버를 허용하기 위해 뱅크에 조직화 되어 있습니다. 현재는 오직 뱅크 0과 1이 글로벌 파라미터를 위해 사용되고 뱅크 2는 유저 변수를 위해 사용됩니다. 뱅크 3은 환경 설정의 인터럽트를 위해 사용됩니다.

모든 모듈의 세팅은 자동적으로 비 휘발성으로 프로세서의 내부 EEPROM에 저장 됩니다.

본 명령과 함께 사용될 수 있는 파라미터와 값을 가진 표는 글로벌 파라미터 리스트를 참고하시기 바랍니다.

내부 기능: 파라미터 형태는 따라오는 0을 무시하도록 변환됩니다 (혹은 마이너스 값). 이 파라미터는 적절한 장치에서 정확한 위치로 옮겨집니다.

연관된 명령어 : GGP, AGP

연상기호: SGP <parameter number>, <bank number>, <값>

이진법 표현:

지시 넘버	타입	모터/뱅크	값
9	<parameter number>	<bank number>	< >

Direct 모드에서 응답:

상태	값
100 - OK	(관계 없음)

예시:

시리얼 주소를 3으로 세팅하십시오.

연상기호 : SGP 66, 0, 3

이진법:

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$09	$42	$00	$00	$00	$00	$03

GGP (글로벌 파라미터 읽기)

모든 글로벌 파라미터는 이 기능을 통해 읽을 수 있습니다. 글로벌 파라미터는 호스트 인터페이스, 주변부, 혹은 어플리케이션 특정 변수와 관련 있습니다. 이러한 파라미터들의 서로 다른 그룹들은 차후 제품을 위한 더 큰 최종 넘버를 허용하기 위해 뱅크에 조직화됩니다. 현재, 뱅크 0과 뱅크 1은 글로벌 파라미터를 위해 사용되고 뱅크 2는 유저변수, 뱅크 3은 인터럽트 환경설정을 위해 사용됩니다.

본 명령과 함께 사용될 수 있는 파라미터와 값을 가진 표는 글로벌 파라미터 리스트를 참고하시기 바랍니다.

내부 기능: 파라미터 형태는 따라오는 0을 무시하도록 변환됩니다 (혹은 마이너스 값). 이 파라미터는 적절한 장치에서 정확한 위치로 옮겨집니다.

연관된 명령어 : SGP, AGP

연상기호: GGP <parameter number>, <bank number>

이진법 표현:

지시 넘버	타입	모터/뱅크	값
10	챕터 6을 참고하십시오	<bank number>	(관계 없음)

Direct 모드에서 응답:

상태	값
100 - OK	(관계 없음)

예시:

시리얼 주소를 읽어 오십시오.

연상기호 : GGP 66, 0

이진법:

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$0a	$42	$00	$00	$00	$00	$00

Reply:

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
기능	호스트 주소	타겟 주소	상태	지시 넘버	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$02	$01	$64	$0a	$00	$00	$00	$01

상태 = 100 (에러 없음), 값 = 1

전용 I/O

전용 I/O 환경설정

EDB 시리즈는 전용 I/O를 지원합니다. I/O 는 프로그램 내에서 쉽게 ON/OFF 할 수 있습니다.
I/O 세팅을 활성화 시키기 위해서 아래의 글로벌 파라미터 (SGP72)를 사용하십시오.

SGP 72 값	설명
0	일반 I/O 환경 설정 활성화
1	ALARM OUT 활성화
2	POS_REACHED 활성화
3	ALARM OUT, POS_REACHED 활성화
4	POS ERROR 활성화
5	ALARM OUT, POS ERROR 활성화
6	POS_REACHED, POS ERROR 활성화
7	모든 전용 I/O 활성화

전용 I/O 환경설정의 ENABLE_IN 입력은 따로 활성화 해야 합니다. 다른 가능한 입력 극성은 글로벌 파라미터 SGP 80을 활용하여 환경 설정을 할 수 있습니다. 챕터 4.1을 참고하십시오.

SGP 80 값	설명
0 (디폴트)	비 활성화된 입력 활성화, 기본 입력 환경 설정
1	입력 low-active 활성화
2	입력 high-active 활성화

위치 혹은 속도 편차가 환경 설정 한 값을 초과 시 POS_ERROR 출력이 활성화 됩니다. 편차 값을 세팅하기 위해서 축 파라미터 SAP 212나 213을 사용하십시오. 최소 값으로 아래의 예시 값의 사용을 추천합니다.

예시:

최대 위치 편차: SAP 212, 0, 1000

최대 속도 편차: SAP 213, 0, 30000

전용 I/O 기능의 설명

I/O 기능	**IN/OUT**	**설명**
ALARM_OUT	OUT	드라이브 에러나 (-short to GND) 온도 과부하 발생시 알람출력이 활성화 됩니다.
POS_REACHED	OUT	엔코더 위치가 타겟 위치 도달 시 도달된 위치 출력이 활성화됩니다.
POS_ERROR	OUT	환경 설정 된 위치 (SAP 212)나 속도 (213) 편차 초과 시 위치 에러 출력이 활성화됩니다. 주의: 다른 모션 명령은 RESET_IN으로 POS_ERROR가 클리어 될 때까지 무시됩니다.
ENABLE_IN	IN	드라이버를 활성화 합니다. 주의: 극성은 글로벌 파라미터 80을 통해 환경설정 되어야 합니다.
RESET_IN	IN	RESET 입력은 POS_ERROR 플래그와 출력을 클리어 해줍니다.
ANALOG_IN	IN	다 목적 아날로그 입력

SIO (입/출력 세팅)

이 명령으로 디지털 출력의 상태를 Low (0) 나 High (1) 로 세팅합니다. 뱅크 2는 이러한 목적을 위해 사용됩니다.

내부 기능: 과거의 값은 특정 출력 라인으로 옮겨집니다.

연관된 명령어 : GIO, WAIT

연상기호: SIO <port number>, <bank number>, <값>

이진법 표현:

지시 넘버	타입	모터/뱅크	값
14	<port number>	<bank number> 2	< > 0/1

뱅크 2는 일반적 디지털 출력의 상태를 Low (0)나 High (1)로 세팅하는데 사용됩니다.

Direct 모드에서 응답:

상태	값
100 - OK	(관계 없음)

예시:

출력 2를 High 로 세팅 하세요. (뱅크 2, 출력 2)

연상기호 : SGP 2, 2, 1

이진법:

Byte Index	0	1	2	3	4	5	6	7
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$0e	$07	$02	$00	$00	$00	$01

출력 포트를 High 나 Low로 세팅하기 위한 명령

I/O 포트	명령	범위
OUT_0	SIO 0, 2, <값>	1/0
OUT_1	SIO 1, 2, <값>	1/0
OUT_2	SIO 2, 2, <값>	1/0

센서 STOP_L, STOP_R, HOME의 풀업 저항을 바꾸기 위한 특별 명령

I/O 포트	명령	범위
STOP_L	SIO 0, 0, <값>	1/0 1:ON 0:OFF
STOP_R		
HOME		

GIO (I/O 읽기)

이 기능을 통하여 아날로그 입력을 읽을 수 있습니다. 디지털 라인은 ADC 채널이 0 ...4095의 범위 내의 12비트 결과를 전달합니다.

Stand-Alone 모드에서 GIO

Stand-Alone 모드에서 요구된 값들은 조건부 점프와 같은 나중의 진행 목적을 위해 누적변수 (accu) 로 복사됩니다.

Direct 모드에서 GIO

Direct 모드에서 이 값은 누적변수에 영향이 없고 응답의 값 필드에서의 하나뿐인 출력입니다. 디지털 출력 라인의 실제 상태 또한 읽기 가능합니다.

내부 기능: 특정 라인을 읽을 수 있습니다.

연관된 명령어 : SIO, WAIT

연상기호: GIO <port number>, <bank number>

이진법 표현:

지시 넘버	타입	모터/뱅크	값
15	<port number>	<bank number>	(관계 없음)

Direct 모드에서 응답:

상태	값
100 - OK	(관계 없음)

예시:

입력 0의 디지털 값을 읽으십시오..

연상기호 : GIO 1, 0

이진법:

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$0f	$00	$00	$00	$00	$00	$00

Reply:

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
기능	호스트 주소	타겟 주소	상태	지시 넘버	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$02	$01	$64	$0f	$00	$00	$00	$01

상태 = 100 (에러 없음), 값 = 1

I/O 뱅크 0 – 디지털 입력

I/O 포트	명령	범위
IN_0	GIO 0, 0	0/1
IN_1	GIO 1, 0	0/1
IN_2	GIO 2, 0	0/1

I/O 뱅크 1 – 아날로그 입력

I/O 포트	명령	범위
IN_0	GIO 0, 1	0...4095

I/O 뱅크 2 – 디지털 출력 상태:

OUT 라인 (SIO 명령에 의해 세팅)의 상태는 뱅크 2를 이용하여 다시 읽기 가능합니다.

I/O 포트	명령	범위
OUT_0	GIO 0, 2, <값>	1/0
OUT_1	GIO 1, 2, <값>	1/0
OUT_2	GIO 2, 2, <값>	1/0

CALC (계산하기)

전의 GAP와 같은 기능에 의해 읽은 누적변수 안의 값은 이 명령으로 수정 가능합니다. 9개의 연산 기능이 선택될 수 있고 하나의 정수 피연산자 값은 반드시 특정되어야 합니다. 결과는 비교나 데이터 전송 후 프로세싱을 위해 누적변수에 다시 쓰여집니다.

연관된 명령어 : CALCX, COMP, JC, AAP, AGP, GAP, GGP

연상기호: CALC <operation>, <값>

<operation> 에는 ADD, SUB, MUL, DIV, MOD, AND, OR, XOR, NOT, LOAD를 사용 가능.

이진법 표현:

지시 넘버	타입	모터/뱅크	값
19	0 ADD -누적기에 더함 1 SUB – 누적기에서 뺌 2 MUL – 누적기에 곱함 3 DIV – 누적기에서 나눔 4 MOD 모듈에서 나눔 5 AND – 논리적으로 누적기와 6 OR – 논리적으로 누적기 혹은 7 XOR – 논리적으로 누적기 배타적 논리합 8 NOT – 논리적으로 변환된 누적기 9 LOAD – load operand를 누적기로 불러오기	(관계 없음)	<operand>

예시:

누적 변수에 -5000을 곱하시오.

연상기호 : CALC MUL, -5000

이진법:

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$13	$02	$00	$FF	$FF	$EC	$78

COMP (비교하기)

특정 넘버는 누적 변수 레지스터 안에서 값과 비교됩니다. 비교 결과는 예로 조건부 점프 지시에 의해 사용되곤 합니다. 이 명령은 Stand-Alone 구동에서만 사용 가능합니다.

호스트 주소와 응답 프로그램이 과부하일 경우 EMCL 프로그램 메모리로 지시를 주기 위해서 사용됩니다.

내부적 기능: 특정 값은 바로 앞에서 얻기나 계산하기 지시의 값을 가지고 있는 내부 누적변수와 비교됩니다 (GAP/GGP/CALC/CALCX 참고) 내부의 연산 상태 플래그가 비교 결과에 따라 세팅 됩니다.

연관된 명령어 : JC (조건부 점프), GAP, GGP, CALC, CALCX

연상기호: COMP <value>

이진법 표현:

지시 넘버	타입	모터/뱅크	값
20	(관계 없음)	(관계 없음)	<비교 값>

예시:

모터의 위치가 1000보다 크거나 같다면 라벨에 의해 주어진 주소로 점프하시오.

연상기호 : GAP 1, 2, 0 // 축 파라미터를 읽습니다. 타입: no.1 (실제위치), 모터:0, 값: 0 (관계 없음)

COMP 1000 // 실제 위치 값을 1000과 비교합니다.

JC GE, Label //조건부 점프, 타입: 5, 크거나 같다면, Label은 반드시 프로그램에 정의되어야 함.

COMP 1000 명령의 이진법 형태:

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$14	$00	$00	$00	$00	$03	$e8

JC (조건부 점프)

JC 지시는 특정 상황이 맞을 시 EMCL 프로그램 메모리의 수정된 주소로 조건부 점프를 하도록 합니다. 상황은 이전 비교 결과를 참고로 할 수 있습니다. 예로 COMP 지시를 참고하십시오. 이 기능은 Stand-Alone에서만 가능합니다.

호스트 주소와 응답은 프로그램이 과부하일 경우 EMCL 프로그램 메모리로 명령을 주기 위해서 사용됩니다. 이 명령은 Direct 모드에서는 불 가능합니다. 자세한 설명은 Host-only 컨트롤 기능을 참고하시기 바랍니다.

연관된 명령어 : JA, COMP, WAIT, CLE

연상기호: JC <상태>, <Label>

<상태> 에는 ZE/NZ/EQ/NE.GT/GE/LT/LE/ETO/EAL/EDV/EPO를 사용할 수 있습니다.

이진법 표현:

지시 넘버	타입	모터/뱅크	값
21	0 ZE -0 1 NZ – 0이 아니면 2 EQ– 같다면 3 NE – 같지 않다면 4 GT – 더 크다면 5 GE– 크거나 같다면 6 LT– 더 작다면 7 LE– 작거나 같다면 8 ETO – Time out error	(관계 없음)	<점프 주소>

예시:

모터의 위치가 1000보다 크거나 같다면 라벨에 의해 주어진 주소로 점프하시오.

```
연상기호 : GAP 1, 0, 0      // 축 파라미터 읽기, 타입: no.1 (실제 위치), 모터: 0, 값: 0 (관계 없음)
           COMP 1000        // 실제 을 1000과 비교합니다.
           JC GE, Label     // 크거나 같다면, Label로 점프합니다.

Label: ROL 0, 1000          // 좌측으로 1000pps의 속도로 회전하시오.
```

Label이 주소 10일때에 JC GE, Label의 이진 형태

Byte Index	0	1	2	3	4	5	6	7
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$15	$05	$00	$00	$00	$00	$0a

JA (항상 점프)

EMCL 프로그램 메모리에서 수정된 주소로 점프합니다. 이 명령은 Stand-Alone 모드에서 사용 가능합니다.

호스트 주소와 응답은 프로그램이 과부하일 경우 EMCL 프로그램 메모리로 명령을 주기 위해서 사용됩니다. 이 명령은 Direct 모드에서는 불 가능합니다.

내부적 기능: EMCL 프로그램 카운터가 과거 값을 세팅됩니다.

연관된 명령어 : JC, WAIT, CSUB

연상기호: JA <Label>

이진법 표현:

지시 넘버	타입	모터/뱅크	값
22	(관계 없음)	(관계 없음)	<점프 주소>

예시:

EMCL 안에서 무한 루프

```
연상기호 :  LOOP:   MVP ABS, 0, 10000
                    WAIT POS, 0, 0
                    MVP ABS, 0, 0
                    WAIT POS, 0, 0
                    JA LOOP              //LOOP로 점프
```

Label LOOP가 주소 20에 있다고 가정 시 JA LOOP의 이진형태

Byte Index	0	1	2	3	4	5	6	7
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$16	$00	$00	$00	$00	$00	$14

CSUB (서브루틴 불러오기)

이 기능으로 EMCL 프로그램 메모리에서 서브루틴을 불러옵니다. Stand-Alone 전용 모드입니다.

호스트 주소와 응답은 프로그램이 과부하일 경우 EMCL 프로그램 메모리로 명령을 주기 위해서 사용됩니다. 이 명령은 Direct 모드에서는 불 가능합니다.

내부적 기능: 지난 값이 중복 기입된 후 실제 EMCL 프로그램 카운터 값이 내부 스텍에 저장됩니다. 내부 스텍 안 엔트리 수는 8로 제한됩니다. 서브루틴 불러오기의 내부화 역시 8로 제한됩니다. 만약 스텍에 공간이 없다면 이 명령은 무시됩니다.

연관된 명령어 : RSUB, JA

연상기호: CSUB <Label>

이진법 표현:

지시 넘버	타입	모터/뱅크	값
23	(관계 없음)	(관계 없음)	<서브루틴 주소>

예시:

서브루틴 불러오기

```
연상기호 :  LOOP:   MVP ABS, 0, 10000
                    CSUB SubW     // 카운터 프로그램을 저장하고 SubW로 점프
                    MVP ABS, 0, 0
            SubW:   WAIT POS, 0, 0
                    WAIT TICKS, 0, 50
                    RSUB          // CSUB 명령을 따르는 명령으로 진행
```

Label SubW가 주소 100에 있다고 가정 시 CSUB SubW의 이진형태

Byte Index	0	1	2	3	4	5	6	7
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$17	$00	$00	$00	$00	$00	$64

RSUB (서브루틴으로부터 복귀하기)

CSUB 명령 후 서브루틴에서 명령으로 복귀합니다. Stand-Alone 전용 모드입니다.

호스트 주소와 응답은 프로그램이 과부하일 경우 EMCL 프로그램 메모리로 명령을 주기 위해서 사용됩니다. 이 명령은 Direct 모드에서는 불 가능합니다.

내부적 기능: 스텍의 과거 값에 EMCL 프로그램 카운터가 세팅 됩니다. 스텍이 비어 있다면 명령이 무시됩니다.

연관된 명령어 : CSUB

연상기호: RSUB

이진법 표현:

지시 넘버	타입	모터/뱅크	값
24	(관계 없음)	(관계 없음)	(관계 없음)

예시:

CSUB 예시를 참고 하십시오.

```
연상기호 :  LOOP:   MVP ABS, 0, 10000
                    CSUB SubW     // 카운터 프로그램을 저장하고 SubW로 점프
                    MVP ABS, 0, 0
            SubW:   WAIT POS, 0, 0
                    WAIT TICKS, 0, 50
                    RSUB          // CSUB 명령을 따르는 명령으로 진행
```

RSUB의 이진 형태

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$18	$00	$00	$00	$00	$00	$00

WAIT (이벤트 발생까지 대기하기)

이 명령은 특정 환경이 맞을 때 까지 EMCL프로그램의 실행을 인터럽트 합니다. Stand-Alone 전용 모드입니다.

호스트 주소와 응답은 프로그램이 과부하일 경우 EMCL 프로그램 메모리로 명령을 주기 위해서 사용됩니다. 이 명령은 Direct 모드에서는 불 가능합니다.

사용될 수 있는 다섯가지 대기 상태:

- **TICKS**: <ticks> 파라미터에 의해 특정된 타이머의 숫자에 도달할 때 까지 대기합니다.
- **POS**: <motor> 파라미터에 의해 특정된 모터의 타겟 위치에 도달할 때까지 대기합니다. 선택적인 타임아웃값 (0은 타임아웃 없음)은 <ticks> 파라미터에 의해 특정 되어야 만합니다.
- **REFSW**: <motor> 파라미터에 의해 특정된 모터의 리퍼런스 스위치가 반응할 때까지 대기합니다. 선택적인 타임아웃값 (0은 타임아웃 없음)은 반드시 <ticks> 파라미터에 의해 특정 되어야 만합니다.
- **LIMSW**: <motor> 파라미터에 의해 특정된 모터의 리미트 스위치가 반응할 때까지 대기합니다. 선택적인 타임아웃값 (0은 타임아웃 없음)은 반드시 <ticks> 파라미터에 의해 특정 되어야 만합니다.
- **RFS**: <motor> 필드에 의해 특정된 모터의 리퍼런스 서치가 도달할 때까지 대기합니다. 선택적인 타임아웃값 (0은 타임아웃 없음)은 반드시 <ticks> 파라미터에 의해 특정 되어야 만합니다.

타임 아웃 제한 도달 후 타임아웃 플래그 (ETO: Time Out Flag)가 세팅됩니다. 그리고 JC ETO 명령으로 이러한 에러를 찾거나 CLE 명령으로 에러를 제거할 수 있습니다.

내부적 기능: 특정 조건이 맞을 때 까지 EMCL 프로그램 카운터가 홀딩 됩니다.
연관된 명령어 : JC, CLE
연상기호: WAIT <상황>, 0, <ticks>
<상황>에는 TICKS/ POS/ REFSW/ LIMSW/ RFS 가 사용 가능합니다.

이진법 표현:

지시 넘버	타입	모터/ 뱅크	값
27	0 TICKS - 타이머 ticks*1	(관계 없음)	ticks*1 의 값
	1 POS - 타겟 위치 도달	0*2	타임아웃을 위한 ticks*1 의 값, 0 은 타임아웃 없음
	2 REFSW - Home 리퍼런스 스위치	0*2	타임아웃을 위한 ticks*1 의 값, 0 은 타임아웃 없음
	3 LIMSW - 리미트 스위치	0*2	타임아웃을 위한 ticks*1 의 값, 0 은 타임아웃 없음
	4 RFS - 홈 리퍼런스 서치 완료	0*2	타임아웃을 위한 ticks*1 의 값, 0 은 타임아웃 없음

***1 1tick 은 10 밀리 세컨드입니다 (1밀리 세컨드 = 1/1000초)**

***2 한 개의모터만 사용할 경우 모터 넘버는 항상 0입니다.**

예: 시간 제한 없이 모터가 타겟 위치에 도달하는 것을 기다리십시오.
연상 기호 : WAIT POS, 0, 0

이진 형태

Byte Index	0	1	2	3	4	5	6	7
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$1b	$01	$00	$00	$00	$00	$00

STOP (EMCL 프로그램 실행 정지)

이 명령은 EMCL 프로그램의 실행을 정지 시킵니다. 호스트 주소와 응답은 지시를 EMCL 프로그램 메모리로 옮기는데 사용됩니다.

이 명령은 모든 Stand-Alone EMCL 프로그램의 끝에 위치 합니다.. 이 명령은 Direct 모드에서는 불 가능합니다.

내부적 기능: EMCL의 명령이 멈춥니다.

연관된 명령어 : none

연상기호: STOP

이진법 표현:

지시 넘버	타입	모터/뱅크	값
28	(관계 없음)	(관계 없음)	(관계 없음)

예시:

연상기호 : STOP

RSUB의 이진 형태

Byte Index	0	1	2	3	4	5	6	7
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$1c	$00	$00	$00	$00	$00	$00

SCO (좌표 세팅)

MVP COORD 명령으로 사용할 모든 축에 20개의 위치 값이 저장 될 수 있습니다. 이 명령은 특정 값으로 좌표를 세팅합니다. 글로벌 파라미터 84에 따라 좌표는 RAM에만 저장되거나 EEPROM에도 같이 저장할 수 있고 시작으로 복사 할 수 있습니다 (디폴트 세팅에서 좌표는 RAM으로만 저장됩니다).

주의: 좌표 넘버 0은 항상 RAM에만 저장됩니다.

내부적 기능: 내부 위치 배열에 과거 값이 저장됩니다.

연관된 명령어 : GCO, CCO, MVP

연상기호: SCO <좌표 넘버>, 0, <위치>

이진법 표현:

지시 넘버	타입	모터/뱅크	값
30	<좌표 넘버> 0...20	0	<위치값> -231 ...+(231-1)

Direct 모드에서 응답:

상태	값
100 - OK	(관계 없음)

예시:

모터 #1의 좌표를 1000으로 세팅하십시오.

연상기호: SCO 1, 0, 1000

이진 형태

Byte Index	0	1	2	3	4	5	6	7
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$1e	$01	$00	$00	$00	$03	$e8

이 명령의 두가지 특별한 기능으로 모든 좌표를 복사하거나 선택된 좌표를 EEPROM으로 복사 할 수 있습니다.

SCO 0, 255, 0 // RAM 에서 EEPROM으로 모든 좌표 복사 (좌표 넘버 0 제외)

SCO <좌표 넘버>, 255, 0 // <좌표 넘버>로 선택된 좌표를 EEPROM으로 복사합니다. 좌표 넘버 1과 20 사이의 값이어야 합니다.

GCO (좌표 읽기)

이 명령은 전에 저장된 좌표를 읽어올 수 있습니다. Stand-Alone 모드에서 요구되는 값은 조건부 점프와 같이 후의 진행 목적을 위해 누적 변수 레지스터로 복사됩니다. Direct 모드에서 값은 누적 변수에 영향 없이 응답의 값 필드에서 오직 하나의 출력입니다. 글로벌 파라미터 84에 따라 좌표는 RAM에만 저장되거나 EEPROM에도 같이 저장할 수 있고 시작으로 복사할 수 있습니다 (디폴트 세팅에서 좌표는 RAM으로만 저장됩니다).

주의: 좌표 넘버 0은 항상 RAM에만 저장됩니다.

내부적 기능: 원하는 값은 내부 좌표에서 읽기 되어지고 누적 변수 레지스터로 복사됩니다. 그리고 Direct 모드에서 응답의 값 필드로 복귀합니다.

연관된 명령어 : SCO, CCO, MVP

연상기호: GCO <좌표 넘버>, 0

이진법 표현:

지시 넘버	타입	모터/뱅크	값
31	<좌표 넘버> 0...20	0	(관계 없음)

Direct 모드에서 응답:

상태	값
100 - OK	(관계 없음)

예시: 좌표 1의 모터 값을 얻으시오.
연상기호: GCO 1, 0

이진 형태

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$1f	$01	$00	$00	$00	$00	$00

Reply:

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
기능	호스트 주소	타겟 주소	상태	지시 넘버	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$02	$01	$64	$0a	$00	$00	$00	$00

이 명령의 두가지 특별한 기능으로 모든 좌표를 복사하거나 선택된 좌표를 EEPROM으로 복사 할 수 있습니다.

SCO 0, 255, 0 // RAM 에서 EEPROM으로 모든 좌표 복사 (좌표 넘버 0 제외)

SCO <좌표 넘버>, 255, 0 // <좌표 넘버>로 선택된 좌표를 EEPROM으로 복사합니다. 좌표 넘버 1과 20 사이의 값이어야 합니다.

CCO (좌표 캡쳐하기)

축의 실제 위치가 선택된 좌표 변수로 복사됩니다. 글로벌 파라미터 84에 따라 좌표는 RAM에만 저장되거나 EEPROM에도 같이 저장할 수 있고 시작으로 복사 할 수 있습니다 (디폴트 세팅에서 좌표는 RAM으로만 저장됩니다). RAM과 EEPROM 사이의 좌표 복사하는 방법은 SCO와 GCO 명령을 참고하십시오.

주의: 좌표 넘버 0은 항상 RAM에만 저장됩니다.

내부적 기능: 선택된 24 bit 포지션 값들은 20 by 3 byte 폭의 좌표 배열로 쓰여집니다.

연관된 명령어 : SCO, GCO, MVP

연상기호: CCO <좌표 넘버>, 0

이진법 표현:

지시 넘버	타입	모터/뱅크	값
32	<좌표 넘버> 0...20	0	(관계 없음)

Direct 모드에서 응답:

상태	값
100 - OK	(관계 없음)

예시:

축 0번의 현재 위치를 좌표 3에 저장하십시오.

연상기호: CCO 3, 0

이진 형태

Byte Index	0	1	2	3	4	5	6	7
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$20	$03	$00	$00	$00	$03	$e8

ACO (누적변수에서 좌표로)

ACO 명령으로 누적변수의 실제 값은 모터의 선택된 좌표로 복사됩니다. 글로벌 파라미터 84에 의해 좌표들은 RAM에만 저장되거나 EEPROM에도 같이 저장 되거나 시작으로 복사됩니다. (디폴트 세팅에서는 RAM에만 저장)

주의: 좌표 넘버 0은 항상 RAM에만 저장됩니다. 좌표 저장에 관한 정보는 SCO 명령부분을 참고하십시오.

내부적 기능: 내부 위치 배열로 누적변수의 실제 값이 저장됩니다.

연관된 명령어 : GCO, CCO, MVP, COORD, SCO

연상기호: ACO <좌표 넘버>, 0

이진법 표현:

지시 넘버	타입	모터/뱅크	값
39	<좌표 넘버> 0...20	0	(관계 없음)

Direct 모드에서 응답:

상태	값
100 - OK	(관계 없음)

예시:

누적 변수의 실제 값을 모터 0의 좌표 1로 복사하십시오.

연상기호: ACO 1, 0

이진 형태

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$27	$01	$00	$00	$00	$00	$e8

CALCX (X 레지스터를 이용한 계산)

이 명령은 CALC와 매우 비슷하지만 두번째 피연산자가 X 레지스터에서 옵니다. X 레지스터는 LOAD나 SWAP 타입으로 불러올 수 있습니다. 비교나 데이터 전송과 같은 진행을 위해 결과는 누적변수에 기록됩니다.

연관된 명령어 : CALC, COMP, JC, AAP, AGP

연상기호: CALCX <동작>

<동작>에는 ADD/SUB/MUL/DIV/MOD/AND/OR/XOR/NOT/LOAD/SWAP

이진법 표현:

지시 넘버	타입	모터/ 뱅크	값
33	0 ADD - X 레지스터를 누적기로 더함	(관계 없음)	(관계 없음)
	1 SUB - X 레지스터를 누적기에서 뺌		
	2 MUL - 누적기와 X 레지스터를 곱함		
	3 DIV - 누적기를 X 레지스터로 나눔		
	4 MOD - 누적기를 X 레지스터로 나눈 후 나머지 계산		
	5 AND - 논리적으로 X 레지스터와 누적기		
	6 OR - 논리적으로 X 레지스터 혹은 누적기		
	7 XOR - 논리적으로 X 레지스터와 누적기의 배타적 논리합		
	8 NOT - 논리적으로 X 레지스터의 변환		
	9 LOAD - 누적기를 X 레지스터로 불러오기		
	10 SWAP - 누적기와 X 레지스터의 스왑		

예시:

누적 변수에 X 레지스터를 곱하시오.

연상기호: CALCX MUL

이진 형태

Byte Index	0	1	2	3	4	5	6	7
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$21	$02	$00	$00	$00	$00	$00

AAP (누적 변수에서 축 파라미터로)

누적변수 레지스터 안에 있는 내용은 특정 축 파라미터로 옮겨집니다. 알맞은 사용을 위하여 누적변수를 불러와야 합니다. 예) 이전의 GAP 지시. 누적 변수는 CALC나 CALCX 지시에 의해 수정되었을 경우도 있습니다.

이 명령으로 함께 사용될 수 있는 파라미터와 값의 표는 챕터 3을 참조하십시오.

연관된 명령어 : AGP, SAP, GAP, SGP, GGP, CALC, CALCX

연상기호: AAP <파라미터 넘버>, 0

이진법 표현:

지시 넘버	타입	모터/뱅크	값
34	<파라미터 넘버>	0*	(관계 없음)

하나의 모터일 때 모터 넘버는 항상 0입니다.

Direct 모드에서 응답:

상태	값
100 - OK	(관계 없음)

예시:

아날로그 인풋 #0에 연결된 전위차계에 의한 모터 포지셔닝:

```
연상기호:   Start:   GIO 0, 1          //아날로그 입력라인 0의 값 얻기
                     CALC MUL, 4       // 4를 곱합니다.
                     AAP 0, 0          // 모터 0의 타겟 위치로 결과를 옮기십시오.
                     JA Start          // Start로 점프
```

AAP 0, 0 명령의 이진 형태

Byte Index	0	1	2	3	4	5	6	7
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$22	$00	$00	$00	$00	$00	$00

AGP (누적 변수에서 글로벌 파라미터로)

누적변수 레지스터 안에 있는 내용은 특정 글로벌 파라미터로 옮겨집니다. 알맞은 사용을 위하여 누적변수를 불러와야 합니다. 예) 이전의 GAP 지시. 누적 변수는 CALC나 CALCX 지시에 의해 수정되었을 경우도 있습니다.

주의: 뱅크 0 안의 글로벌 파라미터는 EEPROM전용이고, 따라서 Stand-Alone 어플리케이션에 의해 자동적으로 수정되어서는 안됩니다. (글로벌 파라미터 리스트는 챕터 6을 참조하십시오).

연관된 명령어 : AAP, SGP, GGP, SAP, GAP

연상기호: AGP <파라미터 넘버>, <뱅크 넘버>

이진법 표현:

지시 넘버	타입	모터/뱅크	값
35	<파라미터 넘버>	<뱅크 넘버>	(관계 없음)

Direct 모드에서 응답:

상태	값
100 - OK	(관계 없음)

이 명령과 함께 사용될 수 있는 뱅크 넘버와 파라미터는 챕터 6을 참고하십시오.

예시:

누적 변수를 EMCL 유저 변수 #3로 복사하십시오.

연상기호: AGP 3, 2

이진 형태

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$23	$03	$02	$00	$00	$00	$00

CLE (에러 플래그 클리어)

이 명령은 내부 에러 플래그를 없애줍니다.

Stand-Alone 모드에서만 사용 가능하고 Direct 모드에서는 절대 사용 되어선 안됩니다.

연관된 명령어 : JC

연상기호: CLE <플래그>

<플래그>에는 ALL/ETO/EDV/EPO를 사용할 수 있습니다.

이진법 표현:

지시 넘버	타입	모터/뱅크	값
36	0 – (ALL) 모든 플래그 1 – (ETO) 타임아웃 플래그 3 – (EDV) 편차 플래그	<뱅크 넘버>	(관계 없음)

예시:

타임아웃 깃발을 리셋 하십시오

연상기호: CLE ETO

이진 형태

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
기능	타겟 주소	지시 넘버	타입	모터/뱅크	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$01	$24	$01	$00	$00	$00	$00	$00

모터 파워 스위치 ON/OFF (UF1)

유저 기능 UF1은 드라이브 스테이지의 파워를 키고 끄는데 사용될 수 있습니다. 일반적으로 드라이브 스테이지는 켜져 있지만 이 명령어를 사용하여 끌 수 있습니다.

UF1 0, 0, 0 -> 드라이브 스테이지의 파워 OFF
UF1 0, 0, 1 **->** 드라이브 스테이지의 파워 ON
UF1 1, 0, 0 **->** 드라이브 스테이지의 상태 체크, 응답이 0이면 OFF, 1이면 ON

사용자 특정 EMCL 명령 확장 (UF0 ...UF7 / 유저 기능)

유저가 정의 할 수 있는 기능 UF0 ... 7 은 사전 정의 되어 있고, 유저 특정 목적을 위한 토픽 없이 가능한 기능입니다. 이러한 기능의 특정 프로그래밍을 위해서는 ERAETECH 본사에 문의 부탁드립니다.

내부 기능: ERAETECH 에 의한 C에서 실행되는 유저 특정 기능을 불러옵니다.

연관된 명령어 : none

연상기호: UF0 ... UF7

이진법 표현:

지시 넘버	타입	모터/뱅크	값
64 ... 71	(유저에 의한 정의)	(유저에 의한 정의)	(유저에 의한 정의)

Direct 모드에서 응답:

Byte Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
기능	타겟 주소	타겟 주소	상태	지시 넘버	Operand Byte3	Operand Byte2	Operand Byte1	Operand Byte0
(hex)	$02	$01	유저정의	64 ... 71	유저정의	유저정의	유저정의	유저정의

EMCL 제어기능

따라오는 기능들은 호스트 제어 목적이기 때문에 Stand-Alone 모드에서는 허용되지 않습니다. 대부분의 케이스에서 유저가 이러한 기능들을 사용할 필요는 없습니다. (명령 139제외)

EMCL 제어 명령은 EMCL 프로그램에서 사용될 수 없기 때문에 연상 기호는 제공하지 않습니다. 이러한 기능들은 EMCL-IDE에 의해서만 사용됩니다. (예: EMCL 어플리케이션을 모듈에 다운로드 하기)

유저 호스트 어플리케이션을 위한 가치가 있는 제어 명령들은:

- 펌웨어 리비젼 읽기 (명령 136, 이 장의 마지막에 설명되어 있는 이 명령의 특별한 응답형태를 확인하십시오).
- 어플리케이션 실행하기 (명령 129)

모든 다른 기능들은 EMCL-IDE의 적절한 기능을 사용함으로 실행됩니다.

지시	설명	타입	모터/뱅크	값
128 - 어플리케이션 중지	작동중인 EMCL Stand-Alone 어플리케이션이 중지 됩니다.	(관계 없음)	(관계 없음)	(관계 없음)
129 - 어플리케이션 실행	EMCL 실행이 시작됩니다. (혹은 계속 진행됩니다)		(관계 없음)	(관계 없음) 시작주소
130 - 어플리케이션 스텝	EMCL 어플리케이션의 다음 명령만 실행됩니다.	(관계 없음)	(관계 없음)	(관계 없음)
131- 어플리케이션 리셋	프로그램 카운터가 0 으로 세팅되고 Stand-Alone 어플리케이션이 멈춥니다 (작동이나 스텝중)	(관계 없음)	(관계 없음)	(관계 없음)
132- 다운로드 모드 시작	타겟 명령 실행이 멈추고 따라오는 모든 명령들이 EMCL 메모리로 이동합니다.	(관계 없음)	(관계 없음)	어플리케이션 시작주소
133- 다운로드 모드 나가기	타겟 명령 실행이 재개됩니다.	(관계 없음)	(관계 없음)	(관계 없음)
134- EMCL 메모리 읽기	특정 프로그램 메모리 위치를 읽어냅니다	(관계 없음)	(관계 없음)	(관계 없음)
135- 어플리케이션 상태 얻기	이 값들중 하나로 복귀합니다. 0- 정지 1- 실행 2- 스텝 3- 리셋	(관계 없음)	(관계 없음)	(관계 없음)
136- 펌웨어 버전 읽기	모듈타입과 펌웨어 수정사항이 문자열이나 이진법 형태중 하나로 복귀합니다.	0- 문자열 1- 이진법	(관계 없음)	(관계 없음)
137- 초기상태로 복구하기	EEPROM 에 저장된 모든 세팅이 공장 초기화 상태로 리셋됩니다. 이 명령은 응답을 보내지 않습니다.	(관계 없음)	(관계 없음)	반드시 1234

명령 136의 특별한 응답 형태:

- 0으로 세팅: 문자열로써 응답:

Byte Index	설명
1	호스트 주소
2...9	버전 문자열 (8자리) 예시: 1360V1.31

*응답형태에 체크섬이 없습니다!

- 1로 세팅: 이진법 형태에서 버전 넘버:
  - 일반 응답형태를 사용하십시오.
  - 따라오는 방법으로 버전 넘버는 응답의 값 필드에서 출력입니다.

Byte Index	설명
1	버전 넘버, 낮은 바이트
2	버전 넘버, 높은 바이트
3	타입 넘버, 낮은 바이트
4	타입 넘버, 높은 바이트

5. 축 파라미터

SAP, GAP, AAP, STAP, RSAP 명령과 함께 사용될 수 있는 모든 축 파라미터에 대한 리스트입니다.

문자 의미:

Access 타입	관련된 명령	설명
R	GAP	읽기 가능한 파라미터
W	SAP, AAP	쓰기 가능한 파라미터

적절한 모터의 작동을 위해 기본 파라미터는 반드시 모터/어플리케이션에 조정되어야 합니다.

많은 경험이 있는 유저를 위한 파라미터 – 불 확실하다면 변경하지 마십시오.

넘버	축 파라미터	설명	범위	Acc
0	목표 위치	위치 모드에서 원하는 위치	-2.147.483.648 ... +2.147.483.647 [µsteps]	RW
1	실제위치	모터의 현재 포지션. 기준점 설정을 덮어쓰기 해야합니다	-2.147.483.648 ... +2.147.483.648 [µsteps]	RW
2	목표 속도	속도 모드에서 원하는 속도. 위치모드에서 파라미터는 하드웨어에 의해 세팅됩니다. - 최대속도까지 가속중 - 0 까지 감속중	-8.388.608 ... +8.388.607 [pps]	RW
3	실제 속도	현재 회전 속도	-8.388.608 ... +8.388.607 [pps]	RW
4	최대 포지셔닝 속도	위치 이동 중 최대 속도	-8.388.608 ... +8.388.607 [pps]	RW
5	최대 가속	가속(및 감속)한계. 본 파라미터의 변경은 가속 팩터 (no.146) 및 가속 제수(no. 137)의 재 측정을 필요로 하며 이는 자동적 으로 이뤄짐.	-8.388.608 ... +8.388.607 [ps]	RW
6	절대적 최대 모터 전류	최대 값은 255 입니다. 이 값은 모듈의 선택된 전류 범위의 100%를 뜻합니다. 두 전류 범위선택이 가능합니다. (Axis 파라미터 179 참조). 너무 높은 값은 모터에 데미지를 가할 수 있기에 가장 중요한 모터 세팅입니다!	0...255 peak= <value> x 4.2A /255 IRMS= <value> x 2.8A /255	RW
7	대기 모터 전류	모터 중지 및 power down 지연 시간 후의 전류의 한계 값	0...255 peak= <value> x 4.2A /255 IRMS= <value> x 2.8A /255	RW

넘버	축 파라미터	설명	범위	Acc
8	목표 위치 도달 플래그	실제 위치와 목표 위치가 같은 것을 말해줍니다.	0/1	R
10	우측 제한 스위치 상태	우측 제한 스위치의 논리적 상태	0/1	R
11	좌측 제한 스위치 상태	좌측 제한 스위치의 논리적 상태	0/1	R
12	우측 제한 스위치 활성화/ 양극성	0 = 우측 제한 스위치 비활성화 됨 1 = 우측 제한 스위치 활성화, input 이 low 일때 모터 정지 3 = 우측 제한 스위치 활성화, input 이 high 이거나 open 상태일 시 모터 정지 (내부 풀업)	0,1,3	RW
13	좌측 제한 스위치 활성화/ 양극성	0 = 좌측 제한 스위치 비활성화 됨 1 = 좌측 제한 스위치 활성화, input 이 low 일때 모터 정지 3 = 좌측 제한 스위치 활성화, input 이 high 이거나 open 상태일 시 모터 정지 (내부 풀업)	0,1,3	RW
15	시작 속도	사다리꼴 상승: pps 에서 동작 시 시작 속도.	-8.388.608 … +8.388.607 [pps]	RW
16	가속 시작	사다리꼴 상승: 모션 동작 시 시작 가속. 브레이크 속도(Axis 파라미터 18)에 도달 하자마자 최대 가속(Axis 파라미터 5)이 바뀝니다.	-4.194.304 … +4.194.303 [pps2]	RW
17	최대 감속	사다리꼴 하강: 최대 감속, 최대 가속 (Axis 파라미터 5)와 같은 값으로 세팅할 수 있습니다.. 그러나 다른 값으로 또한 세팅 가능합니다.	-4.194.304 … +4.194.303 [pps2]	RW
18	브레이크 속도	사다리꼴 하강: 이 속도 이하에서 가속 시작 (Axis 파라미터 16)이 사용되고 –이 속도 이상에서 최대가속 (Axis 파라미터 5)이 사용됩니다.	-8.388.608 … +8.388.607 [pps]	RW
19	최종 감속	사디리꼴 하강: 실제 속도, 브레이크 속도 (Axis 파라미터 18) 이하로 가는 순간 감속이 됩니다.	-4.194.304 … +4.194.303 [pps2]	RW
20	정지 속도	사다리꼴 하강: 타겟 위치에 도달하고 모터가 정지할 때의 속도.	-8.388.608 … +8.388.607 [pps]	RW
21	감속 중지	정지 스위치 우측 제한 스위치/좌측 제한 스위치혹은 가상 정지가 소프트 램프와 함께 사용되었을 시 감속 값. 정지 스위치가 들어가거나 가상 정지에 도달 시 0 이 아닌 값이 자동 리니어 램프에 생성됩니다.	-4.194.304 … +4.194.303 [pps2]	RW
26	좌측 가상 정지	모터가 – 방향으로 작동 시 가상 정지 위치.	-2.147.483.648 … +2.147.483.647	RW
27	우측 가상 정지	모터가 + 방향으로 작동 시 가상 정지 위치	-2.147.483.648 … +2.147.483.647	RW
28	가상 정지 활성화	0 = 양측 가상 정지 비활성 1 = 좌측 가상정지 활성 2 = 우측 가상정지 활성 3 = 양측 가상정지 활성	0, 1, 2, 3	RW

넘버	축 파라미터	설명	범위	Acc
29	가상 정지 모드	0 = 모터 정지를 위해 전류램프 감속 세팅사용 1 = 강하고 빠른 정지 2 = 정지 감속 (Axis 파라미터 21)을 사용하는 리니어램프와 함께 정지.	0, 1, 2	RW
108	CL gamma V 최소	더 높은 속도를 위해서 back EMF 의 효과를 보상합니다. 이 속도에서 보상이 시작됩니다.	-8.388.608 ... +8.388.607 [pps]	RW
109	CL gamma V 최대	더 높은 속도를 위해 back EMF 의 효과를 보상합니다. 이 속도에서 보상이 시작됩니다	-8.388.608 ... +8.388.607 [pps]	RW
110	CL 최대 gamma	더 높은 속도를 위해 back EMF 의 효과를 보상 합니다. 이것은 현재 속도 (Axis 파라미터 108, 109)를 사용하는 스케일에 추가되는 보상의 원인입니다	0 .. 255	RW
111	CL beta	측정된 위치 오차를 보상하는데 사용되는 최대 통신각입니다.	0.. 511	RW
112	CL offset	클로즈 루프 작동을 위한 오프셋. 클로즈 루프 초기화동안 측정됩니다.	-2.147.483.648 ... +2.147.483.647	RW
113	CL 최소 전류	클로즈 루프작동에서 최소 전류 세팅	0.. 255	RW
114	CL 최대 전류	클로즈 루프 작동에서 최대 전류 세팅.	0.. 255	RW
115	CL 정정속도 P	클로즈 루프 제한동안 최대속도를 제어하는 PI 레귤레이터의 파라미터 P	-16.777.216 ... +16.777.217	RW
116	CL 정정속도 I	클로즈 루프 제한동안 최대 속도를 제어하는 PI 레귤레이터의 파라미터 I	-16.777.216 ... +16.777.217	RW
117	CL 정정속도 I 클리핑 (clipping)	클로즈 루프 제한동안 최대 속도를 제어하는 PI 레귤레이터 통합 파트의 에러 전체를 클리핑	-32.768 ... +32.768	RW
118	CL 정정속도 DV clock	D 파트 계산을 위한 클록 clock 디바이더.	-32.768 ... +32.768	RW
119	CL 정정속도 DV 클리핑	최대 속도 이상에서 최대 오차를 제한.	-2.147.483.648 ... +2.147.483.647	RW
120	CL 업스케일 딜레이	클로즈 루프 작동중 모터 전류 증가시 딜레이	-16.777.216 ... +16.777.217	RW
121	CL 다운스케일 딜레이	클로즈 루프 작동 동안 모터 전류 감소시 딜레이	-16.777.216 ... +16.777.217	RW
124	CL 정정 위치 P	감지된 위치 오차를 정정하기 위한 비례적 컨트롤러 P 파라미터. P 파라미터의 결과는 이 파라미터를 65535 으로 나누면 됩니다	-16.777.216 ... +16.777.217	RW

넘버	축 파라미터	설명	범위	Acc
125	CL 최대 정정 허용	위치 오차에 대한 허용 범위.	0… 255	RW
127	상대적 포지셔닝 옵션	리퍼런스 위치와 같이 받음: 0 = 마지막 타겟 위치 1 = 현재 위치 2 = 엔코더 위치	0, 1, 2	RW
129	Closed-Loop 플래그	0 = 클로즈 루프 off 1 = 클로즈 루프 작동 스위치 on (클로즈 루프와 관련 있는 파라미터는 반드시 정확한 값을 세팅해야 합니다). 어떠한 모터작동 초기화 전 클로즈 루프 초기화가 끝날 때 까지 (Axis 파라미터 133 을 "1"로 세팅) 기다리십시오.	0/1	RW
133	클로즈 루프 초기화 깃발	0 = 초기화가 아직 끝나지 않음. 1 = 클로즈 루프 초기화를 마침.	0/1	R
134	포지셔닝 윈도우	타겟 도달한 깃발에 대한 최대 오차	Write: -4.294.967.296 … +4.294.967.295 Read: -2.147.483.648 … +2.147.483.647	RW
136	EncMeanWait	엔코더 필터 관련		RW
137	EncMeanFilter	엔코더 필터 관련		RW
138	EncMeanInt	엔코더 필터 관련		RW
162	쵸퍼 블랭크 타임	콤퍼레이터 블랭크 타임을 선택하십시오. 이 타임은 이벤트와 센서 저항의 알람의 스위칭을 안전하게 보호합니다. 낮은 전류 드라이버에서 1 이나 2 의 세팅이 좋습니다	0… 3	RW
163	쵸퍼 Mode	쵸퍼 Mode 의 선택: 0 – spread cycle 1 – classic const. off time	0/1	RW
164	쵸퍼 히스테레시스 감쇄 decrement	히스테레시스 감쇠 세팅. 이 세팅은 제시간 동안이나 빠른 감쇠 시간동안 히스테레시스의 기울기를 결정합니다. 0 – 빠른 감쇠 3 – 매우 느린 감쇠	0… 3	RW

165	쵸퍼 히스테레시스 종료	히스테레시스 종료 세팅. 감쇠숫자 후 종료 값을 세팅하십시오. 감쇠 인터벌 시간은 Axis 파라미터 164 에 의해 제어됩니다. -3... -1 마이너스 히스테레시스 종료 세팅 0 0 히스테레시스 종료 세팅 1... 12 플러스 히스테레시스 종료 세팅	-3... 12	RW
166	쵸퍼 히스테레시스 시작	히스테레시스 시작 세팅. 이 값은 히스테레시스 종료값에 대한 offset 입니다.	0... 8	RW
167	쵸퍼 off time	오프셋 타임 세팅은 최소 쵸퍼 진동수를 제어합니다. 5µs 에서 20µs 의 범위 안의 오프셋이 적당합니다. 정수 tOff 쵸퍼를 위한 오프셋 세팅: NCLK= 12 + 32*tOFF (최소는 64 clocks) 이 파라미터를 0 으로 세팅하는 것은 모든 드라이버 트랜지스터를 비활성화 시키고 모터는 자유 회전합니다.	0 / 2... 15	RW
168	smartEnergy 전류 최소 (SEIMIN)	CS 값 스케일링에 의한 coolstep™작동을 위해 낮은 모터 전류 제한을 세팅하십시오. (전류 스케일, Axis 파라미터 6 참조) 최소 모터 전류 0 – CS 의 ½ 1 – CS 의 ¼	0 / 1	RW
169	smartEnergy 전류 다운 스텝	모터 전류의 각각의 전류 감소에 대한 상위 스레시홀드 이상에서 StallGuard2™ 읽기의 수를 세팅하십시오. 각 감소마다 StallGuard2™측정의 수 스케일링 : 0... 3: 32, 8, 2, 1 0: 느린 감소 3:빠른 감소	0... 3	RW
170	smartEnergy 히스테레시스	StallGuard2™읽기에 대한 높고 낮은 스레시홀드 사이의 거리를 세팅 하십시오. 상위 스레시홀드 이상에서 모터 전류는 감소하게 됩니다. 히스테레시스 (smartEnergy 히스테레시스 값+1)*32 상위 stallGuard 스레시홀드: (smartEnergy 히스테레시스 시작 + smartEnergy 히스테레시스 + 1) * 32	0... 15	RW
171	smartEnergy 전류 업 스텝	전류 증가 스텝을 세팅 하십시오. 전류는 하위 스레시홀 드 이하의 각각 측정된 stallGuard2™값을 위해 증가합니 다. (smartEnergy 히스테레시스를 참조하십시오). 0: 느린증가 3: 빠른 증가/ 증가 부하에 대한 반응	1... 3	RW
172	smartEnergy 히스테레시스 시작	stallGuard2™값에 대한 낮은 스레시홀드 (smart Energy 전류 업 스텝 참조).	0... 15	RW

173	stallGuard2 필터 활성화	측정의 정확함을 위해 stallGuard2™필터를 활성화 합니다. 세팅이 되면 측정주기를 4 풀 스텝당 한 번으로 감소시킵니다. 대부분의 경우 coolStep™사용전 필터 Mode 를 세팅하는 것이 일반적입니다. 탈조 감지를 위해 스탠다드 Mode 를 사용합니다 0 – 일반 Mode 1 – 필터 Mode	0/1	RW
174	stallGuard2 스레시홀드	사인된 값은 stallGuard2™ 스톨 아웃풋을 위해 스레시홀 드 레벨을 제어하고 읽기에 최상의 측정범위를 세팅합니 다. 낮은 값일수록 민감합니다. 0 은 시작값입니다. 높은 값은 stallGuard2™를 둔감하게 만들고 스톨을 지시하기 위한 더 많은 토크를 필요로 합니다 0 증립 값 1... 63 덜 민감함 -1... -64 더 민감함	-64... 63	RW
175	기울기 (slope) 제어 높은방향	모터 드라이버 아웃풋의 기울기 slope 를 결정합니다. 2 나 3 으로 세팅하거나 디폴트값을 사용합니다 0: lowest slope 3: fastest slope	0... 3	RW
176	기울기 (slope) 제어 낮은 방향	모터 드라이버 아웃풋의 기울기 slope 를 결정합니다. 위와 동일하게 세팅합니다.	0... 3	RW
177	쇼트 방지 비활성화	0: GND 로의 쇼트방지 on 1: GND 로의 쇼트방지 비활성화 디폴트값을 사용하십시오!	0/1	RW
178	쇼트 탐지 타이머	0: 3.2µs 1: 1.6µs 2: 1.2µs 3: 0.8µs 디폴트 값을 사용하십시오!	0..3	RW
179	V 센스	모터 전류 범위를 선택합니다. 두개의 범위가 있습니다. 0 : 모터 전류 범위에서 최대 2.8A RMS 1 : 모터 전류 범위에서 최대 1.5A RMS 각각의 범위 내에서 모터 전류 32 스탭으로 감소 될 수 있습니다 (축 파라미터 6 + 7)		
180	smartEnergy 실제 전류	상태값은 coolStep 에 의해 제어되듯이 실제 모터 전 류 세팅을 제공합니다. 값은 SEIMIN 에 의해 특정 되듯 CS 값으로 올라가고 CS 의 한 부분으로 내려갑니다. 실제 모터 전류 스케일링 요인: 0 ... 31: 1/32, 2/32, ... 32/32	0... 31	RW
181	스톨에서 정지	이 속도 이하에서 모터는 정지하지 않습니다. 이 속도 위에서 stallGuard2™부하 값이 0 에 도달할 경우 모터는 정지합니다	-8.388.608 ... +8.388.607 [pps]	RW

182	smartEnergy 스레시홀드 속도	이 속도 이상에서 coolStep™이 활성화 됩니다		-8.388.608 … +8.388.607 [pps]	RW
183	smartEnergy 느린 작동 전류	스레시 홀드 속도 아래에서 사용되는 모터전류를 세팅합니다.		0… 255 축 파라미터 6 과 같은 스케일링	RW
193	엔코더 Z 상 서치	엔코더 Z 상을 탐색합니다. 9- CW 방향 10- CCW 방향		9/10	RW
195	엔코더 Z 상 서치 속도	엔코더 Z 상 탐색 속도를 세팅합니다.		0 … 51200	RW
200	부스트 전류	가속과 감속 단계에서 사용되는 전류. 0 으로 세팅시 Axis 파라미터 6 에 의해 세팅된것과 같은 전류. 0 으로 세팅시 Axis 파라미터 6 에 의해 세팅된것과 같은 전류가 사용됩니다.		0… 255 Axis 파라미터 6 과 같은 스케일링	RW
206	실제 부하 값	스톨감지에 사용되는 실제 부하값 읽기 (stallGuard2™)		0… 1023	R
208	TMC262 드라이버 에러 깃발	Bit 0	stallGuard2 상태 (1: 도달한 스레시 홀드)	0 … 255	R
		Bit 1	과 온도 (1: 과 온도로 드라이버가 꺼짐)		
		Bit 2	과 온도 경고 (1: 스레시홀드 초과)		
		Bit 3	ground A 로 쇼트 (1: 감지된 쇼트 환경, 드라이버는 꺼져있음),		
		Bit 4	Short to ground B (1: 감지된 쇼트 환경, 드라이버는 꺼져있음)		
		Bit 5	오픈 부하 A (1:지속적인 코일 극성의 마지막 주기동안 쵸퍼 이벤트가 발생하지 않습니다)		
		Bit 6	오픈 부하 B (1: 지속적인 코일 극성의 마지막 주기동안 쵸퍼) 이벤트가 발생하지 않습니다)`		
		Bit 7	정지 (1: 최종 2^20 클락 사이클동안 스텝 입력에서 스텝 충격이 발생하지 않습니다)		
209	엔코더 위치	엔코더 카운터 값		-2.147.483.648 … +2.147.483.647	RW
210	엔코더 선택/ 분해능	0 : 내부 온-보드 홀 센서 엔코더가 선택됩니다 (디폴트). 그렇치 않다면: A/B/N 외부 엔코더 커넥터에 연결된 외부 엔코더의 분해능을 특정하십시오. 분해능은 모터 회전당 혹은 모터회전 x 당 엔코더 라인으로 특정됩니다			RW

212	최대 위치 편차	POS_ERROR 아웃풋 전용을 위한 위치편차 윈도우를 환경설정 합니다		0 ... +2.147.483.647	RW
213	최대 속도 편차	POS_ERROR 아웃풋 전용을 위한 속도편차 윈도우를 환경설정 합니다		0 ... +2.147.483.647	RW
215	절대적 엔코더 위치	앱솔루트 엔코더의 위치		0 ... 16383	RW
254	Step/Dir Mode	0	일반 Mode. Step/dir Mode off.	0,1,2,3,5,6,7	RW
		1	스텝과 방향 인풋 활성화. 낮음에서 높음으로 이행하는 스텝		
		2	스텝과 방향 인풋 활성화, 높음에서 낮음으로 이행하는 스텝		
		3	스텝과 방향 인풋 활성화, 양쪽 방향 이행하는 스텝		
		5	스텝과 방향 인풋 활성화, Step on Low-to- 낮음에서 높음으로 이행하는 스텝, 도치된 방향 인풋		
		6	스텝과 방향 인풋 활성화. 높음에서 낮음으로 이행하는 스텝 도치된 방향 인풋		
		7	스텝과 방향 인풋 활성화. Step on both 양쪽 이행하는 스텝, 도치된 방향 인풋		

5.1 stallGuard2

모듈에 TMC262모터 드라이버 칩이 장착됩니다. TMC262은 스톨 감지를 위해 사용되는 부하측정의 특징을 가지고 있습니다. stallGuard2는 스톨감지 신호뿐 아니라 모터에 센서 없이 부하측정 또한 가능합니다. 측정된 값은 부하, 속도, 그리고 전류세팅의 넓은 범위에서의 모터에 대한부하와 함께 리니어를 변경합니다. 최대 모터부하에서 stallGuard2의 값은 0이 됩니다. 이는 스테이터의 마그네틱 필드와 로터의 마그넷 사이의 90°부하각에 일치합니다. 이는 또한 모터작동의 가장 에너지 효율이 높은 부분입니다.

스톨 감지는 부하가 너무 높을 시 모터가 정지하는 것을 의미합니다. 축파라미터 #174에 의해 환경설정을 할 수 있습니다.

5.2 축 파라미터와 연관된 coolStep

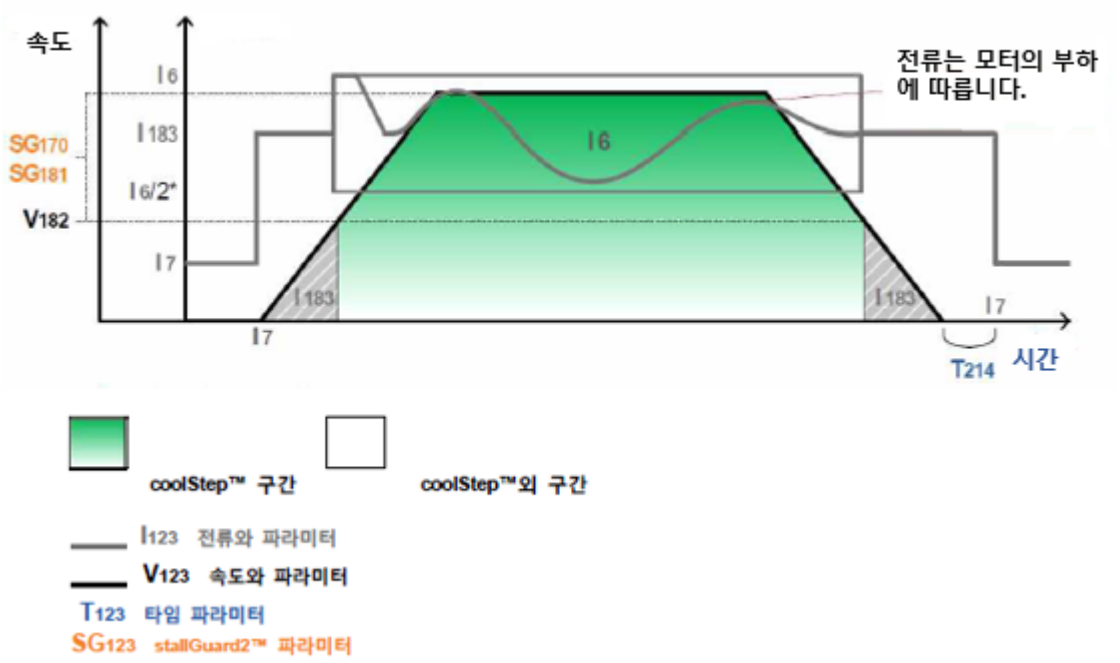
밑의 그림은 파라미터와 연관된 coolStep의 개요를 보여줍니다. 아래의 그림은 오직 한 가지의 예시만을 보여줍니다. 전류의 환경설정에 대한 파라미터들입니다. 다른 파라미터들은 속도 제한이나 시간 조정을 위한 것입니다.

따라오는 조정이 반드시 필요합니다:

- 전류 (I6, I7, I183)와 속도(V182)를 위한 스레시홀드는 확실히 세팅 되어야 합니다
- stallGuard2 특징은 파라미터 SG170 과SG181 를 통해 조정되고 활성화되어야 합니다.
- CoolStep (부하에 따라)에서 전류의 감소와 증가는 파라미터 I169 과 I171 로 환경설정 되어야 합니다.

이 챕터에서 coolStep 과 stallGuard2에 대한 기본 파라미터가 언급 되었습니다.챕터3에서 Axis 파라미터의 리스트에 더 많은 환경설정이 가능한 파라미터에 대한 정보를 제공하고 있습니다

coolStep™ 조정포인트와 스레시홀드



***coolStep 전류의 낮은 스레시홀드는 I6/4까지 조정 가능합니다. 파라미터 168을 참조하세요.**

그림 5.1: coolStep 조정 포인트와 스레시홀드

넘버	축 파라미터	설명
I6	절대 최대 전류 (CS/전류 스케일)	최대 값은 255 입니다. 이 값은 모듈 최대 전류의 100%를 의미합니다. 전류 조정은 0….255 안의 범위이고 32 스텝안에서 조정 가능합니다 (0…255/8 예: 스텝 0=0…7, 스텝 1=8….15) 너무 높은 값은 모터 데미지를 유발하기 때문에 가장 중요한 모터세팅입니다
I7	대기 전류	모터가 정지한 후 전류를 이 초 제한합니다.
I168	smartEnergy 최소 전류 (SEIMIN)	CS 값 스케일링에 의해 작동하는 coolStep 을 위해 낮은 모터 전류제한을 세팅합니다. 최소 모터 전류: 0 –CS 의 ½. 1 – CS 의 ¼.
I169	smartEnergy 전류 다운스텝	모터 전류의 각각의 감소에 필요한 상위 스레시홀드 이상에서의 stallGuard2™ 의 수를 세팅하십시오. 각 감소마다 stallGuard2™의 넘버:: 스케일링: 0… 3: 32, 8, 2, 1 0: 느린 감소 3: 빠른 감소
I171	smartEnergy 전류 업 스텝	전류 증가 스텝을 세팅하시오. 전류는 낮은 스레시홀드(smartEnergy 히스테레시스 참조) 이하의 측정된 stallGuard2™값에 대해 감소합니다. 전류 증가 스텝 사이즈: 스케일링: 0… 3: 1, 2, 4, 8 0: 느린 증가 3: 빠른 증가/ 부하증가에 따른 반응
I186	smartEnergy 느린 작동 전류	스레시 홀드 속도보다 느리게 사용되는 모터 전류를 설정합니다. Axis 파라미터 182 로 스레시홀드 속도를 조정하십시오.
SG170	smartEnergy 히스테레시스	stallGuard2™읽기를 위해 낮은 스레시홀드와 높은 스레시홀드 사이의 거리를 세팅 하십시오. 높은 스레시홀드 이상에서 모터전류는 감소합니다.
SG181	스톨에서 정지	이 속도 이하에서 모터는 정지하지 않습니다. 이 속도 이상에서 stallGuard2™부하값이 0 에 도달 시 모터는 정지합니다.
V182	smartEnergy 스레시홀드 속도	이 속도 이상에서 coolStep™은 활성화됩니다.
T214	파워 다운 딜레이	전류가 대기전류로 바뀌어 내려가기 전 정지기간. 기본값은 200 입니다 (2000m/sec 와 동일한 값입니다).

5.3 Closed Loop 활성화하기

엔코더 기반의 통합된 홀 센서 혹은 외부 증폭 A/B/N 엔코더와 함께 EDB 시리즈를 Closed Loop 스텝 드라이브/ 컨트롤러로 사용하는 것이 가능합니다.
축 파라미터 210을 외부 엔코더의 분해능에 세팅하는 것은 외부 엔코더를 활성화 시키고 엔코더 기반 내부 홀 센서를 비활성화 시킵니다 (챕터5를 참조하십시오).

5.3.1 Closed Loop 기본 세팅 예시

예시 프로그램 1

한바퀴에 10,000step 증가하는 엔코더가 연결되어 있으며 Closed Loop 세팅 후 정회전 10바퀴 역회전 10바퀴 반복 실행.

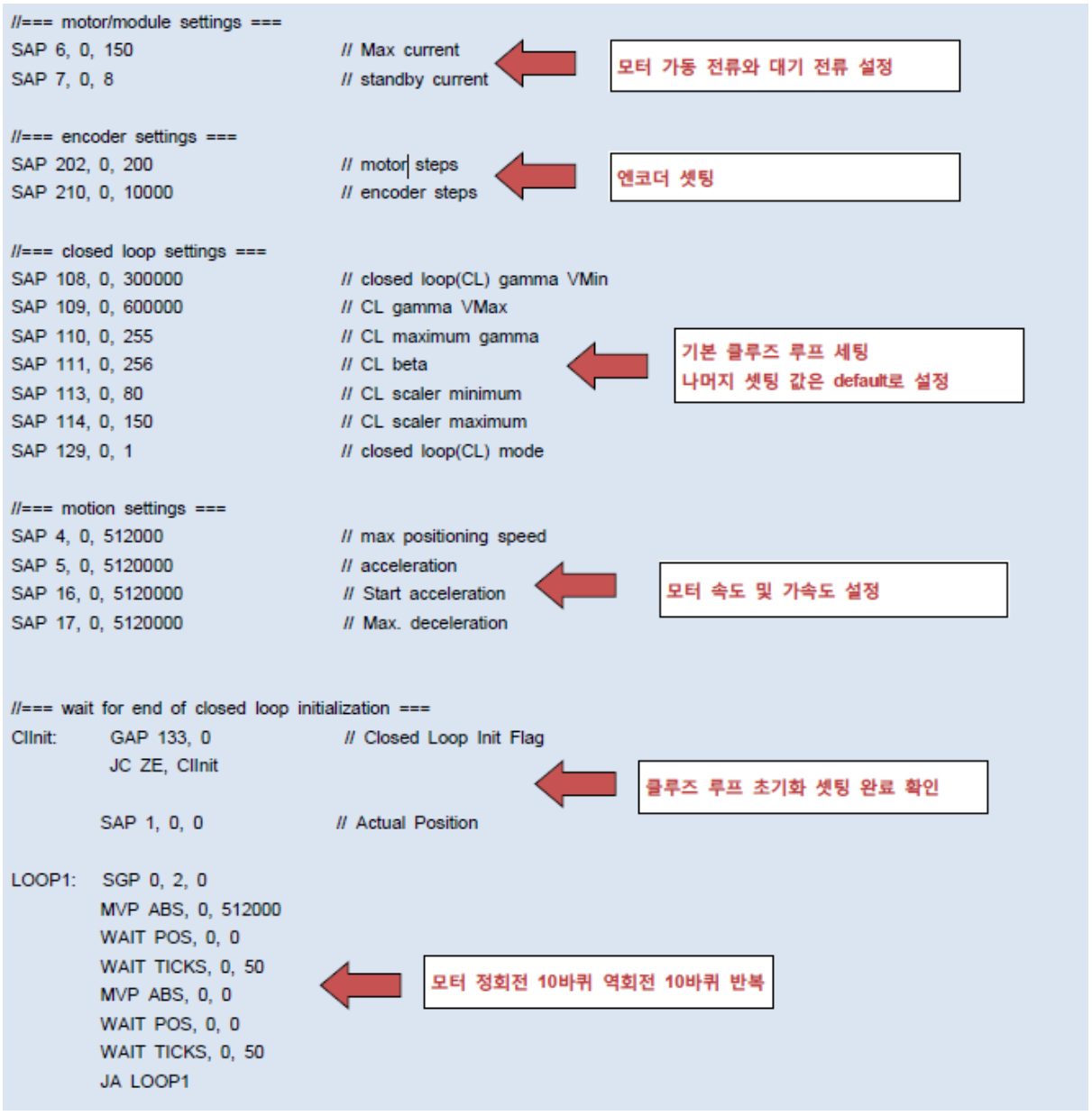


그림 5.2 EDB 시리즈 Closed Loop 기본 세팅

5.3.2 Closed Loop 파라미터

본 제품의 2상 스텝모터 Closed Loop 제어는 스텝 모터 드라이브의 특징을 고려하여 개발된 기존의 PID 제어와는 다른 방향을 따릅니다. 타겟 위치와 속도를 결정하는 램프 생성기와 전류제어, 그리고 위치 제어 (통신 각도 제어)는 각각 독립되어 있습니다. Closed Loop 제어 운영 체제는 아래의 그림에 묘사되어 있습니다.

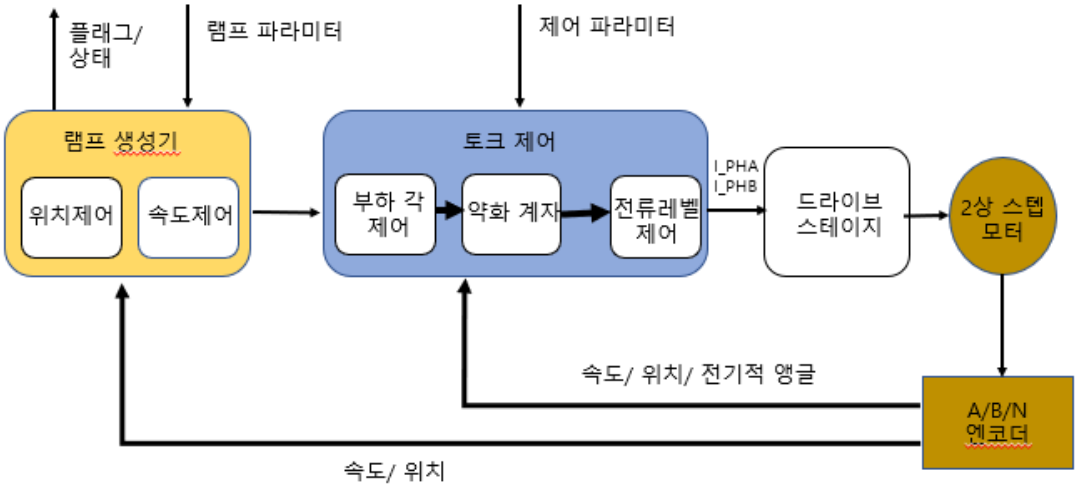


그림 5.3 Closed Loop 제어 운영 체제

부하 각 제어와 전류 레벨 제어는 평행식으로 실행됩니다.

- 부하 각 제어

  전형적인 스텝모터 드라이브와 같이 각 상의 전류는 모터 드라이브로 직접 지정됩니다. 이는 로터에 의해 형성되는 전류 벡터의 결과가 됩니다. 로터의 위치는 엔코더의 피드백에 의해 정해집니다. Closed Loop 모터 제어는 부하의 각도를 모니터 하게됩니다 (드라이브 스테이지 전류 벡터와 엔코더의 각도 사이의 편차). 게다가, 전류 벡터의 방향성은 만약 부하 각도가 특정 한계를 초과하려고 한다면 로터의 위치를 따릅니다. 결국 주어진 한계를 절대 초과하지 않는 부하 각도가 될 것이고 따라서 탈조 현상이 발생하지 않습니다. 따라서 전류 벡터는 부하가 줄어들기 전까지는 더 많은 전류를 공급하게 됩니다. 그림 5.4에서는 부하 각을 제한하는 파라미터를 보여줍니다.

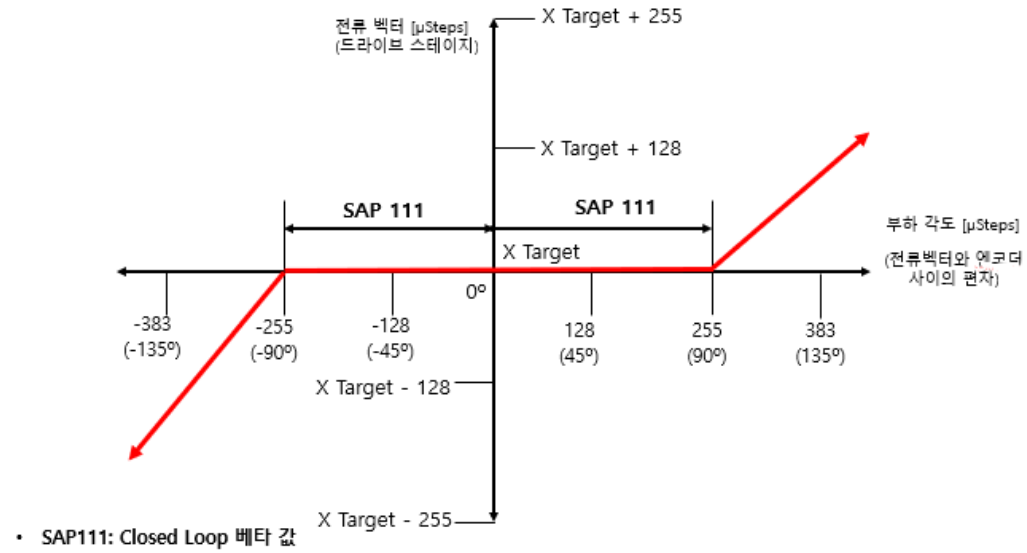


그림 5.4 부하 각도 제어 파라미터

- 전류 레벨 제어

부하 각도 컨트롤과는 평행적으로 EDB 시리즈에서는 가벼운 부하나 부하가 없을 시 에너지 절감을 위해 부하 각도에 따라서 모터 전류 레벨을 제어합니다 (전류 벡터 증폭). 그림 5.5에서는 전류 제어 파라미터의 개요를 보여줍니다.

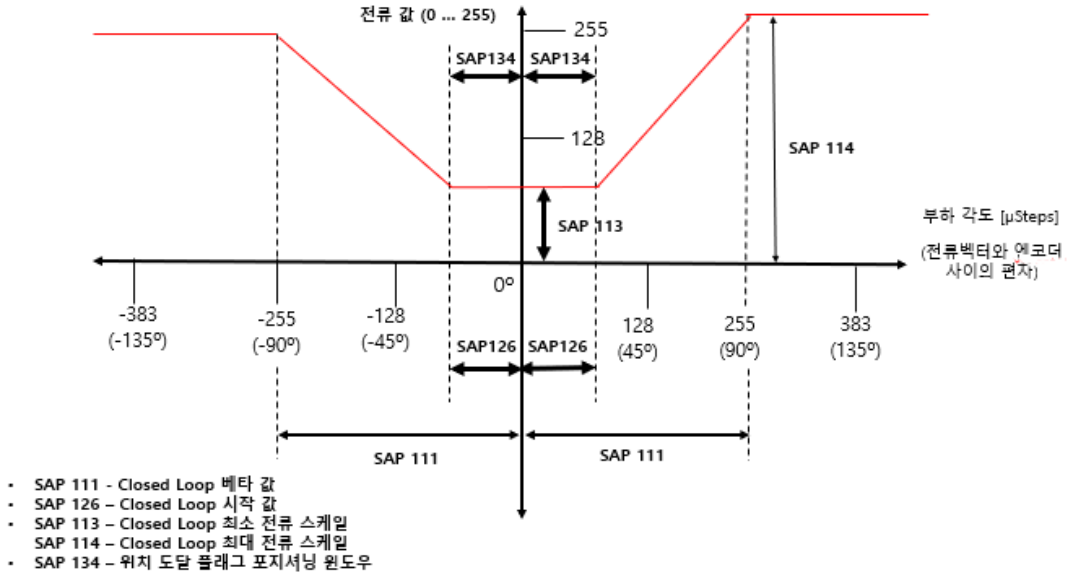


그림 5.5 전류 레벨 제어

파라미터 SAP 120과 121은 얼마나 빠르게 실제 전류가 상승하거나 하강하는지에 대한 딜레이를 세팅하고 그래프의 빨간 선을 따릅니다.

- 약화 계자 (Field Weakening)

EDB 시리즈는 모터의 역기전력으로 인해 전류를 유지하기 어려운 속도에 도달 가능합니다. 이 속도 위에서 부하각 (SAP 111, 디폴트 90°)과 전류 레벨 제어는 최대치에 도달합니다. 스텝 모터를 더 빠른 속도로 돌리기 위하여 역기전력은 반드시 정류 각도 90-180° 사이에서 보상되어야 합니다. 약화 계자 (Field Weakening)를 위한 파라미터는 그림 5.6에 설명되어 있습니다.

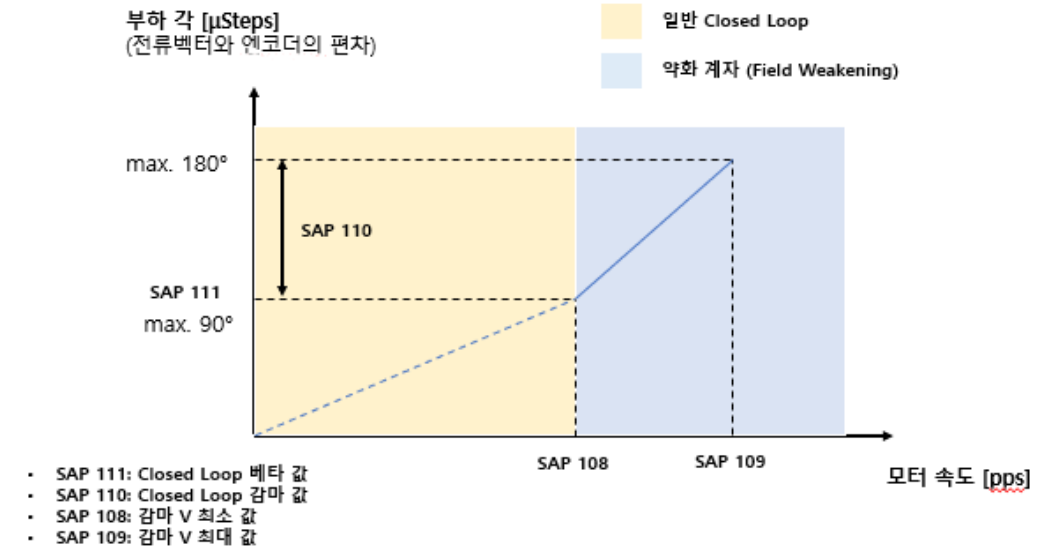


그림 5.6 약화 계자 (Field Weakening)

- 위치 보상

EDB 시리즈는 Closed Loop 포지셔닝을 위한 특별한 기능을 가지고 있습니다. 속도와 가/감속과 같은 포지셔닝 파라미터는 주어진 시간안에 목표 위치에 도달하기 위해 계산됩니다. 만약 타겟 사다리꼴 램프에서 과 부하로 인해 유지되지 어렵다면 EDB 시리즈는 위치가 제 시간 안에 도달하게 하는 특별한 기능을 가지고 있습니다.

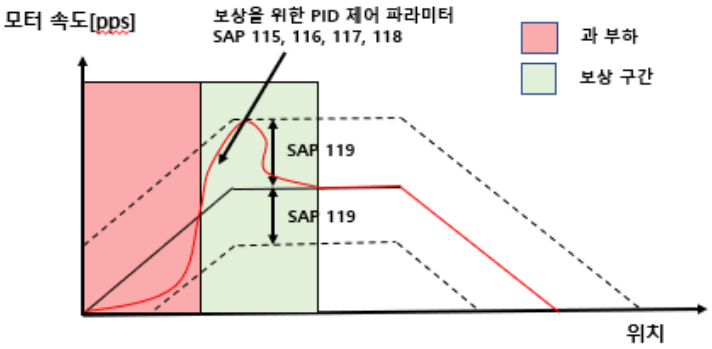


그림 5.7 위치 보상

6. 글로벌 파라미터 (Global Parameter)

글로벌 파라미터는 4개의 뱅크들로 그룹지어집니다.

- Bank 0 (글로벌 환경설정)
- Bank 1 (유저C변수)
- Bank 2 (유저 EMCL 변수)
- Bank 3 (환경설정 인터럽트)

글로벌 파라미터를 쓰거나 읽기 위해서 SGP와 GGP 명령을 사용하십시오.

6.1 Bank 0

64번부터의 번호를 가진 파라미터들은 모듈의 시리얼 주소나 RS485 통신 속도와 같은 환경 설정을 합니다. 필요에 맞게 이러한 파라미터들을 변경할 수 있습니다. 가장 쉽고 좋은 방법은 EMCL-IDE의 적절한 기능을 사용하는 것입니다. 64와 128 사이의 번호를 가진 파라미터들은 EEPROM에만 저장됩니다.

주의:

이러한 파라미터의 SGP명령은 항상 영구적으로 저장되고 STGP 명령을 따로 필요로 하지 않습니다. 이러한 파라미터들을 변경 시 주의를 기울여야 하고 상호적으로 사용하기 위해 EMCL-IDE의 적절한 기능을 사용하십시오.

칼럼 안 문자의 의미

Access 타입	연관된 명령	설명
R	GGP	읽기 가능한 파라미터
W	SGP, AGP	쓰기 가능한 파라미터
E	STGP, RSGP	리셋이나 파워를 킨 후 파라미터는 자동적으로 EEPROM으로부터 복구됩니다. 이러한 파라미터는 STGP를 사용하여 EEPROM에 영구적으로 저장 가능하고 또한 RSGP를 사용하여 확실하게 복구됩니다. (EEPROM에서 RAM으로 복사).

글로벌 파라미터 (Bank 0)

넘버	파라미터	설명			범위	Access
64	EEPROM magic	전원을 입력 후 $E4 와 같은 다른 값에 이 파라미터를 세팅 하는 것은 Axis 과 Grobal 파라미터 (초기 디폴트로)의 재 초기화의 원인이 됩니다.			0 ...255	RWE
65	RS485 통신속도	0	9600 baud	default	0 ... 11	RWE
		1	14400 baud			
		2	19200 baud			
		3	28800 baud			
		4	38400 baud			
		5	57600 baud			
		6	76800 baud	Windows 지원불가!!		
		7	115200 baud			
		8	230400 baud	Windows 지원불가!!		
		9	250000 baud	Windows 지원불가!!		
		10	500000 baud	Windows 지원불가!!		
		11	1000000 baud	Windows 지원불가!!		
66	시리얼 주소	RS485 를 위한 모듈 (타겟) 주소			0 ... 255	RWE
68	시리얼 하트비트	RS485 인터페이스를 위한 시리얼 하트비트. 만약 이 시간제한이 끝나고 더 이상의 명령이 주어지지 않는다면 모터는 정지합니다.			[ms]	RWE
72	IO Mode	전용 IO 를 활성화합니다. [Value Bit 0]: ALARM_OUT 을 활성화. [Value Bit 1]: POS_REACHED 를 활성화. [Value Bit 2]: POS_ERROR 를 활성화 디폴트 모든 전용 아웃풋이 활성화됩니다. 디폴트 값= 7			0...7	RWE
73	환경설정 EEPROM 잠금 플래그	쓰기: EEPROM 을 잠그기 위해 1234 EEPROM 을 열기 위해 4321 읽기: 1=잠긴 EEPROM , 0=열린 EEPROM.			0/1	RWE
75	텔레그램 휴지기간	RS 485 를 통해 응답이 보내지기 전 휴지기간. RS 485 에서 15 로 세팅하는 것이 필요합니다. (RTS 핀에 의해 제어 되는 RS485 용 아답터).			0... 255	RWE
76	시리얼 Host 주소	RS485 를 통해 전달되는 응답 텔레그램에서 사용되는 Host 주소			0... 255	RWE
77	자동시작 Mode	0: 파워 업 (디폴트) 후 EMCL 어플리케이션을 시작하지 마십시오 1: 파워 업 후 EMCL 어플리케이션을 자동으로 시작.			0/1	RWE

80	드라이버 활성화	전용 활성화 인풋을 환경설정합니다. 0: 활성화 인풋(디폴트)를 비활성화 합니다 1: 저- 액티브 2: 고-액티브	0,1,2	RWE
81	EMCL 코드 보호	분해나 오버 라이팅으로부터 EMCL 프로그램을 보호합니다. 0 – 보호없음 1 – 분해로부터 보호 2 – 오버 라이팅으로 부터 보호 3 – 분해와 오버 라이팅으로 부터 보호 분해로부터 보호 신호를 끈다면 프로그램은 첫번째로 지워집니다. 1 이나 3 에서 0 이나 2 로 이 값을 바꾸면 EMCL 프로그램은 삭제됩니다.	0,1,2,3	RWE
84	좌표 저장	0 – 좌표는 RAM 에만 저장됩니다. (하지만 RAM 과 EEPROM 사이에서 전제적으로 복사될 수 있습니다). 1.. – 좌표는 항상 EEPROM 에만 저장됩니다	0/1	RWE
87	시리얼 두번째 주소	RS485 를 위한 두번째 모듈 (타겟)주소	0… 255	RWE
128	EMCL 어플리케이션 상태	0 –정지 1 – 동작 2 – 스텝 3 – 리셋	0… 3	R
129	다운로드 Mode	0 – 일반 Mode 1 – 다운로드 Mode	0/1	R
130	EMCL 프로그램 카운터	현재 실행중인 EMCL 명령의 인덱스		R
132	Tick 타이머	1 밀리세컨드 당 증가하는 32 비트 카운터. 어떠한 시작 값으로 리셋이 가능합니다.	0… 232	RW
133	랜덤 넘버	랜덤 넘버를 선택합니다	0… 2.147.483.647	RW

6.2 Bank 1

글로벌 파라미터 뱅크 1은 일반적으로 유효하지 않습니다. 이는 펌웨어의 유저 특정 확장판을 위해 사용될 것입니다. 유저의 정의 가능한 명령과 함께 이러한 변수는 EMCL 어플리케이션과 확장판 사이의 인터페이스를 구성합니다.

6.3 Bank 2

뱅크 2는 EMCL 어플리케이션에서 사용되기 위해 다 목적 32 비트 변수를 포함합니다. RAM에 위치하고 처음 56 변수들은 EEPROM에도 영구적으로 저장될 수 있습니다. 부팅 후 그 값들은 자동적으로 RAM에 복구됩니다. 256 유저 변수가 가능합니다.

칼럼 안 문자의 의미

Access 타입	연관된 명령	설명
R	GGP	읽기 가능한 파라미터
W	SGP, AGP	쓰기 가능한 파라미터
E	STGP, RSGP	리셋이나 파워를 킨 후 파라미터는 자동적으로 EEPROM으로부터 복구됩니다. 이러한 파라미터는 STGP를 사용하여 EEPROM에 영구적으로 저장 가능하고 또한 RSGP를 사용하여 확실하게 복구됩니다. (EEPROM에서 RAM으로 복사).

EMCL 어플리케이션 (Bank 2)을 위한 다 목적 변수

넘버	글로벌 파라미터	설명	범위	Acc
0 ...55	다 목적 변수 #0...#55	EMCL 어플리케이션에서 사용	-2.147.483.648 ... +2.147.483.647	RWE
56 ...255	다 목적 변수 #56...#255	EMCL 어플리케이션에서 사용	-2.147.483.648 ... +2.147.483.647	RW

7. 힌트와 팁

이 챕터에서는 EMCL의 기능을 사용하는데 있어 몇가지의 힌트와 팁을 줍니다.

예) 내부 설치되어 있는 리퍼런스 포인트 서치 알고리즘을 어떻게 사용하고 파라미터화 하는지.

7.1 RS485 인터페이스 사용하기

PC의 COM 포토에 부착 가능한 대부분의 RS485 컨버터와 함께 데이터 방향은 COM 포트의 RTS 핀에 의해 제어됩니다. 윈도우 2000, XP 혹은 NT4에서 작동됩니다. 작동 시스템의 버그 때문에 윈도우 95, 98이나 윈도우 ME에서는 작동하지 않습니다. 또 다른 문제로 윈도우 2000/XP/NT4는 수신을 위해 방향을 변경하는 것이 너무 느립니다. 이 문제를 해결하기 위해 Direct 모드에서 SGP 75, 0, 15 를 입력함으로 모듈의 텔레그램 휴지 기가 (글로벌 파라미터 #75)을 15 (필요시 이상)로 세팅합니다. 파라미터는 EEPROM 환경설정에 자동적으로 저장됩니다.

8. EMCL 프로그래밍 테크닉과 구조

8.1 초기화

EMCL 프로그램 (다른 프로그램에서와 같이) 의 첫번째 일은 디폴트 세팅과 다른 값들이 필요한 곳의 파라미터들을 초기화 하는 것 입니다. 이 목적을 위해 SAP 와 SGP 명령이 사용됩니다.

8.2 메인 루프

내장형 시스템은 일반적으로 무한하게 실행되는 메인 루프를 사용합니다. 이는 또한 Stand-Alone에서 실행되는 EMCL어플리케이션의 경우와 같습니다. 일반적으로 모듈의 자동 시작 모드는 반드시 on되어 있어야 합니다. 파워를 켠 후 모듈은 처음으로 필요한 모든 초기화를 하는 EMCL프로그램을 시작하고 그 후 메인 루프로 들어가게 됩니다.

예외가 한가지 있습니다, EMCL 루핀이 Direct모드에서 호스트로 불러오기 되었을 때.

대부분의 Stand-Alone EMCL 프로그램은 아래와 같습니다 (전체는 아닙니다).

```
Initialization:
        SAP 4, 0, 500     //최대 속도 설정
        SAP 5, 0, 100     // 최대 가속 설정
Mainloop:
        //이 예시는 두 위치를 왔다갔다하는 동작입니다.
        MVP ABS, 0, 5000
        WAIT POS, 0, 0
        MVP ABS, 0, 0
        WAIT POS, 0, 0
        JA Mainloop         //동작 후 mainloop로 점프하여 무한 루프를 돕니다.
```

8.3 상수 사용하기

프로그램의 더 쉬운 사용을 위해 상수를 이해하기 쉬운 용어로 정의하고 이름 프로그램에서 혼동없이 쉽게 사용할 수 있습니다. EMCL-IDE는 모든 중요한 축 파라미터와 글로벌 파라미터를 위한 상수 이름이 포함된 파일을 제공합니다.

예: // 어떠한 상수를 쉬운 용어로 정의합니다.

```
MaxSpeed = 500
MaxAcc = 100
Position0 = 0
Position1 = 5000

// Initialization
        SAP APMaxPositioningSpeed, Motor0, MaxSpeed
        SAP APMaxAcceleration, Motor0, MaxAcc

MainLoop:
        MVP ABS, Motor0, Position1
        WAIT POS, Motor0, 0
        MVP ABS, Motor0, Position0
        WAIT POS, Motor0, 0
        JA MainLoop
```

다른 값을 위해 상수를 사용하는 것은 프로그램에서 한 번 이상 사용할 시 더욱 쉽게 변경할 수 있습니다. 상수의 정의를 변경할 수 있고 프로그램 안에서 이의 모든 발생을 변경할 필요가 없습니다.

8.4 변수 사용하기

유저 변수는 프로그램에서 변수가 필요할 때 사용 가능합니다. 임시 값으로 저장될 수 있습니다. SGP, GGP 그리고 AGP는 유저변수와 함께 사용될 수 있습니다.

SGP는 정수 값으로 변수를 세팅하기 위해 사용할 수 있습니다 (예: 초기화 단계동안)
GGP는 유저 변수의 정수를 읽거나 이를 누적 변수 레지스터로 복사하는데 사용할 수 있습니다.
AGP는 누적변수 레지스터의 정수를 유저변수로 복사하는데 사용됩니다 (예: 계산결과를 저장)

예:

```
  MyVariable = 42          //유저변수는 문자형태의 이름을 사용하십시오.
  SGP MyVariable, 2, 1234   // 값 1234로 변수를 초기화 합니다.
  ..........................
  GGP MyVariable, 2        // 변수의 내용을 누적기 레지스터로 복사합니다.
  CALC MUL, 2              // 누적 레지스터에 2를 곱합니다.
  AAP MyVariable, 2        // 누적 레지스터의 내용을 변수로 저장합니다.
```

이 외에도, 이러한 변수들은 모듈에서 실행되는 EMCL 프로그램과 HOST 사이의 강력한 통신 방법을 제공합니다. 호스트는 Direct 모드 SGP 명령을 사용함으로 변수를 변경할 수 있습니다 (EMCL프로그램 Direct 모드 동작 중, 명령은 실행 프로그램의 방해 없이 여전히 실행 가능합니다). 만약 EMCL 프로그램이 주기적으로 이 변수를 폴링 한다면 내용의 이러한 변경 안에서 반응될 수 있습니다.

호스트는 또한 Direct 모드에서 GGP를 사용함으로 변수를 폴링 할 수 있고 EMCL 프로그램에 의해 변경 되었는지를 확인할 수 있습니다.

8.5 서브루틴 사용하기

CSUB와 RSUB 명령은 서브루틴을 사용하기 위한 메커니즘을 제공합니다. CSUB명령은 주어진 라벨로 분과 합니다. RSUB 명령이 실행 될 때 제어는 서브루틴을 불러온 CSUB명령을 따릅니다. 이 메커니즘은 또한 그룹을 지을 수 있습니다. CSUB 명령에 의해 서브루틴으로부터 다른 서브루틴을 불러올 수 있습니다. EMCL의 최신 버전에서 그룹이 지어진 서브루틴을 8 레벨까지 허용합니다.

8.6 Direct 모드와 Stand-Alone 모드 혼합하여 사용하기

Direct 모드와 Stand-Alone 모드를 혼합하여 사용할 수 있습니다. Stand-Alone 모드에서 EMCL 프로그램이 실행될 때 Direct 모드 명령 또한 진행이 가능합니다 (그리고 Stand-Alone 모드에서 작동하는 프로그램의 흐름을 방해하지 않습니다). 따라서 쿼리 또한 가능합니다. 예: EMCL 프로그램이 작동하는 동안 Direct 모드에서 모터의 실제 위치.

Stand-alone Mode에서 작동하는 프로그램과 Host 사이의 통신은 EMCL유저변수를 사용하여 쓸 수 있습니다. Host는 EMCL프로그램에 의해 주기적으로 폴링되는 유저 변수의 값을 변경 할 수 있고 (Direct Mode SGP명령을 사용하여) 따라서 EMCL™프로그램은 이러한 변경에 반응 할 수 있습니다. 역으로, EMCL프로그램은 Host에 의해 폴링되는 유저 변수를 변경 가능합니다 (Direct Mode GGP명령을 사용).

EMCL프로그램은 Direct Mode에서 작동명령을 사용하는 Host에 의해 시작될 수 있습니다. 이 방법으로, EMCL루틴들은 Host에 의해 불러오기 되었다고 정의할 수 있습니다. 이러한 경우 특정 루틴으로 점프하는 JA명령을 EMCL프로그램의 시작에 놓는 것을 추천합니다. EMCL루틴 바뀌었을 경우라도 루틴의 엔트리 주소는 바뀌지 않는 것을 보장합니다. (EMCL루틴을 변경할 시 Host 프로그램은 바뀔 필요가 없습니다).

예:

```
// EMCL 루틴으로 점프하기

Func1: JA Func1Start
Func2: JA Func2Start
Func3: JA Func3Start

  Func1Start: MVP ABS, 0, 1000
              WAIT POS, 0, 0
              MVP ABS, 0, 0
              WAIT POS, 0, 0
              STOP

  Func2Start:  ROL 0, 500
               WAIT TICKS, 0, 100
               MST 0
               STOP

  Func3Start:  ROR 0, 1000
               WAIT TICKS, 0, 700
               MST 0
                STOP
```

이 예시는 세가지 간단한 EMCL루틴을 제공합니다. 이들은 첫번째 기능을 불러오기 위해 주소0과 함께 작동명령을 시킴으로써 Host로부터 불러오기 될 수 있거나 두번째 기능을 불러오기 위해 주소1과 함께, 혹은 세번째 기능을 불러오기 위해 주소2와 함께 작동명령을 실행 할 수 있습니다. EMCL-IDE의 Generate symbol file 기능을 사용함으로 EMCL 라벨 (작동명령을 위해 필요한)의 주소를 볼 수 있습니다. EMCL-IDE에 관한 더 자세한 정보는 EMCL-IDE유저 매뉴얼을 참조하십시오.