Smarter. Greener. Together.

Delta Electronics(Korea), Inc.
서울시 금천구 가산디지털 1 로 219
(가산동),1511 호
Tel:(02)515-5303
Fax:(02)515-5302
http://www.deltaww.com/

Delta EtherCAT 서보 프로그래밍 가이드

Delta EtherCAT
프로그래밍 가이드

www.deltaww.com

Smarter. Greener. Together.

서문

저희 제품을 사용해주셔서 감사합니다. 본 사용 매뉴얼은 EtherCAT 관련 정보를 제공합니다. 본 매뉴얼은 다음 내용을 포함합니다

- API 라이브러리 소개
- EtherCAT 관련 설명
- EtherCAT 조작 예제
- API 제어
- Master Configure API 설명
- Master Initial API 설명
- Slave 공통 API 설명
- Motion Slave 공통 API 설명
- Motion Slave CSP API 설명
- Motion Slave CSV API 설명
- Motion Slave CST API 설명
- Motion Slave Home API 설명
- Motion Slave PP API 설명
- Motion Slave Velocity API 설명
- Motion Slave VL API 설명
- Motion Slave Torque API 설명
- Motion Slave User API 설명
- DIO Slave API 설명
- AIO Slave API 설명
- 5621 Slave API 설명
- x62x Slave API 설명
- Delta Servo Slave API 설명
- 8124 Slave API 설명
- 9144 Slave API 설명
- Slave Record Data API 설명
- Master 축 카드 전용 API 설명
- Master Compare API 설명
- DLL 관련 API 설명
- Master User Security API 설명
- PAC MRAM 전용 API 설명
- 70E2 Slave API 설명
- 70X2 Slave API 설명

- 5614 Slave API 설명
- 명령 반환값과 메시지 설명
- DMCNET 및 함수 리스트
- 이전 버전 API 테이블

EtherCAT 제품 특징

델타가 개발한 차세대 Ethernet 제어 자동화 기술(EtherCAT)은 Ethernet 을 기본으로 한 필드버스입니다. 본 제품은 Cycle Time 이 짧은 특징을 바탕으로 고성능 분산형 I/O 시스템이 될 수 있습니다. EtherCAT 은 양방향(full-duplex) Ethernet 물리층을 사용하므로 슬레이브(secondary station)에 두 개 이상의 포트가 생성됩니다. 슬레이브 장치에서 기타 장치가 발견되지 않을 경우, 슬레이브의 컨트롤러는 자동으로 해당 포트를 off 하고 Ethernet 프레임을 반환합니다. 위의 특징을 토대로 EtherCAT 은 네트워크 토폴로지를 지원합니다. 그밖에 EtherCAT 은 시스템 동기화를 위한 분산형 Clock Mechanism 을 제공합니다.

본 운영 매뉴얼의 사용 방법

사용자는 본 매뉴얼을 통해 EtherCAT에 대한 기본적인 정보를 얻을 수 있습니다. 본 매뉴얼은 EtherCAT API 사용법과
간단한 예시들을 소개합니다. 프로그램을 입력하기 전에 우선 제1장과 제2장의 라이브러리 사용법과 EtherCAT 관련 설명을 숙지하시기 바랍니다.

델타 일렉트로닉스의 설명 서비스

사용 시 계속해서 문제가 발생할 경우 대리점이나 당사 고객센터에 문의 주시기 바랍니다.

목록

9 Motion Slave CSP API

18 DIO Slave API

19 AIO Slave API

20 5621 Slave API

21 X62x Slave API

1 API 라이브러리 소개

다음은 EtherCAT DLL 이 제공하는 조작 가능한 라이브러리 및 동적 연결 (DLL) 에 대한 설명입니다. 라이브러리를 호출하여 완료해야 할 각종 기능을 실행할 수 있습니다. 다음 각 절에서 라이브러리를 귀하의 개발 환경으로 불러오는 방법에 대해 소개합니다.

1.1 라이브러리의 사용

1

설치 프로그램 설치가 완료되면 "lib" 폴더에 2 개의 라이브러리가 나타납니다. 해당 라이브러리는 Visual Studio C 및 Borland 개발 환경에서 사용 가능합니다.

라이브러리	개발 환경
EtherCatDll.lib	Visual Studio C++
BCBEtherCAT_DLL.lib	Borland C++Builder 6

1.2 새로운 파일의 편집을 시작합니다

1.2.1 VC 사용

(1) Self Creation 한 프로젝트에 다음 명령을 추가합니다:
#include “EtherCat_DLL.h”
#include “EtherCat_DLL_Err.h”

(2) Visual C 개발 환경에서 Project / Setting / Link 를 선택합니다
Object / Library modules 에서 "..\lib\EtherCat_DLL.lib"를 입력합니다

(3) 설정이 완료되면 API 를 사용하여 EtherCAT DLL 조작을 시작할 수 있습니다.

1.2.2 VB 사용

사용자는 "EtherCat_DLL.bas" 와 "EtherCat_DLL_Err.bas" 2 개의 파일을 Self Creation 한 프로젝트에 추가하면 API 를 사용하여 즉각 EtherCAT DLL 을 조작할 수 있습니다.

1.2.3 VB.Net 사용

사용자는 "EtherCat_DLL.vb" 와 "EtherCat_DLL_Err.vb" 2 개의 파일을 Self Creation 한 프로젝트에 추가하면 API 를 사용하여 즉각 EtherCAT DLL 을 조작할 수 있습니다.

1.2.4 C# 사용

사용자는 "EtherCat_DLL.cs" 와 "EtherCat_DLL_Err.cs" 2 개의 파일을 Self Creation 한 프로젝트에 추가하면 API 를 사용하여 즉각 EtherCAT DLL 을 조작할 수 있습니다.
비고: C# 프로젝트 사용 시 우선 프로젝트 속성의 **"디버그" 페이지**의 **"Unmanaged 프로그램 코드 디버그 사용"** 항목에 체크해야 합니다(아래 그림과 같음). 해당 항목에 체크하지 않을 경우, 블루 스크린 등 심각한 오류가 발생할 수 있습니다.

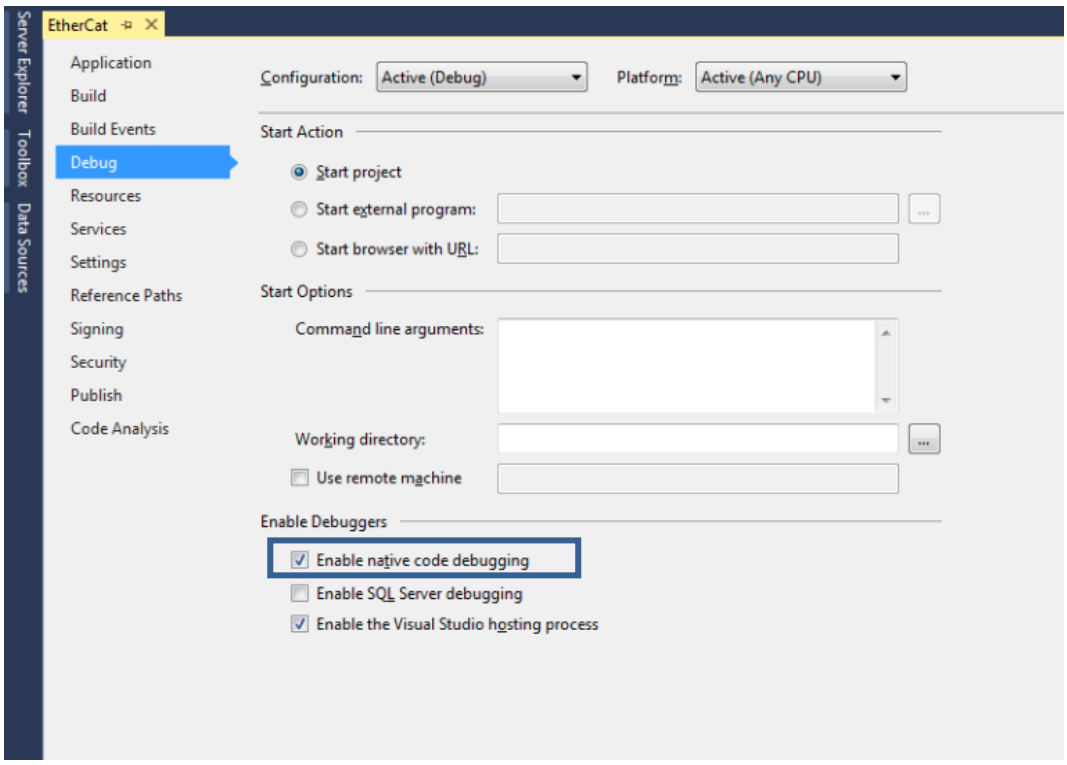


그림 1.2.4.1 C# 프로젝트 속성 설정

(이 페이지는 공란으로 비워둡니다)

1

EtherCAT 관련 설명

2

다음은 EtherCAT으로 연결 가능한 주변 장치 수량, RTX 환경 초기화 및 사용에 대한 설명입니다. 아울러 EtherCAT 관련 DLL이 RTX에서 정상적으로 작동 가능한지의 여부를 검사할 수 있는 RTSS Task Manager 사용법을 소개합니다. 아래의 순서에 따라 설명하기로 합니다.

2

2.1 주변 연결 수량의 정의

현재 주변 장치와의 연결 수량에 대한 정의는 다음과 같습니다. 일반적인 모듈에서의 연결 가능한 수량은 100 Node 이며, 속도주기를 1K (1 ms) 로 설정했을 때 연결 축이 제어하는 수량 (드라이브 및 5621 모듈 포함) 은 60축에 달합니다. 속도주기가 빠를수록 연결 가능한 축 수는 상대적으로 감소하므로 이 점을 특히 유의해야 합니다.

2.2 RTX 사용 환경의 초기화

메인 컴퓨터 부팅이 완료되어도 RTX 관련 실행 환경은 자동으로 설치되지 않습니다. 따라서 사용자가 수동으로 실행해야만 EtherCAT 관련 동작을 실행할 수 있습니다.

(1) RTX 콘솔 on

on은 "시작" > "모든 프로그램" > "IntervalZero" > "RTX 2012" > "RTX Properties"에 위치합니다.

(2) RTX 장치 사용

콘솔을 on한 후, 상단 탭을 "Control" 페이지로 전환하고, Driver의 사용 상태를 확인합니다. 상태가 전부 Stopped일 경우, "Start" 버튼을 클릭하면 RTX 관련 장치를 사용할 수 있습니다.

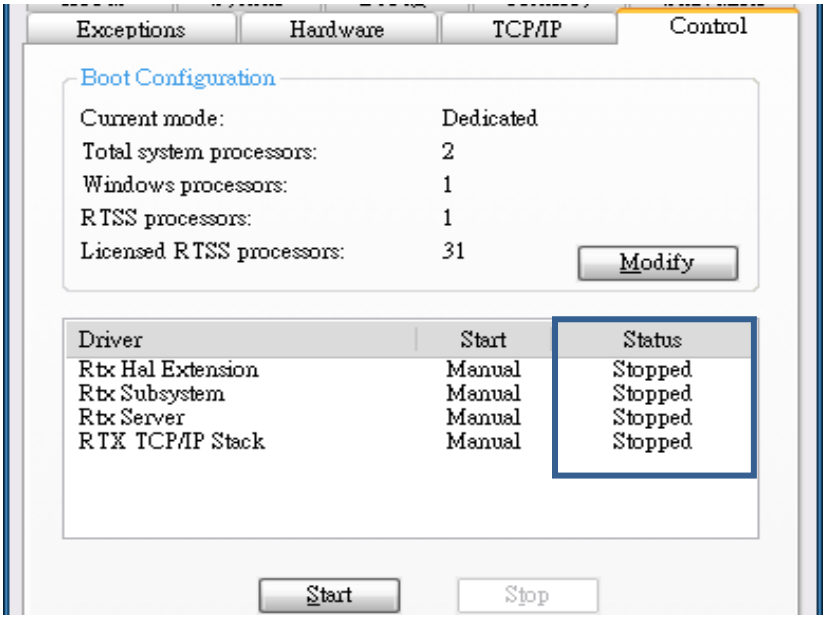


그림 2.2.1 RTX 장치 사용 중지 상태

(3) RTX 장치 작동 상태 확인

RTX 관련 장치 상태가 전부 Running으로 전환되었을 경우, EtherCAT 관련 함수를 바로 사용할 수 있습니다.

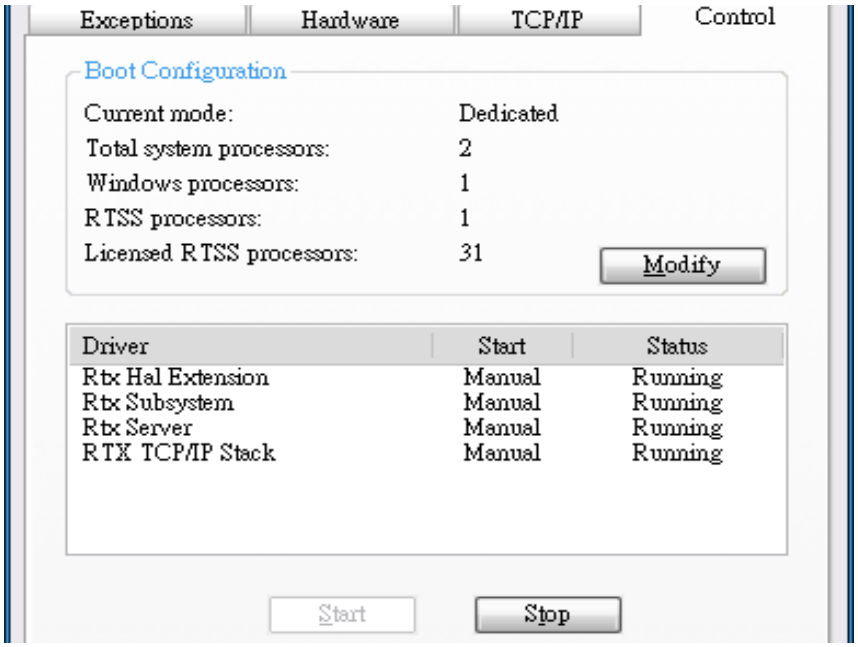


그림 2.2.2 RTX 장치 사용 상태

2

2.3 RTSS Task Manager 소개

사용자는 RTSS Task Manager를 통해 EtherCAT 관련 파일 (ECAT_RTX_RTDLL.rtdll, ECAT_STACK_RTDLL.rtdll) 이 RTX에서 정상적으로 작동하고 있는지의 여부를 확인할 수 있습니다.

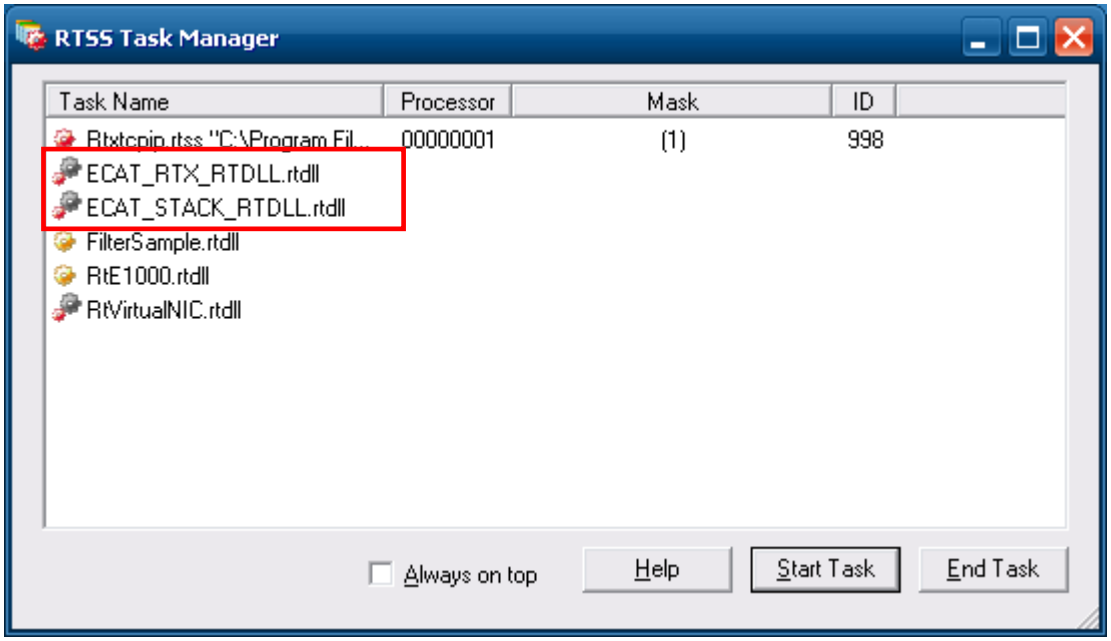


그림 2.3.1 RTSS Task Manager 인터페이스

Task Manager를 on한 후 오류가 발견될 경우, 자체적으로 파일을 다시 디버그할 수 있습니다.

(1) Start Task 버튼을 클릭하면 RtssRun 창이 나타납니다

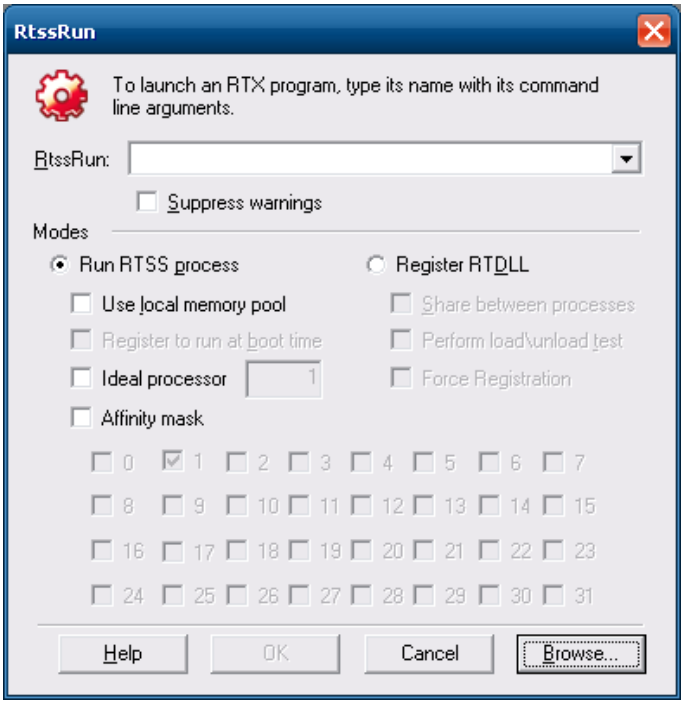


그림 2.3.2 RtssRun 창

(2) Browse를 클릭하고, 오류가 발견된 파일을 선택합니다. 해당 파일은 C:\Windows\system32\에 저장되어 있습니다.

그림 2.3.3 오류 파일 선택

(3) 파일을 선택한 후 다른 옵션은 그대로 둔 상태에서 OK를 누르면 파일을 RTX 시스템으로 즉시 디버그할 수 있습니다.

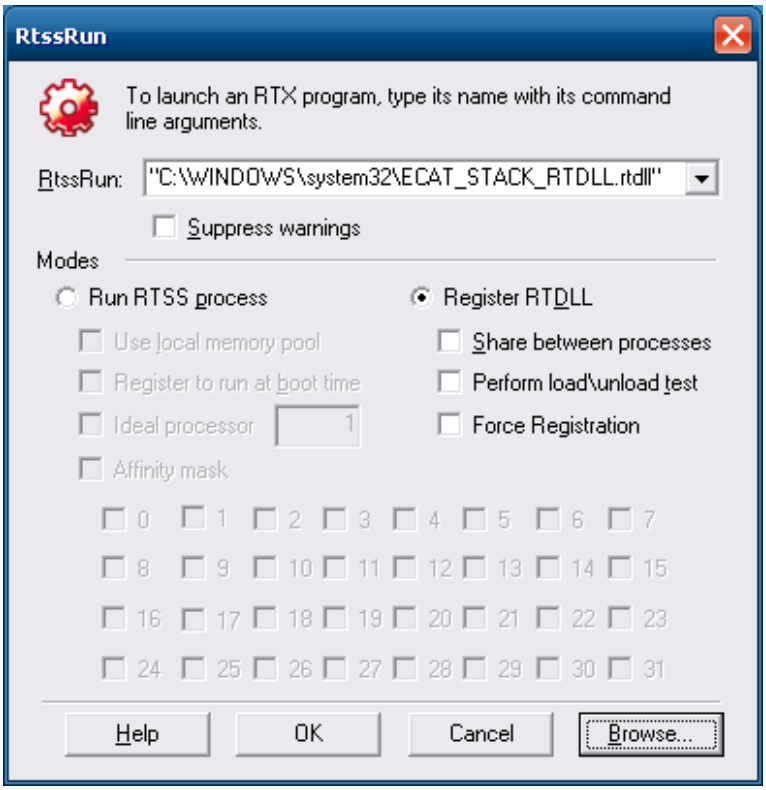


그림 2.3.4 파일 디버그

EtherCAT 조작 예제

3

다음은 EtherCAT 초기화, 원점 복귀, PT, PV, PP, CSP, EtherCAT Slave DI/O, EtherCAT AD/DA, EtherCAT Compare 및 EtherCAT SpeedContinue 관련 함수열과 예시 설명입니다.

3

3.1 EtherCAT 초기화

3.1.1 함수 표

함수의 명칭
_ECAT_Master_Open
_ECAT_Master_Get_CardSeq
_ECAT_Master_Initial
_ECAT_Master_Get_SlaveNum
_ECAT_Master_Reset
_ECAT_Master_Close
_ECAT_Master_Check_Initial_Done

■ 속성

EtherCAT RTX version	EtherCAT Card version
○	○

3

3.1.2 응용 예제

프로그램 외관

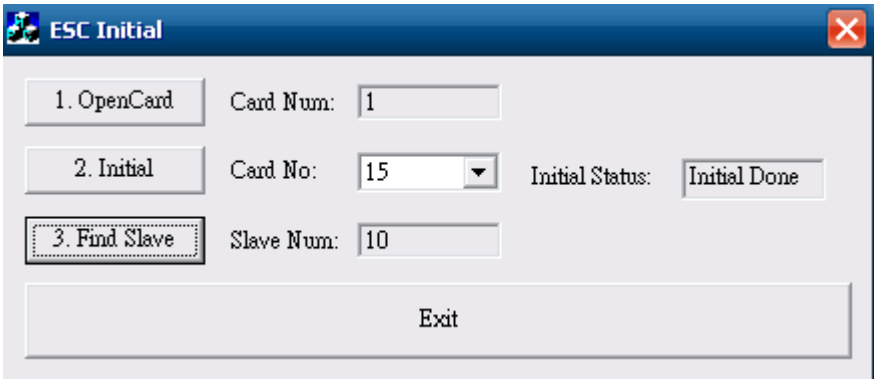


그림 3.1.2.1

(1) 인터페이스 카드 열기

그림 3.1.2.2

"OpenCard" 버튼을 클릭하여 다음 프로그램을 실행합니다:

```
/* EtherCAT 축 카드 수량에 gESCExistCards 변수를 채워 넣습니다*/
RetCode = _ECAT_Master_Open(&gESCExistCards);
```

(2) 인터페이스 카드 초기화

그림 3.1.2.3

"Initial" 버튼을 클릭하여 다음 프로그램을 실행합니다:

```
for(i=0; i< gESCExistCards; i++)
{

  /* PC에서 i번째 인터페이스 카드 번호를 확인합니다. 카드 번호는 Switch에
상응하는 수치를 나타냅니다 */
       RetCode = _ECAT_Master_Get_CardSeq(i, &CardNo);

       /*인터페이스 카드 초기화를 진행합니다 */
      RetCode = _ECAT_Master_Initial(CardNo);
      if(RetCode != 0)
      {
            strMsg.Format("_ECAT_Master_Initial, RetCode = %d", RetCode);

            MessageBox(strMsg);
      }
}
/* Initial 상태를 확인합니다 */
RetCode = _ECAT_Master_Check_Initial_Done(gESCCardNo, &InitialDone);
// Initial Status 표시: InitialDone=1은 "Pre Initial"; InitialDone=0은 "Initial Done",
InitialDone=99는 "Initial Error"라고 나타납니다
```

(3) 연결 모듈 정보를 생성합니다

그림 3.1.2.5

"Find Slave" 버튼을 클릭하여 다음 프로그램을 실행합니다:

```
//연결 모듈 수량을 확인합니다
RetCode = _ECAT_Master_Get_SlaveNum(gESCCardNo, &SlaveNum);
```

상기 프로그램 실행이 완료되면 검색된 Slave 장치 수량이 "Slave Num" 필드에 나타납니다.

(4) 프로그램 종료

그림 3.1.2.6

"Exit" 버튼을 클릭하여 다음 프로그램을 실행합니다:

```
for(i=0; i< gESCExistCards; i++)
{
      _ECAT_Master_Reset(gpESCCardNoList[i]);   // 인터페이스 카드를
리셋합니다
}
_ECAT_Master_Close();;    // 축 카드 실행을 정지합니다
```

3.2 원점 복귀 조작과 동작 제어

3.2.1 함수 표

함수의 명칭
_ECAT_Slave_Home_Config
_ECAT_Slave_Home_Move
_ECAT_Slave_Motion_Sd_Stop

■ **속성**

EtherCAT RTX version	EtherCAT Card version
○	○

3.2.2 응용 예제

프로그램 외관

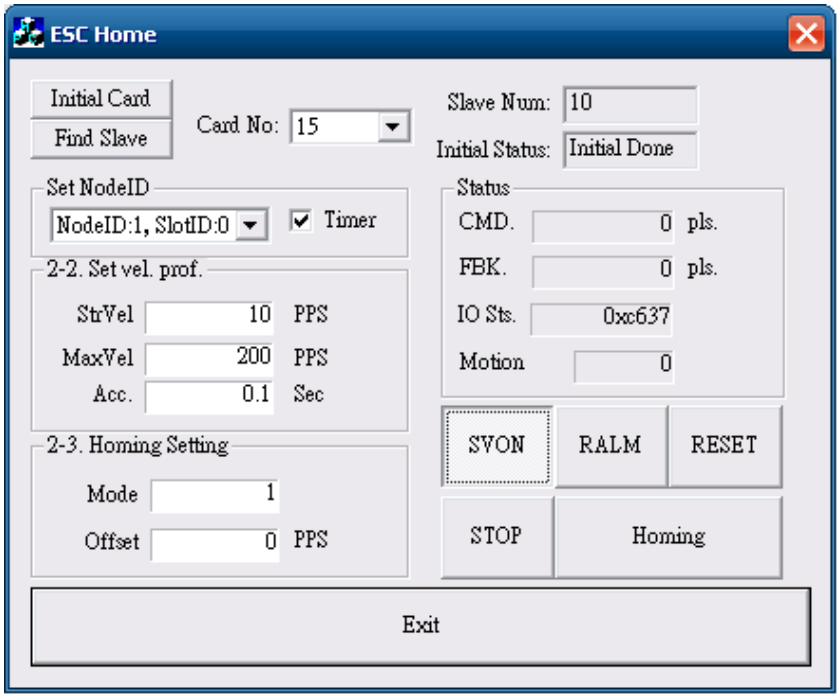


그림 3.2.2.1

(1) 인터페이스 카드의 실행과 초기화

그림 3.2.2.1의 "Initial Card" 버튼을 클릭해 인터페이스 카드 초기화를 실행합니다.
그림 3.2.2.1의 "Find Slave" 버튼을 클릭해 연결된 모듈을 검색합니다.
인터페이스 카드에 대한 자세한 설명은 3.1.2절의 "인터페이스 카드 실행"과 "인터페이스 카드 초기화"의 내용을 참고하십시오.

(2) 동작 제어의 인수 내용값 입력

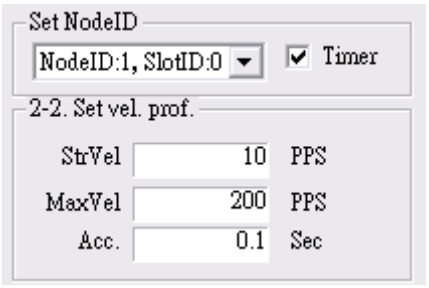


그림 3.2.2.2

Set NodeID 항목: API 함수의 인수와 변수 "AxisNo".

Timer 체크 박스: 체크 시 동작 상태가 표시됩니다. 체크하지 않을 경우 표시되지 않습니다.

StrVel. 항목: 1초간 송출하는 펄스 값. API 함수의 인수와 변수 "FirstSpeed".

MaxVel. 항목: 1초간 송출하는 펄스 값. API 함수의 인수와 변수 "SecondSpeed".

Acc. 항목: 최대 속도에 도달하는데 소요된 시간. API 함수의 인수와 변수 "Tacc".

3

(3) 원점 복귀 옵션 파라미터 설정 (원점 복귀 모드 및 오프셋량)

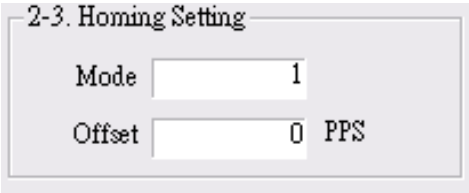


그림 3.2.2.3

Mode 항목: 원점 복귀 모드 1 ~ 35. API 함수의 인수와 변수 "Mode".

Offset 항목: 원점 복귀 오프셋량. API 함수의 인수와 변수 "Offset".

(4) 서보 모터 동력ON / OFF (servo on / servo off) 설정

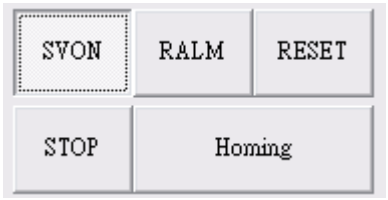


그림 3.2.2.4

그림 3.2.2.4의 "SVON" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode = _ECAT_Slave_Motion_Set_Svon(gESCCardNo, gNodeID,
gSlotID,ON_OFF);
// ON_OFF: 0 – 서보 동력 OFF, 1 – 서보 동력 ON
```

(5) 원점 복귀 조작

그림 3.2.2.4의 "Homing" 버튼을 클릭하여 다음 프로그램을 실행합니다:

```
/* 원점 복귀 모드 설정: 1~35, 오프셋과 속도 파라미터 */
RetCode = _ECAT_Slave_Home_Config(gESCCardNo, gNodeID, gSlotID, Mode,
Offset, StrVel, MaxVel, Tacc);
/* 원점 복귀 동작을 실행합니다 */
RetCode = _ECAT_Slave_Home_Move(gESCCardNo, gNodeID, gSlotID);
```

(6) 원점 복귀 동작 정지

원점 복귀 동작 조작을 정지하려면 그림 3.2.2.4의 "STOP" 버튼을 클릭하여 다음 프로그램을 실행해야 합니다:

```
/* 원점 복귀 동작의 인터럽트 */
RetCode =_ECAT_Slave_Motion_Sd_Stop(gESCCardNo, gNodeID, gSlotID,
Tdec);
```

(7) 프로그램 종료

"Exit" 버튼을 클릭하여 프로그램을 종료합니다.

함수를 건너뛰기 위해서는 반드시 "_ECAT_Master_Reset"와 "_ECAT_Master_Close"를 실행해야 합니다. 이 두가지 API 함수의 사용 방법에 대해서는 3.1.2절 "프로그램 종료"의 함수 작동 부분을 참고하십시오.

3.3 토크 동작 제어

3.3.1 함수 표

함수의 명칭
_ECAT_Slave_PT_Start_Move
_ECAT_Slave_Motion_Emg_Stop

■ **속성**

EtherCAT RTX version	EtherCAT Card version
○	○

3.3.2 응용 예제

프로그램 외관

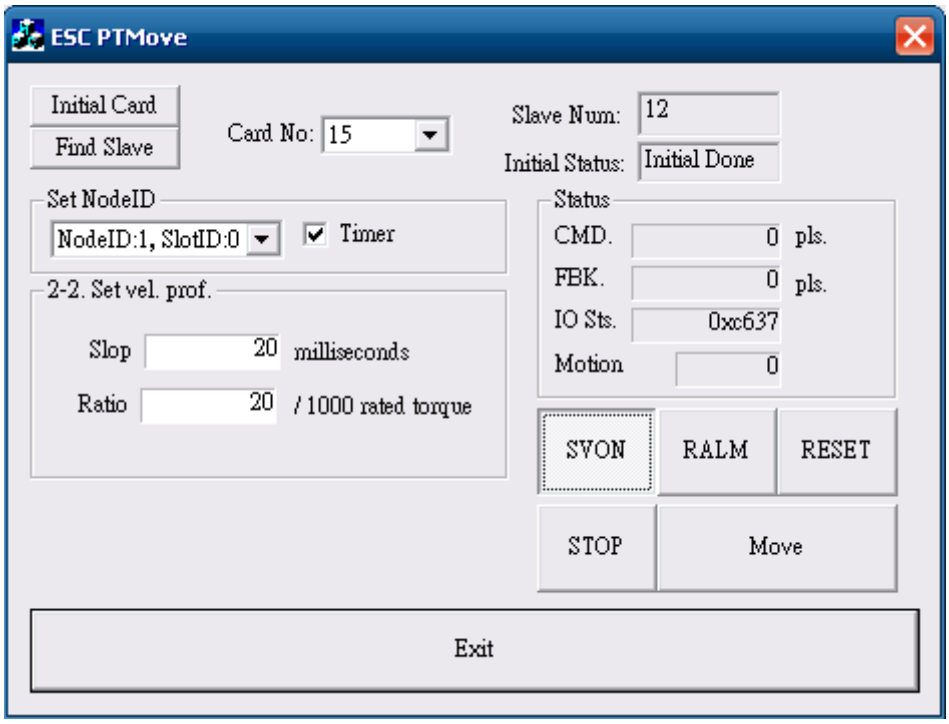


그림 3.3.2.1

3

(1) 인터페이스 카드의 실행과 초기화

그림 3.3.2.1의 "Initial Card" 버튼을 클릭해 인터페이스 카드 초기화를 실행합니다.
그림 3.3.2.1의 "Find Slave" 버튼을 클릭해 연결된 모듈을 검색합니다.
인터페이스 카드에 대한 자세한 설명은 3.1.2절의 "인터페이스 카드 실행"과 "인터페이스 카드 초기화"의 내용을 참고하십시오.

(2) 드라이브의 Node ID를 설정하여 동작 상태 표시하기

그림 3.3.2.2

Node ID 입력 후 "Timer"에 체크하면 체크 박스에 동작 상태가 표시됩니다

Set NodeID 항목: API 함수의 인수와 변수 "AxisNo".

Timer 체크 박스: 체크 시 동작 상태가 표시됩니다. 체크하지 않을 경우 표시되지 않습니다.

(3) Slop와 Ratio 값 입력

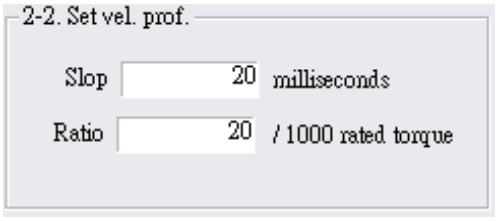


그림 3.3.2.3

Slop 항목: 0부터 100%까지의 정격 토크에 필요한 시간. (단위: 밀리초)

Ratio 항목: 정격 토크 값 천분율. 예를 들어 해당 값이 20일 경우 2% 정격 토크를 의미합니다.

(4) 서보 모터 동력ON / OFF (servo on / servo off) 설정

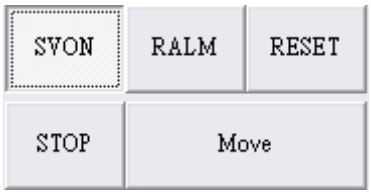


그림 3.3.2.4

그림 3.3.2.4의 "SVON" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode = _ECAT_Slave_Motion_Set_Svon(gESCCardNo, gNodeID, gSlotID, ON_OFF);
// ON_OFF: 0 – 서보 동력 OFF, 1 – 서보 동력 ON
```

(5) 토크 동작 제어

그림 3.3.2.4의 "Move" 버튼을 클릭하여 다음 프로그램을 실행합니다:

```
/* 토크 파라미터를 설정하고 토크 동작을 실행합니다*/
RetCode = _ECAT_Slave_PT_Start_Move(gESCCardNo, gNodeID, gSlotID, Torque, Slope);
// Torque 값이 0을 초과하면 모터가 순방향으로 동작합니다; 값이 0 미만이면 모터가 역방향으로 동작합니다.
```

그림 3.3.2.4의 "STOP" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
/* 모터 토크 동작 정지*/
RetCode = _ECAT_Slave_Motion_Emg_Stop(gESCCardNo, gNodeID, gSlotID);
```

3

(6) 모니터링 표시

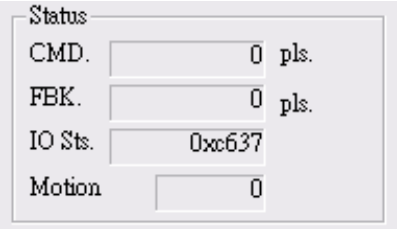


그림 3.3.2.5

동작 계수값:

```
RetCode = _ECAT_Slave_Motion_Get_Command(gESCCardNo, gNodeID, gSlotID, &Cmd);
// 명령값을 확인합니다
RetCode = _ECAT_Slave_Motion_Get_Position(gESCCardNo, gNodeID, gSlotID, &Pos);
// Feedback meter의 수치를 확인합니다
```

동작 상태: Motion Status:

```
RetCode = _ECAT_Slave_Motion_Get_StatusWord(gESCCardNo, gNodeID, gSlotID, &Ststus); // 현재 상태를 확인합니다
RetCode = _ECAT_Slave_Motion_Get_Mdone(gESCCardNo, gNodeID, gSlotID, &MCDone); // 모터의 현재 동작 상태를 확인합니다
```

(7) 동작 계수값 의 리셋 및 ALM 삭제

그림 3.3.2.4의 "RESET" 버튼을 클릭해 리셋 명령을 실행합니다:

```
RetCode=_ECAT_Slave_Motion_Set_Position(gESCCardNo, gNodeID, gSlotID, 0);
//feedback 삭제를 먼저 실행합니다
RetCode=_ECAT_Slave_Motion_Set_Command(gESCCardNo,gNodeID,gSlotID, 0);
//command 삭제를 나중에 실행합니다
```

그림 3.3.2.4의 "RALM" 버튼을 클릭해 ALM 삭제 명령을 실행합니다:

```
RetCode = _ECAT_Slave_Motion_Ralm(gESCCardNo, gNodeID, gSlotID);
//ALM Code 삭제
```

(8) 프로그램 종료

"Exit" 버튼을 클릭하여 프로그램을 종료합니다.

함수를 건너뛰기 위해서는 반드시 "_ECAT_Master_Reset"와 "_ECAT_Master_Close"를 실행해야 합니다. 이 두가지 API 함수의 사용 방법에 대해서는 3.1.2절 "프로그램 종료"의 함수 작동 부분을 참고하십시오.

3.4 고정 속도 동작 제어

3.4.1 함수 표

함수의 명칭
_ECAT_Slave_PV_Start_Move
_ECAT_Slave_Motion_Sd_Stop

■ 속성

EtherCAT RTX version	EtherCAT Card version
○	○

3.4.2 응용 예제

프로그램 외관

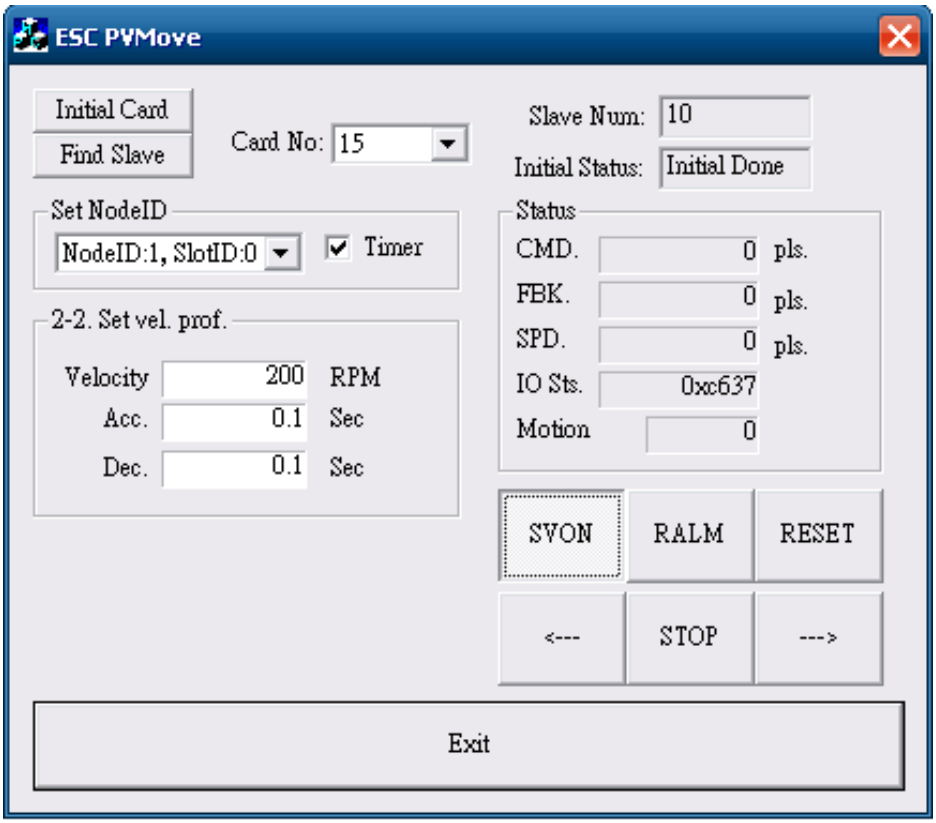


그림 3.4.2.1

3

(1) 인터페이스 카드의 실행과 초기화

그림 3.4.2.1의 "Initial Card" 버튼을 클릭해 인터페이스 카드 초기화를 실행합니다.
그림 3.4.2.1의 "Find Slave" 버튼을 클릭해 연결된 모듈을 검색합니다.
인터페이스 카드에 대한 자세한 설명은 3.1.2절의 "인터페이스 카드 실행"과 "인터페이스 카드 초기화"의 내용을 참고하십시오.

(2) 드라이브의 Node ID를 설정하여 동작 상태 표시하기

그림 3.4.2.2

Node ID, SlotID를 선택하여 "Timer"에 체크하면 체크 박스에 동작 상태가 표시됩니다

Set NodeID 항목: API 함수의 인수와 변수 "AxisNo".

Timer 체크 박스: 체크 시 동작 상태가 표시됩니다. 체크하지 않을 경우 표시되지 않습니다.

(3) 가감속 시간 값과 분당 회전 속도 (RPM) 값 입력

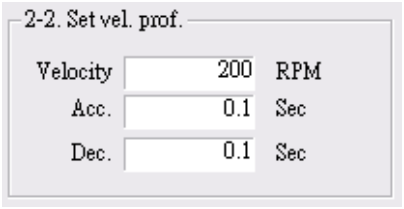


그림 3.4.2.3

Velocity 항목: API 함수의 인수와 변수 "Rpm".

※ 실제 RPM 값은 변수 **Velocity**의 0.1배입니다.

Acc 항목: API 함수의 인수와 변수 "Tacc".

Dec 항목: API 함수의 인수와 변수 "Tdec".

(4) 서보 모터 동력ON / OFF (servo on / servo off) 설정

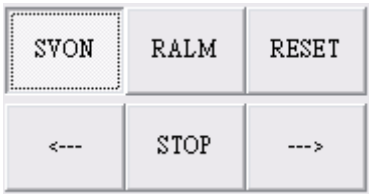


그림 3.4.2.4

그림 3.4.2.4의 "SVON" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode = _ECAT_Slave_Motion_Set_Svon(gESCCardNo, gNodeID, gSlotID,
ON_OFF);
// ON_OFF: 0 – 서보 동력 OFF, 1 – 서보 동력 ON
```

(5) 속도 동작 제어

그림 3.4.2.4의 "←" 또는 "→" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
/* 속도 모드 파라미터(가속 시간 및 감속 시간 값)를 설정하고, 속도 모드 동작을
실행합니다*/
RetCode = _ECAT_Slave_PV_Start_Move(gESCCardNo, gNodeID, gSlotID,
Velocity, Tacc, Tdec);
// rpm의 값이 0 이상일 경우 드라이브의 모터가 정면으로 회전합니다. 반대로
rpm의 값이 0 이하일 경우 반대 방향으로 회전합니다.
```

그림 3.4.2.4의 "STOP" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode = _ECAT_Slave_Motion_Emg_Stop(gESCCardNo, gNodeID, gSlotID);
```

(6) 모니터링 표시

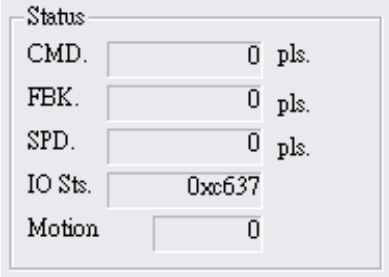


그림 3.4.2.5

동작 계수값:

```
RetCode = _ECAT_Slave_Motion_Get_Command(gESCCardNo, gNodeID,
gSlotID, &Cmd); // 명령값을 확인합니다
RetCode = _ECAT_Slave_Motion_Get_Position(gESCCardNo, gNodeID, gSlotID,
&Pos); // Feedback meter의 수치를 확인합니다
```

동작 상태: Motion Status:

```
RetCode = _ECAT_Slave_Motion_Get_Current_Speed(gESCCardNo,gNodeID,
gSlotID, &Spd); // 동작 실행의 속도 값을 확인합니다
RetCode = _ECAT_Slave_Motion_Get_StatusWord(gESCCardNo, gNodeID,
gSlotID, &Ststus); // 현재 상태를 확인합니다
RetCode = _ECAT_Slave_Motion_Get_Mdone(gESCCardNo, gNodeID, gSlotID,
&MCDone); // 모터의 현재 동작 상태를 확인합니다
```

3

(7) 동작 계수값 의 리셋 및 ALM 삭제

그림 3.4.2.4의 "RESET" 버튼을 클릭해 리셋 명령을 실행합니다:

```
RetCode=_ECAT_Slave_Motion_Set_Position(gESCCardNo, gNodeID, gSlotID, 0);
//feedback 삭제를 먼저 실행합니다
RetCode = _ECAT_Slave_Motion_Set_Command(gESCCardNo,gNodeID,gSlotID,0);
//command 삭제를 나중에 실행합니다
```

그림 3.4.2.4의 "RALM" 버튼을 클릭해 ALM 삭제 명령을 실행합니다:

```
RetCode = _ECAT_Slave_Motion_Ralm(gESCCardNo, gNodeID, gSlotID);
//ALM Code 삭제
```

(8) 프로그램 종료

"Exit" 버튼을 클릭하여 프로그램을 종료합니다.

함수를 건너뛰기 위해서는 반드시 "_ECAT_Master_Reset"와 "_ECAT_Master_Close"를 실행해야 합니다. 이 두가지 API 함수의 사용 방법에 대해서는 3.1.2절 "프로그램 종료"의 함수 작동 부분을 참고하십시오.

3.5 PP 모드 동작 제어

3.5.1 함수 표

함수의 명칭
_ECAT_Slave_PP_Start_Move
_ECAT_Slave_Motion_Sd_Stop

■ **속성**

EtherCAT RTX version	EtherCAT Card version
○	○

3.5.2 응용 예제

프로그램 외관

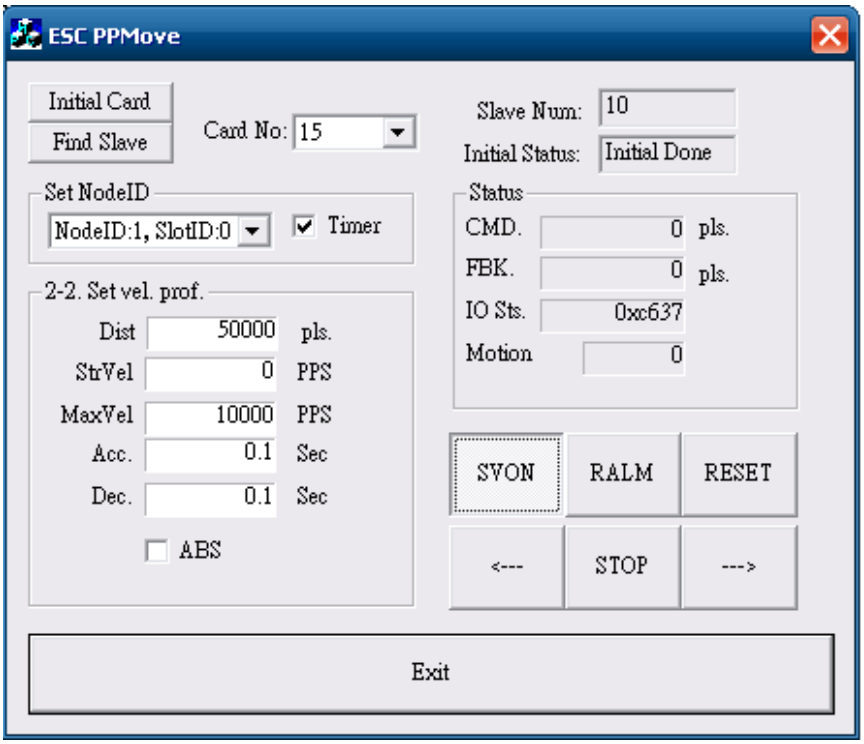


그림 3.5.2.1

(1) 인터페이스 카드의 실행과 초기화

그림 3.5.2.1의 "Initial Card" 버튼을 클릭해 인터페이스 카드 초기화를 실행합니다.
그림 3.5.2.1의 "Find Slave" 버튼을 클릭해 연결된 모듈을 검색합니다.
인터페이스 카드에 대한 자세한 설명은 3.1.2절의 "인터페이스 카드 실행"과 "인터페이스 카드 초기화"의 내용을 참고하십시오.

(2) NodeID, SlotID를 선택하여 동작 상태 표시하기

그림 3.5.2.2

NodeID, SlotID를 선택하여 "Timer"에 체크하면 체크 박스에 동작 상태가 표시됩니다

Set NodeID 항목: API 함수의 인수와 변수 "AxisNo".

Timer 체크 박스: 체크 시 동작 상태가 표시됩니다. 체크하지 않을 경우 표시되지 않습니다.

3

(3) 동작 제어의 인수 내용값 입력

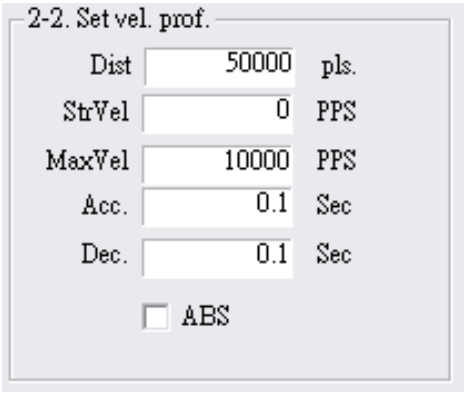


그림 3.5.2.3

Dist. 항목: 동작의 거리.API 함수의 인수와 변수 "Dist".

StrVel 항목: 초기 속도. API 함수의 인수와 변수 "StrVel".

MaxVel 항목: 최대 속도. API 함수의 인수와 변수 "MaxVel".

Acc 항목: 최대 속도에 도달하는데 소요된 시간. API 함수의 인수와 변수 "Tacc".

Dec 항목: 최대 속도에서 0으로 떨어지는데 소요된 시간. API 함수의 인수와 변수 "Tdec".

ABS 체크 박스: 절대 좌표의 방식을 참고하여 동작의 변위를 실행하고자 할 경우 반드시 체크해야 합니다.

(4) 서보 모터 동력ON / OFF (servo on / servo off) 설정

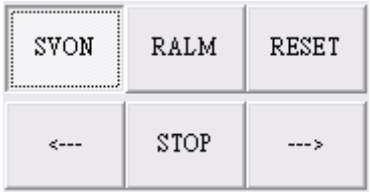


그림 3.5.2.4

그림 3.5.2.4의 "SVON" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode = _ECAT_Slave_Motion_Set_Svon(gESCCardNo, gNodeID, gSlotID, ON_OFF);
// ON_OFF: 0 – 서보 동력 OFF, 1 – 서보 동력 ON
```

(5) 속도 동작 제어 시작

그림 3.5.2.4의 "←" 또는 "→" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode = _ECAT_Slave_PP_Start_Move(gESCCardNo, gNodeID, gSlotID, Dist, StrVel, MaxVel, Tacc, Tdec, gbIsABS);
// gbIsABS-- 0: 상대적 변위 1: 절대적 변위
```

(6) 동작 정지

그림 3.5.2.4의 "STOP" 버튼을 클릭하여 긴급 정지를 실행합니다:

```
RetCode = _ECAT_Slave_Motion_Emg_Stop(gESCCardNo, gNodeID, gSlotID);
```

해당 예시는 긴급 정지 방식을 사용하여 동작의 변위를 정지시키는 것으로 감속 시간을 0으로 설정하여 동작을 신속히 정지시킵니다.

(7) 모니터링 표시

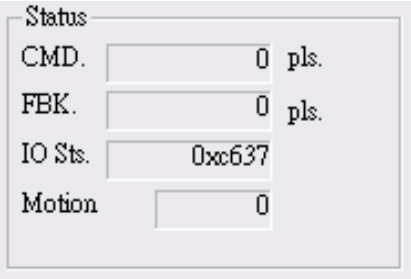


그림 3.5.2.5

동작 계수값:

```
RetCode = _ECAT_Slave_Motion_Get_Command(gESCCardNo, gNodeID,
gSlotID, &Cmd); // 명령값을 확인합니다
RetCode = _ECAT_Slave_Motion_Get_Position(gESCCardNo, gNodeID, gSlotID,
&Pos); // Feedback meter의 수치를 확인합니다
```

동작 상태: Motion Status:

```
RetCode = _ECAT_Slave_Motion_Get_Current_Speed(gESCCardNo,gNodeID,
gSlotID, &Spd); // 동작 실행의 속도 값을 확인합니다
RetCode = _ECAT_Slave_Motion_Get_StatusWord(gESCCardNo, gNodeID,
gSlotID, &Ststus); // 현재 상태를 확인합니다
RetCode = _ECAT_Slave_Motion_Get_Mdone(gESCCardNo, gNodeID, gSlotID,
&MCDone); // 모터의 현재 동작 상태를 확인합니다
```

(8) 동작 계수값 의 리셋 및 ALM 삭제

그림 3.5.2.4의 "RESET" 버튼을 클릭해 리셋 명령을 실행합니다:

```
RetCode = _ECAT_Slave_Motion_Set_Position(gESCCardNo, gNodeID, gSlotID,
0);
//feedback 삭제를 먼저 실행합니다
RetCode = _ECAT_Slave_Motion_Set_Command(gESCCardNo, gNodeID,
gSlotID, 0);
//command 삭제를 나중에 실행합니다
```

그림 3.5.2.4의 "RALM" 버튼을 클릭해 ALM 삭제 명령을 실행합니다:

```
RetCode = _ECAT_Slave_Motion_Ralm(gESCCardNo, gNodeID, gSlotID);
//ALM Code 삭제
```

(9) 프로그램 종료

"Exit" 버튼을 클릭하여 프로그램을 종료합니다.

함수를 건너뛰기 위해서는 반드시 "_ECAT_Master_Reset"와 "_ECAT_Master_Close"를 실행해야 합니다. 이 두가지 API 함수의 사용 방법에 대해서는 3.1.2절 "프로그램 종료"의 함수 작동 부분을 참고하십시오.

3.6 CSP 모드 동작의 제어

3.6.1 함수 표

함수의 명칭
_ECAT_Slave_Motion_Set_Svon
_ECAT_Slave_CSP_Start_Move
_ECAT_Slave_CSP_Start_V_Move
_ECAT_Slave_CSP_Start_Multiaxes_Move
_ECAT_Slave_CSP_Start_Arc_Move
_ECAT_Slave_CSP_Start_Arc2_Move
_ECAT_Slave_CSP_Start_Arc3_Move
_ECAT_Slave_CSP_Start_Spiral_Move
_ECAT_Slave_CSP_Start_Spiral2_Move
_ECAT_Slave_CSP_Start_Heli_Move
_ECAT_Slave_CSP_Start_Sphere_Move
_ECAT_Slave_Motion_Sd_Stop
_ECAT_Slave_Motion_Set_Position
_ECAT_Slave_Motion_Set_Command
_ECAT_Slave_Motion_Ralm
_ECAT_Slave_Motion_Get_Command
_ECAT_Slave_Motion_Get_Position
_ECAT_Slave_Motion_Get_Current_Speed
_ECAT_Slave_Motion_Get_StatusWord
_ECAT_Slave_Motion_Get_Mdone
_ECAT_Master_Check_Initial_Done

■ **속성**

EtherCAT RTX version	EtherCAT Card version
○	○

3.6.2 응용 예제

프로그램 외관

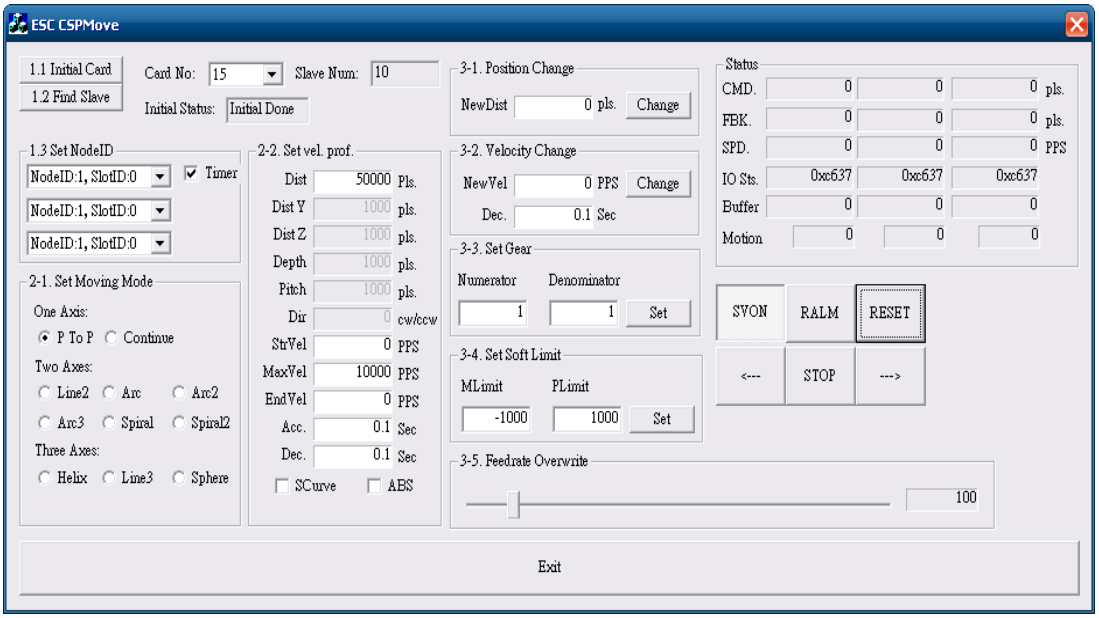


그림 3.6.2.1

(1) 인터페이스 카드의 실행과 초기화

그림 3.6.2.1의 "Initial Card" 버튼을 클릭해 인터페이스 카드 초기화를 실행합니다.
그림 3.6.2.1의 "Find Slave" 버튼을 클릭해 연결된 모듈을 검색합니다.
인터페이스 카드에 대한 자세한 설명은 3.1.2절의 "인터페이스 카드 실행"과 "인터페이스 카드 초기화"의 내용을 참고하십시오.

(2) NodeID, SlotID를 선택하여 동작 상태 표시하기

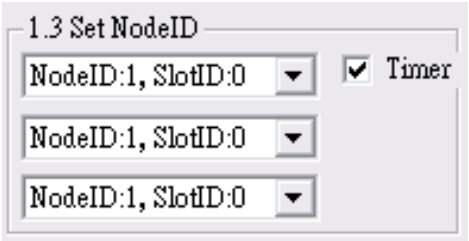


그림 3.6.2.2

NodeID, SlotID를 선택하여 "Timer"에 체크하면 체크 박스에 동작 상태가 표시됩니다

Set NodeID 항목: API 함수의 인수와 변수 "AxisNo".

Timer 체크 박스: 체크 시 동작 상태가 표시됩니다. 체크하지 않을 경우 표시되지 않습니다.

3

(3) 기능 선택

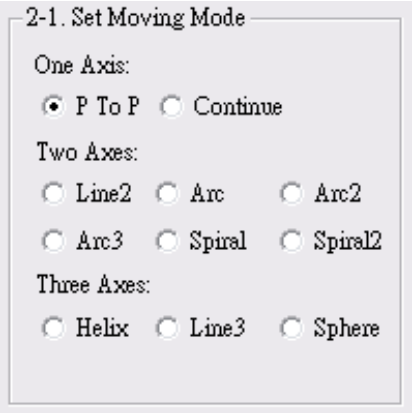


그림 3.6.2.2

단축 동작 제어

P To P 항목: 점대점 동작.

Continue 항목: 직선 동작 모드.

2축 동작 제어

Line2 항목: 선형 보간 동작 제어.

Arc 항목: 원형 보간 동작 제어 형식1.

Arc2 항목: 원형 보간 동작 제어 형식2.

Arc3 항목: 원형 보간 동작 제어 형식3.

Spiral 항목: 나선 보간 동작 제어 형식1.

Spiral2 항목: 나선 보간 동작 제어 형식2.

세 축의 동작 제어

Helix 항목: 나선 보간 동작 제어.

Line3 항목: 선형 보간 동작 제어.

Sphere 항목: 3축 구형 동작 제어.

(4) 단축 동작 제어의 인수 내용값

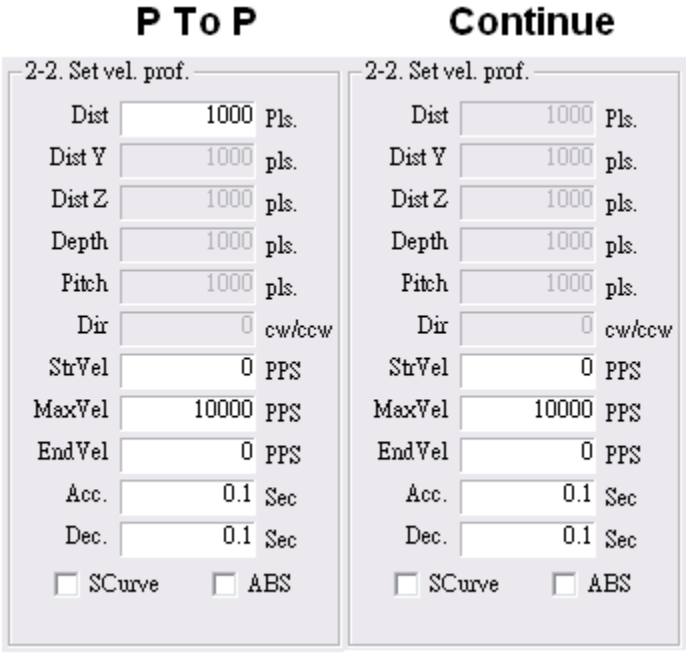


그림 3.6.2.3

Dist. 항목: 지정한 동작 stroke. API 함수의 인수와 변수 "Dist".

StrVel 항목: 동작 초기 속도. API 함수의 인수와 변수 "StrVel".

ConstVel 항목: 동작 정상 속도. API 함수의 인수와 변수 "ConstVel".

EndVel 항목: 동작 완료 속도. API 함수의 인수와 변수 "EndVel".

TPhase1 항목: StrVel에서 ConstVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase1".

TPhase2 항목: ConstVel에서 EndVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase2".

S-Curve 체크 박스: S-curve 속도 곡선을 사용하고자 할 경우 반드시 체크해야 합니다.

ABS 체크 박스: 절대 좌표의 방식을 참고하여 동작의 변위를 실행하고자 할 경우 반드시 체크해야 합니다.

(5) 서보 모터 동력ON / OFF (servo on / servo off) 설정

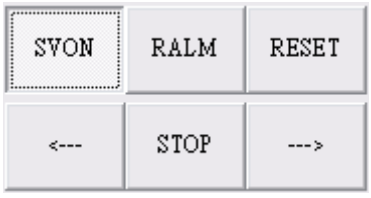


그림 3.6.2.4

3

그림 3.6.2.4의 "SVON" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode =_ECAT_Slave_Motion_Set_Svon(gESCCardNo, gNodeID[i], gSlotID[i], ON_OFF); // ON_OFF: 0 – 서보 동력 OFF, 1 – 서보 동력 ON
```

(6) P To P 항목 선택하여 점대점 동작 제어 실행

그림 3.6.2.4의 "←" 또는 "→" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode = _ECAT_Slave_CSP_Start_Move(gESCCardNo, gNodeID[0], gSlotID[0], Dist[0], StrVel, ConstVel, EndVel, Tacc, Tdec, gbIsSCurve, gbIsABS);
//gblsSCurve 1: T-Curve 2: S-Curve    gblsABS    0: 상대적 변위 1: 절대적 변위
```

(7) P To P 항목 선택하여 위치 변환 또는 속도 변환 동작 제어 실행

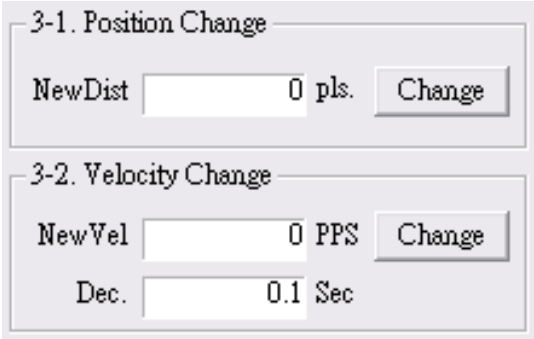


그림 3.6.2.5

그림 3.6.2.4의 "←" 또는 "→" 버튼을 클릭하여 P To P 동작 제어를 실행할 때 동작 중의 위치를 새로운 위치로 변환하려면 그림 3.6.2.5의 "P change"를 클릭하여 다음 프로그램을 실행해야 합니다:

```
RetCode = _ECAT_Slave_CSP_TargetPos_Change(gESCCardNo, gNodeID[0], gSlotID[0], NewPos); // 새로운 위치값을 현재 위치로 변환합니다
```

동작 중인 속도를 새로운 속도로 변환하려면 그림 3.6.2.5의 "V change"를 클릭하여 다음 프로그램:

```
RetCode = _ECAT_Slave_CSP_Velocity_Change(gESCCardNo, gNodeID[0], gSlotID[0], NewSpd, NewTdec); // 새로운 속도값을 현재 속도로 변환합니다
```

(8) P To P 항목 선택하여 Gear 설정 또는 소프트웨어 limit 동작 제어 실행

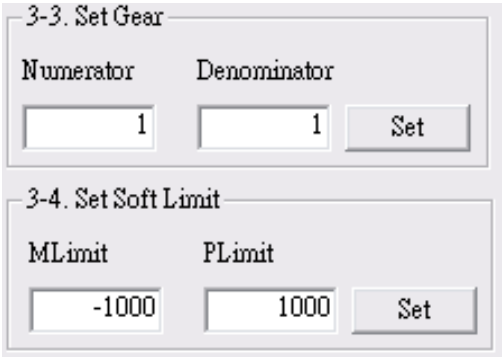


그림 3.6.2.6

그림 3.6.2.4의 "←" 또는 "→" 버튼을 클릭하여 P To P 동작 제어를 실행할 때 Gear 값을 설정하려면 그림 3.6.2.6의 "Set Gear"를 클릭하여 다음 프로그램을 실행합니다:

```
RetCode = _ECAT_Slave_CSP_Set_Gear(gESCCardNo, gNodeID[0], gSlotID[0], Numerator, Denominator, Enable); //새로운 Gear 값을 설정합니다
```

소프트웨어 limit 값을 설정하려면 그림 3.6.2.6의 "Set Soft_Limit"를 클릭하여 다음 프로그램:

```
RetCode = _ECAT_Slave_CSP_Set_Softlimit(gESCCardNo, gNodeID[0], gSlotID[0], MLimit, PLimit, Enable); //소프트웨어 limit 값을 설정합니다
```

(9) P To P 항목 선택하여 FeedrateOverwrite 동작 제어 실행

그림 3.6.2.7

그림 3.6.2.4의 "←" 또는 "→" 버튼을 클릭하여 P To P 동작 제어를 실행할 때 FeedrateOverwrite 제어를 실행하려면 그림 3.6.2.7의 조절바를 움직여 다음 프로그램을 실행해야 합니다:

```
RetCode = _ECAT_Slave_CSP_Feedrate_Overwrite(gESCCardNo, gNodeID[0], gSlotID[0], 2, NewSpd, 0.1); // Mode=2
```

(10) Continue 항목 선택하여 정속 동작 제어 실행

그림 3.6.2.4의 "←" 또는 "→" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode = _ECAT_Slave_CSP_Start_V_Move(gESCCardNo, gNodeID[0], gSlotID[0], 0, StrVel, MaxVel, Tacc, gblsSCurve);
// gblsSCurve 1 : T-Curve 2 : S-Curve
```

3

(11) 2축 동작 모드(Line2, Spiral, Spiral2) 및 동작 stroke 설정

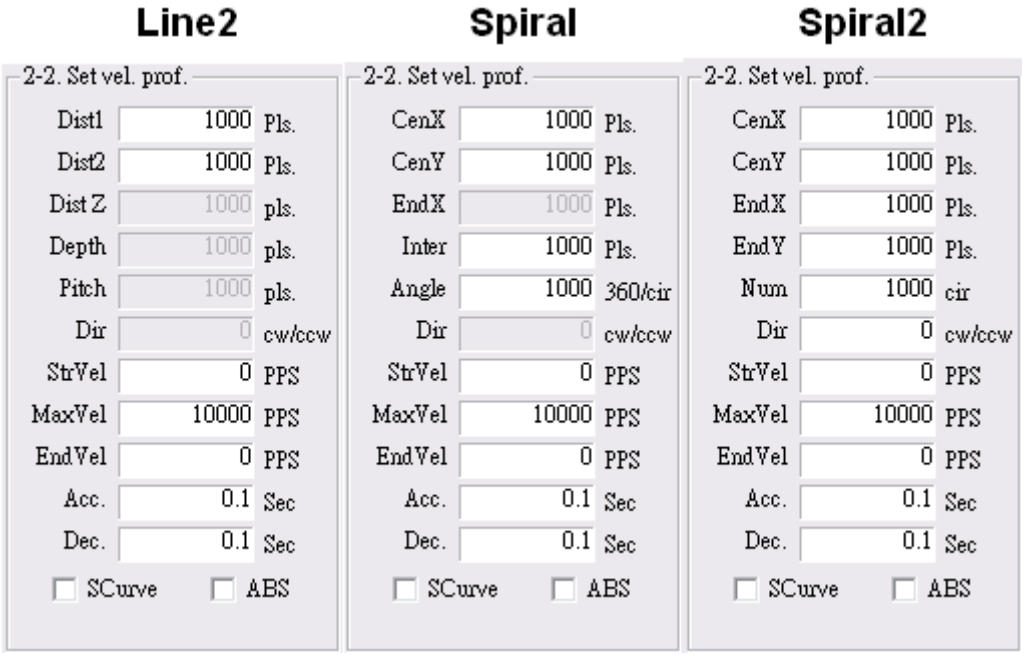


그림 3.6.2.8

Line2 파라미터 설정:

Dist1 항목: X축의 동작 거리 파라미터. API 함수의 인수와 변수 "DisX".

Dist2 항목: Y축의 동작 거리 파라미터. API 함수의 인수와 변수 "DisY".

StrVel 항목: 초기 속도. API 함수의 인수와 변수 "StrVel".

ConstVel 항목: 정상 속도. API 함수의 인수와 변수 "ConstVel".

EndVel 항목: 동작 완료 속도. API 함수의 인수와 변수 "EndVel".

TPhase1 항목: StrVel에서 ConstVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase1".

TPhase2 항목: ConstVel에서 EndVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase2".

S-Curve 체크 박스: S-curve 속도 곡선을 사용하고자 할 경우 반드시 체크해야 합니다.

ABS 체크 박스: 절대 좌표의 방식을 참고하여 동작의 변위를 실행하고자 할 경우 반드시 체크해야 합니다.

Spiral 파라미터 설정:

CenX 항목: 지정축에 상응하는 중심 위치 X. API 함수의 인수와 변수 "CenterPoint".

CenY 항목: 지정축에 상응하는 중심 위치 Y. API 함수의 인수와 변수 "CenterPoint".

Inter 항목: 나선 간의 상대적 위치 거리. API 함수의 인수와 변수 "Spiral_Interval".

Angle 항목: 나선 동작의 총 각도. API 함수의 인수와 변수 "Angle".

StrVel 항목: 초기 속도. API 함수의 인수와 변수 "StrVel".

ConstVel 항목: 정상 속도. API 함수의 인수와 변수 "ConstVel".

EndVel 항목: 동작 완료 속도. API 함수의 인수와 변수 "EndVel".

TPhase1 항목: StrVel에서 ConstVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase1".

TPhase2 항목: ConstVel에서 EndVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase2".

S-Curve 체크 박스: S-curve 속도 곡선을 사용하고자 할 경우 반드시 체크해야 합니다.

ABS 체크 박스: 절대 좌표의 방식을 참고하여 동작의 변위를 실행하고자 할 경우 반드시 체크해야 합니다.

Spiral2 파라미터 설정:

CenX 항목: 지정축에 상응하는 중심 위치 X. API 함수의 인수와 변수 "CenterPoint".

CenY 항목: 지정축에 상응하는 중심 위치 Y. API 함수의 인수와 변수 "CenterPoint".

EndX 항목: 지정축에 상응하는 최종 위치 X. API 함수의 인수와 변수 "EndPoint".

EndY 항목: 지정축에 상응하는 최종 위치 Y. API 함수의 인수와 변수 "EndPoint".

Num 항목: 나선 동작의 회전수. API 함수의 인수와 변수 "CycleNum".

Dir 항목: 지정된 방향 (0: 시계 방향 1: 시계 반대 방향). API 함수의 인수와 변수 "Dir".

StrVel 항목: 초기 속도. API 함수의 인수와 변수 "StrVel".

ConstVel 항목: 정상 속도. API 함수의 인수와 변수 "ConstVel".

EndVel 항목: 동작 완료 속도. API 함수의 인수와 변수 "EndVel".

TPhase1 항목: StrVel에서 ConstVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase1".

TPhase2 항목: ConstVel에서 EndVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase2".

S-Curve 체크 박스: S-curve 속도 곡선을 사용하고자 할 경우 반드시 체크해야 합니다.

ABS 체크 박스: 절대 좌표의 방식을 참고하여 동작의 변위를 실행하고자 할 경우 반드시 체크해야 합니다.

(12) 서보 모터 동력ON / OFF (servo on / servo off) 설정

그림 3.6.2.4의 "SVON" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode =_ECAT_Slave_Motion_Set_Svon(gESCCardNo, gNodeID[i], gSlotID[i], ON_OFF); // ON_OFF: 0 – 서보 동력 OFF, 1 – 서보 동력 ON
```

3

(13) Line2 항목 선택하여 2축 직선 동작 제어 실행

그림 3.6.2.4의 "←" 또는 "→" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode = _ECAT_Slave_CSP_Start_Multiaxes_Move(gESCCardNo, 2,
gNodeID, gSlotID, Dist, StrVel, MaxVel, EndVel, Tacc, Tdec, gbIsSCurve,
gbIsABS);
//gbIsSCurve 1: T-Curve 2: S-Curve    gbIsABS    0: 상대적 변위 1: 절대적 변위
```

(14) Spiral 항목 선택하여 2축 원형 동작 제어 실행(원 중심+각도)

그림 3.6.2.4의 "←" 또는 "→" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode = _ECAT_Slave_CSP_Start_Spiral_Move(gESCCardNo,gNodeID,
gSlotID,CenPoint, Spiral_Interval, Angle, StrVel, MaxVel, EndVel, Tacc, Tdec,
gbIsSCurve, gbIsABS);
//gbIsSCurve 1: T-Curve 2: S-Curve    gbIsABS    0: 상대적 변위 1: 절대적 변위
```

(15) Spiral2 항목 선택하여 2축 원형 동작 제어 실행(최종 좌표+회전수)

그림 3.6.2.4의 "←" 또는 "→" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode = _ECAT_Slave_CSP_Start_Spiral2_Move(gESCCardNo, gNodeID,
gSlotID,CenPoint, EndPoint, CycleNum, Dir, StrVel, MaxVel, EndVel, Tacc, Tdec,
gbIsSCurve, gbIsABS);
//gbIsSCurve 1: T-Curve 2: S-Curve    gbIsABS    0: 상대적 변위 1: 절대적 변위
```

(16) 2축 동작 모드(Arc, Arc2, Arc3) 및 동작 stroke 설정

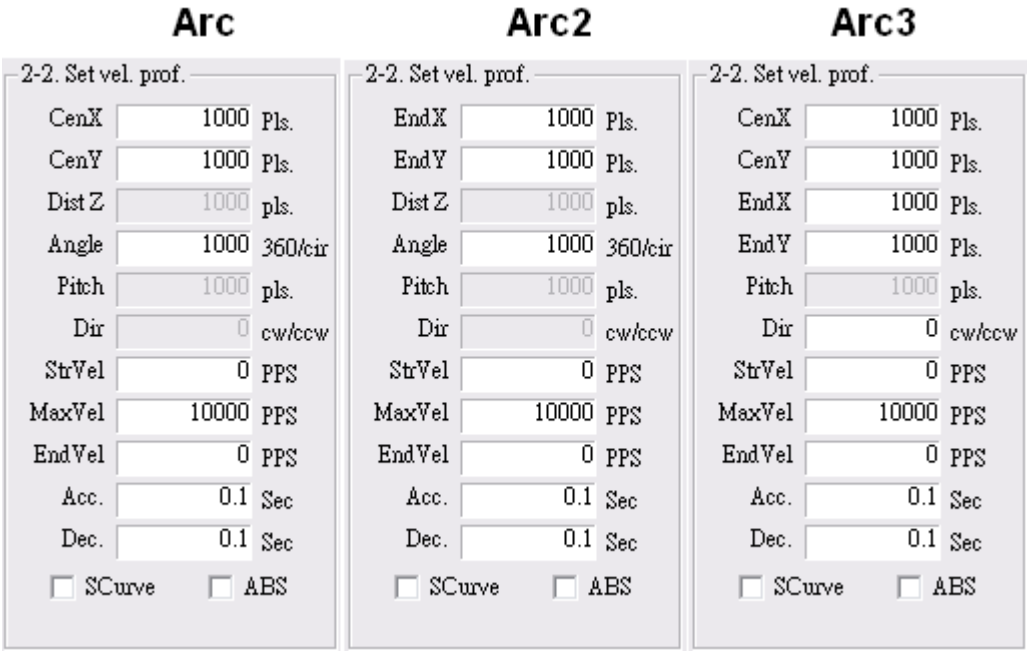


그림 3.6.2.9

Arc 파라미터 설정:

CenX 항목: 지정축에 상응하는 중심 위치 X. API 함수의 인수와 변수 "CenterPoint".

CenY 항목: 지정축에 상응하는 중심 위치 Y. API 함수의 인수와 변수 "CenterPoint".

Angle 항목: 원형의 각도 설정. API 함수의 인수와 변수 "Angle".

StrVel 항목: 초기 속도. API 함수의 인수와 변수 "StrVel".

ConstVel 항목: 정상 속도. API 함수의 인수와 변수 "ConstVel".

EndVel 항목: 동작 완료 속도. API 함수의 인수와 변수 "EndVel".

TPhase1 항목: StrVel에서 ConstVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase1".

TPhase2 항목: ConstVel에서 EndVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase2".

S-Curve 체크 박스: S-curve 속도 곡선을 사용하고자 할 경우 반드시 체크해야 합니다.

ABS 체크 박스: 절대 좌표의 방식을 참고하여 동작의 변위를 실행하고자 할 경우 반드시 체크해야 합니다.

Arc2 파라미터 설정:

EndX 항목: 지정축에 상응하는 최종 위치 X.API 함수의 인수와 변수 "EndPoint".

EndY 항목: 지정축에 상응하는 최종 위치 Y.API 함수의 인수와 변수 "EndPoint".

Angle 항목: 원형의 각도 설정. API 함수의 인수와 변수 "Angle".

StrVel 항목: 초기 속도. API 함수의 인수와 변수 "StrVel".

ConstVel 항목: 정상 속도. API 함수의 인수와 변수 "ConstVel".

EndVel 항목: 동작 완료 속도. API 함수의 인수와 변수 "EndVel".

TPhase1 항목: StrVel에서 ConstVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase1".

TPhase2 항목: ConstVel에서 EndVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase2".

S-Curve 체크 박스: S-curve 속도 곡선을 사용하고자 할 경우 반드시 체크해야 합니다.

ABS 체크 박스: 절대 좌표의 방식을 참고하여 동작의 변위를 실행하고자 할 경우 반드시 체크해야 합니다.

Arc3 파라미터 설정:

CenX 항목: 지정축에 상응하는 중심 위치 X. API 함수의 인수와 변수 "CenterPoint".

CenY 항목: 지정축에 상응하는 중심 위치 Y. API 함수의 인수와 변수 "CenterPoint".

EndX 항목: 지정축에 상응하는 최종 위치 X. API 함수의 인수와 변수 "EndPoint".

EndY 항목: 지정축에 상응하는 최종 위치 Y. API 함수의 인수와 변수 "EndPoint".

Dir 항목: 지정된 방향 (0: 시계 방향 1: 시계 반대 방향). API 함수의 인수와 변수 "Dir".

StrVel 항목: 초기 속도. API 함수의 인수와 변수 "StrVel".

ConstVel 항목: 정상 속도. API 함수의 인수와 변수 "ConstVel".

3

EndVel 항목: 동작 완료 속도. API 함수의 인수와 변수 "EndVel".

TPhase1 항목: StrVel에서 ConstVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase1".

TPhase2 항목: ConstVel에서 EndVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase2".

S-Curve 체크 박스: S-curve 속도 곡선을 사용하고자 할 경우 반드시 체크해야 합니다.

ABS 체크 박스: 절대 좌표의 방식을 참고하여 동작의 변위를 실행하고자 할 경우 반드시 체크해야 합니다.

(17) 서보 모터 동력ON / OFF (servo on / servo off) 설정

그림 3.6.2.4의 "SVON" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode =_ECAT_Slave_Motion_Set_Svon(gESCCardNo, gNodeID[i], gSlotID[i],ON_OFF);
// ON_OFF: 0 – 서보 동력 OFF, 1 – 서보 동력 ON
```

(18) Arc 항목 선택하여 2축 원형 동작 제어 실행(원 중심+각도)

그림 3.6.2.4의 "←" 또는 "→" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode = _ECAT_Slave_CSP_Start_Arc_Move(gESCCardNo, gNodeID, gSlotID,
CenPoint, Angle, StrVel, ConstVel, EndVel, Tacc, Tdec, gbIsSCurve, gbIsABS);
//gbIsSCurve 1: T-Curve 2: S-Curve  gbIsABS  0: 상대적 변위 1: 절대적 변위
```

(19) Spiral 항목 선택하여 2축 원형 동작 제어 실행(최종 좌표+각도)

그림 3.6.2.4의 "←" 또는 "→" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode = _ECAT_Slave_CSP_Start_Arc2_Move(gESCCardNo, gNodeID, gSlotID, EndPoint, Angle, StrVel, MaxVel, EndVel, Tacc, Tdec, gbIsSCurve, gbIsABS);
//gbIsSCurve 1: T-Curve 2: S-Curve  gbIsABS  0: 상대적 변위 1: 절대적 변위
```

(20) Spiral2 항목 선택하여 2축 원형 동작 제어 실행(원 중심+최종 좌표)

그림 3.6.2.4의 "←" 또는 "→" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode = _ECAT_Slave_CSP_Start_Arc3_Move(gESCCardNo, gNodeID, gSlotID, CenPoint,EndPoint,Dir,StrVel, ConstVel,EndVel,Tacc,Tdec,gbIsSCurve, gbIsABS);
//gbIsSCurve 1: T-Curve 2: S-Curve  gbIsABS  0: 상대적 변위 1: 절대적 변위
```

(21) 3축 동작 모드(Heli, Line3, Sphere) 및 동작 stroke 설정

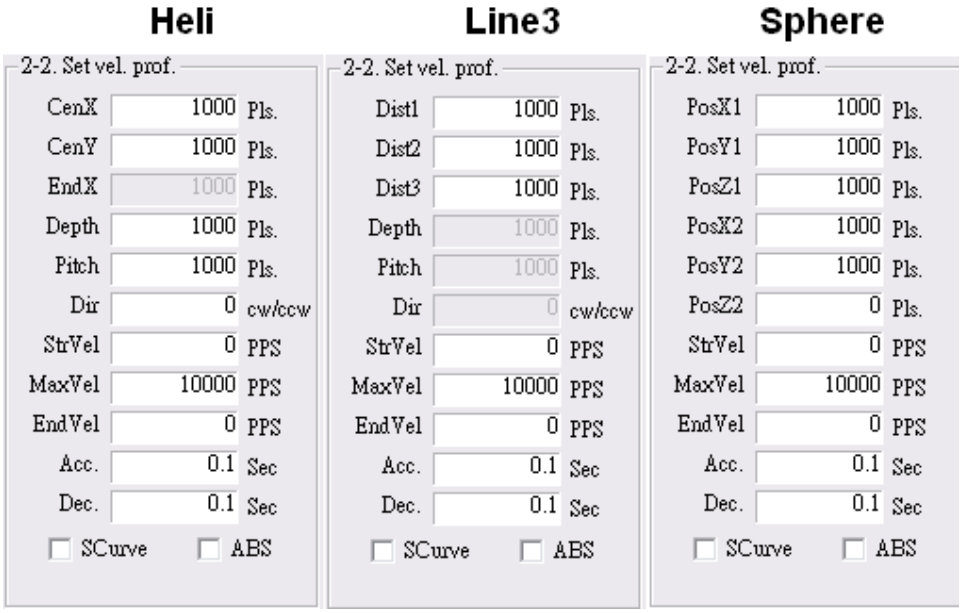


그림 3.6.2.10

Heli 파라미터 설정:

CenX 항목: 지정축에 상응하는 중심 위치 X.API 함수의 인수와 변수 "CenterPoint".

CenY 항목: 지정축에 상응하는 중심 위치 Y.API 함수의 인수와 변수 "CenterPoint".

Depth 항목: 지정축에 상응하는 위치의 깊이(전체 Z방향의 높이) API 함수의 인수와 변수 "Depth".

Pitch 항목: 지정한 두 나선 사이의 상대적 높이. API 함수의 인수와 변수 "Pitch".

Dir 항목: 원형 동작의 방향(0: 시계 방향 1: 시계 반대 방향). API 함수의 인수 "Dir".

StrVel 항목: 동작 초기 속도. API 함수의 인수와 변수 "StrVel".

ConstVel 항목: 동작 정상 속도. API 함수의 인수와 변수 "ConstVel".

EndVel 항목: 동작 완료 속도. API 함수의 인수와 변수 "EndVel".

TPhase1 항목: StrVel에서 ConstVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase1".

TPhase2 항목: ConstVel에서 EndVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase2".

S-Curve 체크 박스: S-curve 속도 곡선을 사용하고자 할 경우 반드시 체크해야 합니다.

ABS 체크 박스: 절대 좌표의 방식을 참고하여 동작의 변위를 실행하고자 할 경우 반드시 체크해야 합니다.

3

Line3 파라미터 설정:

Dist1 항목: X축의 동작 stroke 파라미터. API 함수의 인수와 변수 "DistArray".

Dist2 항목: Y축의 동작 stroke 파라미터. API 함수의 인수와 변수 "DistArray".

Dist3 항목: Z축의 동작 stroke 파라미터. API 함수의 인수와 변수 "DistArray".

StrVel 항목: 초기 속도. API 함수의 인수와 변수 "StrVel".

ConstVel 항목: 정상 속도. API 함수의 인수와 변수 "ConstVel".

EndVel 항목: 동작 완료 속도. API 함수의 인수와 변수 "EndVel".

TPhase1 항목: StrVel에서 ConstVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase1".

TPhase2 항목: ConstVel에서 EndVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase2".

S-Curve 체크 박스: S-curve 속도 곡선을 사용하고자 할 경우 반드시 체크해야 합니다.

ABS 체크 박스: 절대 좌표의 방식을 참고하여 동작의 변위를 실행하고자 할 경우 반드시 체크해야 합니다.

Sphere 파라미터 설정:

PosX1 항목: X축의 동작 stroke 파라미터. API 함수의 인수와 변수 "Target1Point".

PosY1 항목: Y축의 동작 stroke 파라미터. API 함수의 인수와 변수 "Target1Point".

PosZ1 항목: Z축의 동작 stroke 파라미터. API 함수의 인수와 변수 "Target1Point".

PosX2 항목: X축의 동작 stroke 파라미터. API 함수의 인수와 변수 "Target2Point".

PosY2 항목: Y축의 동작 stroke 파라미터. API 함수의 인수와 변수 "Target2Point".

PosZ2 항목: Z축의 동작 stroke 파라미터. API 함수의 인수와 변수 "Target2Point".

StrVel 항목: 초기 속도. API 함수의 인수와 변수 "StrVel".

ConstVel 항목: 정상 속도. API 함수의 인수와 변수 "ConstVel".

EndVel 항목: 동작 완료 속도. API 함수의 인수와 변수 "EndVel".

TPhase1 항목: StrVel에서 ConstVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase1".

TPhase2 항목: ConstVel에서 EndVel까지 소요된 시간. API 함수의 인수와 변수 "TPhase2".

S-Curve 체크 박스: S-curve 속도 곡선을 사용하고자 할 경우 반드시 체크해야 합니다.

ABS 체크 박스: 절대 좌표의 방식을 참고하여 동작의 변위를 실행하고자 할 경우 반드시 체크해야 합니다.

(22) 서보 모터 동력ON / OFF (servo on / servo off) 설정

그림 3.6.2.4의 "SVON" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode =_ECAT_Slave_Motion_Set_Svon(gESCCardNo, gNodeID[i], gSlotID[i], ON_OFF);
// ON_OFF: 0 – 서보 동력 OFF, 1 – 서보 동력 ON
```

(23) Heli 항목 선택하여 3축 나선 동작 제어 실행

그림 3.6.2.4의 "←" 또는 "→" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode = _ECAT_Slave_CSP_Start_Heli_Move(gESCCardNo, gNodeID,
gSlotID,
CenPoint,Depth,Pitch,Dir,StrVel,ConstVel,EndVel,Tacc,Tdec,gbIsSCurve,gbIsABS);
//gbIsSCurve 1: T-Curve 2: S-Curve    gbIsABS    0: 상대적 변위 1: 절대적 변위
```

(24) Line3 항목 선택하여 3축 직선 동작 제어 실행

그림 3.6.2.4의 "←" 또는 "→" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode = _ECAT_Slave_CSP_Start_Multiaxes_Move(gESCCardNo, 3,
gNodeID, gSlotID, Dist, StrVel, MaxVel, EndVel, Tacc, Tdec, gbIsSCurve,
gbIsABS);
//gbIsSCurve 1: T-Curve 2: S-Curve    gbIsABS    0: 상대적 변위 1: 절대적 변위
```

(25) Sphere 항목 선택하여 3축 구형 동작 제어 실행

그림 3.6.2.4의 "←" 또는 "→" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
RetCode = _ECAT_Slave_CSP_Start_Sphere_Move(gESCCardNo, gNodeID,
gSlotID, Dist, Dist2, StrVel, ConstVel, EndVel, Tacc, Tdec, gbIsSCurve,
gbIsABS);
//gbIsSCurve 1: T-Curve 2: S-Curve    gbIsABS    0: 상대적 변위 1: 절대적 변위
```

동작 설명도는 아래의 그림과 같습니다:

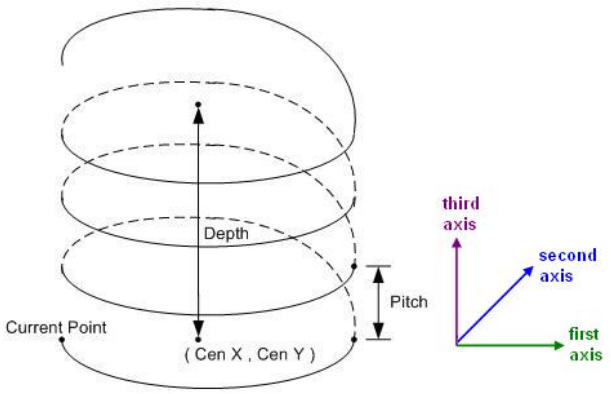


그림 3.6.2.11

3

(26) 모니터링 표시

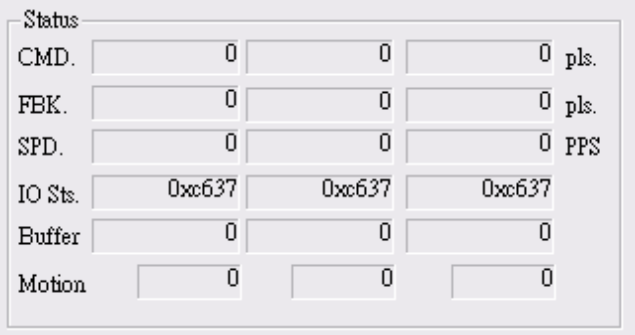


그림 3.6.2.12

동작 계수값:

```
RetCode = _ECAT_Slave_Motion_Get_Command(gESCCardNo, gNodeID,
gSlotID, &Cmd); // 명령값을 확인합니다
RetCode = _ECAT_Slave_Motion_Get_Position(gESCCardNo, gNodeID,
gSlotID, &Pos); // Feedback meter의 수치를 확인합니다
```

동작 상태: Motion Status:

```
RetCode = _ECAT_Slave_Motion_Get_Current_Speed(gESCCardNo,gNodeID,
gSlotID, &Spd); // 동작 실행의 속도 값을 확인합니다
RetCode = _ECAT_Slave_Motion_Get_StatusWord(gESCCardNo, gNodeID,
gSlotID, &Ststus); // 현재 상태를 확인합니다
RetCode = _ECAT_Slave_Motion_Get_Mdone(gESCCardNo, gNodeID, gSlotID,
&MCDone); // 모터의 현재 동작 상태를 확인합니다
RetCode = _ECAT_Slave_Motion_Get_Buffer_Length(gESCCardNo, gNodeID,
gSlotID, &BufLen); // 현재 Buffer 상태를 확인합니다
```

27) 동작 계수값 리셋 및 ALM 삭제

그림 3.6.2.4의 "RESET" 버튼을 클릭해 리셋 명령을 실행합니다:

```
RetCode = _ECAT_Slave_Motion_Set_Position(gESCCardNo, gNodeID,
gSlotID, 0); //feedback 삭제를 먼저 실행합니다
RetCode = _ECAT_Slave_Motion_Set_Command(gESCCardNo, gNodeID,
gSlotID, 0); //command 삭제를 나중에 실행합니다
```

그림 3.6.2.4의 "RALM" 버튼을 클릭해 ALM 삭제 명령을 실행합니다:

```
RetCode = _ECAT_Slave_Motion_Ralm(gESCCardNo, gNodeID, gSlotID);
//ALM Code 삭제
```

28) 동작 정지

그림 3.6.2.4의 "STOP" 버튼을 클릭하여 감속 정지를 실행합니다:

```
RetCode = _ECAT_Slave_Motion_Sd_Stop(gESCCardNo, gNodeID[0],
gSlotID[0], Tdec);
```

해당 예시는 감속 정지 방식을 사용하여 동작의 변위를 정지시키는 것으로 설정한 감속 시간에 따라 천천히 동작을 정지시킵니다.

(29) 프로그램 종료

"Exit" 버튼을 클릭하여 프로그램을 종료합니다.
함수를 건너뛰기 위해서는 반드시 "_ECAT_Master_Reset"와
"_ECAT_Master_Close"를 실행해야 합니다. 이 두가지 API 함수의 사용 방법에 대해서는 3.1.2절 "프로그램 종료"의 함수 작동 부분을 참고하십시오.

3.7 EtherCAT Slave IO 제어(디지털 입력)

3.7.1 함수 표

함수의 명칭
_ECAT_Slave_DIO_Get_Input_Value

■ **속성**

EtherCAT RTX version	EtherCAT Card version
○	○

3

3.7.2 응용 예제

프로그램 외관

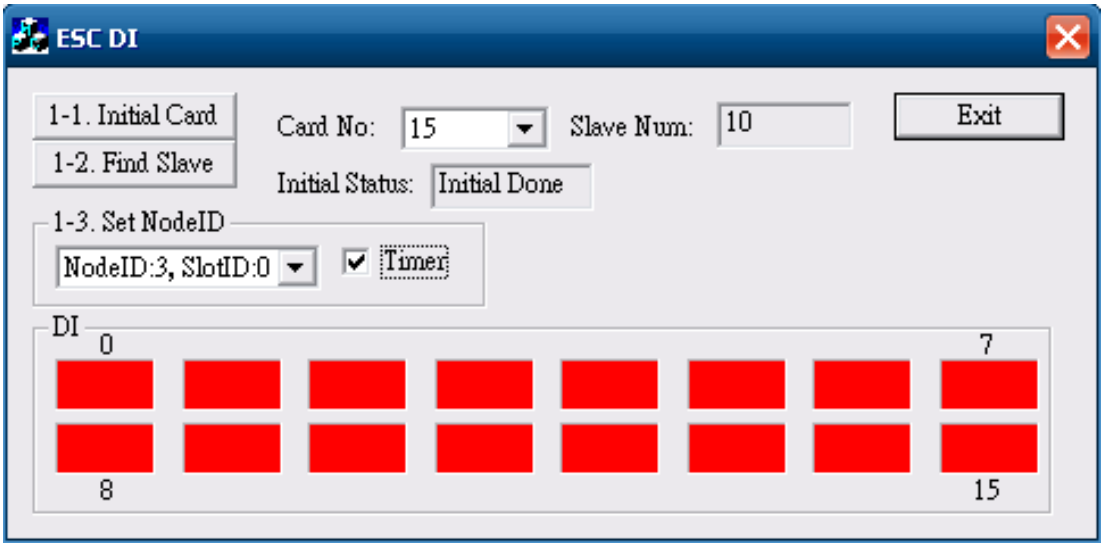


그림 3.7.2.1

(1) 인터페이스 카드의 실행과 초기화

그림 3.7.2.1의 "Initial Card" 버튼을 클릭하여 인터페이스 카드 초기화를 실행합니다.

그림 3.7.2.1의 "Find Slave" 버튼을 클릭해 연결된 모듈을 검색합니다.

인터페이스 카드에 대한 자세한 설명은 3.1.2절의 "인터페이스 카드 실행"과 "인터페이스 카드 초기화"의 내용을 참고하십시오.

(2) NodeID, SlotID 선택 및 모듈 입력 상태 표시

그림 3.7.2.2

NodeID, SlotID를 선택하여 "Timer"에 체크하면 체크 박스에 모듈 입력 상태가 표시됩니다

Set NodeID 항목: API 함수의 인수와 변수 "AxisNo".

Timer 체크 박스: 체크 시 모듈 입력 상태가 표시됩니다. 체크하지 않을 경우 표시되지 않습니다.

(3) 디지털 입력(Slave DI)

디지털 출력 모듈에서 송출한 정보를 디지털 입력 포트에서 확인하려면 R1-EC-60X2 모듈을 사용하여 다음 프로그램을 실행해야 합니다:

```
RetCode = _ECAT_Slave_DIO_Get_Input_Value(gESCCardNo, gNodeID, gSlotID, &gValue);
```

그림 3.7.2.3과 같이 R1-EC-60X2 모듈에 입력할 신호가 있다고 나타나지 않았습니다.

그림 3.7.2.3

(4) 프로그램 종료

"Exit" 버튼을 클릭하여 프로그램을 종료합니다.

함수를 건너뛰기 위해서는 반드시 "_ECAT_Master_Reset"와 "_ECAT_Master_Close"를 실행해야 합니다. 이 두가지 API 함수의 사용 방법에 대해서는 3.1.2절 "프로그램 종료"의 함수 작동 부분을 참고하십시오.

3.8 EtherCAT Slave IO 제어(디지털 출력)

3.8.1 함수 표

함수의 명칭
_ECAT_Slave_DIO_Set_Output_Value
_ECAT_Slave_DIO_Get_Output_Value

■ 속성

EtherCAT RTX version	EtherCAT Card version
○	○

3

3.8.2 응용 예제

프로그램 외관

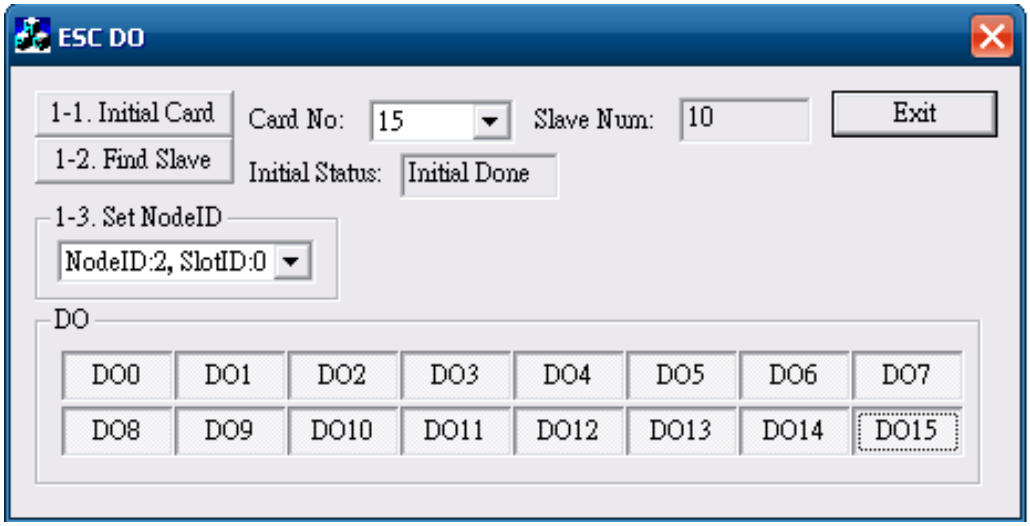


그림 3.8.3.1

(1) 인터페이스 카드의 실행과 초기화

그림 3.8.3.1의 "Initial Card" 버튼을 클릭해 인터페이스 카드 초기화를 실행합니다.

그림 3.8.3.1의 "Find Slave" 버튼을 클릭해 연결된 모듈을 검색합니다.

인터페이스 카드에 대한 자세한 설명은 3.1.2절의 "인터페이스 카드 실행"과 "인터페이스 카드 초기화"의 내용을 참고하십시오.

(2) NodeID, SlotID 선택

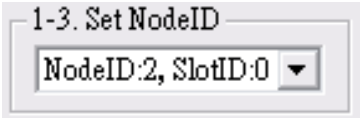


그림 3.8.3.2

NodeID, SlotID를 선택합니다

Set NodeID 항목**:** API 함수의 인수와 변수 "AxisNo"

(3) 디지털 출력 (Slave DO)

디지털 출력 포트에서 정보를 송출하려면 R1-EC-7062 모듈을 사용하여 다음 프로그램을 실행해야 합니다:

```
RetCode = ECAT_Slave_DIO_Set_Output_Value(gESCCardNo,gNodeID,gSlotID,
gValue);
```

아래의 프로그램을 실행하여 디지털 출력 모듈의 출력 상태를 확인할 수 있습니다:

```
RetCode                                                    =
_ECAT_Slave_DIO_Get_Output_Value(gESCCardNo,gNodeID,gSlotID,
&gValue);
```

아래의 그림과 같이 R1-EC-7062 모듈 Port0의 BIT0-15 신호 출력을 설정합니다.

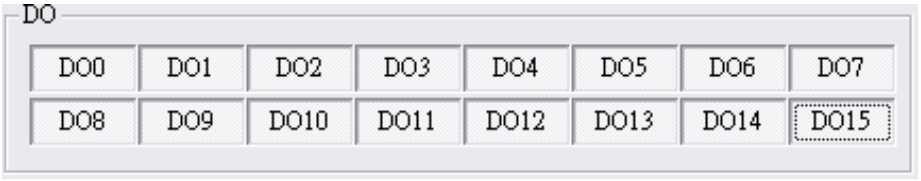


그림 3.8.3.3

(4) 프로그램 종료

"Exit" 버튼을 클릭하여 프로그램을 종료합니다.

함수를 건너뛰기 위해서는 반드시 "_ECAT_Master_Reset"와 "_ECAT_Master_Close"를 실행해야 합니다. 이 두가지 API 함수의 사용 방법에 대해서는 3.1.2절 "프로그램 종료"의 함수 작동 부분을 참고하십시오.

3.9 EtherCAT 아날로그 입력 모듈의 응용-R1-EC-8124

3.9.1 함수 표

함수의 명칭
_ECAT_Slave_AIO_Set_Input_RangeMode
_ECAT_Slave_R1_EC8124_Set_Input_AverageMode
_ECAT_Slave_AIO_Set_Input_ConvstFreq_Mode
_ECAT_Slave_AIO_Get_Input_Value

■ **속성**

EtherCAT RTX version	EtherCAT Card version
○	○

3.9.2 응용 예제

3

프로그램 외관

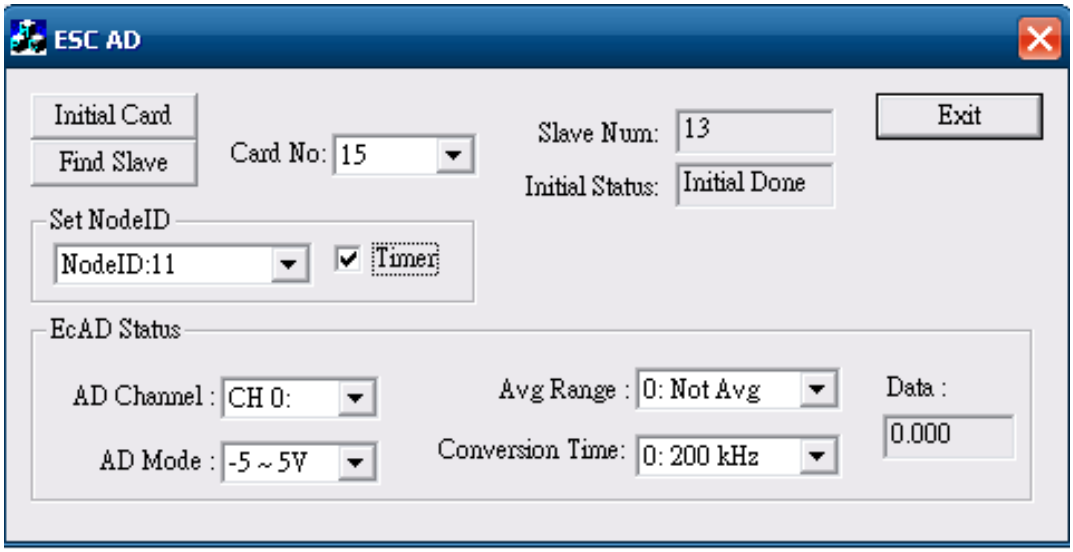


그림 3.9.2.1

(1) 인터페이스 카드의 실행과 초기화

그림 3.9.2.1의 "Open card" 버튼을 클릭해 인터페이스 카드의 초기화를 실행합니다.

그림 3.9.2.1의 "Find Slave" 버튼을 클릭해 연결된 모듈을 검색합니다.

인터페이스 카드에 대한 자세한 설명은 3.1.2절의 "인터페이스 카드 실행"과 "인터페이스 카드 초기화"의 내용을 참고하십시오.

(2) NodeID, SlotID를 선택하여 상태 표시하기

그림 3.9.2.2

NodeID, SlotID를 선택하여 "Timer"에 체크하면 체크 박스에 상태가 표시됩니다

Set NodeID 항목: API 함수의 인수와 변수 "AxisNo".

Timer 체크 박스: 체크 시 상태가 표시됩니다. 체크하지 않을 경우 표시되지 않습니다.

(3) AD Channel / AD Mode / Avg Range / Conversion Time 선택

그림 3.9.2.3

AD Channel 항목: AD 활성화 입력의 Channel 위치를 선택합니다. API 함수의 인수와 변수 "SlotNo".

AD Mode 항목: AD 표시 범위 모드를 선택합니다. API 함수의 인수와 변수 "RangeMode".

Avg Range 항목: 파형 표시 계산 주파수를 선택합니다. API 함수의 인수와 변수 "AvgMode".

Conversion Time 항목: Conversion Time 모드를 선택합니다. API 함수의 인수와 변수 "Mode".

(4) 그림 3.9.2.3의 AD Channel 및 AD Mode를 선택하면 다음 프로그램이 실행됩니다:

```
RetCode = _ECAT_Slave_AIO_Set_Input_RangeMode(gESCCardNo,gNodeID, gSlotID, Mode); // SlotID는 AD Channel입니다
```

그림 3.9.2.3의 Avg Range를 선택하면 다음 프로그램이 실행됩니다:

```
RetCode = _ECAT_Slave_R1_EC8124_Set_Input_AverageMode(gESCCardNo, gNodeID, gSlotID, AvgMode); // SlotID는 AD Channel입니다
```

그림 3.9.2.3의 Conversion Time을 선택하면 다음 프로그램이 실행됩니다:

```
RetCode = _ECAT_Slave_AIO_Set_Input_ConvstFreq_Mode(gESCCardNo, gNodeID, gSlotID, Mode); // SlotID는 AD Channel입니다
```

그림 3.9.2.3의 Data가 나타나면 아래의 프로그램이 실행됩니다:

```
RetCode = _ECAT_Slave_AIO_Get_Input_Value(gESCCardNo,gNodeID,gSlotID, &Value); // SlotID는 AD Channel입니다
```

(5) 프로그램 종료

"Exit" 버튼을 클릭하여 프로그램을 종료합니다.

함수를 건너뛰기 위해서는 반드시 "_ECAT_Master_Reset"와 "_ECAT_Master_Close"를 실행해야 합니다. 이 두가지 API 함수의 사용 방법에 대해서는 3.1.2절 "프로그램 종료"의 함수 작동 부분을 참고하십시오.

3

3.10 EtherCAT 아날로그 출력 모듈의 응용-R1-EC-9144

3.10.1 함수 표

함수의 명칭
_ECAT_Slave_AIO_Set_Output_RangeMode
_ECAT_Slave_AIO_Set_Output_OverRange_Enable
_ECAT_Slave_R1_EC9144_Get_Output_ReturnCode
_ECAT_Slave_AIO_Set_Output_Value
_ECAT_Slave_AIO_Get_Output_Value

■ 속성

EtherCAT RTX version	EtherCAT Card version
○	○

3.10.2 응용 예제

프로그램 외관

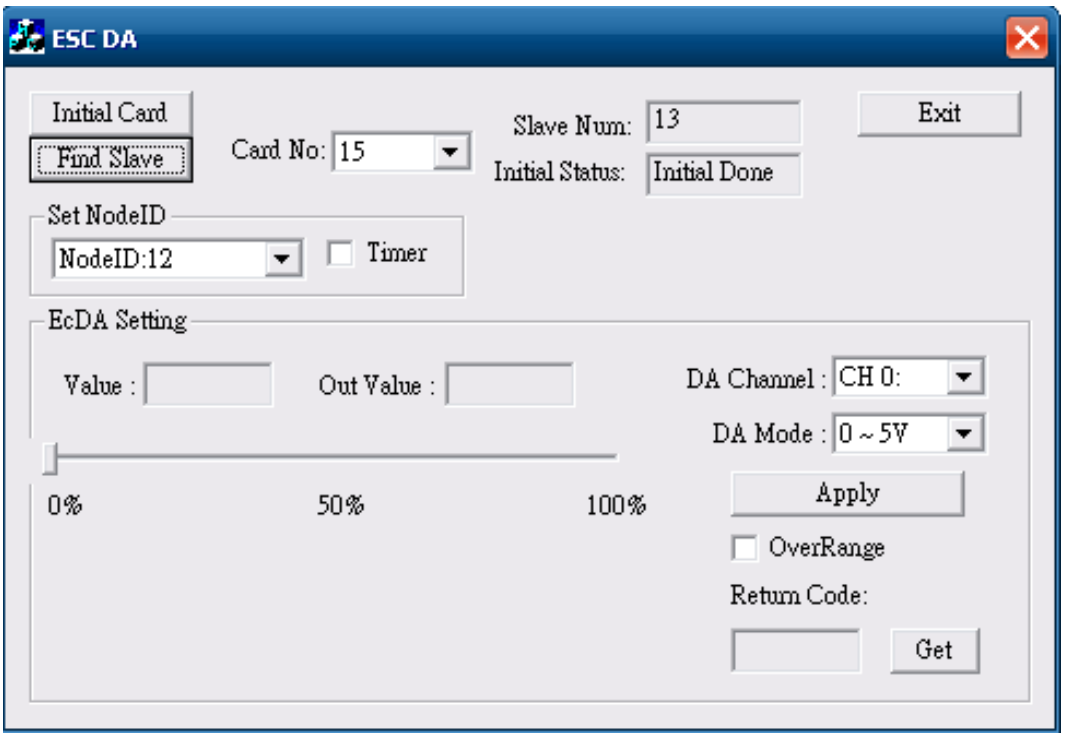


그림 3.10.2.1

(1) 인터페이스 카드의 실행과 초기화

그림 3.10.2.1의 "Initial Card" 버튼을 클릭해 인터페이스 카드 초기화를 실행합니다.

그림 3.10.2.1의 "Find Slave" 버튼을 클릭해 연결된 모듈을 검색합니다.

인터페이스 카드에 대한 자세한 설명은 3.1.2절의 "인터페이스 카드 실행"과 "인터페이스 카드 초기화"의 내용을 참고하십시오.

(2) NodeID, SlotID를 선택하여 상태 표시하기

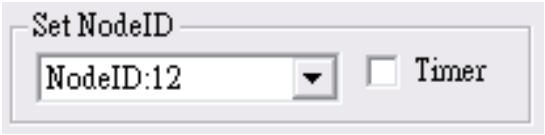


그림 3.10.2.2

NodeID, SlotID를 선택하여 "Timer"에 체크하면 체크 박스에 상태가 표시됩니다

Set NodeID 항목: API 함수의 인수와 변수 "AxisNo".

Timer 체크 박스: 체크 시 상태가 표시됩니다. 체크하지 않을 경우 표시되지 않습니다.

(3) DA Channel 선택 및 DA 표시 모드 선택:

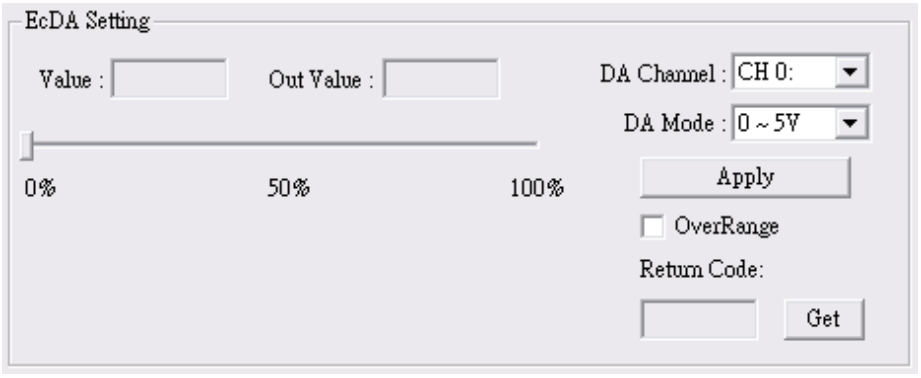


그림 3.10.2.3

DA Channel 항목: 사용할 DA Channel 번호(0 ~ 3)를 입력합니다.

DA Mode 항목: DA 표시 범위 모드, API 함수의 인수와 변수 "Mode"를 선택합니다.

Value 항목: 선택 범위 모드 및 조절바 위치에 따라 예상 전압값을 표시합니다.

OutValue 항목: 실제 출력된 전압값을 표시합니다.

Apply 항목: Apply 버튼을 클릭하면 Value 예상 전압값이 실제 출력값으로 전환됩니다.

Over Range 항목: 해당 항목에 체크하면 전압 출력값을 "10%" 증가시킬 수 있습니다.

Return Code 항목: DA 상태를 표시합니다.

3

(4) 그림 3.10.2.3의 "DA Channel" 및 "DA Mode" 옵션을 선택하면 다음 프로그램이 실행됩니다:

```
/* DA 표시 모드를 설정합니다 */
RetCode = _ECAT_Slave_AIO_Set_Output_RangeMode(gESCCardNo, gNodeID,
gSlotID, Mode);
```

그림 3.10.2.3의 "Apply" 옵션을 선택하면 다음 프로그램이 실행됩니다:

```
/* DA 출력값을 설정 */
RetCode = _ECAT_Slave_AIO_Set_Output_Value(gESCCardNo, gNodeID,
gSlotID, Value);
```

그림 3.10.2.3의 "OverRange" 옵션에 체크하면 다음 프로그램이 실행됩니다:

```
/* Enable Over Range 기능*/
RetCode =_ECAT_Slave_AIO_Set_Output_OverRange_Enable(gESCCardNo,
gNodeID, gSlotID, Enable);
```

그림 3.10.2.3의 Return Code "Get" 옵션을 클릭하면 다음 프로그램이 실행됩니다:

```
/* DA 상태를 확인합니다 */
RetCode =_ECAT_Slave_R1_EC9144_Get_Output_ReturnCode(gESCCardNo,
gNodeID, gSlotID, &RtCode);
```

아래의 프로그램을 실행하면 아날로그 출력 모듈의 출력값을 확인할 수 있고, 해당 값은 그림 3.10.2.3의 "Out Value"에 나타납니다:

```
/* 아날로그 출력 모듈 값이 나타납니다 */
RetCode =_ECAT_Slave_AIO_Set_Output_Value(gESCCardNo, gNodeID,
gSlotID, Value);
```

(5) 프로그램 종료

"Exit" 버튼을 클릭하여 프로그램을 종료합니다.

함수를 건너뛰기 위해서는 반드시 "_ECAT_Master_Reset"와 "_ECAT_Master_Close"를 실행해야 합니다. 이 두가지 API 함수의 사용 방법에 대해서는 3.1.2절 "프로그램 종료"의 함수 작동 부분을 참고하십시오.

3.11 EtherCAT Compare 축 카드의 응용- PCI-L221-B1

3.11.1 함수 표

함수의 명칭
_ECAT_Compare_Set_Channel_Position
_ECAT_Compare_Get_Channel_Position
_ECAT_Compare_Set_Ipulser_Mode
_ECAT_Compare_Set_Channel_Direction
_ECAT_Compare_Set_Channel_Trigger_Time
_ECAT_Compare_Set_Channel_One_Shot
_ECAT_Compare_Set_Channel_Source
_ECAT_Compare_Set_Channel_Enable
_ECAT_Compare_Channel0_Position
_ECAT_Compare_Set_Channel0_Trigger_By_GPIO
_ECAT_Compare_Set_Channel1_Output_Enable
_ECAT_Compare_Set_Channel1_Output_Mode
_ECAT_Compare_Get_Channel1_IO_Status
_ECAT_Compare_Set_Channel1_GPIO_Out
_ECAT_Compare_Set_Channel1_Position_Table
_ECAT_Compare_Get_Channel1_Positioin_Table_Level
_ECAT_Compare_Get_Channel1_Position_Table_Count
_ECAT_Compare_Set_Channel_Polarity
_ECAT_Compare_Reuse_Channel1_Postion_Table
_ECAT_Compare_Reuse_Channel1_Position_Table_Level

■ 속성

EtherCAT RTX version	EtherCAT Card version
○	○

3.11.2 응용 예제

3

프로그램 외관

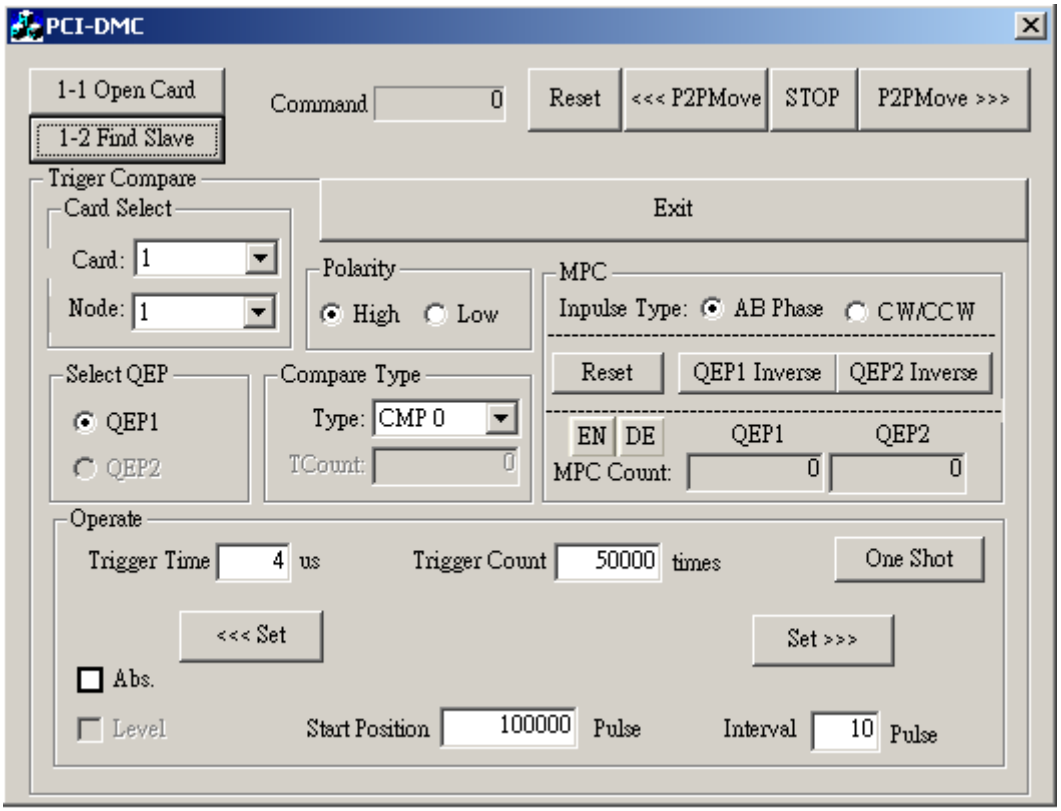


그림 3.11.2.1

(1) 인터페이스 카드의 실행과 초기화

※ 우선 설치한 것이 PCI-L221-B1 인터페이스 카드인지 확인합니다.

그림 3.11.2.1의 "Open card" 버튼을 클릭해 인터페이스 카드의 초기화를 실행합니다.

그림 3.11.2.1의 "Find Slave" 버튼을 클릭해 연결된 모듈을 검색합니다.

인터페이스 카드에 대한 자세한 설명은 3.1.2절의 "인터페이스 카드 실행"과 "인터페이스 카드 초기화"의 내용을 참고하십시오.

(2) 카드 번호, Node 번호, QEP 선택

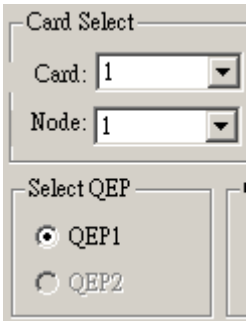


그림 3.11.2.2

Card 항목: 사용할 PCI_DMC_B01 카드 번호를 입력합니다.

Node 항목: Node 번호 사용을 선택하고, QEP1와 QEP2를 동일하게 선택해야 합니다.

QEP1 항목: 예: Channel0 사용을 선택하고, Node 번호는 Node1를 선택해야 합니다.

QEP2 항목: 예: Channel1 사용을 선택하고, Node 번호는 Node2를 선택해야 합니다.

(3) Compare Type Select 및 Polarity 설정:

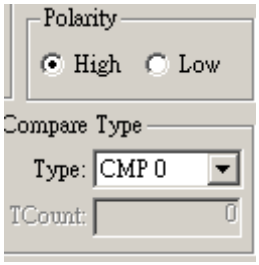


그림 3.11.2.3

High 항목: 해당 항목을 선택하고, 아래의 설정을 실행합니다: **0: High**
rt = _ECAT_Compare_Set_Channel_Polarity (CpCardNo,0);
Low 항목: 해당 항목을 선택하고, 아래의 설정을 실행합니다: **1: Low**
rt = _ECAT_Compare_Set_Channel_Polarity (CpCardNo,1);
Type 항목: Compare1 또는 Compare2 기능을 선택합니다.
rt = _ECAT_Compare_Set_Channel_Source (CpCardNo,Compare_Type,CpQEP);
//Compare_type:0=COMP1,1=COMP2
TCount 항목: Trigger의 실제 실행 횟수.

3

(4) MPC 파라미터 설정:

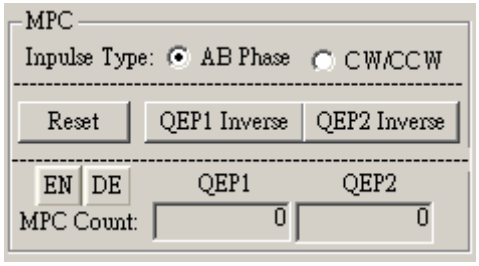


그림 3.11.2.4

EN/DE 항목: Compare 기능 **on / off,** 아래의 설정을 실행합니다.

```
rt = _ECAT_Compare_Set_Channel_Enable (CpCardNo,channel,1);
//1: on
```

Inpulse Type 항목: AB Phase 또는 CW / CCW 모드를 선택합니다. 아래의 설정을 실행합니다:

```
rt = _ECAT_Compare_Set_Ipulser_Mode (CpCardNo,mode);
// 0: AB Phase    1: CW/CCW
```

Reset 항목: 해당 버튼을 클릭해 QEP1와 QEP2 MPC Count를 삭제하고, 아래의 설정을 실행합니다.

```
rt = _ECAT_Compare_Set_Channel_Position (CpCardNo,channel,0);
```

QEP1 Inverse 항목: 역방향 버튼을 클릭해 다음과 같이 설정합니다:

```
rt = _ECAT_Compare_Set_Channel_Direction (CpCardNo,0,dir); // dir:0 or 1
```

QEP2 Inverse 항목: 역방향 버튼을 클릭해 다음과 같이 설정합니다:

```
rt = _ECAT_Compare_Set_Channel_Direction (CpCardNo,1,dir); // dir:0 or 1
```

(5) Operate 관련 정보 설정:

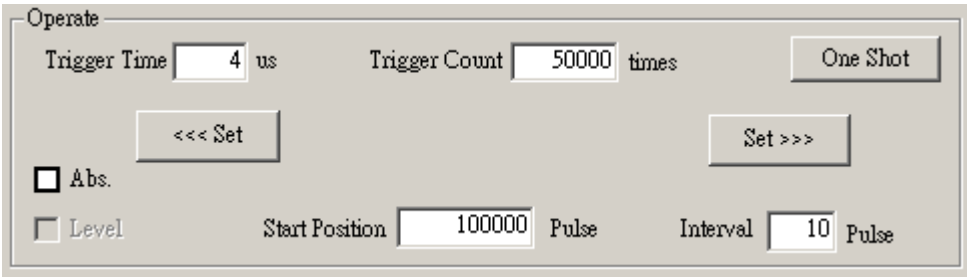


그림 3.11.2.5

Trigger Time 항목: 1회 Trigger 활성화 시간을 입력합니다.

Trigger count 항목: Trigger의 총 활성화 횟수를 입력합니다.

Start Position 항목: Trigger 활성화 시작 위치를 입력합니다.

Interval 항목: Trigger 활성화 주파수를 입력합니다. 예: 10을 입력하면 Trigger 실행 10회당 Pulse가 1회 활성화됨을 의미합니다.

ABS 체크 박스: 절대 좌표의 방식을 참고하여 Trigger 활성화를 실행하려면 반드시 체크해야 합니다.

Level 체크 박스: 체크하면 아래의 설정이 실행됩니다:

```
rt = _ECAT_Compare_Set_Channel1_Output_Mode (CpCardNo,mode);
// mode: 0은 Normal 타입을 의미하며, 1은 맞춤형 타입(자세한 내용은 39장의
Compare API 관련 설명을 참고)을 나타냅니다
```

One Shot 항목: 해당 버튼을 클릭하면 Trigger가 1회만 실행하며, 아래의 설정이 실행됩니다:

```
rt = _ECAT_Compare_Set_Channel_Trigger_Time (CpCardNo,compare_channel,
      time_us); // time_us=Trigger time
rt = _ECAT_Compare_Set_Channel_One_Shot (CpCardNo,compare_channel);
```

Set 항목: >>>와 <<< 두 개의 방향을 선택할 수 있습니다. 해당 버튼을 클릭하면 아래의 설정이 실행됩니다:

```
rt = _ECAT_Compare_Get_Channel_Position (CpCardNo,compare_channel,
      position);
rt = _ECAT_Compare_Set_Channel_Trigger_Time (CpCardNo,compare_channel,
      time_us); // time_us = Trigger time
If CompareType=CMP1   //Compare1
   rt = _ECAT_Compare_Channel0_Position (CpCardNo,start,dir,interval);
   // dir->0:CMP1 , 1:CMP2
else    //Compare2
```

우선 output disable 상태로 만듭니다

```
rt = _ECAT_Compare_Set_Channel1_Output_Enable (CpCardNo,0); // 0:off 1:on
```

Level 체크 박스에 체크하면 아래의 설정이 실행됩니다:

```
rt = _ECAT_Compare_Set_Channel1_Position_Table
(CpCardNo,pos_table,tab_size);
```

체크하지 않을 경우 다음과 같이 설정됩니다:

```
rt = _ECAT_Compare_Set_Channel1_Position_Table_Level
(CpCardNo,pos_table,level_table,tab_size);
```

output enable 상태로 만듭니다

```
rt = _ECAT_Compare_Set_Channel1_Output_Enable (CpCardNo,1); // 0:off 1:on
```

(6) Command 표시 및 테스트 조작 영역:

그림 3.11.2.6

Reset 항목: 해당 버튼을 클릭하면 Motion이 0으로 돌아갑니다.

P2PMove 항목: 해당 버튼을 클릭하면 Motion이 플러스 방향 또는 마이너스 방향으로 진행됩니다.

STOP 항목: 해당 버튼을 클릭하면 Motion 동작이 정지됩니다.

Command 항목: Motion 동작의 현재 위치를 표시합니다.

(7) 프로그램 종료

"Exit" 버튼을 클릭하여 프로그램을 종료합니다.

함수를 건너뛰기 위해서는 반드시 "_ECAT_Master_Reset"와 "_ECAT_Master_Close"를 실행해야 합니다. 이 두가지 API 함수의 사용 방법에 대해서는 3.1.2절 "프로그램 종료"의 함수 작동 부분을 참고하십시오.

3.12 EtherCAT SpeedContinue 응용

3.12.1 함수 표

함수의 명칭
_ECAT_Slave_CSP_Speed_Continue_Enable
_ECAT_Slave_CSP_Speed_Continue_Set_Mode
_ECAT_Slave_CSP_Speed_Continue_Combine_Ratio

■ 속성

EtherCAT RTX version	EtherCAT Card version
○	○

3.12.2 응용 예제

프로그램 외관

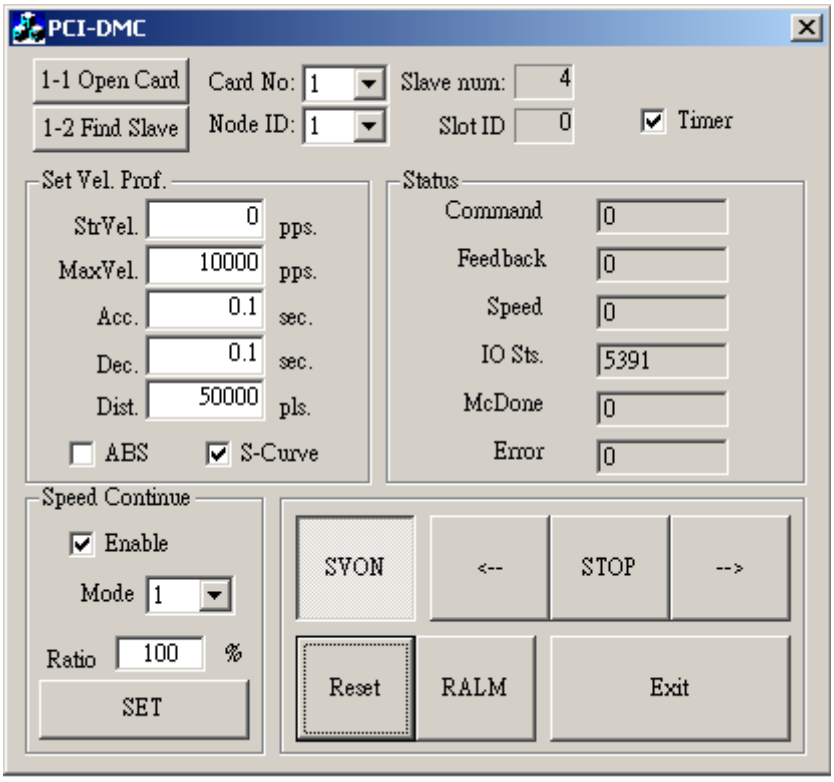


그림 3.12.2.1

3

(1) 인터페이스 카드의 실행과 초기화

그림 3.12.2.1의 "Open card" 버튼을 클릭해 인터페이스 카드의 초기화를 실행합니다.

그림 3.12.2.1의 "Find Slave" 버튼을 클릭해 연결된 모듈을 검색합니다.

인터페이스 카드에 대한 자세한 설명은 3.1.2절의 "인터페이스 카드 실행"과 "인터페이스 카드 초기화"의 내용을 참고하십시오.

(2) 드라이브의 Node ID를 설정하여 동작 상태 표시하기

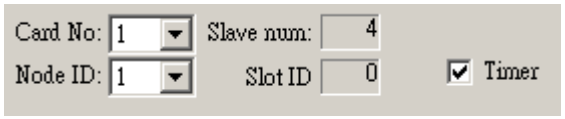


그림 3.12.2.2

Node ID 입력 후 "Timer"에 체크하면 체크 박스에 동작 상태가 표시됩니다

NodeID 항목: API 함수의 인수와 변수 "NodeID".

Timer 체크 박스: 체크 시 동작 상태가 표시됩니다. 체크하지 않을 경우 표시되지 않습니다.

(3) 동작 제어의 인수 내용값 입력

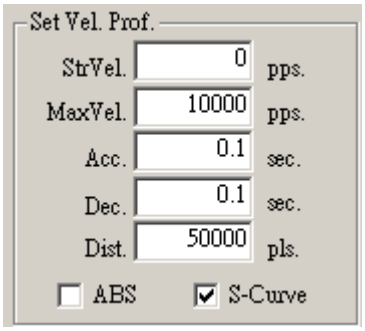


그림 3.12.2.3

StrVel 항목: 초기 속도. API 함수의 인수와 변수 "StrVel".

MaxVel 항목: 최대 속도. API 함수의 인수와 변수 "MaxVel".

Acc 항목: 최대 속도에 도달하는데 소요된 시간. API 함수의 인수와 변수 "acc".

Dec 항목: 최대 속도에서 0으로 떨어지는데 소요된 시간. API 함수의 인수와 변수 "dec".

Dist.항목: 동작 stroke.API 함수의 인수와 변수 "Distance".

ABS 체크 박스: 절대 좌표의 방식을 참고하여 동작의 변위를 실행하고자 할 경우 반드시 체크해야 합니다.

S-Curve 체크 박스: S-curve 속도 곡선을 사용하고자 할 경우 반드시 체크해야 합니다.

(4) Speed Continue 관련 파라미터 설정

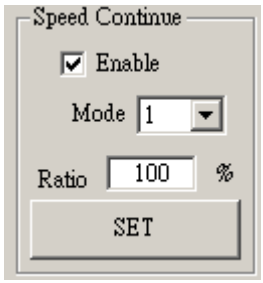


그림 3.12.2.4

Enable 체크 박스: Speed Continue 기능을 실행하려면 반드시 체크해야 합니다.

체크할 경우, 다음 프로그램이 실행됩니다:

```
rt = _ECAT_Slave_CSP_Speed_Continue(gECATCardNo, NodeID, SlotID , enable);
 //0：Disable 1：Enable
```

Mode 항목: 연속 속도 모드 설정. API 함수의 인수와 변수 "Mode".

Ratio 항목: 연속 속도 합성 백분율 설정. API 함수의 인수와 변수 "Ratio".

그림 3.12.2.4의 "SET" 버튼을 클릭하여 다음 프로그램을 실행합니다:

```
rt = _ECAT_Slave_CSP_Speed_Continue_Set_Mode(gECATCardNo, NodeID, SlotID , mode);
//0: 등가속 모드 1: 가속, 감속 모드 2: 최대 속도 모드
rt = _ECAT_Slave_CSP_Speed_Continue_Set_Combine_Ratio(gECATCardNo, NodeID, SlotID , ratio);   //0~100
```

(5) 서보 모터 동력ON / OFF (servo on / servo off) 설정

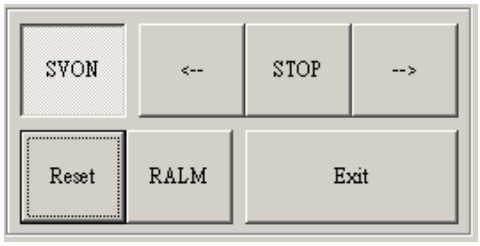


그림 3.12.2.5

그림 3.12.2.5의 "SVON" 버튼을 클릭해 다음 프로그램을 실행합니다:

```
rt = _ECAT_Slave_Motion_Set_Svon(gECATCardNo, NodeID, SlotID , ON_OFF);
// ON_OFF: 0 – 서보 동력 OFF; 1 – 서보 동력 ON.
```

3

(6) 그림 3.12.2.5의 "←" 또는 "→" 버튼을 클릭하여 연속 동작을 실행하면 다음 프로그램이 실행됩니다:

```
rt = _ECAT_Slave_CSP_Start_Move(gECATCardNo, NodeID, SlotID, Distance,
StrVel, ConstVel, EndVel, Tacc, Tdec, SCurve, IsAbs);
// S-curve 속도 단면의 절대 좌표를 참고하여 동작의 변위를 실행합니다
rt = _ECAT_Slave_CSP_Start_Move (gECATCardNo, NodeID, SlotID, Distance,
StrVel, ConstVel, EndVel, Tacc, Tdec, SCurve, IsAbs);
// T-curve 속도 단면의 절대 좌표를 참고하여 동작의 변위를 실행합니다
rt = _ECAT_Slave_CSP_Start_Move(gECATCardNo, NodeID, SlotID, -Distance,
StrVel, ConstVel, EndVel, Tacc, Tdec, SCurve, IsAbs);
// S-curve 속도 단면의 상대 좌표를 참고하여 동작의 변위를 실행합니다
rt = _ECAT_Slave_CSP_Start_Move(gECATCardNo, NodeID, SlotID, -Distance,
StrVel, ConstVel, EndVel, Tacc, Tdec, SCurve, IsAbs);
// T-curve 속도 단면의 상대 좌표를 참고하여 동작의 변위를 실행합니다
```

(7) 모니터링 표시

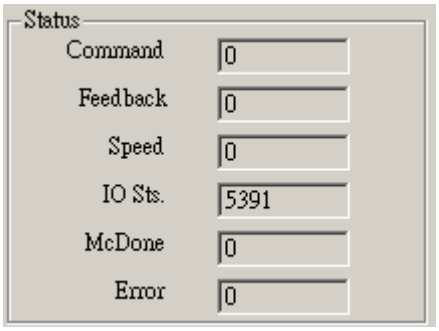


그림 3.12.2.6

동작 계수값:

```
rt = _ECAT_Slave_Motion_Get_Command(gECATCardNo, NodeID, SlotID,
&cmd);
// 명령값을 확인합니다
rt = _ECAT_Slave_Motion_Get_Position(gECATCardNo, NodeID, SlotID, &pos);
// Feedback meter의 수치를 확인합니다
```

동작 상태: Motion status:

```
rt = _ECAT_Slave_Motion_Get_Current_Speed(gECATCardNo, NodeID, SlotID,
&speed);
// 동작 실행의 속도 값을 확인합니다
rt = _ECAT_Slave_Motion_Get_StatusWord(gECATCardNo, NodeID, SlotID,
&MC_status);
// 현재 상태를 확인합니다
```

```
rt = _ECAT_Slave_Motion_Get_Mdone(gECATCardNo, NodeID, SlotID,
&MC_done);
// 모터의 현재 동작 상태를 확인합니다
```


(8) 동작 계수값 리셋, 동작 정지 및 Alm_Reset

그림 3.12.2.5의 "RESET" 버튼을 클릭해 리셋 명령을 실행합니다:

```
rt = _ECAT_Slave_Motion_Set_Position(gECATCardNo, NodeID, SlotID, 0);
//feedback 삭제를 먼저 실행합니다
rt = _ECAT_Slave_Motino_Set_Command(gECATCardNo, NodeID, SlotID, 0);
//command 삭제를 나중에 실행합니다
```

어떤 동작을 실행 중이든 그림 3.12.2.5의 "STOP" 버튼을 클릭하면 긴급 정지를 실행할 수 있습니다:

```
rt = _ECAT_Slave_Motion_Emg_Stop(gECATCardNo, NodeID, SlotID);
```

이 경우 긴급 정지 방식으로 동작의 변위를 중지시키며, 감속 시간을 0으로 설정하여 빠르게 동작을 정지시킬 수 있습니다. 동작 정지에 대한 자세한 내용은 뒷부분의 "동작 정지 제어 API" 장과 절의 설명을 참고하십시오.

그림 3.12.2.6의 Error 필드에 오류 신호가 나타날 때, 그림 3.12.2.5의 "RALM" 버튼을 클릭하여 Alm_Code 리셋 동작을 실행할 수 있습니다.

(9) 프로그램 종료

"Exit" 버튼을 클릭하여 프로그램을 종료합니다.

함수를 건너뛰기 위해서는 반드시 "_ECAT_Master_Reset"와 "_ECAT_Master_Close"를 실행해야 합니다. 이 두가지 API 함수의 사용 방법에 대해서는 3.1.2절 "프로그램 종료"의 함수 작동 부분을 참고하십시오.

(이 페이지는 공란으로 비워둡니다)

API 제어

4

다음은 EtherCAT API 가 사용하는 데이터 유형 및 값의 범위, 모든 API 명령 리스트에 대한 설명입니다.

4.1 데이터의 유형과 범위

4

설치 목록 "inc\VC\" 폴더에 "TYPE_DEF.H" 파일을 저장합니다. 이 파일은 일반적인 데이터 의 유형을 정의합니다. 아래의 표와 같이 해당 파일에 정의된 유형, 명칭, 표시된 의미, 값의 범위가 나타납니다.

명칭	설명	값의 범위
U8	8-bit ASCII 자수	0 ~ 255
I16	16-bit 부호가 있는 정수	-32768 ~ 32767
U16	16-bit 부호가 없는 정수	0 ~ 65535
I32	32-bit 부호가 있는 긴 정수	-2147483648 ~ 2147483647
U32	32-bit 부호가 없는 긴 정수	0 ~ 4294967295
F32	32-bit 단정도 부동소수점	-3.402823E38 ~ 3.402823E38
F64	64-bit 배정도 부동소수점	-1.797683134862315E308 ~ 1.797683134862315E309
Boolean	Boolean 값	TRUE, FALSE

4.2 함수 설명

Master Config API	
_ECAT_Master_Set_CycleTime	Master 명령, 사이클 타임을 설정합니다. □ Initial 앞에서 설정해야 합니다
_ECAT_Master_Get_CycleTime	Master 명령, 해당 축 카드의 사이클 타임을 조회합니다.
_ECAT_Master_NodeID_Alias_Enable	Master 명령, 사용자 정의 Node 번호의 사용 여부를 설정합니다. □Initial 완료 후 사용이 가능합니다.
_ECAT_Master_Get_SerialNo	PAC 또는 축 카드의 시퀀스 번호를 확인합니다.
_ECAT_Master_Get_DLL_SeqID	Master 명령, 현재 DLL 이 사용 중인 시퀀스 ID 를 확인합니다.
_ECAT_Autoconfig_Open_File	Master 명령, 파일에서 모듈 구성 및 DC 데이터를 확인합니다. □ Initial 이 완료 되어야만 사용할 수 있습니다.
_ECAT_Autoconfig_Save_File	Master 명령, 현재의 모듈 구성과 DC 데이터를 파일에 입력합니다.
_ECAT_Autoconfig_Set_Slave_DCTime	Master 명령, 각 Node 의 DC 시간 데이터를 설정합니다.
_EACT_Autoconfig_Clear_ConfigFile	Master 명령, 현재의 로딩 설정을 삭제합니다.
_ECAT_Autoconfig_Set_NodeID_Alias	Master 명령, 각 Node 의 사용자 정의 Node 번호를 설정합니다. □Initial 완료 후 사용이 가능합니다.
_ECAT_Autoconfig_Get_NodeID_Alias	Master 명령, 각 Node 의 사용자 정의 Node 번호를 확인합니다. □Initial 완료 후 사용이 가능합니다.
_ECAT_Autoconfig_Save_NodeID_Alias	Master 명령, 모듈의 메모리 블록에 사용자 정의 Node 번호를 실제로 입력합니다.

Master Initial API	
_ECAT_Master_Open	Master 명령, CreateFile 을 사용하여 축 카드의 장수를 검출하고, 메모리 영역을 생성합니다

4

Master Initial API	
_ECAT_Master_Initial	Master 명령, AutoConfig 를 실행하고, OP 모드로 전환하며, DC 등의 동작을 대기합니다
_ECAT_Master_Reset	Master 명령, 상태를 리셋하고, Init 모드로 변경합니다
_ECAT_Master_Close	Master 명령, 축 카드를 off 하고, 메모리를 해제합니다
_ECAT_Master_Get_CardSeq	Master 명령, 해당 축 카드의 카드 번호를 조회합니다
_ECAT_Master_Get_SlaveNum	Master 명령, 해당 축 카드의 Slave 개수를 조회합니다
_ECAT_Master_Get_Slave_Info	Master 명령, 해당 축의 Information 을 조회합니다
_ECAT_Master_Get_DC_Status	Master 명령, 해당 축 카드의 DC 상태, 시간, 오차 시간을 조회합니다
_ECAT_Master_Get_Connect_Status	Master 명령, 해당 축 카드의 연결 상태를 조회합니다
_ECAT_Master_Get_Api_BufferLength	Master 명령, 현재 시스템에서 처리가 완료되지 않은 각 Node 의 명령들을 확인합니다
_ECAT_Master_Get_Cycle_SpendTime	Master 명령, 매회 Cycle 의 Tx 와 Rx 의 소요시간과 이전의 최대 소요시간을 확인합니다
_ECAT_Master_Check_Initial_Done	Master 명령, DLL 레이어의 Initial 완료 여부를 조회합니다
_ECAT_Master_Get_Initial_ErrorCode	Master 명령, _ECAT_Master_Check_Initial_Done 을 통해 확인한 데이터에 오류가 발생하면 해당 API 를 통해 오류 코드를 확인합니다.
_ECAT_Master_Check_Working_Counter	Master 명령, 현재의 연결 상태에 이상이 없는지 확인합니다.
_ECAT_Master_Get_Return_Code _Message	Master 명령, 현재의 연결 상태에 이상이 없는지 확인합니다.

Slave 공통 API	
_ECAT_Slave_SDO_Send_Message	Slave 공통 명령, 해당 Node 에 SDO(CANopen 포맷)명령을 전송합니다.
_ECAT_Slave_SDO_Read_Message	Slave 공통 명령, 이 Node 의 현재 SDO (CANopen 포맷) 데이터를 확인합니다.

Slave 공통 API	
_ECAT_Slave_SDO_Quick_Send_ Message	Slave 공통 명령, 해당 Node 에 SDO(CANopen 포맷)명령을 전송하나 명령 전송 완료를 기다리지 않습니다
_ECAT_Slave_SDO_Quick_Read _Message	Slave 공통 명령, 이 Node 에 SDO (CANopen 포맷) 확인 명령을 하달하지만 명령의 하달 완료를 기다리지 않습니다.
_ECAT_Slave_SDO_Read_Response	Slave 공통 명령, 이 Node 에 대한 반환 데이터를 확인합니다.
_ECAT_Slave_SDO_Wait_All_Done	Slave 공통 명령, 복수 Node 에 현재 누적된 SDO 명령이 전부 완료될 때까지 대기하라는 명령을 전송합니다
_ECAT_Slave_SDO_Get_ErrorCode	Slave 공통 명령, Send_Message 또는 Read_ Message 에 ERR_ECAT_SDO_Return_Error 오류가 발생할 때, 해당 API 를 사용하여 오류 코드를 확인할 수 있습니다. 오류 코드는 CANopen 또는 모듈 개체 정의에서 참조할 수 있습니다
_ECAT_Slave_SDO_Check_Done	Slave 공통 명령, 단일 Node 에 현재 누적된 SDO 명령이 완료되었는지 확인합니다
_ECAT_Slave_PDO_Get_OD_Data	Slave 공통 명령, 해당 Node 에 대한 임의의 OD 코드 데이터를 확인하며, 해당 데이터는 PDO 구성에 매핑되어야 합니다
_ECAT_Slave_PDO_Set_OD_Data	Slave 공통 명령, 해당 Node 에 임의의 OD 코드 데이터를 전송하며, 해당 데이터는 PDO 구성에 매핑되어야 합니다
_ECAT_Slave_PDO_Get_Information	Slave 공통 명령, 각 Node PDO 구성의 기본 데이터를 확인합니다
_ECAT_Slave_PDO_Get_Detail _Mapping	Slave 공통 명령, 각 Node PDO 구성의 상세 매핑 데이터를 확인합니다
_ECAT_Slave_PDO_Get_Rx_Data	Slave 공통 명령, 모든 Node 의 PDO Rx 구성 데이터를 확인합니다
_ECAT_Slave_PDO_Get_Tx_Data	Slave 공통 명령, 모든 Node 의 PDO Tx 구성 데이터를 확인합니다
_ECAT_Slave_PDO_Set_Tx_Detail_Data	Slave 공통 명령, 각 Node 의 PDO Tx 상세 매핑 데이터를 설정합니다
_ECAT_Slave_PDO_Set_Tx_Data	Slave 공통 명령, 모든 Node 의 PDO Tx 구성 데이터를 입력합니다

4

Motion Slave 공통 API	
_ECAT_Slave_Motion_Get_MoveMode	MotionSlave 명령, 현재의 동작 제어 모드(PP / CSP / Home …)를 확인합니다
_ECAT_Slave_Motion_Get_ControlWord	MotionSlave 명령, 현재의 ControlWord 데이터를 확인합니다
_ECAT_Slave_Motion_Get_StatusWord	MotionSlave 명령, 현재의 StatusWord 데이터를 확인합니다
_ECAT_Slave_Motion_Get_Command	MotionSlave 명령, 현재의 Command 데이터를 확인합니다
_ECAT_Slave_Motion_Get_Position	MotionSlave 명령, 현재의 Position 데이터를 확인합니다
_ECAT_Slave_Motion_Get_Mdone	MotionSlave 명령, 현재의 Mdone 데이터를 확인합니다
_ECAT_Slave_Motion_Get_Current _Speed	MotionSlave 명령, 현재의 Current_Speed 데이터를 확인합니다
_ECAT_Slave_Motion_Set_Svon	MotionSlave 명령, Servo On / Off 를 설정합니다
_ECAT_Slave_Motion_Ralm	MotionSlave 명령, Alm 을 리셋합니다
_ECAT_Slave_Motion_Set_Position	MotionSlave 명령, Position 위치를 설정합니다
_ECAT_Slave_Motion_Set_Command	MotionSlave 명령, Command 위치를 설정합니다
_ECAT_Slave_Motion_Emg_Stop	MotionSlave 명령, 긴급 정지를 실행합니다
_ECAT_Slave_Motion_Sd_Stop	MotionSlave 명령, Slow Down Stop 감속 및 정지합니다
_ECAT_Slave_Motion_Set_TouchProbe _Config	MotionSlave 명령, "트리거 검색" 기능의 실행모드를 설정합니다
_ECAT_Slave_Motion_Set_TouchProbe _QuickStart	MotionSlave 명령, "트리거 검색" 기능을 빨리 활성화합니다
_ECAT_Slave_Motion_Set_TouchProbe _QuickDone	MotionSlave 명령, "트리거 검색" 기능을 빨리 리셋합니다
_ECAT_Slave_Motion_Set_TouchProbe _Disable	MotionSlave 명령, "트리거 검색" 기능을 종료합니다
_ECAT_Slave_Motion_Get_TouchProbe _Status	MotionSlave 명령, 현재 "트리거 검색" 기능의 상태를 확인합니다
_ECAT_Slave_Motion_Get_TouchProbe _Position	MotionSlave 명령, 현재 검색한 위치를 확인합니다

Motion Slave 공통 API	
_ECAT_Slave_Motion_Set_MoveMode	MotionSlave 명령, 현재 동작 모드를 설정합니다
_ECAT_Slave_Motion_Get_Target _Command	MotionSlave 명령, 현재 목표의 데이터를 확인합니다
_ECAT_Slave_Motion_Get_Buffer _Length	MotionSlave 명령, 현재 명령 Buffer 수량을 확인합니다
_ECAT_Slave_Motion_Get_Torque	MotionSlave 명령, 현재의 모터 피드백 토크 값을 확인합니다
_ECAT_Slave_Motion_Set_Alm _Reaction	MotionSlave 명령, Alm 작동 시 반응 모드를 설정합니다.
_ECAT_Slave_Motion_Get_Actual _Command	SlaveMotion 명령, Virtual Command 설정에 관계 없이 현재 Command 데이터를 확인합니다.
_ECAT_Slave_Motion_Get_Actual _Position	SlaveMotion 명령, Virtual Command 설정에 관계 없이 현재 Position 데이터를 확인합니다.

Motion Slave-CSP API	
_ECAT_Slave_CSP_Start_Move	MotionSlaveCSP 명령, 단축 직선 동작
_ECAT_Slave_CSP_Start_V_Move	MotionSlaveCSP 명령, 단축 정속 지속 동작
_ECAT_Slave_CSP_Start_Arc_Move	MotionSlaveCSP 명령, 2 축 원형 동작 (원 중심+각도)
_ECAT_Slave_CSP_Start_Arc2_Move	MotionSlaveCSP 명령, 2 축 원형 동작 (최종좌표+각도)
_ECAT_Slave_CSP_Start_Arc3_Move	MotionSlaveCSP 명령, 2 축 원형 동작 (원 중심+최종좌표)
_ECAT_Slave_CSP_Start_Spiral_Move	MotionSlaveCSP 명령, 2 축 나선 동작 (원 중심+각도)
_ECAT_Slave_CSP_Start_Spiral2_Move	MotionSlaveCSP 명령, 2 축 나선 동작 (최종좌표+회전수)
_ECAT_Slave_CSP_Start_Sphere_Move	MotionSlaveCSP 명령, 3 축 구형 동작 (세 점이 구형을 생성)
_ECAT_Slave_CSP_Start_Heli_Move	MotionSlaveCSP 명령, 3 축 나선 동작
_ECAT_Slave_CSP_Start_Multiaxes _Move	MotionSlaveCSP 명령, 다축 직선 동작
_ECAT_Slave_CSP_Start_MSBRline _Move	MotionSlaveCSP 명령, 다축 3 점 고정 위치 평활 동작

4

Motion Slave-CSP API	
_ECAT_Slave_CSP_Set_Gear	MotionSlaveCSP 명령, 전자 기어비 설정
_ECAT_Slave_CSP_Set_Softlimit	MotionSlaveCSP 명령, 소프트웨어 limit 설정
_ECAT_Slave_CSP_TargetPos_Change	MotionSlaveCSP 명령, 새로운 목표 위치 설정
_ECAT_Slave_CSP_Velocity_Change	MotionSlaveCSP 명령, 새로운 목표 속도 설정
_ECAT_Slave_CSP_Feedrate_Overwrite	MotionSlaveCSP 명령, 다기능 변속 함수
_ECAT_Slave_CSP_Speed_Continue _Enable	MotionSlaveCSP 명령, 속도 연속 기능 활성화 / 활성화 해제
_ECAT_Slave_CSP_Speed_Continue _Set_Mode	MotionSlaveCSP 명령, 속도 연속 모드 설정
_ECAT_Slave_CSP_Speed_Continue _Set_Combine_Ratio	MotionSlaveCSP 명령, 연속된 속도 명령을 백분율로 합성하여 설정
_ECAT_Slave_CSP_Scurve_Rate	MotionSlaveCSP 명령, Scurve 의 T-curve 비율 설정
_ECAT_Slave_CSP_Linear_Speed_Master	MotionSlaveCSP 명령, 최대 속도의 정의 모드를 설정합니다
_ECAT_Slave_CSP_Mask_Axis	SlaveMotionCSP 명령, 다축 또는 단축 명령에서, 일부 몇 개의 축을 정지시켜도 다른 축에 영향을 미치지 않습니다.
_ECAT_Slave_CSP_Sync_Config	SlaveMotionCSP 명령, 동시 작동이 필요한 축을 설정합니다.
_ECAT_Slave_CSP_Sync_Move	SlaveMotionCSP 명령, Config 에서 설정한 축을 동시 작동합니다.
_ECAT_Slave_CSP_Start_Mabrline_Move	SlaveMotionCSP 명령, 다축 3 점 S-Curve 평활 보간 동작.
_ECAT_Slave_CSP_Start_2Segment _Move	SlaveMotionCSP 명령, 단축의 2 단 직선 동작을 지속합니다.
_ECAT_Slave_CSP_Start_PVT_Move	SlaveMotionCSP 명령, 단축 PVT 동작을 실행합니다.
_ECAT_Slave_CSP_Start_PVTComplete _Move	SlaveMotionCSP 명령, 단축 PVT 동작을 실행합니다.
_ECAT_Slave_CSP_Virtual_Set_Enable	SlaveMotionCSP 명령, 가상 위치를 설정합니다.
_ECAT_Slave_CSP_Virtual_Set _Command	SlaveMotionCSP 명령, 가상 위치를 설정하고, 현재 위치를 지정 위치로 설정합니다.
_ECAT_Slave_CSP_Get_SoftLimit_Status	SlaveMotionCSP 명령, 현재 소프트웨어 limit 상태를 확인합니다.
_ECAT_Slave_CSP_Pitch_Set_Interval	SlaveMotionCSP 명령, 구간 보상의 구간 거리를 설정합니다.

Motion Slave-CSP API	
_ECAT_Slave_CSP_Pitch_Set_Mode	SlaveMotionCSP 명령, 구간 보상의 보상 모드를 설정합니다.
_ECAT_Slave_CSP_Pitch_Set_Org	SlaveMotionCSP 명령, 구간 보상의 원점 위치를 설정합니다.
_ECAT_Slave_CSP_Pitch_Set_Rel_Table	SlaveMotionCSP 명령, 구간 보상 내부 각 구간의 상대 보상값을 설정합니다.
_ECAT_Slave_CSP_Pitch_Set_Abs_Table	SlaveMotionCSP 명령, 구간 보상 내부 각 구간의 절대 보상값을 설정합니다.
_ECAT_Slave_CSP_Pitch_Set_Enable	SlaveMotionCSP 명령, 구간 보상 기능을 사용합니다.
_ECAT_Slave_CSP_Start_ECAM_Set_Parameters	SlaveMotionCSP 명령, ECAM 관련 파라미터를 설정합니다.
_ECAT_Slave_CSP_Start_ECAM_Set_DisEngage_and_SingleMove	SlaveMotionCSP 명령, ECAM 단축 동작을 설정합니다.
_ECAT_Slave_CSP_Start_ECAM_Disable	SlaveMotionCSP 명령, CAM 을 종료합니다.
_ECAT_Slave_CSP_Start_ECAM_Get_Status	SlaveMotionCSP 명령, CAM 의 현재 상태를 확인합니다.
_ECAT_Slave_CSP_Start_ECAM_Set_MasterSource	SlaveMotionCSP 명령, 주축 소스를 설정합니다.
_ECAT_Slave_CSP_Start_ECAM_Set_EngageSource	SlaveMotionCSP 명령, 작동 소스를 설정합니다.
_ECAT_Slave_CSP_Start_ECAM_Set_CompensateSource	SlaveMotionCSP 명령, ECAM 보상 소스를 설정합니다.
_ECAT_Slave_CSP_Start_ECAM_Set_Compensate_Parameters	SlaveMotionCSP 명령, ECAM 보상 파라미터를 설정합니다. CAM 작동 시 파라미터를 설정할 수 없습니다.
_ECAT_Slave_CSP_Start_ECAM_Table_Move	SlaveMotionCSP 명령, ECAM 수동으로 표 만들기 기능.
_ECAT_Slave_CSP_Start_ECAM_Velocity_Move	SlaveMotionCSP 명령, ECAM 속도 구역 표 만들기 기능.
_ECAT_Slave_CSP_Start_ECAM_Flying_Shears_Move	SlaveMotionCSP 명령, ECAM 자동 연속 절단 기능.
_ECAT_Slave_CSP_Start_ECAM_Intermittence_Print_Move	SlaveMotionCSP 명령, ECAM 인터벌 인쇄 기능.

Motion Slave-CSV API	
_ECAT_Slave_CSV_Start_Move	MotionSlaveCSV 명령, 단축 속도 제어 동작
_ECAT_Slave_CSV_Multi_Start_Move	MotionSlaveCSV 명령, 다축의 동기 속도 제어 동작

4

Motion Slave-CST API	
_ECAT_Slave_CST_Start_Move	MotionSlaveCST 명령, 단축의 고정 토크 제어 동작
_ECAT_Slave_CST_Multi_Start_Move	MotionSlaveCST 명령, 다축의 동기 고정 토크 제어 동작

MotionSlave-Home API	
_ECAT_Slave_Home_Config	MotionSlaveHome 명령, Home 모드 설정
_ECAT_Slave_Home_Move	MotionSlaveHome 명령, Home 으로 돌아가기 실행
_ECAT_Slave_Home_Status	MotionSlaveHome 명령, 현재 Home 상태 확인

Motion Slave-PP API	
_ECAT_Slave_PP_Start_Move	MotionSlavePP 명령, 단축 점대점 동작
_ECAT_Slave_PP_Advance_Config	MotionSlavePP 명령, PP 모드 동작의 고급 설정

Motion Slave-Velocity API	
_ECAT_Slave_PV_Start_Move	MotionSlaveVelocity 명령, 단축의 고정 회전속도 동작
_ECAT_Slave_PV_Advance_Config	MotionSlaveVelocity 명령, PV 모드 동작 고급 설정

Motion Slave-VL API	
_ECAT_Slave_VL_Start_Move	MotionSlaveVelocity 명령, 인버터 단축의 고정 회전속도 동작

Motion Slave-Torque API	
_ECAT_Slave_PT_Start_Move	MotionSlaveTorque 명령, 단축의 고정 토크 동작
_ECAT_Slave_PT_Advance_Config	MotionSlaveTorque 명령, PT 모드 동작의 고급 설정

Motion Slave-User API	
_ECAT_Slave_User_Motion_Control_Set_Enable_Mode	MotionSlave User Motion Control 명령, 해당 Group 의 활성화 상태 설정 Enable 전에 먼저 Motion_Control_Set_Type 을 사용해야 합니다
_ECAT_Slave_User_Motion_Control_Get_Enable_Mode	MotionSlave User Motion Control 명령, 현재 해당 Group 의 활성화 상태 확인

Motion Slave-User API	
_ECAT_Slave_User_Motion_Control_Set_Type	MotionSlave User Motion Control 명령, 해당 Group 의 Motion 모드를 설정합니다
_ECAT_Slave_User_Motion_Control_Set_Data	MotionSlave User Motion Control 명령, 해당 Group 각 축의 모드에 상응하는 데이터를 입력합니다
_ECAT_Slave_User_Motion_Control_Clear_Data	MotionSlave User Motion Control 명령, 해당 Group 각 축에 입력한 데이터를 삭제합니다
_ECAT_Slave_User_Motion_Control_Get_DataCnt	MotionSlave User Motion Control 명령, 해당 Group 에서 처리가 완료되지 않은 데이터 비트수를 확인합니다
_ECAT_Slave_User_Motion_Control_Ralm	MotionSlave User Motion Control 명령, 해당 Group 의 모든 축의 Alm 을 리셋합니다
_ECAT_Slave_User_Motion_Control_Svon	MotionSlave User Motion Control 명령, 해당 Group 의 모든 축을 Svon / off 합니다
_ECAT_Slave_User_Motion_Control_Get_Alm	MotionSlave User Motion Control 명령, 해당 축의 현재 Alm 상태를 확인합니다

DIO Slave API	
_ECAT_Slave_DIO_Get_Input_Value	DIOSlave 명령, Input 값을 확인합니다
_ECAT_Slave_DIO_Get_Output_Value	DIOSlave 명령, Output 값을 확인합니다
_ECAT_Slave_DIO_Set_Output_Value	DIOSlave 명령, Output 값을 설정합니다
_ECAT_Slave_DIO_Get_Single_Input_Value	DIOSlave 명령, 단일 Input 값을 확인합니다
_ECAT_Slave_DIO_Get_Single_Output_Value	DIOSlave 명령, 단일 Output 값을 확인합니다
_ECAT_Slave_DIO_Set_Single_Output_Value	DIOSlave 명령, 단일 Output 값을 설정합니다
_ECAT_Slave_DIO_Set_Output_Error_Mode	DIOSlave 명령, 각 출력 지점의 Error on 여부를 설정할 때 자동으로 사전 설정값으로 넘어가는 기능
_ECAT_Slave_DIO_Set_Output_Error_Value	DIOSlave 명령, 각 출력 지점의 Error 모드가 on 이고, Error 상태를 트리거할 때 사전 설정한 출력값

4

AIO Slave API	
_ECAT_Slave_AIO_Get_Input_Value	AIOSlave 명령, Input 값을 확인합니다
_ECAT_Slave_AIO_Set_Output_Value	AIOSlave 명령, Output 값을 설정합니다
_ECAT_Slave_AIO_Get_Output_Value	AIOSlave 명령, Output 값을 확인합니다

5621 Slave API	
_ECAT_Slave_R1_EC5621_Set_Output _Mode	5621Slave 명령, 출력 펄스 포맷을 설정합니다
_ECAT_Slave_R1_EC5621_Set_Input _Mode	5621Slave 명령, 수신 펄스 포맷을 설정합니다
_ECAT_Slave_R1_EC5621_Set_ORG _Inverse	5621Slave 명령, ORG 접점 모드의 역방향 여부를 설정합니다
_ECAT_Slave_R1_EC5621_Set_QZ _Inverse	5621Slave 명령, QZ 접점 모드의 역방향 여부를 설정합니다
_ECAT_Slave_R1_EC5621_Set_Home _SpMode	5621Slave 명령, Home 으로 돌아가기 시, 특수 모드 사용 여부를 설정합니다
_ECAT_Slave_R1_EC5621_Set_MEL _Inverse	5621Slave 명령, MEL 접점 모드의 역방향 여부를 설정합니다
_ECAT_Slave_R1_EC5621_Set_PEL _Inverse	5621Slave 명령, PEL 접점 모드의 역방향 여부를 설정합니다
_ECAT_Slave_R1_EC5621_Set_Svon _Inverse	5621Slave 명령, Svon 접점 모드의 역방향 여부를 설정합니다
_ECAT_Slave_R1_EC5621_Set_Home _Slow_Down	5621Slave 명령, 원점((+) (-) limit 모드는 무효)감지 후 역방향으로 이동할 경우, 1 단의 감속 시간과 정지 후 대기시간을 설정합니다. 동일한 방향으로 이동할 경우 WaitTime 만 유효합니다.
_ECAT_Slave_R1_EC5621_Get_IO_ Status	5621Slave 명령, IO 상태를 확인합니다
_ECAT_Slave_R1_EC5621_Get_Single _IO_Status	5621Slave 명령, 단일 IO 의 상태를 확인합니다

x62x Slave API	
_ECAT_Slave_R1_ECx62x_Set_Output _Mode	x62xSlave 명령, 출력 펄스 포맷을 설정합니다
_ECAT_Slave_R1_ECx62x_Set_Input _Mode	x62xSlave 명령, 수신 펄스 포맷을 설정합니다

x62x Slave API	
_ECAT_Slave_R1_ECx62x_Set_ORG_Inverse	x62xSlave 명령, ORG 접점 모드의 역방향 여부를 설정합니다
_ECAT_Slave_R1_ECx62x_Set_QZ_Inverse	x62xSlave 명령, QZ 접점 모드의 역방향 여부를 설정합니다
_ECAT_Slave_R1_ECx62x_Set_Home_SpMode	x62xSlave 명령, Home 으로 돌아가기 시, 특수 모드 사용 여부를 설정합니다
_ECAT_Slave_R1_ECx62x_Set_MEL_Inverse	x62xSlave 명령, MEL 접점 모드의 역방향 여부를 설정합니다
_ECAT_Slave_R1_ECx62x_Set_PEL_Inverse	x62xSlave 명령, PEL 접점 모드의 역방향 여부를 설정합니다
_ECAT_Slave_R1_ECx62x_Set_Svon_Inverse	x62xSlave 명령, Svon 접점 모드의 역방향 여부를 설정합니다
_ECAT_Slave_R1_ECx62x_Set_Home_Slow_Down	x62xSlave 명령, Home 에서 limit 에 도달한 감속 시간을 설정합니다
_ECAT_Slave_R1_ECx62x_Get_IO_Status	x62xSlave 명령, IO 상태를 확인합니다
_ECAT_Slave_R1_ECx62x_Get_Single_IO_Status	x62xSlave 명령, 단일 IO 의 상태를 확인합니다

Delta Servo Slave API	
_ECAT_Slave_DeltaServo_Write_Parameter	DeltaServoSlave 명령, 드라이브의 파라미터를 입력합니다.
_ECAT_Slave_DeltaServo_Read_Parameter	DeltaServoSlave 명령, 드라이브의 파라미터를 확인합니다.
_ECAT_Slave_DeltaServo_Read_Parameter_Info	DeltaServoSlave 명령, 드라이브의 파라미터 정보를 확인합니다.
_ECAT_Slave_DeltaServo_Set_Velocity_Limit	DeltaServoSlave 명령, 최대 제한 속도를 설정합니다.
_ECAT_Slave_DeltaServo_Set_Compare_Enable	DeltaServoSlave 명령, 드라이브 Compare 파라미터와 상응하는 드라이브 P5-59 관련 파라미터를 입력합니다.
_ECAT_Slave_DeltaServo_Get_Compare_Enable	DeltaServoSlave 명령, 입력한 드라이브 Compare 파라미터와 상응하는 드라이브 P5-59 관련 파라미터를 확인합니다.
_ECAT_Slave_DeltaServo_Set_Compare_Config	DeltaServoSlave 명령, 드라이브 Compare 수량과 좌표를 입력합니다.

4

8124 Slave API	
_ECAT_Slave_R1_EC8124_Set_Input _RangeMode	8124Slave 전용 명령, Input 값 범위를 설정합니다
_ECAT_Slave_R1_EC8124_Set_Input _ConvstFreq_Mode	8124Slave 전용 명령, Input 샘플링 주파수를 설정합니다
_ECAT_Slave_R1_EC8124_Set_Input _Enable	8124Slave 전용 명령, Input 의 활성화 여부를 설정합니다
_ECAT_Slave_R1_EC8124_Get_Input _RangeMode	8124Slave 전용 명령, 현재 Input 값 범위를 확인합니다
_ECAT_Slave_R1_EC8124_Set_Input _AverageMode	8124Slave 전용 명령, Input 의 평균값 횟수를 설정합니다

9144 Slave API	
_ECAT_Slave_R1_EC9144_Set_Output _RangeMode	9144Slave 전용 명령, Output 의 범위를 설정합니다
_ECAT_Slave_R1_EC9144_Set_Output _Enable	9144Slave 전용 명령, Output 의 활성화 여부를 설정합니다
_ECAT_Slave_R1_EC9144_Get_Output _ReturnCode	9144Slave 전용 명령, Output 의 반환 상태를 확인합니다

Slave Record Data API	
_ECAT_Slave_Record_Set_Type	Slave 전용 명령, 각 축의 Record 데이터 종류를 설정합니다
_ECAT_Slave_Record_Set_Enable	Slave 전용 명령, 각 축의 Record 기능 실행 여부를 설정합니다
_ECAT_Slave_Record_Get_Cnt	Slave 전용 명령, 각 축의 현재 Record 데이터 비트수를 확인합니다
_ECAT_Slave_Record_Read_Data	Slave 전용 명령, 각 축의 Record 데이터를 확인합니다
_ECAT_Slave_Record_Clear_Data	Slave 전용 명령, 현재 저장된 각 축의 Record 데이터를 삭제합니다
_ECAT_Slave_Record_Multi_Set_Enable	Slave 전용 명령, 다축의 Record 기능 실행 여부를 동시에 설정합니다
_ECAT_Slave_Record_Multi_Clear_Data	Slave 전용 명령, 현재 저장된 다축의 Record 데이터를 동시에 삭제합니다

Master 축 카드 전용 API	
_ECAT_GPIO_Set_Output	축 카드의 GPIO 출력 지점 상태를 설정합니다
_ECAT_GPIO_Get_Output	축 카드의 GPIO 출력 지점 상태를 확인합니다
_ECAT_GPIO_Get_Input	축 카드의 GPIO 입력 지점 상태를 확인합니다

Master Compare API	
_ECAT_Compare_Set_Channel_Position	해당 Channel 의 새로운 Position 계수값을 설정합니다
_ECAT_Compare_Get_Channel_Position	해당 Channel 의 현재 Position 계수값을 확인합니다
_ECAT_Compare_Set_Ipulser_Mode	펄스 인터페이스 모듈의 입력 phase mode 를 설정합니다
_ECAT_Compare_Set_Channel_Direction	해당 Channel 의 펄스 방향을 설정합니다
_ECAT_Compare_Set_Channel_Trigger_Time	Trigger 활성화의 지속시간을 설정합니다
_ECAT_Compare_Set_Channel_One_Shot	Trigger 를 1 회 활성화 상태로 설정합니다
_ECAT_Compare_Set_Channel_Source	비교 소스를 설정합니다
_ECAT_Compare_Set_Channel_Enable	Compare 기능의 On / Off 를 설정합니다
_ECAT_Compare_Channel0_Position	Compare Type0 실행
_ECAT_Compare_Set_Channel0_Trigger_By_GPIO	Compare Type0 의 트리거 조건을 GPIO 가 제어하도록 설정합니다
_ECAT_Compare_Set_Channel1_Output_Enable	Compare Type1 의 출력 On / Off 를 설정합니다
_ECAT_Compare_Set_Channel1_Output_Mode	Compare Type1 의 출력 모드를 설정합니다
_ECAT_Compare_Get_Channel1_IO_Status	Compare Type1 의 I/O 상태를 확인합니다.
_ECAT_Compare_Set_Channel1_GPIO_Out	Compare Type1 의 GPIO 출력 bit 상태(Pin15)를 설정합니다
_ECAT_Compare_Set_Channel1_Position_Table	Compare Type1 의 관련 데이터를 설정합니다
_ECAT_Compare_Set_Channel1_Position_Table_Level	Compare Type1 의 Level 관련 데이터를 설정합니다
_ECAT_Compare_Get_Channel1_Position_Table_Count	이미 실행한 Trigger 의 횟수(Compare Type1 사용)를 확인합니다
_ECAT_Compare_Set_Channel_Polarity	Compare Trigger 의 레벨을 설정합니다
_ECAT_Compare_Reuse_Channel1_Position_Table	지난 번에 설정한 Compare 조건으로 Compare Type1 을 다시 한 번 실행합니다

4

Master Compare API	
_ECAT_Compare_Reuse_Channel1_Position _Table_Level	지난 번에 설정한 Compare 조건으로 Compare Type1(Level 모드)을 다시 한 번 실행합니다

DLL 관련 API	
_ECAT_Master_Get_DLL_Path	EtherCat_DLL.dll 의 상세 경로 데이터를 확인합니다
_ECAT_Master_Get_DLL_Version	EtherCat_DLL.dll 버전 데이터를 확인합니다
_ECAT_Master_Get_DLL_Path_Single	ECAT_RTX_DLL.dll 또는 PCI_L221.dll 의 상세 경로 데이터를 확인합니다
_ECAT_Master_Get_DLL_Version_Single	ECAT_RTX_DLL.dll 또는 PCI_L221.dll 의 버전 데이터를 확인합니다

Master User Security API	
_ECAT_Security_Check_Verifykey	보안키를 확인합니다
_ECAT_Security_Get_Check_Verifykey _State	보안키의 해제 여부를 확인합니다
_ECAT_Security_Write_Verifykey	보안키를 입력합니다
_ECAT_Security_Get_Write_Verifykey _State	보안키 입력 완료 여부를 확인합니다
_ECAT_Security_Check_UserPassword	사용자 암호를 확인합니다
_ECAT_Security_Get_Check _UserPassword_State	사용자 암호의 해제 여부를 확인합니다
_ECAT_Security_Write_UserPassword	사용자 암호를 입력합니다
_ECAT_Security_Get_Write _UserPassword_State	사용자 암호 입력 완료 여부를 확인합니다

PAC MRAM 전용 API	
_ECAT_Master_MRAM_Write_Word_Data	U16 데이터를 PAC MRAM 의 지정된 위치에 입력합니다
_ECAT_Master_MRAM_Read_Word_Data	PAC MRAM 의 지정된 위치에 입력한 U16 데이터를 확인합니다
_ECAT_Master_MRAM_Write_DWord _Data	U32 데이터를 PAC MRAM 의 지정된 위치에 입력합니다
_ECAT_Master_MRAM_Read_DWord _Data	PAC MRAM 의 지정된 위치에 입력한 U32 데이터를 확인합니다

70E2 Slave API	
_ECAT_Slave_R1_EC70E2_Set_Output _Enable	70E2Slave 전용 명령, Output 의 활성화 여부를 설정합니다

70X2 Slave API	
_ECAT_Slave_R1_EC70X2_Set_Output _Enable	70X2 slave 모듈 전용 명령, Output 의 활성화 여부를 설정합니다.

5614 Slave API	
_ECAT_Slave_R1_EC5614_Set_MJ_ Config	5614Slave 전용 명령, MPG 또는 JOG 의 파라미터 내용을 설정합니다
_ECAT_Slave_R1_EC5614_Set_MJ_ Enable	5614Slave 전용 명령, MPG 또는 JOG 의 활성화 여부를 설정합니다
_ECAT_Slave_R1_EC5614_Get_IO_ Status	5614Slave 전용 명령, 모듈의 IO 상태를 확인합니다
_ECAT_Slave_R1_EC5614_Get_MPG _Counter	5614Slave 전용 명령, 모듈의 MPG Counter 를 확인합니다

(이 페이지는 공란으로 비워둡니다)

Master Config API

5

다음은 Master Cycle Time, 설정, 사용자 정의 Node 번호 설정, 모듈 구성 및 DC 데이터 확인 및 입력, 각 Node 의 DCTime 설정 등 Master Config 관련 명령 API 사용법에 대한 설명입니다.

5

Master Config API 테이블

함수의 명칭	설명
_ECAT_Master_Set_CycleTime	Master 명령, 사이클 타임을 설정합니다. ※Initial 앞에서 설정해야 합니다.
_ECAT_Master_Get_CycleTime	Master 명령, 해당 축 카드의 사이클 타임을 조회합니다.
_ECAT_Master_NodeID_Alias_Enable	Master 명령, 사용자 정의 Node 번호의 사용 여부를 설정합니다. ※Initial 앞에서 설정해야 합니다.
_ECAT_Master_Get_SerialNo	PAC 또는 축 카드의 시퀀스 번호를 확인합니다.
_ECAT_Master_Get_DLL_SeqID	Master 명령, 현재 DLL 이 사용 중인 시퀀스 ID 를 확인합니다.
_ECAT_Autoconfig_Open_File	Master 명령, 파일에서 모듈 구성 및 DC 데이터를 확인합니다. ※ Initial 앞에서 설정해야 합니다.
_ECAT_Autoconfig_Save_File	Master 명령, 현재의 모듈 구성과 DC 데이터를 파일에 입력합니다.
_ECAT_Autoconfig_Set_Slave_DCTime	Master 명령, 각 Node 의 DC 시간 데이터를 설정합니다.
_EACT_Autoconfig_Clear_ConfigFile	Master 명령, 현재의 로딩 설정을 삭제합니다.
_ECAT_Autoconfig_Set_NodeID_Alias	Master 명령, 각 Node 의 사용자 정의 Node 번호를 설정합니다. ※Initial 완료 후 사용이 가능합니다.
_ECAT_Autoconfig_Get_NodeID_Alias	Master 명령, 각 Node 의 사용자 정의 Node 번호를 확인합니다. ※Initial 완료 후 사용이 가능합니다.
_ECAT_Autoconfig_Save_NodeID_Alias	Master 명령, 모듈의 메모리 블록에 사용자 정의 Node 번호를 실제로 입력합니다.

5.1 _ECAT_Master_Set_CycleTime

■ **포맷**

U16 PASCAL _ECAT_Master_Set_CycleTime (U16 CardNo, U16 Mode)

■ **목적**

Master 명령, 사이클 타임을 설정합니다. ※ Initial 앞에서 설정해야 합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
Mode	U16	번호 단위	통신 주기 시간 (us) 0 : 2000 us 1 : 1000 us 2 : 500 us 3 : 250 us 4 : 125 us

■ **예제**

```
U16 Status;
U16 CardNo=16;
U16 Mode = 3;
// 카드 on 후 초기화 이전에 실행
Status = _ECAT_Master_Set_CycleTime (CardNo, Mode);
```

5.2 _ECAT_Master_Get_CycleTime

5

■ **포맷**

U16 PASCAL _ECAT_Master_Get_CycleTime (U16 CardNo, U16 *CycleTime)

■ **목적**

Master 명령, 해당 축 카드의 사이클 타임을 조회합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
CycleTime	U16*	시간 단위-us	통신 주기 시간 (us) (※ 현재 Master 는 1000, 500, 250, 125 등 4 종류의 사이클 타임만 지원합니다. 각 Slave 부분에 대한 설명은 Slave 자체의 내용을 참고하시기 바랍니다.)

■ **예제**

```
U16 Status;
U16 CardNo=16;
U16 CycleTime =0;
// Master가 현재 설정한 CycleTime을 확인합니다
Status = _ECAT_Master_Get_CycleTime (CardNo, &CycleTime);
```

5.3 _ECAT_Master_NodeID_Alias_Enable

■ **포맷**

U16 PASCAL _ECAT_Master_NodeID_Alias_Enable (U16 CardNo, U16 Enable)

■ **목적**

Master 명령, 사용자 정의 Node 번호의 사용 여부를 설정합니다.

※ Initial 앞에서 설정해야 합니다. **Node 번호를 설정하지 않은 Slave 가 있을 경우, 오류 메시지가 나타납니다(0x1004)**.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
Enable	U16	옵션 단위	Enable 설명 0: 사용자 정의 Node 번호 모드 사용 off 1: 사용자 정의 Node 번호 모드 사용 on

■ **예제**

```
U16 Status;
U16 CardNo=16, Enable =1;
// 카드 on 후 초기화 이전에 실행
Status = _ECAT_Master_NodeID_Alias_Enable (CardNo, Enable);
```

5.4 _ECAT_Get_SerialNo

■ 포맷

U16 PASCAL _ECAT_Get_SerialNo (U16 CardNo, U32* SerialNo)

■ **목적**

PAC 또는 축 카드의 시퀀스 번호를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
SerialNo	U32	번호 단위	시퀀스

■ **예제**

```
U16 Status;
U16 CardNo=16;
U32 SerialNo=0;

Status = _ECAT_Get_SerialNo (CardNo, &SerialNo);
```

5.5 _ECAT_Master_Get_DLL_SeqID

■ **포맷**

U16 PASCAL _ECAT_Master_Get_DLL_SeqID (U16 CardNo, U16 *SeqID)

■ **목적**

Master 명령, 현재 DLL 이 사용 중인 시퀀스 ID 를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
SeqID	U16*	번호 단위	현재 DLL 이 사용한 시퀀스 ID

■ **예제**

```
U16 Status;
U16 CardNo = 16;
U16 SeqID = 0;

Status = _ECAT_Master_Get_DLL_SeqID (CardNo, &SeqID);
```

5.6 _ECAT_Autoconfig_Open_File

5

■ **포맷**

U16 PASCAL _ECAT_Autoconfig_Open_File (U16 CardNo, I8 *FilePath)

■ **목적**

Master 명령, 파일에서 모듈 구성 및 DC 데이터를 확인합니다.
※Initial 앞에서 설정해야 합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
FilePath	I8*	번호 단위	이미 존재하는 Config 데이터 파일

■ **예제**

```
U16 Status;
U16 CardNo=16;
I8 FilePath[255];
strcpy(FilePath, "C:\\EtherCAT_Information.dat");

Status = _ECAT_Autoconfig_Open_File (CardNo, FilePath);
```

5.7 _ECAT_Autoconfig_Save_File

■ **포맷**

U16 PASCAL _ECAT_Autoconfig_Save_File (U16 CardNo, I8 *FilePath)

■ **목적**

Master 명령, 현재의 모듈 구성과 DC 데이터를 파일에 입력합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
FilePath	I8*	번호 단위	현재 모듈 구성 및 DC 데이터를 데이터 파일에 저장합니다

■ **예제**

```
U16 Status;
U16 CardNo=16;
I8 FilePath[255];
strcpy(FilePath, "C:\\EtherCAT_Information.dat");

Status = _ECAT_Autoconfig_Save_File (CardNo, FilePath);
```

5.8 _ECAT_Autoconfig_Set_Slave_DCTime

5

■ **포맷**

U16 PASCAL _ECAT_Autoconfig_Set_Slave_DCTime (U16 CardNo, U16 NodeID, U16 Mode)

■ **목적**

Master 명령, 각 Node 의 DC 시간 데이터를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
Mode	U16	번호 단위	각 Node 의 DC 시간 데이터를 설정합니다 Mode 0 : 2000 us 1 : 1000 us 2 : 500 us 3 : 250 us 4 : 125 us

■ **예제**

```
U16 Status;
U16 CardNo=16,NodeID=1;
U16 Mode=1;

Status = _ECAT_Autoconfig_Set_Slave_DCTime (CardNo, NodeID, Mode);
```

5.9 _EACT_Autoconfig_Clear_ConfigFile

■ **포맷**

U16 PASCAL _EACT_Autoconfig_Clear_ConfigFile (U16 CardNo)

■ **목적**

Master 명령, 현재 _ECAT_Autoconfig_Open_File 을 통해 로드한 설정 파일을 삭제합니다. 설정 파일을 잘못 열거나 장치에 원래의 설정에 적합하지 않은 변경 사항이 있을 경우에 주로 활용됩니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호

■ **예제**

```
U16 Status;
U16 CardNo=16;

Status = _EACT_Autoconfig_Clear_ConfigFile (CardNo);
```

5.10 _ECAT_Autoconfig_Set_NodeID_Alias

5

■ **포맷**

U16 PASCAL _ECAT_Autoconfig_Set_NodeID_Alias (U16 CardNo, U16 NodeID, U16 MapNodeID)

■ **목적**

Master 명령, 각 Node 의 사용자 정의 Node 번호를 설정합니다.
※Initial 완료 후 사용이 가능합니다

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
MapNodeID	U16	번호 단위	NodeID 번호를 지정합니다

■ **예제**

```
U16 Status;
U16 CardNo=16,NodeID=1;
U16 MapNodeID;

Status = _ECAT_Autoconfig_Set_NodeID_Alias (CardNo, NodeID, MapNodeID)
```

5.11 _ECAT_Autoconfig_Get_NodeID_Alias

■ **포맷**

U16 PASCAL _ECAT_Autoconfig_Get_NodeID_Alias (U16 CardNo, U16 RealNodeID, U16 *MapNodeID)

■ **목적**

Master 명령, 각 Node 의 사용자 정의 Node 번호를 확인합니다.
※Initial 완료 후 사용이 가능합니다

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
RealNodeID	U16	번호 단위	NodeID 번호
MapNodeID	U16*	번호 단위	NodeID 번호를 지정합니다

■ **예제**

```
U16 Status;
U16 CardNo=16,NodeID=1;
U16 MapNodeID;

Status = _ECAT_Autoconfig_Get_NodeID_Alias (CardNo, NodeID, &MapNodeID);
```

5.12 _ECAT_Autoconfig_Save_NodeID_Alias

5

■ **포맷**

U16 PASCAL _ECAT_Autoconfig_Save_NodeID_Alias (U16 CardNo)

■ **목적**

Master 명령, 모듈의 메모리 블록에 사용자 정의 Node 번호를 실제로 입력합니다.
비고: 해당 API 가 전송되면 자동으로 disconnect 되고, 필수 데이터를 모듈 메모리에 입력합니다. 야스카와와 델타 서보 구동은 R1-EC 시리즈 모듈에 대해 각각 Knob 또는 파라미터 P3-00 를 사용하여 Node 번호를 조정할 수 있습니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호

■ **예제**

```
U16 Status;
U16 CardNo=16, Enable =1;

Status = _ECAT_Autoconfig_Save_NodeID_Alias (CardNo)
```

Master Initial API

6

다음은 카드 on / off 및 초기화, Master 리셋, 카드 번호 및 Slave 개수 확인, 각 Slave 데이터 확인, DC 상태 확인, 축 카드 연결 상태 및 연결 상태 이상 유무 확인, 각 Node 에서 처리되지 않은 명령 확인, Cycle 소요 시간 확인, Initial 완료 여부 확인 등 Master Initial 관련 명령 API 의 사용법에 대한 설명입니다.

6

Master Initial API 테이블

함수의 명칭	설명
_ECAT_Master_Open	Master 명령, CreateFile 을 사용하여 축 카드의 장수를 검출하고, 메모리 영역을 생성합니다.
_ECAT_Master_Initial	Master 명령, AutoConfig 를 실행하고, OP 모드로 전환하며, DC 등의 동작을 대기합니다.
_ECAT_Master_Reset	Master 명령, 상태를 리셋하고, Init 모드로 변경합니다.
_ECAT_Master_Close	Master 명령, 축 카드를 off 하고, 메모리를 해제합니다.
_ECAT_Master_Get_CardSeq	Master 명령, 해당 축 카드의 카드 번호를 조회합니다.
_ECAT_Master_Get_SlaveNum	Master 명령, 해당 축 카드의 Slave 개수를 조회합니다.
_ECAT_Master_Get_Slave_Info	Master 명령, 해당 Node 번호의 Information 을 조회합니다.
_ECAT_Master_Get_DC_Status	Master 명령, 해당 축 카드의 DC 상태, 시간, 오차 시간을 조회합니다.
_ECAT_Master_Get_Connect_Status	Master 명령, 해당 축 카드의 연결 상태를 조회합니다.
_ECAT_Master_Get_Api_BufferLength	Master 명령, 현재 시스템에서 처리가 완료되지 않은 각 Node 의 명령들을 확인합니다.
_ECAT_Master_Get_Cycle_SpendTime	Master 명령, 매회 Cycle 의 Tx 와 Rx 의 소요시간과 이전의 최대 소요시간을 확인합니다.
_ECAT_Master_Check_Initial_Done	Master 명령, DLL 레이어의 Initial 완료 여부를 조회합니다.
_ECAT_Master_Get_Initial_ErrorCode	Master 명령, _ECAT_Master_Check_Initial_Done 을 통해 확인한 데이터에 오류가 발생하면 해당 API 를 통해 오류 코드를 확인합니다.
_ECAT_Master_Check_Working_Counter	Master 명령, 현재의 연결 상태에 이상이 없는지 확인합니다.
_ECAT_Master_Get_Return_Code_Message	Master 명령, 반환 코드에 상응하는 메시지를 확인합니다.

6.1 _ECAT_Master_Open

■ **포맷**

U16 PASCAL _ECAT_Master_Open (U16 *Cardnum)

■ **목적**

Master 명령, 축 카드 장수와 EtherCat 코어 수량을 검출하고, 메모리 영역을 생성합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호

■ **예제**

```
U16 Status;
U16 Cardnum=0;

Status = _ECAT_Master_Open(&Cardnum);
```

6.2 _ECAT_Master_Initial

■ **포맷**

U16 PASCAL _ECAT_Master_Initial (U16 CardNo)

■ **목적**

Master 명령, AutoConfig 를 실행하고, 모듈을 OP 모드로 전환합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호

■ **예제**

```
U16 Status;
U16 CardNo=16;

Status = _ECAT_Master_Initial(CardNo);
```

6.3 _ECAT_Master_Reset

6

■ **포맷**

U16 PASCAL _ECAT_Master_Reset (U16 CardNo)

■ **목적**

Master 명령, Master 상태를 리셋하고, 모듈을 Initial 모드로 전환합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호

■ **예제**

```
U16 Status;
U16 CardNo=16;

Status = _ECAT_Master_Reset(CardNo);
```

6.4 _ECAT_Master_Close

■ **포맷**

U16 PASCAL _ECAT_Master_Close()

■ **목적**

Master 명령, 모든 축 카드와 코어를 off 하고, 메모리를 해제합니다.

■ **예제**

```
U16 Status;

Status = _ECAT_Master_Close();
```

6.5 _ECAT_Master_Get_CardSeq

■ **포맷**

U16 PASCAL _ECAT_Master_Get_CardSeq (U16 CardSeq, U16 * CardNo)

■ **목적**

Master 명령, 해당 축 카드의 카드 번호를 조회합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardSeq	U16	번호 단위	축 카드 시리얼 넘버
CardNo	U16*	번호 단위	카드 번호

■ **예제**

```
U16 Status;
U16 CardSeq=0,   CardNo ;

Status = _ECAT_Master_Get_CardSeq (CardSeq, &CardNo);
```

6.6 _ECAT_Master_Get_SlaveNum

■ **포맷**

U16 PASCAL _ECAT_Master_Get_SlaveNum (U16 CardNo, U16 *Slavenum)

■ **목적**

Master 명령, 해당 축 카드의 Slave 개수를 조회합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
Slavenum	U16*	수치 단위	해당 카드 번호가 연결된 Slave 수량

■ **예제**

```
U16 Status;
U16 CardNo=16, Slavenum=0;

Status = _ECAT_Master_Get_SlaveNum(CardNo, &Slavenum);
```

6.7 _ECAT_Master_Get_Slave_Info

■ **포맷**

U16 PASCAL _ECAT_Master_Get_Slave_Info (U16 CardNo, U16 SeqID, U16 *NodeID, U32 *VenderID, U32 *ProductCode, U32 *RevisionNo, U32 *DCTime)

■ **목적**

Master 명령, 해당 Node 번호의 Information 을 조회합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
SeqID	U16	번호 단위	모듈의 물리적 시퀀스 ID
NodeID	U16*	번호 단위	모듈에 상응하는 Node 번호
VenderID	U32*	번호 단위	제조사 코드
ProductCode	U32*	번호 단위	제품 코드
RevisionNo	U32*	번호 단위	버전 식별 코드
DCTime	U32*	수치 단위	모듈 DC 시간

■ **예제**

```
U16 Status;
U16 CardNo=16, SeqID =2, NodeID =2;
U32 VenderID, ProductCode, RevisionNo, DCTime;

Status = _ECAT_Master_Get_Slave_Info(CardNo, SeqID, NodeID , &VenderID,
&ProductCode, &RevisionNo, & DCTime);
```

■ 비고:

EtherCAT 주국(primary station)은 제한적으로 모듈을 지원합니다. 사용하는 슬레이브(secondary station) 모듈(델타 제품 아님)이 주국에 거부되었을 경우, 문제 해결을 위해 우선 해당 슬레이브 모듈의 VenderID, ProductCode, RevisionNo 를 확인하고, 델타에 문의해주시기 바랍니다. 상기 데이터 확인 방법: Edit 소프트웨어를 사용하여 해당 슬레이브 모듈에 상응하는 XML 파일 열기

VenderID : 1A05 or 1DD

ProductCode : 00005500

RevisionNo : 00100000

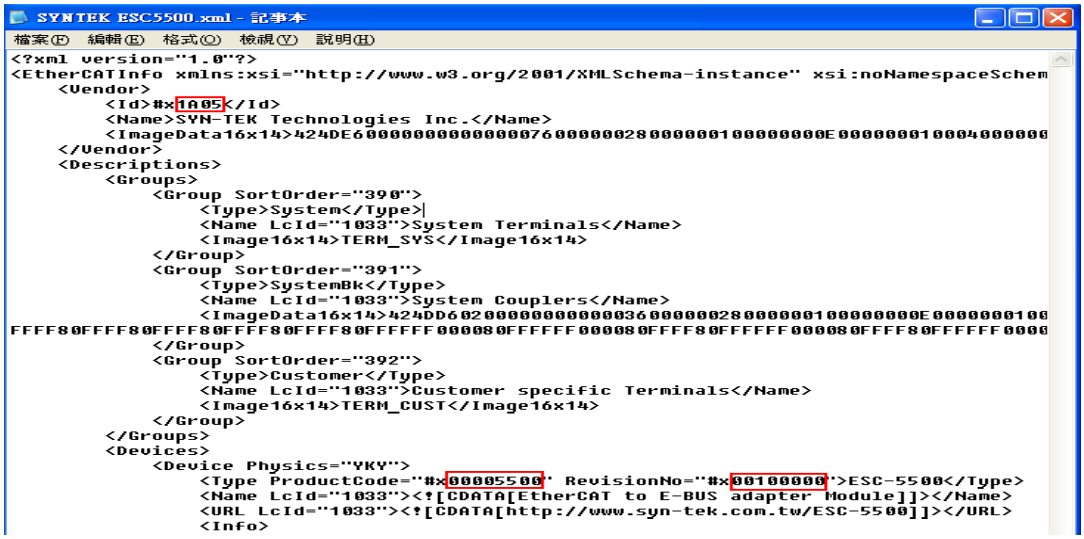


그림 6.7.1 XML 파일 내용

6.8 _ECAT_Master_Get_DC_Status

6

■ 포맷

U16 PASCAL _ECAT_Master_Get_DC_Status (U16 CardNo, U32 *State, I32 *Time, I32 *OffsetTime)

■ 목적

Master 명령, 해당 축 카드의 DC 상태, 시간, 오차 시간을 조회합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
State	U32*	번호 단위	DC 의 현재 상태. 0: 교정 중 1: 초기 교정 완료
Time	I32*	수치 단위	현재 DC 교정 시간 (정상 상태일 경우 대략 Cycle Time 의 절반)
OffsetTime	I32*	수치 단위	DC 교정 follow time

■ 예제

```
U16 Status;
U16 CardNo=16;
U32 State;
I32 Time, OffsetTime;

Status = _ECAT_Master_Get_DC_Status(CardNo, &State, &Time, &OffsetTime);
```

6.9 _ECAT_Master_Get_Connect_Status

■ **포맷**

U16 PASCAL _ECAT_Master_Get_Connect_Status (U16 CardNo, U16 * MasterStatus)

■ **목적**

Master 명령, 해당 축 카드의 연결 상태를 조회합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
MasterStatus	U16*	번호 단위	현재 Master 상태 1: Init 모드 2: Pre-OP 모드 4: Safe-OP 모드 8: OP 모드

■ **예제**

```
U16 Status;
U16 CardNo=16;
U16 MasterStatus=0;

Status = _ECAT_Master_Get_Connect_Status(CardNo, &MasterStatus);
```

6.10 _ECAT_Master_Get_Api_BufferLength

6

■ **포맷**

U16 PASCAL _ECAT_Master_Get_Api_BufferLength (U16 CardNo, U16 SlaveNo, U16 *BuffLength)

■ **목적**

Master 명령, 현재 시스템에서 처리가 완료되지 않은 각 Node 의 명령들을 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
SlaveNo	U16	번호 단위	Node 번호
BuffLength	U16*	수치 단위	현재 API 명령 Buffer 의 누적 길이

■ **예제**

```
U16 Status;
U16 CardNo=16, SlaveNo=1, BuffLength;

Status = _ECAT_Master_Get_Api_BufferLength(CardNo, SlaveNo , &BuffLength);
```

6.11 _ECAT_Master_Get_Cycle_SpendTime

■ **포맷**

U16 PASCAL _ECAT_Master_Get_Cycle_SpendTime (U16 CardNo, F64 *Tx_Time, F64 *Tx_MaxTime, F64 *Rx_Time, F64 *Rx_MaxTime)

■ **목적**

Master 명령, 매회 Cycle 의 Tx 와 Rx 의 소요시간과 이전의 최대 소요시간을 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
Tx_Time	F64*	수치 단위	Tx 소요 시간(us) 확인
Tx_MaxTime	F64*	수치 단위	이전 Tx 의 최대 소요 시간(us) 확인
Rx_Time	F64*	수치 단위	Rx 소요 시간(us) 확인
Rx_MaxTime	F64*	수치 단위	이전 Rx 의 최대 소요 시간(us) 확인

■ **예제**

```
U16 Status;
U16 CardNo=16;
F64 Tx_Time, Tx_MaxTime, Rx_Time, Rx_MaxTime;

Status = _ECAT_Master_Get_Cycle_SpendTime(CardNo, &Tx_Time, &Tx_MaxTime,
&Rx_Time, &Rx_MaxTime);
```

6.12 _ECAT_Master_Check_Initial_Done

■ **포맷**

U16 PASCAL _ECAT_Master_Check_Initial_Done (U16 CardNo, U16 *InitDone)

■ **목적**

Master 명령, DLL 레이어의 Initial 완료 여부를 조회합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
InitDone	U16*	번호 단위	0: 완료 1: 처리 중 99: 오류 발생

■ **예제**

```
U16 Status;
U16 CardNo=16, InitDone;

Status = _ECAT_Master_Check_Initial_Done(CardNo, &InitDone);
```

6.13 _ECAT_Master_Get_Initial_ErrorCode

■ **포맷**

U16 PASCAL _ECAT_Master_Get_Initial_ErrorCode (U16 CardNo)

■ **목적**

Master 명령, _ECAT_Master_Check_Initial_Done 을 통해 확인한 데이터가 99(오류 발생)이면 해당 API 를 통해 오류 코드를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호

■ **예제**

```
U16 Status;
U16 CardNo=16;

Status = _ECAT_Master_Get_Initial_ErrorCode(CardNo);
```

6.14 _ECAT_Master_Check_Working_Counter

6

■ **포맷**

U16 PASCAL _ECAT_Master_Check_Working_Counter (U16 CardNo, U16 *Abnormal_Flag, U16 *Working_Slave_Cnt)

■ **목적**

Master 명령, 현재의 연결 상태에 이상이 없는지 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
Abnormal_Flag	U16*	옵션 단위	0: 정상 1: 비정상
Working_Slave_Cnt	U16*	수치 단위	현재 통신이 검출한 Slave 수량. 정상적인 상황에서는 _ECAT_Master_Get_SlaveNum 에서 도출한 값과 일치하며, 비정상적인 상황에서는 해당 값에 근거하여 실제 회로의 어느 부분에서 오류가 발생했는지 판단할 수 있습니다.

■ **예제**

```
U16 Status;
U16 CardNo=16;
U16 Abnormal_Flag, Working_Slave_Cnt;

Status = _ECAT_Master_Check_Working_Counter(CardNo, &Abnormal_Flag,
&Working_Slave_Cnt);
```

6.15 _ECAT_Master_Get_Return_Code_Message

■ **포맷**

U16 PASCAL _ECAT_Master_Get_Return_Code_Message (U16 ReturnCode, I8 *Message);

■ **목적**

Master 명령, 반환 코드에 상응하는 메시지를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
ReturnCode	U16	번호 단위	반환 코드
Message	I8*	옵션 단위	반환 코드에 상응하는 메시지를 확인합니다

■ **예제**

```
U16 CardNo=16;
U16 Rt, ReturnCode;
I8 Message[500]= {0};
ReturnCode = _ECAT_Master_Get_Initial_ErrorCode(CardNo);
Rt = _ECAT_Master_Get_Return_Code_Message(ReturnCode, Message);
```

(이 페이지는 공란으로 비워둡니다)

Slave 공통 API

7

다음은 SDO 명령 전송, SDO 데이터 확인, SDO 오류 코드 확인, SDO 명령 완료 여부 검사, PDO 의 OD 데이터 확인, PDO 의 OD 데이터 설정, PDO 구성 정보 확인, PDO Tx 와 Rx 의 데이터 확인 및 입력 등 Slave 공통 관련 명령 API 의 사용법에 대한 설명입니다.

7

Slave 공통 API 테이블

함수의 명칭	설명
_ECAT_Slave_SDO_Send_Message	Slave 공통 명령, 해당 Node 에 SDO (CANopen 포맷) 명령을 전송합니다.
_ECAT_Slave_SDO_Read_Message	Slave 공통 명령, 이 Node 의 현재 SDO (CANopen 포맷) 데이터를 확인합니다.
_ECAT_Slave_SDO_Quick_Send_Message	Slave 공통 명령, 해당 Node 에 SDO (CANopen 포맷)명령을 전송하나 명령 전송 완료를 기다리지 않습니다.
_ECAT_Slave_SDO_Quick_Read_Message	Slave 공통 명령, 이 Node 에 SDO (CANopen 포맷) 확인 명령을 하달하지만 명령의 하달 완료를 기다리지 않습니다.
_ECAT_Slave_SDO_Read_Response	Slave 공통 명령, 이 Node 에 대한 반환 데이터를 확인합니다.
_ECAT_Slave_SDO_Wait_All_Done	Slave 공통 명령, 복수 Node 에 현재 누적된 SDO 명령이 전부 완료될 때까지 대기하라는 명령을 전송합니다.
_ECAT_Slave_SDO_Get_ErrorCode	Slave 공통 명령, Send_Message 또는 Read_Message 에 ERR_ECAT_SDO_Return_Error 오류가 발생할 때, 해당 API 를 사용하여 오류 코드를 확인할 수 있습니다. 오류 코드는 CANopen 또는 모듈 개체 정의에서 참조할 수 있습니다.
_ECAT_Slave_SDO_Check_Done	Slave 공통 명령, 단일 Node 에 현재 누적된 SDO 명령이 완료되었는지 확인합니다.
_ECAT_Slave_PDO_Get_OD_Data	Slave 공통 명령, 해당 Node 에 임의의 OD 코드 데이터를 확인하며, 해당 데이터는 PDO 구성에 매핑되어야 합니다.
_ECAT_Slave_PDO_Set_OD_Data	Slave 공통 명령, 해당 Node 에 임의의 OD 코드 데이터를 전송하며, 해당 데이터는 PDO 구성에 매핑되어야 합니다.
_ECAT_Slave_PDO_Get_Information	Slave 공통 명령, 각 Node PDO 구성의 기본 데이터를 확인합니다.
_ECAT_Slave_PDO_Get_Detail_Mapping	Slave 공통 명령, 각 Node PDO 구성의 상세 매핑 데이터를 확인합니다.
_ECAT_Slave_PDO_Get_Rx_Data	Slave 공통 명령, 모든 Node 의 PDO Rx 구성 데이터를 확인합니다.
_ECAT_Slave_PDO_Get_Tx_Data	Slave 공통 명령, 모든 Node 의 PDO Rx 구성 데이터를 확인합니다.

함수의 명칭	설명
_ECAT_Slave_PDO_Set_Tx_Detail_Data	Slave 공통 명령, 각 Node 의 PDO Tx 상세 매핑 데이터를 설정합니다.
_ECAT_Slave_PDO_Set_Tx_Data	Slave 공통 명령, 모든 Node 의 PDO Tx 구성 데이터를 입력합니다.

7.1 _ECAT_Slave_SDO_Send_Message

■ **포맷**

U16 PASCAL _ECAT_Slave_SDO_Send_Message (U16 CardNo, U16 NodeID, U16 SlotNo, U16 Index, U16 SubIndex, U16 DataSize, U8 *Data)

■ **목적**

Slave 공통 명령, 해당 Node 에 SDO (CANopen 포맷) 명령을 전송합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Index	U16	수치 단위	CANopen Object Dictionary 의 index
SubIndex	U16	수치 단위	CANopen Object Dictionary 의 subindex
DataSize	U16	수치 단위	메시지를 발송한 데이터량의 크기, 단위는 Byte
Data	U8*	수치 단위	메시지를 발송한 데이터

■ **예제**

```
U16 Status;
U16 CardNo=16,NodeID=1,SlotNo=0;
U16 Index=0x6040, SubIndex=0, DataSize=4;
U8 Data[4]={0};

Status = _ECAT_Slave_SDO_Send_Message(CardNo, NodeID, SlotNo,
Index, SubIndex, DataSize, Data);
```

7.2 _ECAT_Slave_SDO_Read_Message

■ 포맷

U16 PASCAL _ECAT_Slave_SDO_Read_Message (U16 CardNo, U16 NodeID, U16 SlotNo, U16 Index, U16 SubIndex, U16 DataSize, U8 *Data)

■ 목적

Slave 공통 명령, 이 Node 의 현재 SDO (CANopen 포맷) 데이터를 확인합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Index	U16	수치 단위	CANopen Object Dictionary 의 index
SubIndex	U16	수치 단위	CANopen Object Dictionary 의 subindex
DataSize	U16	수치 단위	메시지를 수집한 데이터량의 크기, 단위는 Byte
Data	U8*	수치 단위	메시지를 수집한 데이터

■ 예제

```
U16 Status;
U16 CardNo=16,NodeID=1,SlotNo=0;
U16 Index=0x1000, SubIndex=0, DataSize=4;
U8 Data[4] = {0};

Status =_ECAT_Slave_SDO_Read_Message(CardNo, NodeID, SlotNo, Index,
SubIndex, DataSize, &Data[0]);
```

7.3 _ECAT_Slave_SDO_Quick_Send_Message

■ **포맷**

U16 PASCAL _ECAT_Slave_SDO_Quick_Send_Message (U16 CardNo, U16 NodeID, U16 SlotNo, U16 Index, U16 SubIndex, U16 DataSize, U8 *Data)

■ **목적**

Slave 공통 명령, 해당 Node 에 SDO (CANopen 포맷)명령을 전송하나 명령 전송 완료를 기다리지 않습니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Index	U16	수치 단위	CANopen Object Dictionary 의 index
SubIndex	U16	수치 단위	CANopen Object Dictionary 의 subindex
DataSize	U16	수치 단위	메시지를 수집한 데이터량의 크기, 단위는 Byte
Data	U8*	수치 단위	메시지를 수집한 데이터

■ **예제**

```
U16 Status;
U16 CardNo=16,NodeID=1,SlotNo=0;
U16 Index=0x6040, SubIndex=0, DataSize=4;
U8 Data[4] = {0};

Status = _ECAT_Slave_SDO_Quick_Send_Message (CardNo, NodeID, SlotNo,
Index, SubIndex, DataSize, &Data[0]);
```

7.4 _ECAT_Slave_SDO_Quick_Read_Message

■ 포맷

U16 PASCAL _ECAT_Slave_SDO_Quick_Read_Message (U16 CardNo, U16 NodeID, U16 SlotNo, U16 Index, U16 SubIndex, U16 DataSize)

■ 목적

Slave 공통 명령, 이 Node 에 SDO (CANopen 포맷) 확인 명령을 하달하지만 명령의 하달 완료를 기다리지 않습니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Index	U16	수치 단위	CANopen Object Dictionary 의 index
SubIndex	U16	수치 단위	CANopen Object Dictionary 의 subindex
DataSize	U16	수치 단위	메시지를 수집한 데이터량의 크기, 단위는 Byte

■ 예제

```
U16 Status;
U16 CardNo=16,NodeID=1,SlotNo=0;
U16 Index=0x6040, SubIndex=0, DataSize=4;

Status = _ECAT_Slave_SDO_Quick_Read_Message (CardNo, NodeID, SlotNo,
Index, SubIndex, DataSize);
```

7.5 _ECAT_Slave_SDO_Read_Response

■ 포맷

U16 PASCAL _ECAT_Slave_SDO_Read_Response (U16 CardNo, U16 NodeID, U16 SlotNo, U16* Done, U8* Data, U32* ErrorCode)

■ 목적

Slave 공통 명령, 이 Node 에 대한 반환 데이터를 확인합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Done	U16*	번호 단위	0: 완료 1: 실행 중 2: 오류
Data	U8	수치 단위	메시지를 수집한 데이터
ErrorCode	U32*	번호 단위	오류 코드

■ 예제

```
U16 Status;
U16 CardNo=16,NodeID=1,SlotNo=0;
U16 Index=0x6040, SubIndex=0, DataSize=4;
U8 Data[4] = {0};
U32 ErrorCode;

Status = _ECAT_Slave_SDO_Read_Response (CardNo, NodeID, SlotNo,
&Done, Data, &ErrorCode);
```

7.6 _ECAT_Slave_SDO_Wait_All_Done

■ 포맷

U16 PASCAL _ECAT_Slave_SDO_Wait_All_Done (U16 CardNo, U16 AxisNum, U16* NodeID, U16* SlotNo)

■ 목적

Slave 공통 명령, 복수 Node 번호에 현재 누적된 SDO 명령이 전부 완료될 때까지 대기합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNum	U16	수치 단위	관련 축수
NodeID	U16*	번호 단위	NodeID 번호
SlotNo	U16*	번호 단위	SlotID 번호

■ **예제**

```
U16 Status;
U16 AxisNum = 2;
U16 CardNo=16, NodeID[2] = {0,1} ,SlotNo[2] = {0,0};

Status = _ECAT_Slave_SDO_Wait_All_Done(CardNo, AxisNum ,NodeID, SlotNo);
```

7.7 _ECAT_Slave_SDO_Get_ErrorCode

■ **포맷**

U16 PASCAL _ECAT_Slave_SDO_Get_ErrorCode (U16 CardNo, U16 NodeID, U16 SlotNo, U32* ErrorCode)

■ **목적**

Slave 공통 명령, Send_Message 또는 Read_Message 에 ERR_ECAT_SDO_Return_Error 오류가 발생할 때, 해당 API 를 사용하여 오류 코드를 확인할 수 있습니다. 오류 코드는 CANopen 또는 모듈 개체 정의에서 참조할 수 있습니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
ErrorCode	U32*	번호 단위	오류 코드

■ **예제**

```
U16 Status;
U16 CardNo=16,NodeID=1,SlotNo=0;
U32 ErrorCode;

Status = _ECAT_Slave_SDO_Get_ErrorCode(CardNo, NodeID, SlotNo, &ErrorCode);
```

7

_ECAT_Slave_SDO_Get_ErrorCode 에서 ErrorCode 테이블 가져오기

코드	설명
0x05 03 00 00	세그먼트 전송 시 전환 비트에 변화 없음
0x05 04 00 00	SDO 전송 시간 초과
0x05 04 00 01	명령 코드 무효 또는 알 수 없음
0x05 04 00 05	메모리 overflow
0x06 01 00 05	임의의 Object 에 대한 조작을 지원하지 않음
0x06 01 00 00	하나의 Object 를 확인함
0x06 03 00 02	하나의 RO Object 를 입력함
0x06 02 00 00	Object 가 Object Dictionary 에 존재하지 않음
0x06 04 00 41	Object 를 PDO 에 매핑할 수 없음
0x06 04 00 42	매핑할 Object 수량과 길이가 PDO 데이터 길이를 초과함
0x06 04 00 43	일반 파라미터가 호환되지 않음
0x06 04 00 47	장치의 일반 메모리가 호환되지 않음
0x06 06 00 00	하드웨어 오류로 인하여 조작 실패
0x06 07 00 10	데이터 유형이 매칭되지 않음, 서버 파라미터 길이가 매칭되지 않음
0x06 07 00 12	데이터 유형이 매칭되지 않음, 서버 파라미터 길이가 매우 김
0x06 07 00 13	데이터 유형이 매칭되지 않음, 서버 파라미터 길이가 매우 짧음
0x06 09 00 11	sub index 가 존재하지 않음
0x06 09 00 30	입력한 데이터 값이 범위를 초과함
0x06 09 00 31	입력한 데이터 값이 매우 큼
0x06 09 00 32	입력한 데이터 값이 매우 작음
0x06 09 00 36	최대값이 최소값보다 작음
0x08 00 00 00	일반 오류
0x08 00 00 20	응용 프로그램에 데이터를 전송 또는 저장할 수 없음
0x08 00 00 21	로컬 제어 원인으로 응용 프로그램에 데이터를 전송 또는 저장할 수 없음
0x08 00 00 22	현재 장치 상태의 원인으로 응용 프로그램에 데이터를 전송 또는 저장할 수 없음
0x08 00 00 23	Object Dictionary 에 오류 발생, 또는 Object Dictionary 를 찾지 못함

7.8 _ECAT_Slave_SDO_Check_Done

■ **포맷**

U16 PASCAL _ECAT_Slave_SDO_Check_Done (U16 CardNo, U16 NodeID, U16 SlotNo, U16 *Done)

■ **목적**

Slave 공통 명령, 단일 Node 에 현재 누적된 SDO 명령이 완료되었는지 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Done	U16*	번호 단위	Done 정의 0: 완료 1: 완료되지 않음

■ **예제**

```
U16 Status;
U16 CardNo=16, NodeID = 1 ,SlotNo = 0, Done;

Status = _ECAT_Slave_SDO_Check_Done (CardNo, NodeID, SlotNo, &Done);
```

7.9 _ECAT_Slave_PDO_Get_OD_Data

■ **포맷**

U16 PASCAL _ECAT_Slave_PDO_Get_OD_Data (U16 CardNo, U16 NodeID, U16 SlotNo, U16 IOType, U16 ODIndex, U16 ODSubIndex, U16 ByteSize, U8 *Data)

■ **목적**

Slave 공통 명령, 해당 Node 에 임의의 OD 코드 데이터를 확인하며, 해당 데이터는 PDO 구성에 매핑되어야 합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
IOType	U16	번호 단위	IOType 0 : Rx 1 : Tx
ODIndex	U16	번호 단위	데이터의 Object Index 를 저장합니다
ODSubIndex	U16	번호 단위	변수의 subindex 를 반드시 저장합니다
ByteSize	U16	수치 단위	데이터 공간 크기
Data	U8*	수치 단위	OD 코드 위치를 지정한 데이터 내용을 확인합니다

■ 예제

```
U16 Status;
U8 Data = 0;
U16 CardNo=16, NodeID = 1 ,SlotNo = 0, IOType = 0;
U16 ODIndex = 0x1810, ODSubIndex = 0x01, ByteSize = 0x43;

Status = _ECAT_Slave_PDO_Get_OD_Data (CardNo, NodeID, SlotNo, IOType,
ODIndex, ODSubIndex, ByteSize, &Data);
```

7.10 _ECAT_Slave_PDO_Set_OD_Data

■ **포맷**

U16 PASCAL _ECAT_Slave_PDO_Set_OD_Data (U16 CardNo, U16 NodeID, U16 SlotNo, U16 ODIndex, U16 ODSubIndex, U8 *Data)

■ **목적**

Slave 공통 명령, 해당 Node 에 임의의 OD 코드 데이터를 전송하며, 해당 데이터는 PDO 구성에 매핑되어야 하며, Tx 만 사용 가능합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
ODIndex	U16	번호 단위	데이터의 Object Index 를 저장합니다
ODSubIndex	U16	번호 단위	변수의 subindex 를 반드시 저장합니다
Data	U8*	수치 단위	OD 코드 위치를 지정한 데이터 내용

■ 예제

```
U16 Status;
U8 Data = 0;
U16 CardNo=16, NodeID = 1 ,SlotNo = 0, IOType = 0;
U16 ODIndex = 0x1780, ODSubIndex = 0x01

Status = _ECAT_Slave_PDO_Set_OD_Data (CardNo, NodeID, SlotNo, ODIndex,
ODSubIndex, &Data);
```

7.11 _ECAT_Slave_PDO_Get_Information

■ 포맷

U16 PASCAL _ECAT_Slave_PDO_Get_Information (U16 CardNo, U16 NodeID, U16 SlotNo, U16 IOType, U16 *ODCnt, U16 *StartIndex)

■ 목적

Slave 명령, 각 Node PDO 구성의 기본 데이터를 확인합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
IOType	U16	번호 단위	0 : Master RX 1 : Master Tx
ODCnt	U16*	수치 단위	해당 Node 에서 해당 IOType 이 반드시 보유해야 할 OD 수량
StartIndex	U16*	수치 단위	해당 Node 의 시작 데이터 위치

■ 예제

```
U16 Status;
U16 CardNo=16, NodeID = 1 ,SlotNo = 0, IOType = 0;
U16 ODCnt, StartIndex;

Status = _ECAT_Slave_PDO_Get_Information(CardNo, NodeID, SlotNo, IOType,
&ODCnt, &StartIndex);
```

7.12 _ECAT_Slave_PDO_Get_Detail_Mapping

■ 포맷

U16 PASCAL _ECAT_Slave_PDO_Get_Detail_Mapping (U16 CardNo, U16 NodeID, U16 SlotNo, U16 IOType, U16 ODSeqID, U16 *ODIndex, U16 *ODSubIndex, U16 *ODByteSize, U16 *ODStartIndex)

■ 목적

Slave 명령, 각 Node PDO 구성의 상세 매핑 데이터를 확인합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
IOType	U16	번호 단위	0 : Master RX 1 : Master Tx
ODSeqID	U16	번호 단위	OD 시퀀스 번호
ODIndex	U16*	번호 단위	데이터의 Object Index 를 저장합니다
ODSubIndex	U16*	번호 단위	변수의 subindex 를 반드시 저장합니다
ByteSize	U16*	수치 단위	데이터 공간 크기
ODStartIndex	U16*	수치 단위	해당 OD 의 시작 데이터 위치, Byte Size 배열 Index (단일값)

■ **예제**

```
U16 Status;
U16 CardNo=16, NodeID = 1 ,SlotNo = 0, IOType = 0, ODSeqID = 0, ODCnt ,
StartIndex;
U16 ODIndex[8]={0}, ODSubIndex[8]={0}, ODBitSize[8]={0}, ODStartIndex[8]={0};

Status = _ECAT_Slave_PDO_Get_Information(CardNo, NodeID, SlotNo, IOType,
&ODCnt, &StartIndex);

  for (ODSeqID = 0; ODSeqID < ODCnt; ODSeqID++)
  {
       Status = _ECAT_Slave_PDO_Get_Detail_Mapping(CardNo, NodeID, SlotNo,
       IOType, ODSeqID, &ODIndex[ODSeqID], &ODSubIndex[ODSeqID],
       &ODBitSize[ODSeqID],&ODStartIndex[ODSeqID]);
  }
```

7.13 _ECAT_Slave_PDO_Get_Rx_Data

■ **포맷**

U16 PASCAL _ECAT_Slave_PDO_Get_Rx_Data (U16 CardNo, BYTE *Data)

■ **목적**

Slave 명령, 모든 Node 의 PDO Rx 구성 데이터를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
Data	BYTE	수치 단위	모든 Node 의 Rx 데이터, 데이터의 총 크기 0x600

■ **예제**

```
U16 Status;
U16 CardNo=16;
BYTE Data[0x600] = {0};

Status = _ECAT_Slave_PDO_Get_Rx_Data(CardNo, &Data);
```

7.14 _ECAT_Slave_PDO_Get_Tx_Data

■ **포맷**

U16 PASCAL _ECAT_Slave_PDO_Get_Tx_Data (U16 CardNo, BYTE *Data)

■ **목적**

Slave 명령, 모든 Node 의 PDO Tx 구성 데이터를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
Data	BYTE	수치 단위	모든 Node 의 Tx 데이터, 데이터의 총 크기 0x600

■ **예제**

```
U16 Status;
U16 CardNo=16;
BYTE Data[0x600] = {0};

Status = _ECAT_Slave_PDO_Get_Tx_Data(CardNo, &Data);
```

7.15 _ECAT_Slave_PDO_Set_Tx_Data

■ **포맷**

U16 PASCAL _ECAT_Slave_PDO_Set_Tx_Data (U16 CardNo, BYTE *Data)

■ **목적**

Slave 명령, 모든 Node 의 PDO Tx 구성 데이터를 입력합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
Data	BYTE	수치 단위	모든 Node 의 Tx 데이터, 데이터의 총 크기 0x600

■ **예제**

```
U16 Status;
U16 CardNo=16;
BYTE Data[0x600] = {0};
Status = _ECAT_Slave_PDO_Get_Tx_Data(CardNo, &Data);

// TxData 내부의 데이터를 직접 수정합니다
Data[0x001] = 0x01;

Status = _ECAT_Slave_PDO_Set_Tx_Data(CardNo, &Data);
```

7.16 _ECAT_Slave_PDO_Set_Tx_Detail_Data

■ **포맷**

U16 PASCAL _ECAT_Slave_PDO_Set_Tx_Detail_Data (U16 CardNo, U16 NodeID, U16 SlotNo, U16 ODStartIndex, U16 ByteSize, U8 *Data)

■ **목적**

Slave 명령, 각 Node 의 PDO Tx 상세 매핑 데이터를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
StartIndex	U16	수치 단위	해당 Node 의 시작 데이터 위치, Byte Size 배열 Index (단일값)
ByteSize	U16	수치 단위	전송할 Tx 데이터 크기
Data	U8*	수치 단위	전송할 Tx 데이터

■ **예제**

```
U16 Status;
U16 CardNo=16, NodeID = 1 ,SlotNo = 0, IOType = 1;
U16 ODCnt, StartIndex = 0x60, ByteSize = 4;
U8 Data[4]={0, 1, 0, 1};

Status = _ECAT_Slave_PDO_Get_Information(CardNo, NodeID, SlotNo, IOType,
&ODCnt, &StartIndex);

Status = _ECAT_Slave_PDO_Set_Tx_Detail_Data(CardNo, NodeID, SlotNo,
StartIndex, ByteSize, &Data);
```

(이 페이지는 공란으로 비워둡니다)

Motion Slave 공통 API

8

다음은 현재 동작 제어 모드 확인 및 설정, ControlWord 데이터 확인, StatusWord 데이터 확인, 현재 Command 데이터 확인 및 설정, 현재 Position 데이터 확인 및 설정, Mdone 데이터 확인, 현재 Speed 데이터 확인, Servo On / Off 설정, Alarm 삭제, 긴급 정지 및 감속 정지 설정, Latch 관련 설정, 목표 명령 데이터 확인, 현재 명령 Buffer 수량 확인, 모터 피드백 토크값 확인, Alarm 작동 시 반응 모드 설정 등 Motion Slave 공통 관련 API 사용법에 대한 설명입니다.

8

Motion Slave 공통 API 테이블

함수의 명칭	설명
_ECAT_Slave_Motion_Get_MoveMode	MotionSlave 명령, 현재의 동작 제어 모드 (PP / CSP / Home …)를 확인합니다.
_ECAT_Slave_Motion_Get_ControlWord	MotionSlave 명령, 현재의 ControlWord 데이터를 확인합니다.
_ECAT_Slave_Motion_Get_StatusWord	MotionSlave 명령, 현재의 StatusWord 데이터를 확인합니다.
_ECAT_Slave_Motion_Get_Command	MotionSlave 명령, 현재의 Command 데이터를 확인합니다.
_ECAT_Slave_Motion_Get_Position	MotionSlave 명령, 현재의 Position 데이터를 확인합니다.
_ECAT_Slave_Motion_Get_Mdone	MotionSlave 명령, 현재의 Mdone 데이터를 확인합니다.
_ECAT_Slave_Motion_Get_Current_Speed	MotionSlave 명령, 현재의 Current_Speed 데이터를 확인합니다.
_ECAT_Slave_Motion_Set_Svon	MotionSlave 명령, Servo On / Off 를 설정합니다
_ECAT_Slave_Motion_Ralm	MotionSlave 명령, Alm 을 리셋합니다.
_ECAT_Slave_Motion_Set_Position	MotionSlave 명령, Position 위치를 설정합니다.
_ECAT_Slave_Motion_Set_Command	MotionSlave 명령, Command 위치를 설정합니다.
_ECAT_Slave_Motion_Emg_Stop	MotionSlave 명령, 긴급 정지를 실행합니다.
_ECAT_Slave_Motion_Sd_Stop	MotionSlave 명령, Slow Down Stop 감속 및 정지합니다
_ECAT_Slave_Motion_Set_TouchProbe_Config	MotionSlave 명령, "트리거 검색" 기능의 실행모드를 설정합니다.
_ECAT_Slave_Motion_Set_TouchProbe_QuickStart	MotionSlave 명령, "트리거 검색" 기능을 빨리 활성화합니다.
_ECAT_Slave_Motion_Set_TouchProbe_QuickDone	MotionSlave 명령, "트리거 검색" 기능을 빨리 리셋합니다.
_ECAT_Slave_Motion_Set_TouchProbe_Disable	MotionSlave 명령, "트리거 검색" 기능을 종료합니다.
_ECAT_Slave_Motion_Get_TouchProbe_Status	MotionSlave 명령, 현재 "트리거 검색" 기능의 상태를 확인합니다.
_ECAT_Slave_Motion_Get_TouchProbe_Position	MotionSlave 명령, 현재 검색한 위치를 확인합니다.
_ECAT_Slave_Motion_Set_MoveMode	MotionSlave 명령, 현재 동작 모드를 설정합니다.

함수의 명칭	설명
_ECAT_Slave_Motion_Get_Target_Position	MotionSlave 명령, 현재 목표 위치의 데이터를 확인합니다.
_ECAT_Slave_Motion_Get_Buffer_Length	MotionSlave 명령, 현재 명령 Buffer 수량을 확인합니다.
_ECAT_Slave_Motion_Get_Torque	MotionSlave 명령, 현재의 모터 피드백 토크 값을 확인합니다.
_ECAT_Slave_Motion_Set_Alm_Reaction	MotionSlave 명령, Alm 작동 시 반응 모드를 설정합니다.
_ECAT_Slave_Motion_Get_Actual_ Command	SlaveMotion 명령, Virtual Command 설정에 관계 없이 현재 Command 데이터를 확인합니다.
_ECAT_Slave_Motion_Get_Actual_Position	SlaveMotion 명령, Virtual Command 설정에 관계 없이 현재 Position 데이터를 확인합니다.

8.1 _ECAT_Slave_Motion_Get_MoveMode

8

■ 포맷

U16 PASCAL _ECAT_Slave_Motion_Get_MoveMode (U16 CardNo, U16 AxisNo, U16 SlotNo, U8 *Mode)

■ 목적

MotionSlave 명령, 현재의 동작 제어 모드 (PP / CSP / Home …)를 확인합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Mode	U8*	번호 단위	1: PP 모드 2: Velocity 모드 3: PV 모드 4: PT 모드 6: Home 모드 7: IP 모드 8: CSP 모드 9: CSV 모드 10: CST 모드

■ 예제

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
U8 Mode;

Status = _ECAT_Slave_Motion_Get_MoveMode (CardNo, AxisNo, SlotNo, &Mode);
```

8.2 _ECAT_Slave_Motion_Get_ControlWord

■ 포맷

U16 PASCAL _ECAT_Slave_Motion_Get_ControlWord (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 *ControlWord)

■ 목적

MotionSlave 명령, 현재의 ControlWord 데이터를 확인합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
ControlWord	U16*	수치 단위	ControlWord 데이터 (관련 정의는 아래의 그림을 참고하고, 저장 관련 부분은 각 Slave 정보를 참고하시기 바랍니다)

■ 예제

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
U16 ControlWord ;

Status = _ECAT_Slave_Motion_Get_ControlWord (CardNo, AxisNo, SlotNo,
&ControlWord);
```

Controlword Description

MSB								LSB							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
High-Byte								Low-Byte							

그림 8.2.1 Control Word 해당 비트

Bit	Name	Mandatory
0	Switch On	✓
1	Disable Voltage	✓
2	Quick Stop	✓
3	Enable Operation	✓
4	Operation Mode Specific	
5	Operation Mode Specific	
6	Operation Mode Specific	
7	Reset Fault	✓
8	Halt	
9	Reserved	
10	Reserved	
11	Manufacturer Specific	
12	Manufacturer Specific	
13	Manufacturer Specific	
14	Manufacturer Specific	
15	Manufacturer Specific	

그림 8.2.2 Control Word 비트 의의 설명

8.3 _ECAT_Slave_Motion_Get_StatusWord

■ 포맷

U16 PASCAL _ECAT_Slave_Motion_Get_StatusWord (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 * StatusWord)

■ 목적

MotionSlave 명령, 현재의 StatusWord 데이터를 확인합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
StatusWord	U16*	수치 단위	StatusWord 데이터 (관련 정의는 뒤의 그림을 참고하고, 저장 관련 부분은 각 Slave 정보를 참고하시기 바랍니다)

■ 예제

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
U16 StatusWord;

Status = _ECAT_Slave_Motion_Get_StatusWord (CardNo, AxisNo, SlotNo,
&StatusWord);
```

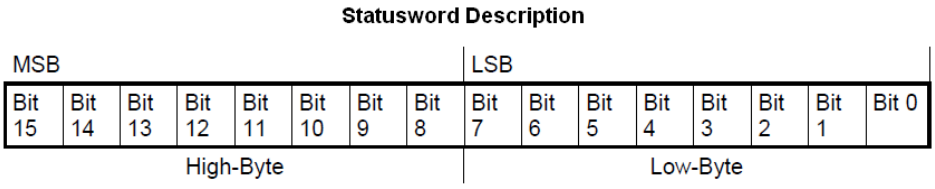


그림 8.3.1 Status Word 해당 비트

Bit	Name	Mandatory
0	Ready to Switch On	✓
1	Switched On	✓
2	Operation Enabled	✓
3	Fault	✓
4	Voltage Disabled	✓
5	Quick Stop	✓
6	Switch On Disabled	✓
7	Warning	
8	Manufacturer Specific	
9	Remote	✓
10	Target Reached	✓
11	Internal Limit Active	✓
12	Operation Mode Specific	
13	Operation Mode Specific	
14	Manufacturer Specific	
15	Manufacturer Specific	

그림 8.3.2 Status Word 해당 비트 의의

8.4 _ECAT_Slave_Motion_Get_Command

■ **포맷**

U16 PASCAL _ECAT_Slave_Motion_Get_Command (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 *Command)

■ **목적**

MotionSlave 명령, 현재의 Command 데이터를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Command	I32*	펄스 수	MotionSlave 명령, 현재의 Command 데이터를 확인합니다 CSP 모드에서 나타내는 것은 현재 명령 위치입니다 CSV 모드에서 나타내는 것은 현재 명령 속도입니다 CST 모드에서 나타내는 것은 현재 명령 토크 천분율입니다

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 Command=0;

Status = _ECAT_Slave_Motion_Get_Command (CardNo, AxisNo, SlotNo,
&Command);
```

8.5 _ECAT_Slave_Motion_Get_Position

8

■ 포맷

U16 PASCAL _ECAT_Slave_Motion_Get_Position (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 *Position)

■ 목적

MotionSlave 명령, 현재의 Position 데이터를 확인합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Position	I32*	펄스 수	현재 Position 데이터를 확인합니다

■ 예제

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0;
I32 Position=0;

Status = _ECAT_Slave_Motion_Get_Position (CardNo, AxisNo, SlotNo, &Position);
```

8.6 _ECAT_Slave_Motion_Get_Mdone

■ **포맷**

U16 PASCAL _ECAT_Slave_Motion_Get_Mdone (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 *Mdone)

■ **목적**

MotionSlave 명령, 현재의 Mdone 데이터를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Mdone	U16*	번호 단위	CS 계열 모드에서 0: 정지 상태 1: 가속 중 2: 등속 상태(CSP) / 목표 속도에 도달(CSV) / 목표 Torque 에 도달(CST) 3: 감속 중 5: MailBox 처리 중 Profile 계열 모드에서 0: 정지 상태 1: 동작 중

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
U16 Mdone;

Status = _ECAT_Slave_Motion_Get_Mdone (CardNo, AxisNo, SlotNo, &Mdone);
```

8.7 _ECAT_Slave_Motion_Get_Current_Speed

8

■ 포맷

U16 PASCAL _ECAT_Slave_Motion_Get_Current_Speed (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 *Speed)

■ 목적

MotionSlave 명령, 현재의 Current_Speed 데이터를 확인합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Speed	I32*	수치 단위	현재 Speed 데이터를 확인합니다

■ 예제

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 Speed;

Status = _ECAT_Slave_Motion_Get_Current_Speed (CardNo, AxisNo, SlotNo,
&Speed);
```

8.8 _ECAT_Slave_Motion_Set_Svon

■ **포맷**

U16 PASCAL _ECAT_Slave_Motion_Set_Svon (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ **목적**

MotionSlave 명령, Servo On / Off 를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Enable	U16	옵션 단위	0 : Servo OFF 1 : Servo ON

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
U16 Enable=1;

Status = _ECAT_Slave_Motion_Set_Svon(CardNo, AxisNo, SlotNo, Enable);
```

8.9 _ECAT_Slave_Motion_Ralm

■ 포맷

U16 PASCAL _ECAT_Slave_Motion_Ralm (U16 CardNo, U16 AxisNo, U16 SlotNo)

■ 목적

MotionSlave 명령, Alm 을 리셋합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호

■ 예제

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;

Status = _ECAT_Slave_Motion_Ralm(CardNo, AxisNo, SlotNo);
```

8.10 _ECAT_Slave_Motion_Set_Position

■ **포맷**

U16 PASCAL _ECAT_Slave_Motion_Set_Position (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 NewPosition)

■ **목적**

MotionSlave 명령, Position 위치를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
NewPosition	I32	수치 단위	Position 위치를 설정합니다

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 NewPosition=2500000;

Status = _ECAT_Slave_Motion_Set_Position(CardNo, AxisNo, SlotNo, NewPosition);
```

8.11 _ECAT_Slave_Motion_Set_Command

8

■ 포맷

U16 PASCAL _ECAT_Slave_Motion_Set_Command (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 NewCommand)

■ 목적

MotionSlave 명령, Command 위치를 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
NewCommand	I32	수치 단위	Command 위치를 설정합니다

■ 예제

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 NewCommand=3000000;

Status = _ECAT_Slave_Motion_Set_Command(CardNo, AxisNo, SlotNo,
NewCommand);
```

8.12 _ECAT_Slave_Motion_Get_Actual_Command

■ **포맷**

U16 PASCAL _ECAT_Slave_Motion_Get_Actual_Command (U16 CardNo, U16 AxisNo, U16 SlotNo, I32* ActualCommand)

■ **목적**

SlaveMotion 명령, Virtual Command 설정에 관계 없이 현재 Command 데이터를 확인합니다.

CSP 모드에서 나타내는 것은 현재 명령 위치입니다; CSV 모드에서 나타내는 것은 현재 명령 속도입니다; CST 모드에서 나타내는 것은 현재 명령 토크 천분율입니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	축 번호
SlotNo	U16	번호 단위	Node 번호
ActualCommand	I32*	수치 단위	모드에 따라 단위를 결정합니다

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 ActualCommand;

Status = _ECAT_Slave_Motion_ Get_Actual_Command (CardNo, AxisNo, SlotNo,
&ActualCommand);
```

8.13 _ECAT_Slave_Motion_Get_Actual_Position

8

■ 포맷

U16 PASCAL _ECAT_Slave_Motion_Get_Actual_Position (U16 CardNo, U16 AxisNo, U16 SlotNo, I32* ActualPositon)

■ 목적

SlaveMotion 명령, Virtual Command 설정에 관계 없이 현재 Position 데이터를 확인합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	축 번호
SlotNo	U16	번호 단위	Node 번호
ActualPosition	I32*	수치 단위	Virtual Command 설정에 관계 없이 현재 Position 데이터를 확인합니다.

■ 예제

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 ActualPosition;

Status = _ECAT_Slave_Motion_Get_Actual_Position(CardNo, AxisNo, SlotNo,
&ActualPosition);
```

8.14 _ECAT_Slave_Motion_Emg_Stop

■ **포맷**

U16 PASCAL _ECAT_Slave_Motion_Emg_Stop (U16 CardNo, U16 AxisNo, U16 SlotNo)

■ **목적**

MotionSlave 명령, 긴급 정지를 실행합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;

Status = _ECAT_Slave_Motion_Emg_Stop(CardNo, AxisNo, SlotNo);
```

8.15 _ECAT_Slave_Motion_Sd_Stop

8

■ 포맷

U16 PASCAL _ECAT_Slave_Motion_Sd_Stop (U16 CardNo, U16 AxisNo, U16 SlotNo, F64 Tdec)

■ 목적

MotionSlave 명령, Slow Down Stop 감속 및 정지합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Tdec	F64	시간 단위	지정된 감속 시간 CSP, CSV, CST 모드의 시간 단위는 초입니다 HOME, PP, PV, PT 모드의 시간 단위는 드라이브 단위입니다 inc/s^2 (0x6083 Sub 0)

■ 예제

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
F64 Tdec=0.1;

Status = _ECAT_Slave_Motion_Sd_Stop(CardNo, AxisNo, SlotNo, Tdec);
```

8.16 _ECAT_Slave_Motion_Set_TouchProbe_Config

■ 포맷

U16 PASCAL _ECAT_Slave_Motion_Set_TouchProbe_Config (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 TriggerMode, U16 Signal_Source)

■ 목적

MotionSlave 명령, "트리거 검색" 기능의 실행모드를 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
TriggerMode	U16	옵션 단위	사용한 Slave CANopen 60B8 bit1 의 정의를 참고하십시오
Signal_Source	U16	옵션 단위	사용한 Slave CANopen 60B8 bit2 의 정의를 참고하십시오

■ 예제

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
U16 TriggerMode =1, Signal_Source=1;

Status = _ECAT_Slave_Motion_Set_TouchProbe_Config(CardNo, AxisNo, SlotNo,
TriggerMode, Signal_Source);
```

8.17 _ECAT_Slave_Motion_Set_TouchProbe_QuickStart

8

■ 포맷

U16 PASCAL _ECAT_Slave_Motion_Set_TouchProbe_QuickStart (U16 CardNo, U16 AxisNo, U16 SlotNo)

■ 목적

MotionSlave 명령, "트리거 검색" 기능을 빨리 활성화합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호

■ 예제

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;

Status = _ECAT_Slave_Motion_Set_TouchProbe_QuickStart(CardNo, AxisNo, SlotNo);
```

8.18 _ECAT_Slave_Motion_Set_TouchProbe_ QuickDone

■ **포맷**

U16 PASCAL _ECAT_Slave_Motion_Set_TouchProbe_QuickDone (U16 CardNo, U16 AxisNo, U16 SlotNo)

■ **목적**

MotionSlave 명령, "트리거 검색" 기능을 빨리 리셋합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;

Status = _ECAT_Slave_Motion_Set_TouchProbe_QuickDone(CardNo, AxisNo,
SlotNo);
```

8.19 _ECAT_Slave_Motion_Set_TouchProbe_Disable

8

■ 포맷

U16 PASCAL _ECAT_Slave_Motion_Set_TouchProbe_Disable (U16 CardNo, U16 AxisNo, U16 SlotNo)

■ 목적

MotionSlave 명령, "트리거 검색" 기능을 종료합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호

■ 예제

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;

Status = _ECAT_Slave_Motion_Set_TouchProbe_Disable(CardNo, AxisNo, SlotNo);
```

8.20 _ECAT_Slave_Motion_Get_TouchProbe_Status

■ **포맷**

U16 PASCAL _ECAT_Slave_Motion_Get_TouchProbe_Status (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 *Status)

■ **목적**

MotionSlave 명령, 현재 "트리거 검색" 기능의 상태를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Status	U16*	수치 단위	사용한 Slave CANopen 60B9 의 정의를 참고하십시오

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
U16 Status;

Status = _ECAT_Slave_Motion_Get_TouchProbe_Status(CardNo, AxisNo, SlotNo,
&Status);
```

8.21 _ECAT_Slave_Motion_Get_TouchProbe_Position

8

■ 포맷

U16 PASCAL _ECAT_Slave_Motion_Get_TouchProbe_Position (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 *LatchPosition)

■ 목적

MotionSlave 명령, 현재 검색한 위치를 확인합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
LatchPosition	I32*	수치 단위	사용한 Slave CANopen 60BA 의 read back 값을 참고하십시오

■ 예제

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 LatchPosition;

Status = _ECAT_Slave_Motion_Get_TouchProbe_Position(CardNo, AxisNo, SlotNo,
&LatchPosition);
```

8.22 _ECAT_Slave_Motion_Set_MoveMode

■ **포맷**

U16 PASCAL _ECAT_Slave_Motion_Set_MoveMode (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 MoveMode)

■ **목적**

MotionSlave 명령, 현재 동작 모드를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
MoveMode	U16	옵션 단위	MoveMode 설명 0: Null 모드 1: PP 모드 3: PV 모드 4: PT 모드 6: Home 모드 8: CSP 모드 9: CSV 모드 10: CST 모드

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0, MoveMode=1;

Status = _ECAT_Slave_Motion_Set_MoveMode(CardNo, AxisNo, SlotNo, MoveMode);
```

8.23 _ECAT_Slave_Motion_Get_Target_Command

8

■ 포맷

U16 PASCAL _ECAT_Slave_Motion_Get_Target_Command (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 *TargetCommand)

■ 목적

MotionSlave 명령, 현재 목표 명령의 데이터를 확인합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
TargetCommand	I32*	수치 단위	CSP, PP 모드: 목표 위치 CSV, PV 모드: 목표 속도 CST, PT 모드: 목표 토크 Home 모드: 효과 없음 (반환 0)

■ 예제

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 TargetCommand = 0;

Status = _ECAT_Slave_Motion_Get_Target_Command (CardNo, AxisNo, SlotNo,
&TargetCommand);
```

8.24 _ECAT_Slave_Motion_Get_Buffer_Length

■ **포맷**

U16 PASCAL _ECAT_Slave_Motion_Get_Buffer_Length (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 *BufferLength)

■ **목적**

MotionSlave 명령, 현재 명령 Buffer 수량을 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
BufferLength	U16*	수치 단위	현재 명령의 Buffer 수량

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0, BufferLength = 0;

Status = _ECAT_Slave_Motion_Get_Buffer_Length (CardNo, AxisNo, SlotNo,
& BufferLength);
```

8.25 _ECAT_Slave_Motion_Get_Torque

8

■ **포맷**

U16 PASCAL _ECAT_Slave_Motion_Get_Torque (U16 CardNo, U16 AxisNo, U16 SlotNo, I16 *Torque)

■ **목적**

MotionSlave 명령, 현재의 모터 피드백 토크 값을 확인합니다 (단위: 천분율).

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Torque	I16*	수치 단위	현재의 모터 피드백 토크 값

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I16 Torque = 0;

Status = _ECAT_Slave_Motion_Get_Torque (CardNo, AxisNo, SlotNo, &Torque);
```

8.26 _ECAT_Slave_Motion_Set_Alm_Reaction

■ 포맷

U16 PASCAL _ECAT_Slave_Motion_Set_Alm_Reaction (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Fault_Type, U16 Waring_Type);

■ 목적

MotionSlave 명령, Alm 작동 시 반응 모드를 설정합니다.

※ Group 에서도 유효합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Fault_Type	U16	옵션 단위	오류(Fault) 발생 시의 반응 모드를 설정합니다. 0: 새로운 명령을 정지하거나 자동 거부하지 않습니다. 1: 상위 트리거가 실행되면 새로운 명령을 자동 거부하지 않고 현재 동작을 정지합니다. 2: 해당 상태가 해제될 때까지 정지 상태를 유지합니다.
Warning_Type	U16	옵션 단위	경고(Warning) 발생 시의 반응 모드를 설정합니다. 0: 새로운 명령을 정지하거나 자동 거부하지 않습니다. 1: 상위 트리거가 실행되면 새로운 명령을 자동 거부하지 않고 현재 동작을 정지합니다. 2: 해당 상태가 해제될 때까지 정지 상태를 유지합니다.

8

■ 예제

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I16 Fault_Type = 2, Waring_Type = 1;

Status = _ECAT_Slave_Motion_Set_Alm_Reaction (CardNo, AxisNo, SlotNo,
Fault_Type, Waring_Type);
```

Motion Slave CSP API

9

다음은 단축 직선 동작, 단축 정속 지속 동작, 2 축 원형 동작, 양축 나선 동작, 3 축 구형 동작, 3 축 나선 동작, 다축 직선 동작, 다축 3 점 고정 위치 평활 동작, 전자 기어비 설정, 소프트웨어 limit 설정 및 확인, 새로운 목표 위치 설정, 새로운 목표 속도 설정, 다기능 변속 설정, 속도 연속 기능 활성화 및 해제, 속도 연속 모드 설정, 연속된 속도 명령을 백분율로 합성하여 설정, S 곡선의 T 곡선 비율 설정, 최대 속도 정의 모드 설정, 동시 동작 축 설정, 다축 3 점 S 곡선 평활 보간, 단축 2 단 직선 동작, 단축 PVT 동작, 가상 위치 설정, 구역 보상 기능 등 Motion Slave CSP 관련 API 사용법에 대한 설명입니다.

9

Motion Slave CSP API 테이블

함수의 명칭	설명
_ECAT_Slave_CSP_Start_Move	MotionSlaveCSP 명령, 단축 직선 동작.
_ECAT_Slave_CSP_Start_V_Move	MotionSlaveCSP 명령, 단축 정속 지속 동작.
_ECAT_Slave_CSP_Start_Arc_Move	MotionSlaveCSP 명령, 2 축 원형 동작. (원 중심+각도)
_ECAT_Slave_CSP_Start_Arc2_Move	MotionSlaveCSP 명령, 2 축 원형 동작. (최종 좌표+각도)
_ECAT_Slave_CSP_Start_Arc3_Move	MotionSlaveCSP 명령, 2 축 원형 동작. (원 중심+최종 좌표)
_ECAT_Slave_CSP_Start_Spiral_Move	MotionSlaveCSP 명령, 2 축 나선 동작. (원 중심+각도)
_ECAT_Slave_CSP_Start_Spiral2_Move	MotionSlaveCSP 명령, 2 축 나선 동작 (최종 좌표+회전수)
_ECAT_Slave_CSP_Start_Sphere_Move	MotionSlaveCSP 명령, 3 축 구형 동작 (세 점이 구형을 생성)
_ECAT_Slave_CSP_Start_Heli_Move	MotionSlaveCSP 명령, 3 축 나선 동작
_ECAT_Slave_CSP_Start_Multiaxes _Move	MotionSlaveCSP 명령, 다축 직선 동작.
_ECAT_Slave_CSP_Start_Msbrline_Move	MotionSlaveCSP 명령, 다축 3 점 고정 위치 평활 동작.
_ECAT_Slave_CSP_Set_Gear	MotionSlaveCSP 명령, 전자 기어비를 설정합니다.
_ECAT_Slave_CSP_Set_Softlimit	MotionSlaveCSP 명령, 소프트웨어 limit 를 설정합니다.
_ECAT_Slave_CSP_TargetPos_Change	MotionSlaveCSP 명령, 새로운 목표 위치를 설정합니다.
_ECAT_Slave_CSP_Velocity_Change	MotionSlaveCSP 명령, 새로운 목표 속도를 설정합니다.
_ECAT_Slave_CSP_Feedrate_Overwrite	MotionSlaveCSP 명령, 다기능 변속 함수.
_ECAT_Slave_CSP_Speed_Continue _Enable	MotionSlaveCSP 명령, 속도 연속 기능 활성화 / 활성화 해제.
_ECAT_Slave_CSP_Speed_Continue_Set_ Mode	MotionSlaveCSP 명령, 속도 연속 모드를 설정합니다.
_ECAT_Slave_CSP_Speed_Continue_Set_ Combine_Ratio	MotionSlaveCSP 명령, 속도 연속을 설정합니다.
_ECAT_Slave_CSP_Scurve_Rate	MotionSlaveCSP 명령, S curve 의 T curve 비율을 설정합니다.
_ECAT_Slave_CSP_Liner_Speed_Master	MotionSlaveCSP 명령, 최대 속도의 정의 모드를 설정합니다

9

함수의 명칭	설명
_ECAT_Slave_CSP_Mask_Axis	SlaveMotionCSP 명령, 다축 또는 단축 명령에서, 일부 몇 개의 축을 정지시켜도 다른 축에 영향을 미치지 않습니다
_ECAT_Slave_CSP_Sync_Config	SlaveMotionCSP 명령, 동시 작동이 필요한 축을 설정합니다.
_ECAT_Slave_CSP_Sync_Move	SlaveMotionCSP 명령, Config 에서 설정한 축을 동시 작동합니다.
_ECAT_Slave_CSP_Start_Mabrline_Move	SlaveMotionCSP 명령, 다축 3 점 SCurve 평활 보간 동작
_ECAT_Slave_CSP_Start_2Segment _Move	SlaveMotionCSP 명령, 단축의 2 단 직선 동작을 지속합니다.
_ECAT_Slave_CSP_Start_PVT_Move	SlaveMotionCSP 명령, 단축 PVT 동작을 실행합니다.
_ECAT_Slave_CSP_Start_PVTComplete _Move	SlaveMotionCSP 명령, 단축 PVT 동작을 실행합니다.
_ECAT_Slave_CSP_Virtual_Set_Enable	SlaveMotionCSP 명령, 가상 위치를 설정합니다.
_ECAT_Slave_CSP_Virtual_Set _Command	SlaveMotionCSP 명령, 가상 위치를 설정하고, 현재 위치를 지정 위치로 설정합니다.
_ECAT_Slave_CSP_Get_SoftLimit_Status	SlaveMotionCSP 명령, 현재 소프트웨어 limit 상태를 확인합니다.
_ECAT_Slave_CSP_Pitch_Set_Interval	SlaveMotionCSP 명령, 구간 보상의 구간 거리를 설정합니다.
_ECAT_Slave_CSP_Pitch_Set_Mode	SlaveMotionCSP 명령, 구간 보상의 보상 모드를 설정합니다.
_ECAT_Slave_CSP_Pitch_Set_Org	SlaveMotionCSP 명령, 구간 보상의 원점 위치를 설정합니다.
_ECAT_Slave_CSP_Pitch_Set_Rel_Table	SlaveMotionCSP 명령, 구간 보상 내부 각 구간의 상대 보상값을 설정합니다.
_ECAT_Slave_CSP_Pitch_Set_Abs_Table	SlaveMotionCSP 명령, 구간 보상 내부 각 구간의 절대 보상값을 설정합니다.
_ECAT_Slave_CSP_Pitch_Set_Enable	SlaveMotionCSP 명령, 구간 보상 기능을 사용합니다.
_ECAT_Slave_CSP_Start_ECAM_Set_ Parameters	SlaveMotionCSP 명령, ECAM 관련 파라미터를 설정합니다.
_ECAT_Slave_CSP_Start_ECAM_Set_ DisEngage_and_SingleMove	SlaveMotionCSP 명령, ECAM 단축 동작을 설정합니다.
_ECAT_Slave_CSP_Start_ECAM_Disable	SlaveMotionCSP 명령, CAM 을 종료합니다.

함수의 명칭	설명
_ECAT_Slave_CSP_Start_ECAM_Get_Status	SlaveMotionCSP 명령, CAM 의 현재 상태를 확인합니다.
_ECAT_Slave_CSP_Start_ECAM_Set_MasterSource	SlaveMotionCSP 명령, 주축 소스를 설정합니다.
_ECAT_Slave_CSP_Start_ECAM_Set_EngageSource	SlaveMotionCSP 명령, 작동 소스를 설정합니다.
_ECAT_Slave_CSP_Start_ECAM_Set_CompensateSource	SlaveMotionCSP 명령, ECAM 보상 소스를 설정합니다.
_ECAT_Slave_CSP_Start_ECAM_Set_Compensate_Parameters	SlaveMotionCSP 명령, ECAM 보상 파라미터를 설정합니다. CAM 작동 시 파라미터를 설정할 수 없습니다.
_ECAT_Slave_CSP_Start_ECAM_Table_Move	SlaveMotionCSP 명령, ECAM 수동으로 표 만들기 기능.
_ECAT_Slave_CSP_Start_ECAM_Velocity_Move	SlaveMotionCSP 명령, ECAM 속도 구역 표 만들기 기능.
_ECAT_Slave_CSP_Start_ECAM_Flying_Shears_Move	SlaveMotionCSP 명령, ECAM 자동 연속 절단 기능.
_ECAT_Slave_CSP_Start_ECAM_Intermittence_Print_Move	SlaveMotionCSP 명령, ECAM 인터벌 인쇄 기능.

9.1 _ECAT_Slave_CSP_Start_Move

9

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Start_Move (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 Dist, I32 Strvel, I32 ConstVel, I32 EndVel, F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

■ **목적**

MotionSlave CSP 명령, 단축 직선 동작.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Dist	I32	펄스 수	지정된 동작 stroke
StrVel	I32	펄스 수/초	동작 초기 속도
ConstVel	I32	펄스 수/초	동작 정상 속도
EndVel	I32	펄스 수/초	동작 완료 속도
TPhase1	F64	초	StartVel 에서 ConstVel 까지의 시간
TPhase2	F64	초	ConstVel 에서 EndVel 까지의 시간
Scurve	U16	옵션 단위	1 : T-curve (Default) 2 : S-curve
Abs_Rel	U16	옵션 단위	0: 상대적 변위 (Default) 1: 절대적 변위

■ **예제**

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0;
I32 Dist=12000000, Strvel=0, ConstVel =2000000, EndVel=0;
F64 TPhase1=0.1, TPhase2=0.1;
U16 Scurve=0, Abs_Rel=1;

Status = _ECAT_Slave_CSP_Start_Move (CardNo, AxisNo, SlotNo, Dist, Strvel,
ConstVel,
EndVel, Tacc, Tdec, Scurve, Abs_Rel);
```

9.2 _ECAT_Slave_CSP_Start_V_Move

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Start_V_Move (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Dir, I32 Strvel, I32 MaxVel, F64 Tacc, U16 Scurve)

■ **목적**

MotionSlave CSP 명령, 단축 정속 지속 동작.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Dir	U16	옵션 단위	동작의 방향. 0: (+) 방향; 1: (-) 방향
Strvel	I32	펄스 수/초	초기 속도의 파라미터
MaxVel	I32	펄스 수/초	최대 속도 파라미터
Tacc	F64	초	지정된 가속 시간
Scurve	U16	옵션 단위	1 : T-curve (Default) 2 : S-curve

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 Dir=1, Strvel=0, MaxVel=2000000;
F64 Tacc =0.1;
U16 Scurve=1, Abs_Rel=1;

Status = _ECAT_Slave_CSP_Start_V_Move (CardNo, AxisNo, SlotNo, Dir, Strvel, MaxVel, Tacc, Scurve, Abs_Rel);
```

9.3 _ECAT_Slave_CSP_Start_Arc_Move

9

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Start_Arc_Move (U16 CardNo, U16 *AxisNo, U16 *SlotNo,I32 *CenterPoint, F64 Angle , I32 Strvel, I32 ConstVel, I32 EndVel, F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

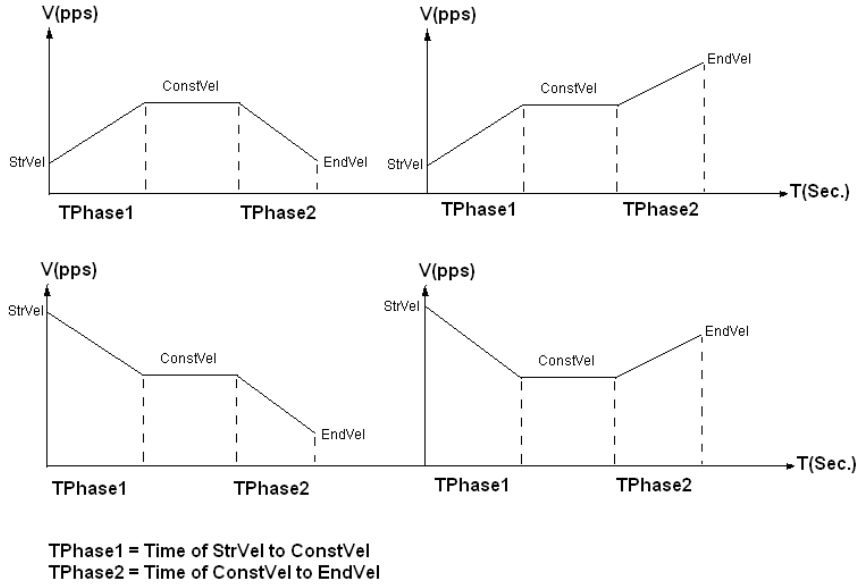
■ **목적**

MotionSlave CSP 명령, 2 축 원형 동작 (원 중심+각도).

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16*	번호 단위 배열	원형 보간을 실행하는데 사용되는 각종 Node 번호 ID 배열을 저장합니다 AxisNo Array[0] 첫 번째 세트의 Node 번호를 배열합니다 AxisNo Array[1] 두 번째 세트의 Node 번호를 배열합니다
SlotNo	U16*	번호 단위 배열	SlotID 번호 배열
CenterPoint	I32*	펄스 수	지정축에 상응하는 원 중심의 위치 X, Y
Angle	F64	도(°)	원형 각도 설정.
Strvel	I32	펄스 수/초	동작 초기 속도 파라미터
ConstVel	I32	펄스 수/초	동작 정상 속도 파라미터
EndVel	I32	펄스 수/초	동작 완료 속도 파라미터
TPhase1	F64	초	StartVel 에서 ConstVel 까지의 시간
TPhase2	F64	초	ConstVel 에서 EndVel 까지의 시간
Scurve	U16	옵션 단위	1 : T-curve (Default) 2 : S-curve
Abs_Rel	U16	옵션 단위	0: 상대적 변위 (Default) 1: 절대적 변위

■ 설명



TPhase1 = Time of StrVel to ConstVel
TPhase2 = Time of ConstVel to EndVel

그림 9.3.1 TPhase1 및 TPhase2 설명

■ 예제

```
U16 Status;
U16 CardNo=0, AxisNoArray[2]={1,2}, SlotID[2]={0, 0};
I32 CenterPoint[2] ={50000,50000};
F64 Angle=180, TPhase1=0.2, TPhase2=0.1;
I32 StrVel=0, ConstVel =50000, EndVel=20000;
U16 Scurve =1, Abs_Rel =0;

Status = _ECAT_Slave_CSP_Start_Arc_Move (CardNo, AxisNoArray, SlotID,
CenterPoint,
Angle,Strvel, ConstVel, EndVel, TPhase1, TPhase2, Scurve, Abs_Rel);
```

9.4 _ECAT_Slave_CSP_Start_Arc2_Move

9

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Start_Arc2_Move (U16 CardNo, U16 *AxisNo, U16 *SlotNo,I32 *EndPoint, F64 Angle, I32 Strvel, I32 ConstVel, I32 EndVel, F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

■ **목적**

MotionSlave CSP 명령, 2 축 원형 동작 (최종 좌표+각도).

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16*	번호 단위 배열	원형 보간을 실행하는데 사용되는 각종 Node 번호 ID 배열을 저장합니다 AxisNo Array[0] 첫 번째 세트의 Node 번호를 배열합니다 AxisNo Array[1] 두 번째 세트의 Node 번호를 배열합니다
SlotNo	U16*	번호 단위 배열	SlotID 번호 배열
EndPoint	I32*	펄스 수	지정축에 상응하는 최종 위치 X, Y
Angle	F64	도(°)	원형 각도 설정.
Strvel	I32	펄스 수/초	동작 초기 속도 파라미터
ConstVel	I32	펄스 수/초	동작 정상 속도 파라미터
EndVel	I32	펄스 수/초	동작 완료 속도 파라미터
TPhase1	F64	초	StartVel 에서 ConstVel 까지의 시간
TPhase2	F64	초	ConstVel 에서 EndVel 까지의 시간
Scurve	U16	옵션 단위	1 : T-curve (Default) 2 : S-curve
Abs_Rel	U16	옵션 단위	0: 상대적 변위 (Default) 1: 절대적 변위

■ 예제

```
U16 Status;
U16 CardNo=0, AxisNoArray[2]={1,2}, SlotID[2]={0, 0};
I32 EndPoint [2]= {100000,100000};
F64 Angle=180, TPhase1=0.2, TPhase2=0.1;
I32 StrVel=0, ConstVel =50000, EndVel=20000;
U16 Scurve =1, Abs_Rel =0;

Status = _ECAT_Slave_CSP_Start_Arc2_Move(CardNo, AxisNoArray, SlotID, EndPoint,
Angle, Strvel, ConstVel, EndVel, TPhase1, TPhase2, Scurve, Abs_Rel);
```


9.5 _ECAT_Slave_CSP_Start_Arc3_Move

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Start_Arc3_Move (U16 CardNo, U16 *AxisNo, U16 *SlotNo,I32 *CenterPoint, I32 *EndPoint, U16 Dir, I32 StrVel, I32 ConstVel, I32 EndVel, F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

■ 목적

MotionSlave CSP 명령, 2 축 원형 동작(원 중심+최종 좌표).

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16*	번호 단위 배열	원형 보간을 실행하는데 사용되는 각종 Node 번호 ID 배열을 저장합니다 AxisNo Array[0] 첫 번째 세트의 Node 번호를 배열합니다 AxisNo Array[1] 두 번째 세트의 Node 번호를 배열합니다
SlotNo	U16*	번호 단위 배열	SlotID 번호 배열
CenterPoint	I32*	펄스 수	지정축에 상응하는 원 중심의 위치 X, Y
EndPoint	I32*	펄스 수	지정축에 상응하는 최종 위치 X, Y
Dir	U16	옵션 단위	나선 원형 동작의 방향. 0: 시계 방향; 1: 시계 반대 방향
StrVel	I32	펄스 수/초	동작 초기 속도 파라미터
ConstVel	I32	펄스 수/초	동작 정상 속도 파라미터

명칭	데이터 타입	단위	설명
EndVel	I32	펄스 수/초	동작 완료 속도 파라미터
TPhase1	F64	초	StartVel 에서 ConstVel 까지의 시간
TPhase2	F64	초	ConstVel 에서 EndVel 까지의 시간
Scurve	U16	옵션 단위	1 : T-curve (Default) 2 : S-curve
Abs_Rel	U16	옵션 단위	0: 상대적 변위 (Default) 1: 절대적 변위

■ **예제**

```
U16 Status, CardNo=0, AxisNoArray[2]={1,2}, SlotID[2]={0, 0}, Dir=1, Scurve =1,
Abs_Rel =0;
I32 CenterPoint[2] = {50000,50000}, EndPoint[2] ={10000,100000};
I32 StrVel=0, ConstVel =50000, EndVel=20000;
F64 TPhase1=0.2, TPhase2=0.1;

Status = _ECAT_Slave_CSP_Start_Arc3_Move(CardNo, AxisNoArray, SlotID,
CenterPoint, EndPoint, Angle, Strvel, ConstVel, EndVel, TPhase1, TPhase2, Scurve,
Abs_Rel);
```

9.6 _ECAT_Slave_CSP_Start_Spiral_Move

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Start_Spiral_Move (U16 CardNo, U16 *AxisNo, U16 *SlotNo, I32 *CenterPoint, I32 Spiral_Interval, F64 Angle, I32 StrVel, I32 ConstVel, I32 EndVel, F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

■ **목적**

MotionSlave CSP 명령, 2 축 나선 동작 (원 중심+각도).

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16*	번호 단위 배열	원형 보간을 실행하는데 사용되는 각종 Node 번호 ID 배열을 저장합니다 AxisNo Array[0] 첫 번째 세트의 Node 번호를 배열합니다 AxisNo Array[1] 두 번째 세트의 Node 번호를 배열합니다
SlotNo	U16*	번호 단위 배열	SlotID 번호 배열
CenterPoint	I32*	펄스 수	지정축에 상응하는 원 중심의 위치 X, Y
Spiral_Interval	I32*	펄스 수	나선 간 상대적 위치의 거리
Angle	F64	수치 단위	나선 동작의 총 각도 (360 도 1 회전)
Strvel	I32	펄스 수/초	동작 초기 속도 파라미터
ConstVel	I32	펄스 수/초	동작 정상 속도 파라미터
EndVel	I32	펄스 수/초	동작 완료 속도 파라미터
TPhase1	F64	초	StartVel 에서 ConstVel 까지의 시간
TPhase2	F64	초	ConstVel 에서 EndVel 까지의 시간
Scurve	U16	옵션 단위	1 : T-curve (Default) 2 : S-curve
Abs_Rel	U16	옵션 단위	0: 상대적 변위 (Default) 1: 절대적 변위

9

■ 예제

```
U16 Status;
U16 CardNo=0, AxisNo[2]={1,2}, SlotID[2]={0, 0}, Scurve =1, Abs_Rel =0;
I32 CenterPoint[2] ={50000,50000}, Spiral_Interval = 5000;
I32 StrVel=0, ConstVel =50000, EndVel=20000;
F64 Angel, TPhase1 = 0.2, TPhase2 = 0.1;

Status = _ECAT_Slave_CSP_Start_Spiral_Move(CardNo, AxisNo, SlotID, CenterPoint,
Spiral_Interval, Angle , StrVel, ConstVel, EndVel, TPhase1, TPhase2, Scurve,
Abs_Rel);
```

9.7 _ECAT_Slave_CSP_Start_Spiral2_Move

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Start_Spiral2_Move (U16 CardNo, U16 *AxisNo, U16 *SlotNo, I32 *CenterPoint, I32 EndPoint, U16 Dir, U16 CycleNum, I32 StrVel, I32 ConstVel, I32 EndVel, F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

■ 목적

MotionSlave CSP 명령, 2 축 나선 동작 (최종좌표+회전수).

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16*	번호 단위 배열	원형 보간을 실행하는데 사용되는 각종 Node 번호 ID 배열을 저장합니다 AxisNo Array[0] 첫 번째 세트의 Node 번호를 배열합니다 AxisNo Array[1] 두 번째 세트의 Node 번호를 배열합니다
SlotNo	U16*	번호 단위 배열	SlotID 번호 배열
CenterPoint	I32*	펄스 수	지정축에 상응하는 원 중심의 위치 X, Y
EndPoint	I32*	펄스 수	지정축에 상응하는 최종 위치 X, Y
Dir	U16	옵션 단위	나선 원형 동작의 방향. 0: 시계 방향; 1: 시계 반대 방향
CycleNum	U16	수치 단위	나선 동작의 권회수
Strvel	I32	펄스 수/초	동작 초기 속도 파라미터

명칭	데이터 타입	단위	설명
ConstVel	I32	펄스 수/초	동작 정상 속도 파라미터
EndVel	I32	펄스 수/초	동작 완료 속도 파라미터
TPhase1	F64	초	StartVel 에서 ConstVel 까지의 시간
TPhase2	F64	초	ConstVel 에서 EndVel 까지의 시간
Scurve	U16	옵션 단위	1 : T-curve (Default) 2 : S-curve
Abs_Rel	U16	옵션 단위	0: 상대적 변위 (Default) 1: 절대적 변위

■ **예제**

```
U16 Status, CardNo=0, AxisNo[2]={1,2}, SlotID[2]={0, 0},Dir=0, Scurve =1, Abs_Rel =0,
U16 CycleNum=2;
I32 CenterPoint[2] ={50000,50000}, EndPoint [2] ={60000,100000};
I32 StrVel=0, ConstVel =50000, EndVel=20000;
F64 TPhase1=0.2, TPhase2=0.1;
Status = _ECAT_Slave_CSP_Start_Spiral2_Move(CardNo, AxisNo, SlotID, CenterPoint,
EndPoint, Dir, CycleNum, StrVel, ConstVel, EndVel, TPhase1, TPhase2, Scurve,
Abs_Rel);
```

9.8 _ECAT_Slave_CSP_Start_Sphere_Move

9

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Start_Sphere_Move (U16 CardNo, U16 *AxisNo, U16 *SlotNo, I32 * Target1Point, I32 Target2Point, I32 StrVel, I32 ConstVel, I32 EndVel, F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

■ **목적**

MotionSlave CSP 명령, 3 축 구형 동작, 세 점이 구형을 형성합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16*	번호 단위 배열	원형 보간을 실행하는데 사용되는 각종 Node 번호 ID 배열을 저장합니다 AxisNo Array[0] 첫 번째 세트의 Node 번호를 배열합니다 AxisNo Array[1] 두 번째 세트의 Node 번호를 배열합니다 AxisNo Array[2] 세 번째 세트의 Node 번호를 배열합니다
SlotNo	U16*	번호 단위 배열	SlotID 번호 배열
Target1Point	I32*	펄스 수 배열	Target1Point[0] curve 의 임의의 지점 X 위치 저장 Target1Point[1] curve 의 임의의 지점 Y 위치 저장 Target1Point[2] curve 의 임의의 지점 Z 위치 저장
Target2Point	I32*	펄스 수 배열	Target2Point[0] 최종점 X 위치 저장 Target2Point[1] 최종점 Y 위치 저장 Target2Point[2] 최종점 Z 위치 저장
Strvel	I32	펄스 수/초	동작 초기 속도 파라미터
ConstVel	I32	펄스 수/초	동작 정상 속도 파라미터
EndVel	I32	펄스 수/초	동작 완료 속도 파라미터
TPhase1	F64	초	StartVel 에서 ConstVel 까지의 시간
TPhase2	F64	초	ConstVel 에서 EndVel 까지의 시간
Scurve	U16	옵션 단위	1 : T-curve (Default) 2 : S-curve
Abs_Rel	U16	옵션 단위	0: 상대적 변위 (Default) 1: 절대적 변위

■ 예제

```
U16 Status, CardNo=0, AxisNo[2]={1,2}, SlotID[2]={0, 0}, Scurve =1, Abs_Rel =0;
I32 Target1Point [2] ={25000,50000,20000}, Target2Point [2] ={95000,110000,60000};
I32 StrVel=0, ConstVel =50000, EndVel=20000;
F64 TPhase1=0.2, TPhase2=0.1;
Status = _ECAT_Slave_CSP_Start_Sphere_Move (CardNo, AxisNo, SlotID,
Target1Point, Target2Point, StrVel, ConstVel, EndVel, TPhase1, TPhase2, Scurve,
Abs_Rel);
```


9.9 _ECAT_Slave_CSP_Start_Heli_Move

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Start_Heli_Move (U16 CardNo, U16 *AxisNo, U16 *SlotNo,I32 *CenterPoint, I32 Depth, I32 Pitch, U16 Dir, I32 Strvel, I32 ConstVel, I32 EndVel, F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

■ 목적

MotionSlave CSP 명령, 3 축 나선 동작.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16*	번호 단위 배열	원형 보간을 실행하는데 사용되는 각종 Node 번호 ID 배열을 저장합니다 AxisNo Array[0] 첫 번째 세트의 Node 번호를 배열합니다 AxisNo Array[1] 두 번째 세트의 Node 번호를 배열합니다 AxisNo Array[2] 세 번째 세트의 Node 번호를 배열합니다
SlotNo	U16*	번호 단위 배열	SlotID 번호 배열
CenterPoint	I32*	펄스 수	지정축에 상응하는 원 중심의 위치 X, Y
Depth	I32	펄스 수	지정축에 상응하는 상대적 위치의 깊이 (전체 Z 방향의 높이)
Pitch	I32	펄스 수	지정된 두 개의 상대적 나선간의 높이
Dir	U16	옵션 단위	나선 원형 동작의 방향. 0: 시계 방향; 1: 시계 반대 방향

명칭	데이터 타입	단위	설명
Strvel	I32	펄스 수/초	동작 초기 속도 파라미터
ConstVel	I32	펄스 수/초	동작 정상 속도 파라미터
EndVel	I32	펄스 수/초	동작 완료 속도 파라미터
TPhase1	F64	초	StartVel 에서 ConstVel 까지의 시간
TPhase2	F64	초	ConstVel 에서 EndVel 까지의 시간
Scurve	U16	옵션 단위	1 : T-curve (Default) 2 : S-curve
Abs_Rel	U16	옵션 단위	0: 상대적 변위 (Default) 1: 절대적 변위

■ **예제**

```
U16 Status, CardNo=0, AxisNoArray[2]={1,2}, SlotID[2]={0, 0}, Dir=1, Scurve =1,
Abs_Rel =0;
I32 CenterPoint[2]= {50000,50000}, Depth =10000, Pitch = 20000;
I32 StrVel=0, ConstVel =50000, EndVel=20000;
F64 TPhase1=0.2, TPhase2=0.1;

Status = _ECAT_Slave_CSP_Start_Heli_Move (CardNo, AxisNoArray, SlotID,
CenterPoint , Depth, Pitch, Dir, Strvel, ConstVel, EndVel, TPhase1, TPhase2, Scurve,
Abs_Rel);
```

9.10 _ECAT_Slave_CSP_Start_Multiaxes_Move

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Start_Multiaxes_Move (U16 CardNo, U16 AxisNum, U16 *AxisArray, U16 *SlotArray, I32 *DistArray, I32 Strvel, I32 ConstVel, I32 EndVel, F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

■ **목적**

MotionSlave CSP 명령, 다축 직선 동작.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNum	U16	수치 단위	관련 축수
AxisArray	U16*	번호 단위 배열	Node 번호 ID 배열, 수량은 AxisNum 입니다 AxisNo Array[0] 첫 번째 세트의 Node 번호를 배열합니다 AxisNo Array[1] 두 번째 세트의 Node 번호를 배열합니다 ….
SlotArray	U16*	번호 단위 배열	SlotID 번호 배열, 수량은 AxisNum 입니다
DistArrary	I32*	펄스 수 배열	각 축이 실행해야 할 stroke 배열, 수량은 AxisNum 와 동일합니다
Strvel	I32	펄스 수/초	동작 초기 속도 파라미터
ConstVel	I32	펄스 수/초	동작 정상 속도 파라미터
EndVel	I32	펄스 수/초	동작 완료 속도 파라미터
TPhase1	F64	초	StartVel 에서 ConstVel 까지의 시간
TPhase2	F64	초	ConstVel 에서 EndVel 까지의 시간
Scurve	U16	옵션 단위	1 : T-curve (Default) 2 : S-curve
Abs_Rel	U16	옵션 단위	0: 상대적 변위 (Default) 1: 절대적 변위

■ **예제**

```
U16 Status, CardNo=0, AxisNum =2, AxisArray [2]={1,2}, SlotArray [2]={0, 0};
U16 Scurve =1, Abs_Rel =0;
I32 DistArrary [2]= {100000,200000}, StrVel=0, ConstVel =50000, EndVel=20000;
F64 TPhase1=0.2, TPhase2=0.1;

Status = _ECAT_Slave_CSP_Start_Multiaxes_Move (CardNo, AxisNum, AxisArray,
SlotArray, DistArrary, Strvel, ConstVel, EndVel, TPhase1, TPhase2, Scurve, Abs_Rel);
```

9.11 _ECAT_Slave_CSP_Start_Msbrline_Move

9

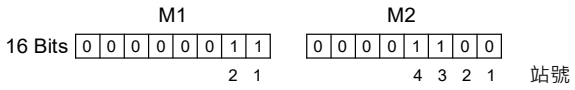
■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Start_Msbrline_Move (U16 CardNo, U16 AxisNum, U16 *AxisArray, U16 *SlotArray, U16 ArcNodeBit, I32 *Target1Point, I32 *Target2Point , U16 Mode, F64 Parameter, F64 ArcAngle1, F64 ArcAngle2, F64 SpeedRatio, I32 Strvel, I32 ConstVel, I32 EndVel, F64 TPhase1, F64 TPhase2, U16 Scurve, U16 Abs_Rel)

■ 목적

MotionSlave CSP 명령, 다축 3 점 고정 위치 평활 동작.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNum	U16	번호 단위	관련 축수
AxisArray	U16*	번호 단위 배열	Node 번호 ID 배열, 수량은 AxisNum 입니다 AxisNo Array[0] 첫 번째 세트의 Node 번호를 배열합니다 AxisNo Array[1] 두 번째 세트의 Node 번호를 배열합니다 ….
SlotArray	U16*	번호 단위 배열	SlotID 배열, 수량은 AxisNum 와 동일합니다
ArcNodeBit	U16	수치 단위	링크된 동작이 Arc(G02 / G03)인지 판단하고, Node 번호를 정의합니다 그림 9.11.1 과 같이 여러 개의 M1 및 M2 조합으로 라인 세그먼트를 분리합니다  M1 라인 세그먼트가 Node 번호 3&4 의 Arc 동작에서, M2 라인 세그먼트가 Node 번호 1&2 의 Arc 동작에서 완료될 경우, 해당 값은 0x30C 가 되어야 합니다
TargetPoint1	I32*	펄스 수 배열	첫 번째 목표 지점 배열, 수량은 AxisNum 와 동일합니다. M1 이 직선일 때 M1 의 최종 지점과 같습니다. M1 이 Arc 일 때 Arc 축에 상응하는 곳에 원 중심을 입력하고, 나머지 축에 시작 지점과 동일한 위치를 입력합니다

명칭	데이터 타입	단위	설명
TargetPoint2	I32*	펄스 수 배열	두 번째 목표 지점 배열, 수량은 AxisNum 와 동일합니다. M2 이 직선일 때 M2 의 최종 지점과 같습니다. M2 이 Arc 일 때 Arc 축에 상응하는 곳에 원 중심을 입력하고, 나머지 축에 첫 번째 목표와 동일한 위치를 입력합니다
Mode	U16	옵션 단위	0: M1, M2 모두 직선입니다. 1: M1 은 직선, M2 는 원형입니다. 2: M1 은 원형, M2 는 직선입니다. 3: M1, M2 는 원형입니다
Parameter	F64	수치 단위	stroke 오차량, 오차란 첫 번째 목표 지점과의 최단 거리를 의미합니다. 즉, M1 과 M2 가 코너를 도는 직각 또는 원형의 거리 입니다 (pulse). {해당 값이 0 인 경우 각 벡터 직선 연결 시 R 각 동작을 실행하지 않는 것과 같으므로 기기에 쉽게 진동이 발생합니다. 해당 값이 크면 합리적 범위까지 자동 조절됩니다 (시작점 및 두 번째 목표 지점 고정)}.
ArcAngle1	F64	도(°)	M1 이 Arc 일 때, M1 의 각도, M1 이 Line 일 경우 해당 값은 사용되지 않습니다
ArcAngle2	F64	도(°)	M2 이 Arc 일 때, M2 의 각도, M2 가 Line 일 경우 해당 값은 사용되지 않습니다
SpeedRatio	F64	수치 단위	SpdRatio = M1 및 M2 의 속도 비율, SpdM2 / SpdM1 과 같습니다 (해당 파라미터로 2 단 동작의 속도 변화를 조정합니다).
Strvel	I32	펄스 수/초	동작 초기 속도 파라미터
ConstVel	I32	펄스 수/초	동작 정상 속도 파라미터
EndVel	I32	펄스 수/초	동작 완료 속도 파라미터
TPhase1	F64	초	StartVel 에서 ConstVel 까지의 시간
TPhase2	F64	초	ConstVel 에서 EndVel 까지의 시간
Scurve	U16	옵션 단위	1 : T-curve (Default) 2 : S-curve

9

■ 설명예제

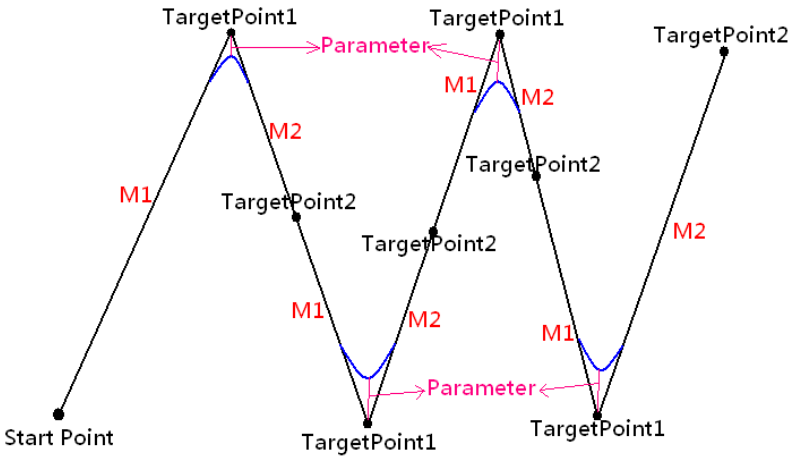


그림 9.11.1 Mode = 0 새로운 벡터 속도로 변경

■ 예제

```
U16 Status;
U16 CardNo=0, AxisNum =2, AxisArray [2]={1,2}, SlotArray [2]={0, 0}, ArcNodeBit = 0;
U16 Mode =1, Scurve =1, Abs_Rel =0;
I32 TargetPoint1 [2]= {100000,200000}, TargetPoint2 [2]= {100000,200000};
I32 StrVel=0, ConstVel =50000, EndVel=20000;
F64 Parameter = 2 , ArcAngle1 = 0, ArcAngle2 = 0, SpeedRatio = 1;
F64 TPhase1=0.2, TPhase2=0.1;

Status = _ECAT_Slave_CSP_Start_Msbrline_Move (CardNo, AxisNum, AxisArray,
SlotArray, ArcNodeBit, TargetPoint1, TargetPoint2, Mode, Parameter, ArcAngle1,
ArcAngle2, SpeedRatio, Strvel, ConstVel, EndVel, TPhase1, TPhase2, Scurve,
Abs_Rel);
```

9.12 _ECAT_Slave_CSP_Set_Gear

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Set_Gear (U16 CardNo, U16 AxisNo, U16 SlotNo, I16 Nummerator, I16 Denominator, I16 Enable)

■ **목적**

MotionSlave CSP 명령, 전자 기어비를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Nummerator	F64	수치 단위	Gear 분자 값 설정
Denominator	F64	수치 단위	Gear 분모 값 설정
Enable	I16	옵션 단위	0: Gear 기능을 종료합니다 1: Gear 기능 on

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I16Enable=1;
F64 Nummerator=1.0, Denominator=2.0;

Status = _ECAT_Slave_CSP_Set_Gear(CardNo, AxisNo, SlotNo, Nummerator,
Denominator, Enable);
```

9.13 _ECAT_Slave_CSP_Set_Softlimit

9

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Set_Softlimit (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 PosiLimit, I32 NegaLimit, U16 Mode)

■ 목적

MotionSlave CSP 명령, 소프트웨어 limit 를 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
PosiLimit	I32	수치 단위	소프트웨어의 (+) limit 값 설정
NegaLimit	I32	수치 단위	소프트웨어의 (-) limit 값 설정
Mode	U16	옵션 단위	1: limit 에 도달한 후 즉시 정지합니다 2: limit 도달 후 감속 정지 (감속 시간은 0.01 Sec 로 고정)

■ 예제

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I16 PosiLimit =1000000, NegaLimit =-1000000, Mode =1;

Status = _ECAT_Slave_CSP_Set_Softlimit (CardNo, AxisNo, SlotNo, PosiLimit, NegaLimit, Mode);
```

9.14 _ECAT_Slave_CSP_TargetPos_Change

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_TargetPos_Change (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 NewTargetCmd)

■ **목적**

MotionSlave CSP 명령, 새로운 목표 위치 설정.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
NewTargetCmd	I32	펄스 수	대체할 새로운 위치 파라미터

■ **예제**

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0;
I32 NewTargetCmd =2000000;

Status=_ECAT_Slave_CSP_TargetPos_Change (CardNo, AxisNo, SlotNo,
NewTargetCmd);
```

9.15 _ECAT_Slave_CSP_Velocity_Change

9

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Velocity_Change (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 NewTargetSpd, F64 Tsec)

■ 목적

MotionSlave CSP 명령, 새로운 목표 속도를 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
NewTargetSpd	I32	펄스 수/초	대체할 새로운 속도 파라미터
Tsec	F64	초	속도 전환 시 지정한 가속 / 감속 시간

■ 예제

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0;
I32 NewTargetSpd =1000000;
F64 Tsec=0.1;

Status = _ECAT_Slave_CSP_Velocity_Change (CardNo, AxisNo, SlotNo,
NewTargetSpd,Tsec);
```

9.16 _ECAT_Slave_CSP_Feedrate_Overwrite

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Feedrate_Overwrite (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Mode, I32 NewSpeed, F64 Tsec)

■ **목적**

MotionSlave CSP 명령, 다기능 변속 함수.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Mode	U16	옵션 단위	0: Velocity _Change 와 동일하며, 실행되는 동작이 있을 경우 현재 동작의 벡터 속도를 변경합니다. 1: 현재 동작 여부와 상관 없이 실행 가능하며, 현재 동작 및 모든 후속 동작의 벡터 속도를 변경할 수 있습니다. 2: 현재 동작 여부와 상관 없이 실행 가능하고, 현재 동작 및 모든 후속 동작의 벡터 속도 비율을 변경할 수 있으며, 범위는 0% ~ 1000%입니다.
NewSpeed	I32	펄스 수/초	Mode=0, 1 일 때, 속도 파라미터를 대체하려고 합니다 Mode=2 일 때, 벡터 속도 비율값 범위는 0% ~ 1000%입니다
Tsec	F64	초	속도 전환 시 지정한 가속/감속 시간

9

■ 설명

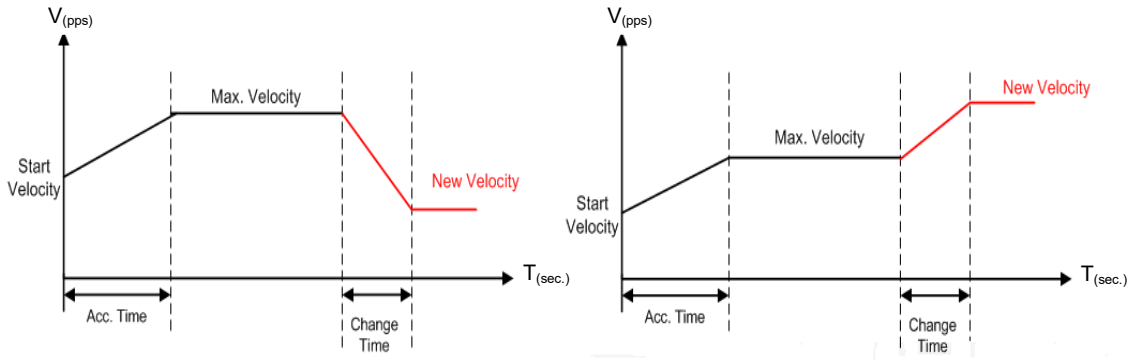


그림 9.16.1 Mode = 0 새로운 벡터 속도로 변경

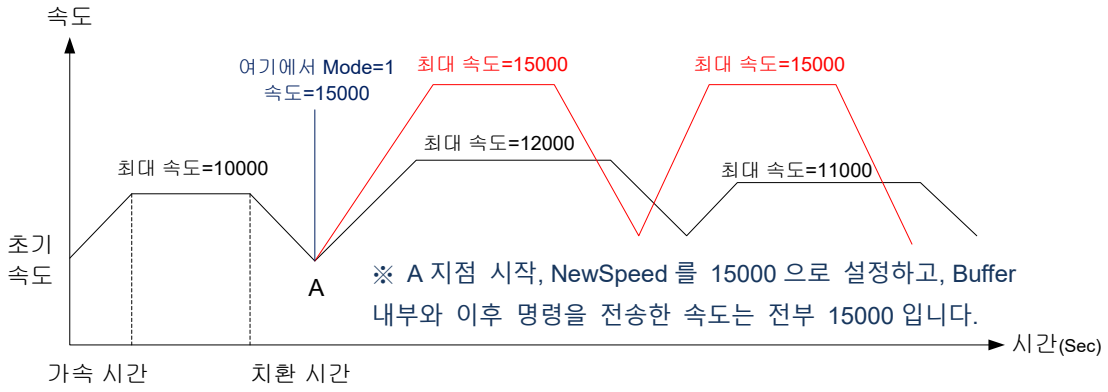


그림 9.16.2 Mode = 1 새로운 벡터 속도로 변경

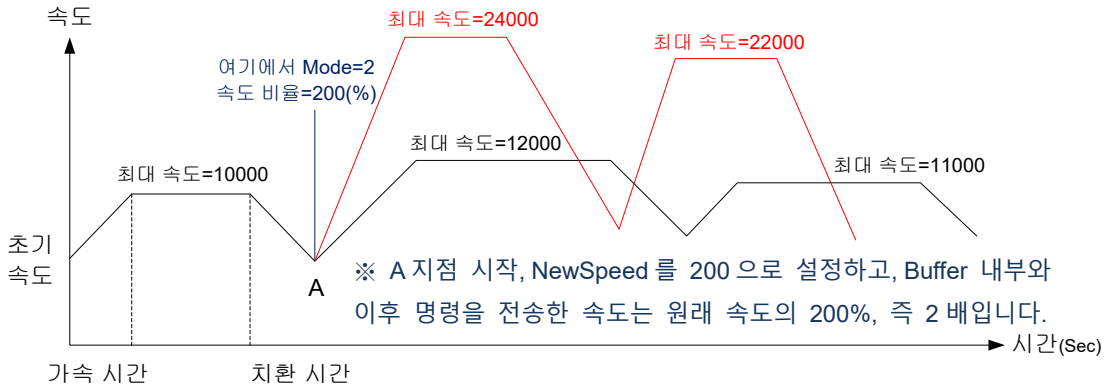


그림 9.16.3 Mode = 2 새로운 벡터 속도 비율 변경

■ **예제**

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0, Mode=0;
I32 NewSpeed =3000;
F64 Tsec=0.1;

Status = _ECAT_Slave_CSP_Feedrate_Overwrite (CardNo, AxisNo, SlotNo, Mode,
NewSpeed, Tsec);
```


9.17 _ECAT_Slave_CSP_Speed_Continue_Enable

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Speed_Continue_Enable (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ **목적**

MotionSlave CSP 명령, 속도 연속 기능 활성화 / 활성화 해제.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Enable	U16	옵션 단위	0: Disable 연속 속도 기능 1: Enable 연속 속도 기능

■ **예제**

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0, Enable=1;

Status = _ECAT_Slave_CSP_Speed_Continue_Enable (CardNo, AxisNo, SlotNo,
Enable);
```

9.18 _ECAT_Slave_CSP_Speed_Continue_Set_Mode

9

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Speed_Continue_Set_Mode (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Mode)

■ 목적

MotionSlave CSP 명령, 속도 연속 모드를 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Mode	U16	옵션 단위	0: 등가속 모드 1: 가속, 감속 모드 2: 최대 속도 모드

■ 설명

Mode 값은 0 입니다

설정 조건이 Dist 1000, MaxVel 20000, Tacc 와 Tdec 이 0.1 일 때.
원래의 가속 및 감속 시간 조건 하에 상기 조건은 Dist 의 이동 과정에서 MaxVel 의 값이 20000 에 도달할 수 없음을 나타냅니다. 따라서 mode 값이 0 으로 설정될 경우 동일한 비율로 MaxVel, Tacc, Tdec 를 감소시킵니다. 동등한 가속도로 완료됩니다. 그림 9.18.1 에서 알 수 있듯이 검은색이 원래 계획된 경로이고, 빨간색이 실제 경로입니다.

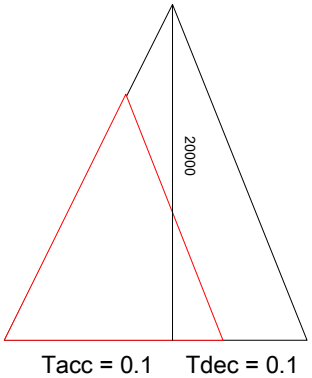


그림 9.18.1

Mode 값은 1 입니다

설정 조건이 Dist 1000, MaxVel 20000, Tacc 와 Tdec 이 0.1 일 때.
원래의 가속 및 감속 시간 조건 하에 상기 조건은 Dist 의 이동 과정에서 MaxVel 의 값이 20000 에 도달할 수 없음을 나타냅니다. 따라서 mode 값이 1 로 설정될 경우 Tacc 과 Tdec 는 계속해서 0.1 을 유지하지만 MaxVel 값은 자동으로 감소합니다.
그림 9.18.2 에서 알 수 있듯이 검은색이 원래 계획된 경로이고, 빨간색이 실제 경로입니다.

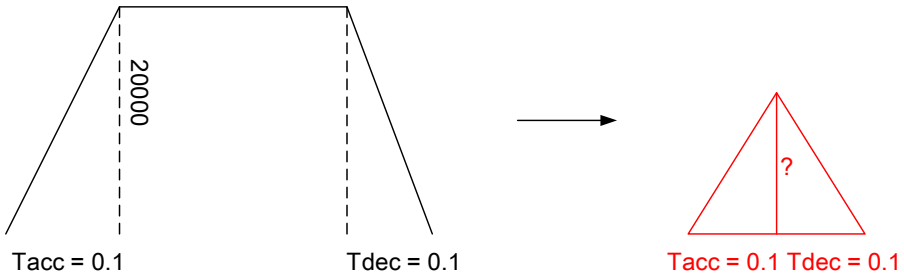


그림 9.18.2

Mode 값은 2 입니다

설정 조건이 Dist 1000, MaxVel 20000, Tacc 와 Tdec 이 0.1 일 때.
원래의 가속 및 감속 시간 조건 하에 상기 조건은 Dist 의 이동 과정에서 MaxVel 의 값이 20000 에 도달할 수 없음을 나타냅니다. 따라서 mode 값이 2 로 설정될 경우 MaxVel 값은 변하지 않지만 Tacc 과 Tdec 는 실제 상황에 따라 계속해서 변합니다.
그림 9.18.3 에서 알 수 있듯이 검은색이 원래 계획된 경로이고, 빨간색이 실제 경로입니다.

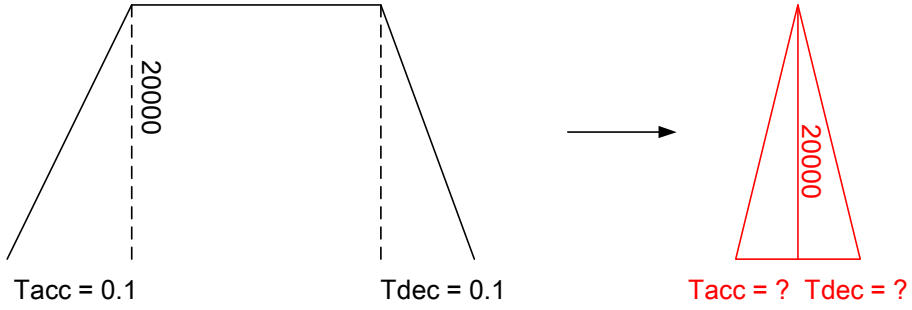


그림 9.18.3

9

■ 예제

```
U16Status;
U16 CardNo=16, AxisNo=1, SlotNo=0, Mode=1;

Status = _ECAT_Slave_CSP_Speed_Continue_Set_Mode (CardNo, AxisNo, SlotNo,
Mode);
```

9.19 _ECAT_Slave_CSP_Speed_Continue_Set_ Combine_ Ratio

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Speed_Continue_Set_Combine_Ratio (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Ratio)

■ 목적

MotionSlave CSP 명령, 연속된 속도 명령을 백분율로 합성하여 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Ratio	U16	수치 단위	합성 백분율 (0 ~ 100)

■ 설명

Ratio 값은 100 입니다

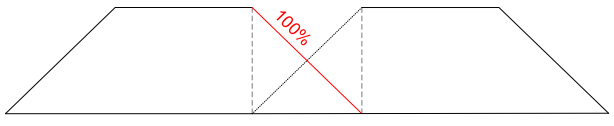


그림 9.19.1

Ratio 값은 50 입니다

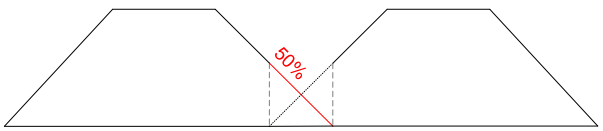


그림 9.19.2

■ 예제

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0, Ratio=100;

Status = _ECAT_Slave_CSP_Speed_Continue_Set_Combine_Ratio (CardNo, AxisNo,
SlotNo, Ratio);
```


9.20 _ECAT_Slave_CSP_Scurve_Rate

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Scurve_Rate (U16 CardNo, U16 AxisNo,
U16 SlotNo, U16 Ratio)

■ 목적

MotionSlave CSP 명령, Scurve 의 Tcurve 비율을 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Ratio	U16	수치 단위	가감속 구간을 중심점으로 T_Curve 모드 이전 / 이후의 백분율 (0 ~ 100)

■ 설명

가속 구간 중심점 A 와 감속 구간 중심점 B 를 시작점으로 설정합니다. scurve_rate = 60 은 가속 구간 이전 20%는 S_Curve 로 진행되고, 다음 60%는 T_Curve 로 진행되며, 이후 20%는 다시 S_Curve 로 진행됨을 의미합니다. 감속 구간도 마찬가지입니다. B 지점에서 Sd_Stop 명령을 전송하면 Sd Mode = 0 이 설정된 상황에서 감속 시간이 단축되고, C 지점에서 종료하게 됩니다.

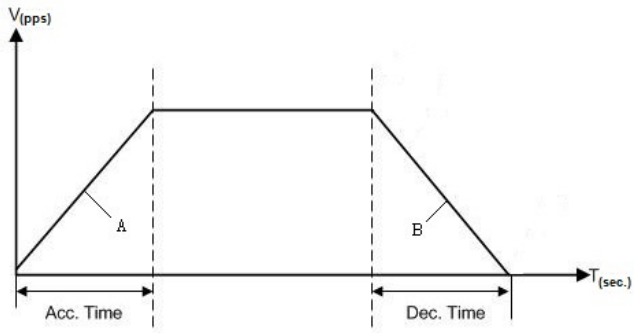


그림 9.20.1 scurve_rate 설정 설명

■ **예제**

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0;
U16 Ratio =100;

Status = _ECAT_Slave_CSP_Scurve_Rate (CardNo, AxisNo, SlotNo, Ratio);
```

9.21 _ECAT_Slave_CSP_Liner_Speed_Master

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Liner_Speed_Master (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Mode)

■ **목적**

MotionSlave CSP 명령, 최대 속도의 정의 모드를 설정합니다.

파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Mode	U16	옵션 단위	0: 벡터 속도 (Default 설정). 1: 주행 거리가 가장 먼 축의 최대 성분 속도.

■ 설명

Mode 파라미터 설정 표시

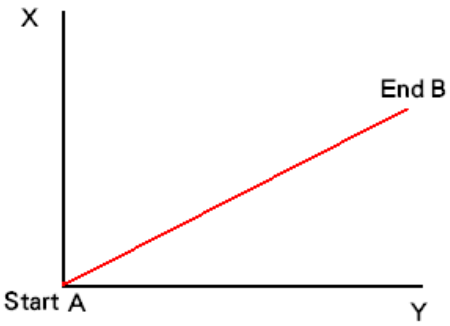


그림 9.21.1 Mode = 0 벡터 속도는 Start ~ End 입니다 (A□)

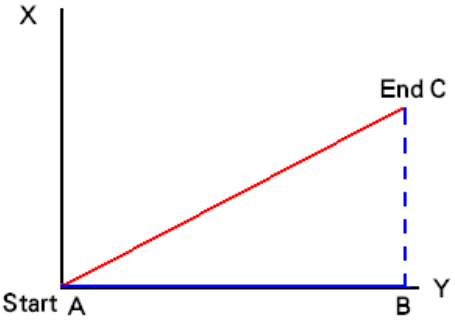


그림 9.21.2 Mode = 1 명령 전송 속도는 비교적 먼 축인 Y 축의 성분 속도이며(A□),
X 축의 성분 속도는 자동으로 산출됩니다(B□) ~ (A□)

■ 예제

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0, Mode=1;

Status = _ECAT_Slave_CSP_Liner_Speed_Master (CardNo, AxisNo, SlotNo, Mode);
```

9.22 _ECAT_Slave_CSP_Mask_Axis

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Mask_Axis (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Mode)

■ **목적**

SlaveMotionCSP 명령, 다축 또는 단축 명령에서 일부 몇 개의 축을 정지시켜도 다른 축에 영향을 미치지 않습니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNum	U16	수치 단위	정지한 축 수
AxisArray	U16*	수치 단위	정지한 Node 번호 ID 배열, 수량은 AxisNum 입니다 AxisNo Array[0] 첫 번째 세트의 Node 번호를 배열합니다 AxisNo Array[1] 두 번째 세트의 Node 번호를 배열합니다 ….
SlotArray	U16*	수치 단위	정지한 SlotID 번호 배열, 수량은 AxisNum 입니다

■ **예제**

```
U16 Status;
U16 CardNo=0, MoveAxisNum =3, MoveAxisArray[3]={1,2, 3}, MoveSlotArray[3]={0, 0, 0};
U16 StopAxisNum = 2, StopAxisArray[2]={2, 3}, StopSlotArray[2]={0, 0};
I32 DistArrary[3]= {100000,200000, 300000}, StrVel=0, ConstVel =50000, EndVel=20000;
F64 TPhase1=0.2, TPhase2=0.1;
U16 Scurve =1, Abs_Rel =0;

Status = _ECAT_Slave_CSP_Start_Multiaxes_Move(CardNo, MoveAxisNum, MoveAxisArray, MoveSlotArray, DistArrary, Strvel, ConstVel, EndVel, TPhase1, TPhase2, Scurve, Abs_Rel);
Status = _ECAT_Slave_CSP_Mask_Axis(CardNo, StopAxisNum, StopAxisArray, StopSlotArray)
```

9.23 _ECAT_Slave_CSP_Sync_Config

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Sync_Config (U16 CardNo, U16 AxisNum, U16 *AxisArray, U16 *SlotArray, U16 Enable)

■ 목적

SlaveMotionCSP 명령, 동시 작동이 필요한 축을 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNum	U16	수치 단위	동시 동작 축수를 설정합니다
AxisArray	U16*	번호 단위 배열	Node 번호 ID 배열, 수량은 AxisNum 입니다 AxisArray[0] 첫 번째 세트의 Node 번호를 배열합니다 AxisArray[1] 두 번째 세트의 Node 번호를 배열합니다 ….
SlotArray	U16*	번호 단위 배열	SlotID 번호 배열, 수량은 AxisNum 입니다
Enable	U16	옵션 단위	동시 이동을 사용합니다

■ 예제

```
U16 Status;
U16 CardNo = 16, AxisNum = 2, AxisArray[2] = {0, 1}, SlotArray[2] = {0, 0};
U16 Enable = 1;

Status = _ECAT_Slave_CSP_Sync_Config (CardNo, AxisNum, AxisArray, SlotArray, Enable);
```

9.24 _ECAT_Slave_CSP_Sync_Move

9

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Sync_Move (U16 CardNo)

■ **목적**

SlaveMotionCSP 명령, Config 에서 설정한 축을 동시 작동합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호

■ **예제**

```
U16 Status;
U16 CardNo=16;

Status = _ECAT_Slave_CSP_Sync_Move (CardNo);
```

9.25 _ECAT_Slave_CSP_Start_Mabrline_Move

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Start_Mabrline_Move (U16 CardNo, U16 AxisNum, U16 *AxisArray, U16 *SlotArray, I32 *Target1Point, I32 *Target2Point, I32 StrVel, I32 First_ConstVel, I32 Second_ConstVel, I32 EndVel, F64 Tacc_Step1, F64 Tacc_Step2, U16 Abs_Rel)

■ 목적

SlaveMotionCSP 명령, 다축 3 점 SCurve 평활 보간 동작.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNum	U16	수치 단위	동시 동작 축수를 설정합니다
AxisArray	U16*	번호 단위 배열	Node 번호 ID 배열, 수량은 AxisNum 입니다 AxisArray[0] 첫 번째 세트의 Node 번호를 배열합니다 AxisArray[1] 두 번째 세트의 Node 번호를 배열합니다 ….
SlotArray	U16*	번호 단위 배열	SlotID 번호 배열, 수량은 AxisNum 입니다
Enable	U16	옵션 단위	동시 이동을 사용합니다
Target1Point	I32*	펄스 수 배열	첫 번째 지점 펄스 수 배열, 수량은 AxisNum 와 동일합니다 Target1Point[0] 첫 번째 축의 첫 번째 지점 위치를 배열합니다 Target1Point[1] 두 번째 축의 첫 번째 지점 위치를 배열합니다 ….
Target2Point	I32*	펄스 수 배열	두 번째 지점 펄스 수 배열, 수량은 AxisNum 와 동일합니다 Target2Point[0] 첫 번째 축의 첫 번째 지점 위치를 배열합니다 Target2Point[1] 두 번째 축의 첫 번째 지점 위치를 배열합니다 ….
StrVel	I32	펄스 수/초	동작 초기 속도 파라미터
First_ConstVel	I32	펄스 수/초	시작점부터 첫 번째 지점까지의 동작 정상 속도 파라미터

명칭	데이터 타입	단위	설명
Second_ConstVel	I32	펄스 수/초	첫 번째 지점부터 두 번째 지점까지의 동작 정상 속도 파라미터
EndVel	I32	펄스 수/초	동작 완료 속도 파라미터
Tacc_Step1	F64	초	First_ConstVel 부터 0 까지의 소요 시간
Tacc_Step2	F64	초	Second_ConstVel 부터 0 까지의 소요 시간
Abs_Rel	U16	옵션 단위	0: 상대적 변위 (Default) 1: 절대적 변위

■ **예제**

```
U16 Status;
U16 CardNo = 16, AxisNum = 2, AxisArray[2] = {0, 1}, SlotArray[2] = {0, 0};
I32 Target1Point[2] = {20000, 40000};
I32 Target2Point[2] = {40000, 80000};
I32 StrVel = 0, First_ConstVel = 100000, Second_ConstVel = 200000, EndVel = 0;
F64 Tacc_Step1 = 0.1, TaccStep2 = 0.1;
U16 Abs_Rel = 1;

Status = _ECAT_Slave_CSP_Start_Mabrline_Move (CardNo, AxisNum, AxisArray,
SlotArray, Target1Point, Target2Point, StrVel, First_ConstVel, Second_ConstVel,
EndVel,
Tacc_Step1, Tacc_Step2, Abs_Rel);
```

9.26 _ECAT_Slave_CSP_Start_2Segment_Move

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Start_2Segment_Move (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 SegMode, I32 Dist, I32 Dist2, I32 StrVel, I32 MaxVel, I32 MaxVel2, I32 EndVel, F64 Tacc, F64 Tsec, F64 Tdec, U16 scurve, U16 Abs_Rel)

■ 목적

SlaveMotionCSP 명령, 단축 연속 2 단 직선 동작

SegMode: Segment 모드

0: 첫 번째 구간 거리 이동이 완료되어야 Tsec 가 / 감속을 통해 두 번째 구간 거리에 도달

1: 첫 번째 구간 거리 이동 중에 Tsec 가 / 감속을 통해 두 번째 구간 거리로 연결

Dist: 첫 번째 구간 이동 거리

Dist2: 두 번째 구간 이동 거리

StrVel: 첫 번째 구간 stroke 초기 속도

MaxVel: 첫 번째 구간 stroke 에 설정한 최대 속도

MaxVel2: 두 번째 구간 stroke 에 설정한 최대 속도

EndVel: 두 번째 구간 stroke 최종 속도

Tacc: 첫 번째 구간 stroke 가속 시간

Tsec: 첫 번째 구간 stroke 에서 두 번째 stroke 로 전환되는 가 / 감속 시간

Tdec: 두 번째 구간 stroke 감속 시간

SCurve :

1 : TCurve

2 : SCurve

IsAbs: 상대 / 절대 이동.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	축 번호
SlotNo	U16	번호 단위	Node 번호
SegMode	U16	옵션 단위	Segment 모드
Dist	I32	옵션 단위	1 단 이동 거리
Dist2	I32	옵션 단위	2 단 이동 거리
StrVel	I32	옵션 단위	첫 번째 구간 stroke 초기 속도
MaxVel	I32	옵션 단위	1 단 stroke 에 설정된 최대 속도
MaxVel2	I32	옵션 단위	2 단 stroke 에 설정된 최대 속도

명칭	데이터 타입	단위	설명
EndVel	I32	옵션 단위	두 번째 구간 stroke 최종 속도
Tacc	F64	옵션 단위	첫 번째 구간 stroke 가속 시간
Tsec	F64	옵션 단위	1 단 stroke 를 2 단 stroke 로 전환하는 가속 / 감속 시간
Tdec	F64	옵션 단위	두 번째 구간 stroke 감속 시간
scurve	U16	옵션 단위	1: Tcurve; 2: SCurve
Abs_Rel	U16	옵션 단위	상대 / 절대 이동

■ **예제**

```
U16 Status;
U16 CardNo = 16, AxisNo=0, SlotNo=0, SegMode=0, scurve=1, Abs_Rel=0;
I32 Dist=0, Dist2=0, StrVel=0, MaxVel=0, MaxVel2=0, EndVel=0;
F64 Tacc=0, Tsec=0, Tdec=0;

Status = _ECAT_Slave_CSP_Start_2Segment_Move (CardNo, AxisNo, SlotNo,
SegMode,
Dist, Dist2, StrVel, MaxVel, MaxVel2, EndVel, Tacc, Tsec, Tdec, Scurve, Abs_Rel);
```

9.27 _ECAT_Slave_CSP_Start_PVT_Move

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Start_PVT_Move (U16 CardNo, U16 NodeID, U16 SlotID, I32 DataCnt, I32 *TargetPos, I32 *TargetTime, I32 *TargetVel)

■ **목적**

SlaveMotionCSP 명령, 단축 PVT 동작을 실행합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	Node 번호 일련번호
SlotID	U16	번호 단위	SlotID 번호
DataCnt	I32	수치 단위	데이터 비트수
TargetPos	I32*	펄스 수	목표 위치
TargetTime	I32*	밀리초	목표 시간
TargetVel	I32*	펄스 수/초	목표 속도

■ 예제

```
U16 Status;
U16 CardNo=16, NodeID=7, SlotID=0;
I32 DataCnt=3, TargetPos[3]={0, 20000, 30000}, TargetTime[3]={0, 1000, 2000},
TargetVel[3]={0,11000, 0};
Status = _ECAT_Slave_CSP_Start_PVT_Move (CardNo, NodeID, SlotID, DataCnt,
TargetPos, TargetTime, TargetVel);
```


9.28 _ECAT_Slave_CSP_Start_PVTComplete_Move

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Start_PVTComplete_Move (U16 CardNo, U16 NodeID, U16 SlotID, I32 DataCnt, I32 *TargetPos, I32 *TargetTime, I32 StrVel, I32 EndVel);

■ 목적

SlaveMotionCSP 명령, 단축 PVT 동작을 실행합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	Node 번호 일련번호
SlotID	U16	번호 단위	SlotID 번호
DataCnt	I32	수치 단위	데이터 비트수
TargetPos	I32*	펄스 수	목표 위치
TargetTime	I32*	밀리초	목표 시간
StrVel	I32	펄스 수/초	초기 속도
EndVel	I32	펄스 수/초	동작 완료 속도

■ 예제

```
U16 Status;
U16 CardNo=16, NodeID=7, SlotID=0;
I32 DataCnt=4, TargetPos[4]={0, 20000, 30000, 40000}, TargetTime[4]={0, 4000,
10000, 15000}, StrVel=10000, EndVel=0;

Status = _ECAT_Slave_CSP_Start_PVTComplete_Move (CardNo, NodeID, SlotID,
DataCnt, TargetPos, TargetTime, StrVel, EndVel);
```

9.29 _ECAT_Slave_CSP_Virtual_Set_Enable

9

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Virtual_Set_Enable (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ 목적

SlaveMotionCSP 명령, 가상 위치를 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	축 번호
SlotNo	U16	번호 단위	Node 번호
Enable	U16	수치 단위	실행

■ 예제

```
U16 Status;
U16 CardNo = 16, AxisNo=0, SlotNo=0, Enable=1;

Status = _ECAT_Slave_CSP_Virtual_Set_Enable(CardNo, AxisNo, SlotNo, Enable);
```

9.30 _ECAT_Slave_CSP_Virtual_Set_Command

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Virtual_Set_Command (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 Command)

■ **목적**

SlaveMotionCSP 명령, 가상 위치를 설정하고, 현재 위치를 지정 위치로 설정합니다.
Command: 현재 위치의 값을 지정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	축 번호
SlotNo	U16	번호 단위	Node 번호
Command	I32	수치 단위	현재 위치의 값을 지정합니다

■ **예제**

```
U16 Status;
U16 CardNo = 16, AxisNo=0, SlotNo=0, Command=0;

Status = _ECAT_Slave_CSP_Virtual_Set_Command(CardNo, AxisNo, SlotNo,
Command);
```

9.31 _ECAT_Slave_CSP_Get_SoftLimit_Status

9

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Get_SoftLimit_Status (U16 CardNo, U16 NodeID, U16 SlotNo, U16 *Status)

■ **목적**

SlaveMotionCSP 명령, 현재 소프트웨어 limit 상태를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	Node 번호 일련번호
SlotNo	U16	번호 단위	SlotNo 번호
Status	U16*	수치 단위	현재 소프트웨어의 (+) limit 또는 (-) limit 에 도달했는지 확인합니다 (Bit0: (-) limit Bit1: (+) limit)

■ **예제**

```
U16 Status;
U16 CardNo=16, NodeID=7, SlotNo=0, Status=0;

Status = _ECAT_Slave_CSP_Get_SoftLimit_Status (CardNo, NodeID, SlotNo, &Status);
```

9.32 _ECAT_Slave_CSP_Pitch_Set_Interval

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Pitch_Set_Interval (U16 CardNo, U16 AxisNo, U16 Slot No, I32 Interval)

■ **목적**

SlaveMotionCSP 명령, 구간 보상의 구간 거리를 설정합니다.
InterVal: 구간 거리.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	축 번호
SlotNo	U16	번호 단위	Node 번호
Interval	I32	수치 단위	구간 거리

■ **예제**

```
U16 Status;
U16 CardNo = 16, AxisNo=0, SlotNo=0, Interval=0

Status = _ECAT_Slave_CSP_Pitch_Set_Interval(CardNo, AxisNo, SlotNo, Interval);
```

9.33 _ECAT_Slave_CSP_Pitch_Set_Mode

9

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Pitch_Set_Mode (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Mode)

■ 목적

SlaveMotionCSP 명령, 구간 보상의 보상 모드를 설정합니다

Mode :

0: 단방향 보상 모드

1: 양방향 보상 모드.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	축 번호
SlotNo	U16	번호 단위	Node 번호
Mode	U16	수치 단위	모드

■ 예제

```
U16 Status;
U16 CardNo = 16, AxisNo=0, SlotNo=0, Mode=0;

Status = _ECAT_Slave_CSP_Pitch_Set_Mode(CardNo, AxisNo, SlotNo, Mode);
```

9.34 _ECAT_Slave_CSP_Pitch_Set_Org

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Pitch_Set_Org (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Dir, I32 OrgPos)

■ 목적

SlaveMotionCSP 명령, 구간 보상의 원점 위치를 설정합니다

Dir :

0: (+) 방향 원점 위치를 설정합니다

1: (-) 방향 원점 위치를 설정합니다

OrgPos: 구간 보상의 시작 위치.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	축 번호
SlotNo	U16	번호 단위	Node 번호
Dir	U16	수치 단위	(+) (-) 방향 원점 위치를 설정합니다
OrgPos	I32	수치 단위	구간 보상의 시작 위치

■ 예제

```
U16 Status;
U16 CardNo = 16, AxisNo=0, SlotNo=0, Dir=0;
I32 OrgPos=0;

Status = _ECAT_Slave_CSP_Pitch_Set_Org(CardNo, AxisNo, SlotNo, Dir, OrgPos);
```

9.35 _ECAT_Slave_CSP_Pitch_Set_Rel_Table

9

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Pitch_Set_Rel_Table (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Dir, I32* Table, U16 Num)

■ 목적

SlaveMotionCSP 명령, 구간 보상 내부 각 구간의 상대 보상값을 설정합니다
Dir :
0: (+) 방향 보상값을 설정합니다
1: (-) 방향 보상값을 설정합니다
*Table: 각 구간의 보상값 배열, 배열 수량은 Num 설정값 이하일 수 없습니다
Num: 보상값 배열의 개수.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	축 번호
SlotNo	U16	번호 단위	Node 번호
Dir	U16	수치 단위	(+) (-) 방향 보상값 설정
Table	I32*	수치 단위	각 구간의 보상값 배열
Num	U16	수치 단위	보상값 배열의 개수

■ 예제

```
U16 Status;
U16 CardNo = 16, AxisNo=0, SlotNo=0, Dir=0;
I32 Table[3] = {0, 1, 2};

Status = _ECAT_Slave_CSP_Pitch_Set_Rel_Table (CardNo, AxisNo, SlotNo, Dir,
&Table, Num);
```

9.36 _ECAT_Slave_CSP_Pitch_Set_Abs_Table

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Pitch_Set_Abs_Table (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Dir, I32* Table, U16 Num)

■ 목적

SlaveMotionCSP 명령, 구간 보상 내부 각 구간의 절대 보상값을 설정합니다
Dir :
0: (+) 방향 보상값을 설정합니다
1: (-) 방향 보상값을 설정합니다
*Table: 각 구간의 보상값 배열, 배열 수량은 Num 설정값 이하일 수 없습니다
Num: 보상값 배열의 개수.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	축 번호
SlotNo	U16	번호 단위	Node 번호
Dir	U16	옵션 단위	(+) (-) 방향 보상값 설정
Table	I32*	옵션 단위	각 구간의 보상값 배열
Num	U16	옵션 단위	보상값 배열의 개수

■ 예제

```
U16 Status;
U16 CardNo = 16, AxisNo=0, SlotNo=0, Dir=0, Num=0;
I32 Table[3]={0, 1, 2};

Status = _ECAT_Slave_CSP_Pitch_Set_Abs_Table (CardNo, AxisNo, SlotNo, Dir,
&Table, Num);
```

9.37 _ECAT_Slave_CSP_Pitch_Set_Enable

9

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Pitch_Set_Enable (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ 목적

SlaveMotionCSP 명령, 구간 보상 기능을 사용합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
BitNum	U16	번호 단위	Output 지점 번호

■ 예제

```
U16 Status;
U16 CardNo = 16, AxisNo=0, SlotNo=0, Enable=1

Status = _ECAT_Slave_CSP_Pitch_Set_Enable(CardNo, AxisNo, SlotNo, Enable);
```

9.38 _ECAT_Slave_CSP_Start_ECAM_Set_Parameters

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Start_ECAM_Set_Parameters (U16 CardNo, U16 ECAMNo, I32 Gear_numerator, I32 Gear_denominator, F64 Slave_mm_PUU, F64 Master_mm_Pulse, U16 Engage_Mode, I32 Engage_PreLed, I32 Cycle_PreLead, I32 DisEngage_Position, U16 DI_DisEngage, U16 DisEngage_Mode, U16 DisEngage_TurnOff)

■ **목적**

SlaveMotionCSP 명령, ECAM 관련 파라미터를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
ECAMNo	U16	번호 단위	ECAM 사용 번호 (0 ~ 4)
Gear_numerator	I32	수치 단위	기어비 (분자)
Gear_denominator	I32	수치 단위	기어비 (분모)
Slave_mm_PUU	F64	수치 단위	CAM 1mm 에 상응하는 Pulse 수
Master_mm_Pulse	F64	수치 단위	주축 1mm 에 상응하는 Pulse 수
Engage_Mode	U16	옵션 단위	작동 시기, 0: 즉각 작동, 1: DI
Engage_PreLed	I32	수치 단위	CAM 작동 후 주축의 전위량을 대기합니다
Cycle_PreLead	I32	수치 단위	CAM 분리 후 주축의 거리를 대기합니다
DisEngage_Position	I32	수치 단위	분리 시작 위치 (주축에서 해당 위치 CAM 축까지 이동한 후 분리 시작)
DI_DisEngage	U16	옵션 단위	0: DI 분리를 사용하지 않음, 1: DI Control
DisEngage_Mode	U16	옵션 단위	0: 분리하지 않음 1: 분리 시작 위치를 벗어난 후 정지 구간으로 진입합니다 2: 분리 시작 위치를 벗어난 후 CAM 분리 이후의 대기 거리 구간으로 진입합니다 (Cycle_PreLead) 3: 분리 시작 위치를 벗어난 후 단축 동작으로 연결됩니다 비고: DI_DisEngage 가 DI Control=> DisEngage_Mode 를 사용하여 1 과 2 를 동일하게 설정할 경우, 정지 구간으로 진입합니다
DisEngage_TurnOff	U16	옵션 단위	0: 분리 후 CAM 을 off 하지 않음, 1: 분리 후 CAM off

9

■ 예제

```
U16 Status;
U16 CardNo = 16, ECAMNo =0, Engage_Mode = 0, DI_DisEngage = 0, DisEngage_Mode = 0
DisEngage_TurnOf = 0
I32 Gear_numerator = 10, Gear_denominator=128, Engage_PreLed = 2000,
Cycle_PreLead = 10000, DisEngage_Position = 34284
F64 Slave_mm_PUU = 53.052, Master_mm_Pulse = 12.732

Status = _ECAT_Slave_CSP_Start_ECAM_Set_Parameters (CardNo, ECAMNo,
Gear_numerator, Gear_denominator, Slave_mm_PUU, Master_mm_Pulse, Engage_Mode,
Engage_PreLed, Cycle_PreLead, DisEngage_Position, DI_DisEngage, DisEngage_Mode,
DisEngage_TurnOff);
```

9.39 _ECAT_Slave_CSP_Start_ECAM_Set_DisEngage_and_SingleMove

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Start_ECAM_Set_DisEngage_and_SingleMove (U16 CardNo, U16 ECAMNo, I32 Dist, I32 ConstVel, I32 EndVel, F64 Tacc, F64 Tdec, U16 Abs)

■ 목적

SlaveMotionCSP 명령, ECAM 단축 동작을 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
ECAMNo	U16	번호 단위	ECAM 사용 번호 (0 ~ 4)
Dist	I32	수치 단위	거리
ConstVel	I32	수치 단위	최대 속도
EndVel	I32	수치 단위	최종 속도
Tacc	F64	수치 단위	가속 시간 (초)
Tdec	F64	수치 단위	감속 시간 (초)
Abs	U16	옵션 단위	0: 상대적 동작 1: 절대적 동작

■ 예제

```
U16 Status;
U16 CardNo = 16, ECAMNo =0, Abs = 0
I32 Dist=20000, ConstVel =30000, EndVel = 0
F64 Tacc = 0.5, Tdec = 0.1

Status = _ECAT_Slave_CSP_Start_ECAM_Set_DisEngage_and_SingleMove (CardNo,
ECAMNo, Dist, ConstVel, EndVel, Tacc, Tdec, Abs);
```


9.40 _ECAT_Slave_CSP_Start_ECAM_Disable

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Start_ECAM_Disable (U16 CardNo, U16 ECAMNo, U16 Mode)

■ 목적

SlaveMotionCSP 명령, CAM 을 종료합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
ECAMNo	U16	번호 단위	ECAM 사용 번호 (0 ~ 4)
Mode	U16	옵션 단위	0: 즉각 off, 1: CAM 이 해당 회전을 완료하면 off

■ 예제

```
U16 Status;
U16 CardNo = 16, ECAMNo =0, Mode = 0

Status = _ECAT_Slave_CSP_Start_ECAM_Disable (CardNo, ECAMNo, Mode);
```

9.41 _ECAT_Slave_CSP_Start_ECAM_Get_Status

9

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Start_ECAM_Get_Status (U16 CardNo, U16 ECAMNo, U16 *Status)

■ 목적

SlaveMotionCSP 명령, CAM 의 현재 상태를 확인합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
ECAMNo	U16	번호 단위	ECAM 사용 번호 (0 ~ 4)
Status	U16*	옵션 단위	0: 상태 off 1: 전위 상태 2: 작동 상태 3: 상태 정지

■ 예제

```
U16 Status;
U16 CardNo = 16, ECAMNo =0, ECAMStatus

Status = _ECAT_Slave_CSP_Start_ECAM_Get_Status (CardNo, ECAMNo, &
ECAMStatus);
```

9.42 _ECAT_Slave_CSP_Start_ECAM_Set _MasterSource

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Start_ECAM_Set_MasterSource (U16 CardNo, U16 ECAMNo, U16 MasterNodeID, U16 MasterSlotID, U16 Master_Source)

■ 목적

SlaveMotionCSP 명령, 주축 소스를 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
ECAMNo	U16	번호 단위	ECAM 사용 번호 (0 ~ 4)
MasterNodeID	U16	번호 단위	NodeID
MasterSlotID	U16	번호 단위	SlotID
Master_Source	U16	옵션 단위	0: follow cmd 1: follow pos

■ 예제

```
U16 Status;
U16 CardNo = 16, ECAMNo =0, MasterNodeID=6, MasterSlotID=0, Master_Source=0;

Status = _ECAT_Slave_CSP_Start_ECAM_Set_MasterSource (CardNo, ECAMNo,
MasterNodeID, MasterSlotID, Master_Source);
```

9.43 _ECAT_Slave_CSP_Start_ECAM_Set _EngageSource

9

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Start_ECAM_Set_EngageSource (U16 CardNo, U16 ECAMNo, U16 SourceType, U16 NodeID, U16 SlotID, U16 SourceNo)

■ 목적

SlaveMotionCSP 명령, 작동 소스를 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
ECAMNo	U16	번호 단위	ECAM 사용 번호 (0 ~ 4)
SourceType	U16	옵션 단위	0: 모듈 DI, 1: Local DI, 2: 모듈 Latch, 3: Local Latch
NodeID	U16	번호 단위	NodeID
SlotID	U16	번호 단위	SlotID
SourceNo	U16	번호 단위	SourceNo (0 ~ 15)

■ 예제

```
U16 Status;
U16 CardNo = 16, ECAMNo =0, SourceType =0, NodeID =5, SlotID =0, SourceNo =0;

Status = _ECAT_Slave_CSP_Start_ECAM_Set_EngageSource (CardNo, ECAMNo,
SourceType, NodeID, SlotID, SourceNo);
```

9.44 _ECAT_Slave_CSP_Start_ECAM_Set _CompensateSource

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Start_ECAM_Set_CompensateSource (U16 CardNo, U16 ECAMNo, U16 SourceType, U16 NodeID, U16 SlotID, U16 SourceNo)

■ 목적

SlaveMotionCSP 명령, ECAM 보상 소스를 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
ECAMNo	U16	번호 단위	ECAM 사용 번호 (0 ~ 4)
SourceType	U16	옵션 단위	0: 모듈 DI, 1: Local DI, 2: 모듈 Latch, 3: Local Latch
NodeID	U16	번호 단위	NodeID
SlotID	U16	번호 단위	SlotID
SourceNo	U16	번호 단위	SourceNo (0 ~ 15)

■ 예제

```
U16 Status;
U16 CardNo = 16, ECAMNo =0, SourceType =0, NodeID =5, SlotID =0, SourceNo =0;

Status = _ECAT_Slave_CSP_Start_ECAM_Set_CompensateSource (CardNo, ECAMNo,
SourceType, NodeID, SlotID, SourceNo);
```

9.45 _ECAT_Slave_CSP_Start_ECAM_Set _Compensate_Parameters

9

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Start_ECAM_Set_Compensate_Parameters (U16 CardNo, U16 ECAMNo, I32 TargetPos, I32 Max_Compensate_Ratio, I32 MaskRatio, F64 CompensateTime)

■ 목적

SlaveMotionCSP 명령, ECAM 보상 파라미터를 설정합니다. CAM 작동 시 파라미터를 설정할 수 없습니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
ECAMNo	U16	번호 단위	ECAM 사용 번호 (0 ~ 4)
TargetPos	I32	수치 단위	맞출 위치
Max_Compensate_Ratio	I32	수치 단위	단위(ms) 시간 동안 최대 허용 수정률(%), 0 ~ 100% 매회 허용하는 최대 수정 펄스양(C) 제한은 다음과 같습니다: |C| <= (Master_Pulse_P) X Max_ Compensate_Ratio %
MaskRatio	I32	수치 단위	마스크 설정(%), 0 ~ 95% 위치를 맞춘 후 주축의 펄스 수 증가가 마스크 거리(M)를 초과할 때까지 대기해야만 다음 위치 맞추기가 가능합니다. |M| >= (Master_Pulse_P) X MaskRatio %
CompensateTime	F64	수치 단위	필터 보상 시간 (ms)

■ 예제

```
U16 Status;
U16 CardNo = 16, ECAMNo =0;
I32 TargetPos = 10000, Max_Compensate_Ratio = 50, MaskRatio = 50,
CompensateTime = 100;

Status = _ECAT_Slave_CSP_Start_ECAM_Set_Compensate_Parameters (CardNo,
ECAMNo, TargetPos, Max_Compensate_Ratio, MaskRatio, CompensateTime);
```

9.46 _ECAT_Slave_CSP_Start_ECAM_Table_Move

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Start_ECAM_Table_Move (U16 CardNo, U16 ECAMNo, U16 NodeID, U16 SlotID, I32 RegionNum, I32 *DataArray, I32 Master_Pulse)

■ **목적**

SlaveMotionCSP 명령, ECAM 수동으로 표 만들기 기능.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
ECAMNo	U16	번호 단위	ECAM 사용 번호 (0 ~ 4)
NodeID	U16	번호 단위	NodeID
SlotID	U16	번호 단위	SlotID
RegionNum	I32	수치 단위	구역 수 (RegionNum + 1 비트 데이터 생성)
DataArray	I32*	수치 단위	위치 데이터 배열입니다 (단위: pulse)
Master_Pulse	I32	수치 단위	상응하는 CAM 이 1 회전할 때 주축의 Pulse 수

■ **예제**

```
U16 Status;
U16 CardNo = 16, ECAMNo =0, NodeID = 5, SlotID = 0;
I32 RegionNum = 7, DataArray[8]={0, 13413, 27524, 42421, 57578, 72475, 86586, 100000};
I32 Master_Pulse = 34284;

Status = _ECAT_Slave_CSP_Start_ECAM_Table_Move (CardNo, ECAMNo, NodeID, SlotID, DataNum, DataArray, Master_Pulse);
```

9.47 _ECAT_Slave_CSP_Start_ECAM_Velocity_Move

9

■ **포맷**

U16 PASCAL _ECAT_Slave_CSP_Start_ECAM_Velocity_Move (U16 CardNo, U16 ECAMNo, U16 NodeID, U16 SlotID, I32 RegionNum, I32 Percent[5], I32 SCurveRegion, I32 TotalLength, I32 Master_Pulse_P, U16 Construct_Mode)

■ **목적**

SlaveMotionCSP 명령, ECAM 속도 구역 표 만들기 기능.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
ECAMNo	U16	번호 단위	ECAM 사용 번호 (0 ~ 4)
NodeID	U16	번호 단위	NodeID
SlotID	U16	번호 단위	SlotID
RegionNum	I32	수치 단위	구역 수 (RegionNum + 1 비트 데이터 생성)
Percent[5]	I32	수치 단위	배열은 순서대로 대기 구간, 가속 구간, 등속 구간, 감속 구간, 정지 구간의 백분율 % 수이며, 총합은 100 이 되어야 합니다.
SCurveRegion	I32	수치 단위	S-Curve 구역 수량. 해당 값은 RegionNum 이상일 수 없습니다. *정지 구간의 백분율.
TotalLength	I32	수치 단위	CAM 축 circle 의 펄스 수.
Master_Pulse_P	I32	수치 단위	상응하는 CAM 이 1 회전할 때 주축의 Pulse 수
Construct_Mode	U16	옵션 단위	표 만들기 방식. 0: 수동으로 TotalLength & Master_Pulse_P 를 입력합니다 1: TotalLength 에 따라 상응하는 Master_Pulse_P 를 산출합니다 2: Master_Pulse_P 에 따라 상응하는 TotalLength 를 산출합니다

■ 예제

```
U16 Status;
U16 CardNo = 16, ECAMNo =0, NodeID = 5, SlotID = 0;
I32 DataNum = 200, Percent[5]={5, 30, 50 ,10, 5}, SCurveRegion = 0, Construct_Mode = 0;
I32 TotalLength = 100000, Master_Pulse_P = 34284;

Status = _ECAT_Slave_CSP_Start_ECAM_Table_Move (CardNo, ECAMNo, NodeID, SlotID, DataNum, Percent, SCurveRegion, TotalLength, Master_Pulse_P, Construct_Mode);
```


9.48 _ECAT_Slave_CSP_Start_ECAM_Flying _Shears_Move

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Start_ECAM_Flying_Shears_Move (U16 CardNo, U16 ECAMNo, U16 NodeID, U16 SlotID, F64 GearNum_A, F64 GearNum_B, I32 KnifeNUm, F64 SlaveDiameter, F64 EncoderDiameter, I32 EncoderPulseRev, I32 SlavePUURev, I32 CutLength, I32 RegionNum, I32 Region[3], I32 Slave_Prelead, U16 PreLeadMode)

■ 목적

SlaveMotionCSP 명령, ECAM 자동 연속 절단 기능.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
ECAMNo	U16	번호 단위	ECAM 사용 번호 (0 ~ 4)
NodeID	U16	번호 단위	NodeID
SlotID	U16	번호 단위	SlotID
GearNum_A	F64	수치 단위	CAM 을 연결하는 기어 비율
GearNum_B	F64	수치 단위	CAM 을 연결하는 기어 비율
KnifeNUm	I32	수치 단위	절단 공구 수량
SlaveDiameter	F64	수치 단위	절단기 직경 (CAM)
EncoderDiameter	F64	수치 단위	인코더 직경(주축)
EncoderPulseRev	I32	수치 단위	주축이 1 회전하는 Pulse 수 (pulse / rev)

9

명칭	데이터 타입	단위	설명
SlavePUURev	I32	수치 단위	절단기(CAM)가 1 회전하는 Pulse 수
CutLength	I32	수치 단위	절단 길이 (mm)
RegionNum	I32	수치 단위	구역 수 (RegionNum + 1 비트 데이터 생성)
Region[3]	I32	수치 단위	각 구역이 차지하는 도수 (가속 구간 (감속 구간), 동시 구간, S-Curve) ※Region 의 설정: Region[0], Region[1], Region[2] 각각 (가속 구간(감속 구간), 동시 구간, S-Curve)의 각도를 의미 잔여 각도 = 360 – (2* Region[0] + Region[1] + 2*Region[2]) 예를 들어 잔여 각도가 < Region[2] / 2 이면 오류 코드가 반환됩니다
Slave_Prelead	I32	수치 단위	CAM 축 전위 길이(pulse). PreLeadMode = 1 이어야만 유효합니다
PreLeadMode	U16	옵션 단위	전위 구간 동작 모드. 0: 전위 상태에서 CAM 의 가속 동작이 발생하지 않음. 1: 전위 구간에서 가속 동작이 발생.

■ **예제**

```
U16 Status;
U16 CardNo = 16, ECAMNo =0, NodeID = 5, SlotID = 0, PreLeadMode = 0;
F64 GearNum_A = 1, GearNum_B = 1, SlaveDiameter = 599.995, EncoderDiameter = 250;
I32 KnifeNUm = 1, EncoderPulseRev = 10000, SlavePUURev = 100000, CutLength = 3000, DataNum = 72, Region[3] = {30, 80, 100}, Slave_Prelead = 3000;


Status = _ECAT_Slave_CSP_Start_ECAM_Flying_Shears_Move (CardNo, ECAMNo, NodeID, SlotID, GearNum_A, GearNum_B, KnifeNUm, SlaveDiameter, EncoderDiameter, EncoderPulseRev, SlavePUURev, CutLength, DataNum, Region[3], Slave_Prelead, PreLeadMode);
```

9.49 _ECAT_Slave_CSP_Start_ECAM_Intermittence _Print_Move

■ 포맷

U16 PASCAL _ECAT_Slave_CSP_Start_ECAM_Intermittence_Print_Move (U16 CardNo, U16 ECAMNo, U16 NodeID, U16 SlotID, I32 RegionNum, I32 PrintLength, I32 BlankLength, F64 GearNum_A, F64 GearNum_B, I32 SlavePUURev, F64 SlaveDiameter, I32 MasterPulseRev, F64 MasterDiameter, I32 HoldingAreaDeg, I32 SCurveDeg, I32 SynIncreaseDeg)

■ 목적

SlaveMotionCSP 명령, ECAM 인터벌 인쇄 기능.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
ECAMNo	U16	번호 단위	ECAM 사용 번호 (0 ~ 4)
NodeID	U16	번호 단위	NodeID
SlotID	U16	번호 단위	SlotID
RegionNum	I32	수치 단위	구역 수 (RegionNum + 1 비트 데이터 생성)
PrintLength	I32	수치 단위	인쇄 길이 (mm)
BlankLength	I32	수치 단위	공백 길이 (mm)
GearNum_A	I32	수치 단위	기어비 A
GearNum_B	I32	수치 단위	기어비 B
SlavePUURev	I32	수치 단위	종축이 1 회전하는 Pulse 수 (pulse / rev)
SlaveDiameter	F64	수치 단위	종축 롤러 직경
MasterPulseRev	I32	수치 단위	주축 인코더가 1 회전하는 펄스 수 (pulse / rev)
MasterDiameter	F64	수치 단위	주축의 인쇄 롤러 직경
HoldingAreaDeg	I32	수치 단위	대기 구간 각도
SCurveDeg	I32	수치 단위	S-Curve 각도
SynIncreaseDeg	I32	수치 단위	동시 구간 증가 각도

9

■ 예제

```
U16 Status;
U16 CardNo = 16, ECAMNo =0, NodeID = 5, SlotID = 0;
F64 GearNum_A = 1, GearNum_B = 1, SlaveDiameter = 500, MasterDiameter = 500;
I32 SlavePUURev = 50000, MasterPulseRev = 34284, HoldingAreaDeg = 30,
SCurveDeg = 40, SynIncreaseDeg = 30;

Status = _ECAT_Slave_CSP_Start_ECAM_Intermittence_Print_Move (CardNo, ECAMNo,
NodeID, SlotID, DataNum, PrintLength, BlankLength, GearNum_A, GearNum_B,
SlavePUURev, SlaveDiameter, MasterPulseRev, MasterDiameter, HoldingAreaDeg,
SCurveDeg, SynIncreaseDeg);
```

Motion Slave CSV API 10

다음은 단축 속도 제어 동작, 다축의 동기 속도 제어 동작 등 Motion Slave CSV 관련 API 에 대한 설명입니다.

10

Motion Slave CSV API 테이블

함수의 명칭	설명
_ECAT_Slave_CSV_Start_Move	MotionSlaveCSV 명령, 단축 속도 제어 동작
_ECAT_Slave_CSV_Multi_Start_Move	MotionSlaveCSV 명령, 다축의 동기 속도 제어 동작

10.1 _ECAT_Slave_CSV_Start_Move

■ **포맷**

U16 PASCAL _ECAT_Slave_CSV_Start_Move (U16 CardNo, U16 AxisNo, U16 SlotNo,I32 Target_Velocity, F64 Acceleration, U16 Curve_Mode, U16 Acc_Type)

■ **목적**

MotionSlave CSV 명령, 단축 속도 제어 동작.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Target_Velocity	I32	inc/s	목표 속도, (0x60FF Sub 0) inc 은 드라이브 내부에 설정한 단위값이기 때문에 사용하는 드라이브의 매뉴얼을 참고하시기 바랍니다.
Acceleration	F64	초 펄스 수 s^2	목표 속도에 도달한 시간 inc/s^2 (0x6083 Sub 0)
Curve_Mode	U16	옵션 단위	1 : T-curve (Default) 2 : S-curve
Acc_Type	U16	옵션 단위	0: Acceleration 의 단위는 시간(초)입니다 1: Acceleration 의 단위는 가속도(펄스 수/s^2)입니다

■ 예제

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0, Curve_Mode = 1, Acc_Type = 0;
I32 Target_Velocity =600000;
F64 Acceleration = 0.2;

Status = _ECAT_Slave_CSV_Start_Move (CardNo, AxisNo, SlotNo, Target_Velocity,
Acceleration, Curve_Mode, Acc_Type);
```


10.2 _ECAT_Slave_CSV_Multi_Start_Move

■ 포맷

U16 PASCAL _ECAT_Slave_CSV_Multi_Start_Move (U16 CardNo, U16 AxisNum, U16 *AxisNo, U16 *SlotNo, I32 *Target_Velocity, F64 *Acceleration, U16 Curve_Mode, U16 Acc_Type)

■ 목적

Slave Motion CSV 명령, 다축의 동기 속도 제어 동작.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNum	U16	수치 단위	관련 축수
AxisNo	U16*	번호 단위	NodeID 번호
SlotNo	U16*	번호 단위	SlotID 번호
Target_Velocity	I32*	inc/s	목표 속도, (0x60FF Sub 0) inc 은 드라이브 내부에 설정한 단위값이기 때문에 사용하는 드라이브의 매뉴얼을 참고하시기 바랍니다.
Acceleration	F64*	초 펄스 수/s^2	목표 속도에 도달한 시간 inc/s^2 (0x6083 Sub 0)
Curve_Mode	U16	옵션 단위	1 : T-curve (Default) 2 : S-curve
Acc_Type	U16	옵션 단위	0: Acceleration 의 단위는 시간(초)입니다 1: Acceleration 의 단위는 가속도(펄스 수/s^2)입니다

■ 예제

```
U16 Status;
U16 CardNo=16, AxisNum = 2;
U16 AxisNo[2]={1, 2}, SlotNo[2]={0, 0}, Curve_Mode = 1, Acc_Type = 0;
I32 Target_Velocity[2] ={600000, 600000};
F64 Acceleration[2] = {0.2, 0.2};

Status = _ECAT_Slave_CSV_Multi_Start_Move (CardNo, AxisNum, &AxisNo, &SlotNo,
&Target_Velocity, &Acceleration, Curve_Mode, Acc_Type);
```

Motion Slave CST API 11

다음은 단축의 고정 토크 제어 동작, 다축의 동기 고정 토크 제어 동작 등 Motion Slave CST 관련 API 사용법에 대한 설명입니다.

11

Motion Slave CST API 테이블

함수의 명칭	설명
_ECAT_Slave_CST_Start_Move	MotionSlaveCST 명령, 단축의 고정 토크 제어 동작
_ECAT_Slave_CST_Multi_Start_Move	MotionSlaveCST 명령, 다축의 동기 고정 토크 제어 동작

11.1 _ECAT_Slave_CST_Start_Move

■ **포맷**

U16 PASCAL _ECAT_Slave_CST_Start_Move (U16 CardNo, U16 AxisNo, U16 SlotNo,I16 Target_Torque, U32 Slope, U16 Curve_Mode)

■ **목적**

MotionSlave CST 명령, 단축의 고정 토크 제어 동작.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Target_Torque	I16	1/1000	1 ~ 1000
Slope	U32	0.1 %/s	토크의 상승 경사도를 설정합니다
Curve_Mode	U16	옵션 단위	1 : T-curve (Default) 2 : S-curve

■ **예제**

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0, Curve_Mode = 1;
I16 Target_Torque = 2;
U32 Slope = 10;

Status = _ECAT_Slave_CST_Start_Move (CardNo, AxisNo, SlotNo, Target_Torque,
Slope, Curve_Mode);
```

11.2 _ECAT_Slave_CST_Multi_Start_Move

■ **포맷**

U16 PASCAL _ECAT_Slave_CST_Multi_Start_Move (U16 CardNo, U16 AxisNum, U16 *AxisNo, U16 *SlotNo, I16 * Target_Torque, U32 *Slope, U16 Curve_Mode)

■ **목적**

Slave Motion CST 명령, 다축의 동기 고정 토크 제어 동작.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNum	U16	수치 단위	관련 축수
AxisNo	U16*	번호 단위	NodeID 번호
SlotNo	U16*	번호 단위	SlotID 번호
Target_Velocity	I16*	1/1000	1 ~ 1000
Slope	U32*	0.1 %/s	토크의 상승 경사도를 설정합니다
Curve_Mode	U16	옵션 단위	1 : T-curve (Default) 2 : S-curve

■ **예제**

```
U16 Status;
U16 CardNo=16, AxisNum = 2;
U16 AxisNo[2]={1, 2}, SlotNo[2]={0, 0}, Curve_Mode = 1, Acc_Type = 0;
I16 Target_Torque[2] ={2, 2};
U32 Slope[2] = {10, 10};

Status = _ECAT_Slave_CST_Multi_Start_Move (CardNo, AxisNum, &AxisNo, &SlotNo,
&Target_Velocity, &Slope, Curve_Mode);
```

(이 페이지는 공란으로 비워둡니다)

Motion Slave Home API 12

다음은 Home 모드 설정, Home 실행, Home 으로 돌아가기 상태 확인 등 Motion Slave Home 관련 API 사용법에 대한 설명입니다.

12

Motion Slave Home API 테이블

함수의 명칭	설명
_ECAT_Slave_Home_Config	MotionSlaveHome 명령, Home 모드 설정
_ECAT_Slave_Home_Move	MotionSlaveHome 명령, Home 으로 돌아가기 실행
_ECAT_Slave_Home_Status	MotionSlaveHome 명령, 현재 Home 으로 돌아가기 상태 확인

12.1 _ECAT_Slave_Home_Config

■ **포맷**

U16 PASCAL _ECAT_Slave_Home_Config (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Mode, I32 Offset, U32 FirstVel, U32 SecondVel, U32 Acceleration)

■ **목적**

MotionSlave Home 명령, Home 모드를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Mode	U16	옵션 단위	각 드라이브의 정의를 참고하시기 바랍니다 (0x6098 Sub 0)
Offset	I32	inc	원점 복귀 오프셋량 (0x607C Sub 0) inc 은 드라이브 내부에 설정한 단위값이기 때문에 사용하는 드라이브의 매뉴얼을 참고하시기 바랍니다.
FirstVel	U32	펄스/초	inc/s, Speed during search for switch (0x6099 Sub 1)
SecondVel	U32	펄스/초	inc/s, Speed during search for zero (0x6099 Sub 2)
Acceleration	U32	펄스/s^2	inc/s^2 (0x609A Sub 0)

■ **예제**

```
U16 Status;
U16 CardNo=16, AxisNo=1, SlotNo=0, Mode=1;
I32 Offset=200;
U32 FirstVel =600000, SecondVel =100000;
U32 Acceleration =3;

/*원점 복귀 모드설정*/
Status = _ECAT_Slave_Home_Config (CardNo, AxisNo, SlotNo, Mode, Offset, FirstVel,
SecondVel, Acceleration);
```

원점 복귀 모드

#1. (-) limit 과 index pulse 정보를 이용한 원점 복귀 모드

이 방법을 사용할 때 만약 (-) limit 의 스위치가 off(낮은 전위로)일 때 초기 방향은 왼쪽을 향해 변이 동작을 실행합니다. 원점 위치는 (-) limit 스위치에 도달했을 때 오른쪽 방향의 index pulse 를 향해 변위합니다.

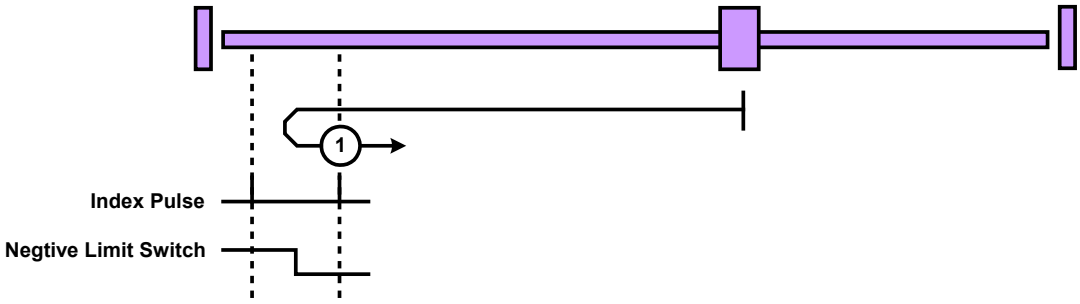


그림 12.1.1

#2. (+) limit 과 index pulse 정보를 이용한 원점 복귀 모드

이 방법을 사용할 때 만약 (+) limit 의 스위치가 off(낮은 전위로)일 때 초기 방향은 오른쪽을 향해 변이 동작을 실행합니다. 이 원점 위치는 (+) limit 스위치에 도달했을 때 좌측의 index pulse 로 이동합니다.

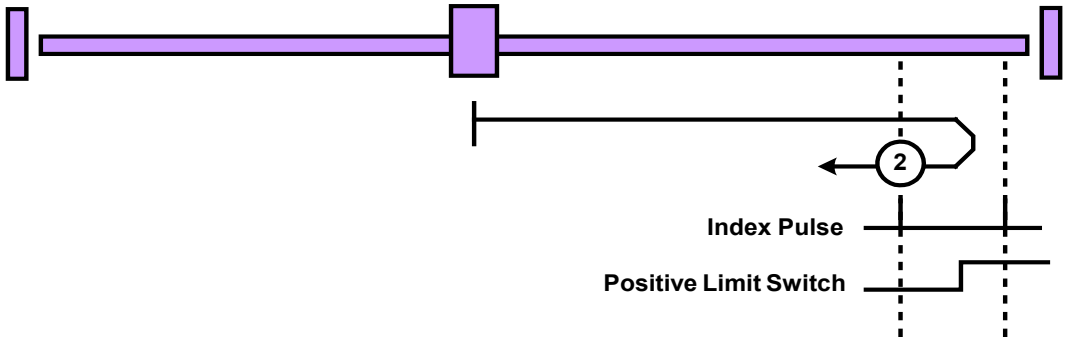


그림 12.1.2

12

#3 and 4. 원점 스위치의 순방향 전위와 index pulse 정보를 이용한 원점 복귀 모드

사용 방법 3 또는 4 의 초기 변위 방향은 원점에 따라 현재 상태를 On / Off 한 것입니다. 이 원점의 위치는 해당 원점의 스위치로 상태를 변경할 때 왼쪽 또는 오른쪽 중 어느 한쪽으로 향하는 index pulse 에 의해 결정됩니다. 원점 복귀를 실행한 시작 지점이 index pulse 일 경우 반드시 변위 방향을 되돌려야 합니다. 변위 방향을 되돌린 후 이 지점에서 발생하는 변화는 원점 스위치의 현재 상태에 의해 결정됩니다.

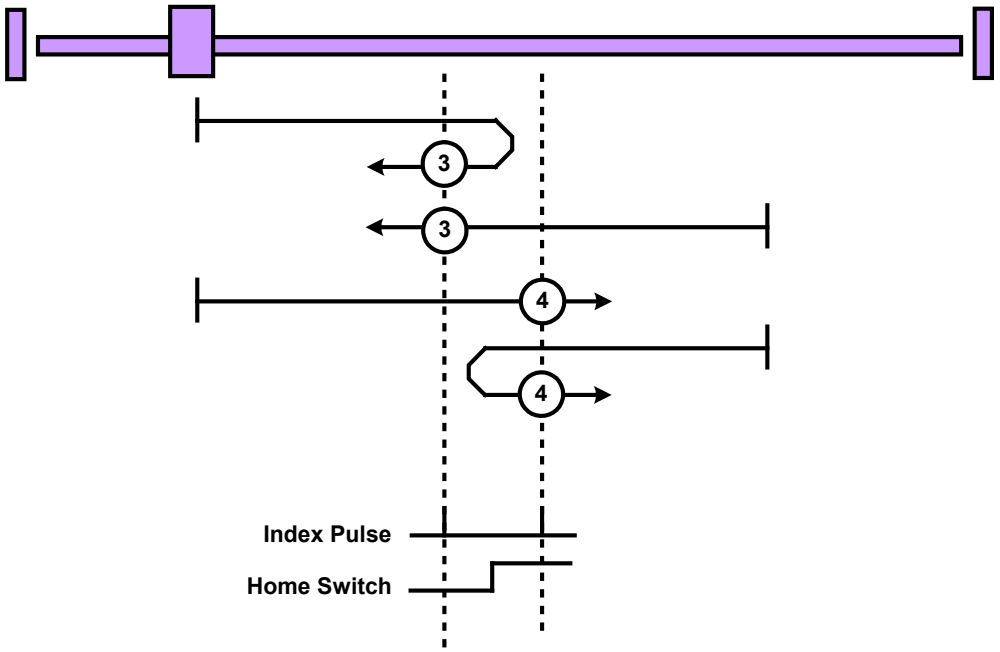


그림 12.1.3

#5 and 6. 원점 스위치의 마이너스 방향 전위와 index pulse 정보를 이용한 원점 복귀 모드

사용 방법 5 또는 6 의 초기 변위 방향은 원점에 따라 현재 상태를 On / Off 한 것입니다. 이 원점의 위치는 해당 원점의 스위치로 상태를 변경할 때 왼쪽 또는 오른쪽 중 어느 한쪽으로 향하는 index pulse 에 의해 결정됩니다. 원점 복귀를 실행한 시작 지점이 index pulse 일 경우 반드시 변위 방향을 되돌려야 합니다. 변위 방향을 되돌린 후 이 지점에서 발생하는 변화는 원점 스위치의 현재 상태에 의해 결정됩니다.

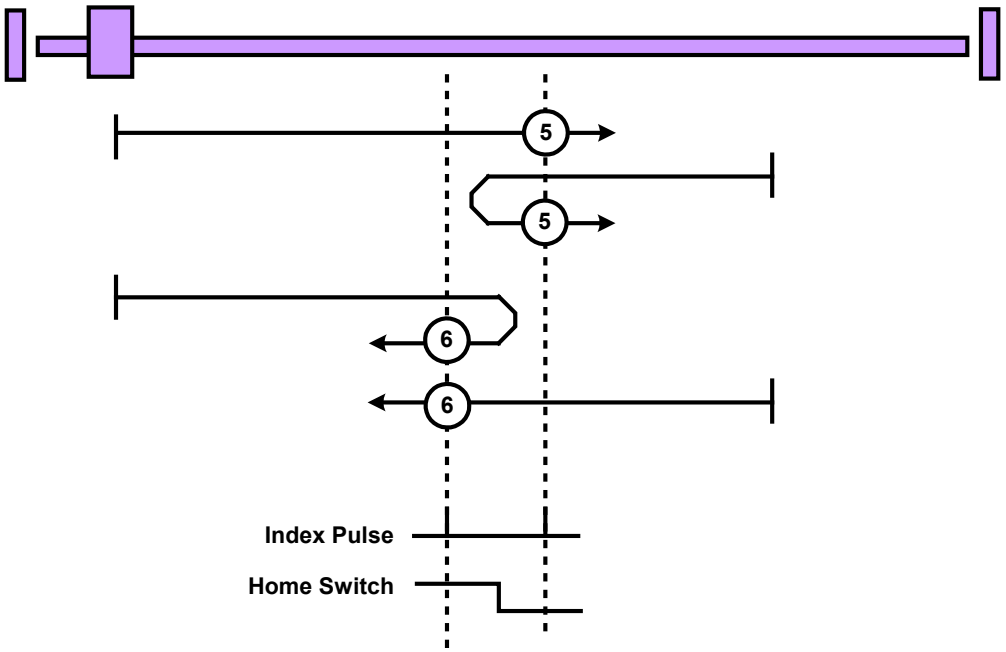


그림 12.1.4

#7 to 14. index pulse 와 원점 스위치에 기반한 원점 복귀 모드

이 방법은 변위 과정 중에 사용된 원점 스위치 중 1 개를 사용합니다. 사실상 해당 축의 위치가 이 스위치를 통과할 때 '순간적으로' 사용될 수 있습니다. 사용 방법 7 부터 10 의 초기 변위 방향은 오른쪽이고, 사용 방법 11 부터 14 까지의 초기 변위 방향은 왼쪽입니다. 이 밖에도 원점 스위치는 변위 시작점에 있다면 사용 상태가 됩니다. 이 때의 초기 변위 방향은 찾아낸 가장자리(edge)에 의해 결정됩니다. 이 원점 회귀 위치는 (+) 엣지 또는 (-) 엣지 중 어느 한쪽의 원점 스위치의 index pulse 신호입니다. 아래 두 그림에서 보여지는 것처럼, 만약 초기 변위 방향이 원점 스위치를 만나지 않을 경우 동작이 limit 스위치에 만날 때 비로소 변위 방향을 되돌릴 수 있습니다.

12

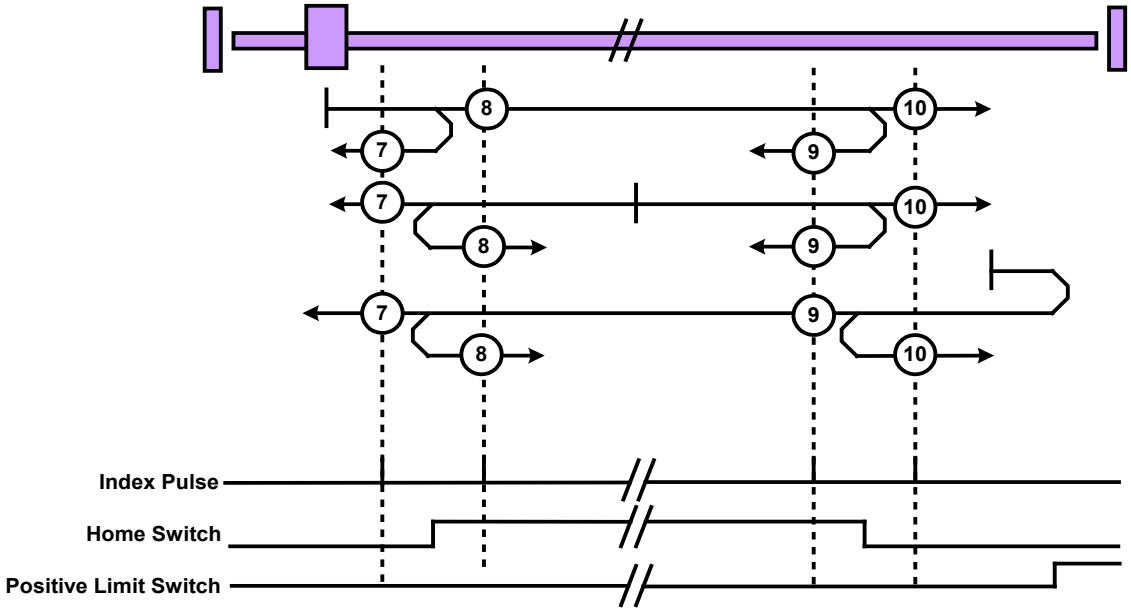


그림 12.1.5

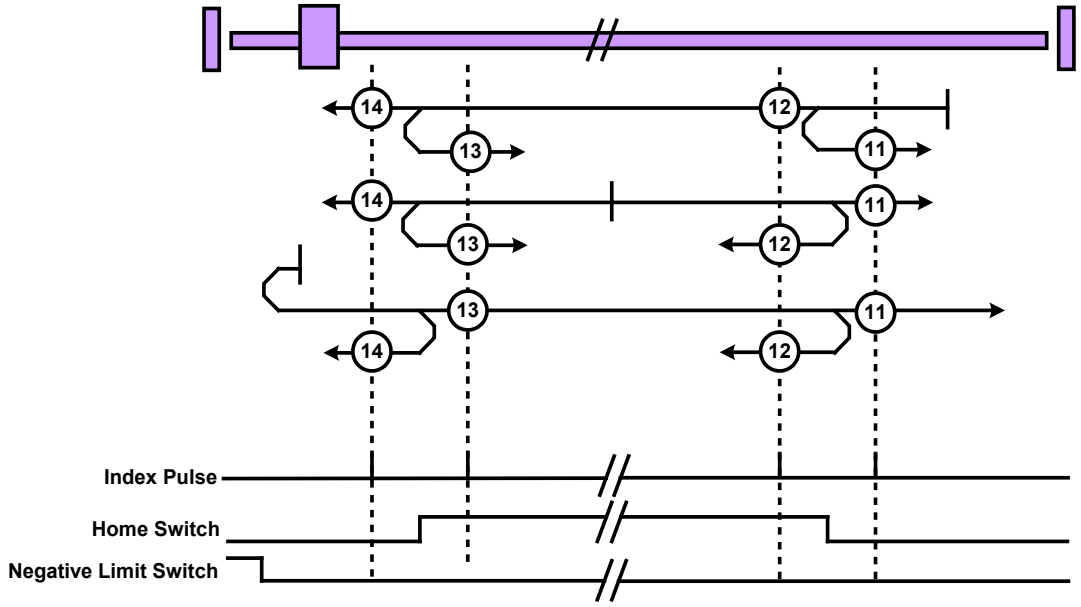


그림 12.1.6

#15 and 16 보류

원점 복귀 모드의 향후 확장 사용을 위해 남겨둡니다.

#17 to 30. index pulse 와 무관한 원점 복귀 모드

이런 방법은 방법 1 에서 14 와 비슷하지만 둘의 차이는 이와 유사한 원점 위치 복귀는 index pulse 에 의존하지 않는다는 것에 있습니다. 대부분의 의존은 원점 또는 limit 스위치의 전환으로 원점 복귀를 실행합니다. 예: 방법 19 와 방법 20 은 아래 7.7 그림에 나오는 것처럼 방법 3 과 방법 4 의 원점 복귀 모드와 유사합니다.

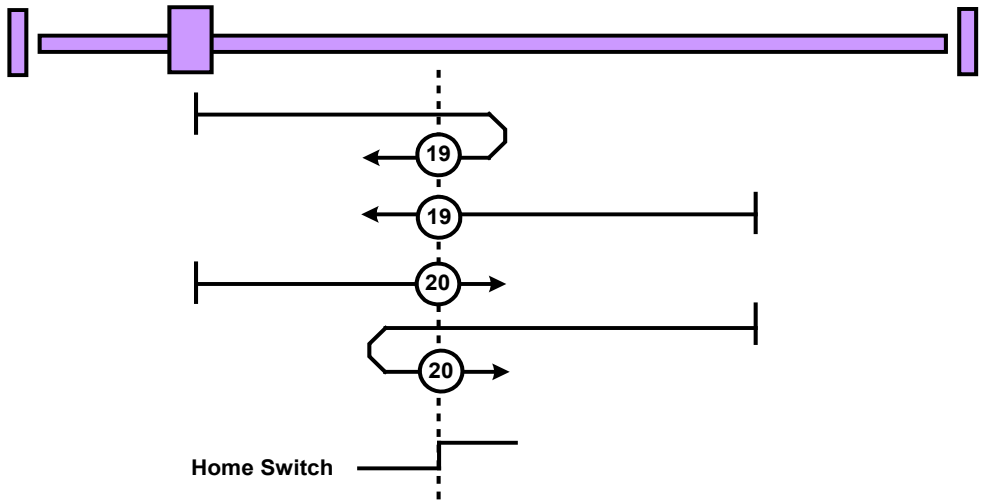


그림 12.1.7

#31 and 32 보류

원점 복귀 모드의 향후 확장 사용을 위해 남겨둡니다.

#33 to 34. index pulse 와 관련 원점 복귀 모드

방법 **33** 또는 **34** 를 사용하는데 있어, 원점 복귀의 플러스 방향과 마이너스 방향의 차이점은 원점 복귀 시 선택하는 방향
위치가 확인된 index pulse 에 따라 결정된다는 것입니다.

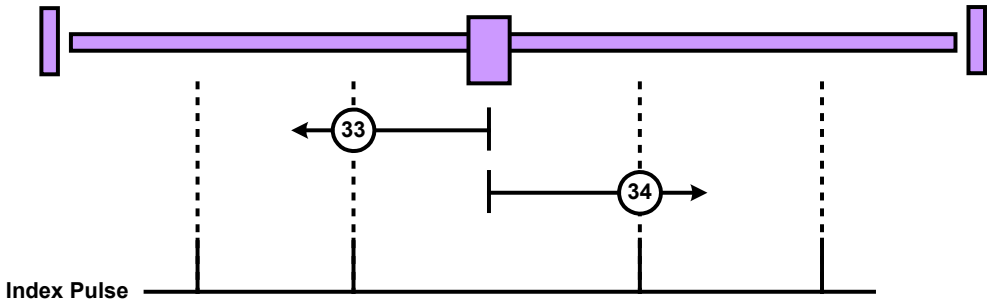


그림 12.1.8

#35. 현재 position 위치에 따라 원점 복귀 모드를 실행합니다

모드 35 는 현재 위치를 원점 복귀의 위치로 여깁니다.

12

12.2 _ECAT_Slave_Home_Move

■ **포맷**

U16 PASCAL _ECAT_Slave_Home_Move (U16 CardNo, U16 AxisNo, U16 SlotNo)

■ **목적**

MotionSlave Home 명령, Home 으로 돌아가기를 실행합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호

■ **예제**

```
U16 Status;
U16 CardNo =1 , AxisNo =1, SlotNo=0;

/*원점 복귀 실행*/
Status = _ECAT_Slave_Home_Move (CardNo, AxisNo, SlotNo);
```

12.3 _ECAT_Slave_Home_Status

■ 포맷

U16 PASCAL _ECAT_Slave_Home_Status (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 *Status)

■ 목적

MotionSlave Home 명령, 현재 Home 으로 돌아가기 상태를 확인합니다.
※ 이 API 는 Home 모드에서만 사용 가능합니다. 다른 동작 모드에서 사용할 경우 ReturnCode ERR_ECAT_MODE_NOT_SUPPORT(4612)가 나타납니다

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Status	U16*	수치 단위	Mode : 0: 아직 동작되지 않았거나 Home 으로 돌아가기 완료 1: Home 동작 실행 중 2: Home 으로 돌아가기 동작이 완료되지 않은 상태에서 중지됨 3: Home 으로 돌아가기 동작 중 Error 발생

■ 예제

```
U16 Status;
U16 CardNo =1 , AxisNo =1, SlotNo=0;

/*원점 복귀 상태 확인*/
Status = _ECAT_Slave_Home_Status (CardNo, AxisNo, SlotNo, &Status);
```

(이 페이지는 공란으로 비워둡니다)

Motion Slave PP API

13

다음은 단축 직선 동작, PP 모드 동작 고급 설정 등 Motion Slave PP 관련 API 사용법에 대한 설명입니다.

13

Motion Slave PP API 테이블

함수의 명칭	설명
_ECAT_Slave_PP_Start_Move	MotionSlavePP 명령, 단축 직선 동작
_ECAT_Slave_PP_Advance_Config	MotionSlavePP 명령, PP 모드 동작의 고급 설정

13.1 _ECAT_Slave_PP_Start_Move

■ **포맷**

U16 PASCAL _ECAT_Slave_PP_Start_Move (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 TargetPos, U32 ConstVel, U32 Acceleration, U32 Deceleration, U16 Abs_Rel)

■ **목적**

MotionSlave PP 명령, 단축 직선 동작.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
TargetPos	I32	inc	지정된 동작 stroke (0x607A Sub 0) inc 은 드라이브 내부에 설정한 단위값이기 때문에 사용하는 드라이브의 매뉴얼을 참고하시기 바랍니다.
ConstVel	U32	펄스 수/초	inc/s (0x6081 Sub 0)
Acceleration	U32	펄스 수/s^2	inc/s^2 (0x6083 Sub 0)
Deceleration	U32	펄스 수/s^2	inc/s^2 (0x6084 Sub 0)
Abs_Rel	U16	옵션 단위	0: 상대적 변위 (Default) 1: 절대적 변위

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0, Abs_Rel=1;
I32 Dist=5000000;
U32 MaxVel=2000000;
U32 TAcc = 100, Tdec = 100; // A2E: 3000rpm 까지 가속에 필요한 시간(ms)

Status = _ECAT_Slave_PP_Start_Move(CardNo, AxisNo, SlotNo, Dist, MaxVel, TAcc, TDec, Abs_Rel);
```

13.2 _ECAT_Slave_PP_Advance_Config

■ **포맷**

U16 PASCAL _ECAT_Slave_PP_Advance_Config (U16 CardNo, U16 AxisNo, U16 SlotNo,U16 SetBit, I32 End_Vel, I32 Min_Range_Limit, I32 Max_Range_Limit, I32 Min_Soft_Limit, I32 Max_Soft_Limit)

■ **목적**

MotionSlave PP 명령, PP 모드 동작의 고급 설정이 가능합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
SetBit	U16	옵션 단위	Bit 형식으로 설정할 항목을 설정합니다. 해당 목록은 아래와 같습니다↓ Bit 0 → End_Vel Bit 1 → Min_Range_Limit Bit 2 → Max_Range_Limit Bit 3 → Min_Soft_Limit Bit 4 → Max_Soft_Limit
End_Vel	I32	펄스 수/초	최종 속도, inc (0x607A Sub 0)
Min_Range_Limit	I32	inc	동작의 상대 위치의 최소 범위 설정 (0x607B Sub 1)
Max_Range_Limit	I32	inc	동작의 상대 위치의 최대 범위 설정 (0x607B Sub 2)
Min_Soft_Limit	I32	inc	동작의 절대 위치의 최소 범위 설정 (0x607D Sub 1)
Max_Soft_Limit	I32	inc	동작의 절대 위치의 최대 범위 설정 (0x607D Sub 2)

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0, SetBit=0x07;
//End_Vel, Min_Range_Limit&Max_Range_Limit 설정
I32 End_Vel =50, Min_Range_Limit =1, Max_Range_Limit =200,;
I32 Min_Soft_Limit= 2, Max_Soft_Limit=180 ;

Status = _ECAT_Slave_PP_Advance_Config(CardNo, AxisNo, SlotNo, SetBit, End_Vel,
Min_Range_Limit, Max_Range_Limit, Min_Soft_Limit, Max_Soft_Limit);
```

(이 페이지는 공란으로 비워둡니다)

Motion Slave Velocity API 14

다음은 단축의 고정 회전속도 동작, PV 모드 동작 고급 설정 등 Motion Slave Velocity 관련 API 사용법에 대한 설명입니다.

14

Motion Slave Velocity API 테이블

함수의 명칭	설명
_ECAT_Slave_PV_Start_Move	MotionSlaveVelocity 명령, 단축의 고정 회전속도 동작
_ECAT_Slave_PV_Advance_Config	MotionSlaveVelocity 명령, PV 모드 동작의 고급 설정

14.1 _ECAT_Slave_PV_Start_Move

■ **포맷**

U16 PASCAL _ECAT_Slave_PV_Start_Move (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 TargetVel, U32 Acceleration, U32 Deceleration)

■ **목적**

MotionSlave Velocity 명령, 단축의 고정 회전속도 동작을 실행합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
TargetVel	I32	inc/s	목표 속도, (0x60FF Sub 0) inc 은 드라이브 내부에 설정한 단위값이기 때문에 사용하는 드라이브의 매뉴얼을 참고하시기 바랍니다.
Acceleration	U32	inc/s^2	가속도 (0x6083 Sub 0)
Deceleration	U32	inc/s^2	감속도 (0x6084 Sub 0)

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 TargetVel=300;
F64 Acceleration = 5, Deceleration = 5;

Status = _ECAT_Slave_PV_Start_Move (CardNo, AxisNo, SlotNo, TargetVel,
Acceleration, Deceleration);
```

14.2 _ECAT_Slave_PV_Advance_Config

■ **포맷**

U16 PASCAL _ECAT_Slave_PV_Advance_Config (U16 CardNo, U16 AxisNo,
U16 SlotNo, U16 SetBit, U16 Max_Torque, U16 Velocity_Window,
U16 Velocity_Window_Time, U16 Velocity_Threshold, U16 Velocity_Threshold_Time)

■ **목적**

MotionSlave Velocity 명령, PV 모드 동작의 고급 설정이 가능합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
SetBit	U16	옵션 단위	Bit 형식으로 설정할 항목을 설정합니다. 해당 목록은 아래와 같습니다 Bit 0 → Max_Torque Bit 1 → Velocity_Window Bit 2 → Velocity_Window_Time Bit 3 → Velocity_Threshold Bit 4 → Velocity_Threshold_Time
Max_Torque	U16	1/1000	1 ~ 1000 (0x6072 Sub 0)
Velocity_Window	U16	inc/s	설정한 속도 구간 (0x606D Sub 0)
Velocity_Window_Time	U16	밀리초	설정한 지속 시간 (0x606E Sub 0) 동작 속도와 설정한 속도 간의 차이가 Velocity_Window 에서 설정한 값보다 작고, 지속 시간이 설정한 Velocity_Window_Time 을 초과하면 Status_Word 의 bit10-Target_Reached 에 불이 들어옵니다.
Velocity_Threshold	U16	inc/s	설정한 속도 구간 (0x606F Sub 0)
Velocity_Threshold_Time	U16	밀리초	설정한 지속 시간 (0x6070 Sub 0) 동작 속도가 설정한 Velocity_Threshold 값에 도달하고, 지속 시간이 Velocity_Threshold_Time 이 설정한 시간을 초과하면 Status_Word 의 bit12-Speed 에 불이 꺼집니다.

■ 예제

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0, SetBit=0x03; //Max_Torque, Velocity_Window 설정;
U16 Max_Torque=200, Velocity_Window=20, Velocity_Window_Time=3;
U16 Velocity_Threshold =10, Velocity_Threshold_Time =3;

Status = _ECAT_Slave_PV_Advance_Config (CardNo , AxisNo , SlotNo , SetBit , Max_Torque, Velocity_Window, Velocity_Window_Time, Velocity_Threshold, Velocity_Threshold_Time);
```

Motion Slave VL API

15

다음은 인버터 단축의 고정 회전속도 동작인 Motion Slave VL 관련 API 사용법에 대한 설명입니다.

Motion Slave VL API 테이블

함수의 명칭	설명
_ECAT_Slave_VL_Start_Move	Motion Slave Velocity 명령, 인버터 단축의 고정 회전속도 동작

15

15.1 _ECAT_Slave_VL_Start_Move

■ **포맷**

U16 PASCAL _ECAT_Slave_VL_Start_Move (U16 CardNo, U16 AxisNo, U16 SlotNo, I32 TargetVel, U32 Acceleration, U32 Deceleration)

■ **목적**

MotionSlave VL 명령, 인버터 단축의 고정 회전속도 동작을 실행합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
TargetVel	I32	inc/s	목표 속도, (0x6042 Sub 0) inc 은 드라이브 내부에 설정한 단위값이기 때문에 사용하는 드라이브의 매뉴얼을 참고하시기 바랍니다.
Acceleration	U32	inc/s^2	가속도 (0x604F Sub 0)
Deceleration	U32	inc/s^2	감속도 (0x6050 Sub 0)

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I32 TargetVel=300;
F64 Acceleration = 50, Deceleration = 50;

Status = _ECAT_Slave_VL_Start_Move (CardNo, AxisNo, SlotNo, TargetVel,
Acceleration, Deceleration);
```

Motion Slave Torque API 16

다음은 단축의 고정 토크 동작, PT 모드 동작 고급 설정 등 Motion Slave Torque 관련 API 사용법에 대한 설명입니다.

16

Motion Slave Torque API 테이블

함수의 명칭	설명
_ECAT_Slave_PT_Start_Move	MotionSlaveTorque 명령, 단축의 고정 토크 동작
_ECAT_Slave_PT_Advance_Config	MotionSlaveTorque 명령, PT 모드 동작의 고급 설정

16.1 _ECAT_Slave_PT_Start_Move

■ **포맷**

U16 PASCAL _ECAT_Slave_PT_Start_Move (U16 CardNo, U16 AxisNo, U16 SlotNo, I16 Target_Torque, U32 Slope, I16 Troque_Profile)

■ **목적**

MotionSlave Torque 명령, 단축의 고정 토크 동작을 실행합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Target_Torque	I16	1/1000	0.1% rated Torque (0x6071 Sub 0)
Slope	U32	0.1 %/s	토크의 상승 경사도 설정 0.1 %/s (0x6087 Sub 0)

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
I16 Target_Torque =2;
U32 Slope=10;

Status = _ECAT_Slave_PT_Start_Move (CardNo, AxisNo, SlotNo, Target_Torque, Slope);
```

16.2 _ECAT_Slave_PT_Advance_Config

■ **포맷**

U16 PASCAL _ECAT_Slave_PT_Advance_Config (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 SetBit, U16 Max_Current, I16 Troque_Profile)

■ **목적**

MotionSlave Torque 명령, PT 모드 동작의 고급 설정이 가능합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
SetBit	U16	옵션 단위	Bit 형식으로 설정할 항목을 설정합니다 Bit 0 → Max_Current (0x6073 Sub 0) Bit 1 → Troque_Profile (0x6088, Sub 0)
Max_Current	U16	1/1000	1 ~ 1000 (0x6073 Sub 0)
Troque_Profile	I16	옵션 단위	0: 선형 변화 1: Sin 파 변화 나머지: 드라이브 매뉴얼 참조 상세 설명은 사용하는 드라이브의 매뉴얼을 참고하시기 바랍니다.

■ **예제**

```
U16 Status;
U16 CardNo=16,AxisNo=1,SlotNo=0;
U16 Max_Current =200, SetBit = 0, Troque_Profile = 0;

Status = _ECAT_Slave_PT_Advance_Config (CardNo, AxisNo, SlotNo, Max_Current,
Troque_Profile);
```

(이 페이지는 공란으로 비워둡니다)

16

Motion Slave User API 17

다음은 Group 의 모든 관련 설정 등 Motion Slave User 관련 API 사용법에 대한 설명입니다.

17

Motion Slave User API 테이블

함수의 명칭	설명
_ECAT_Slave_User_Motion_Control_ Set_Enable_Mode	MotionSlave User Motion Control 명령, 해당 Group 의 활성화 상태를 설정합니다. ※주의 Enable 전에 Set_Motion_Control_Type 과 각 축의 Svon&Alm 상태를 먼저 사용해야 합니다.
_ECAT_Slave_User_Motion_Control_ Get_Enable_Mode	MotionSlave User Motion Control 명령, 현재 해당 Group 의 활성화 상태를 확인합니다.
_ECAT_Slave_User_Motion_Control_ Set_Type	MotionSlave User Motion Control 명령, 해당 Group 의 Motion 모드를 설정합니다.
_ECAT_Slave_User_Motion_Control_ Set_Data	MotionSlave User Motion Control 명령, 해당 Group 각 축의 모드에 상응하는 데이터를 입력합니다.
_ECAT_Slave_User_Motion_Control_ Clear_Data	MotionSlave User Motion Control 명령, 해당 Group 각 축에 입력한 데이터를 삭제합니다.
_ECAT_Slave_User_Motion_Control_ Get_DataCnt	MotionSlave User Motion Control 명령, 해당 Group 에서 처리가 완료되지 않은 데이터 비트수를 확인합니다.
_ECAT_Slave_User_Motion_Control_ Ralm	MotionSlave User Motion Control 명령, 해당 Group 의 모든 축의 Alm 을 리셋합니다.
_ECAT_Slave_User_Motion_Control_ Svon	MotionSlave User Motion Control 명령, 해당 Group 의 모든 축을 Svon / off 합니다.
_ECAT_Slave_User_Motion_Control_ Get_Alm	MotionSlave User Motion Control 명령, 해당 축의 현재 Alm 상태를 확인합니다.

17.1 _ECAT_Slave_User_Motion_Control_Set _Enable_Mode

■ **포맷**

U16 PASCAL _ECAT_Slave_User_Motion_Control_Set_Enable_mode (U16 CardNo, U16 GroupNo, U16 Mode)

■ **목적**

MotionSlave User Motion Control 명령, 해당 Group 의 활성화 상태 설정
※주의 Enable 전에 Set_Motion_Control_Type 과 각 축의 Svon&Alm 상태를 먼저 사용해야 합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
GroupNo	U16	번호 단위	Group 번호
Mode	U16	옵션 단위	Mode 설명 0 : Disable 1 : Enable 2 : Pause

■ **예제**

```
U16 Status;
U16 CardNo=16, GroupNo =1;
U16 Mode =1;

Status =_ECAT_Slave_User_Motion_Control_Set_Enable_Mode (CardNo, GroupNo, Mode);
```

17.2 _ECAT_Slave_User_Motion_Control_Get _Enable_Mode

■ **포맷**

U16 PASCAL _ECAT_Slave_User_Motion_Control_Get_Enable_mode (U16 CardNo, U16 GroupNo, U16* Mode)

■ **목적**

MotionSlave User Motion Control 명령, 현재 해당 Group 의 활성화 상태 확인

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
GroupNo	U16	번호 단위	Group 번호
Mode	U16*	옵션 단위	Mode 설명 0 : Disable 1 : Enable 2 : Pause

■ **예제**

```
U16 Status;
U16 CardNo=16, GroupNo =1;
U16 Mode ;

Status =_ECAT_Slave_User_Motion_Control_Get_Enable_Mode (CardNo, GroupNo, &Mode);
```

17.3 _ECAT_Slave_User_Motion_Control_Set_Type

■ **포맷**

U16 PASCAL _ECAT_Slave_User_Motion_Control_Set_Type (U16 CardNo, U16 GroupNo, U16 AxisNum, U16 *AxisNo, U16 *SlotNo, U16 Type)

■ **목적**

MotionSlave User Motion Control 명령, 해당 Group 의 Motion 모드를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
GroupNo	U16	번호 단위	Group 번호
AxisNum	U16	수치 단위	관련 축수
AxisNo	U16*	번호 단위 배열	Node 번호 ID 배열, 수량은 AxisNum 입니다 AxisNo Array[0] 첫 번째 세트의 Node 번호를 배열합니다 AxisNo Array[1] 두 번째 세트의 Node 번호를 배열합니다 ….
SlotNo	U16*	번호 단위 배열	SlotID 번호 배열, 수량은 AxisNum 입니다
Type	U16	옵션 단위	모드 설명 0: SoftMotion (Delta 내 생성) 1 : User CSP Control (사용자가 데이터의 CSP 모드를 생성) 2 : User CSV Control (사용자가 데이터의 CSV 모드를 생성)

■ **예제**

```
U16 Status;
U16 CardNo=16, GroupNo =1, AxisNum=2, AxisNoArray[2] ={1,2}, SlotNoArray[2]
={0,0};
U16 Type=0;

Status =_ECAT_Slave_User_Motion_Control_Set_Type (CardNo, GroupNo ,AxisNum ,
AxisNoArray, SlotNoArray, Type);
```

17.4 _ECAT_Slave_User_Motion_Control_Set_Data

■ **포맷**

U16 PASCAL _ECAT_Slave_User_Motion_Control_Set_Data (U16 CardNo, U16 GroupNo, I32 *Data)

■ **목적**

MotionSlave User Motion Control 명령, 해당 Group 각 축의 모드에 상응하는 데이터를 입력합니다.
(PAC 의 최대 데이터 비트수는 800 비트이며, 축 카드의 최대 데이터 비트수는 100 비트입니다)

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
GroupNo	U16	번호 단위	Group 번호
Data	I32*	수치 단위 배열	Control Data 배열, 수량은 AxisNum 입니다 DataArray[0] 첫 번째 세트의 데이터 저장 DataArray[1] 두 번째 세트의 데이터 저장 …

■ **예제**

```
U16 Status;
U16 CardNo=16, GroupNo =1;
I32 DataArray[2]={12,33};

Status = _ECAT_Slave_User_Motion_Control_Set_Data (CardNo, GroupNo,
DataArray);
```

17.5 _ECAT_Slave_User_Motion_Control_Clear_Data

■ **포맷**

U16 PASCAL _ECAT_Slave_User_Motion_Control_Clear_Data (U16 CardNo, U16 GroupNo)

■ **목적**

MotionSlave User Motion Control 명령, 해당 Group 각 축에 입력한 데이터를 삭제합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
GroupNo	U16	번호 단위	Group 번호

■ **예제**

```
U16 Status;
U16 CardNo=16, GroupNo =1;

Status = _ECAT_Slave_User_Motion_Control_Clear_Data (CardNo, GroupNo);
```

17.6 _ECAT_Slave_User_Motion_Control _Get_DataCnt

■ **포맷**

U16 PASCAL _ECAT_Slave_User_Motion_Control_Get_DataCnt (U16 CardNo, U16 GroupNo, U16* Counter)

■ **목적**

MotionSlave User Motion Control 명령, 해당 Group 에서 처리가 완료되지 않은 데이터 비트수를 확인합니다.
(PAC 의 최대 데이터 비트수는 800 비트이며, 축 카드의 최대 데이터 비트수는 100 비트입니다)

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
GroupNo	U16	번호 단위	Group 번호
Counter	U16*	수치 단위	처리되지 않은 데이터 비트 값

■ **예제**

```
U16 Status;
U16 CardNo=16, GroupNo =1;
U16 Counter;

Status = _ECAT_Slave_User_Motion_Control_Get_DataCnt (CardNo, GroupNo,
&Counter);
```

17.7 _ECAT_Slave_User_Motion_Control_Ralm

■ **포맷**

U16 PASCAL _ECAT_Slave_User_Motion_Control_Ralm (U16 CardNo, U16 GroupNo)

■ **목적**

MotionSlave User Motion Control 명령, 해당 Group 의 모든 축의 Alm 을 리셋합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
GroupNo	U16	번호 단위	Group 번호

■ **예제**

```
U16 Status;
U16 CardNo=16, GroupNo =1;

Status = _ECAT_Slave_User_Motion_Control_Ralm(CardNo, GroupNo);
```

17.8 _ECAT_Slave_User_Motion_Control_Svon

17

■ **포맷**

U16 PASCAL _ECAT_Slave_User_Motion_Control_Svon (U16 CardNo, U16 GroupNo, U16 ON_OFF)

■ **목적**

MotionSlave User Motion Control 명령, 해당 Group 의 모든 축을 Svon / off 합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
GroupNo	U16	번호 단위	Group 번호
ON_OFF	U16	수치 단위	ON_OFF 설명 0 : Svon Off 1 : Svon On

■ **예제**

```
U16 Status;
U16 CardNo=16, GroupNo =1;
U16 ON_OFF=1;

Status = _ECAT_Slave_User_Motion_Control_Svon(CardNo, GroupNo,ON_OFF);
```

17.9 _ECAT_Slave_User_Motion_Control_Get_Alm

■ **포맷**

U16 PASCAL _ECAT_Slave_User_Motion_Control_Get_Alm (U16 CardNo, U16 GroupNo, U16 *Alm)

■ **목적**

MotionSlave User Motion Control 명령, 해당 축의 현재 Alm 상태를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
GroupNo	U16	번호 단위	Group 번호
Alm	U16*	수치 단위	Alm 설명 0: Alm 없음 1: Group 의 한 축에 Alm 상태 발생

■ **예제**

```
U16 Status;
U16 CardNo=16, GroupNo =1;
U16 Alm ;

Status = _ECAT_Slave_User_Motion_Control_Get_Alm (CardNo, GroupNo, &Alm);
```

(이 페이지는 공란으로 비워둡니다)

DIO Slave API

18

다음은 DI/O 값 확인 또는 DO 값 설정, 오류 on 일 때 DO 에서 사전 설정값 출력 여부 설정 등 DIO Slave 관련 API 사용법에 대한 설명입니다.

18

DIO Slave API 테이블

함수의 명칭	설명
_ECAT_Slave_DIO_Get_Input_Value	DIOSlave 명령, Input 값을 확인합니다.
_ECAT_Slave_DIO_Get_Output_Value	DIOSlave 명령, Output 값을 확인합니다.
_ECAT_Slave_DIO_Set_Output_Value	DIOSlave 명령, Output 값을 설정합니다.
_ECAT_Slave_DIO_Get_Single_Input_Value	SlaveDIO 명령, 단일 Input 값을 확인합니다.
_ECAT_Slave_DIO_Get_Single_Output_Value	SlaveDIO 명령, 단일 Output 값을 확인합니다.
_ECAT_Slave_DIO_Set_Single_Output_Value	SlaveDIO 명령, 단일 Output 값을 설정합니다.
_ECAT_Slave_DIO_Set_Output_Error_Mode	SlaveDIO 명령, Error on 일 때 각 출력 지점에서 자동으로 사전 설정값으로 넘어가는 기능을 설정합니다.
_ECAT_Slave_DIO_Set_Output_Error_Value	SlaveDIO 명령, Error 모드가 on 이고, Error 상태를 트리거할 때 각 출력 지점에서 사전 설정한 출력 상태를 설정합니다.

18.1 _ECAT_Slave_DIO_Get_Input_Value

■ **포맷**

U16 PASCAL _ECAT_Slave_DIO_Get_Input_Value (U16 CardNo, U16 NodeID, U16 SlotNo, U16 *Value)

■ **목적**

DIOSlave 명령, Input 값을 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Value	U16*	수치 단위	수신한 데이터

■ **예제**

```
U16 Status;
U16 CardNo=16 , NodeID =1, SlotNo=0;
U16 Value;

Status =_ECAT_Slave_DIO_Get_Input_Value(CardNo, NodeID, SlotNo, &Value)
```

18.2 _ECAT_Slave_DIO_Get_Output_Value

■ **포맷**

U16 PASCAL _ECAT_Slave_DIO_Get_Output_Value (U16 CardNo, U16 NodeID, U16 SlotNo, U16 *Value)

■ **목적**

DIOSlave 명령, Output 값을 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Value	U16*	수치 단위	디지털 출력 모듈에서 출력한 값 확인

■ **예제**

```
U16 Status;
U16 CardNo=16 , NodeID =1, SlotNo=0;
U16 Value;

Status = _ECAT_Slave_DIO_Get_Output_Value(CardNo, NodeID, SlotNo, &Value);
```

18.3 _ECAT_Slave_DIO_Set_Output_Value

■ 포맷

U16 PASCAL _ECAT_Slave_DIO_Set_Output_Value (U16 CardNo, U16 NodeID, U16 SlotNo, U16 Value)

■ 목적

DIOSlave 명령, Output 값을 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Value	U16	수치 단위	디지털 출력 모듈에서 출력할 값 설정

■ 예제

```
U16 Status;
U16 CardNo=16 , NodeID =1, SlotNo=0;
U16 Value=0xFFFF;

Status = _ECAT_Slave_DIO_Set_Output_Value(CardNo, NodeID, SlotNo, Value);
```

18.4 _ECAT_Slave_DIO_Get_Single_Input_Value

■ **포맷**

U16 PASCAL _ECAT_Slave_DIO_Get_Single_Input_Value (U16 CardNo, U16 NodeID, U16 SlotNo, U16 BitNum, U16 *Value)

■ **목적**

SlaveDIO 명령, 단일 Input 값을 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
BitNum	U16	번호 단위	Input 지점 번호
Value	U16*	수치 단위	디지털 입력 모듈에서 수신한 값 확인

■ **예제**

```
U16 Status;
U16 CardNo=16, NodeID =1, SlotNo=0, BitNum=1;
U16 Value;

Status = _ECAT_Slave_DIO_Get_Single_Input_Value(CardNo, NodeID, SlotNo,
BitNum,&Value);
```

18.5 _ECAT_Slave_DIO_Get_Single_Output_Value

■ **포맷**

U16 PASCAL _ECAT_Slave_DIO_Get_Single_Output_Value (U16 CardNo, U16 NodeID, U16 SlotNo, U16 BitNum, U16 *Value)

■ **목적**

SlaveDIO 명령, 단일 Output 값을 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
BitNum	U16	번호 단위	Output 지점 번호
Value	U16*	수치 단위	디지털 출력 모듈에서 출력된 값을 확인합니다

■ **예제**

```
U16 Status;
U16 CardNo=16, NodeID =1, SlotNo=0, BitNum=1;
U16 Value;

Status = _ECAT_Slave_DIO_Get_Single_Output_Value(CardNo, NodeID, SlotNo,
BitNum,&Value);
```

18.6 _ECAT_Slave_DIO_Set_Single_Output_Value

■ **포맷**

U16 PASCAL _ECAT_Slave_DIO_Set_Single_Ouput_Value (U16 CardNo, U16 NodeID, U16 SlotNo, U16 BitNum, U16 Value)

■ **목적**

SlaveDIO 명령, 단일 Output 값을 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
BitNum	U16	번호 단위	Input 지점 번호
Value	U16	수치 단위	디지털 출력 모듈에서 출력한 값 설정

■ **예제**

```
U16 Status;
U16 CardNo=16, NodeID =1, SlotNo=0, BitNum=1;
U16 Value=1;

Status = _ECAT_Slave_DIO_Set_Single_Output_Value(CardNo, NodeID, SlotNo,
BitNum,Value);
```

18.7 _ECAT_Slave_DIO_Set_Output_Error_Mode

■ **포맷**

U16 PASCAL _ECAT_Slave_DIO_Set_Output_Error_Mode (U16 CardNo, U16 NodeID, U16 SlotNo, U16 BitMode)

■ **목적**

SlaveDIO 명령, Error on 일 때 각 출력 지점에서 자동으로 사전 설정값으로 넘어가는 기능을 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
BitMode	U16	번호 단위	Bit0 ~ 15 는 각각 출력 지점 Y0 ~ Y15 를 의미하며, 0 은 해당 기능 off, 1 은 해당 기능 on 을 의미합니다.

■ **예제**

```
U16 Status;
U16 CardNo=16, NodeID =1, SlotNo=0;
U16 BitMode = 0x0F;// 앞의 8 개 지점의 기능을 on 합니다

Status = _ECAT_Slave_DIO_Set_Output_Error_Mode(CardNo, NodeID, SlotNo,
BitMode);
```

18.8 _ECAT_Slave_DIO_Set_Output_Error_Value

■ **포맷**

U16 PASCAL _ECAT_Slave_DIO_Set_Output_Error_Value (U16 CardNo, U16 NodeID, U16SlotNo, U16 Value)

■ **목적**

SlaveDIO 명령, Error 모드가 on 이고, Error 상태를 트리거할 때 각 출력 지점에서 사전 설정한 출력 상태를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Value	U16	번호 단위	Bit0 ~ 15 는 각각 출력 지점 Y0 ~ Y15 를 의미하며, 0 은 출력 지점 OFF, 1 은 출력 지점 ON 을 의미합니다.

■ **예제**

```
U16 Status;
U16 CardNo=16, NodeID =1, SlotNo=0;
U16 Value = 0x0F;// 앞의 8 개 지점을 오류 시 출력 ON 으로 변경합니다

Status = _ECAT_Slave_DIO_Set_Output_Error_Value(CardNo, NodeID, SlotNo, Value);
```

AIO Slave API

19

다음은 AI/O 값 확인 또는 AO 값 설정 등 AIO Slave 관련 API 사용법에 대한 설명입니다.

19

AIO Slave API 테이블

함수의 명칭	설명
_ECAT_Slave_AIO_Get_Input_Value	AIOSlave 명령, Input 값을 확인합니다.
_ECAT_Slave_AIO_Set_Output_Value	AIOSlave 명령, Output 값을 설정합니다.
_ECAT_Slave_AIO_Get_Output_Value	AIOSlave 명령, Output 값을 확인합니다.

19.1 _ECAT_Slave_AIO_Get_Input_Value

■ **포맷**

U16 PASCAL _ECAT_Slave_AIO_Get_Input_Value (U16 CardNo, U16 NodeID, U16 SlotNo, U16 *Value)

■ **목적**

AIO Slave 명령, Input 값을 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Value	U16*	수치 단위	원격 모듈 데이터 확인

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, NodeID =1, SlotNo=0;
U16 Value;

Status= _ECAT_Slave_AIO_Get_Input_Value (CardNo, NodeID, SlotNo, &Value);
```

19.2 _ECAT_Slave_AIO_Set_Output_Value

■ **포맷**

U16 PASCAL _ECAT_Slave_AIO_Set_Output_Value (U16 CardNo, U16 NodeID, U16 SlotNo, U16 Value)

■ **목적**

AIO Slave 명령, Output 값을 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Value	U16	수치 단위	출력값은 0 ~ 65535 입니다

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, NodeID =1, SlotNo=0;
U16 Value=0x5ff;

Status= _ECAT_Slave_AIO_Set_Output_Value (CardNo, NodeID, SlotNo, Value);
```

19.3 _ECAT_Slave_AIO_Get_Output_Value

19

■ **포맷**

U16 PASCAL _ECAT_Slave_AIO_Get_Output_Value (U16 CardNo, U16 NodeID, U16 SlotNo, U16* Value)

■ **목적**

AIO Slave 명령, Output 값을 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Value	U16*	수치 단위	출력값은 0 ~ 65535 입니다

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, NodeID =1, SlotNo=0;
U16 Value=0;

Status= _ECAT_Slave_AIO_Get_Output_Value (CardNo, NodeID, SlotNo, &Value);
```

5621 Slave API

20

다음은 펄스 출력 및 수신 포맷 설정, 원점 접점의 역방향 여부 설정, Z 상 접점의 역방향 여부 설정, 원점 복귀의 특수 모드 사용 여부 설정, (+) (-) limit 접점의 역방향 여부 설정, Svon 접점의 역방향 여부 설정, IO 지점 상태 확인 등 5621 Slave 관련 API 의 사용법에 대한 설명입니다.

5621 Slave API 테이블

함수의 명칭	설명
_ECAT_Slave_R1_EC5621_Set_Output_Mode	5621Slave 명령, 출력 펄스 포맷을 설정합니다.
_ECAT_Slave_R1_EC5621_Set_Input_Mode	5621Slave 명령, 수신 펄스 포맷을 설정합니다.
_ECAT_Slave_R1_EC5621_Set_ORG_Inverse	5621Slave 명령, ORG 접점 모드의 역방향 여부를 설정합니다.
_ECAT_Slave_R1_EC5621_Set_QZ_Inverse	5621Slave 명령, QZ 접점 모드의 역방향 여부를 설정합니다.
_ECAT_Slave_R1_EC5621_Set_Home _SpMode	5621Slave 명령, Home 으로 돌아갈 경우, 특수 모드 사용 여부를 설정합니다.
_ECAT_Slave_R1_EC5621_Set_MEL_Inverse	5621Slave 명령, MEL 접점 모드의 역방향 여부를 설정합니다.
_ECAT_Slave_R1_EC5621_Set_PEL_Inverse	5621Slave 명령, PEL 접점 모드의 역방향 여부를 설정합니다.
_ECAT_Slave_R1_EC5621_Set_Svon_Inverse	5621Slave 명령, Svon 접점 모드의 역방향 여부를 설정합니다.
_ECAT_Slave_R1_EC5621_Set_Home_Slow_ Down	Slave5621 명령, 원점((+) (-) limit 모드는 무효)감지 후 역방향으로 이동할 경우, 1 단의 감속 시간과 정지 후 대기시간을 설정합니다. 동일한 방향으로 이동할 경우 WaitTime 만 유효합니다.
_ECAT_Slave_R1_EC5621_Get_IO_Status	5621Slave 명령, IO 상태를 확인합니다
_ECAT_Slave_R1_EC5621_Get_Single_IO _Status	5621Slave 명령, 단일 IO 의 상태를 확인합니다

20.1 _ECAT_Slave_R1_EC5621_Set_Output_Mode

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC5621_Set_Output_Mode (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 RangeMode)

■ **목적**

5621Slave 명령, 출력 펄스 포맷을 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Mode	U16	수치 단위	0 : A/B Phase 1 : CW/CCW 2 : PLS/DIR

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
U16 Mode=1;

Status= _ECAT_Slave_R1_EC5621_Set_Output_Mode (CardNo, AxisNo, SlotNo, Mode);
```

20

20.2 _ECAT_Slave_R1_EC5621_Set_Input_Mode

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC5621_Set_Input_Mode (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 RangeMode)

■ **목적**

5621Slave 명령, 수신 펄스 포맷을 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Mode	U16	수치 단위	0 : A/B Phase 1 : CW/CCW 2 : Command pulse

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
U16 Mode=1;

Status= _ECAT_Slave_R1_EC5621_Set_Input_Mode (CardNo, AxisNo, SlotNo, Mode);
```

20.3 _ECAT_Slave_R1_EC5621_Set_ORG_Inverse

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC5621_Set_ORG_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ **목적**

5621Slave 명령, ORG 접점 모드의 역방향 여부를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Enable	U16	옵션 단위	0: High 전위 트리거 1: Low 전위 트리거

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_EC5621_Set_ORG_Inverse (CardNo, AxisNo,SlotNo,
Enable);
```

20.4 _ECAT_Slave_R1_EC5621_Set_QZ_Inverse

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC5621_Set_QZ_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ **목적**

5621Slave 명령, QZ 접점 모드의 역방향 여부를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Enable	U16	옵션 단위	0: High 전위 트리거 1: Low 전위 트리거

■ **예제**

```
U16 Status = 0;
U16 CardNo=16 , AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_EC5621_Set_QZ_Inverse (CardNo, AxisNo,SlotNo,
Enable);
```

20.5 _ECAT_Slave_R1_EC5621_Set_Home_SpMode

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC5621_Set_Home_SpMode (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Mode)

■ **목적**

5621Slave 명령, Home 으로 돌아갈 경우, 특수 모드 사용 여부를 설정합니다 (특수 응용에 해당하며, 초저속으로 QZ 를 찾아냅니다).

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Mode	U16	수치 단위	0 : Mode 0 (Normal) 1 : Mode 1 (Special)

■ **예제**

```
U16 Status = 0;
U16 CardNo=16 , AxisNo =1, SlotNo=0;
U16 Mode=0;

Status= _ECAT_Slave_R1_EC5621_Set_Home_SpMode (CardNo, AxisNo,SlotNo, Mode);
```

20.6 _ECAT_Slave_R1_EC5621_Set_MEL_Inverse

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC5621_Set_MEL_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ **목적**

5621Slave 명령, MEL 접점 모드의 역방향 여부를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Enable	U16	옵션 단위	0: High 전위 트리거 1: Low 전위 트리거

■ **예제**

```
U16 Status = 0;
U16 CardNo=16 , AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_EC5621_Set_MEL_Inverse (CardNo, AxisNo, SlotNo, Enable);
```

20.7 _ECAT_Slave_R1_EC5621_Set_PEL_Inverse

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC5621_Set_PEL_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ **목적**

5621Slave 명령, PEL 접점 모드의 역방향 여부를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Enable	U16	옵션 단위	0: High 전위 트리거 1: Low 전위 트리거

■ **예제**

```
U16 Status = 0;
U16 CardNo=16 , AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_EC5621_Set_PEL_Inverse (CardNo, AxisNo, SlotNo,
Enable);
```

20.8 _ECAT_Slave_R1_EC5621_Set_Svon_Inverse

20

■ 포맷

U16 PASCAL _ECAT_Slave_R1_EC5621_Set_Svon_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ 목적

5621Slave 명령, Svon 접점 모드의 역방향 여부를 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Enable	U16	옵션 단위	0: High 전위 트리거 1: Low 전위 트리거

■ 예제

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_EC5621_Set_Svon_Inverse (CardNo, AxisNo, SlotNo, Enable);
```

20.9 _ECAT_Slave_R1_EC5621_Set_Home _Slow_Down

■ 포맷

U16 PASCAL _ECAT_Slave_R1_EC5621_Set_Home_Slow_Down (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable, U16 SlowDoneTime, U16 WaitTime)

■ 목적

Slave5621 명령, 원점((+) (-) limit 모드는 무효)감지 후 역방향으로 이동할 경우, 1 단의 감속 시간과 정지 후 대기시간을 설정합니다. 동일한 방향으로 이동할 경우 WaitTime 만 유효합니다. SlowDoneTime(ms): 1 단 정지 이전의 감속 시간 WaitTime(ms): 1 단 정지 이후의 대기 시간.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	축 번호
SlotNo	U16	번호 단위	Node 번호
Enable	U16	수치 단위	실행
SlowDoneTime	U16	수치 단위	1 단 정지 이전의 감속 시간
WaitTime	U16	수치 단위	1 단 정지 이후의 대기 시간

■ 예제

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0, Enable=1, SlowDoneTime=1, WaitTime=1;

Status= _ECAT_Slave_R1_EC5621_Set_Home_Slow_Down (CardNo, AxisNo, SlotNo, Enable, SlowDoneTime, WaitTime);
```

20

20.10 _ECAT_Slave_R1_EC5621_Get_IO_Status

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC5621_Get_IO_Status (U16 CardNo, U16 NodeID, U16 SlotNo, U16 *IOStatus)

■ **목적**

5621Slave 명령, IO 상태를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
IOStatus	U16*	수치 단위	IO 지점의 상태 확인

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, NodeID =7, SlotNo=0;
U16 IOStatus;

Status= _ECAT_Slave_R1_EC5621_Get_IO_Status (CardNo, NodeID, SlotNo,
&IOStatus);
```

20.11 _ECAT_Slave_R1_EC5621_Get_Single_IO_Status

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC5621_Get_Single_IO_Status (U16 CardNo, U16 NodeID, U16 SlotNo, U16 BitNo, U16 *IOStatus)

■ **목적**

5621Slave 명령, 단일 IO 의 상태를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
BitNo	U16	번호 단위	IO 지점 번호
IOStatus	U16*	수치 단위	IO 지점의 상태 확인

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, NodeID =7, SlotNo=0;
U16 IOStatus, BitNo=1;

Status= _ECAT_Slave_R1_EC5621_Get_Single_IO_Status (CardNo, NodeID, SlotNo, BitNo, &IOStatus);
```

(이 페이지는 공란으로 비워둡니다)

x62x Slave API

21

다음은 펄스 출력 및 수신 포맷 설정, 원점 접점의 역방향 여부 설정, Z 상 접점의 역방향 여부 설정, 원점 복귀의 특수 모드 사용 여부 설정, (+) (-) limit 접점의 역방향 여부 설정, Svon 접점의 역방향 여부 설정, IO 지점 상태 확인 등 x62x Slave 관련 API 의 사용법에 대한 설명입니다.

x62x Slave API 테이블

함수의 명칭	설명
_ECAT_Slave_R1_ECx62x_Set_Output_Mode	x62xSlave 명령, 출력 펄스 포맷을 설정합니다
_ECAT_Slave_R1_ECx62x_Set_Input_Mode	x62xSlave 명령, 수신 펄스 포맷을 설정합니다
_ECAT_Slave_R1_ECx62x_Set_ORG_Inverse	x62xSlave 명령, ORG 접점 모드의 역방향 여부를 설정합니다
_ECAT_Slave_R1_ECx62x_Set_QZ_Inverse	x62xSlave 명령, QZ 접점 모드의 역방향 여부를 설정합니다
_ECAT_Slave_R1_ECx62x_Set_Home_SpMode	x62xSlave 명령, Home 으로 돌아가기 시, 특수 모드 사용 여부를 설정합니다
_ECAT_Slave_R1_ECx62x_Set_MEL_Inverse	x62xSlave 명령, MEL 접점 모드의 역방향 여부를 설정합니다
_ECAT_Slave_R1_ECx62x_Set_PEL_Inverse	x62xSlave 명령, PEL 접점 모드의 역방향 여부를 설정합니다
_ECAT_Slave_R1_ECx62x_Set_Svon_Inverse	x62xSlave 명령, Svon 접점 모드의 역방향 여부를 설정합니다
_ECAT_Slave_R1_ECx62x_Set_Home_Slow_Down	x62xSlave 명령, Home 에서 limit 에 도달한 감속 시간을 설정합니다
_ECAT_Slave_R1_ECx62x_Get_IO_Status	x62xSlave 명령, IO 상태를 확인합니다
_ECAT_Slave_R1_ECx62x_Get_Single_IO_Status	x62xSlave 명령, 단일 IO 의 상태를 확인합니다

21.1 _ECAT_Slave_R1_ECx62x_Set_Output_Mode

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_ECx62x_Set_Output_Mode (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Mode)

■ **목적**

x62xSlave 명령, 출력 펄스 포맷을 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Mode	U16	수치 단위	0 : A/B Phase 1 : CW/CCW 2 : PLS/DIR

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
U16 Mode=1;

Status= _ECAT_Slave_R1_ECx62x_Set_Output_Mode (CardNo, AxisNo, SlotNo, Mode);
```

21.2 _ECAT_Slave_R1_ECx62x_Set_Input_Mode

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_ECx62x_Set_Input_Mode (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 RangeMode)

■ **목적**

x62xSlave 명령, 수신 펄스 포맷을 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Mode	U16	수치 단위	0 : A/B Phase 1 : CW/CCW 2 : PLS/DIR

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
U16 Mode=1;

Status= _ECAT_Slave_R1_ECx62x_Set_Input_Mode (CardNo, AxisNo, SlotNo, Mode);
```

21.3 _ECAT_Slave_R1_ECx62x_Set_ORG_Inverse

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_ECx62x_Set_ORG_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ **목적**

x62xSlave 명령, ORG 접점 모드의 역방향 여부를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Enable	U16	옵션 단위	0: High 전위 트리거 1: Low 전위 트리거

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_ECx62x_Set_ORG_Inverse (CardNo, AxisNo,SlotNo,
Enable);
```

21.4 _ECAT_Slave_R1_ECx62x_Set_QZ_Inverse

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_ECx62x_Set_QZ_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ **목적**

x62xSlave 명령, QZ 접점 모드의 역방향 여부를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Enable	U16	옵션 단위	0: High 전위 트리거 1: Low 전위 트리거

■ **예제**

```
U16 Status = 0;
U16 CardNo=16 , AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_ECx62x_Set_QZ_Inverse (CardNo, AxisNo,SlotNo,
Enable);
```

21.5 _ECAT_Slave_R1_ECx62x_Set_Home_SpMode

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_ECx62x_Set_Home_SpMode (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Mode)

■ **목적**

x62xSlave 명령, Home 으로 돌아가기 시, 특수 모드 사용 여부를 설정합니다 (특수 응용에 해당하며, 초저속으로 QZ 를 찾아냅니다).

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Mode	U16	수치 단위	0 : Mode 0 (Normal) 1 : Mode 1 (Special)

■ **예제**

```
U16 Status = 0;
U16 CardNo=16 , AxisNo =1, SlotNo=0;
U16 Mode=0;

Status= _ECAT_Slave_R1_ECx62x_Set_Home_SpMode (CardNo, AxisNo,SlotNo, Mode);
```

21.6 _ECAT_Slave_R1_ECx62x_Set_MEL_Inverse

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_ECx62x_Set_MEL_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ **목적**

x62xSlave 명령, MEL 접점 모드의 역방향 여부를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Enable	U16	옵션 단위	0: High 전위 트리거 1: Low 전위 트리거

■ **예제**

```
U16 CardNo=16 , AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_ECx62x_Set_MEL_Inverse (CardNo, AxisNo, SlotNo,
Enable);
```

21.7 _ECAT_Slave_R1_ECx62x_Set_PEL_Inverse

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_ECx62x_Set_PEL_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ **목적**

x62xSlave 명령, PEL 접점 모드의 역방향 여부를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Enable	U16	옵션 단위	0: High 전위 트리거 1: Low 전위 트리거

■ **예제**

```
U16 Status = 0;
U16 CardNo=16 , AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_ECx62x_Set_PEL_Inverse (CardNo, AxisNo, SlotNo,
Enable);
```

21.8 _ECAT_Slave_R1_ECx62x_Set_Svon_Inverse

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_ECx62x_Set_Svon_Inverse (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable)

■ **목적**

x62xSlave 명령, Svon 접점 모드의 역방향 여부를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Enable	U16	옵션 단위	0: High 전위 트리거 1: Low 전위 트리거

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, AxisNo =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_ECx62x_Set_Svon_Inverse (CardNo, AxisNo, SlotNo, Enable);
```

21.9 _ECAT_Slave_R1_ECx62x_Set_Home_Slow_Down

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_ECx62x_Set_Home_Slow_Down (U16 CardNo, U16 NodeID, U16 SlotNo, U16 Enable, U16 SlowDoneTime, U16 WaitTime)

■ **목적**

x62xSlave 명령, Home 에서 limit 에 도달한 감속 시간을 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Enable	U16	옵션 단위	0: off 1: 시동
SlowDoneTime	U16	밀리초	limit 에 도달한 감속 시간
WaitTime	U16	밀리초	limit 에 도달한 이후의 대기 시간

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, NodeID =7, SlotNo=0;
U16 Enable=1, SlowDoneTime = 1000, WaitTime= 1000;

Status= _ECAT_Slave_R1_ECx62x_Set_Home_Slow_Down (CardNo, NodeID, SlotNo, Enable, SlowDoneTime, WaitTime);
```

21.10 _ECAT_Slave_R1_ECx62x_Get_IO_Status

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_ECx62x_Get_IO_Status (U16 CardNo, U16 NodeID, U16 SlotNo, U16 *IOStatus)

■ **목적**

x62xSlave 명령, IO 상태를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
IOStatus	U16*	수치 단위	IO 지점의 상태 확인

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, NodeID =7, SlotNo=0;
U16 IOStatus;

Status= _ECAT_Slave_R1_ECx62x_Get_IO_Status (CardNo, NodeID, SlotNo,
&IOStatus);
```

21.11 _ECAT_Slave_R1_ECx62x_Get_Single_IO_Status

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_ECx62x_Get_Single_IO_Status (U16 CardNo, U16 NodeID, U16 SlotNo, U16 BitNo, U16 *IOStatus)

■ **목적**

x62xSlave 명령, 단일 IO 의 상태를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
BitNo	U16	번호 단위	IO 지점 번호
IOStatus	U16*	수치 단위	IO 지점의 상태 확인

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, NodeID =7, SlotNo=0;
U16 IOStatus, BitNo=1;

Status= _ECAT_Slave_R1_ECx62x_Get_Single_IO_Status (CardNo, NodeID, SlotNo, BitNo, &IOStatus);
```

(이 페이지는 공란으로 비워둡니다)

Delta Servo Slave API

22

다음은 드라이브 파라미터 입력 및 확인, 드라이브 회전속도 최대 제한 설정, 드라이브 Compare 관련 파라미터 입력 및 확인 등 Delta Servo Slave 관련 API 사용법에 대한 설명입니다.

22

Delta Servo Slave API 테이블

함수의 명칭	설명
_ECAT_Slave_DeltaServo_Write_Parameter	DeltaServoSlave 명령, 드라이브의 파라미터를 입력합니다.
_ECAT_Slave_DeltaServo_Read_Parameter	DeltaServoSlave 명령, 드라이브의 파라미터를 확인합니다.
_ECAT_Slave_DeltaServo_Read_Parameter_Info	DeltaServoSlave 명령, 드라이브의 파라미터 정보를 확인합니다.
_ECAT_Slave_DeltaServo_Set_Velocity_Limit	DeltaServoSlave 명령, 최대 제한 속도를 설정합니다.
_ECAT_Slave_DeltaServo_Set_Compare_Enable	Slave-Delta Servo 명령, 드라이브 Compare 파라미터와 상응하는 드라이브 P5-59 관련 파라미터를 입력합니다.
_ECAT_Slave_DeltaServo_Get_Compare_Enable	Slave-Delta Servo 명령, 입력한 드라이브 Compare 파라미터와 상응하는 드라이브 P5-59 관련 파라미터를 확인합니다.
_ECAT_Slave_DeltaServo_Set_Compare_Config	Slave-Delta Servo 명령, 드라이브 Compare 수량과 좌표를 입력합니다.

22.1 _ECAT_Slave_DeltaServo_Write_Parameter

■ **포맷**

U16 PASCAL _ECAT_Slave_DeltaServo_Write_Parameter (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Page, U16 Index, I32 WriteData)

■ **목적**

DeltaServoSlave 명령, 드라이브의 파라미터를 입력합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Page	U16	수치 단위	장치 (드라이브) 그룹 번호
Index	U16	수치 단위	드라이브 파라미터 그룹의 index
WriteData	I32	수치 단위	해당 그룹의 색인에 입력하고자 하는 데이터

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, AxisNo=1, SlotNo=0;
U16 Page =3, Index =0; // P3-00
I32 WriteData = 1;

Status= _ECAT_Slave_DeltaServo_Write_Parameter (CardNo, AxisNo, SlotNo, Page, Index, WriteData);
```

22

22.2 _ECAT_Slave_DeltaServo_Read_Parameter

■ **포맷**

U16 PASCAL _ECAT_Slave_DeltaServo_Read_Parameter (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Page, U16 Index, I32* ReadData)

■ **목적**

DeltaServoSlave 명령, 드라이브의 파라미터를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Page	U16	수치 단위	장치 (드라이브) 그룹 번호
Index	U16	수치 단위	드라이브 파라미터 그룹의 index
ReadData	I32*	수치 단위	해당 그룹 색인이 반환한 데이터

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, AxisNo = 1, SlotNo = 0;
U16 Page = 2, Index = 12; // P2-12
I32 ReadData = 0;

Status= _ECAT_Slave_DeltaServo_Read_Parameter (CardNo, AxisNo, SlotNo, Page, Index, &ReadData);
```

22.3 _ECAT_Slave_DeltaServo_Read_Parameter_Info

■ **포맷**

U16 PASCAL _ECAT_Slave_DeltaServo_Read_Parameter_Info (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Page, U16 Index, U16 *ParaType, U16 *DataSize, U16 *DataType)

■ **목적**

DeltaServoSlave 명령, 드라이브의 파라미터 정보를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Page	U16	수치 단위	장치 (드라이브) 그룹 번호
Index	U16	수치 단위	드라이브 파라미터 그룹의 index
ParaType	U16*	수치 단위	해당 수치의 의미는 다음과 같습니다 0: 해당 파라미터 사용 불가 1: 해당 파라미터 읽기만 가능 2: Servo On 일 때 해당 파라미터 설정 불가 3: 해당 파라미터 설정 후 정전 재실행해야 함 4: 정전 후 해당 파라미터를 저장하지 않음 5: 해당 파라미터에 특별한 사항 없음
DataSize	U16*	수치 단위	해당 파라미터 값의 크기 (Unit: Byte)
DataType	U16*	수치 단위	해당 수치의 의미는 다음과 같습니다 1: 해당 파라미터는 10 진수로 표시됩니다 2: 해당 파라미터는 16 진수로 표시됩니다 3: 해당 파라미터는 사용자 정의 포맷으로 나타나므로 드라이브 매뉴얼을 참조하시기 바랍니다 드라이브 매뉴얼을 참조하시기 바랍니다

■ 예제

```
U16 Status = 0;
U16 CardNo=16, AxisNo=1, SlotNo=0;
U16 Page =1;
U16 Index =0;
U16 ParaType = 0, DataSize=0, DataType = 0;

Status = _ECAT_Slave_DeltaServo_Read_Parameter_Info (CardNo, AxisNo, SlotNo,
Page, Index, &ParaType, &DataSize, &DataType);
```

22.4 _ECAT_Slave_DeltaServo_Set_Velocity_Limit

■ 포맷

U16 PASCAL _ECAT_Slave_DeltaServo_Set_Velocity_Limit (U16 CardNo, U16 AxisNo, U16 SlotNo, U32 LimitValue)

■ 목적

DeltaServoSlave 명령, 최대 제한 속도를 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
LimitValue	U32	수치 단위	회전속도 제한, 단위: rpm

■ 예제

```
U16 Status = 0;
U16 CardNo=16, AxisNo = 1, SlotNo = 0;
U32 LimitValue = 100;

Status=_ECAT_Slave_DeltaServo_Set_Velocity_Limit (CardNo, AxisNo, SlotNo,
LimitValue);
```

22.5 _ECAT_Slave_DeltaServo_Set_Compare_Enable

■ **포맷**

U16 PASCAL _ECAT_Slave_DeltaServo_Set_Compare_Enable (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 Enable, U16 CompareSource, U16 SignalLength, U16 SignalPolarity)

■ **목적**

Slave-Delta Servo 명령, 드라이브 Compare 파라미터와 상응하는 드라이브 P5-59 관련 파라미터를 입력합니다.

Enable: 드라이브 Compare 기능 사용(1) / 사용 중지(0).

CompareSource: 비교한 좌표 위치 소스

0: CaptureAxes (A2E 에는 해당 기능 없음)

1: AUX ENC (광학 자) 소스 만들기 (A2R-E 만 사용 가능)

2: 외부 입력 Pulse Command (A2E 에는 해당 기능 없음)

3: Main ENC (메인 인코더).

SignalLength: 출력 신호 길이, 단위 ms.

SignalPolarity: 출력 신호의 극.

0: NO (Normal Open) 항상 신호 on.

1: NC (Normal Close) 항상 신호 off.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	축 번호
SlotNo	U16	번호 단위	Node 번호
Enable	U16	수치 단위	드라이브 Compare 기능 사용(1) / 사용 중지(0)
CompareSource	U16	수치 단위	비교한 좌표 위치 소스
SignalLength	U16	수치 단위	출력 신호 길이
SignalPolarity	U16	수치 단위	출력 신호의 극

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, AxisNo = 1, SlotNo = 0, Enable=1, CompareSource=1,
SignalLength=1, SignalPolarity=1;

Status=_ECAT_Slave_DeltaServo_Set_Compare_Enable (CardNo, AxisNo, SlotNo,
Enable, CompareSource, SignalLength, SignalPolarity);
```

22.6 _ECAT_Slave_DeltaServo_Get_Compare_Enable

■ **포맷**

U16 PASCAL _ECAT_Slave_DeltaServo_Get_Compare_Enable (U16 CardNo, U16 AxisNo, U16 SlotNo, U16* Enable, U16* CompareSource, U16* SignalLength, U16* SignalPolarity)

■ **목적**

Slave-Delta Servo 명령, 입력한 드라이브 Compare 파라미터와 상응하는 드라이브 P5-59 관련 파라미터를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	축 번호
SlotNo	U16	번호 단위	Node 번호
Enable	U32	수치 단위	드라이브 Compare 기능 사용(1) / 사용 중지(0)
CompareSource	U32	수치 단위	비교한 좌표 위치 소스
SignalLength	U32	수치 단위	출력 신호 길이
SignalPolarity	U32	수치 단위	출력 신호의 극

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, AxisNo = 1, SlotNo = 0, Enable, CompareSource, SignalLength, SignalPolarity;

Status=_ECAT_Slave_ DeltaServo_Get_Compare_Enable (CardNo, AxisNo, SlotNo, &Enable, &CompareSource, &SignalLength, &SignalPolarity);
```

22.7 _ECAT_Slave_DeltaServo_Set_Compare_Config

■ **포맷**

U16 PASCAL _ECAT_Slave_DeltaServo_Set_Compare_Config (U16 CardNo, U16 AxisNo, U16 SlotNo, U16 CompareNum, I32* ComparePos)

■ **목적**

Slave-Delta Servo 명령, 드라이브 Compare 수량과 좌표를 입력합니다.
CompareNum: 좌표 배열 수량.
*ComparePos: 좌표 배열, 배열 길이는 CompareNum 이상이어야 합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
AxisNo	U16	번호 단위	축 번호
SlotNo	U16	번호 단위	Node 번호
CompareNum	U16	수치 단위	좌표 배열 수량
ComparePos	I32*	수치 단위	좌표 배열, 배열 길이는 CompareNum 이상이어야 합니다

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, AxisNo = 1, SlotNo = 0, CompareNum=1;
I32 ComparePos;

Status=_ECAT_Slave_DeltaServo_Set_Compare_Config(CardNo, AxisNo, SlotNo,
CompareNum, ComparePos);
```

(이 페이지는 공란으로 비워둡니다)

8124 Slave API

23

다음은 Input 관련 설정 등 8124 Slave 관련 API 사용법에 대한 설명입니다.

23

8124 Slave API 테이블

함수의 명칭	설명
_ECAT_Slave_R1_EC8124_Set _Input_RangeMode	8124Slave 전용 명령, Input 값 범위를 설정합니다.
_ECAT_Slave_R1_EC8124_Set_Input _ConvstFreq_Mode	8124Slave 전용 명령, Input 샘플링 주파수를 설정합니다.
_ECAT_Slave_R1_EC8124_Set_Input_Enable	8124Slave 전용 명령, Input 의 활성화 여부를 설정합니다.
_ECAT_Slave_R1_EC8124_Get_Input _RangeMode	8124Slave 전용 명령, 현재 Input 값 범위를 확인합니다.
_ECAT_Slave_R1_EC8124_Set_Input _AverageMode	8124Slave 전용 명령, Input 의 평균값 횟수를 설정합니다.

23.1 _ECAT_Slave_R1_EC8124_Set_Input _RangeMode

■ 포맷

U16 PASCAL _ECAT_Slave_R1_EC8124_Set_Input_RangeMode (U16 CardNo, U16 NodeID, U16 SlotNo, U16 RangeMode)

■ 목적

8124Slave 전용 명령, Input 값 범위를 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
RangeMode	U16	수치 단위	옵션은 0 ~ 1 입니다

수치	정의
0	출력 범위: ±5 V (예상값)
1	출력 범위: ±10 V

■ 예제

```
U16 Status = 0;
U16 CardNo=16 , NodeID =1, SlotNo=0;
U16 RangeMode=1;

Status= _ECAT_Slave_R1_EC8124_Set_Input_RangeMode (CardNo, NodeID,SlotNo,
RangeMode);
```

23.2 _ECAT_Slave_R1_EC8124_Set_Input _ConvstFreq_Mode

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC8124_Set_Input_ConvstFreq_Mode (U16 CardNo, U16 NodeID, U16 SlotNo, U16 RangeMode)

■ **목적**

8124Slave 전용 명령, Input 샘플링 주파수를 설정합니다.

■ **파라미터**

<table>
<tr><th>명칭</th><th>데이터 타입</th><th>단위</th><th>설명</th></tr>
<tr><td>CardNo</td><td>U16</td><td>번호 단위</td><td>카드 번호</td></tr>
<tr><td>NodeID</td><td>U16</td><td>번호 단위</td><td>NodeID 번호</td></tr>
<tr><td>SlotNo</td><td>U16</td><td>번호 단위</td><td>SlotID 번호</td></tr>
<tr><td>RangeMode</td><td>U16</td><td>수치 단위</td><td>옵션은 0 ~ 6 입니다
<table>
<tr><th>수치</th><th>전환 주파수 (kHz)</th></tr>
<tr><td>0</td><td>200</td></tr>
<tr><td>1</td><td>100</td></tr>
<tr><td>2</td><td>50</td></tr>
<tr><td>3</td><td>25</td></tr>
<tr><td>4</td><td>12.5</td></tr>
<tr><td>5</td><td>6.25</td></tr>
<tr><td>6</td><td>3.125</td></tr>
</table></td></tr>
</table>

■ **예제**

```
U16 Status = 0;
U16 CardNo=16 , NodeID =1, SlotNo=0;
U16 Mode=2;

Status= _ECAT_Slave_R1_EC8124_Set_Input_ConvstFreq_Mode (CardNo,
NodeID,SlotNo, Mode);
```

23.3 _ECAT_Slave_R1_EC8124_Set_Input_Enable

23

■ 포맷

U16 PASCAL _ECAT_Slave_R1_EC8124_Set_Input_Enable (U16 CardNo, U16 NodeID, U16 SlotNo, U16 Enable)

■ 목적

8124Slave 전용 명령, Input 의 활성화 여부를 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Enable	U16	수치 단위	0 : Disable 1 : Enable

■ 예제

```
U16 Status = 0;
U16 CardNo=16 , NodeID =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_EC8124_Set_Input_Enable (CardNo, NodeID,SlotNo,
Enable);
```

23.4 _ECAT_Slave_R1_EC8124_Get_Input _RangeMode

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC8124_Get_Input_RangeMode (U16 CardNo, U16 NodeID, U16 SlotNo, U16* RangeMode)

■ **목적**

8124Slave 전용 명령, 현재 Input 값 범위를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
RangeMode	U16*	수치 단위	옵션은 0 ~ 1 입니다 수치 / 정의 0 / 출력 범위: ±5 V (예상값) 1 / 출력 범위: ±10 V

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, NodeID =1, SlotNo=0;
U16 RangeMode;

Status= _ECAT_Slave_R1_EC8124_Get_Input_RangeMode (CardNo, NodeID,SlotNo,
&RangeMode);
```

23.5 _ECAT_Slave_R1_EC8124_Set_Input _AverageMode

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC8124_Set_Input_AverageMode (U16 CardNo, U16 NodeID, U16 SlotNo, U16 Avg_Times)

■ **목적**

AIOSlave 명령, Input 의 평균값 횟수를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Avg_Times	U16	수치 단위	평균값 횟수, 값의 범위는 1 ~ 127 입니다

■ **예제**

```
U16 Status = 0;
U16 CardNo=16 , NodeID =1, SlotNo=0;
U16 Avg_Times =5;

Status= _ECAT_Slave_R1_EC8124_Set_Input_AverageMode (CardNo,
NodeID,SlotNo, Avg_Times);
```

9144 Slave API

24

다음은 Output 관련 설정 등 9144 Slave 관련 API 사용법에 대한 설명입니다.

9144 Slave API 테이블

함수의 명칭	설명
_ECAT_Slave_R1_EC9144_Set_Output _RangeMode	9144Slave 전용 명령, Output 의 범위를 설정합니다.
_ECAT_Slave_R1_EC9144_Set_Output _Enable	9144Slave 전용 명령, Output 의 활성화 여부를 설정합니다.
_ECAT_Slave_R1_EC9144_Get_Output _ReturnCode	9144Slave 전용 명령, Output 의 반환 상태를 확인합니다.

24.1 _ECAT_Slave_R1_EC9144_Set_Output _RangeMode

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC9144_Set_Output_RangeMode (U16 CardNo, U16 NodeID, U16 SlotNo, U16 RangeMode)

■ **목적**

9144Slave 전용 명령, Output 의 범위를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
RangeMode	U16	수치 단위	옵션은 0 ~ 6 입니다

수치	정의
0	출력 범위: 0 ~ 5 V (예비 설정값)
1	출력 범위: 0 ~ 10 V
2	출력 범위: ±5 V
3	출력 범위: ±10 V
4	출력 범위: 4 ~ 20 mA
5	출력 범위: 0 ~ 20 mA
6	출력 범위: 0 ~ 24 mA

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, NodeID =1, SlotNo=0;
U16 RangeMode=3;

Status= _ECAT_Slave_R1_EC9144_Set_Output_RangeMode (CardNo,
NodeID,SlotNo, RangeMode);
```

24.2 _ECAT_Slave_R1_EC9144_Set_Output_Enable

24

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC9144_Set_Output_Enable (U16 CardNo, U16 NodeID, U16 SlotNo, U16 Enable)

■ **목적**

9144Slave 전용 명령, Output 의 활성화 여부를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Enable	U16	수치 단위	0 : Disable 1 : Enable

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, NodeID =1, SlotNo=0;
U16 Enable=1;

Status= _ECAT_Slave_R1_EC9144_Set_Output_Enable (CardNo, NodeID, SlotNo, Enable);
```

24.3 _ECAT_Slave_R1_EC9144_Get_Output _ReturnCode

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC9144_Get_Output_ReturnCode (U16 CardNo, U16 NodeID, U16 SlotNo, U16 *ReturnCode)

■ **목적**

9144Slave 전용 명령, Output 의 반환 상태를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
*ReturnCode	U16*	수치 단위	_ECAT_Slave_AIO_Get_Output_ReturnCode 리스트 상세 보기

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, NodeID =1, SlotNo=0;
U16 ReturnCode;
Status= _ECAT_Slave_R1_EC9144_Get_Output_ReturnCode (CardNo, NodeID,
SlotNo, &ReturnCode);
```

_ECAT_Slave_AIO_Get_Output_ReturnCode 리스트

Bit	설명
0	Iout_3 error: 전류 출력 채널 3 에서 고장이 발생할 경우 해당 비트는 1 입니다.
1	Iout_2 error: 전류 출력 채널 2 에서 고장이 발생할 경우 해당 비트는 1 입니다.
2	Iout_1 error: 전류 출력 채널 1 에서 고장이 발생할 경우 해당 비트는 1 입니다.
3	Iout_0 error: 전류 출력 채널 0 에서 고장이 발생할 경우 해당 비트는 1 입니다.
4	Vout_3 error: 전압 출력 채널 3 에서 고장이 발생할 경우 해당 비트는 1 입니다.
5	Vout_2 error: 전압 출력 채널 2 에서 고장이 발생할 경우 해당 비트는 1 입니다.
6	Vout_1 error: 전압 출력 채널 1 에서 고장이 발생할 경우 해당 비트는 1 입니다.
7	Vout_0 error: 전압 출력 채널 0 에서 고장이 발생할 경우 해당 비트는 1 입니다.
8	온도가 너무 높음: 컨버전 칩의 온도가 150 도 이상에 달하며, 해당 비트는 1 입니다.
9	Ramp active: 임의의 출력 채널에 전압 스윙이 나타나며, 해당 비트는 1 입니다.
10	PEC error: 해당 비트는 1 이며, SPI 인터페이스를 통해 수신한 마지막 데이터에 PEC 오류가 존재함을 의미합니다.
11	User toggle: 해당 비트는 데이터 통신 점검에 사용할 수 있습니다.
12	DC-DC3 : 전류 모드에서 채널 3 의 DC-DC 변환기가 전압을 유지할 수 없을 경우 해당 비트는 1 입니다. 이 상황에서 Iout_3 error 는 동시에 1 이 됩니다. 전압 모드에서 채널 3 의 DC-DC 변환기가 기대 전압인 15 V 에 도달할 수 없을 경우 해당 비트는 1 입니다.
13	DC-DC2 : 전류 모드에서 채널 2 의 DC-DC 변환기가 전압을 유지할 수 없을 경우 해당 비트는 1 입니다. 이 상황에서 Iout_2 error 는 동시에 1 이 됩니다. 전압 모드에서 채널 2 의 DC-DC 변환기가 기대 전압인 15 V 에 도달할 수 없을 경우 해당 비트는 1 입니다.
14	DC-DC1 : 전류 모드에서 채널 1 의 DC-DC 변환기가 전압을 유지할 수 없을 경우 해당 비트는 1 입니다. 이 상황에서 Iout_1error 는 동시에 1 이 됩니다. 전압 모드에서 채널 1 의 DC-DC 변환기가 기대 전압인 15 V 에 도달할 수 없을 경우 해당 비트는 1 입니다.
15	DC-DC0 : 전류 모드에서 채널 0 의 DC-DC 변환기가 전압을 유지할 수 없을 경우 해당 비트는 1 입니다. 이 상황에서 Iout_0 error 는 동시에 1 이 됩니다. 전압 모드에서 채널 0 의 DC-DC 변환기가 기대 전압인 15 V 에 도달할 수 없을 경우 해당 비트는 1 입니다.

Slave Record Data API 25

다음은 Record 관련 설정 등 Slave Record Data 관련 API 사용법에 대한 설명입니다.

25

Slave Record Data API 테이블

함수의 명칭	설명
_ECAT_Slave_Record_Data_Set_Type	Slave 전용 명령, 각 축의 Record 데이터 종류를 설정합니다.
_ECAT_Slave_Record_Data_Set_Enable	Slave 전용 명령, 각 축의 Record 기능 실행 여부를 설정합니다.
_ECAT_Slave_Record_Data_Get_Cnt	Slave 전용 명령, 각 축의 현재 Record 데이터 비트수를 확인합니다.
_ECAT_Slave_Record_Data_ReadData	Slave 전용 명령, 각 축의 Record 데이터를 확인합니다.
_ECAT_Slave_Record_Clear_Data	Slave 전용 명령, 현재 저장된 각 축의 Record 데이터를 삭제합니다.
_ECAT_Slave_Record_Multi_Set_Enable	Slave 전용 명령, 다축의 Record 기능 실행 여부를 동시에 설정합니다.
_ECAT_Slave_Record_Multi_Clear_Data	Slave 전용 명령, 다축의 현재 저장된 Record 데이터를 동시에 삭제합니다.

25.1 _ECAT_Slave_Record_Set_Type

25

■ 포맷

U16 PASCAL _ECAT_Slave_Record_Set_Type (U16 CardNo, U16 NodeID, U16 SlotNo, U16 MonitorIndex, U16 IOType, U16 Index, U16 SubIndex)

■ 목적

Slave 전용 명령, 각 축의 Record 데이터 종류를 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
MonitorIndex	U16	단위	옵션은 0 ~ 8 입니다 (각 Slot 마다 최대 동시에 8 개의 OD 기록 가능)
IOType	U16	수치 단위	모니터링할 OD 코드 형태 설정 (0: Input, 1: Output)
Index	U16	수치 단위	모니터링할 OD 코드 (CANopen 참조)
SubIndex	U16	수치 단위	모니터링할 OD 코드 Sub 코드 (CANopen 참조)

■ 예제

```
U16 Status = 0;
U16 CardNo=16 , NodeID =1, SlotNo=0;
U16 MonitorIndex =0;
U16 IOType =0, Index=1, SubIndex=1;

Status= _ECAT_Slave_Record_Set_Type (CardNo, NodeID, SlotNo, MonitorIndex,
IOType, Index, SubIndex);
```

25

25.2 _ECAT_Slave_Record_Set_Enable

■ **포맷**

U16 PASCAL _ECAT_Slave_Record_Set_Enable (U16 CardNo, U16 NodeID, U16 SlotNo, U16 Enable)

■ **목적**

Slave 전용 명령, 각 축의 Record 기능 실행 여부를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Enable	U16	수치 단위	Enable 값은 Bit 에 따라 해당 Monitor Index 를 확인합니다

■ **예제**

```
U16 Status = 0;
U16 CardNo=16 , NodeID =1, SlotNo=0;
U16 Enable =0x07; // Monitor 0, 1, 2 세개의 Index 확인

Status= _ECAT_Slave_Record_Set_Enable (CardNo, NodeID, SlotNo, Enable);
```

25.3 _ECAT_Slave_Record_Get_Cnt

■ **포맷**

U16 PASCAL _ECAT_Slave_Record_Get_Cnt (U16 CardNo, U16 NodeID, U16 SlotNo, U16 *Cnt)

■ **목적**

Slave 전용 명령, 각 축의 현재 Record 데이터 비트수를 확인합니다 (최대 800 비트).

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Cnt	U16*	수치 단위	각 축의 현재 Record 데이터 비트수 확인

■ **예제**

```
U16 Status = 0;
U16 CardNo=16 , NodeID =1, SlotNo=0;
U16 Cnt ;

Status= _ECAT_Slave_Record_Get_Cnt (CardNo, NodeID, SlotNo, &Cnt);
```

25.4 _ECAT_Slave_Record_Read_Data

25

■ **포맷**

U16 PASCAL _ECAT_Slave_Record_Read_Data (U16 CardNo, U16 NodeID, U16 SlotNo, U32 *Data)

■ **목적**

Slave 전용 명령, 각 축의 Record 데이터를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Data	U32*	수치 단위	각 축의 Record 데이터를 확인합니다. Data 는 한번에 8 개의 필드를 회신하므로, 1 개의 Data[8]를 사용하여 데이터를 확인해야 합니다. MonitorIndex 에서 데이터 종류를 설정하지 않았을 경우, 0 이라고 반환됩니다.

■ **예제**

```
U16 Status = 0;
U16 CardNo=16 , NodeID=1, SlotNo=0;
U16 MonitorIndex =0;
U32 Data[8];

Status= _ECAT_Slave_Record_Read_Data (CardNo, NodeID, SlotNo, Data);
```

25.5 _ECAT_Slave_Record_Clear_Data

■ **포맷**

U16 PASCAL _ECAT_Slave_Record_Clear_Data (U16 CardNo, U16 NodeID, U16 SlotNo)

■ **목적**

Slave 전용 명령, 현재 저장된 각 축의 Record 데이터를 삭제합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호

■ **예제**

```
U16 Status = 0;
U16 CardNo=16 , NodeID=1, SlotNo=0;

Status= _ECAT_Slave_Record_Clear_Data (CardNo, NodeID, SlotNo);
```

25

25.6 _ECAT_Slave_Record_Multi_Set_Enable

■ **포맷**

U16 PASCAL _ECAT_Slave_Record_Multi_Set_Enable (U16 CardNo, U16 NodeNum, U16 *NodeIDArray, U16 *SlotNoArray, U16 Enable)

■ **목적**

Slave 전용 명령, 다축의 Record 기능 실행 여부를 동시에 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeNum	U16	수치 단위	Node 수량 설정
NodeIDArray	U16*	번호 단위 배열	다축 Record 기능의 Node 번호 ID 배열 설정, 수량은 AxisNum NodeIDArray [0] 첫 번째 세트의 Node 번호를 배열합니다 NodeIDArray [1] 두 번째 세트의 Node 번호를 배열합니다 … 이 러한 방식으로 유추합니다.
NodeIDArray	U16*	번호 단위 배열	SlotID 번호 배열, 수량은 AxisNum 입니다
Enable	U16	옵션 단위	Enable 값은 Bit 에 따라 해당 Monitor Index 를 확인합니다

■ **예제**

```
U16 Status = 0;
U16 CardNo=16 , NodeNum = 2, NodeID[2]={0, 1}, SlotNo[2]={0, 0};
U16 Enable =0x07; // Monitor 0, 1, 2 세개의 Index 확인

// 다축 Record 기능 사용
Status= _ECAT_Slave_Record_Multi_Set_Enable (CardNo, NodeNum, NodeIDArray, SlotIDArray, Enable);
```

25.7 _ECAT_Slave_Record_Multi_Clear_Data

■ **포맷**

U16 PASCAL _ECAT_Slave_Record_Multi_Clear_Data (U16 CardNo, U16 NodeNum, U16 *NodeIDArray, U16 *SlotNoArray)

■ **목적**

Slave 전용 명령, 다축의 현재 저장된 Record 데이터를 동시에 삭제합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeNum	U16	수치 단위	Node 수량 설정
NodeIDArray	U16*	번호 단위 배열	다축 Record 삭제 기능의 Node 번호 ID 배열 설정, 수량은 AxisNum NodeIDArray [0] 첫 번째 세트의 Node 번호를 배열합니다 NodeIDArray [1] 두 번째 세트의 Node 번호를 배열합니다 … 이 러한 방식으로 유추합니다.
NodeIDArray	U16*	번호 단위 배열	SlotID 번호 배열, 수량은 AxisNum 입니다

■ **예제**

```
U16 Status = 0;
U16 CardNo=16 , NodeNum = 2, NodeIDArray[2] = {0, 1}, SlotIDArray[2] = {0, 0};
// 다축 Record 데이터 삭제
Status= _ECAT_Slave_Record_Multi_Clear_Data (CardNo, NodeNum, NodeIDArray,
SlotIDArray);
```

(이 페이지는 공란으로 비워둡니다)

Master 축 카드 전용 API 26

다음은 축 카드에서의 GPIO 출력 지점 상태 설정 및 확인, 축 카드에서의 GPIO 입력 지점 상태 확인 등 Master 축 카드 전용 관련 API 사용법에 대한 설명입니다.

26

Master 축 카드 전용 API 테이블

함수의 명칭	설명
_ECAT_GPIO_Set_Output	축 카드의 GPIO 출력 지점 상태를 설정합니다.
_ECAT_GPIO_Get_Output	축 카드의 GPIO 출력 지점 상태를 확인합니다.
_ECAT_GPIO_Get_Input	축 카드의 GPIO 입력 지점 상태를 확인합니다.

26.1 _ECAT_GPIO_Set_Output

■ **포맷**

U16 PASCAL _ECAT_GPIO_Set_Output (U16 CardNo, U16 Data)

■ **목적**

축 카드의 GPIO 출력 지점 상태를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
Data	U16	수치 단위	지정한 출력 지점 상태

■ **예제**

```
U16 Status = 0;
U16 CardNo, Data;

CardNo = 0;
Data = 0xF;
Status= _ECAT_GPIO_Set_Output( CardNo, Data );
```

26.2 _ECAT_GPIO_Get_Output

■ **포맷**

U16 PASCAL _ECAT_GPIO_Get_Output (U16 CardNo, U16* Data)

■ **목적**

축 카드의 GPIO 출력 지점 상태를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
Data	U16*	수치 단위	반환한 출력 지점 상태

■ **예제**

```
U16 Status = 0;
U16 CardNo, Data;

CardNo = 0;
Status= _ECAT_GPIO_Get_Output( CardNo, &Data );
```

26.3 _ECAT_GPIO_Get_Input

■ **포맷**

U16 PASCAL _ECAT_GPIO_Get_Input (U16 CardNo, U16* Data)

■ **목적**

축 카드의 GPIO 입력 지점 상태를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
Data	U16*	수치 단위	반환한 입력 지점 상태

■ **예제**

```
U16 Status = 0;
U16 CardNo, Data;

CardNo = 0;
Status= _ECAT_GPIO_Get_Input( CardNo, &Data );
```

Master Compare API 27

다음은 펄스 입력 모듈의 phase mode 설정, Compare 관련 설정 등 Master Compare 관련 API 사용법에 대한 설명입니다.

27

Master Compare API 테이블

함수의 명칭	설명
_ECAT_Compare_Set_Channel_Position	해당 Channel 의 새로운 Position 계수값을 설정합니다
_ECAT_Compare_Get_Channel_Position	해당 Channel 의 현재 Position 계수값을 확인합니다
_ECAT_Compare_Set_Ipulser_Mode	펄스 인터페이스 모듈의 입력 phase mode 를 설정합니다
_ECAT_Compare_Set_Channel_Direction	해당 Channel 의 펄스 방향을 설정합니다
_ECAT_Compare_Set_Channel_Trigger_Time	Trigger 활성화의 지속시간을 설정합니다
_ECAT_Compare_Set_Channel_One_Shot	Trigger 를 1 회 활성화 상태로 설정합니다
_ECAT_Compare_Set_Channel_Source	비교 소스를 설정합니다
_ECAT_Compare_Set_Channel_Enable	Compare 기능의 On / Off 를 설정합니다
_ECAT_Compare_Channel0_Position	Compare Type0 실행
_ECAT_Compare_Set_Channel0_Trigger_By_GPIO	Compare Type0 의 트리거 조건을 GPIO 가 제어하도록 설정합니다
_ECAT_Compare_Set_Channel1_Output_Enable	Compare Type1 의 출력 On / Off 를 설정합니다
_ECAT_Compare_Set_Channel1_Output_Mode	Compare Type1 의 출력 모드를 설정합니다
_ECAT_Compare_Get_Channel1_IO_Status	Compare Type1 의 I/O 상태를 확인합니다.
_ECAT_Compare_Set_Channel1_GPIO_Out	Compare Type1 의 GPIO 출력 bit 상태(Pin15)를 설정합니다
_ECAT_Compare_Set_Channel1_Position_Table	Compare Type1 의 관련 데이터를 설정합니다
_ECAT_Compare_Set_Channel1_Position_Table_Level	Compare Type1 의 Level 관련 데이터를 설정합니다
_ECAT_Compare_Get_Channel1_Position_Table_Count	이미 실행한 Trigger 의 횟수(Compare Type1 사용)를 확인합니다
_ECAT_Compare_Set_Channel_Polarity	Compare Trigger 의 레벨을 설정합니다
_ECAT_Compare_Reuse_Channel1_Position_Table	지난 번에 설정한 Compare 조건으로 Compare Type1 을 다시 한 번 실행합니다
_ECAT_Compare_Reuse_Channel1_Position_Table_Level	지난 번에 설정한 Compare 조건으로 Compare Type1(Level 모드)을 다시 한 번 실행합니다

27.1 _ECAT_Compare_Set_Channel_Position

■ **포맷**

U16 PASCAL _ECAT_Compare_Set_Channel_Position (U16 CardNo, U16 CompareChannel, I32 Position)

■ **목적**

해당 QEP Channel 의 새로운 계수값을 설정합니다. (이 수치와 모듈 Position 의 의미는 다릅니다)

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 15 입니다
Compare_Channel	U16	번호 단위	Channel No 는 0 ~ 1 입니다
Position	I32	수치 단위	설정하고자 하는 QEP Channel 의 새로운 계수값

■ **예제**

```
U16 CardNo = 0;
U16 Compare_Channel = 0;
I32 position = 0;

U16 status = _ECAT_Compare_Set_Channel_Position (CardNo, compare_channel, position);
```

27

27.2 _ECAT_Compare_Get_Channel_Position

■ **포맷**

U16 PASCAL _ECAT_Compare_Get_Channel_Position (U16 CardNo, U16 compare_Channel, I32 *position)

■ **목적**

해당 QEP Channel 의 현재 계수값을 확인합니다. (이 수치와 모듈 Position 의 의미는 다릅니다)

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 15 입니다
Compare_Channel	U16	번호 단위	Channel No 는 0 ~ 1 입니다
Position	I32*	펄스 수	해당 QEP Channel 의 현재 계수값을 확인합니다

■ **예제**

```
U16 CardNo = 0;
U16 Compare_channel = 0;
I32 position;

U16 status = _ECAT_Compare_Get_Channel_Position (CardNo, compare_channel, &position);
```

27.3 _ECAT_Compare_Set_Ipulser_Mode

■ **포맷**

U16 PASCAL _ECAT _Compare_Set _Ipulser_Mode (U16 CardNo, U16 mode)

■ **목적**

펄스 인터페이스 모듈의 입력 phase mode 를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 15 입니다
Mode	U16	수치 단위	0 : AB Phase 1 : CW/CCW

■ **예제**

```
U16 CardNo = 0;
U16 mode = 0;       //AB Phase

U16 status = _ECAT _Compare_Set _Ipulser_Mode (CardNo, mode);
```

27.4 _ECAT_Compare_Set_Channel_Direction

■ **포맷**

U16 PASCAL _ECAT _Compare_Set _Channel_Direction (U16 CardNo,
U16 compare_channel, U16 dir)

■ **목적**

해당 QEP Channel 의 펄스 방향을 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 15 입니다
Compare_channel	U16	번호 단위	Channel No 는 0 ~ 1 입니다
Dir	U16	수치 단위	0：Normal 1：Inverse

■ **예제**

```
U16 CardNo = 0;
U16 compare_channel = 0;
U16 dir = 1; //Inverse

U16 status = _ECAT_Compare_Set _Channel_Direction (CardNo, compare_channel,
dir);
```

27.5 _ECAT_Compare_Set_Channel_Trigger_Time

■ **포맷**

U16 PASCAL _ECAT _Compare_Set _Channel_Trigger_Time (U16 CardNo, U16 compare_channel, U32 time_us)

■ **목적**

Trigger 활성화의 지속시간을 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 15 입니다
Compare_channel	U16	번호 단위	Channel No 는 0 ~ 1 입니다
Time_us	U32	시간 단위	각 Trigger 활성화의 지속시간을 입력합니다

■ **예제**

```
U16 CardNo = 0;
U16 compare_channel = 0;
U16 time_us = 20; //20us

U16 status = _ECAT_Compare_Set _Channel_Trigger_Time(CardNo,
compare_channel, time_us);
```

비고:

1. Compare Type0 상태에서 Trigger Time 의 최소 설정 값은 1us 입니다. 입력 값이 0 이면 0.8us 라고 나타납니다.
2. Compare Type1 상태에서 Trigger Time 의 최소 설정 값은 3us 입니다. 입력 값이 3 미만이면 3us 라고 나타납니다.

27

27.6 _ECAT_Compare_Set_Channel_One_Shot

■ **포맷**

U16 PASCAL _ECAT_Compare_Set _Channel_One_Shot (U16 CardNo,
U16 compare_channel)

■ **목적**

Trigger 활성화를 1 회 실행하는 것으로 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 5 입니다
Compare_channel	U16	번호 단위	Channel No 는 0 ~ 1 입니다

■ **예제**

```
U16 CardNo = 0;
U16 compare_channel=0;

U16 status = _ECAT_Compare_Set _Channel_One_Shot (CardNo, compare_channel);
```

27.7 _ECAT_Compare_Set_Channel_Source

■ **포맷**

U16 PASCAL _ECAT_Compare_Set _Channel_Source (U16 CardNo,
U16 compare_channel, U16 source)

■ **목적**

비교 소스를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 15 입니다
Compare_channel	U16	번호 단위	Channel No 는 0 ~ 1 입니다
Source	U16	번호 단위	0 : QEP1 1 : QEP2

■ **예제**

```
U16 CardNo = 0;
U16 compare_channel = 0;
U16 source = 0;

U16 status = _ECAT_Compare_Set _Channel_source (CardNo, compare_channel,
source);
```

27.8 _ECAT_Compare_Set_Channel_Enable

■ **포맷**

U16 PASCAL _ECAT_Compare_Set _Channel_Enable (U16 CardNo,
U16 compare_channel, U16 enable)

■ **목적**

FPGA 의 Compare 기능 on / off 를 설정합니다. 비교 배열을 on (Channel1_output_enable) 하지 않을 경우, 실제 Compare 는 비교 가능한 위치가 없으므로 트리거가 실행되지 않습니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 15 입니다
Compare_channel	U16	번호 단위	Channel No 는 0 ~ 1 입니다
enable	U16	수치 단위	0: Compare 기능 off. 1: Compare 기능 on.

■ **예제**

```
U16 CardNo = 0, ;
U16 compare_channel = 0;
U16 enable = 1;//Compare 기능 on

U16 status = _ECAT_Compare_Set Channel_Enable (CardNo, compare_channel,
enable);
```

27.9 _ECAT_Compare_Channel0_Position

■ **포맷**

U16 PASCAL _ECAT_Compare_Channel0_Position (U16 CardNo, I32 Start, U16 Dir, U16 Interval, U32 TriggerCount);

■ **목적**

Compare Type 0 을 실행합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 15 입니다
Start	I32	수치 단위	Compare 의 시작 위치
Dir	U16	수치 단위	0: 순방향 1: 역방향
Interval	U16	펄스 수	펄스 Compare 의 간격 수
Trigger_cnt	U32	수치 단위	Trigger Compare 의 총수

■ **예제**

```
U16 CardNo = 0;
I32 start = 100000;
U16 dir = 0;
U16 Interval = 10; //10 pulse
U32 trigger_cnt = 50000;

U16 status=_ECAT_Compare_Channel0_Position(CardNo, start, dir, interval,
trigger_cnt);
```

27.10 _ECAT_Compare_Set_Channel0_Trigger _By_GPIO

27

■ **포맷**

U16 PASCAL _ECAT_Compare_Set_Channel0_Trigger_By_GPIO (U16 CardNo, U16 dir, U16 interval, I32 trigger_cnt)

■ **목적**

CompareType0 의 트리거 조건을 GPIO 가 제어하도록 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 15 입니다
Dir	U16	수치 단위	0: 순방향 1: 역방향
Interval	U16	수치 단위	펄스 Compare 의 간격 수
Trigger _cnt	I32	수치 단위	Trigger Compare 의 총수

■ **예제**

```
U16 CardNo = 0;
U16 Dir = 1;
U16 Interval = 10;
I32 trigger_cnt = 50000;

U16 status = _ECAT_Compare_Set_Channel0_Trigger_By_GPIO (CardNo, Dir, Interval, trigger_cnt);
```

27.11 _ECAT_Compare_Set_Channel1_Output_Enable

■ **포맷**

U16 PASCAL _ECAT_Compare_Set_Channel1_Output_Enable (U16 CardNo, U16 on_off)

■ **목적**

Compare Position 의 저장 배열을 사용하면 FPGA 내부의 Compare 기능이 실행되어 실제 위치와 배열의 저장 위치를 통해 순서대로 비교하기 시작합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 15 입니다
On_off	U16	수치 단위	0 : Off 1 : On

■ **예제**

```
U16 CardNo = 0;
U16 on_off = 1;

U16 status = _ECAT_Compare_Set_Channel1_Output_Enable (CardNo, on_off);
```

27

27.12 _ECAT_Compare_Set_Channel1_Output_Mode

■ 포맷

U16 PASCAL _ECAT_Compare_Set_Channel1_Output_Mode (U16 CardNo, U16 Mode)

■ 목적

Compare Type1 은 출력 모드를 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 15 입니다
Mode	U16	수치 단위	0: Normal 모드 1: 사용자 모드

■ 예제

```
U16 CardNo = 0;
U16 mode = 1;
U16 status = _ECAT_Compare_Set_Channel1_Output_Mode (CardNo, mode);
```

■ Normal 모드

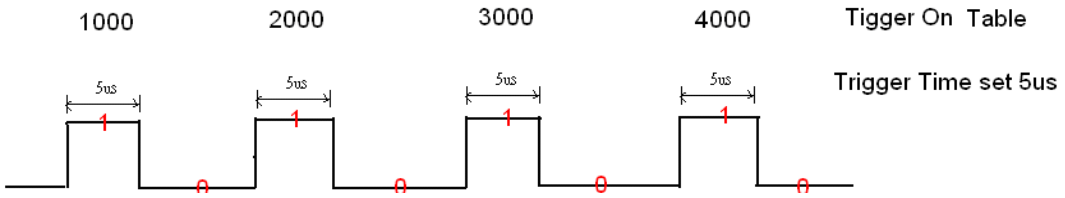


그림 27.12.1 Compare Output Normal 모드

※Trigger On 의 위치 테이블은 API "_DMC_01_channel1_position_compare_table"의 "table_size" 파라미터에서 설정합니다.

※Trigger 시간은 API "_DMC_01_set_compare_channel_trigger_time"의 "time_us" 파라미터에서 설정합니다.

■ 사용자 모드

사용자 모드로 설정하면 level table 값은 아래의 그림과 같이 0x88880000 으로 설정합니다.

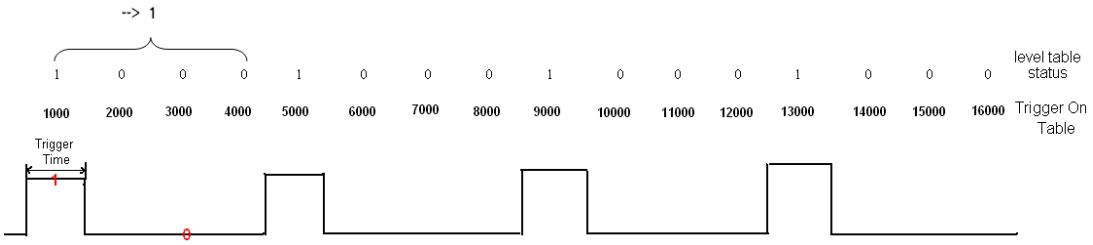


그림 27.12.2 Compare Output 사용자 모드

※사용자 모드에서 Trigger On 의 위치 테이블은 API "_DMC_01_channel1_position_compare_table_level"의 "level_table" 파라미터에서 설정합니다.

※Trigger 시간은 다음 위치까지 소요된 시간입니다 (ex. 위치 1000 에서 위치 2000 까지 소요된 시간).

27.13 _ECAT_Compare_Get_Channel1_IO_Status

■ **포맷**

U16 PASCAL _ECAT_Compare_Get_Channel1_IO_Status (U16 CardNo, U16* io_status)

■ **목적**

Compare Type1 의 I/O 상태를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 15 입니다
Io_status	U16*	수치 단위	Compare 의 I/O 상태 (Bit0 GPIO IN; Bit1 Power 상태; Bit2: level Table 사용 여부)

■ **예제**

```
U16 CardNo = 0;
U16 io_status;

U16 status = _ECAT_Compare_Get_Channel1_IO_Status (CardNo,& io_status);
```

27.14 _ECAT_Compare_Set_Channel1_GPIO_Out

■ **포맷**

U16 PASCAL _ECAT_Compare_Set_Channel1_GPIO_Out (U16 CardNo, U16 on_off)

■ **목적**

GPIO 의 출력핀 상태(Pin15)를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 15 입니다
On_off	U16	수치 단위	0 : Off 1 : On

■ **예제**

```
U16 CardNo = 0;
U16 on_off = 1;

U16 status = _ECAT_Compare_Set_Channel1_GPIO_Out (CardNo, on_off);
```

27.15 _ECAT_Compare_Set_Channel1_Position_Table

■ **포맷**

U16 PASCAL _ECAT_Compare_Set_Channel1_Position_Table (U16 CardNo, I32* pos_table, U32 table_size)

■ **목적**

Compare Type1 의 관련 데이터를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 15 입니다
Pos_table	I32*	수치 단위	데이터에 따라 Compare 해야 할 Pos_Table 을 계산합니다
Table_size	U32	수치 단위	Pos_table 이 Compare 해야 할 크기

■ **예제**

```
U16 CardNo = 0;
I32 pos_table[4] = {1000,2000,3000,4000};
U32 table_size =4;

U16 status = _ECAT_Compare_Set_Channel1_Position_Table (CardNo, pos_table, table_size);
```

비고: Table_Size 최대 설정값은 100000 입니다.

27.16 _ECAT_Compare_Set_Channel1_Position _Table_Level

■ 포맷

U16 PASCAL _ECAT_Compare_Set_Channel1_Position_Table_Level (U16 CardNo, I32* pos_table, U32* level_table, U32 table_size)

■ 목적

Compare Type1 의 Level 관련 데이터를 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 15 입니다
Pos_table	I32*	수치 단위	데이터에 따라 Compare 해야 할 Pos_Table 을 계산합니다
Level_table	U32*	수치 단위	사용자 정의 Compare 의 Level_Table 을 설정합니다
Table_size	U32	수치 단위	Pos_table 이 Compare 해야 할 크기

■ 예제

```
U16 CardNo = 0;
I32 pos_table[4] ={1000,2000,3000,4000};
U32 level_table[4] ={1,0,0,0};
U32 table_size =4;

U16 status =_ECAT_Compare_Set_Channel1_Position_Table_Level (CardNo,
pos_table, level_table,table_size);
```

비고: Table_Size 최대 설정값은 100000 입니다.

27.17 _ECAT_Compare_Get_Channel1_Position _Table_Count

27

■ **포맷**

U16 PASCAL _ECAT_Compare_Get_Channel1_Position_Table_Count (U16 CardNo, U32* cnt)

■ **목적**

이미 실행한 Trigger 의 횟수(Compare Type1 사용)를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 15 입니다
Cnt	U32*	수치 단위	이미 실행한 Trigger 의 횟수(Compare Type1 사용)를 확인합니다

■ **예제**

```
U16 CardNo = 0;
U32 cnt;

U16 status = _ECAT_Compare_Get_Channel1_Position_Table_Count (CardNo,& cnt);
```

27.18 _ECAT_Compare_Set_Channel_Polarity

■ **포맷**

U16 PASCAL _ECAT_Compare_Set_Channel_Polarity (U16 CardNo, U16 inverse)

■ **목적**

Compare Trigger 의 레벨을 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 15 입니다
Inverse	U16	수치 단위	0 : Normal 1 : Inverse

■ **예제**

```
U16 CardNo = 0;
U16 inverse = 1;

U16 status = _ECAT_Compare_Set_Channel_Polarity (CardNo, inverse);
```

27.19 _ECAT_Compare_Reuse_Channel1_Position _Table

■ **포맷**

U16 PASCAL _ECAT_Compare_Reuse_Channel1_Position_Table (U16 CardNo)

■ **목적**

지난 번에 설정한 Compare 조건으로 Compare Type1 을 다시 한 번 실행합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0 ~ 15 입니다

■ **예제**

```
U16 CardNo = 0;

U16 status = _ECAT_Compare_Reuse_Channel1_Position_Table (CardNo);
```

27.20 _ECAT_Compare_Reuse_Channel1_Position _Table_Level

■ **포맷**

U16 PASCAL _ECAT_Compare_Reuse_Channel1_Position_Table_Level (U16 CardNo)

■ **목적**

지난 번에 설정한 Compare 조건으로 Compare Type1(Level 모드)을 다시 한 번 실행합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	인터페이스 카드 번호 CardNo 는 0~15 입니다

■ **예제**

```
U16 CardNo = 0;

U16 status = _ECAT_Compare_Reuse_Channel1_Position_Table_Level (CardNo);
```

(이 페이지는 공란으로 비워둡니다)

DLL 관련 API

28

다음은 EtherCat_DLL.dll 상세 경로 데이터 및 버전 데이터 확인 등 DLL 관련 API 사용법에 대한 설명입니다.

28

DLL 관련 API 테이블

함수의 명칭	설명
_ECAT_Master_Get_DLL_Path	EtherCat_DLL.dll 의 상세 경로 데이터를 확인합니다
_ECAT_Master_Get_DLL_Version	EtherCat_DLL.dll 버전 데이터를 확인합니다
_ECAT_Master_Get_DLL_Path_Single	ECAT_RTX_DLL.dll 또는 PCI_L221.dll 의 상세 경로 데이터를 확인합니다
_ECAT_Master_Get_DLL_Version_Single	ECAT_RTX_DLL.dll 또는 PCI_L221.dll 의 버전 데이터를 확인합니다

28.1 _ECAT_Master_Get_DLL_Path

■ **포맷**

U16 PASCAL _ECAT_Master_Get_DLL_Path (I8 *lpFilePath, U32 nSize, U32 *nLength)

■ **목적**

EtherCat_DLL.dll 상세 경로 데이터를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
lpFilePath	I8*	문자 배열 단위	EtherCat_DLL.dll 의 상세 경로 데이터 문자
nSize	U32	수치 단위	경로 데이터의 검색 길이
nLength	U32*	수치 단위	반환된 경로 데이터의 실제 길이

■ **예제**

```
U16 Status = 0;
I8 FilePath[128];
U32 nSize=128, nLength=0;
Status= _ECAT_Master_Get_DLL_Path(FilePath, nSize, &nLength);
```

28.2 _ECAT_Master_Get_DLL_Version

■ 포맷

U16 PASCAL _ECAT_Master_Get_DLL_Version (I8 *lpBuf, U32 nSize, U32 *nLength)

■ 목적

EtherCat_DLL.dll 버전 데이터를 확인합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
lpBuf	I8*	문자 배열 단위	EtherCat_DLL.dll 의 상세 버전 데이터 문자
nSize	U32	수치 단위	버전 데이터의 검색 길이
nLength	U32*	수치 단위	반환된 버전의 데이터의 실제 길이

■ 예제

```
U16 Status = 0;
I8 Version[128];
U32 nSize=128, nLength=0;
Status= _ECAT_Master_Get_DLL_Version(Version, nSize, &nLength);
```

28.3 _ECAT_Master_Get_DLL_Path_Single

28

■ 포맷

U16 PASCAL _ECAT_Master_Get_DLL_Path_Single (U16 CardNo, I8 *lpFilePath, U32 nSize, U32 *nLength)

■ 목적

ECAT_RTX_DLL.dll 또는 PCI_L221.dll 의 상세 경로 데이터를 확인합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
lpFilePath	I8*	문자 배열 단위	ECAT_RTX_DLL.dll 또는 PCI_L221.dll 의 상세 경로 데이터 문자
nSize	U32	수치 단위	경로 데이터의 검색 길이
nLength	U32*	수치 단위	반환된 경로 데이터의 실제 길이

■ 예제

```
U16 Status = 0, CardNo=16;
I8 FilePath[128];
U32 nSize=128, nLength=0;

Status= _ECAT_Master_Get_DLL_Path_Single(CardNo, FilePath, nSize, &nLength);
```

28.4 _ECAT_Master_Get_DLL_Version_Single

■ **포맷**

U16 PASCAL _ECAT_Master_Get_DLL_Version_Single (U16 CardNo, I8 *lpBuf, U32 nSize, U32 *nLength)

■ **목적**

ECAT_RTX_DLL.dll 또는 PCI_L221.dll 의 버전 데이터를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
lpBuf	I8*	문자 배열 단위	ECAT_RTX_DLL.dll 또는 PCI_L221.dll 의 상세 버전 데이터 문자
nSize	U32	수치 단위	버전 데이터의 검색 길이
nLength	U32*	수치 단위	반환된 버전의 데이터의 실제 길이

■ **예제**

```
U16 Status = 0, CardNo=16;
I8 Version[128];
U32 nSize=128, nLength=0;
Status= _ECAT_Master_Get_DLL_Version_Single(CardNo, Version, nSize, &nLength);
```

(이 페이지는 공란으로 비워둡니다)

Master User Security API 29

다음은 보안키 및 사용자 암호와 관련한 Master User Security 관련 API 사용법에 대한 설명입니다.

29

Master User Security API 테이블

함수의 명칭	설명
_ECAT_Security_Check_Verifykey	보안키를 확인합니다
_ECAT_Security_Get_Check_Verifykey_State	보안키의 해제 여부를 확인합니다
_ECAT_Security_Write_Verifykey	보안키를 입력합니다
_ECAT_Security_Get_Write_Verifykey_State	보안키 입력 완료 여부를 확인합니다
_ECAT_Security_Check_UserPassword	사용자 암호를 확인합니다
_ECAT_Security_Get_Check_UserPassword_ State	사용자 암호의 해제 여부를 확인합니다
_ECAT_Security_Write_UserPassword	사용자 암호를 입력합니다
_ECAT_Security_Get_Write_UserPassword_State	사용자 암호 입력 완료 여부를 확인합니다

29.1 _ECAT_Security_Check_Verifykey

■ **포맷**

U16 PASCAL _ECAT_Security_Check_Verifykey (U16 CardNo, U32 *Verifykey)

■ **목적**

보안키를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
Verifykey	U32*	수치 배열	8 자리의 보안키를 입력합니다

■ **예제**

```
U16 Status = 0;
U16 CardNo;
U32 Key[8] = {0, 1, 2, 3, 4 ,5 ,6 ,7}

CardNo = 0;
Status= _ECAT_Security_Check_Verifykey ( CardNo, Key );
```

29.2 _ECAT_Security_Get_Check_Verifykey_State

■ **포맷**

U16 PASCAL _ECAT_Security_Get_Check_Verifykey_State (U16 CardNo, U16 *State)

■ **목적**

보안키 확인 상태의 통과 여부를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
State	U16*	수치 단위	State 반환값 설명: 0: OK 1: Fail 2: Operating 3: Check Api 를 사용해야만 정확한 조작 데이터를 얻게 됩니다 4: Security 통신 오류

■ **예제**

```
U16 Status = 0;
U16 CardNo, State;

CardNo = 0;
Status= _ECAT_Security_Get_Check_Verifykey_State ( CardNo, &State );
```

29.3 _ECAT_Security_Write_Verifykey

29

■ 포맷

U16 PASCAL _ECAT_Security_Write_Verifykey (U16 CardNo, U32 *Verifykey)

■ 목적

보안키를 입력합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
Verifykey	U32*	수치 배열	8 자리의 보안키를 입력합니다

■ 예제

```
U16 Status = 0;
U16 CardNo;
U32 Verifykey[8] = {0, 1, 2, 3, 4, 5, 6, 7};

CardNo = 0;
Status= _ECAT_Security_Write_Verifykey ( CardNo, Verifykey);
```

29.4 _ECAT_Security_Get_Write_Verifykey_State

■ **포맷**

U16 PASCAL _ECAT_Security_Get_Write_Verifykey_State (U16 CardNo, U16 *State)

■ 목적

보안키 입력 상태의 완료 여부를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
State	U16*	수치 단위	State 반환값 설명: 0: OK 1: Fail 2: Operating 3: Write API 를 사용해야만 정확한 조작 데이터를 얻게 됩니다 4: Security 통신 오류

■ **예제**

```
U16 Status = 0;
U16 CardNo, State;

CardNo = 0;
Status= _ECAT_Security_Get_Write_Verifykey_State( CardNo, &State );
```

29.5 _ECAT_Security_Check_UserPassword

29

■ **포맷**

U16 PASCAL _ECAT_Security_Check_UserPassword (U16 CardNo, U32 *UserPassword)

■ 목적

사용자 암호를 확인합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
UserPassword	U32*	수치 배열	8 자리의 사용자 비밀번호를 입력합니다

■ 예제

```
U16 Status = 0;
U16 CardNo;
U32 UserPassword[8] = {0, 1, 2, 3, 4, 5, 6, 7};

CardNo = 0;
Status= _ECAT_Security_Check_UserPassword ( CardNo, UserPassword);
```

29.6 _ECAT_Security_Get_Check_UserPassword _State

■ **포맷**

U16 PASCAL _ECAT_Security_Get_Check_UserPassword_State (U16 CardNo, U16 *State)

■ 목적

사용자 암호 확인 상태의 통과 여부를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
State	U16*	수치 단위	State 반환값 설명: 0: OK 1: Fail 2: Operating 3: Check API 를 사용해야만 정확한 조작 데이터를 얻게 됩니다. 4: Security 통신 오류입니다.

■ **예제**

```
U16 Status = 0;
U16 CardNo, State;

CardNo = 0;
Status= _ECAT_Security_Get_Check_UserPassword_State ( CardNo, &State );
```

29.7 _ECAT_Security_Write_UserPassword

29

■ **포맷**

U16 PASCAL _ECAT_Security_Write_UserPassword (U16 CardNo, U32 *UserPassword)

■ **목적**

사용자 암호를 입력합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
UserPassword	U32*	수치 배열	8 자리의 사용자 비밀번호를 입력합니다

■ 예제

```
U16 Status = 0;
U16 CardNo;
U32 UserPassword [8] = {0, 1, 2, 3, 4, 5, 6, 7};

CardNo = 0;
Status= _ECAT_Security_Write_UserPassword( CardNo, UserPassword);
```

29.8 _ECAT_Security_Get_Write_UserPassword_State

■ **포맷**

U16 PASCAL _ECAT_Security_Get_Write_UserPassword_State (U16 CardNo, U16 *State)

■ 목적

사용자 암호 입력 상태의 통과 여부를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
State	U16*	수치 단위	State 반환값 설명: 0: OK 1: Fail 2: Operating 3: Write API 를 사용해야만 정확한 조작 데이터를 얻게 됩니다 4: Security 통신 오류

■ **예제**

```
U16 Status = 0;
U16 CardNo, State;

CardNo = 0;
Status= _ECAT_Security_Get_Write_UserPassword_State ( CardNo, &State );
```

(이 페이지는 공란으로 비워둡니다)

PAC MRAM 전용 API 30

다음은 PAC MRAM 확인 입력 등 PAC MRAM 관련 API 사용법에 대한 설명입니다.

PAC MRAM 전용 API 테이블

함수의 명칭	설명
_ECAT_Master_MRAM_Write_Word_Data	U16 데이터를 PAC MRAM의 지정된 위치에 입력합니다
_ECAT_Master_MRAM_Read_Word_Data	PAC MRAM의 지정된 위치에 입력한 U16 데이터를 확인합니다
_ECAT_Master_MRAM_Write_DWord_Data	U32 데이터를 PAC MRAM의 지정된 위치에 입력합니다
_ECAT_Master_MRAM_Read_DWord_Data	PAC MRAM의 지정된 위치에 입력한 U32 데이터를 확인합니다

30.1 _ECAT_Master_MRAM_Write_Word_Data

■ **포맷**

U16 PASCAL _ECAT_Master_MRAM_Write_Word_Data (U16 CardNo, U32 Index, U32 DataNum, U16 *Data)

■ **목적**

U16 데이터를 PAC MRAM의 지정된 위치에 입력합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
Index	U32	번호 단위	0 ~ 32767 (Word는 2byte를 차지하므로 32766에만 도달하면 됩니다)
DataNum	U32	수치 단위	데이터 수
Data	U16*	수치 단위	데이터

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, data=1;
U32 Index=32766, DataNum=1;

Status= _ECAT_Master_MRAM_Write_Word_Data(CardNo, Index, DataNum, &data);
```

30.2 _ECAT_Master_MRAM_Read_Word_Data

■ **포맷**

U16 PASCAL _ECAT_Master_MRAM_Read_Word_Data (U16 CardNo, U32 Index, U32 DataNum, U16 *Data)

■ **목적**

PAC MRAM 의 지정된 위치에 입력한 U16 데이터를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
Index	U32	번호 단위	0 ~ 32767 (Word 는 2byte 를 차지하므로 32766 에만 도달하면 됩니다)
DataNum	U32	수치 단위	데이터 수
Data	U16*	수치 단위	데이터

■ **예제**

```
U16 Status = 0;
U16 CardNo=16, data=1;
U32 Index=32766, DataNum=1;

Status= _ECAT_Master_MRAM_Read_Word_Data(CardNo, Index, DataNum, &data);
```

30.3 _ECAT_Master_MRAM_Write_DWord_Data

30

■ 포맷

U16 PASCAL _ECAT_Master_MRAM_Write_DWord_Data (U16 CardNo, U32 Index, U32 DataNum, U32 *Data)

■ 목적

U32 데이터를 PAC MRAM 의 지정된 위치에 입력합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
Index	U32	번호 단위	0 ~ 32767 (DWORD 는 4 byte 를 차지하므로 32764 에만 도달하면 됩니다)
DataNum	U32	수치 단위	데이터 수
Data	U16*	수치 단위	데이터

■ 예제

```
U16 Status = 0;
U16 CardNo=16;
U32 data=2;
U32 Index=32764, DataNum=1;

Status= _ECAT_Master_MRAM_Write_DWord_Data(CardNo, Index, DataNum, &data);
```

30.4 _ECAT_Master_MRAM_Read_DWord_Data

■ **포맷**

U16 PASCAL _ECAT_Master_MRAM_Read_DWord_Data (U16 CardNo, U32 Index, U32 DataNum, U32 *Data)

■ **목적**

PAC MRAM 의 지정된 위치에 입력한 U32 데이터를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
Index	U32	번호 단위	0 ~ 32767 (DWORD 는 4 byte 를 차지하므로 32764 에만 도달하면 됩니다)
DataNum	U32	수치 단위	데이터 수
Data	U16*	수치 단위	데이터

■ **예제**

```
U16 Status = 0;
U16 CardNo=16;
U32 data=2;
U32 Index=32764, DataNum=1;

Status= _ECAT_Master_MRAM_Read_DWord_Data(CardNo, Index, DataNum, &data);
```

(이 페이지는 공란으로 비워둡니다)

70E2 Slave API

31

다음은 70E2 Output 의 활성화 여부 설정 등 70E2 Slave 관련 API 사용법에 대한 설명입니다.

70E2 Slave API 테이블

함수의 명칭	설명
_ECAT_Slave_R1_EC70E2_Set_Output _Enable	Slave Delta Module 70E2 전용 명령, Output 의 활성화 여부를 설정합니다.

31.1 _ECAT_Slave_R1_EC70E2_Set_Output_Enable

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC70E2_Set_Output_Enable (U16 CardNo, U16 NodeID, U16 SlotNo, U16 Enable)

■ **목적**

70E2 slave 모듈 전용 명령, Output 의 활성화 여부를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Enable	U16	수치 단위	0: 모든 출력 지점의 위치를 업데이트하지 않습니다 1: 모든 출력 지점의 위치를 업데이트합니다

■ **예제**

```
U16 Status = 0;
U16 CardNo = 16, NodeID = 0, SlotNo = 0;
U16 Enable = 1; // 70E2 의 모든 지점에서 업데이트를 받도록 설정

Status = _ECAT_Slave_R1_EC70E2_Set_Output_Enable ( CardNo, NodeID, SlotNo, Enable );
```

70X2 Slave API

32

다음은 70X2 Output 의 활성화 여부 설정 등 70X2 Slave 관련 API 사용법에 대한 설명입니다.

70X2 Slave API 테이블

함수의 명칭	설명
_ECAT_Slave_R1_EC70X2_Set_Output _Enable	70X2 slave 모듈 전용 명령, Output 의 활성화 여부를 설정합니다.

32.1 _ECAT_Slave_R1_EC70X2_Set_Output_Enable

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC70X2_Set_Output_Enable (U16 CardNo, U16 NodeID, U16 SlotNo, U16 Enable)

■ **목적**

70X2 slave 모듈 전용 명령, Output 의 활성화 여부를 설정합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Enable	U16	수치 단위	0: 모든 출력 지점의 위치를 업데이트하지 않습니다 1: 모든 출력 지점의 위치를 업데이트합니다

■ **예제**

```
U16 Status = 0;
U16 CardNo = 16, NodeID = 0, SlotNo = 0;
U16 Enable = 1; // 70X2 의 모든 지점에서 업데이트를 받도록 설정

Status = _ECAT_Slave_R1_EC70X2_Set_Output_Enable ( CardNo, NodeID, SlotNo, Enable );
```

5614 Slave API

33

다음은 MPG 또는 JOG 관련 설정, 모듈 IO 상태 확인 등 5614 Slave 관련 API 사용법에 대한 설명입니다.

33

5614 Slave API 테이블

함수의 명칭	설명
_ECAT_Slave_R1_EC5614_Set_MJ_Config	5614 Slave 전용 명령, MPG 또는 JOG 의 파라미터 내용을 설정합니다.
_ECAT_Slave_R1_EC5614_Set_MJ_Enable	5614 Slave 전용 명령, MPG 또는 JOG 의 활성화 여부를 설정합니다.
_ECAT_Slave_R1_EC5614_Get_IO_Status	5614 Slave 전용 명령, 모듈의 IO 상태를 확인합니다.
_ECAT_Slave_R1_EC5614_Get_MPG_Counter	5614 Slave 전용 명령, 모듈의 MPG Counter 를 확인합니다.

33.1 _ECAT_Slave_R1_EC5614_Set_MJ_Config

■ 포맷

U16 PASCAL _ECAT_Slave_R1_EC5614_Set_MJ_Config (U16 CardNo, U16 MJNo, U16 MJType, U16 NodeID, U16 SlotNo, U16 AxisNum, U16 *AxisArray, U16 *SlotArray, I32 *MaxSpeedArray, F64 *TaccArray, F64 *RatioArray)

■ 목적

Slave Delta Module 5614 전용 명령, MPG 또는 JOG 의 파라미터 내용을 설정합니다.

■ 파라미터

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
MJNo	U16	번호 단위	MPG 또는 JOG 는 최대 8 세트를 지원하며 번호는 0 ~ 7 입니다
MJType	U16	번호 단위	MPG 모드 설정 0: Jog 모드 사용 1: x1 의 핸드휠 사용 2: x4 의 핸드휠 사용
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
AxisNum	U16	수치 단위	MPG / JOG 가 사용해야 할 축 수 (MPG 는 최대 6 축 사용, JOG 는 최대 2 축 사용)
AxisArray	U16*	번호 배열	축을 사용하는 Node 번호 배열
SlotArray	U16*	번호 배열	축을 사용하는 Slot 배열
MaxSpeedArray	I32*	수치 배열	축을 사용하는 속도 제한치 배열 (단위: pps)
TaccArray	F64*	수치 배열	축을 사용하는 가속 시간 제한치 시퀀스 (단위: sec) (속도값은 MaxSpeedArray 를 기준으로 함)
RatioArray	F64*	수치 배열	MPG 전용, 핸드휠의 펄스 출력 비율

33

■ **예제**

```
U16 Status = 0, CardNo = 16, NodeID = 0, SlotNo = 0, AxisNum = 2, MJNo = 2,
MJType= 1;
U16 AxisArray[2] = {1, 2};
U16 SlotArray[2] = {0, 0};
I32 MaxSpeedArray[2] = {100000, 200000};
F64 TaccArray[2] = {0.1, 0.1};
F64 RatioArray[2] = {1, 1};
Status = _ECAT_Slave_R1_EC5614_Set_MJ_Config( CardNo, NodeID, SlotNo, MJNo,
MJType, AxisNum, AxisArray, SlotArray, MaxSpeedArray, TaccArray, RatioArray );
```

33.2 _ECAT_Slave_R1_EC5614_Set_MJ_Enable

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC5614_Set_MJ_Enable (U16 CardNo, U16 MJNo, U16 Enable)

■ **목적**

Slave Delta Module 5614 전용 명령, MPG 또는 JOG 의 활성화 여부를 설정합니다.
※Enable 하기 전에 우선 _ECAT_Slave_R1_EC5614_Set_MJ_Config 를 사용하여 관련 파라미터를 설정해야 합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
MJNo	U16	번호 단위	MPG 또는 JOG 는 최대 8 세트를 지원하며 번호는 0 ~ 7 입니다
Enable	U16	번호 단위	0: MPG 또는 JOG 사용 중지 1: MPG 또는 JOG 사용

■ 예제

```
U16 Status = 0, CardNo = 16, NodeID = 0, SlotNo = 0, AxisNum = 2, MJNo = 1,
MJType= 1;
U16 Enable = 1;
U16 AxisArray[2] = {1, 2};
U16 SlotArray[2] = {0, 0};
I32 MaxSpeedArray[2] = {100000, 200000};
F64 TaccArray[2] = {0.1, 0.1};
F64 RatioArray[2] = {1, 1};

// 우선 관련 파라미터를 설정해야 함
Status = _ECAT_Slave_R1_EC5614_Set_MJ_Config( CardNo, NodeID, SlotNo, MJNo,
MJType, AxisNum, AxisArray, SlotArray, MaxSpeedArray, TaccArray, RatioArray );

// 설정에 성공해야만 사용 가능
If(Status == 0)
    Status = _ECAT_Slave_R1_EC5614_Set_MJ_Enable(CardNo, MJNo, Enable);
```

33.3 _ECAT_Slave_R1_EC5614_Get_IO_Status

33

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC5614_Get_IO_Status (U16 CardNo, U16 NodeID, U16 SlotNo, U16 *IOStatus)

■ **목적**

Slave Delta Module 5614 전용 명령, 모듈의 IO 상태를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
IOStatus	U16*	수치 단위	모듈 IO 상태

■ **상태 설명**

Bit	Meaning	Bit	Meaning
0	JY-	8	X-Axis Select Status
1	JY+	9	Y-Axis Select Status
2	JX-	10	Z-Axis Select Status
3	JX+	11	U-Axis Select Status
4	×1 Select Status	12	W-Axis Select Status
5	×10 Select Status	13	C-Axis Select Status
6	×100 Select Status	14	Reserved
7	Reserved	15	MPG Enable

■ **예제**

```
U16 Status = 0, CardNo = 16, NodeID = 0, SlotNo = 0;
U16 IOStatus = 0;

Status = _ECAT_Slave_R1_EC5614_Get_IO_Status (CardNo, NodeID, SlotNo,
&IOStatus);
```

33.4 _ECAT_Slave_R1_EC5614_Get_MPG_Counter

■ **포맷**

U16 PASCAL _ECAT_Slave_R1_EC5614_Get_MPG_Counter (U16 CardNo, U16 NodeID, U16 SlotNo, I32 *Counter)

■ **목적**

Slave Delta Module 5614 전용 명령, 모듈의 MPG Counter 를 확인합니다.

■ **파라미터**

명칭	데이터 타입	단위	설명
CardNo	U16	번호 단위	카드 번호
NodeID	U16	번호 단위	NodeID 번호
SlotNo	U16	번호 단위	SlotID 번호
Counter	I32*	수치 단위	모듈의 MPG Counter

■ **예제**

```
U16 Status = 0, CardNo = 16, NodeID = 0, SlotNo = 0;
I32 Counter = 0;

Status = _ECAT_Slave_R1_EC5614_Get_MPG_Counter(CardNo, NodeID, SlotNo,
&Counter);
```

(이 페이지는 공란으로 비워둡니다)

33

34 명령 반환값과 메시지 설명

다음은 명령 반환값과 메시지 설명에 대한 내용입니다.

34

34.1 반환값 오류 요약

EtherCatDll 의 API 라이브러리를 사용할 경우, API 함수는 대부분 1 개의 계수값을 반환합니다 (하단 표 참조). 해당 API 함수가 반환한 값이 0 일 경우, 해당 API 함수가 성공적으로 실행되었음을 의미합니다. 반면, 해당 API 함수가 다른 오류 코드를 반환할 경우, 운영 프로그램 또는 하드웨어 연결에 문제가 있음을 의미합니다. 해당 오류 코드 관련 설명에 따라 문제를 해결하십시오.

오류 반환값 리스트

반환 코드 오류 (십진법)	오류 코드 표시명	오류 문제 설명
0	ERR_ECAT_NO_ERROR	오류가 없습니다
1	ERR_ECAT_HW_NO_INITIALIZE	하드웨어가 아직 초기화되지 않음
2	ERR_ECAT_HW_PWM_INITIAL	하드웨어 초기화 실패 (AM335x, RT8139)
3	ERR_ECAT_HW_HAS_INITIALIZED	하드웨어 초기화를 반복합니다
16	ERR_ECAT_EEPROM_READ	모듈 EEPROM 확인 실패
17	ERR_ECAT_EEPROM_WRITE	모듈 EEPROM 입력 실패
18	ERR_ECAT_ENVIRONMENT_ RECORD_DISABLE	파일 기능 무효화를 설정합니다
19	ERR_ECAT_ENVIRONMENT_ RECORD_NO_MATCH	설정한 파일 환경과 온라인 환경이 맞지 않습니다
20	ERR_ECAT_ENVIRONMENT_ RECORD_FILE_OPEN	파일 실행 설정 실패
21	ERR_ECAT_ENVIRONMENT_ RECORD_NOT_CREATE	설정한 파일 구조가 생성되지 않았습니다
22	ERR_ECAT_XML_FILE_PATH	ENI 파일 경로 오류 (WIN32 버전에서만 사용)
23	ERR_ECAT_DEVICE_OPEN	Master 초기화 설정 오류
24	ERR_ECAT_NO_DEVICE	Master 초기 입력 모듈 정보 오류
25	ERR_ECAT_NO_MASTER	Master 가 생성되지 않았습니다
26	ERR_ECAT_NO_SLAVE	Slave 가 존재하지 않습니다
27	ERR_ECAT_UNKNOWN_SLAVE	알 수 없는 모듈 유형이 존재합니다
28	ERR_ECAT_IST_CREATE	IST 생성 실패
29	ERR_ECAT_MASTER_CREATE	Master 생성 실패 (WIN32 버전에서만 사용)
30	ERR_ECAT_MASTER_REQUEST_ STATE	Master 전환 상태값 오류

반환 코드 오류 (십진법)	오류 코드 표시명	오류 문제 설명
31	ERR_ECAT_MASTER_OPERATION_NOT_READY	Master 가 OP 상태로 전환되지 않았습니다
32	ERR_ECAT_DELTA_NODE_ID_ALIAS_READ	델타 드라이브 닉네임 확인 실패
33	ERR_ECAT_MASTER_GET_SERIAL_NO_WRONG	PAC/축 카드 시퀀스 번호 확인 실패
34	ERR_ECAT_MASTER_GET_SERIAL_NO_TIMEOUT	PAC/축 카드 시퀀스 번호 확인 시간 초과
128	ERR_ECAT_PIPELINE_CORE_TIMER_CREATE	스케줄러 키 이벤트 생성 실패
129	ERR_ECAT_PIPELINE_CREATE	스케줄러 기능 생성 실패
130	ERR_ECAT_COMMAND_ENQUEUE	버퍼 영역에 스케줄러 명령 새로 추가하기 실패
131	ERR_ECAT_API_BUFFER_ENQUEUE	버퍼 영역에 스케줄러 명령 새로 추가하기 실패
256	ERR_ECAT_NODE_ID	모듈 Node 번호 오류
257	ERR_ECAT_SLOT_ID	모듈 슬롯 번호 오류
258	ERR_ECAT_SDO_DOWNLOAD	SDO 데이터 설정 오류
259	ERR_ECAT_SDO_UPLOAD	SDO 데이터 확인 오류
260	ERR_ECAT_GET_PROCESS_DATA	지정한 PDO 데이터를 확인할 수 없습니다
512	ERR_ECAT_DIO_CHANNEL_INVALID	DIO Channel 설정 오류 (WIN32 버전에서만 사용)
513	ERR_ECAT_ADDA_CHANNEL_INVALID	AIO Channel 설정 오류 (WIN32 버전에서만 사용)
514	ERR_ECAT_MOTION_NOT_FINISHED	위치 설정 오류 (WIN32 버전에서만 사용)
515	ERR_ECAT_SET_PULSE_MODE	모듈 입출력 모드 오류 (WIN32 버전에서만 사용)
516	ERR_ECAT_SET_SOFTLIMIT	소프트웨어 limit 설정 오류 (WIN32 버전에서만 사용)
517	ERR_ECAT_SET_POSITION	설정 위치 오류 (WIN32 버전에서만 사용)
518	ERR_ECAT_GET_SPEED	PDO 속도 데이터 확인 불가 (WIN32 버전에서만 사용)
519	ERR_ECAT_GET_MCDONE	PDO StatusWord 데이터 확인 불가 (WIN32 버전에서만 사용)

반환 코드 오류 (십진법)	오류 코드 표시명	오류 문제 설명
520	ERR_ECAT_SET_HOME_CONFIG	Homing 파라미터 설정 오류 (WIN32 버전에서만 사용)
521	ERR_ECAT_SET_P2P_CONFIG	PP 동작 파라미터 설정 오류 (WIN32 버전에서만 사용)
522	ERR_ECAT_SET_PT_CONFIG	PT 동작 파라미터 설정 오류 (WIN32 버전에서만 사용)
523	ERR_ECAT_SET_PV_CONFIG	PV 동작 파라미터 설정 오류 (WIN32 버전에서만 사용)
524	ERR_ECAT_SET_CSP_CONFIG	CSP 단축 직선 동작 파라미터 설정 오류 (WIN32 버전에서만 사용)
525	ERR_ECAT_SET_MULTI_AXES_ LINE_CONFIG	CSP 다축 직선 동작 파라미터 설정 오류 (WIN32 버전에서만 사용)
526	ERR_ECAT_SET_MULTI_AXES_ ARC_CONFIG	CSP 양축 원형 동작 파라미터 설정 오류 (WIN32 버전에서만 사용)
768	ERR_ECAT_MD1_SET_GEAR	설정 모드 1 의 기어비 파라미터 오류 (WIN32 버전에서만 사용)
769	ERR_ECAT_MD1_SET_P_CHANGE	설정 모드 1 의 위치 변화 파라미터 오류 (WIN32 버전에서만 사용)
770	ERR_ECAT_MD1_SET_V_CHANGE	설정 모드 1 의 속도 변화 파라미터 오류 (WIN32 버전에서만 사용)
771	ERR_ECAT_MD1_SET_SOFTLIMIT	설정 모드 1 의 소프트웨어 limit 의 파라미터 오류 (WIN32 버전에서만 사용)
772	ERR_ECAT_MD1_SET_SLD	설정 모드 1 의 감속 정지 파라미터 오류 (WIN32 버전에서만 사용)
773	ERR_ECAT_MD1_SET_HOME_ CONFIG	설정 모드 1 의 Homing 파라미터 오류 (WIN32 버전에서만 사용)
774	ERR_ECAT_MD1_SET_P2P_ CONFIG	설정 모드 1 의 단축 동작 파라미터 오류 (WIN32 버전에서만 사용)
775	ERR_ECAT_MD1_SET_V_MOVE_ CONFIG	설정 모드 1 의 단축 연속 동작 파라미터 오류 (WIN32 버전에서만 사용)
776	ERR_ECAT_MD1_SET_LINE_ CONFIG	설정 모드 1 의 다축 동작 파라미터 오류 (WIN32 버전에서만 사용)

반환 코드 오류 (십진법)	오류 코드 표시명	오류 문제 설명
777	ERR_ECAT_MD1_SET_ARC_ CONFIG	설정 모드 1 의 원형 동작 파라미터 오류 (WIN32 버전에서만 사용)
1024	ERR_ECAT_PATH_NOT_SUPPORT	CSP 동작 모드 오류 (WIN32 버전에서만 사용)
1025	ERR_ECAT_PATH_AXIS_NUM	CSP 동작 축수 오류 (WIN32 버전에서만 사용)
1026	ERR_ECAT_PATH_AXIS_NO	CSP 동작 축 번호 오류 (WIN32 버전에서만 사용)
1027	ERR_ECAT_PATH_PARA	CSP 동작 파라미터 오류 (WIN32 버전에서만 사용)
1028	ERR_ECAT_PATH_ISR_FUNC_ EVENT	ISR 함수 지표 이벤트 연결 오류 (WIN32 버전에서만 사용)
1029	ERR_ECAT_PATH_AXISNO_ UNDER_GROUP	CSP 그룹 축 번호 오류 (WIN32 버전에서만 사용)
1152	ERR_ECAT_PATH_ROBOT_NOT_ SUPPORT	SRC 동작 모드 오류 (WIN32 버전에서만 사용)
1153	ERR_ECAT_PATH_ROBOT_STOP	SRC 정지 파라미터 오류 (WIN32 버전에서만 사용)
1154	ERR_ECAT_PATH_ROBOT_STOP	SRC 사용 축 수가 실제 축 수를 초과함 (WIN32 버전에서만 사용)
1155	ERR_ECAT_PATH_ROBOT_ BUFFER_FULL	SRC 모션 명령 Buffer 가 가득 참 (WIN32 버전에서만 사용)
3840	ERR_ESI_INITIAL	ESI Parser 가 초기화를 진행하지 않았거나 초기화에 실패했습니다
3841	ERR_ESI_OPEN_DEVICE	적절한 하드웨어 설정을 찾을 수 없습니다
3842	ERR_ESI_CREATE_CANOPEN_OD _LIST	지원하는 CANOpen OD 리스트를 생성할 수 없습니다
3843	ERR_ESI_NO_DATA_TYPE_INFO	DataType 섹터 정보를 생성할 수 없습니다
3844	ERR_ESI_NO_OBJECT_INFO	Object 섹터 정보를 생성할 수 없습니다
3845	ERR_ESI_CREATE_SYNC_ MANAGER	SM 섹터 또는 RxPDO / TxPDO 섹터 정보를 생성할 수 없습니다
3846	ERR_ESI_CREATE_FMMU_ CONTROL	Fmmu 섹터 정보를 생성할 수 없습니다
3847	ERR_ESI_NO_PDO_CHANNEL	지정한 PDO Channel No 가 존재하지 않습니다

반환 코드 오류 (십진법)	오류 코드 표시명	오류 문제 설명
3848	ERR_ESI_NO_PDO_MAPPING	지정한 PDO Channel No 에 OD 가 전혀 없습니다
3849	ERR_ESI_PDO_MAPPING_INSERT	CANOpen OD 를 새로 추가할 수 없습니다
3850	ERR_ESI_PDO_MAPPING_DELETE	CANOpen OD 를 삭제할 수 없습니다
3851	ERR_ESI_CREATE_DISTRIBUTED_CLOCK	Dc Setting 정보를 생성할 수 없습니다
4080	ERR_ESI_ENI_INFORMATION_INITIAL	ProcessData 또는 InitialCommand 등 이후 정보를 생성할 수 없습니다
4081	ERR_ESI_ENI_FILE_INITIAL	ESI FILE 에 필요한 정보를 생성할 수 없습니다
4082	ERR_ESI_ENI_FILE_SAVE	ESI FILE 파일을 저장할 수 없습니다
4096	ERR_ECAT_NO_SLAVE_FOUND	Slave 에 접속하거나 Slave 를 발견하지 못했습니다
4097	ERR_ECAT_INITIAL_TIMEOUT	Initial 대기 시간이 다소 깁니다
4098	ERR_ECAT_MODE_CHANGE_FAILED	OP 모드 또는 Init 모드로 전환할 수 없습니다
4099	ERR_ECAT_SLAVE_ID	전송한 Node 번호에 오류가 있습니다
4100	ERR_ECAT_ALIAS_SLAVE_ID	Node 번호 닉네임이 중복되거나 최대 제한 축수를 초과했습니다
4352	ERR_ECAT_NEED_INITIAL	해당 조작을 실행하기 전에 먼저 Initial 을 실행해야 합니다
4353	ERR_ECAT_NEED_RESET	해당 API 는 Initial 상태에서 실행할 수 없습니다
4354	ERR_ECAT_NEED_CONNECT	해당 조작을 실행하기 전에 먼저 링크를 생성해야 합니다
4355	ERR_ECAT_NEED_DC_OP	DC 교정 완료 후 OP 모드로 전환해야만 실행 가능합니다
4356	ERR_ECAT_NEED_RALM	Alm 이 있을 경우, ResetAlm 을 해야만 실행 가능함
4357	ERR_ECAT_NEED_SVON	먼저 Svon 해야만 실행 가능합니다
4358	ERR_ECAT_NEED_HOMECONFIG	HomeMove 를 실행하기 전에 먼저 HomeConfig 를 실행해야 합니다
4359	ERR_ECAT_NEED_STOP	해당 API 는 Motion 동작을 실행해야만 실행 가능합니다
4608	ERR_ECAT_RING_BUFFER_FULL	API Ring Buffer(MailBox)가 가득 찼습니다

반환 코드 오류 (십진법)	오류 코드 표시명	오류 문제 설명
4609	ERR_ECAT_API_PARAMETER	API 파라미터 입력 오류
4610	ERR_ECAT_SLAVE_TYPE	해당 모듈은 해당 API 를 지원하지 않습니다
4611	ERR_ECAT_TARGET_REACHED	Target 이 대기 시간을 초과하여 On 이 실행되지 않음
4612	ERR_ECAT_MODE_NOT_ SUPPORT	해당 이동 모드에서는 해당 API 를 지원하지 않습니다
4613	ERR_ECAT_MOTION_TYPE	해당 축은 해당 동작 모드를 지원하지 않습니다 (Group 의 제한을 받기 때문으로 추정됩니다)
4614	ERR_ECAT_PDO_NOT_MAPPING	해당 동작 모드의 OD 필수 항목이 PDO 에 매핑되지 않았습니다
4615	ERR_ECAT_MODULE_REVISION	해당 모듈의 Revision 버전을 지원하지 않습니다
4616	ERR_ECAT_SPEED_CONTINUE_ MODE	SpeedContinue 에 모드를 사용하지 않은 축이 있습니다
4617	ERR_ECAT_HOME_MODE	Homing 파라미터의 Mode 설정 오류
4618	ERR_ECAT_HOME_OFFSET	Homing 파라미터의 Offset 설정 오류
4619	ERR_ECAT_HOME_FIRST_SPEED	Homing 파라미터의 FirstVel 설정 오류
4620	ERR_ECAT_HOME_SECOND_ SPEED	Homing 파라미터의 SecondVel 설정 오류
4621	ERR_ECAT_HOME_ACC	Homing 파라미터의 Acc 설정 오류
4622	ERR_ECAT_MRAM_INDEX	정전 시 메모리 유지 Offset 설정 오류
4623	ERR_ECAT_MRAM_INDEX_OUT_ RANGE	정전 시 메모리 유지 Range 설정 오류
4864	ERR_ECAT_PDO_TX_FAILED	PDO 유형의 명령 전송 실패
4865	ERR_ECAT_SDO_TIMEOUT	SDO 의 응답 대기시간이 다소 깁니다
4866	ERR_ECAT_SDO_RETURN	Slave 가 SDO 오류 코드를 반환하면 _ECAT_Slave_CANopen_Get_ErrorCode 를 사용하여 코드를 확인할 수 있습니다
4867	ERR_ECAT_PDO_RX_FAILED	PDO 유형의 명령 반환 실패
4868	ERR_ECAT_MAILBOX	MailBox 전송 실패
4869	ERR_ECAT_SDO_BUFFER_FULL	SDO 의 명령 비상주 영역이 가득 찼습니다
5120	ERR_ECAT_GROUP_NUMBER	입력한 GroupNo 가 최대 모듈 수 제한치(10)를 초과했습니다.

반환 코드 오류 (십진법)	오류 코드 표시명	오류 문제 설명
5121	ERR_ECAT_GROUP_ENABLE	해당 Group 이 Enable 또는 Pause 되지 않았습니다.
5122	ERR_ECAT_GROUP_PAUSE	해당 동작은 Group Pause 일 경우에만 사용 가능합니다.
5123	ERR_ECAT_GROUP_SLAVE	해당 Slave 가 다른 Group 에 의해 활성화되었거나 해당 Group 내에서 중복 존재합니다.
5124	ERR_ECAT_GROUP_MODE	해당 Group Mode 는 일시적으로 지원하지 않습니다.
5125	ERR_ECAT_GROUP_ALREADY_USED	해당 GroupNo 기능이 이미 작용 중입니다.
5126	ERR_ECAT_GROUP_TYPE	Group 기능은 미설정 모드일 때 활성화할 수 없고, 활성화 이후에는 모드를 설정할 수 없습니다.
5127	ERR_ECAT_GROUP_SVON	Group 내부의 각 축을 Svon 상태로 만드십시오.
5128	ERR_ECAT_GROUP_ALM	Group 내부의 각 축에 Alm 상태가 없는지 확인하십시오.
5129	ERR_ECAT_GROUP_DATA_BUFFER	Data Buffer 가 가득 찼습니다 (799 비트).
5130	ERR_ECAT_GROUP_TIMEOUT	Group 조작 시, 하위 계층의 응답이 없습니다.
5376	ERR_ECAT_SERVO_PARA_EMPTY	해당 드라이브 파라미터가 존재하지 않습니다.
5377	ERR_ECAT_SERVO_PARA_RO	해당 드라이브 파라미터는 읽기만 가능합니다.
5378	ERR_ECAT_SERVO_COMPARE_ENABLE	드라이브 Compare 기능 설정에 실패했습니다.
5632	ERR_ECAT_RECORD_TYPE	Record 기능이 활성화되었을 때, 트리거할 데이터를 수정할 수 없습니다.
5888	ERR_ECAT_MPG_ENABLE	해당 MJ 그룹은 이미 활성화되었습니다. 활성화 해제 후 다시 실행하시기 바랍니다.
5889	ERR_ECAT_MPG_CONFIG	해당 MJ 그룹의 기본 데이터를 설정하지 않았습니다. 우선 Config 를 실행하시기 바랍니다.

반환 코드 오류 (십진법)	오류 코드 표시명	오류 문제 설명
8192	ERR_ECAT_ROBOT_TYPE	해당 API 가 이 Robot Type 을 지원하지 않거나 입력한 Robot Type 이 잘못된 번호입니다.
8193	ERR_ECAT_ROBOT_INITIAL	해당 API 는 우선 Initial Robot System 을 실행해야 합니다.
8194	ERR_ECAT_ROBOT_UPDATE	Robot 과의 내부 코어 업데이트 메시지가 나타날 때 오류가 발생했습니다.
8195	ERR_ECAT_ROBOT_FAILED	해당 API 는 코어와 연결 시 오류가 발생했습니다
8196	ERR_ECAT_ROBOT_RESET	해당 API 는 Motion_State Disable 해야만 실행 가능합니다.
8197	ERR_ECAT_ROBOT_PARAMETER	해당 API 에 명령한 파라미터의 오류입니다.
8198	ERR_ECAT_ROBOT_IN_MOTION	로봇이 동작 중입니다.
8199	ERR_ECAT_ROBOT_BUFFER_ FULL	명령 스케줄링이 가득 찼습니다.
8200	ERR_ECAT_ROBOT_NEED_ SERVO_OFF	현재 Servo ON 상태입니다.
8201	ERR_ECAT_ROBOT_V_CHANGING	변속 파라미터 설정값이 너무 높습니다.
12288	ERR_ECAT_SECURITY_ OPERATING	Security 관련 조작이 진행 중입니다. 잠시 후에 다시 실행하시기 바랍니다.
12289	ERR_ECAT_SECURITY_NEED_ LOGIN	해당 동작은 로그인한 후에만 조작 가능합니다.
12290	ERR_ECAT_SECURITY_CONNECT	Security 코어와의 통신에 실패했습니다.
16384	ERR_ECAT_SPLC_CONNECT_ FAILED	SoftPLC 실행 시 연결에 실패했습니다.
16385	ERR_ECAT_SPLC_RTSS_FAILED	SoftPLC Kernel 내부 초기화에 실패했습니다.
16386	ERR_ECAT_SPLC_KERENL_FILE	SoftPLC Kernel 실행 시 실패했습니다.
16387	ERR_ECAT_SPLC_ONGOING	SoftPLC 을 사용 중이므로 일시적으로 해당 API 를 조작할 수 없습니다.
16388	ERR_ECAT_SPLC_SDO_FAILED	SoftPLC SDO 명령 전송에 실패했습니다.
16389	ERR_ECAT_SPLC_ALREADY_RUN	SoftPLC 이 실행 중이므로 일시적으로 해당 API 를 조작할 수 없습니다.

반환 코드 오류 (십진법)	오류 코드 표시명	오류 문제 설명
16390	ERR_ECAT_SPLC_MOTION_ CONFIG_RUN	해당 함수가 실행 중입니다. 종료 확인 후 다시 조작하시기 바랍니다.
16391	ERR_ECAT_SPLC_MOT_MODE	해당 API 는 해당 상태 머신에서 실행할 수 없습니다
16392	ERR_ECAT_SPLC_HOMING_ ERROR	Home 동작이 실패했습니다.
16393	ERR_ECAT_SPLC_MOTION_TYPE _ERROR	동작 모드 오류
16394	ERR_ECAT_SPLC_MOTION_ ERROR	SD_StopFB 에 의해 동작이 중지되었습니다
16395	ERR_ECAT_SPLC_MOTION_Jog_ ERROR	Jog FB 동작 Forward 와 Backward 동시 트리거
16896	ERR_ECAT_SPLC_AXISREF_ STRUCT	FB 입력 단자에서 잘못된 구조를 사용했습니다.
16897	ERR_ECAT_SPLC_MASTER_ INDEX_NOT_FOUND	FB 입력 단자 카드 번호가 실제 카드 번호와 대응하지 않습니다.
16898	ERR_ECAT_SPLC_AXIS_INDEX_ NOT_FOUND	FB 입력 단자 Node 번호가 실제 모듈 Node 번호와 대응하지 않습니다.
16899	ERR_ECAT_SPLC_MODULE_TYPE	FB 입력 항목 Node 번호와 실제 모듈 유형이 동일하지 않습니다
16900	ERR_ECAT_SPLC_FIFO_FULL	SoftPLC Kernel Buffer 가 가득 차서 새로운 명령을 수신할 수 없습니다.
16901	ERR_ECAT_SPLC_PARAM_INPUT_ ERROR	FB 입력 단자 파라미터 오류입니다.
16902	ERR_ECAT_SPLC_AXISNUM_ ERROR	FB 입력 단자 축 수 오류입니다.
16903	ERR_ECAT_SPLC_RECORD_FIFO _FULL	SPLC 기록 기능 FB 의 Buffer 가 가득 찼습니다
16904	ERR_ECAT_SPLC_INVALID_ MEMORY_PTR	SPLC 의 메모리 지표에 오류가 발생했습니다
17408	ERR_ECAT_SPLC_LICENCE_ ERROR	사용하는 Master 또는 하드웨어에 SPLC 사용 권한이 없습니다.
17920	ERR_ECAT_SPLC_EVENT_FAILED	SPLC 이벤트 연결 실패
17921	ERR_ECAT_SPLC_ SHAREMEMORY_FAILED	SPLC ShareMemory 생성 실패

반환 코드 오류 (십진법)	오류 코드 표시명	오류 문제 설명
17922	ERR_ECAT_SPLC_MASTER_NOT_FOUND	SPLC 카드 on 실패
17923	ERR_ECAT_SPLC_MASTER_CONNECT_FAILED	SPLC 연결 실패
17924	ERR_ECAT_SPLC_MASTER_START_SPLC_FAILED	SPLC 상태에서 대기 모드 전환으로 실패
17925	ERR_ECAT_SPLC_MASTER_OPEN_eCLR_FAILED	SPLC 실행 eCLR 실패
17926	ERR_ECAT_SPLC_MASTER_DISCONNECT_FAILED	SPLC 분리 실패
17927	ERR_ECAT_SPLC_MASTER_CLOSE_FAILED	SPLC 카드 off 실패
32768	ERR_PATH_BOARD_INIIT	내부 보간기만 사용 가능합니다
32769	ERR_PATH_BOARD_NUMBER	내부 보간기만 사용 가능합니다
32770	ERR_PATH_INITIAL_BOARD_NUMBER	내부 보간기만 사용 가능합니다
32771	ERR_PATH_BASE_ADDR_ERROR	내부 보간기만 사용 가능합니다
32772	ERR_PATH_BASE_ADDR_CONFLICT	내부 보간기만 사용 가능합니다
32773	ERR_PATH_DUPLICATE_BOARD_SETTING	내부 보간기만 사용 가능합니다
32774	ERR_PATH_DUPLICATE_IRQ_SETTING	내부 보간기만 사용 가능합니다
32775	ERR_PATH_ENC_NUMBER	내부 보간기만 사용 가능합니다
32776	ERR_PATH_MODULE_NUMBER	내부 보간기만 사용 가능합니다
32777	ERR_PATH_TIMER_VALUE	내부 보간기만 사용 가능합니다
32778	ERR_PATH_ENABLE	내부 보간기만 사용 가능합니다
32779	ERR_PATH_RANGE	내부 보간기만 사용 가능합니다
32780	ERR_PATH_MEMORY_ALLOC	내부 보간기만 사용 가능합니다
32781	ERR_PATH_MOTION_BUSY	내부 보간기만 사용 가능합니다
32782	ERR_PATH_MOTION_NO_START	내부 보간기만 사용 가능합니다
32783	ERR_PATH_WRONG_SPEED	내부 보간기만 사용 가능합니다
32784	ERR_PATH_WRONG_ACCTIME	내부 보간기만 사용 가능합니다
32785	ERR_PATH_IO_ALAM	내부 보간기만 사용 가능합니다
32786	ERR_PATH_OPEN_FILE_FAILED	내부 보간기만 사용 가능합니다

반환 코드 오류 (십진법)	오류 코드 표시명	오류 문제 설명
32787	ERR_PATH_MEMORY_ALLOCATE	내부 보간기만 사용 가능합니다
32788	ERR_PATH_MEMORY_NOT_FREE	내부 보간기만 사용 가능합니다
32789	ERR_PATH_OUTPUT_FILE_NOT_CLOSE	내부 보간기만 사용 가능합니다
32790	ERR_PATH_MOVE_AXIS_NOT_MATCH	내부 보간기만 사용 가능합니다
32791	ERR_PATH_PITCH_ZERO	내부 보간기만 사용 가능합니다
32792	ERR_PATH_TIMEOUT	내부 보간기만 사용 가능합니다
32793	ERR_PATH_PCI_BIOS_NOT_EXIST	내부 보간기만 사용 가능합니다
32794	ERR_PATH_BUFFER_FULL	내부 보간기만 사용 가능합니다
32795	ERR_PATH_ERROR	내부 보간기만 사용 가능합니다
32796	ERR_PATH_REACH_SWLIMIT	내부 보간기만 사용 가능합니다
32797	ERR_PATH_NO_SUPPRT_MODE	내부 보간기만 사용 가능합니다
32798	ERR_PATH_AXIS_CORRELATION	내부 보간기만 사용 가능합니다
32799	ERR_PATH_FEEDHOLD_SUPPROT	내부 보간기만 사용 가능합니다
32800	ERR_PATH_SD_STOP_ON	내부 보간기만 사용 가능합니다
32801	ERR_PATH_VELOCITY_CHANGE_SUPER	내부 보간기만 사용 가능합니다
32802	ERR_PATH_COMMAND_SET	내부 보간기만 사용 가능합니다
32803	ERR_PATH_SDO_MESSAGE_CHOKE	내부 보간기만 사용 가능합니다
32804	ERR_PATH_VELOCITY_CHANGE_BUFFER_FEEDHOLD	내부 보간기만 사용 가능합니다
32805	ERR_PATH_VELOCITY_CHANGE_SYNC_MOVE	내부 보간기만 사용 가능합니다
32806	ERR_PATH_VELOCITY_CHANGE_SD_ON	내부 보간기만 사용 가능합니다
32807	ERR_PATH_POS_CHANGE_MODE	내부 보간기만 사용 가능합니다
32808	ERR_PATH_BUFFER_LENGTH	내부 보간기만 사용 가능합니다
32809	ERR_PATH_2SEG_DISTANCE	내부 보간기만 사용 가능합니다
32810	ERR_PATH_ARC_CENTER_POSITION	내부 보간기만 사용 가능합니다
32811	ERR_PATH_ARC_END_POSITION	내부 보간기만 사용 가능합니다
32812	ERR_PATH_ARC_ANGLE_CALC	내부 보간기만 사용 가능합니다
32813	ERR_PATH_ARC_RADIUS_CALC	내부 보간기만 사용 가능합니다

반환 코드 오류 (십진법)	오류 코드 표시명	오류 문제 설명
32814	ERR_PATH_GEAR_SETTING	내부 보간기만 사용 가능합니다
32815	ERR_PATH_CAM_TABLE	내부 보간기만 사용 가능합니다
32816	ERR_PATH_AXES_NUMBER	내부 보간기만 사용 가능합니다
32817	ERR_PATH_SPIRAL_END_ POSITION	내부 보간기만 사용 가능합니다
32818	ERR_PATH_SPEED_MODE_SLAVE	내부 보간기만 사용 가능합니다
32819	ERR_PATH_SPEED_MODE_SET_ SLAVE	내부 보간기만 사용 가능합니다
32820	ERR_PATH_VELOCITY_CHANGE	내부 보간기만 사용 가능합니다
32821	ERR_PATH_BACKLASH_STEP	내부 보간기만 사용 가능합니다
32822	ERR_PATH_BACKLASH_STATUS	내부 보간기만 사용 가능합니다
32823	ERR_PATH_DIST_OVER	내부 보간기만 사용 가능합니다
32824	ERR_PATH_ECAT_DLL_ERROR_ CODE	내부 보간기만 사용 가능합니다
32825	ERR_PATH_ECAT_NEED_ENABLE	먼저 활성화해야만 사용이 가능합니다
32826	ERR_PATH_ECAT_ECAM_ENABLE	ECAM 활성화 중이므로 사용할 수 없습니다
32827	ERR_PATH_ECAT_ECAM_ MASTERSOURCE	ECAM 의 주축을 설정하지 않았습니다
53248	ERR_RTX_RTSS_LOAD	RTSS 를 on 또는 off 할 수 없습니다.
53249	ERR_RTX_CONNECT_LINK_ FAILED	RTSS 와의 공유 메모리 연결에 실패했거나 Security 를 통과하지 않았습니다.
53250	ERR_RTX_EVENT_FAILED	RTSS 와의 이벤트 연결에 실패했습니다.
53251	ERR_RTX_CONNECT_FAILED	RTSS 와의 공유 메모리 응답 확인에 실패했습니다.
53252	ERR_RTX_CONFIG_EDITED	컴퓨터를 다시 시작해야만 RTX 기본 설정이 가능합니다.
53253	ERR_RTX_SECURITY_FAILED	소프트웨어 복제품을 사용 중일 수 있습니다.
53254	ERR_RTX_COMMANDING	RTX 전용 특수 명령 시스템의 응답이 없습니다.
53255	ERR_RTX_SYSTEM_NOT_ SUPPORT	해당 API 는 사용자의 RTSS 에서 지원하지 않습니다.
53256	ERR_RTX_NOT_SUPPORT	해당 API 는 RTX version 에서 지원하지 않습니다.

반환 코드 오류 (십진법)	오류 코드 표시명	오류 문제 설명
53257	ERR_RTX_THREAD_CREATE_ FAILED	대기 연결 상태의 RTX Thread 실행 실패
53258	ERR_RTX_RTSS_START_FAILED	RTX 시스템 서버 실행 실패
53504	ERR_RTX_WIN32_SYSTEM_NOT_ SUPPORT	Win32 시스템은 Cycle Callback 기능을 지원하지 않습니다.
53505	ERR_RTX_CALLBACK_CLOSE	Cycle Callback 기능이 활성화되면 Callback Function 을 변경할 수 없습니다.
53506	ERR_RTX_CALLBACK_FUNCTION	Callback Function 을 정확히 설정하지 않을 경우, Cycle_Callback 기능을 활성화할 수 없습니다.
53507	ERR_RTX_CALLBACK_THREAD	Callback 기능이 활성화될 때, Thread 가 제대로 실행되지 않았습니다.
53760	ERR_RTX_ERRORLOG_NOT_ ENABLE	API Error Log 기록 기능을 사용하지 않았습니다.
53761	ERR_RTX_ERRORLOG_COUNT_ ERROR	Error Log Count 지정 오류입니다.
57344	ERR_CARD_NO_FOUND	카드를 찾을 수 없습니다.
57345	ERR_CARD_NO_RESPONSE	카드가 응답하지 않습니다.
57346	ERR_CARD_CONNECT_FAILED	드라이브 프로그램과의 연결 오류입니다.
57347	ERR_CARD_MEMORY_NOT_ENO UGH	Record 기능 사용 시, 수량이 축 카드 제한치(24 모듈)를 초과했습니다.
57348	ERR_CARD_LOAD_AUTOCONFIG_ FILE	구성 파일 확인에 실패했습니다.
57349	ERR_CARD_SECURITY_FAILED	보안 시스템 인증에 실패했습니다.
57350	ERR_CARD_UPGRADE_CREATE_ THREAD_FAILED	업데이트 윈도우 생성에 실패했습니다.
57351	ERR_CARD_UPGRADE_NO_ RESPONSE	업데이트가 응답하지 않습니다.
57352	ERR_CARD_UPGRADE_NO_ RESOURSE	업데이트 리소스 확인 실패.
57353	ERR_CARD_UPGRADE_LOAD_ RESOURSE	업데이트 리소스 확인 실패.
57354	ERR_CARD_UPGRADE_TIMEOUT	업데이트 기간이 만료되었습니다.
57355	ERR_CARD_UPGRADE_FAILED	업데이트에 실패했습니다.
61440	ERR_ECAT_DLL_IS_USED	EtherCAT_DLL 이 중복 실행되었습니다.

반환 코드 오류 (십진법)	오류 코드 표시명	오류 문제 설명
61441	ERR_ECAT_NO_DLL_FOUND	EtherCAT_DLL 과 RTSS, 축 카드 DLL 의 연결이 실패했습니다.
61442	ERR_ECAT_NO_RTSS_DLL_ FOUND	EtherCAT_DLL 과 RTSSDLL 의 연결이 실패했습니다.
61443	ERR_ECAT_NO_CARD_DLL_ FOUND	EtherCAT_DLL 과 축 카드 DLL 의 연결이 실패했습니다.
61444	ERR_ECAT_NO_ESI_DLL_FOUND,	EtherCAT DLL 과 ESI 가 조작한 DLL 의 연결이 실패했습니다.
61445	ERR_ECAT_SAME_CARD_ NUMBER	RTSS 또는 축 카드 번호가 중복됩니다.
61446	ERR_ECAT_CARDNO_ERROR	존재하지 않는 카드 번호를 사용했습니다.
61447	ERR_ECAT_GET_DLL_PATH	DLL 경로를 확인할 수 없습니다.
61448	ERR_ECAT_GET_DLL_VERSION	DLL 버전을 확인할 수 없습니다.
61449	ERR_ECAT_NOT_SUPPORT	EtherCAT 은 현재 해당 DMC 유형의 API 를 지원하지 않습니다.
65535	ERR_ECAT_LOADLIB_EMPTY	RTSS 에서 DLL 리소스 불러오기 실패

34.2 상세한 오류 코드 내용

34

10 진수 번호	0	16 진수 번호	0x0
오류 코드 식별 번호	ERR_ECAT_NO_ERROR		
오류 설명	오류가 없습니다		
오류 해결	없음		

10 진수 번호	1	16 진수 번호	0x1
오류 코드 식별 번호	ERR_ECAT_HW_NO_INITIALIZE		
오류 설명	하드웨어가 아직 초기화되지 않아 다른 API 를 호출합니다		
오류 해결	하드웨어 동작을 초기화합니다		

10 진수 번호	2	16 진수 번호	0x2
오류 코드 식별 번호	ERR_ECAT_HW_PWM_INITIAL		
오류 설명	하드웨어 초기화 실패		
오류 해결	하드웨어 네트워크 통신 설정이 불가능하므로 실제 연결 상태의 정상 여부를 검사합니다		

10 진수 번호	3	16 진수 번호	0x3
오류 코드 식별 번호	ERR_ECAT_HW_HAS_INITIALIZED		
오류 설명	하드웨어 초기화를 반복합니다		
오류 해결	초기화 중복 취소 프로그램		

10 진수 번호	16	16 진수 번호	0x10
오류 코드 식별 번호	ERR_ECAT_EEPROM_READ		
오류 설명	모듈 EEPROM 확인 실패		
오류 해결	하드웨어 네트워크 통신 설정이 불가능하므로 실제 연결 상태의 정상 여부를 검사합니다		

10 진수 번호	17	16 진수 번호	0x11
오류 코드 식별 번호	ERR_ECAT_EEPROM_WRITE		
오류 설명	모듈 EEPROM 입력 실패		
오류 해결	하드웨어 네트워크 통신 설정이 불가능하므로 실제 연결 상태의 정상 여부를 검사합니다		

10 진수 번호	18	16 진수 번호	0x12
오류 코드 식별 번호	ERR_ECAT_ENVIRONMENT_RECORD_DISABLE		
오류 설명	파일 기능 무효화를 설정합니다		
오류 해결	시스템 리소스 옵션 오류		

10 진수 번호	19	16 진수 번호	0x13
오류 코드 식별 번호	ERR_ECAT_ENVIRONMENT_RECORD_NO_MATCH		
오류 설명	설정한 파일 환경과 온라인 환경이 맞지 않습니다		
오류 해결	하드웨어 네트워크 통신 설정이 불가능하므로 실제 연결 상태의 정상 여부를 검사합니다		

10 진수 번호	20	16 진수 번호	0x14
오류 코드 식별 번호	ERR_ECAT_ENVIRONMENT_RECORD_FILE_OPEN		
오류 설명	파일 실행 설정 실패		
오류 해결	해당 경로에 DAT 파일을 제대로 저장했는지 검사합니다		

10 진수 번호	21	16 진수 번호	0x15
오류 코드 식별 번호	ERR_ECAT_ENVIRONMENT_RECORD_NOT_CREATE		
오류 설명	설정한 파일 구조가 생성되지 않았습니다		
오류 해결	정확한 Initial 프로세스를 실행하십시오		

10 진수 번호	23	16 진수 번호	0x17
오류 코드 식별 번호	ERR_ECAT_DEVICE_OPEN		
오류 설명	Master 초기화 설정 오류		
오류 해결	Master 구조 초기화 오류, 하드웨어 통신 연결의 정상 여부를 검사합니다		

10 진수 번호	24	16 진수 번호	0x18
오류 코드 식별 번호	ERR_ECAT_NO_DEVICE		
오류 설명	Master 초기 입력 모듈 정보 오류		
오류 해결	모듈에 대응하는 dat 파일의 존재 여부를 검사합니다		

10 진수 번호	25	16 진수 번호	0x19
오류 코드 식별 번호	ERR_ECAT_NO_MASTER		
오류 설명	Master 가 생성되지 않았습니다		
오류 해결	Master 구조를 생성합니다		

10 진수 번호	26	16 진수 번호	0x1A
오류 코드 식별 번호	ERR_ECAT_NO_SLAVE		
오류 설명	Slave 가 존재하지 않습니다		
오류 해결	하드웨어 네트워크 통신 설정이 불가능하므로 실제 연결 상태의 정상 여부를 검사합니다		

10 진수 번호	27	16 진수 번호	0x1B
오류 코드 식별 번호	ERR_ECAT_UNKNOWN_SLAVE		
오류 설명	알 수 없는 모듈 유형이 존재합니다		
오류 해결	지원하지 않는 모듈을 삭제하십시오		

10 진수 번호	28	16 진수 번호	0x1C
오류 코드 식별 번호	ERR_ECAT_IST_CREATE		
오류 설명	IST 생성 실패		
오류 해결	시스템 리소스 옵션 오류		

10 진수 번호	29	16 진수 번호	0x1D
오류 코드 식별 번호	ERR_ECAT_MASTER_CREATE		
오류 설명	Master 생성 실패		
오류 해결	Master 구조 초기화 오류, 하드웨어 통신 연결의 정상 여부를 검사합니다		

10 진수 번호	30	16 진수 번호	0x1D
오류 코드 식별 번호	ERR_ECAT_MASTER_REQUEST_STATE		
오류 설명	Master 전환 상태값 오류		
오류 해결	전환된 상태 값을 수정합니다		

10 진수 번호	31	16 진수 번호	0x1F
오류 코드 식별 번호	ERR_ECAT_MASTER_OPERATION_NOT_READY		
오류 설명	Master 가 OP 상태로 전환되지 않았습니다		
오류 해결	OP 상태로 전환될 때까지 대기합니다		

10 진수 번호	32	16 진수 번호	0x20
오류 코드 식별 번호	ERR_ECAT_DELTA_NODE_ID_ALIAS_READ		
오류 설명	델타 드라이브 닉네임 확인 실패		
오류 해결	하드웨어 네트워크 통신 설정이 불가능하므로 실제 연결 상태의 정상 여부를 검사합니다 델타 드라이브 펌웨어 버전이 공용 버전인지 확인합니다		

10 진수 번호	33	16 진수 번호	0x21
오류 코드 식별 번호	ERR_ECAT_MASTER_GET_SERIAL_NO_WRONG		
오류 설명	PAC/축 카드 시퀀스 번호 확인 실패		
오류 해결	현재 사용 중인 PAC/축 카드가 최신 FPGA 펌웨어로 업데이트되었는지 확인하시기 바랍니다		

10 진수 번호	34	16 진수 번호	0x22
오류 코드 식별 번호	ERR_ECAT_MASTER_GET_SERIAL_NO_TIMEOUT		
오류 설명	PAC/축 카드 시퀀스 번호 확인 시간 초과		
오류 해결	현재 사용 중인 PAC/축 카드가 최신 FPGA 펌웨어로 업데이트되었는지 확인하시기 바랍니다		

10 진수 번호	128	16 진수 번호	0x80
오류 코드 식별 번호	ERR_ECAT_PIPELINE_CORE_TIMER_CREATE		
오류 설명	스케줄러 키 이벤트 생성 실패		

10 진수 번호	129	16 진수 번호	0x81
오류 코드 식별 번호	ERR_ECAT_PIPELINE_CREATE		
오류 설명	스케줄러 기능 생성 실패		
오류 해결	시스템 리소스 옵션 오류		

10 진수 번호	130	16 진수 번호	0x82
오류 코드 식별 번호	ERR_ECAT_COMMAND_ENQUEUE		
오류 설명	버퍼 영역에 스케줄러 명령 새로 추가하기 실패		
오류 해결	모듈은 해당 명령의 추가 기능을 지원하지 않습니다		

10 진수 번호	131	16 진수 번호	0x83
오류 코드 식별 번호	ERR_ECAT_API_BUFFER_ENQUEUE		
오류 설명	버퍼 영역에 스케줄러 명령 새로 추가하기 실패		
오류 해결	모듈은 해당 명령의 추가 기능을 지원하지 않습니다		

10 진수 번호	256	16 진수 번호	0x100
오류 코드 식별 번호	ERR_ECAT_NODE_ID		
오류 설명	모듈 Node 번호 오류		
오류 해결	설정 파라미터 검사		

10 진수 번호	257	16 진수 번호	0x101
오류 코드 식별 번호	ERR_ECAT_SLOT_ID		
오류 설명	모듈 슬롯 번호 오류		
오류 해결	설정 파라미터 검사		

10 진수 번호	258	16 진수 번호	0x102
오류 코드 식별 번호	ERR_ECAT_SDO_DOWNLOAD		
오류 설명	SDO 데이터 설정 오류		
오류 해결	설정 파라미터 검사		

10 진수 번호	259	16 진수 번호	0x103
오류 코드 식별 번호	ERR_ECAT_SDO_UPLOAD		
오류 설명	SDO 데이터 확인 오류		
오류 해결	설정 파라미터 검사		

10 진수 번호	260	16 진수 번호	0x104
오류 코드 식별 번호	ERR_ECAT_GET_PROCESS_DATA		
오류 설명	지정한 PDO 데이터를 확인할 수 없습니다		
오류 해결	설정 파라미터 검사		

10 진수 번호	512	16 진수 번호	0x200
오류 코드 식별 번호	ERR_ECAT_DIO_CHANNEL_INVALID		
오류 설명	DIO Channel 설정 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	513	16 진수 번호	0x201
오류 코드 식별 번호	ERR_ECAT_ADDA_CHANNEL_INVALID		
오류 설명	AIO Channel 설정 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	514	16 진수 번호	0x202
오류 코드 식별 번호	ERR_ECAT_MOTION_NOT_FINISHED		
오류 설명	위치 설정 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	515	16 진수 번호	0x203
오류 코드 식별 번호	ERR_ECAT_SET_PULSE_MODE		
오류 설명	모듈 입출력 모드 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	516	16 진수 번호	0x204
오류 코드 식별 번호	ERR_ECAT_SET_SOFTLIMIT		
오류 설명	소프트웨어 limit 설정 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	517	16 진수 번호	0x205
오류 코드 식별 번호	ERR_ECAT_SET_POSITION		
오류 설명	설정 위치 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	518	16 진수 번호	0x206
오류 코드 식별 번호	ERR_ECAT_GET_SPEED		
오류 설명	PDO 속도 데이터 확인 불가 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	519	16 진수 번호	0x207
오류 코드 식별 번호	ERR_ECAT_GET_MCDONE		
오류 설명	PDO StatusWord 데이터 확인 불가 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	520	16 진수 번호	0x208
오류 코드 식별 번호	ERR_ECAT_SET_HOME_CONFIG		
오류 설명	Homing 파라미터 설정 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	521	16 진수 번호	0x209
오류 코드 식별 번호	ERR_ECAT_SET_P2P_CONFIG		
오류 설명	PP 동작 파라미터 설정 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	522	16 진수 번호	0x20a
오류 코드 식별 번호	ERR_ECAT_SET_PT_CONFIG		
오류 설명	PT 동작 파라미터 설정 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	523	16 진수 번호	0x20b
오류 코드 식별 번호	ERR_ECAT_SET_PV_CONFIG		
오류 설명	PV 동작 파라미터 설정 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	524	16 진수 번호	0x20c
오류 코드 식별 번호	ERR_ECAT_SET_CSP_CONFIG		
오류 설명	CSP 단축 직선 동작 파라미터 설정 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	525	16 진수 번호	0x20d
오류 코드 식별 번호	ERR_ECAT_SET_MULTI_AXES_LINE_CONFIG		
오류 설명	CSP 다축 직선 동작 파라미터 설정 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	526	16 진수 번호	0x20e
오류 코드 식별 번호	ERR_ECAT_SET_MULTI_AXES_ARC_CONFIG		
오류 설명	CSP 양축 원형 동작 파라미터 설정 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	768	16 진수 번호	0x300
오류 코드 식별 번호	ERR_ECAT_MD1_SET_GEAR		
오류 설명	설정 모드 1 의 기어비 파라미터 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	769	16 진수 번호	0x301
오류 코드 식별 번호	ERR_ECAT_MD1_SET_P_CHANGE		
오류 설명	설정 모드 1 의 위치 변화 파라미터 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	770	16 진수 번호	0x302
오류 코드 식별 번호	ERR_ECAT_MD1_SET_V_CHANGE		
오류 설명	설정 모드 1 의 속도 변화 파라미터 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	771	16 진수 번호	0x303
오류 코드 식별 번호	ERR_ECAT_MD1_SET_SOFTLIMIT		
오류 설명	설정 모드 1 의 소프트웨어 limit 의 파라미터 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	772	16 진수 번호	0x304
오류 코드 식별 번호	ERR_ECAT_MD1_SET_SLD		
오류 설명	설정 모드 1 의 감속 정지 파라미터 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	773	16 진수 번호	0x305
오류 코드 식별 번호	ERR_ECAT_MD1_SET_HOME_CONFIG		
오류 설명	설정 모드 1 의 Homing 파라미터 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	774	16 진수 번호	0x306
오류 코드 식별 번호	ERR_ECAT_MD1_SET_P2P_CONFIG		
오류 설명	설정 모드 1 의 단축 동작 파라미터 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	775	16 진수 번호	0x307
오류 코드 식별 번호	ERR_ECAT_MD1_SET_V_MOVE_CONFIG		
오류 설명	설정 모드 1 의 단축 연속 동작 파라미터 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	776	16 진수 번호	0x308
오류 코드 식별 번호	ERR_ECAT_MD1_SET_LINE_CONFIG		
오류 설명	설정 모드 1 의 다축 동작 파라미터 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	777	16 진수 번호	0x309
오류 코드 식별 번호	ERR_ECAT_MD1_SET_ARC_CONFIG		
오류 설명	설정 모드 1 의 원형 동작 파라미터 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	1024	16 진수 번호	0x400
오류 코드 식별 번호	ERR_ECAT_PATH_NOT_SUPPORT		
오류 설명	CSP 동작 모드 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	1025	16 진수 번호	0x401
오류 코드 식별 번호	ERR_ECAT_PATH_AXIS_NUM		
오류 설명	CSP 동작 축수 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	1026	16 진수 번호	0x402
오류 코드 식별 번호	ERR_ECAT_PATH_AXIS_NO		
오류 설명	CSP 동작 축 번호 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	1027	16 진수 번호	0x403
오류 코드 식별 번호	ERR_ECAT_PATH_PARA		
오류 설명	CSP 동작 파라미터 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	1028	16 진수 번호	0x404
오류 코드 식별 번호	ERR_ECAT_PATH_ISR_FUNC_EVENT		
오류 설명	ISR 함수 지표 이벤트 연결 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	1029	16 진수 번호	0x405
오류 코드 식별 번호	ERR_ECAT_PATH_AXISNO_UNDER_GROUP		
오류 설명	CSP 그룹 축 번호 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	1152	16 진수 번호	0x480
오류 코드 식별 번호	ERR_ECAT_PATH_ROBOT_NOT_SUPPORT		
오류 설명	SRC 동작 모드 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	1153	16 진수 번호	0x481
오류 코드 식별 번호	ERR_ECAT_PATH_ROBOT_STOP		
오류 설명	SRC 정지 파라미터 오류 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	1154	16 진수 번호	0x482
오류 코드 식별 번호	ERR_ECAT_PATH_ROBOT_AXIS_OVERFLOW		
오류 설명	SRC 사용 축 수가 실제 축 수를 초과함 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	1155	16 진수 번호	0x483
오류 코드 식별 번호	ERR_ECAT_PATH_ROBOT_BUFFER_FULL		
오류 설명	SRC 모션 명령 Buffer 가 가득 참 (WIN32 버전에서만 사용)		
오류 해결	내부 테스트용 입니다		

10 진수 번호	3840	16 진수 번호	0xF00
오류 코드 식별 번호	ERR_ESI_INITIAL		
오류 설명	ESI Parser 가 초기화를 진행하지 않았거나 초기화에 실패했습니다		
오류 해결	초기화를 다시 실행합니다		

10 진수 번호	3841	16 진수 번호	0xF01
오류 코드 식별 번호	ERR_ESI_OPEN_DEVICE		
오류 설명	적절한 하드웨어 설정을 찾을 수 없습니다		
오류 해결	수신 ESI 파일 경로를 변경합니다		

10 진수 번호	3842	16 진수 번호	0xF02
오류 코드 식별 번호	ERR_ESI_CREATE_CANOPEN_OD_LIST		
오류 설명	지원하는 CANOpen OD 리스트를 생성할 수 없습니다		
오류 해결	ESI File 포맷의 오류 여부를 검사합니다		

10 진수 번호	3843	16 진수 번호	0xF03
오류 코드 식별 번호	ERR_ESI_NO_DATA_TYPE_INFO		
오류 설명	DataType 섹터 정보를 생성할 수 없습니다		
오류 해결	ESI File 포맷의 오류 여부를 검사합니다		

10 진수 번호	3844	16 진수 번호	0xF04
오류 코드 식별 번호	ERR_ESI_NO_OBJECT_INFO		
오류 설명	Object 섹터 정보를 생성할 수 없습니다		
오류 해결	ESI File 포맷의 오류 여부를 검사합니다		

10 진수 번호	3845	16 진수 번호	0xF05
오류 코드 식별 번호	ERR_ESI_CREATE_SYNC_MANAGER		
오류 설명	SM 섹터 또는 RxPDO / TxPDO 섹터 정보를 생성할 수 없습니다		
오류 해결	ESI File 포맷의 오류 여부를 검사합니다		

10 진수 번호	3846	16 진수 번호	0xF06
오류 코드 식별 번호	ERR_ESI_CREATE_FMMU_CONTROL		
오류 설명	Fmmu 섹터 정보를 생성할 수 없습니다		
오류 해결	ESI File 포맷의 오류 여부를 검사합니다		

10 진수 번호	3847	16 진수 번호	0xF07
오류 코드 식별 번호	ERR_ESI_NO_PDO_CHANNEL		
오류 설명	지정한 PDO Channel No 가 존재하지 않습니다		
오류 해결	입력한 값의 오류 여부를 검사합니다		

10 진수 번호	3848	16 진수 번호	0xF08
오류 코드 식별 번호	ERR_ESI_NO_PDO_MAPPING		
오류 설명	지정한 PDO Channel No 에 OD 가 전혀 없습니다		
오류 해결	입력한 값의 오류 여부를 검사합니다		

10 진수 번호	3843	16 진수 번호	0xF03
오류 코드 식별 번호	ERR_ESI_NO_DATA_TYPE_INFO		
오류 설명	DataType 섹터 정보를 생성할 수 없습니다		
오류 해결	ESI File 포맷의 오류 여부를 검사합니다		

10 진수 번호	3849	16 진수 번호	0xF09
오류 코드 식별 번호	ERR_ESI_PDO_MAPPING_INSERT		
오류 설명	CANOpen OD 를 새로 추가할 수 없습니다		
오류 해결	PDO Channel No 가 잘못되었거나 OD 개수가 제한치에 도달했습니다		

10 진수 번호	3850	16 진수 번호	0xF0A
오류 코드 식별 번호	ERR_ESI_PDO_MAPPING_DELETE		
오류 설명	CANOpen OD 를 삭제할 수 없습니다		
오류 해결	PDO Channel No 가 잘못되었거나 OD 개수가 0 입니다		

10 진수 번호	3851	16 진수 번호	0xF0B
오류 코드 식별 번호	ERR_ESI_CREATE_DISTRIBUTED_CLOCK		
오류 설명	Dc Setting 정보를 생성할 수 없습니다		
오류 해결	ESI File 포맷의 오류 여부를 검사합니다		

10 진수 번호	4080	16 진수 번호	0xFF0
오류 코드 식별 번호	ERR_ESI_ENI_INFORMATION_INITIAL		
오류 설명	ProcessData 또는 InitialCommand 등 이후 정보를 생성할 수 없습니다		
오류 해결	ESI File 포맷의 오류 여부를 검사합니다		

10 진수 번호	4081	16 진수 번호	0xFF1
오류 코드 식별 번호	ERR_ESI_ENI_FILE_INITIAL		
오류 설명	ESI FILE 에 필요한 정보를 생성할 수 없습니다		
오류 해결	ESI File 포맷의 오류 여부를 검사합니다		

10 진수 번호	4082	16 진수 번호	0xFF2
오류 코드 식별 번호	ERR_ESI_ENI_FILE_SAVE		
오류 설명	ESI FILE 파일을 저장할 수 없습니다		
오류 해결	출력 경로가 정확한지 검사합니다		

10 진수 번호	4096	16 진수 번호	0x1000
오류 코드 식별 번호	ERR_ECAT_NO_SLAVE_FOUND		
오류 설명	Slave 에 접속하거나 Slave 를 발견하지 못했습니다		
오류 해결	하드웨어 배선이 정확한지, 모든 모듈이 당사의 지원 규범에 부합하는지 확인하시기 바랍니다. Autoconfig 관련 파일을 업데이트하시기 바랍니다.		

10 진수 번호	4097	16 진수 번호	0x1001
오류 코드 식별 번호	ERR_ECAT_INITIAL_TIMEOUT		
오류 설명	Initial 대기 시간이 다소 깁니다		
오류 해결	다음 조건을 확인하십시오: 1. 현재 프로그램을 종료하고 RTSS 가 제대로 종료되었는지 확인하십시오. 종료되지 않았을 경우 수동으로 RTSS 를 종료해주십시오. 2. 드라이브 또는 모듈 하드웨어에서 사용자가 설정한 DC 시간을 지원하지 않습니까? 3. 드라이브 또는 모듈에서 통신 이상이 발생했습니까? 4. 드라이브 또는 모듈 전원을 재실행한 후 다시 한번 시도해주십시오. 5. EcNavi 를 가장 기본적인 Master 로 설정한 후 다시 한번 시도해주십시오. 6. 해당 문제가 해결되지 않을 경우 당사로 연락 바랍니다.		

10 진수 번호	4098	16 진수 번호	0x1002
오류 코드 식별 번호	ERR_ECAT_MODE_CHANGE_FAILED		
오류 설명	OP 모드 또는 Init 모드로 전환할 수 없습니다		
오류 해결	다음 조건을 확인하십시오: 1. 현재 프로그램을 종료하고 RTSS 가 제대로 종료되었는지 확인하십시오. 종료되지 않았을 경우 수동으로 RTSS 를 종료해주십시오. 2. 드라이브 또는 모듈 하드웨어에서 사용자가 설정한 DC 시간을 지원하지 않습니까? 3. 드라이브 또는 모듈에서 통신 이상이 발생했습니까? 4. 드라이브 또는 모듈 전원을 재실행한 후 다시 한번 시도해주십시오. 5. EcNavi 를 가장 기본적인 Master 로 설정한 후 다시 한번 시도해주십시오. 6. 해당 문제가 해결되지 않을 경우 당사로 연락 바랍니다.		

10 진수 번호	4099	16 진수 번호	0x1003
오류 코드 식별 번호	ERR_ECAT_SLAVE_ID		
오류 설명	전송한 Node 번호에 오류가 있습니다		
오류 해결	해당 Node 번호에 상응하는 Slave 가 존재하는지 확인하시기 바랍니다		

10 진수 번호	4100	16 진수 번호	0x1004
오류 코드 식별 번호	ERR_ECAT_ALIAS_SLAVE_ID		
오류 설명	Node 번호 닉네임이 중복되거나 최대 제한 축수를 초과했습니다		
오류 해결	각 Slave 에 각기 다른 Node 번호 닉네임이 있는지 확인하시기 바랍니다. Node 번호 닉네임은 최대 축 수의 제한치를 초과하여 설정할 수 없습니다. 최대 축 수의 제한치에 대한 내용은 사용하는 플랫폼의 매뉴얼을 참조하시기 바랍니다.		

10 진수 번호	4352	16 진수 번호	0x1100
오류 코드 식별 번호	ERR_ECAT_NEED_INITIAL		
오류 설명	해당 조작을 실행하기 전에 먼저 Initial 을 실행해야 합니다		
오류 해결	Sample 의 정확한 Initial 프로세스를 참고하시기 바랍니다		

10 진수 번호	4353	16 진수 번호	0x1101
오류 코드 식별 번호	ERR_ECAT_NEED_RESET		
오류 설명	해당 API 는 Initial 상태에서 실행할 수 없습니다		
오류 해결	Sample 의 정확한 Reset System 프로세스를 참고하시기 바랍니다		

10 진수 번호	4354	16 진수 번호	0x1102
오류 코드 식별 번호	ERR_ECAT_NEED_CONNECT		
오류 설명	해당 조작을 실행하기 전에 먼저 링크를 생성해야 합니다		
오류 해결	Sample 의 정확한 Initial 프로세스를 참고하시기 바랍니다		

10 진수 번호	4355	16 진수 번호	0x1103
오류 코드 식별 번호	ERR_ECAT_NEED_DC_OP		
오류 설명	DC 교정 완료 후 OP 모드로 전환해야만 실행 가능합니다		
오류 해결	Sample 의 정확한 Initial 프로세스를 참고하시기 바랍니다		

10 진수 번호	4356	16 진수 번호	0x1104
오류 코드 식별 번호	ERR_ECAT_NEED_RALM		
오류 설명	Alarm 이 있을 경우, ResetAlarm 을 해야만 실행 가능합니다		
오류 해결	조작하는 Slave 의 Alarm 유무 여부를 확인하시기 바랍니다. Alarm 이 발생할 경우, ResetAlarm 을 사용하여 삭제해야만 동작 실행이 가능합니다		

10 진수 번호	4357	16 진수 번호	0x1105
오류 코드 식별 번호	ERR_ECAT_NEED_SVON		
오류 설명	먼저 Svon 해야만 실행 가능합니다		
오류 해결	_ECAT_Slave_Motion_Set_Svon 을 통해 Servo 를 실행하시기 바랍니다		

10 진수 번호	4358	16 진수 번호	0x1106
오류 코드 식별 번호	ERR_ECAT_NEED_HOMECONFIG		
오류 설명	HomeMove 를 실행하기 전에 먼저 HomeConfig 를 실행해야 합니다		
오류 해결	Home_move 를 실행하기 전에 Home_config 를 실행했는지 확인하시기 바랍니다		

10 진수 번호	4359	16 진수 번호	0x1107
오류 코드 식별 번호	ERR_ECAT_NEED_STOP		
오류 설명	해당 API 는 Motion 동작을 실행해야만 실행 가능합니다		
오류 해결	매뉴얼의 해당 API 관련 설명을 참조하시기 바랍니다		

10 진수 번호	4608	16 진수 번호	0x1200
오류 코드 식별 번호	ERR_ECAT_RING_BUFFER_FULL		
오류 설명	API Ring Buffer(MailBox)가 가득 찼습니다		
오류 해결	API Buffer 가 제한치에 도달했습니다. _ECAT_Master_Get_Api_BufferLength 를 통해 현재 수량을 확인하시기 바랍니다. 각 플랫폼의 API Buffer 수량에 관한 설명은 각 플랫폼의 매뉴얼을 참조하시기 바랍니다		

10 진수 번호	4609	16 진수 번호	0x1201
오류 코드 식별 번호	ERR_ECAT_API_PARAMETER		
오류 설명	API 파라미터 입력 오류		
오류 해결	해당 API 의 파라미터가 정확한지 확인하시기 바랍니다		

10 진수 번호	4610	16 진수 번호	0x1202
오류 코드 식별 번호	ERR_ECAT_SLAVE_TYPE		
오류 설명	해당 모듈은 해당 API 를 지원하지 않습니다		
오류 해결	해당 API 가 해당 모듈 또는 드라이브에 적합한지 확인하려면 매뉴얼을 참조하시기 바랍니다		

10 진수 번호	4611	16 진수 번호	0x1203
오류 코드 식별 번호	ERR_ECAT_TARGET_REACHED		
오류 설명	Target Reached 이 대기 시간을 초과하여 ON 상태로 전환되지 않았습니다		
오류 해결	연결 상태의 정상 여부와 드라이브의 Alarm 유무 여부를 확인하시기 바랍니다		

10 진수 번호	4612	16 진수 번호	0x1204
오류 코드 식별 번호	ERR_ECAT_MODE_NOT_SUPPORT		
오류 설명	해당 이동 모드에서는 해당 API 를 지원하지 않습니다		
오류 해결	현재 operation mode 는 해당 API 를 지원하지 않습니다. 매뉴얼을 참조하여 설정한 operation mode 가 정확한지 확인하시기 바랍니다; 사용한 조작이 일반 API 구조일 경우 (사용자 정의 PDO 가 아니며, PDO Mapping 에 자유롭게 입력한 조작), 당사에 연락 바랍니다.		

10 진수 번호	4613	16 진수 번호	0x1205
오류 코드 식별 번호	ERR_ECAT_MOTION_TYPE		
오류 설명	해당 축은 해당 동작 모드를 지원하지 않습니다 (Group 의 제한을 받기 때문으로 추정됩니다)		
오류 해결	조작한 Motion 축이 해당 동작 모드를 지원하지 않거나 해당 축이 Group 에 결합되었으므로 일시적으로 조작할 수 없습니다. 해당 동작에 대한 지원 여부를 확인하려면 해당 축의 매뉴얼을 확인하시기 바랍니다. Group 에서 발생한 문제일 경우, 우선 해당 Group 의 활성화를 해제하시기 바랍니다.		

10 진수 번호	4614	16 진수 번호	0x1206
오류 코드 식별 번호	ERR_ECAT_PDO_NOT_MAPPING		
오류 설명	해당 동작 모드의 OD 필수 항목이 PDO 에 매핑되지 않았습니다		
오류 해결	조작한 모듈이 설정 파일에서 해당 동작 모드에 상응하는 기본 항목(OD)을 PDO 에 매핑하지 않았을 경우, DAT 파일을 다시 만들거나 해당 모듈에서 해당 동작 모드를 지원하는지 확인하시기 바랍니다.		

10 진수 번호	4615	16 진수 번호	0x1207
오류 코드 식별 번호	ERR_ECAT_MODULE_REVISION		
오류 설명	해당 모듈의 Revision 버전을 지원하지 않습니다		
오류 해결	해당 모듈 또는 드라이브의 현재 버전에 dat 파일이 포함되지 않았을 경우, 제조업체에 최신 xml 파일과 dat 파일을 요청하시기 바랍니다.		

10 진수 번호	4616	16 진수 번호	0x1208
오류 코드 식별 번호	ERR_ECAT_SPEED_CONTINUE_MODE		
오류 설명	SpeedContinue 에 모드를 사용하지 않은 축이 있습니다		
오류 해결	Continue 모드를 사용한 축 가운데 SpeedContinue 모드를 사용한 축이 있습니다.		

10 진수 번호	4617	16 진수 번호	0x1209
오류 코드 식별 번호	ERR_ECAT_HOME_MODE		
오류 설명	Homing 파라미터의 Mode 설정 오류		
오류 해결	설정 파라미터 검사		

10 진수 번호	4618	16 진수 번호	0x120a
오류 코드 식별 번호	ERR_ECAT_HOME_OFFSET		
오류 설명	Homing 파라미터의 Offset 설정 오류		
오류 해결	설정 파라미터 검사		

10 진수 번호	4619	16 진수 번호	0x120b
오류 코드 식별 번호	ERR_ECAT_HOME_FIRST_SPEED		
오류 설명	Homing 파라미터의 FirstVel 설정 오류		
오류 해결	설정 파라미터 검사		

10 진수 번호	4620	16 진수 번호	0x120c
오류 코드 식별 번호	ERR_ECAT_HOME_SECOND_SPEED		
오류 설명	Homing 파라미터의 SecondVel 설정 오류		
오류 해결	설정 파라미터 검사		

10 진수 번호	4621	16 진수 번호	0x120d
오류 코드 식별 번호	ERR_ECAT_HOME_ACC		
오류 설명	Homing 파라미터의 Acc 설정 오류		
오류 해결	설정 파라미터 검사		

10 진수 번호	4622	16 진수 번호	0x120e
오류 코드 식별 번호	ERR_ECAT_MRAM_INDEX		
오류 설명	정전 시 메모리 유지 Offset 설정 오류		
오류 해결	설정 파라미터 검사		

10 진수 번호	4623	16 진수 번호	0x120f
오류 코드 식별 번호	ERR_ECAT_MRAM_INDEX_OUT_RANGE		
오류 설명	정전 시 메모리 유지 Range 설정 오류		
오류 해결	설정 파라미터 검사		

10 진수 번호	4864	16 진수 번호	0x1300
오류 코드 식별 번호	ERR_ECAT_PDO_TX_FAILED		
오류 설명	PDO 유형의 명령 전송 실패		
오류 해결	다음 항목을 확인해주십시오: 1. 현재 프로그램을 종료하고 RTSS 가 제대로 종료되었는지 확인하십시오. 종료되지 않았을 경우 수동으로 RTSS 를 종료해주십시오. 2. 드라이브 또는 모듈 하드웨어에서 사용자가 설정한 DC 시간을 지원하지 않습니까? 3. 드라이브 또는 모듈에서 통신 이상이 발생했습니까? 4. 드라이브 또는 모듈 전원을 재실행한 후 다시 한번 시도해주십시오. 5. EcNavi 를 가장 기본적인 Master 로 설정한 후 다시 한번 시도해주십시오. 6. 해당 문제가 해결되지 않을 경우 당사로 연락 바랍니다.		

10 진수 번호	4865	16 진수 번호	0x1301
오류 코드 식별 번호	ERR_ECAT_SDO_TIMEOUT		
오류 설명	SDO 의 응답 대기시간이 다소 깁니다		
오류 해결	다음 항목을 확인해주십시오: 1. 현재 프로그램을 종료하고 RTSS 가 제대로 종료되었는지 확인하십시오. 종료되지 않았을 경우 수동으로 RTSS 를 종료해주십시오. 2. 드라이브 또는 모듈 하드웨어에서 사용자가 설정한 DC 시간을 지원하지 않습니까? 3. 드라이브 또는 모듈에서 통신 이상이 발생했습니까? 4. 드라이브 또는 모듈 전원을 재실행한 후 다시 한번 시도해주십시오. 5. EcNavi 를 가장 기본적인 Master 로 설정한 후 다시 한번 시도해주십시오. 6. 해당 문제가 해결되지 않을 경우 당사로 연락 바랍니다.		

10 진수 번호	4866	16 진수 번호	0x1302
오류 코드 식별 번호	ERR_ECAT_SDO_RETURN		
오류 설명	Slave 가 SDO 오류 코드를 반환하면 _ECAT_Slave_CANopen_Get_ErrorCode 를 사용하여 코드를 확인할 수 있습니다		
오류 해결	실행한 OD 코드, 데이터 내용, 데이터 크기가 전부 해당 Slave 규범에 부합하는지 확인하시기 바랍니다		

10 진수 번호	4867	16 진수 번호	0x1303
오류 코드 식별 번호	ERR_ECAT_PDO_RX_FAILED		
오류 설명	PDO 유형의 명령 반환 실패		
오류 해결	다음 항목을 확인해주십시오: 1. 현재 프로그램을 종료하고 RTSS 가 제대로 종료되었는지 확인하십시오. 종료되지 않았을 경우 수동으로 RTSS 를 종료해주십시오. 2. 드라이브 또는 모듈 하드웨어에서 사용자가 설정한 DC 시간을 지원하지 않습니까? 3. 드라이브 또는 모듈에서 통신 이상이 발생했습니까? 4. 드라이브 또는 모듈 전원을 재실행한 후 다시 한번 시도해주십시오. 5. EcNavi 를 가장 기본적인 Master 로 설정한 후 다시 한번 시도해주십시오. 6. 해당 문제가 해결되지 않을 경우 당사로 연락 바랍니다.		

10 진수 번호	4868	16 진수 번호	0x1304
오류 코드 식별 번호	ERR_ECAT_MAILBOX		
오류 설명	MailBox 전송 실패		
오류 해결	다음 항목을 확인해주십시오: 1. 현재 프로그램을 종료하고 RTSS가 제대로 종료되었는지 확인하십시오. 종료되지 않았을 경우 수동으로 RTSS 를 종료해주십시오. 2. 드라이브 또는 모듈 하드웨어에서 사용자가 설정한 DC시간을 지원하지 않습니까? 3. 드라이브 또는 모듈에서 통신 이상이 발생했습니까? 4. 드라이브 또는 모듈 전원을 재실행한 후 다시 한번 시도해주십시오. 5. EcNavi 를 가장 기본적인 Master 로 설정한 후 다시 한번 시도해주십시오. 6. 해당 문제가 해결되지 않을 경우 당사로 연락 바랍니다.		

10 진수 번호	4869	16 진수 번호	0x1305
오류 코드 식별 번호	ERR_ECAT_SDO_BUFFER_FULL		
오류 설명	SDO 의 명령 비상주 영역이 가득 찼습니다		
오류 해결	SDO Buffer 가 제한치에 도달했습니다. 각 플랫폼의 SDO Buffer 수량에 관한 설명은 각 플랫폼의 매뉴얼을 참조하시기 바랍니다		

10 진수 번호	5120	16 진수 번호	0x1400
오류 코드 식별 번호	ERR_ECAT_GROUP_NUMBER		
오류 설명	입력한 GroupNo 가 최대 모듈 수 제한치(10)를 초과했습니다		
오류 해결	현재 최대 GroupNo 제한치는 10 모듈입니다. 제한치를 초과하여 설정하지 마십시오		

10 진수 번호	5121	16 진수 번호	0x1401
오류 코드 식별 번호	ERR_ECAT_GROUP_ENABLE		
오류 설명	해당 Group 이 Enable 또는 Pause 되지 않았습니다		
오류 해결	해당 조작을 실행하기 전에 우선 Group 이 활성화되었는지, 일시 정지 상태로 전환되었는지 확인하시기 바랍니다		

10 진수 번호	5122	16 진수 번호	0x1402
오류 코드 식별 번호	ERR_ECAT_GROUP_PAUSE		
오류 설명	해당 동작은 Group Pause 일 경우에만 사용 가능합니다		
오류 해결	해당 조작은 Group 상태에서 일시 정지일 경우에만 지원 가능합니다		

10 진수 번호	5123	16 진수 번호	0x1403
오류 코드 식별 번호	ERR_ECAT_GROUP_SLAVE		
오류 설명	해당 Slave 가 다른 Group 에 의해 활성화되었거나 해당 Group 내에서 중복 존재합니다		
오류 해결	1 개의 Slave 는 1 개의 Group 내부에서만 활성화할 수 있습니다		

10 진수 번호	5124	16 진수 번호	0x1404
오류 코드 식별 번호	ERR_ECAT_GROUP_MODE		
오류 설명	해당 Group Mode 는 일시적으로 지원하지 않습니다		
오류 해결	선택한 Mode 는 현재 지원하지 않습니다. 조작 매뉴얼과 버전 업데이트 파일을 확인하시기 바랍니다.		

10 진수 번호	5125	16 진수 번호	0x1405
오류 코드 식별 번호	ERR_ECAT_GROUP_ALREADY_USED		
오류 설명	해당 GroupNo 기능이 이미 작용 중입니다		
오류 해결	선택한 Group 이 이미 활성화 되었거나 일시 정지 상태이므로 해당 조작 명령을 지원하지 않습니다; GroupNo 가 정확한지 확인하거나 우선 해당 Group 의 활성화를 해제하시기 바랍니다		

10 진수 번호	5126	16 진수 번호	0x1406
오류 코드 식별 번호	ERR_ECAT_GROUP_TYPE		
오류 설명	Group 기능은 미설정 모드일 때 활성화할 수 없고, 활성화 이후에는 모드를 설정할 수 없습니다		
오류 해결	Group 을 활성화하기 전에 우선 _ECAT_Slave_User_Motion_Control_Set_Type 을 실행하고, 오류 코드가 없는지 확인하시기 바랍니다. _ECAT_Slave_User_Motion_Control_Set_Type 을 실행할 때 Group 은 반드시 활성화 해제 상태여야 합니다.		

10 진수 번호	5127	16 진수 번호	0x1407
오류 코드 식별 번호	ERR_ECAT_GROUP_SVON		
오류 설명	Group 내부의 각 축을 Svon 상태로 만드십시오		
오류 해결	Group 을 활성화하기 전에 모든 Motion 축이 Svon 상태가 되었는지 확인하시기 바랍니다.		

10 진수 번호	5128	16 진수 번호	0x1408
오류 코드 식별 번호	ERR_ECAT_GROUP_ALM		
오류 설명	Group 내부의 축에 Alarm 상태가 나타났습니다		
오류 해결	Group 을 활성화하기 전에 모든 Motion 축이 Alm 이 없는 상태로 유지되고 있는지 확인하시기 바랍니다.		

10 진수 번호	5129	16 진수 번호	0x1409
오류 코드 식별 번호	ERR_ECAT_GROUP_DATA_BUFFER		
오류 설명	Data Buffer 가 가득 찼습니다(799 비트)		
오류 해결	사용자 정의 데이터 모드의 Buffer 정도는 800 이므로 _ECAT_Slave_User_Motion_Control_Get_DataCnt 를 통해 현재 비트 수를 확인할 수 있습니다. 비트 수가 799 미만이어야만 명령을 정확히 전달할 수 있습니다.		

10 진수 번호	5130	16 진수 번호	0x140A
오류 코드 식별 번호	ERR_ECAT_GROUP_TIMEOUT		
오류 설명	Group 조작 시, 하위 계층의 응답이 없습니다		
오류 해결	통신 이상 여부 및 기타 조작의 정상 작동 여부를 확인해주십시오. 원인을 찾을 수 없을 경우 당사로 연락바랍니다		

10 진수 번호	5376	16 진수 번호	0x1500
오류 코드 식별 번호	ERR_ECAT_SERVO_PARA_EMPTY		
오류 설명	해당 드라이브 파라미터가 존재하지 않습니다		
오류 해결	Delta 드라이브에 전달한 명령을 지원하는 파라미터 코드가 있는지 확인하시기 바랍니다		

10 진수 번호	5377	16 진수 번호	0x1501
오류 코드 식별 번호	ERR_ECAT_SERVO_PARA_RO		
오류 설명	해당 드라이브 파라미터는 읽기만 가능합니다		
오류 해결	해당 드라이브 파라미터는 읽기만 가능하므로 입력할 수 없습니다; 드라이브 매뉴얼을 확인하시기 바랍니다		

10 진수 번호	5378	16 진수 번호	0x1502
오류 코드 식별 번호	ERR_ECAT_SERVO_COMPARE_ENABLE		
오류 설명	드라이브 Compare 기능 설정에 실패했습니다		
오류 해결	해당 드라이브 펌웨어 버전이 최신 버전인지 확인하시기 바랍니다		

10 진수 번호	5632	16 진수 번호	0x1600
오류 코드 식별 번호	ERR_ECAT_RECORD_TYPE		
오류 설명	Record 기능이 활성화되었을 때, 트리거할 데이터를 수정할 수 없습니다		
오류 해결	Record 기능 활성화를 해제한 후 다시 설정하시기 바랍니다		

10 진수 번호	5888	16 진수 번호	0x1700
오류 코드 식별 번호	ERR_ECAT_MPG_ENABLE		
오류 설명	해당 MJ 그룹은 이미 활성화되었습니다. 활성화 해제 후 다시 실행하시기 바랍니다		
오류 해결	동일한 MJ 그룹은 동일한 시간에 1 회만 활성화할 수 있습니다. 우선 활성화를 해제한 후 다시 실행하시기 바랍니다.		

10 진수 번호	5889	16 진수 번호	0x1701
오류 코드 식별 번호	ERR_ECAT_MPG_CONFIG		
오류 설명	해당 MJ 그룹의 기본 데이터를 설정하지 않았습니다. 우선 Config 를 실행하시기 바랍니다		
오류 해결	활성화 전에 우선 MJ 그룹을 설정해야 합니다		

10 진수 번호	8192	16 진수 번호	0x2000
오류 코드 식별 번호	ERR_ECAT_ROBOT_TYPE		
오류 설명	해당 API 가 이 Robot Type 을 지원하지 않거나 입력한 Robot Type 이 잘못된 번호입니다		
오류 해결	API 매뉴얼의 Robot 내용을 참조하여, Robot 코드를 확인하시기 바랍니다		

10 진수 번호	8193	16 진수 번호	0x2001
오류 코드 식별 번호	ERR_ECAT_ROBOT_INITIAL		
오류 설명	해당 API 는 우선 Initial Robot System 을 실행해야 합니다		
오류 해결	API 매뉴얼의 Robot 내용을 참조하여, 조작 프로세스를 확인하시기 바랍니다		

10 진수 번호	8194	16 진수 번호	0x2002
오류 코드 식별 번호	ERR_ECAT_ROBOT_UPDATE		
오류 설명	Robot 과의 내부 코어 업데이트 메시지가 나타날 때 오류가 발생했습니다		
오류 해결	통신 이상 여부 및 기타 조작의 정상 작동 여부를 확인해주십시오. 원인을 찾을 수 없을 경우 당사로 연락바랍니다		

10 진수 번호	8195	16 진수 번호	0x2003
오류 코드 식별 번호	ERR_ECAT_ROBOT_FAILED		
오류 설명	해당 API 는 코어와 연결 시 오류가 발생했습니다		
오류 해결	통신 이상 여부 및 기타 조작의 정상 작동 여부를 확인해주십시오. 원인을 찾을 수 없을 경우 당사로 연락바랍니다		

10 진수 번호	8196	16 진수 번호	0x2004
오류 코드 식별 번호	ERR_ECAT_ROBOT_RESET		
오류 설명	해당 API 는 Motion_State Disable 해야만 실행 가능합니다		
오류 해결	API 매뉴얼의 Robot 내용을 참조하여, 조작 프로세스를 확인하시기 바랍니다.		

10 진수 번호	8197	16 진수 번호	0x2005
오류 코드 식별 번호	ERR_ECAT_ROBOT_PARAMETER		
오류 설명	해당 API 에 명령한 파라미터의 오류입니다		
오류 해결	API 매뉴얼의 Robot 내용을 참조하여, 해당 API 에 명령한 관련 파라미터가 정확한지 확인하시기 바랍니다		

10 진수 번호	8198	16 진수 번호	0x2006
오류 코드 식별 번호	ERR_ECAT_ROBOT_IN_MOTION		
오류 설명	로봇이 동작 중입니다		
오류 해결	동작이 완료될 때까지 대기한 후 해당 API 명령을 전송하시기 바랍니다		

10 진수 번호	8199	16 진수 번호	0x2007
오류 코드 식별 번호	ERR_ECAT_ROBOT_BUFFER_FULL		
오류 설명	명령 스케줄링이 가득 찼습니다		
오류 해결	현재 스케줄링이 완료될 때까지 대기한 후 API 명령을 전송하시기 바랍니다		

10 진수 번호	8200	16 진수 번호	0x2008
오류 코드 식별 번호	ERR_ECAT_ROBOT_NEED_SERVO_OFF		
오류 설명	현재 Servo ON 상태입니다		
오류 해결	해당 API 는 ServoOFF 상태에서 설정해야 합니다		

10 진수 번호	8201	16 진수 번호	0x2009
오류 코드 식별 번호	ERR_ECAT_ROBOT_V_CHANGING		
오류 설명	변속 파라미터 설정값이 너무 높습니다		
오류 해결	API 매뉴얼의 Robot 내용을 참조하여, 변속 파라미터 설정 범위를 확인하시기 바랍니다		

10 진수 번호	12288	16 진수 번호	0x3000
오류 코드 식별 번호	ERR_ECAT_SECURITY_OPERATING		
오류 설명	Security 관련 조작이 진행 중입니다. 잠시 후에 다시 실행하시기 바랍니다		
오류 해결	Security 관련 조작은 동일한 시간에 1 개만 실행할 수 있습니다. 매뉴얼 설명을 참조하시기 바랍니다		

10 진수 번호	12289	16 진수 번호	0x3001
오류 코드 식별 번호	ERR_ECAT_SECURITY_NEED_LOGIN		
오류 설명	해당 동작은 로그인한 후에만 조작 가능합니다		
오류 해결	Security 부분 조작은 로그인 과정을 거쳐야 합니다. 상세한 프로세스는 매뉴얼의 설명을 참조하시기 바랍니다		

10 진수 번호	12290	16 진수 번호	0x3002
오류 코드 식별 번호	ERR_ECAT_SECURITY_CONNECT		
오류 설명	Security 코어와의 통신에 실패했습니다		
오류 해결	PAC 또는 PC 를 재부팅한 후 다시 한번 시도해주십시오. 해당 상황이 자주 발생할 경우 당사로 연락 바랍니다		

10 진수 번호	16384	16 진수 번호	0x4000
오류 코드 식별 번호	ERR_ECAT_SPLC_CONNECT_FAILED		
오류 설명	SoftPLC 실행 시 연결에 실패했습니다		
오류 해결	통신 상태가 정상인지, 다른 조작에 반환 오류 코드가 존재하는지 확인하고, 매뉴얼의 SoftPLC 재시작 내용을 참조하시기 바랍니다		

10 진수 번호	16385	16 진수 번호	0x4001
오류 코드 식별 번호	ERR_ECAT_SPLC_RTSS_FAILED		
오류 설명	SoftPLC Kernel 내부 초기화에 실패했습니다		
오류 해결	SoftPLC 실행 권한 보유 여부를 확인한 후 매뉴얼의 SoftPLC 재실행 내용을 참고하시기 바랍니다		

10 진수 번호	16386	16 진수 번호	0x4002
오류 코드 식별 번호	ERR_ECAT_SPLC_KERENL_FILE		
오류 설명	SoftPLC Kernel 실행 시 실패했습니다		
오류 해결	SoftPLC 실행 권한 보유 여부를 확인한 후 매뉴얼의 SoftPLC 재실행 내용을 참고하시기 바랍니다		

10 진수 번호	16387	16 진수 번호	0x4003
오류 코드 식별 번호	ERR_ECAT_SPLC_ONGOING		
오류 설명	SoftPLC 을 사용 중이므로 일시적으로 해당 API 를 조작할 수 없습니다		
오류 해결	API 매뉴얼의 SoftPLC 내용을 참조하여, 관련 조작 프로세스를 확인해주십시오		

10 진수 번호	16388	16 진수 번호	0x4004
오류 코드 식별 번호	ERR_ECAT_SPLC_SDO_FAILED		
오류 설명	SoftPLC SDO 명령 전송에 실패했습니다		
오류 해결	통신 이상 여부 및 기타 조작의 정상 작동 여부를 확인하십시오. 원인을 찾을 수 없을 경우 당사로 연락바랍니다.		

10 진수 번호	16389	16 진수 번호	0x4005
오류 코드 식별 번호	ERR_ECAT_SPLC_ALREADY_RUN		
오류 설명	SoftPLC 이 실행 중이므로 일시적으로 해당 API 를 조작할 수 없습니다		
오류 해결	API 매뉴얼의 SoftPLC 내용을 참조하여, 관련 조작 프로세스를 확인해주십시오		

10 진수 번호	16390	16 진수 번호	0x4006
오류 코드 식별 번호	ERR_ECAT_SPLC_MOTION_CONFIG_RUN		
오류 설명	해당 함수가 실행 중입니다. 종료 확인 후 다시 조작하시기 바랍니다		
오류 해결	API 매뉴얼의 SoftPLC 내용을 참조하여, 관련 조작 프로세스를 확인해주십시오		

10 진수 번호	16391	16 진수 번호	0x4007
오류 코드 식별 번호	ERR_ECAT_SPLC_MOT_MODE		
오류 설명	해당 API 는 해당 상태 머신에서 실행할 수 없습니다		
오류 해결	API 매뉴얼의 SoftPLC 내용을 참조하여, 관련 조작 프로세스를 확인해주십시오		

10 진수 번호	16392	16 진수 번호	0x4008
오류 코드 식별 번호	ERR_ECAT_SPLC_HOMING_ERROR		
오류 설명	Home 동작이 실패했습니다		
오류 해결	서보 Fault 또는 Warning 오류입니다		

10 진수 번호	16393	16 진수 번호	0x4009
오류 코드 식별 번호	ERR_ECAT_SPLC_MOTION_TYPE_ERROR		
오류 설명	동작 모드 오류		
오류 해결	동작 FB 유형이 정확히 일치하는지 확인하시기 바랍니다. SMotion 버전과 PLCOpen 버전 FB 가 있습니다		

10 진수 번호	16394	16 진수 번호	0x400a
오류 코드 식별 번호	ERR_ECAT_SPLC_MOTION_ERROR		
오류 설명	SD_Stop FB 에 의해 동작이 중지되었습니다		
오류 해결	SD_Stop 이 활성화될 때 동작 명령을 전송하면 해당 오류 코드가 나타나며, SD_Stop FB 활성화를 해제하면 정상적인 조작이 가능합니다.		

10 진수 번호	16395	16 진수 번호	0x400b
오류 코드 식별 번호	ERR_ECAT_SPLC_MOTION_Jog_ERROR		
오류 설명	Jog FB 동작 Forward 와 Backward 동시 트리거		
오류 해결	Forward 와 Backward 의 핀 활성화를 해제하시기 바랍니다		

10 진수 번호	16896	16 진수 번호	0x4200
오류 코드 식별 번호	ERR_ECAT_SPLC_AXISREF_STRUCT		
오류 설명	FB 입력 단자에서 잘못된 구조를 사용했습니다		
오류 해결	API 매뉴얼의 SoftPLC 내용을 참조하여, 관련 조작 프로세스를 확인하십시오.		

10 진수 번호	16897	16 진수 번호	0x4201
오류 코드 식별 번호	ERR_ECAT_SPLC_MASTER_INDEX_NOT_FOUND		
오류 설명	FB 입력 단자 카드 번호가 실제 카드 번호와 대응하지 않습니다		
오류 해결	사용하는 카드 번호를 확인하시기 바랍니다		

10 진수 번호	16898	16 진수 번호	0x4202
오류 코드 식별 번호	ERR_ECAT_SPLC_AXIS_INDEX_NOT_FOUND		
오류 설명	FB 입력 단자 Node 번호가 실제 모듈 Node 번호와 대응하지 않습니다		
오류 해결	사용하는 모듈 Node 번호를 확인하시기 바랍니다		

10 진수 번호	16899	16 진수 번호	0x4203
오류 코드 식별 번호	ERR_ECAT_SPLC_MODULE_TYPE		
오류 설명	FB 입력 항목 Node 번호와 실제 모듈 유형이 동일하지 않습니다		
오류 해결	사용하는 모듈 형태를 확인하시기 바랍니다		

10 진수 번호	16900	16 진수 번호	0x4204
오류 코드 식별 번호	ERR_ECAT_SPLC_FIFO_FULL		
오류 설명	SoftPLC Kernel Buffer 가 가득 차서 새로운 명령을 수신할 수 없습니다		
오류 해결	PAC 또는 PC 를 재부팅한 후 다시 한번 시도해주십시오. 해당 상황이 자주 발생할 경우 당사로 연락 바랍니다		

10 진수 번호	16901	16 진수 번호	0x4205
오류 코드 식별 번호	ERR_ECAT_SPLC_PARAM_INPUT_ERROR		
오류 설명	FB 입력 단자 파라미터 오류입니다		
오류 해결	사용하는 파라미터가 정확한지 확인하시기 바랍니다		

10 진수 번호	16902	16 진수 번호	0x4206
오류 코드 식별 번호	ERR_ECAT_SPLC_AXISNUM_ERROR		
오류 설명	FB 입력 단자 축 수 오류입니다		
오류 해결	사용하는 축 수가 정확한지 확인하시기 바랍니다		

10 진수 번호	16903	16 진수 번호	0x4207
오류 코드 식별 번호	ERR_ECAT_SPLC_RECORD_FIFO_FULL		
오류 설명	SPLC 기록 기능 FB 의 Buffer 가 가득 찼습니다		
오류 해결	Ring Buffer 공간이 가득 차서 새로운 계수값을 다시 기록할 수 없습니다. 이미 기록된 계수값을 확인해야만 새로운 계수값으로 업데이트할 수 있습니다.		

10 진수 번호	16904	16 진수 번호	0x4208
오류 코드 식별 번호	ERR_ECAT_SPLC_INVALID_MEMORY_PTR		
오류 설명	SPLC 의 메모리 지표에 오류가 발생했습니다		
오류 해결	MP 에서 설정한 메모리 공간 지표 오류입니다. MP 사용 매뉴얼을 참조하시기 바랍니다.		

10 진수 번호	17408	16 진수 번호	0x4400
오류 코드 식별 번호	ERR_ECAT_SPLC_LICENCE_ERROR		
오류 설명	사용하는 Master 또는 하드웨어에 SPLC 사용 권한이 없습니다		
오류 해결	사용하는 하드웨어 플랫폼과 소프트웨어에 SoftPLC 열기 기능이 있는지 확인하시기 바랍니다		

10 진수 번호	17920	16 진수 번호	0x4600
오류 코드 식별 번호	ERR_ECAT_SPLC_EVENT_FAILED		
오류 설명	SPLC 이벤트 연결 실패		
오류 해결	다음 조건을 확인하십시오: 1. PAC 전원을 재실행한 후 다시 한번 시도해주십시오. 2. 해당 문제가 해결되지 않을 경우 당사로 연락바랍니다.		

10 진수 번호	17921	16 진수 번호	0x4601
오류 코드 식별 번호	ERR_ECAT_SPLC_SHAREMEMORY_FAILED		
오류 설명	SPLC ShareMemory 생성 실패		
오류 해결	다음 조건을 확인하십시오: 1. PAC 전원을 재실행한 후 다시 한번 시도해주십시오. 2. 해당 문제가 해결되지 않을 경우 당사로 연락바랍니다.		

10 진수 번호	17922	16 진수 번호	0x4602
오류 코드 식별 번호	ERR_ECAT_SPLC_MASTER_NOT_FOUND		
오류 설명	SPLC 카드 on 실패		
오류 해결	다음 조건을 확인하십시오: 1. PAC 전원을 재실행한 후 다시 한번 시도해주십시오. 2. EcNavi 를 사용하여 연결 상태가 정상인지 확인해 주십시오. 3. 해당 문제가 해결되지 않을 경우 당사로 연락바랍니다.		

10 진수 번호	17923	16 진수 번호	0x4603
오류 코드 식별 번호	ERR_ECAT_SPLC_MASTER_CONNECT_FAILED		
오류 설명	SPLC 연결 실패		
오류 해결	다음 조건을 확인하십시오: 1. PAC 전원을 재실행한 후 다시 한번 시도해주십시오. 2. EcNavi 를 사용하여 연결 상태가 정상인지 확인해 주십시오. 3. 해당 문제가 해결되지 않을 경우 당사로 연락바랍니다.		

10 진수 번호	17924	16 진수 번호	0x4604
오류 코드 식별 번호	ERR_ECAT_SPLC_MASTER_START_SPLC_FAILED		
오류 설명	SPLC 상태에서 대기 모드 전환으로 실패		
오류 해결	다음 조건을 확인하십시오: 1. PAC 전원을 재실행한 후 다시 한번 시도해주십시오. 2. 해당 문제가 해결되지 않을 경우 당사로 연락바랍니다.		

10 진수 번호	17925	16 진수 번호	0x4605
오류 코드 식별 번호	ERR_ECAT_SPLC_MASTER_OPEN_eCLR_FAILED		
오류 설명	SPLC 실행 eCLR 실패		
오류 해결	다음 조건을 확인하십시오: 1. PAC 전원을 재실행한 후 다시 한번 시도해주십시오. 2. 해당 문제가 해결되지 않을 경우 당사로 연락바랍니다.		

10 진수 번호	17926	16 진수 번호	0x4606
오류 코드 식별 번호	ERR_ECAT_SPLC_MASTER_DISCONNECT_FAILED		
오류 설명	SPLC 분리 실패		
오류 해결	다음 조건을 확인하십시오: 1. PAC 전원을 재실행한 후 다시 한번 시도해주십시오. 2. EcNavi 를 사용하여 연결 상태가 정상인지 확인해 주십시오. 3. 해당 문제가 해결되지 않을 경우 당사로 연락바랍니다.		

10 진수 번호	17927	16 진수 번호	0x4607
오류 코드 식별 번호	ERR_ECAT_SPLC_MASTER_CLOSE_FAILED		
오류 설명	SPLC 카드 off 실패		
오류 해결	다음 조건을 확인하십시오: 1. PAC 전원을 재실행한 후 다시 한번 시도해주십시오. 2. EcNavi 를 사용하여 연결 상태가 정상인지 확인해 주십시오. 3. 해당 문제가 해결되지 않을 경우 당사로 연락바랍니다.		

10 진수 번호	32768	16 진수 번호	0x8000
오류 코드 식별 번호	ERR_PATH_BOARD_INIIT		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32769	16 진수 번호	0x8001
오류 코드 식별 번호	ERR_PATH_BOARD_NUMBER		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32770	16 진수 번호	0x8002
오류 코드 식별 번호	ERR_PATH_INITIAL_BOARD_NUMBER		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32771	16 진수 번호	0x8003
오류 코드 식별 번호	ERR_PATH_BASE_ADDR_ERROR		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32772	16 진수 번호	0x8004
오류 코드 식별 번호	ERR_PATH_BASE_ADDR_CONFLICT		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32773	16 진수 번호	0x8005
오류 코드 식별 번호	ERR_PATH_DUPLICATE_BOARD_SETTING		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32774	16 진수 번호	0x8006
오류 코드 식별 번호	ERR_PATH_DUPLICATE_IRQ_SETTING		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32775	16 진수 번호	0x8007
오류 코드 식별 번호	ERR_PATH_ENC_NUMBER		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32776	16 진수 번호	0x8008
오류 코드 식별 번호	ERR_PATH_MODULE_NUMBER		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32777	16 진수 번호	0x8009
오류 코드 식별 번호	ERR_PATH_TIMER_VALUE		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32778	16 진수 번호	0x800a
오류 코드 식별 번호	ERR_PATH_ENABLE		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32779	16 진수 번호	0x800b
오류 코드 식별 번호	ERR_PATH_RANGE		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32780	16 진수 번호	0x800c
오류 코드 식별 번호	ERR_PATH_MEMORY_ALLOC		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32781	16 진수 번호	0x800d
오류 코드 식별 번호	ERR_PATH_MOTION_BUSY		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32782	16 진수 번호	0x800e
오류 코드 식별 번호	ERR_PATH_MOTION_NO_START		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32783	16 진수 번호	0x800f
오류 코드 식별 번호	ERR_PATH_WRONG_SPEED		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32784	16 진수 번호	0x8010
오류 코드 식별 번호	ERR_PATH_WRONG_ACCTIME		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32785	16 진수 번호	0x8011
오류 코드 식별 번호	ERR_PATH_IO_ALAM		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32786	16 진수 번호	0x8012
오류 코드 식별 번호	ERR_PATH_OPEN_FILE_FAILED		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32787	16 진수 번호	0x8013
오류 코드 식별 번호	ERR_PATH_MEMORY_ALLOCATE		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32788	16 진수 번호	0x8014
오류 코드 식별 번호	ERR_PATH_MEMORY_NOT_FREE		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32789	16 진수 번호	0x8015
오류 코드 식별 번호	ERR_PATH_OUTPUT_FILE_NOT_CLOSE		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32790	16 진수 번호	0x8016
오류 코드 식별 번호	ERR_PATH_MOVE_AXIS_NOT_MATCH		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32791	16 진수 번호	0x8017
오류 코드 식별 번호	ERR_PATH_PITCH_ZERO		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32792	16 진수 번호	0x8018
오류 코드 식별 번호	ERR_PATH_TIMEOUT		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32793	16 진수 번호	0x8019
오류 코드 식별 번호	ERR_PATH_PCI_BIOS_NOT_EXIST		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32794	16 진수 번호	0x801a
오류 코드 식별 번호	ERR_PATH_BUFFER_FULL		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32795	16 진수 번호	0x801b
오류 코드 식별 번호	ERR_PATH_ERROR		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32796	16 진수 번호	0x801c
오류 코드 식별 번호	ERR_PATH_REACH_SWLIMIT		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32797	16 진수 번호	0x801d
오류 코드 식별 번호	ERR_PATH_NO_SUPPRT_MODE		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32798	16 진수 번호	0x801e
오류 코드 식별 번호	ERR_PATH_AXIS_CORRELATION		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32799	16 진수 번호	0x801f
오류 코드 식별 번호	ERR_PATH_FEEDHOLD_SUPPROT		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32800	16 진수 번호	0x8020
오류 코드 식별 번호	ERR_PATH_SD_STOP_ON		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32801	16 진수 번호	0x8021
오류 코드 식별 번호	ERR_PATH_VELOCITY_CHANGE_SUPER		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32802	16 진수 번호	0x8022
오류 코드 식별 번호	ERR_PATH_COMMAND_SET		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32803	16 진수 번호	0x8023
오류 코드 식별 번호	ERR_PATH_SDO_MESSAGE_CHOKE		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32804	16 진수 번호	0x8024
오류 코드 식별 번호	ERR_PATH_VELOCITY_CHANGE_BUFFER_FEEDHOLD		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32805	16 진수 번호	0x8025
오류 코드 식별 번호	ERR_PATH_VELOCITY_CHANGE_SYNC_MOVE		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32806	16 진수 번호	0x8026
오류 코드 식별 번호	ERR_PATH_VELOCITY_CHANGE_SD_ON		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32807	16 진수 번호	0x8027
오류 코드 식별 번호	ERR_PATH_POS_CHANGE_MODE		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32808	16 진수 번호	0x8028
오류 코드 식별 번호	ERR_PATH_BUFFER_LENGTH		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32809	16 진수 번호	0x8029
오류 코드 식별 번호	ERR_PATH_2SEG_DISTANCE		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32810	16 진수 번호	0x802a
오류 코드 식별 번호	ERR_PATH_ARC_CENTER_POSITION		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32811	16 진수 번호	0x802b
오류 코드 식별 번호	ERR_PATH_ARC_END_POSITION		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32812	16 진수 번호	0x802c
오류 코드 식별 번호	ERR_PATH_ARC_ANGLE_CALC		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32813	16 진수 번호	0x802d
오류 코드 식별 번호	ERR_PATH_ARC_RADIUS_CALC		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32814	16 진수 번호	0x802e
오류 코드 식별 번호	ERR_PATH_GEAR_SETTING		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32815	16 진수 번호	0x802f
오류 코드 식별 번호	ERR_PATH_CAM_TABLE		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32816	16 진수 번호	0x8030
오류 코드 식별 번호	ERR_PATH_AXES_NUMBER		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32817	16 진수 번호	0x8031
오류 코드 식별 번호	ERR_PATH_SPIRAL_END_POSITION		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32818	16 진수 번호	0x8032
오류 코드 식별 번호	ERR_PATH_SPEED_MODE_SLAVE		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32819	16 진수 번호	0x8033
오류 코드 식별 번호	ERR_PATH_SPEED_MODE_SET_SLAVE		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32820	16 진수 번호	0x8034
오류 코드 식별 번호	ERR_PATH_VELOCITY_CHANGE		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32821	16 진수 번호	0x8035
오류 코드 식별 번호	ERR_PATH_BACKLASH_STEP		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32822	16 진수 번호	0x8036
오류 코드 식별 번호	ERR_PATH_BACKLASH_STATUS		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32823	16 진수 번호	0x8037
오류 코드 식별 번호	ERR_PATH_DIST_OVER		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32824	16 진수 번호	0x8038
오류 코드 식별 번호	ERR_PATH_ECAT_DLL_ERROR_CODE		
오류 설명	내부 보간기만 사용 가능합니다		
오류 해결	내부 테스트용 입니다		

10 진수 번호	32825	16 진수 번호	0x8039
오류 코드 식별 번호	ERR_PATH_ECAT_NEED_ENABLE		
오류 설명	먼저 활성화해야만 사용이 가능합니다		
오류 해결	해당 API 는 먼저 활성화해야만 사용이 가능합니다		

10 진수 번호	32826	16 진수 번호	0x803a
오류 코드 식별 번호	ERR_PATH_ECAT_ECAM_ENABLE		
오류 설명	ECAM 활성화 중이므로 사용할 수 없습니다		
오류 해결	해당 API 는 ECAM 이 활성화되었을 때 사용할 수 없습니다		

10 진수 번호	32827	16 진수 번호	0x803b
오류 코드 식별 번호	ERR_PATH_ECAT_ECAM_MASTERSOURCE		
오류 설명	ECAM 의 주축을 설정하지 않았습니다		
오류 해결	ECAM 이 활성화되기 전에 주축 소스를 설정해야 합니다		

10 진수 번호	53248	16 진수 번호	0xD000
오류 코드 식별 번호	ERR_RTX_RTSS_LOAD		
오류 설명	RTSS 를 on 또는 off 할 수 없습니다		
오류 해결	RTX 환경이 정확히 생성되었는지 확인해주십시오. 필요 시 재부팅 후 다시 실행해 주십시오		

10 진수 번호	53249	16 진수 번호	0xD001
오류 코드 식별 번호	ERR_RTX_CONNECT_LINK_FAILED		
오류 설명	RTSS 와의 공유 메모리 연결에 실패했거나 Security 를 통과하지 않았습니다		
오류 해결	RTX 환경이 정확히 생성되었는지 확인해주십시오. 필요 시 재부팅 후 다시 실행해 주십시오		

10 진수 번호	53250	16 진수 번호	0xD002
오류 코드 식별 번호	ERR_RTX_EVENT_FAILED		
오류 설명	RTSS 와의 이벤트 연결에 실패했습니다		
오류 해결	RTX 환경이 정확히 생성되었는지 확인해주십시오. 필요 시 재부팅 후 다시 실행해 주십시오		

10 진수 번호	53251	16 진수 번호	0xD003
오류 코드 식별 번호	ERR_RTX_CONNECT_FAILED		
오류 설명	RTSS 와의 공유 메모리 응답 확인에 실패했습니다		
오류 해결	RTX 환경이 정확히 생성되었는지 확인해주십시오. 필요 시 재부팅 후 다시 실행해 주십시오		

10 진수 번호	53252	16 진수 번호	0xD004
오류 코드 식별 번호	ERR_RTX_CONFIG_EDITED		
오류 설명	컴퓨터를 다시 시작해야만 RTX 기본 설정이 가능합니다		
오류 해결	RTX 자체에 일부 파라미터와 EtherCAT 간의 대응 관계가 존재하므로 관련 설정을 임의로 조정하지 마시기 바랍니다; 해당 오류가 나타날 경우, 컴퓨터를 다시 시작하면 관련 파라미터가 자동으로 설정됩니다.		

10 진수 번호	53253	16 진수 번호	0xD005
오류 코드 식별 번호	ERR_RTX_SECURITY_FAILED		
오류 설명	소프트웨어 복제품을 사용 중일 수 있습니다		
오류 해결	Security 오류입니다. 사용하는 하트웨어가 정확한지 확인하시기 바랍니다		

34

10 진수 번호	53254	16 진수 번호	0xD006
오류 코드 식별 번호	ERR_RTX_COMMANDING		
오류 설명	RTX 전용 특수 명령 시스템의 응답이 없습니다		
오류 해결	다음 항목을 확인해주십시오: 1. 현재 프로그램을 종료하고 RTSS 가 제대로 종료되었는지 확인하십시오. 종료되지 않았을 경우 수동으로 RTSS 를 종료해주십시오. 2. 드라이브 또는 모듈 하드웨어에서 사용자가 설정한 DC 시간을 지원하지 않습니까? 3. 드라이브 또는 모듈에서 통신 이상이 발생했습니까? 4. 드라이브 또는 모듈 전원을 재실행한 후 다시 한번 시도해주십시오. 5. EcNavi 를 가장 기본적인 Master 로 설정한 후 다시 한번 시도해주십시오. 6. 해당 문제가 해결되지 않을 경우 당사로 연락 바랍니다.		

10 진수 번호	53255	16 진수 번호	0xD007
오류 코드 식별 번호	ERR_RTX_RTSS_SYSTEM_NOT_SUPPORT		
오류 설명	해당 API 는 사용자의 RTSS 에서 지원하지 않습니다		
오류 해결	매뉴얼의 API 지원 리스트에서 해당 API 가 지원하는 시스템 리스트를 참고해 주십시오		

10 진수 번호	53256	16 진수 번호	0xD008
오류 코드 식별 번호	ERR_RTX_NOT_SUPPORT		
오류 설명	해당 API 는 RTX version 에서 지원하지 않습니다		
오류 해결	매뉴얼의 API 지원 리스트에서 해당 API 가 지원하는 시스템 리스트를 참고해 주십시오		

10 진수 번호	53257	16 진수 번호	0xD009
오류 코드 식별 번호	ERR_RTX_THREAD_CREATE_FAILED		
오류 설명	대기 연결 상태의 RTX Thread 실행 실패		
오류 해결	다음 조건을 확인하십시오: 1. PAC 전원을 재실행한 후 다시 한번 시도해주십시오. 2. 해당 문제가 해결되지 않을 경우 당사로 연락바랍니다.		

10 진수 번호	53258	16 진수 번호	0xD00a
오류 코드 식별 번호	ERR_RTX_RTSS_START_FAILED		
오류 설명	RTX 시스템 서버 실행 실패		
오류 해결	다음 조건을 확인하십시오: 1. PAC 전원을 재실행한 후 다시 한번 시도해주십시오. 2. 해당 문제가 해결되지 않을 경우 당사로 연락바랍니다.		

10 진수 번호	53504	16 진수 번호	0xD100
오류 코드 식별 번호	ERR_RTX_WIN32_SYSTEM_NOT_SUPPORT		
오류 설명	Win32 시스템은 Cycle Callback 기능을 지원하지 않습니다		
오류 해결	사용하는 시스템이 RTX 이고, 호출한 것이 ECAT_RTX_RTDLL.rtdll 이 맞는지 확인하시기 바랍니다		

10 진수 번호	53505	16 진수 번호	0xD101
오류 코드 식별 번호	ERR_RTX_CALLBACK_CLOSE		
오류 설명	Cycle Callback 기능이 활성화되면 Callback Function 을 변경할 수 없습니다		
오류 해결	Cycle Callback 기능 활성화를 해제한 후 다시 설정하시기 바랍니다		

10 진수 번호	53506	16 진수 번호	0xD102
오류 코드 식별 번호	ERR_RTX_CALLBACK_FUNCTION		
오류 설명	Callback Function 을 정확히 설정하지 않을 경우, Cycle_Callback 기능을 활성화할 수 없습니다		
오류 해결	_ECAT_Master_Callback_Set_Function 을 사용하여 호출하려는 Callback 함수를 설정하시기 바랍니다		

10 진수 번호	53507	16 진수 번호	0xD103
오류 코드 식별 번호	ERR_RTX_CALLBACK_THREAD		
오류 설명	Callback 기능이 활성화될 때, Thread 가 제대로 실행되지 않았습니다		
오류 해결	RTX 환경이 정확히 생성되었는지 확인해주십시오. 필요 시 재부팅 후 다시 실행해 주십시오		

10 진수 번호	53760	16 진수 번호	0xD200
오류 코드 식별 번호	ERR_RTX_ERRORLOG_NOT_ENABLE		
오류 설명	API Error Log 기록 기능을 사용하지 않았습니다		
오류 해결	우선 Error Log 기능을 사용해야 합니다		

10 진수 번호	53761	16 진수 번호	0xD201
오류 코드 식별 번호	ERR_RTX_ERRORLOG_COUNT_ERROR		
오류 설명	Error Log Count 지정 오류입니다		
오류 해결	지정한 Count 가 제한치를 초과하지 않았는지 확인하시기 바랍니다		

10 진수 번호	57344	16 진수 번호	0xE000
오류 코드 식별 번호	ERR_CARD_NO_FOUND		
오류 설명	컴퓨터에서 어떠한 축 카드도 찾지 못했습니다		
오류 해결	축 카드가 정상적으로 설치되었는지, 드라이브 설치가 완료되었는지, 정상적으로 설치되었는지, 드라이브가 장치 관리자에 제대로 나타났는지 확인하시기 바랍니다		

10 진수 번호	57345	16 진수 번호	0xE001
오류 코드 식별 번호	ERR_CARD_NO_RESPONSE		
오류 설명	축 카드가 전송한 명령에 대해 아무런 응답이 없습니다		
오류 해결	API 조작 기간이 만료되었습니다. 일반적으로 소프트웨어 문제이니 제조업체에 문의하시기 바랍니다		

10 진수 번호	57346	16 진수 번호	0xE002
오류 코드 식별 번호	ERR_CARD_CONNECT_FAILED		
오류 설명	축 카드와 드라이브 프로그램과의 연결 오류입니다		
오류 해결	드라이브 프로그램 설치가 완료되었는지 확인하시기 바랍니다. 필요한 경우 컴퓨터를 다시 시작하시기 바랍니다		

10 진수 번호	57347	16 진수 번호	0xE003
오류 코드 식별 번호	ERR_CARD_MEMORY_NOT_ENOUGH		
오류 설명	Record 기능 사용 시, 수량이 축 카드 제한치(24 모듈)를 초과했습니다		
오류 해결	하드웨어 자체의 제한이므로 일정 log 가 24 모듈 데이터를 초과하면 RTX 버전을 구입하시기 바랍니다		

10 진수 번호	57348	16 진수 번호	0xE004
오류 코드 식별 번호	ERR_CARD_LOAD_AUTOCONFIG_FILE		
오류 설명	구성 파일 확인에 실패했습니다		
오류 해결	구성 파일이 제대로 저장되었는지 확인하고 Initial 을 재실행하시기 바랍니다		

10 진수 번호	57349	16 진수 번호	0xE005
오류 코드 식별 번호	ERR_CARD_SECURITY_FAILED		
오류 설명	보안 시스템 인증에 실패했습니다		
오류 해결	판매 업체에 문의하시기 바랍니다		

10 진수 번호	57350	16 진수 번호	0xE006
오류 코드 식별 번호	ERR_CARD_UPGRADE_CREATE_THREAD_FAILED		
오류 설명	업데이트 윈도우 생성에 실패했습니다		
오류 해결	Initial 을 재실행하고 업데이트하시기 바랍니다		

10 진수 번호	57351	16 진수 번호	0xE007
오류 코드 식별 번호	ERR_CARD_UPGRADE_NO_RESPONSE		
오류 설명	업데이트가 응답하지 않습니다		
오류 해결	Initial 을 다시 업데이트해주십시오. 문제가 계속 해결되지 않을 경우 제조업체에 연락 바랍니다		

10 진수 번호	57352	16 진수 번호	0xE008
오류 코드 식별 번호	ERR_CARD_UPGRADE_NO_RESOURSE		
오류 설명	업데이트 리소스 확인 실패		
오류 해결	DLL 버전과 작업 시스템을 확인해 주십시오		

10 진수 번호	57353	16 진수 번호	0xE009
오류 코드 식별 번호	ERR_CARD_UPGRADE_LOAD_RESOURSE		
오류 설명	업데이트 리소스 확인 실패		
오류 해결	DLL 버전과 작업 시스템을 확인해 주십시오		

10 진수 번호	57354	16 진수 번호	0xE00A
오류 코드 식별 번호	ERR_CARD_UPGRADE_TIMEOUT		
오류 설명	축 카드 펌웨어 업데이트 기간이 만료되었습니다		
오류 해결	Initial 을 다시 업데이트해주십시오. 문제가 계속 해결되지 않을 경우 제조업체에 연락 바랍니다		

10 진수 번호	57355	16 진수 번호	0xE00B
오류 코드 식별 번호	ERR_CARD_UPGRADE_FAILED		
오류 설명	축 카드 펌웨어 업데이트에 실패했습니다		
오류 해결	정상적으로 Flash Burn 을 실행할 수 없습니다. 제조 업체에 문의하시기 바랍니다		

10 진수 번호	61440	16 진수 번호	0xF000
오류 코드 식별 번호	ERR_ECAT_DLL_IS_USED		
오류 설명	EtherCAT_DLL 이 중복 실행되었습니다		
오류 해결	여러 개의 프로그램이 동시에 EtherCAT_DLL 을 사용하지 않는지 확인하시기 바랍니다		

10 진수 번호	61441	16 진수 번호	0xF001
오류 코드 식별 번호	ERR_ECAT_NO_DLL_FOUND		
오류 설명	EtherCAT_DLL 과 RTSS, 축 카드 DLL 의 연결이 실패했습니다		
오류 해결	RTSS 환경 또는 Windows 환경이 존재하는지 확인하시기 바랍니다		

10 진수 번호	61442	16 진수 번호	0xF002
오류 코드 식별 번호	ERR_ECAT_NO_RTSS_DLL_FOUND		
오류 설명	EtherCAT_DLL 과 RTSSDLL 의 연결이 실패했습니다		
오류 해결	RTSS 환경이 존재하는지 확인하시기 바랍니다		

10 진수 번호	61443	16 진수 번호	0xF003
오류 코드 식별 번호	ERR_ECAT_NO_CARD_DLL_FOUND		
오류 설명	EtherCAT_DLL 과 축 카드 DLL 의 연결이 실패했습니다		
오류 해결	Windows 환경이 존재하는지 확인하시기 바랍니다		

10 진수 번호	61444	16 진수 번호	0xF004
오류 코드 식별 번호	ERR_ECAT_NO_ESI_DLL_FOUND		
오류 설명	EtherCAT DLL 과 ESI 가 조작한 DLL 의 연결이 실패했습니다		
오류 해결	파일이 동일한 목록에 존재하는지 확인하시기 바랍니다		

10 진수 번호	61445	16 진수 번호	0xF005
오류 코드 식별 번호	ERR_ECAT_SAME_CARD_NUMBER		
오류 설명	RTSS 또는 축 카드 번호가 중복됩니다		
오류 해결	축 카드 번호가 중복되지 않는지 확인하시기 바랍니다		

10 진수 번호	61446	16 진수 번호	0xF006
오류 코드 식별 번호	ERR_ECAT_CARDNO_ERROR		
오류 설명	존재하지 않는 카드 번호를 사용했습니다		
오류 해결	카드 상의 DIP Switch 가 조정하는 번호를 확인하시기 바랍니다		

10 진수 번호	61447	16 진수 번호	0xF007
오류 코드 식별 번호	ERR_ECAT_GET_DLL_PATH		
오류 설명	DLL 경로를 확인할 수 없습니다		
오류 해결	DLL 설치 경로를 확인하시기 바랍니다		

10 진수 번호	61448	16 진수 번호	0xF008
오류 코드 식별 번호	ERR_ECAT_GET_DLL_VERSION		
오류 설명	DLL 버전을 확인할 수 없습니다		
오류 해결	DLL 버전을 확인하시기 바랍니다		

10 진수 번호	61449	16 진수 번호	0xF009
오류 코드 식별 번호	ERR_ECAT_NOT_SUPPORT		
오류 설명	EtherCAT 은 현재 해당 DMC 유형의 API 를 지원하지 않습니다		
오류 해결	EtherCAT 기능과 비슷한 API 를 사용할 수 있습니다. 상세한 지원 대조표는 매뉴얼을 참조하시기 바랍니다		

34

10 진수 번호	65535	16 진수 번호	0xFFFF
오류 코드 식별 번호	ERR_ECAT_LOADLIB_EMPTY		
오류 설명	RTSS 에서 DLL 리소스 불러오기 실패		
오류 해결	다음 조건을 확인하십시오: 1. PAC 전원을 재실행한 후 다시 한번 시도해주십시오. 2. 해당 문제가 해결되지 않을 경우 당사로 연락바랍니다.		

DMCNET 병합 함수 표 35

다음은 DMCNET 의 병합 함수 표에 대한 설명입니다.

35.1 DMCNET 함수 테이블

35

EtherCAT 이 지원하는 본사의 DMCNET 함수를 통해 사용자는 기존의 DMCNET 시스템에서 보다 신속히 EtherCAT 시스템으로 전환할 수 있습니다. 함수의 시작 부분이 기존의 _DMC_01_에서 _ECAT_01 로 변경되었으나 그밖에 다른 인수는 그대로 유지됩니다. 그러나 반환된 오류 코드는 기존대로 EtherCAT 코드를 참조하시기 바랍니다. 여기에서는 EtherCAT 이 지원하는 DMCNET 함수에 대해서만 설명하오니 자세한 인수 관련 내용 및 사용법은 DMCNET 프로그램 개발 매뉴얼을 참고하시기 바랍니다.

하드웨어 초기화 API	
_ECAT_01_open	응용 프로그램이 시작될 때 시스템 리소스를 초기화합니다
_ECAT_01_close	모든 시스템 리소스를 비움
_ECAT_01_get_CardNo_seq	호스트에 있는 모든 PCI-DMC-F01 인터페이스 카드 번호 확인
_ECAT_01_pci_initial	해당 PCI 인터페이스 카드 초기화

인터페이스 카드 조작 API	
_ECAT_01_initial_bus	축 카드 한 장의 외부 busbar(External Bus)를 초기화
_ECAT_01_initial_bus2	외부의 모든 busbar(External Bus)를 초기화
_ECAT_01_start_ring	ring 통신 시작
_ECAT_01_get_device_table	장치 리스트를 확인합니다
_ECAT_01_get_node_table	Node 번호 리스트를 확인합니다
_ECAT_01_check_card_running	해당 인터페이스 카드의 사용 여부 검사
_ECAT_01_reset_card	해당 인터페이스 카드 리셋
_ECAT_01_check_nodeno	해당 Node 번호가 이미 존재하는지 확인합니다
_ECAT_01_get_master_connect_status	Master Card 와 확장 모듈의 현재 연결 상태 확인
_ECAT_01_get_DLL_path	EtherCat_DLL.dll 의 상세 경로 데이터 확인
_ECAT_01_get_DLL_version	EtherCat_DLL.dll 버전 데이터를 확인합니다.
_ECAT_01_get_DLL_version_Single	EtherCat_DLL.dll 버전 데이터를 확인합니다.
_ECAT_01_get_cycle_time	현재 검색 중인 장치의 사이클 시간 확인

서보 드라이브의 파라미터가 API 를 확인/입력합니다	
_ECAT_01_read_servo_parameter	서보 드라이브의 파라미터 데이터를 확인합니다
_ECAT_01_write_servo_parameter	서보 드라이브의 파라미터 데이터를 입력합니다

SDO 프로토콜의 API 를 사용합니다	
_ECAT_01_send_message	SDO 명령 정보를 데이터 버퍼 영역으로 보냅니다
_ECAT_01_send_message2	SDO 명령 정보를 데이터 버퍼 영역으로 보내고, 데이터가 서보에 전달된 것이 확인되면 종료
_ECAT_01_send_message3	데이터 버퍼 영역에 SDO 명령 정보를 보내지만 명령을 전송하면 즉시 데이터 버퍼 영역을 종료
_ECAT_01_get_message	데이터 버퍼 영역에 전달된 SDO 명령 정보를 확인합니다

P to P 모션 컨트롤 사용 패킷 프로토콜 API	
_ECAT_01_set_sdo_driver_speed_profile	패킷 프로토콜의 속도 파라미터를 사용하여 설정
_ECAT_01_start_sdo_driver_a_move	절대성의 동작 변위 시작
_ECAT_01_start_sdo_driver_r_move	상대성의 동작 변위 시작

원점 복귀 모션 컨트롤 사용 패킷 프로토콜 API	
_ECAT_01_set_home_config	원점 복귀의 각 옵션을 설정합니다
_ECAT_01_set_home_move	원점 복귀 동작 실행 시작 설정
_ECAT_01_escape_home_move	원점 복귀 동작을 정지시킵니다

속도 모션 컨트롤 사용 패킷 프로토콜 API	
_ECAT_01_set_velocity_mode	속도 동작 제어 파라미터의 옵션
_ECAT_01_set_velocity	속도 동작 실행을 설정합니다
_ECAT_01_set_velocity_stop	속도 동작 제어를 중지합니다
_ECAT_01_set_velocity_torque_limit	velocity 모드에서 최대 torque 제한 설정

토크 모션 컨트롤 사용 패킷 프로토콜 API	
_ECAT_01_set_torque_mode	토크 모션 컨트롤 파라미터의 옵션
_ECAT_01_set_torque	토크 동작의 실행 설정
_ECAT_01_set_torque_stop	토크 동작 정지
_ECAT_01_set_torque_velocity_limit	torque 모드에서 최대 velocity 제한 설정

연속 속도 API	
_ECAT_01_speed_continue	연속 속도 기능의 활성화 여부
_ECAT_01_speed_continue_mode	연속 속도 모드
_ECAT_01_speed_continue_combine_ratio	연속 속도 합성 비율

35

PDO 프로토콜의 API 를 사용합니다	
_ECAT_01_ipo_set_svon	대응하는 장치에 Power On / Off 명령 전송 (Servo ON / OFF) — 모든 Slave 장치에 적용
_ECAT_01_get_buffer_length	세로 열에서 실행되지 않은 모션 명령 개수 확인

동작 제어 API 를 중지합니다	
_ECAT_01_emg_stop	Buffer 내부의 모든 모션 명령을 즉각 중단
_ECAT_01_sd_stop	Buffer 내부의 모든 모션 명령이 감속 시간에 따라 감속 정지
_ECAT_01_set_scurve_rate	가속, 감속 구간, S_Curve 모드의 백분율을 설정합니다.

동작 상태 API	
_ECAT_01_motion_done	Master Card 가 현재 실행 중인 동작 단계 반환
_ECAT_01_motion_status	Master Card 의 현재 동작 상태 반환

동작 계수값 API	
_ECAT_01_get_command	command 계수값 확인
_ECAT_01_set_command	새로운 command 계수값 설정
_ECAT_01_get_position	현재 position 계수값 확인
_ECAT_01_set_position	새로운 position 계수값 설정
_ECAT_01_get_target_pos	현재 위치의 position 값 확인
_ECAT_01_get_current_speed	현재 동작의 성분 속도를 확인합니다

소프트웨어 limit API	
_ECAT_01_set_soft_limit	소프트웨어 (+)/(-) limit 의 참조값 설정
_ECAT_01_enable_soft_limit	limit 의 실행 여부와 limit 의 도달 이후의 정지 방식 설정
_ECAT_01_disable_soft_limit	소프트웨어 limit off

단축 동작 제어 API	
_ECAT_01_start_tr_move	T-curve 속도 단면의 상대 좌표를 참조하여 동작의 변위 실행
_ECAT_01_start_sr_move	S-curve 속도 단면의 상대 좌표를 참조하여 동작의 변위 실행
_ECAT_01_start_ta_move	T-curve 속도 단면의 절대 좌표를 참조하여 동작의 변위 실행
_ECAT_01_start_sa_move	S-curve 속도 단면의 절대 좌표를 참조하여 동작의 변위 실행
_ECAT_01_p_change	새로운 위치값을 현재의 위치로 전환 설정

단축 동작 제어 API	
_ECAT_01_v_change	새로운 속도값을 현재의 동작 속도로 전환 설정
_ECAT_01_feedrate_overwrite	동작 벡터 속도 또는 벡터 속도 비율 변경
_ECAT_01_start_v3_move	EndVel 의 단축 동작 변위 증가

2 축 선형 보간 동작 제어 API	
_ECAT_01_start_tr_move_xy	T-curve 속도 단면의 상대 좌표를 참조하여 양축 선형 보간 동작 실행
_ECAT_01_start_sr_move_xy	S-curve 속도 단면의 상대 좌표를 참조하여 양축 선형 보간 동작 실행
_ECAT_01_start_ta_move_xy	T-curve 속도 단면의 절대 좌표를 참조하여 양축 선형 보간 동작 실행
_ECAT_01_start_sa_move_xy	S-curve 속도 단면의 절대 좌표를 참조하여 양축 선형 보간 동작 실행
_ECAT_01_start_v3_move_xy	EndVel 의 양축 선형 보간 동작 증가

2 축 원형 보간 모션 컨트롤 API	
_ECAT_01_start_tr_arc_xy	T-curve 속도 단면의 상대 좌표를 참조하여 양축 원형 보간 동작 실행 (조건: 중심 좌표, 각도)
_ECAT_01_start_sr_arc_xy	S-curve 속도 단면의 상대 좌표를 참조하여 양축 원형 보간 동작 실행 (조건: 중심 좌표, 각도)
_ECAT_01_start_ta_arc_xy	T-curve 속도 단면의 절대 좌표를 참조하여 양축 원형 보간 동작 실행 (조건: 중심 좌표, 각도)
_ECAT_01_start_sa_arc_xy	S-curve 속도 단면의 절대 좌표를 참조하여 양축 원형 보간 동작 실행 (조건: 중심 좌표, 각도)
_ECAT_01_start_tr_arc2_xy	T-curve 속도 단면의 상대 좌표를 참조하여 양축 원형 보간 동작 실행 (조건: 최종 좌표, 각도)
_ECAT_01_start_sr_arc2_xy	S-curve 속도 단면의 상대 좌표를 참조하여 양축 원형 보간 동작 실행 (조건: 최종 좌표, 각도)
_ECAT_01_start_ta_arc2_xy	T-curve 속도 단면의 절대 좌표를 참조하여 양축 원형 보간 동작 실행 (조건: 최종 좌표, 각도)
_ECAT_01_start_sa_arc2_xy	S-curve 속도 단면의 절대 좌표를 참조하여 양축 원형 보간 동작 실행 (조건: 최종 좌표, 각도)
_ECAT_01_start_tr_arc3_xy	T-curve 속도 단면의 상대 좌표를 참조하여 양축 원형 보간 동작 실행 (조건: 중심 좌표, 최종 좌표)
_ECAT_01_start_sr_arc3_xy	S-curve 속도 단면의 상대 좌표를 참조하여 양축 원형 보간 동작 실행 (조건: 중심 좌표, 최종 좌표)
_ECAT_01_start_ta_arc3_xy	T-curve 속도 단면의 절대 좌표를 참조하여 양축 원형 보간 동작 실행 (조건: 중심 좌표, 최종 좌표)

35

2 축 원형 보간 모션 컨트롤 API	
_ECAT_01_start_sa_arc3_xy	S-curve 속도 단면의 절대 좌표를 참조하여 양축 원형 보간 동작 실행 (조건: 중심 좌표, 최종 좌표)
_ECAT_01_start_spiral_xy	양축 나선 동작 (조건: X, Y 축의 중심 좌표)
_ECAT_01_start_spiral2_xy	양축 나선 동작 (조건: X, Y 축 중심 좌표, X, Y 축 최종 좌표)
_DMC_0F1_start_v3_arc_xy	EndVel 의 양축 원형 보간 동작을 증가시킵니다 (조건: 중심 좌표, 각도)
_ECAT_01_start_v3_arc2_xy	EndVel 의 양축 원형 보간 동작을 증가시킵니다 (조건: 최종 좌표, 각도)
_ECAT_01_start_v3_arc3_xy	EndVel 의 양축 원형 보간 동작을 증가시킵니다 (조건: 중심 좌표, 최종 좌표)
_ECAT_01_start_v3_spiral_xy	EndVel 의 양축 나선 동작 증가 (조건: X, Y 축의 중심 좌표)
_ECAT_01_start_v3_spiral2_xy	EndVel 의 양축 나선 동작 증가 (조건: X, Y 축 중심 좌표, X, Y 축 최종 좌표)

3 축 선형 보간 모션 컨트롤 API	
_ECAT_01_start_tr_move_xyz	T-curve 속도 단면의 상대 좌표를 참조하여 3 축 선형 보간 동작 실행
_ECAT_01_start_sr_move_xyz	S-curve 속도 단면의 상대 좌표를 참조하여 3 축 선형 보간 동작 실행
_ECAT_01_start_ta_move_xyz	T-curve 속도 단면의 절대 좌표를 참조하여 3 축 선형 보간 동작 실행
_ECAT_01_start_sa_move_xyz	S-curve 속도 단면의 절대 좌표를 참조하여 3 축 선형 보간 동작 실행
_ECAT_01_start_v3_move_xyz	EndVel 의 3 축 선형 보간 동작을 증가시킵니다

3 축 나선 보간 모션 컨트롤 API	
_ECAT_01_start_tr_heli_xy	T-curve 속도 단면의 상대 좌표를 참조하여 3 축 나선 보간 동작 실행
_ECAT_01_start_sr_heli_xy	S-curve 속도 단면의 상대 좌표를 참조하여 3 축 나선 보간 동작 실행
_ECAT_01_start_ta_heli_xy	T-curve 속도 단면의 절대 좌표를 참조하여 3 축 나선 보간 동작 실행
_ECAT_01_start_sa_heli_xy	S-curve 속도 단면의 절대 좌표를 참조하여 3 축 나선 보간 동작 실행
_ECAT_01_start_v3_heli_xy	EndVel 의 3 축 나선 보간 동작 증가
_ECAT_01_start_v3_sphere_xyz	세 점이 원을 형성합니다

속도 모션 컨트롤 API	
_ECAT_01_tv_move	T-curve 속도 단면의 속도 모션 컨트롤 참조
_ECAT_01_sv_move	S-curve 속도 단면의 속도 동작 제어를 참고합니다

원격 확장 모듈 제어 API	
_ECAT_01_get_rm_input_value	원격 확장 모듈 입력 포트의 bit 0 부터 bit 15 까지의 값을 확인합니다
_ECAT_01_set_rm_output_value	원격 확장 모듈 출력 포트의 bit 0 부터 bit 15 까지의 값을 설정합니다
_ECAT_01_get_rm_output_value	원격 확장 모듈의 출력 값을 확인합니다

4 모듈 펄스 인터페이스 전용 API	
_ECAT_01_set_rm_04pi_ipulser_mode	펄스 인터페이스 모듈의 입력 phase mode 를 설정합니다
_ECAT_01_set_rm_04pi_opulser_mode	펄스 인터페이스 모듈 출력 phase mode 설정
_ECAT_01_set_rm_04pi_svon_polarity	POWER ON(SVON)의 레벨을 설정합니다
_ECAT_01_04pi_set_poweron	POWER ON (Servo ON) 기능을 사용합니다
_ECAT_01_rm_04PI_get_buffer	실행되지 않은 동작 명령을 확인합니다

4 모듈 아날로그 출력 원격 확장 모듈 API	
_ECAT_01_rm_04da_set_output_value	DA 출력 수치를 설정합니다
_ECAT_01_rm_04da_get_output_value	DA 출력 수치를 확인합니다
_ECAT_01_rm_04da_set_output_range	DA 출력 범위를 설정합니다
_ECAT_01_rm_04da_set_output_enable	bit 출력의 On / Off 를 설정합니다
_ECAT_01_rm_04da_read_data	DA 출력 수치를 확인합니다

4 모듈 아날로그 입력 원격 확장 모듈 API	
_ECAT_01_set_04ad_input_range	AD 의 Input 범위를 설정합니다
_ECAT_01_get_04ad_input_range	AD 의 현재 Input 설정 범위를 확인합니다
_ECAT_01_set_04ad_conversion_time	AD 전환 속도를 설정합니다
_ECAT_01_get_04ad_conversion_time	현재 설정된 AD 전환 속도를 확인합니다
_ECAT_01_get_04ad_data	입력된 전압 수치를 확인합니다
_ECAT_01_set_04ad_average_mode	AD 평균 기능 모드를 설정합니다
_ECAT_01_get_04ad_average_mode	AD 평균 기능 모드를 확인합니다
_ECAT_01_set_04ad_input_enable	AD Channel 의 Input 활성화 여부를 설정합니다

35

Slave 데이터	
_ECAT_01_get_devicetype	Slave 장치 타입을 확인합니다
_ECAT_01_get_slave_version	Slave 펌웨어 버전 번호를 확인합니다

경보 메시지 API	
_ECAT_01_set_ralm	출력된 서보 드라이브 경보 메시지의 리셋
_ECAT_01_set_ralm_servo_all	축 카드 연결 서보 드라이브 경고 메시지 리셋

다축 동작 제어 API	
_ECAT_01_multi_axes_move	2 축 이상의 동작 제어를 설정합니다
_ECAT_01_liner_speed_master	다축 직선 동작의 동작 속도 설정
_ECAT_01_start_v3_multi_axes	EndVel 의 다축 (2 축 이상) 동작 제어를 증가시킵니다

Security API	
_misc_security	사용자 Key+SerialNo 암호화를 통해 인증번호를 생성합니다

limit 의 역방향 API	
_ECAT_01_rm_04pi_set_MEL_polarity	(-) limit 의 순방향 / 역방향 설정
_ECAT_01_rm_04pi_get_MEL_polarity	(-) limit 의 현재 상태 확인
_ECAT_01_rm_04pi_set_PEL_polarity	(+) limit 의 순방향/역방향 설정
_ECAT_01_rm_04pi_get_PEL_polarity	(+) limit 의 현재 상태 확인

기타 API	
_misc_app_get_circle_endpoint	원형 보간 동작에 필요한 최종 좌표 (X, Y) 확인
_misc_app_get_circle_center_point	원형 보간 동작에 필요한 중심 좌표 (X, Y) 확인
_misc_set_record_debuging	Debug 기록 기능의 실행 여부
_misc_open_record_debuging_file	출력 Debug 기록의 저장 위치 설정
_ECAT_01_get_gear	Gear 설정값 확인
_ECAT_01_set_gear	Gear 기능 설정 및 해당 파라미터 설정

이전 버전 유지 API 36

다음은 이전 버전 API 및 신규 버전 API의 대조표입니다.

36.1 이전 버전 API 일람표

36

다음은 명칭을 변경하였으나 계속 유지되고 있는 API 리스트입니다. 사용자는 API 명칭 변경으로 인한 불편 없이 본 리스트를 통해 새로운 명칭의 API 를 찾을 수 있습니다.

이전 버전 API 명칭	신규 버전 API 명칭
_ECAT_Slave_ESC5621_Set_Output_Mode	_ECAT_Slave_R1_EC5621_Set_Output _Mode
_ECAT_Slave_ESC5621_Set_Input_Mode	_ECAT_Slave_R1_EC5621_Set_Input_Mode
_ECAT_Slave_ESC5621_Set_ORG_Inverse	_ECAT_Slave_R1_EC5621_Set_ORG _Inverse
_ECAT_Slave_ESC5621_Set_QZ_Inverse	_ECAT_Slave_R1_EC5621_Set_QZ_Inverse
_ECAT_Slave_ESC5621_Set_MEL_Inverse	_ECAT_Slave_R1_EC5621_Set_MEL _Inverse
_ECAT_Slave_ESC5621_Set_PEL_Inverse	_ECAT_Slave_R1_EC5621_Set_PEL_Inverse
_ECAT_Slave_ESC5621_Set_Svon_Inverse	_ECAT_Slave_R1_EC5621_Set_Svon _Inverse
_ECAT_Slave_ESC5621_Set_Home_SpMode	_ECAT_Slave_R1_EC5621_Set_Home _SpMode
_ECAT_Slave_ESC8124_Set_Input _RangeMode	_ECAT_Slave_R1_EC8124_Set_Input _RangeMode
_ECAT_Slave_ESC8124_Set_Input _ConvstFreq_Mode	_ECAT_Slave_R1_EC8124_Set_Input _ConvstFreq_Mode
_ECAT_Slave_ESC8124_Set_Input_Enable	_ECAT_Slave_R1_EC8124_Set_Input _Enable
_ECAT_Slave_ESC8124_Get_Input _RangeMode	_ECAT_Slave_R1_EC8124_Get_Input _RangeMode
_ECAT_Slave_ESC8124_Set_Input _AverageMode	_ECAT_Slave_R1_EC8124_Set_Input _AverageMode
_ECAT_Slave_ESC9144_Set_Output _RangeMode	_ECAT_Slave_R1_EC9144_Set_Output _RangeMode
_ECAT_Slave_ESC9144_Set_Output_Enable	_ECAT_Slave_R1_EC9144_Set_Output _Enable
_ECAT_Slave_ESC70E2_Set_Output_Enable	_ECAT_Slave_R1_EC70E2_Set_Output _Enable
_ECAT_Slave_ESC5614_Set_MJ_Config	_ECAT_Slave_R1_EC5614_Set_MJ_Config
_ECAT_Slave_ESC5614_Set_MJ_Enable	_ECAT_Slave_R1_EC5614_Set_MJ_Enable
_ECAT_Slave_ESC5614_Get_IO_Status	_ECAT_Slave_R1_EC5614_Get_IO_Status
_ECAT_Slave_ESC5614_Get_MPG_Counter	_ECAT_Slave_R1_EC5614_Get_MPG _Counter

수정 이력

발행일	버전	장과 절을 업데이트합니다	업데이트 내용
April, 2016	V1.0 (제 1 판)		
August, 2018	V2.0 (버전 2)	CH03	그림 3.3.2.4 소프트웨어 스크린샷 변경 그림 3.4.2.4 소프트웨어의 프린트 스크린 변경 그림 3.5.2.4 소프트웨어의 프린트 스크린 변경
		CH04	9144 Slave API 테이블 추가 Slave Record Data API 테이블 Master 축 카드 전용 API 테이블 Master Compare API 테이블 DLL 관련 API 테이블 Master User Security API 테이블 PAC MRAM 전용 API 테이블 70E2 Slave API 테이블 70X2 Slave API 테이블 5614 Slave API 테이블
		CH27	Master Compare API 테이블 함수 명칭 수정 _ECAT_Compare_Set_Channel1_Position_Table_Level _ECAT_Compare_Get_Channel1_Position_Table_Count _ECAT_Compare_Set_Channel_Polarity _ECAT_Compare_Reuse_Channel1_Position_Table _ECAT_Compare_Reuse_Channel1_Position_Table_Level 해당 장의 절에서 수정된 명칭 적용

(이 페이지는 공란으로 비워둡니다)