P.A.I.X

NMF Series

LIBRARY MANUAL



목 차

1. 소개 ..... - 7 -
1.1. Network Multi Function ..... - 7 -
1.1.1. 통신방식(Ethernet, Serial) ..... - 8 -
1.1.2. TCP/IP 다중접속 ..... - 8 -
1.2. 알아두기 ..... - 9 -
1.2.1. IP 주소 ..... - 9 -
1.2.2. 보드 ID ..... - 11 -
1.2.3. 일괄 Pin번호 지정방식 ..... - 12 -
1.2.4. 보드 구성 정합성 체크 ..... - 13 -

2. LIBRARY 구성 ..... - 15 -
2.1. 개발언어 지원 ..... - 15 -
2.2. 함수구성 ..... - 16 -
2.3. 반환값 ..... - 17 -

3. 장치 연결 및 설정 ..... - 19 -
3.1. 연결 ..... - 19 -
3.1.1. 연결 ..... - 20 -
3.1.2. 연결 해제 ..... - 21 -
3.1.3. PingCheck ..... - 22 -
3.2. 환경설정 ..... - 25 -
3.2.1. 통신 Time Out 설정 ..... - 25 -
3.2.2. 통신 Time Out 설정 확인 ..... - 26 -
3.2.3. 통신방식 설정 및 확인 ..... - 27 -

4. DIO (DIGITAL INPUT OUTPUT) ..... - 28 -
4.1. 입/출력 상태확인 ..... - 29 -
4.1.1. 입력상태 확인 ..... - 29 -
4.1.2. 출력상태 확인 ..... - 30 -
4.2. 출력설정 ..... - 31 -
4.2.1. 전체 출력설정 ..... - 31 -
4.2.2. 단일 출력Pin 설정 ..... - 32 -
4.2.3. 단일 출력Pin 반전 ..... - 33 -
4.2.4. 다중 출력Pin 설정 ..... - 34 -
4.2.5. 다중 출력Pin 반전 ..... - 35 -
4.3. 출력유지 제한시간 설정 ..... - 36 -
4.3.1. 제한 설정 ..... - 36 -

4.3.2. 설정 확인 - 38 -
4.4. 보드지정 출력설정 - 39 -
4.4.1. 단일 출력Pin 설정 - 39 -
4.4.2. 다중 출력Pin 설정 - 40 -
4.4.3. 다중 보드 출력 설정 - 41 -
4.4.4. 단일 출력Pin 반전 - 42 -
4.4.5. 다중 출력Pin 반전 - 43 -
4.4.6. 다중 보드 출력 반전 - 44 -

5. AO (ANALOG OUTPUT) - 45 -
5.1. 알아두기 - 46 -
5.1.1. 출력범위 - 46 -
5.1.2. 출력 예시 - 46 -
5.2. 출력설정 - 48 -
5.2.1. 단일 출력Pin 설정 - 48 -
5.2.2. 다중 출력Pin 설정 - 49 -
5.3. 보드지정 출력설정 - 50 -
5.3.1. 단일 출력Pin 설정 - 50 -
5.3.2. 다중 출력Pin 설정 - 51 -
5.3.3. 다중 보드 출력 설정 - 52 -

6. AI (ANALOG INPUT) - 53 -
6.1. 알아두기 - 54 -
6.1.1. 입력방식 - 54 -
6.1.2. Sampling (샘플링) - 54 -
6.1.3. 메모리 관리 - 57 -
6.2. 범위 설정 - 59 -
6.3. 샘플링 설정 - 61 -
6.3.1. 단일 Pin 시작 - 61 -
6.3.2. 다중 Pin 시작 - 62 -
6.3.3. 단일 Pin 정지 - 63 -
6.3.4. 다중 Pin 정지 - 64 -
6.4. 보드지정 샘플링 설정 - 65 -
6.4.1. 단일 Pin 시작 - 65 -
6.4.2. 다중 Pin 시작 - 66 -
6.4.3. 다중 보드 시작 - 67 -
6.4.4. 단일 Pin 정지 - 68 -
6.4.5. 다중 Pin 정지 - 69 -
6.4.6. 다중 보드 정지 - 70 -
6.5. Sampling Rate - 71 -
6.5.1. 단일 Pin 확인 - 71 -

6.5.2. 다중 Pin 확인 - 73 -
6.5.3. 단일보드 Pin 확인 - 74 -
6.5.4. 단일보드 다중Pin 확인 - 75 -
6.5.5. 다중보드 다중Pin 확인 - 76 -
6.6. 외부 메모리 설정 - 77 -
6.6.1. 전체 설정 - 77 -
6.6.2. 보드지정 설정 - 78 -
6.6.3. 보드지정 설정 확인 - 79 -
6.7. 데이터 확인 - 80 -
6.7.1. 샘플링 데이터 - 80 -

7. EXTERNAL PIN - 83 -
7.1. Count - 84 -
7.1.1. 알아두기 - 85 -
7.1.2. 기능 시작 - 89 -
7.1.3. 기능 시작(확장) - 91 -
7.1.4. 기능 정지 - 94 -
7.1.5. 상태 확인 - 95 -
7.1.6. 측정 시간 확인 - 96 -
7.1.7. 입력필터 설정 - 97 -
7.1.8. 설정필터 확인 - 99 -
7.2. Trigger - 100 -

8. 특수기능 - 101 -
8.1. 시리얼 통신 - 101 -
8.1.1. 사용설정 - 102 -
8.1.2. 환경설정 - 103 -
8.1.3. 데이터 송신 - 104 -
8.1.4. 데이터 수신 - 105 -
8.2. 연결 지원 - 106 -
8.2.1. 포트 변경 - 106 -

9. 정보확인 - 107 -
9.1. 전체 상태 확인 - 107 -
9.2. 장치 구성 - 108 -
9.3. 장치 리스트 - 110 -

10. 구조체 - 111 -
10.1. 구성 정보 - 111 -
10.2. 전체 상태 정보 - 112 -

10.3. DIO 상태 정보 ..... - 113 -
10.4. AO 상태 정보 ..... - 114 -
10.5. AI 상태 정보 ..... - 115 -
10.6. AI Sampling 정보 ..... - 116 -
10.7. External Pin 정보 ..... - 117 -
10.8. Count 상태 정보 ..... - 118 -
10.9. Count 측정시간 정보 ..... - 119 -

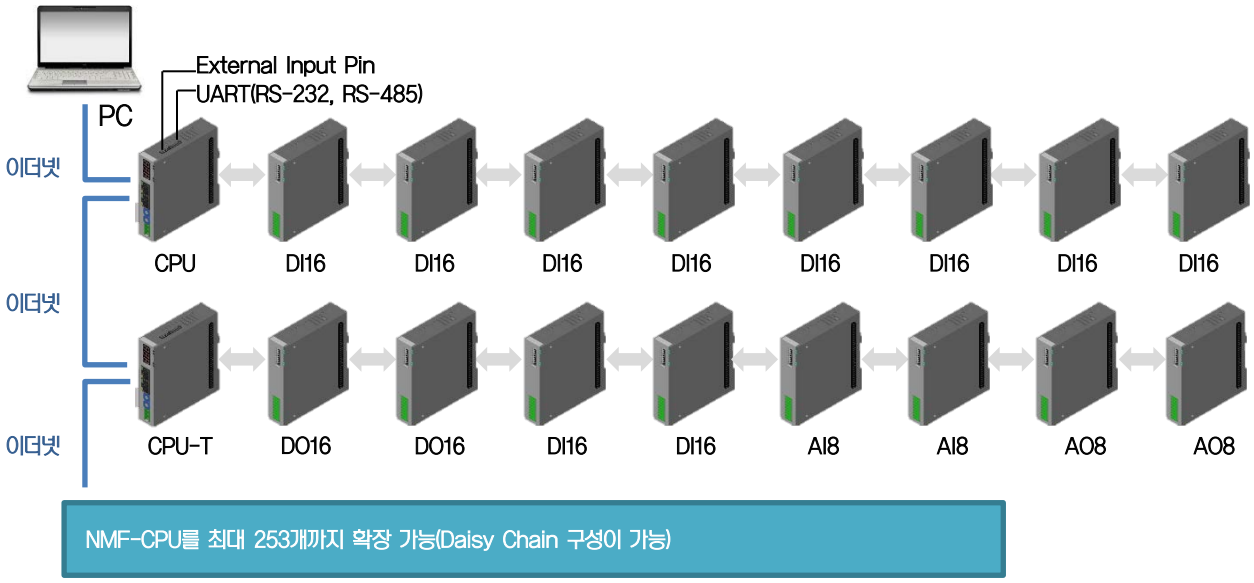
1. 소개

본 문서는 NMF(Nework Multi Function) 시리즈 제품의 구성과 장치 제어를 위한 Win32 API형태 C함수의 내용을 설명하고 있습니다.

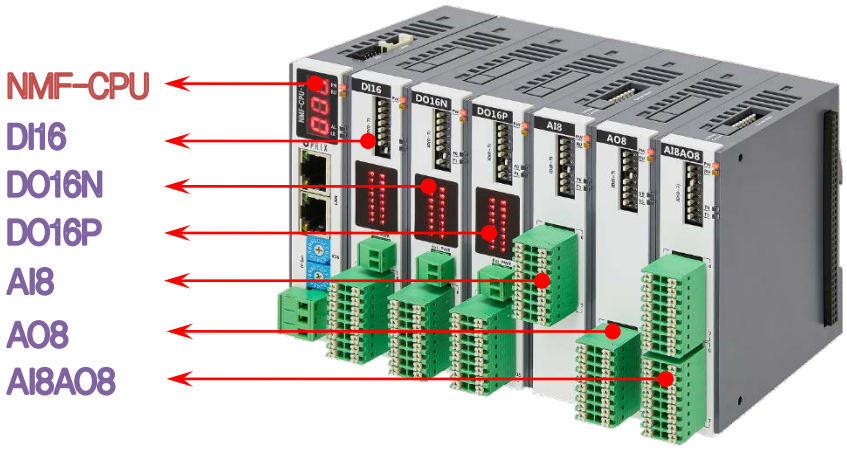
1.1. Network Multi Function

본 제품은 NMF (Network Multi Function) 라는 명칭을 사용합니다.

NMF-CPU 장치와 디지털/아날로그 입출력 장치를 **최대 8개까지 연결**하여 다양한 시스템을 구성할 수 있는 **혼합형 입출력 장치**입니다. Ethernet을 통해 상위 제어기(PC)와 통신을 수행하며 2 Port Ethernet HUB를 내장하고 있어 Daisy-Chain형태로 최대 253개까지 연결이 가능합니다.



<그림 1-1> 장치 연결 구성도



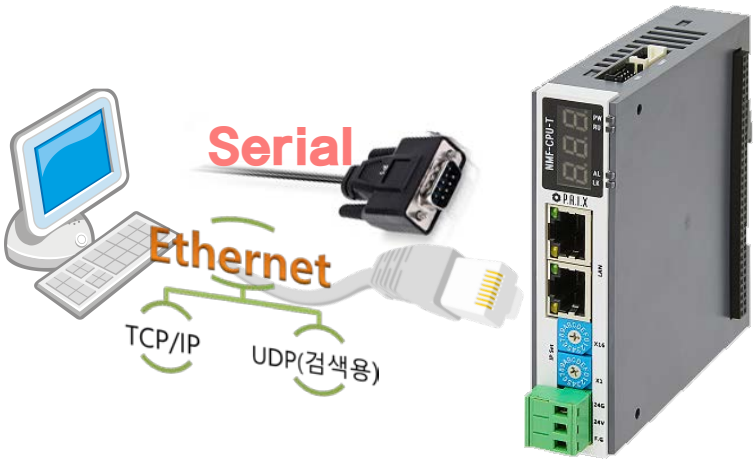
위와 같이 16점 입력장치(DI16)를 8대 연결하여 128점 입력으로 사용하거나, 디지털 및 아날로그 입출력장치를 혼합 구성하여 사용할 수 있습니다.

다양한 구성을 통해 여러 사용자의 요구조건에 부합되며 편의를 극대화 하였습니다.

1.1.1. 통신방식(Ethernet, Serial)

통신 방식은 Ethernet과 Serial 두가지 형태로 접속을 지원합니다.

Ethernet접속은 다시 TCP/IP와 UDP프로토콜의 2가지를 지원합니다.



<그림 1-2> 통신 방식

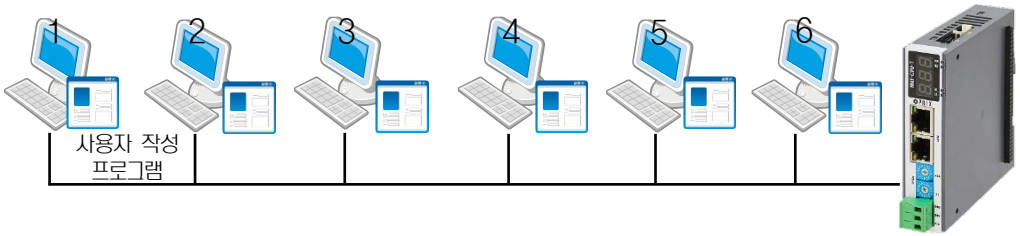
※ Serial 통신을 사용하실 경우에는 Protocol-Manual을 참고하여 주십시오.

1.1.2. TCP/IP 다중접속

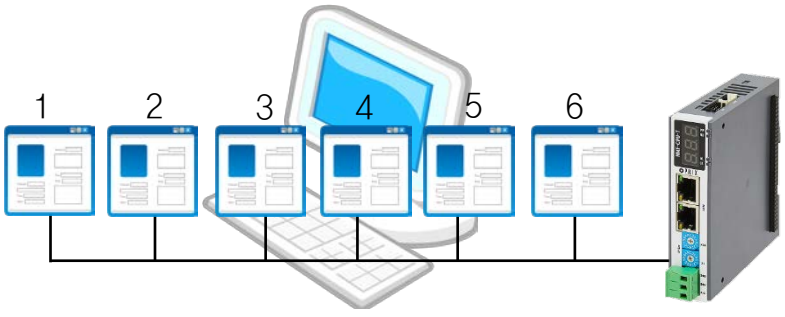
TCP/IP서버 형태의 Socket으로 운영되어,

최대 연결할 수 있는 Client(본 Library를 사용하여 작성한 사용자 프로그램)는 총 6개까지 입니다.

따라서, 1개의 장치에 여러대의 PC(각 PC한개별로 1개의 사용자 작성프로그램 운영)가 동시에 접속하는 경우 동시 접속이 가능한 PC는 6개까지 지원될 수 있습니다.



하나의 PC에서 여러개의 사용자 작성 프로그램이 동시에 장치에 접속하는 경우에도 총 6개의 프로그램이 접속할 수 있습니다.

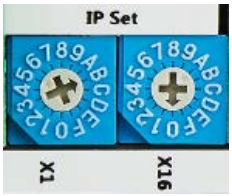


1.2. 알아두기

1.2.1. IP 주소

장치와 상위 제어기(PC)의 Ethernet 통신을 위해서는 IP 주소가 설정되어야 합니다.
전면부 로터리 스위치로 해당 장치의 번호를 선택할 수 있으며, IP주소의 마지막 번호가 됩니다.

1) 각 장치번호를 설정하는 로터리 스위치가 IP 설정 스위치 입니다.
2) 여러 장치를 Hub에 연결하여 사용하실 경우, 서로 중첩되지 않도록 IP를 설정하여 주십시오.
3) 로타리 스위치는 16진수 조합으로 이루어 집니다.
4) IP계산방식은 IP✖16 + IP✖1 입니다.
5) IP Address는 FND창에 Display 됩니다.(마지막 표시는 Firmware 버전입니다.)
6) 출하시 설정값 : 1.92 / 16.8 / 000./ 011 /0.0.1(192. 168. 0. 011)(버전 0.0.1 의 경우)



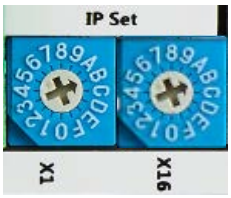
IP SET 스위치 계산방법
0X16 + B(11) = 11

<그림 1-3 공장 출하 설정:192.168.0.11>



<그림 1-4 FND표시 순서>

7) IP변경 예시 : 1.92 / 16.8 / 000./ 187 /0.0.1(192. 168. 0. 011) 버전 0.0.1



IP SET 스위치 계산방법
B(11) X16 + B(11) = 187

<그림 1-5 IP변경: 192.168.0.187>



<그림 1-6 FND표시 순서>

장치 IP 255번(0xF, 0xF)은 장치의 Bootloader 상태 진입을 위한 고정 IP이며, 브로드캐스트 사용을 위해 사용자가 지정할 수 없는 번호 입니다.

Memo

☑ TCP (Transmition Control Protocol)

인터넷 상의 컴퓨터들 사이에서 데이터를 메시지의 형태로 보내기 위해 IP와 함께 사용되는 프로토콜입니다. IP가 실제 데이터의 전송을 관장하는 동안, TCP는 데이터의 패킷을 추적 관리합니다. TCP는 연결지향 프로토콜이라고 알려져 있는데, 이것은 메시지들이 각단의 응용 프로그램들에 의해 교환되는 시간동안 연결이 확립되고 유지되는 것을 의미합니다. TCP는 IP가 처리할 수 있도록 메시지를 여러 개의 패킷들로 확실히 나누고, 반대편에서는 완전한 메시지로 패킷들을 재조립할 책임이 가집니다.

☑ UDP (User Datagram Protocol)

UDP는 IP를 사용하는 네트워크 내에서 컴퓨터들 간에 메시지들이 교환될 때 제한된 서비스만을 제공하는 통신 프로토콜입니다.

TCP와 마찬가지로 UDP도 한 컴퓨터에서 다른 컴퓨터로 데이터그램이라고 불리는 실제 데이터 단위를 받기 위해 IP를 사용합니다.

그러나, UDP는 TCP와는 달리 메시지를 패킷(데이터그램)으로 나누고, 반대편에서 재조립하는 등의 서비스는 제공하지 않으며, 특히 도착하는 데이터 패킷들의 순서를 제공하지 않습니다. 이 말은 UDP를 사용하는 응용프로그램은, 전체 메시지가 올바른 순서로 도착했는지에 대해 확인할 수 있어야 한다는 것을 의미한다. 교환해야할 데이터가 매우 적은(그러므로 재조립해야할 메시지도 매우 적은) 네트워크 응용프로그램들은 처리시간 단축을 위해 TCP 보다 UDP를 더 좋아할 수 있습니다. 일례로, 비디오나 오디오 신호전송에 UDP를 사용하게 됩니다.

UDP는 IP 계층에서 제공되지 않는 두 개의 서비스를 제공하는데, 하나는 다른 사용자 요청을 구분하기 위한 포트 번호와, 도착한 데이터의 손상여부를 확인하기 위한 CheckSum 기능입니다.

☑ IP주소

IP는 인터넷상의 한 컴퓨터에서 다른 컴퓨터로 데이터를 보내는데 사용되는 프로토콜입니다.

인터넷상의 각 컴퓨터들은 다른 컴퓨터와 구별될 수 있도록 적어도 한 개 이상의 고유한 주소를 갖는데, 연결된 기기를 식별하는 번호로 32bit로 구성되며, 4개의 10진수 형태로 표시됩니다. IPv4(Internet Protocol Version 4)규격을 사용합니다.

192.168.100.11

붉은색으로 표시된 부분을 동일 네트워크 구성을 확인하기 위한 주소로 사용하며, 파란색으로 표시된 주소가 호스트 주소로써, 장치에서는 전면부 로터리 스위치를 통해 변경 및 설정이 가능합니다.

IP주소의 호스트 주소 8bit가 모두 1이거나 0인 경우는 주소가 유효하지 않습니다.

즉, 0 주소는 호스트로 지정되지 않고 네트워크 지정에 사용되므로 사용할 수 없으며,

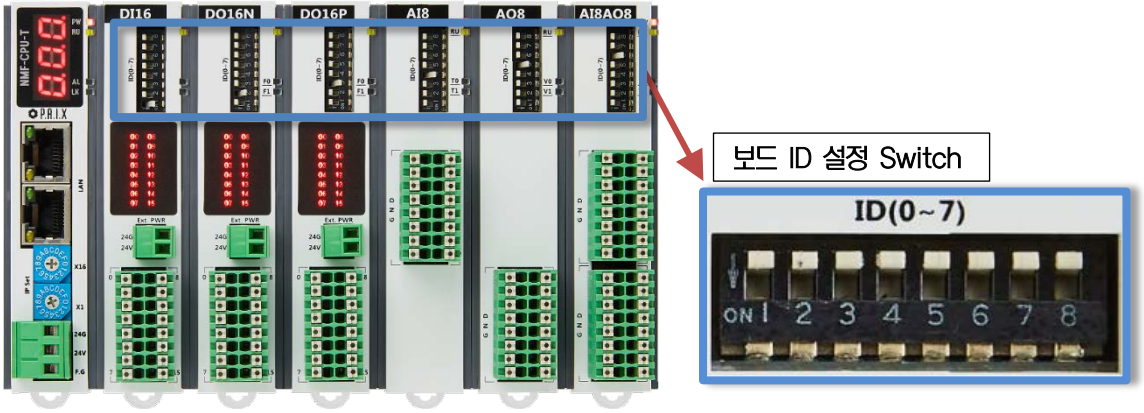
255 주소는 네트워크 상의 모든 호스트에 메시지를 브로드캐스트하는데 사용되므로 사용할 수 없습니다. 1번은 일반적으로 해당 주소의 네트워크의 이름으로 사용됩니다.

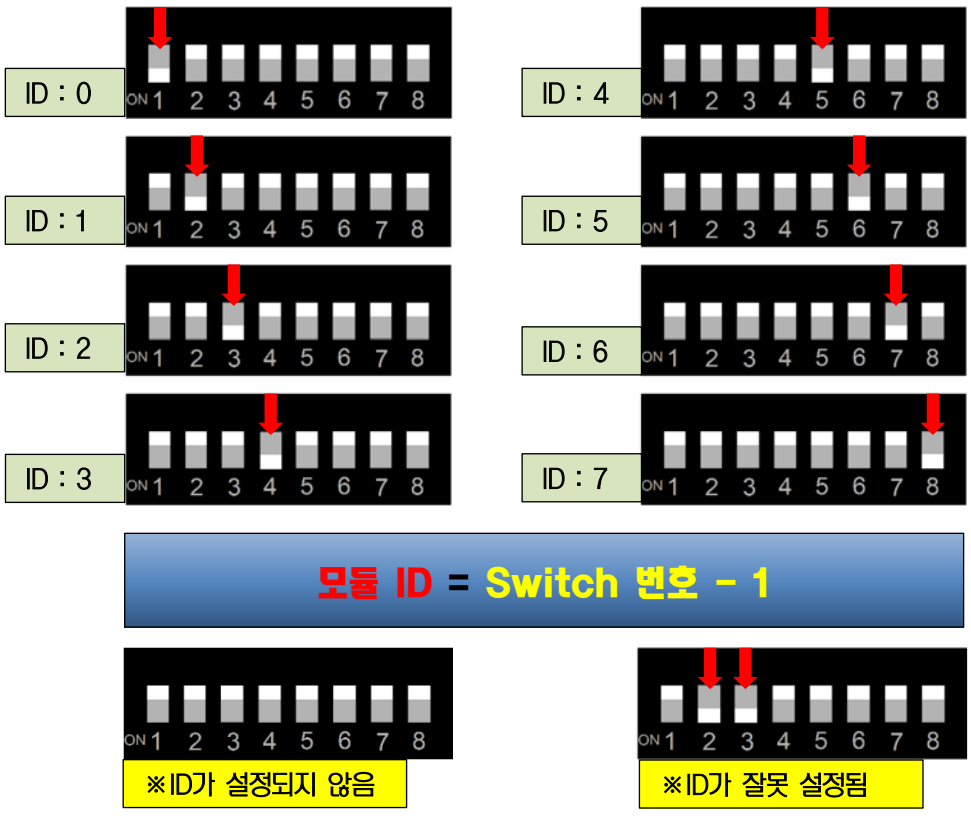
1.2.2. 보드 ID

NMF CPU에 증착되어 사용되는 DI, DO, AI, AO 장치를 하나의 **보드(Module)**로 지칭합니다.

장치 전면부는 아래와 같이 구성되며, 제어를 위해 각 보드의 ID를 설정해야 합니다.

보드 ID는 중복될 수 없으며, 0에서 7까지 하나의 값을 갖도록 합니다.





위와같이 스위치 번호 1번을 ON으로 설정하게되면 해당 보드의 ID는 0번이 됩니다.

보드 ID를 변경하는 경우, 반드시 장치는 전원이 RESET 되어야 합니다.

한 개의 보드에서 스위치를 2개 이상 ON으로 설정하거나, 모두 OFF 상태로 적용되면, 오작동의 원인이 될 수 있습니다. **ID 구성에 주의**하여 주십시오.

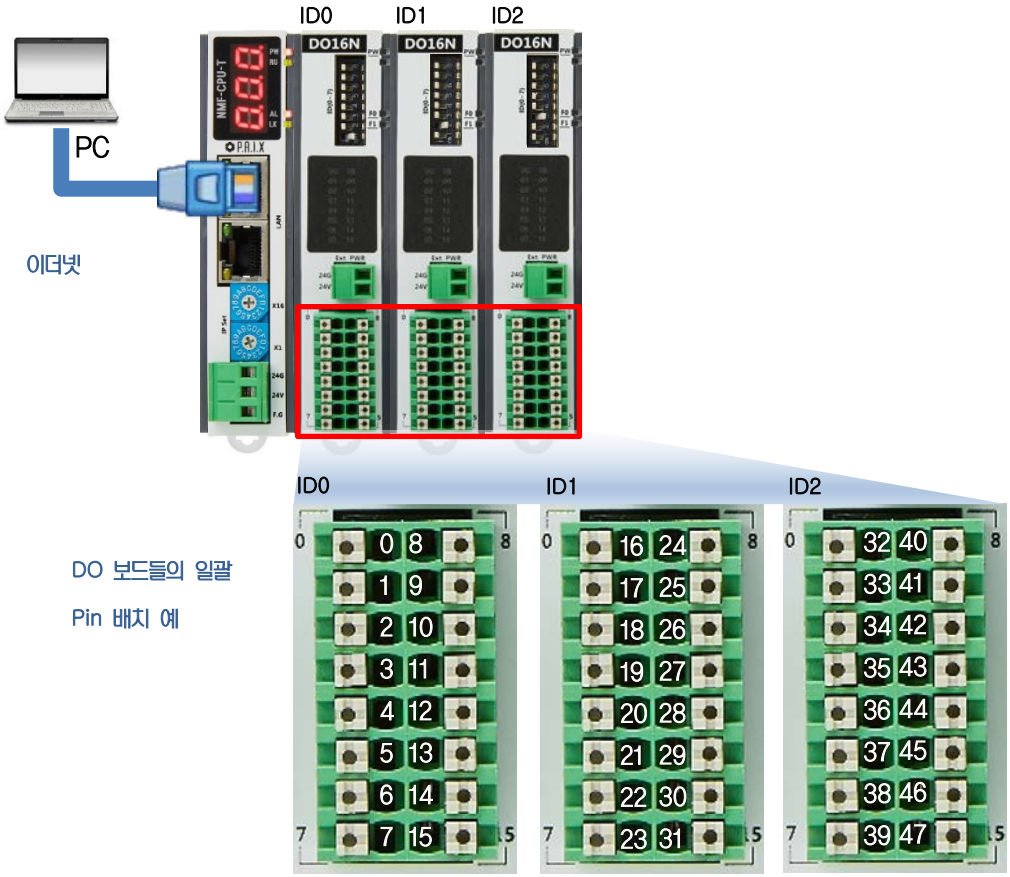
보드ID 구성과 CPU보드와의 연결순서는 제어에 영향을 주지 않습니다.

1.2.3. 일괄 Pin번호 지정방식

일반적으로 기능 동작을 위해서 제어하고자 하는 보드(DO, DI, AI, AO 등)의 ID를 지정하고 보드 내의 Pin을 지정합니다. 예를 들어 DO보드 ID 1번에 5번 Pin을 ON하는 기능을 수행하고자 한다면 [DO:1번 보드: 5번 Pin]형태로 지정합니다. (보드 지정 방식)

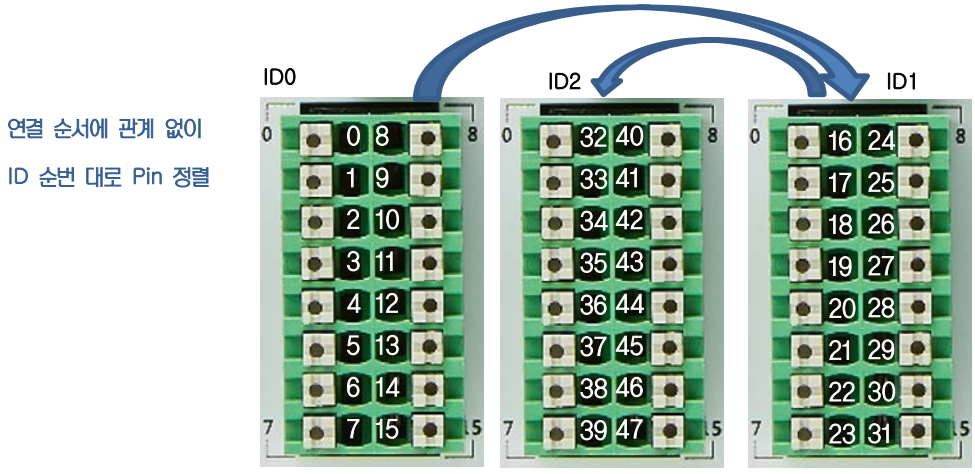
또한, 기존 자사 DIO 제품들과 호환하여 사용 할 수 있도록 Pin번호로 지정하여 제어 할 수 있습니다.(일괄 Pin번호 지정방식)

아래 그림은 DO 보드들을 일괄 Pin번호 지정 방식으로 제어 시 Pin 배치의 예시 입니다.



이와 같이, 동일한 형태를 가진 DO, DI, AO, AI 보드 각각의 Pin들을 정렬 할 수 있으며, 물리적인 연결순서와 관계없이 장치에 설정된 ID번호 순서로 적용됩니다.

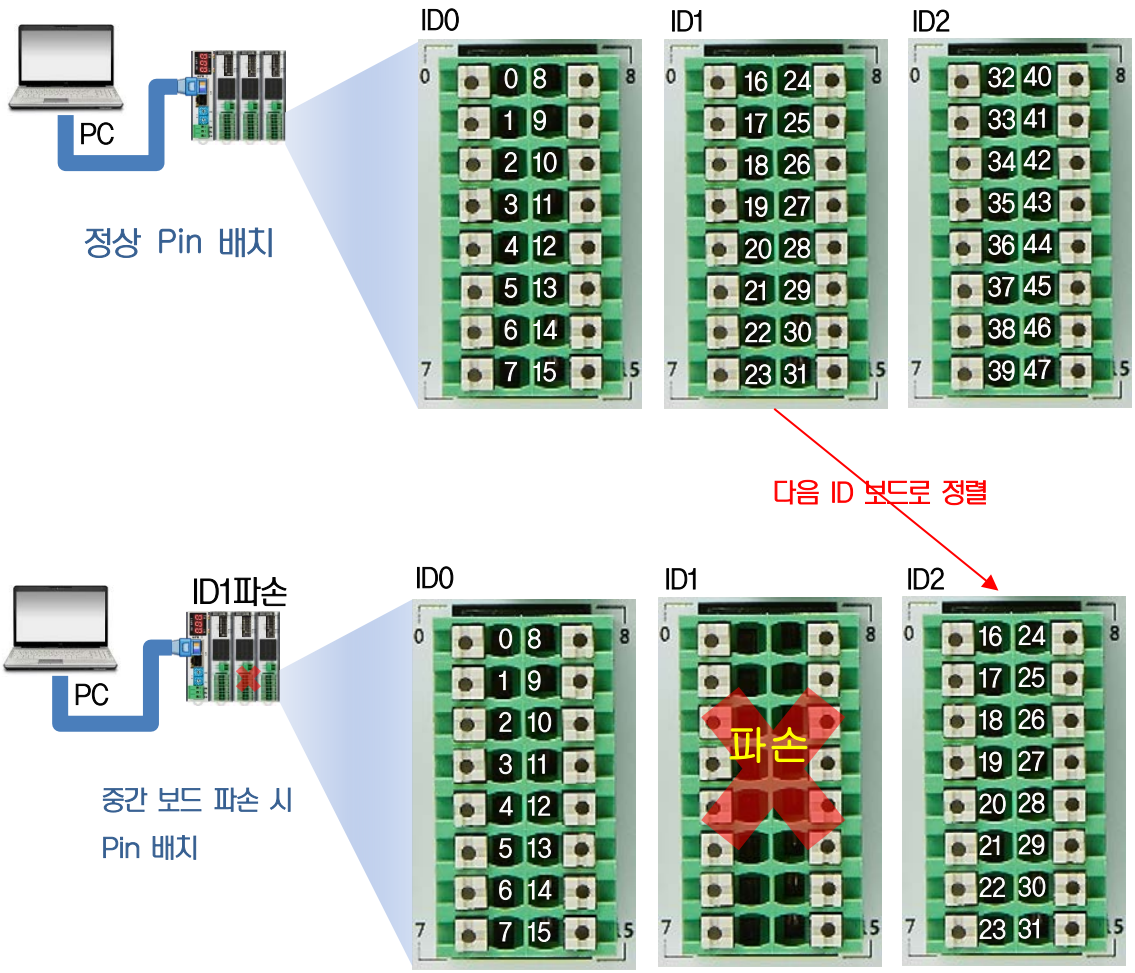
DO, DI보드는 최대 128Pin, AO, AI보드는 최대 64Pin 까지 지정할 수 있습니다.



1.2.4. 보드 구성 정합성 체크

일괄 Pin번호 지정 방식은 ID의 순번대로 Pin을 정렬하는데, 보드들 중 중간 DO보드가 파손된다면, 다음 예와 같은 문제가 발생합니다.(다른 타입의 보드도 같은 문제 발생)

예) DO 1번 보드가 파손되고, Pin 기준 방식으로 출력을 할 경우, DO 2번의 Pin으로 출력이 나가는 오동작이 발생 할 수 있음.



이 경우 보드 구성 정합성 체크를 하여 부정합 알람이 발생합니다. 안전상 모든 입출력 명령을 적용할 수 없으며, 반환값은 -106(보드 구성 정보가 부정합)이 출력됩니다.

보드 구성 정합성 체크는 기동 시 보드들의 Reference 구성이 현재 구성과 맞는가를 확인하는 것으로, 보드의 ID오류 등으로 인한 사고 발생을 사전에 차단하기 위한 확인 작업입니다.

Reference 구성은 NMF에 저장되어 있는 보드 구성 정보로, 출고 시 사양 주문에 따라 저장이 되며, 추 후 변경도 가능합니다. 만약 사용 중 보드를 추가 할 경우 부정합 알람이 발생을 하는데, 다음 예와 같은 조작으로 Reference 구성을 재 설정하면, 바뀐 Reference 구성으로 정합성을 체크하게 됩니다.

예) 출고 시 DI[0], DI[1], DO[2], DO[3]인 Reference 구성을 변경 할 때 입니다. ([N]에서 N=ID)

출고 시 구성은 정합하므로 Alarm 없이 작동을 합니다.

여기에 DI를 추가하여, DI[0], DI[1], DO[2], DO[3], DI[4]로 변경 할 경우 Reference 구성 DI[0], DI[1], DO[2], DO[3]과 변경된 구성이 달라 부정합 Alarm이 발생합니다.

IP Switch를 0으로 설정 뒤 전원을 Reset하면, 바뀐 DI[0], DI[1], DO[2], DO[3], DI[4] 구성이 저장됩니다.

이렇게 바뀐 구성이 Reference 구성으로 설정되어, 정합 판정을 하게 됩니다.

출고 시 구성

DI[0], DI[1], DO[2], DO[3]



정합

Reference구성

ID0	ID1	ID2	ID3
DI	DI	DO	DO

DI 보드 추가

DI[0], DI[1], DO[2], DO[3],



DI[4]



부정합

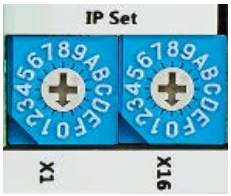
Reference구성

ID0	ID1	ID2	ID3
DI	DI	DO	DO

부정합 알람 발생



IP 스위치를 0으로 설정 후 전원 Reset



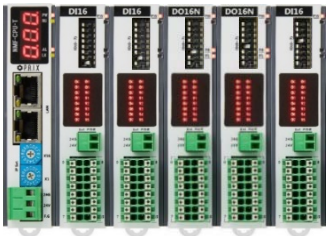
저장

현재 보드 구성 -> Reference 구성

ID0	ID1	ID2	ID3	ID4
DI	DI	DO	DO	DI

DI 보드가 추가 된 구성

DI[0], DI[1], DO[2], DO[3], DI[4]



정합

Reference구성

ID0	ID1	ID2	ID3	ID4
DI	DI	DO	DO	DI

2.Library 구성

2.1. 개발언어 지원

장치 제어를 위한 각 언어별 파일을 지원합니다.

사용하시는 프로그래밍 툴 및 언어에 따라 참고하시기 바랍니다.

NMF.dll (x32, x64)

- 공용 Dynamic Linking Library

NMF.h

- C, C++

NMF.lib

- Visual C++ Library

NMFb.lib

- C++ Builder Library

NMF.bas

- Visual Basic 6.0

NMF.vb

- Visual Basic .net

NMF.cs

- C#

NMF.pas

- Delphi

2.2. 함수구성

NMF 는 메인보드(CPU Board)에 사용자의 필요형태에 따라 DI, DO, AI, AO 장치를 최대 8개까지 결합하여 사용하는 장치로써, 그 용도에 따라 사용되는 함수가 달라집니다.

보드 종류에 따른 구분

nmf_DI□□□	Digital Input
nmf_DO□□□	Digital Output
nmf_AI□□□	Analog Input
nmf_AO□□□	Analog Output

보드 개수에 따른 구분

nmf_□□Brd□□□	단일 보드
nmf_□□mBrd□□□	다중 보드

위의 "nmf_"라는 접두사는 다른 API 함수와 충돌을 방지하기 위하여 본 라이브러리의 모든 함수에 적용됩니다. nmf 접두사 뒤에 따라오는 보드의 명칭으로 보다 쉽게 제어에 적용하실 수 있습니다.

다중 제어 함수는 같은 보드에만 반영됩니다. 즉, AO, AI 보드가 동일한 다중 함수로 동작될 수 없습니다. 각 함수의 접두사 및 장착된 보드의 형태를 파악하여 장치 제어에 적용하여 주십시오.

그 외 일반적인 연결 및 설정 함수들은 보드 구성과는 별개로 명칭을 따르지 않습니다.

2.3. 반환값

함수의 실행결과로 반환되어지는 값으로써, 설명은 아래의 표를 참고하여 주십시오.

반환값	정의	설명
0	nmf_R_OK	정상
-1	nmf_R_NOT_CONNECT	장치와 연결이 끊어진 경우
-2	nmf_R_SOCKET_ERR	소켓 에러
-3	nmf_R_UNKOWN	지원되지 않는 함수 호출
-4	nmf_R_INVALID	함수 인자값에 오류발생
-5	nmf_R_SYNTAX_ERR	함수 호출 시, 구문 오류 발생
-6	nmf_R_CONN_ERR	연결되어 있지 않음
-7	nmf_R_INVALID_NMF	유효하지 않은 NMF장치번호
-8	nmf_R_FUNC_NOT_ENOUGH	함수별 지정된 길이만큼 전송/수신하여야 하나 부족한 경우 (예:수행 결과로 수신한 응답 데이터가 원하는 길이보다 부족한 경우)
-9	nmf_R_NOT_RESPONSE	함수 호출 시, 응답이 없는 경우
-10	nmf_R_CMDNO_ERR	함수 호출 시, 식별코드에 오류 발생
-11	nmf_R_NOT_EXISTS	네트워크 장치가 식별되지 않는 경우, 방화벽이나 연결 상태를 확인
-12	nmf_R_ICMP_LOAD_FAIL	ICMP.DLL 로드 실패, nmf_PingCheck 호출시 발생. PC에 DLL유무를 확인
-13	nmf_R_FILE_LOAD_FAIL	F/W file 로드 실패
-14	nmf_R_SYNCHRONIZE	동기화 오류(뮤텍스,세마포어등에서 대기오류)
-15	nmf_R_SOCKET_TX_TMO	소켓 데이터 전송중 Time-out
-16	nmf_R_SOCKET_RX_TMO	소켓 데이터 수신중 Time-out
-17	nmf_R_BUF_OVERFLOW	지정 Buffer크기를 초과(ex.수신버퍼 초과)
-18	nmf_R_MCU_FAIL	MCU동작상의 실패
-19	nmf_R_FUNC_FAIL	함수수행에 실패한 경우
-20	nmf_R_HW_PART_NONE	수행하려고 하는 기능의 H/W장치나 구성이 없는경우 예)CPU모델에서 Count 기능 수행
-100	nmf_R_BRD_INVALID	함수에서 지정한 보드가 없거나 범위를 초과한 경우 예)DO보드가 2개인데 3번 보드에 1Pin에 ON명령 하달시
-101	nmf_R_BRD_MISMATCH_DO	함수에서 지정한 보드가 없거나 DO가 아닌 경우 예)DI, AO, AI 보드에 DO명령이 전달됨
-102	nmf_R_BRD_MISMATCH_DI	함수에서 지정한 보드가 없거나 DI가 아닌 경우 예)AO, AI, DO 보드에 DI명령이 전달됨
-103	nmf_R_BRD_MISMATCH_AI	함수에서 지정한 보드가 없거나 AI가 아닌 경우 예)DO, DI, AO 보드에 AI명령이 전달됨
-104	nmf_R_BRD_MISMATCH_AO	함수에서 지정한 보드가 없거나 AO가 아닌 경우

		예)DO, DI, AI 보드에 AO명령이 전달됨
-105	nmf_R_BRD_INVALID_PIN	해당 보드의 Pin 범위를 초과
-106	nmf_R_BRD_MISMATCH_COMPO	보드 구성 정보가 부정합 할 경우
-200	nmf_R_AI_SRATE_ERR	AI 샘플링 적용 시, 샘플링 속도 설정 범위 초과 (0~ 100000Hz)
-201	nmf_R_AI_INVALID_RANGE	AI 입력 전압 Range 설정 범위 초과 (0 ~ 1)
-202	nmf_R_AI_MEM_BLOCK_RANGE_ERR	AI 외부 메모리 설정 시 최대 Block 초과 외부 메모리 참고
-300	nmf_R_AO_RANGE_OVER	AO에서 출력 전압 Range를 넘어선 경우 예) ±5V Range에 6V가 설정 될 경우)
-1001	nmf_R_ARG_RNG_OVER	지정한 함수의 인자가 설정 범위를 초과한 경우 인자의 순서대로 에러코드가 증가함. 예) 3번째 인자 오류 시 (-1003)
-1501	nmf_R_ARG_ERR	지정한 함수의 인자 오류가 있는 경우 인자의 순서대로 에러코드가 증가함. 예) 3번째 인자 오류 시 (-1503)
-20001	nmf_R_FW_ARG_RNG_OVER	지정한 함수를 수행하기위해 전송되는 프로토콜에 인자가 설정 범위를 초과한 경우, 인자의 순서대로 에러코드가 증가함. 예) 3번째 인자 오류 시 (-20003)
-20501	nmf_R_FW_ARG_ERR	지정한 함수를 수행하기위해 전송되는 프로토콜에 인자 오류가 있는 경우, 인자의 순서대로 에러코드가 증가함. 예) 3번째 인자 오류 시 (-20503)

3.장치 연결 및 설정

장치를 상위 제어기(PC)에 연결하기 위한 네트워크 환경설정 등을 수행하는 함수입니다.

3.1. 연결

장치를 제어하기 전에 먼저 상위제어기(PC)와 통신 연결을 수행하여야 합니다.

관련함수

No	명령	설명
1	nmf_Connect	장치와 연결합니다.
2	nmf_Disconnect	장치와의 연결을 해제 합니다.
3	nmf_PingCheck	장치와의 네트워크 연결상태를 확인합니다.

3.1.1. 연결

nmf_Connect	Public

장치와 상위제어기(PC) 간의 연결을 수행합니다.

구문

short WINAPI **nmf_Connect** (short nNmfNo, short nIpAddr0, short nIpAddr1, short nIpAddr2);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호 (주소 4번째 자리이며, 전면 로터리 스위치 값)
2	nIpAddr0	IN	장치 IP주소 첫번째 (ex, 192.xxx.xxx.xxx)
3	nIpAddr1	IN	장치 IP주소 두번째 (ex, xxx.168.xxx.xxx)
4	nIpAddr2	IN	장치 IP주소 세번째 (ex, xxx.xxx.0.xxx)

주의사항

No	일람
1	장치 연결이 성공한 이후, 연결을 해제하기 전까지는 중복하여 수행할 필요가 없습니다.

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호

nRet = nmf_PingCheck(nNmfNo, 192, 168, 0, 200);   // 장치와의 네트워크 연결 상태를 확인합니다.
if ( nRet != nmf_R_OK )  return;      // 네트워크 연결 상태를 확인할 수 없습니다.

// 장치와 연결을 시도합니다.
nRet = nmf_Connect(nNmfNo, 192, 168, 0);
if ( nRet != nmf_R_OK )  return;      // 장치 연결 실패

/* 장치 운용 */

// 장치 연결을 해제합니다.
nmf_Disconnect(nNmfNo);
```

관련 참조

nmf_Connect, nmf_Disconnect, nmf_PingCheck

3.1.2. 연결 해제

nmf_Disconnect	Public

장치와 상위제어기(PC) 간의 연결을 해제합니다.

구문

void WINAPI **nmf_Disconnect** (short nNmfNo);

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호

사용 예

nmf_Connect 항목을 참고하여 주십시오.

관련 참조

nmf_Connect, nmf_Disconnect, nmf_PingCheck

3.1.3. PingCheck

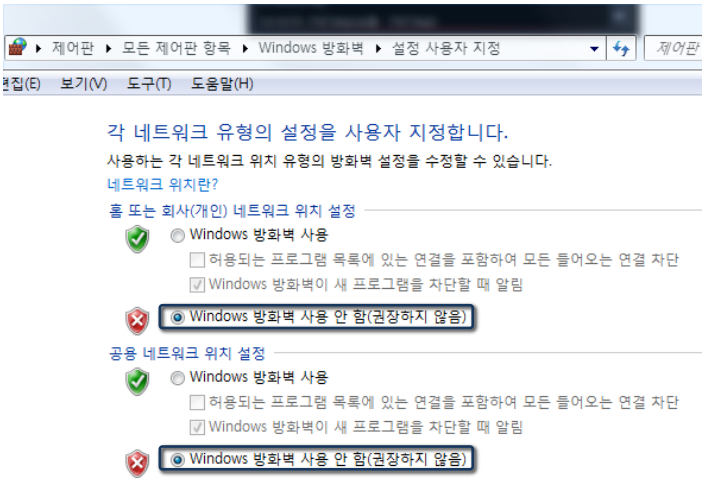
nmf_PingCheck	Public

장치와의 네트워크 연결을 확인하기 위한 Ping(Packet InterNet Groper) Check를 수행합니다.

(1) 보안 설정 해제

※ Ping 테스트를 진행하기 위해서는 Windows 보안 설정을 해제하여야 합니다. 그 방법은

가. Windows 방화벽 해제

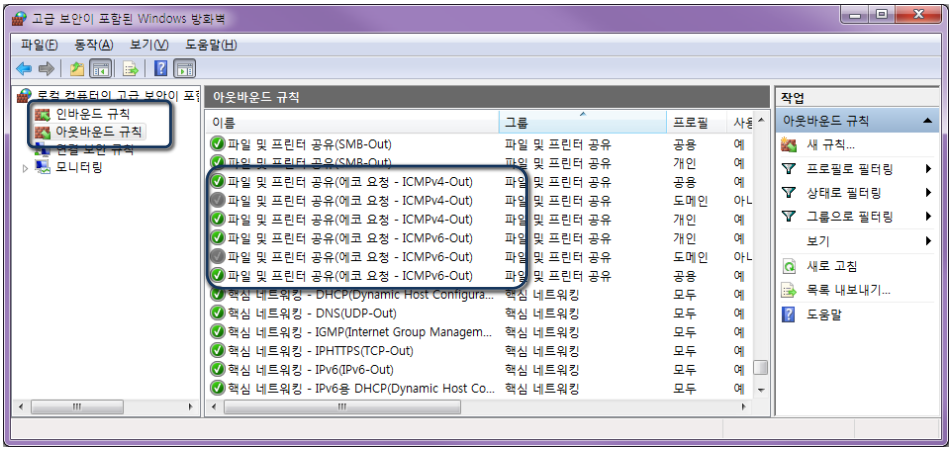


나. Ping관련된 부분(ICMP)만 보안 설정을 해제

1. Windows 2008 Server / 7 / 10

[제어판] - [네트워크 상태 및 작업] - [Windows 방화벽] - [고급 설정] - [아웃바운드 규칙]

"파일 및 프린터 공유(...)" 항목의 ICMP로 시작하는 항목들 수정



더블 클릭하여, "속성 창"의 "일반 탭"에서 **사용함**을 선택하시거나,

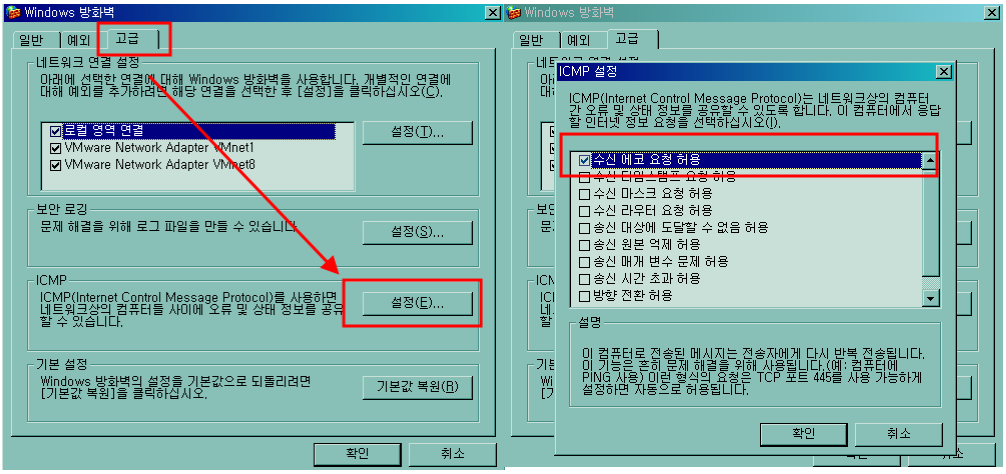
우클릭하여 생성되는 팝업창의 내용을 **규칙 사용**으로 선택하여 주십시오.

참고 : 인바운드 규칙은 받는 패킷을 확인하며, 아웃바운드 규칙은 보내는 패킷을 확인합니다.

2. Windows XP

[내 네트워크 환경] – [네트워크 연결 보기] – [Windows 방화벽 설정 변경] – [고급]

– ICMP 항목의 [설정(E)…] – [수신 에코 요청 허용] 체크



구문

short WINAPI **nmf_PingCheck** (short nNmfNo, short nIpAddr0, short nIpAddr1,
short nIpAddr2, int lWaitTime);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호 (주소 4번째 자리이며, 전면 로터리 스위치 값)
2	nIpAddr0	IN	장치 IP주소 첫번째 (ex, 192.xxx.xxx.xxx)
3	nIpAddr1	IN	장치 IP주소 두번째 (ex, xxx.168.xxx.xxx)
4	nIpAddr2	IN	장치 IP주소 세번째 (ex, xxx.xxx.0.xxx)
5	lWaitTime	IN	응답 대기시간(ms, 출하값 50ms) ※항상 0보다 큰 값을 입력하여 주십시요.

사용 예

nmf_Connect 항목을 참고하여 주십시오.

주의사항

No	내용
1	nmf_Connect 함수를 통한 연결 확인이 아닌 네트워크 대역의 물리적 연결상태를 확인합니다.

관련 참조

nmf_Connect, nmf_Disconnect, nmf_PingCheck

Memo

☑ Ping (Packet InterNet Groper)

Ping은 현재의 네트워크 상태를 체크할 때 쓰이는 ICMP중 하나입니다.
Ping은 현 네트워크의 특정 호스트에 일정 패킷을 보내 응답이 있는지 체크하는 명령어 입니다.
따라서, 특정 호스트가 연결되어 있는지를 알 수 있게 해줍니다.
하지만, ICMP를 방화벽으로 막아 놓을 시에 응답이 없을 수 있습니다

☑ ICMP (Internet Control Message Protocol)

호스트 서버와 인터넷 게이트웨이 사이에서 메시지를 제어하고 에러를 알려주는 프로토콜입니다.
ICMP는 IP 데이터그램을 사용하지만, 메시지는 TCP/IP 소프트웨어에 의해 처리되며, 응용프로그램 사용자에게 직접적으로 보이지는 않습니다. 예로 Ping 명령어는 인터넷 접속을 테스트하기 위해 ICMP를 사용합니다.
ICMP는 IETF(The Internet Engineering Task Force)의 RFC(Request for Comments) 792에 정의되어 있습니다.

3.2. 환경설정

장치의 통신방식 및 데이터의 송수신 제한시간을 설정합니다.

관련함수

No	명령	설명
1	nmf_TMOSet	네트워크 데이터 송수신의 제한시간을 설정합니다.(ms단위)
2	nmf_TMOGet	설정된 제한시간을 확인합니다.(ms단위)
3	nmf_ProtocolMethodSet	장치의 통신방식을 설정합니다.
4	nmf_ProtocolMethodGet	장치의 통신방식을 확인합니다.

3.2.1. 통신 Time Out 설정

nmf_TMOSet	Public

장치와 통신 중, 함수 명령에 대한 송수신 제한시간을 설정합니다.

구문

```
short WINAPI nmf_TMOSet (short nNmfNo, short nTMO_Conn,
                         short nTMO_Tx, short nTMO_Rx);
```

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nTMO_Conn	IN	장치와 연결을 시도할 때 사용되는 제한시간입니다.(ms)
3	nTMO_Tx	IN	장치로 명령을 전송할 때 사용되는 제한시간입니다.(ms)
4	nTMO_Rx	IN	장치로 전송된 명령의 응답을 기다리는 제한시간입니다.(ms)

주의사항

No	내용
1	무선이나 장거리에 대한 송수신 시간이 조정되어야 하는 경우 사용하여 주십시오.

사용 예

nmf_TMOGet 항목을 참고하여 주십시오.

관련 참조

nmf_Connect, nmf_Disconnect, nmf_PingCheck

3.2.2. 통신 Time Out 설정 확인

nmf_TMOGet	Public

설정한 송수신 제한시간을 확인합니다.

구문

```
short WINAPI nmf_TMOGet (short nNmfNo, short *pnRetTMO_Conn,
                         short *pnRetTMO_Tx, short *pnRetTMO_Rx);
```

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	pnRetTMO_Conn	OUT	장치와 연결을 시도할 때 사용되는 제한시간입니다.(ms)
3	pnRetTMO_Tx	OUT	장치로 명령을 전송할 때 사용되는 제한시간입니다.(ms)
4	pnRetTMO_Rx	OUT	장치로 전송된 명령의 응답을 기다리는 제한시간입니다.(ms)

사용 예

```
short nRet;
short nNmfNo = 11;    // 장치 ID

// IP 192.168.0.11번 장치와 연결을 시도합니다.
nRet = nmf_Connect(nNmfNo, 192, 168, 0);
if ( nRet != 0 )  return;      // 장치 연결 실패

// 연결 시 50ms, Tx_Send 시 30ms, Rx_Receive 시 30ms Time Out
nRet = nmf_TMOSet(nNmfNo, 50, 30, 30);
if ( nRet != 0 )  printf("제한시간 설정 실패!");


short nRetTMO_Conn;
short nRetTMO_Tx;
short nRetTMO_Rx;
// 장치에 설정된 TimeOut 값을 확인합니다.
nRet = nmf_TMOGet(nNmfNo, &nRetTMO_Conn, &nRetTMO_Tx, &nRetTMO_Rx);
if ( nRet != 0 )
{
    printf("제한시간 확인 실패!");
}
printf("연결 시도 제한시간 : %d\n", nRetTMO_Conn);
printf("명령 전송 제한시간 : %d\n", nRetTMO_Tx);
printf("명령 응답대기 제한시간 : %d\n", nRetTMO_Rx);

// 장치 연결을 해제합니다.
nmf_Disconnect(nNmfNo);
```

관련 참조

nmf_Connect, nmf_Disconnect, nmf_PingCheck

3.2.3. 통신방식 설정 및 확인

nmf_ProtocolMethodSet **nmf_ProtocolMethodGet**	Public

Ethernet 통신방식으로 TCP와 UDP 설정을 수행하며, 설정된 현재 통신방식을 확인합니다.

구문

short WINAPI **nmf_ProtocolMethodSet** (short nNmfNo, short nMethod);

short WINAPI **nmf_ProtocolMethodGet** (short nNmfNo);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nMethod 반환값	IN	통신방식 구분 / 통신방식: 0 – TCP (출하값); 1 – UDP

주의사항

No	일람
1	nmf_ProtocolMethodGet 명령 시, 현재 통신방식은 **반환값**으로 확인할 수 있습니다.
2	UDP를 사용하는 경우, 보안 설정 해제 항목을 참고하여 주십시오.

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호

nRet = nmf_PingCheck(nNmfNo, 192, 168, 0, 200);   // 장치와의 연결을 확인합니다.
if ( nRet != nmf_R_OK )      return;      // 장치가 확인되지 않습니다.

nRet = nmf_ProtocolMethodSet(nNmfNo, 0);
if ( nRet != nmf_R_OK )      return;      // 통신방식 설정 실패

nRet = nmf_Connect(nNmfNo, 192, 168, 0);      // 장치와 연결합니다.
if ( nRet != nmf_R_OK )      return;      // 장치 연결 실패

nRet = nmf_ProtocolMethodGet(nNmfNo);
if ( nRet != nmf_R_OK )      return;      // 통신방식 확인 실패
```

관련 참조

nmf_PingCheck, nmf_TMOSet, nmf_TMOGet,

4.DIO (Digital Input Output)

디지털 입/출력 기능을 지원하는 DI, DO 보드입니다. 각 16 Pin, 8개의 보드를 증착하여 최대 128 Pin의 운영이 가능합니다. 외부 전원을 이용한 출력을 사용하며, 상세 사양은 H/W 매뉴얼을 확인하여 주십시오.



관련함수

No	명령	설명
	입/출력 상태 확인 (일괄 Pin번호 지정방식)	
1	nmf_DIGet	입력 상태를 확인합니다.
2	nmf_DOGet	출력 상태를 확인합니다.
	출력설정 (일괄 Pin번호 지정방식)	
3	nmf_DOSet	출력 상태를 설정합니다.
4	nmf_DOSetPin	단일 출력Pin을 선택하여 출력을 설정합니다.
5	nmf_DOSetTogPin	단일 출력Pin을 선택하여 현재의 출력을 반전시킵니다.
6	nmf_DOSetPins	출력Pin을 여러 개 선택하여 출력을 설정합니다.
7	nmf_DOSetTogPins	출력Pin을 여러 개 선택하여 현재의 출력을 반전시킵니다.
	출력유지 제한시간 설정 (일괄 Pin번호 지정방식)	
8	nmf_DOSetLimitTime	출력유지 제한시간을 설정합니다.
9	nmf_DOGetLimitTime	설정된 출력유지 제한시간을 확인합니다.
	출력설정 (보드 지정)	
10	nmf_DOBrdSetPin	DO보드의 출력Pin을 하나 선택하여 출력을 제어합니다.
11	nmf_DOBrdSetPins	DO보드의 출력Pin을 여러 개 선택하여 출력을 제어합니다.
12	nmf_DOmBrdSetPins	DO보드 및 출력Pin을 여러 개 선택하여 출력을 제어합니다.
13	nmf_DOBrdSetTogPin	DO보드의 출력Pin을 하나 선택하여 출력을 반전시킵니다.
14	nmf_DOBrdSetTogPins	DO보드의 출력Pin을 여러 개 선택하여 출력을 반전시킵니다.
15	nmf_DOmBrdSetTogPins	DO보드 및 출력Pin을 여러 개 선택하여 출력을 반전시킵니다.

4.1. 입/출력 상태확인

4.1.1. 입력상태 확인

nmf_DIGet	DI

장치에 연결된 모든 DI보드의 입력상태를 확인합니다.

구문

short WINAPI **nmf_DIGet** (short nNmfNo, short *pnInStatus);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	pnInStatus	OUT	확인할 입력상태 short 형 배열 포인터 (크기 128개 고정) 값 0: 입력 OFF 값 1: 입력 ON

주의사항

No	일람
1	지정되는 Pin번호는 『일괄 Pin번호 지정방식』을 참고하세요.
2	입력상태를 받는 배열 수량은 최대 128개로 고정됩니다.

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호

/*   장치가 연결된 상태 입니다.   */

// 입력 상태 읽기
short nInNum = 128;
short nInStatus[128];
nRet = nmf_DIGet(nNmfNo, nInStatus);
if ( nRet != nmf_R_OK )      return;      // 입력상태 확인 실패

for (int i=0; i< nInNum; i++)
{
    if (nInStatus [i] == 1)          printf("입력 %d핀 ON상태", i);
    if (nInStatus [i] == 0)          printf("입력 %d핀 OFF상태", i);
}
```

관련 참조

nmf_DIGet, nmf_DOGet, nmf_DOSet, nmf_DOSetPin, nmf_DOSetTogPin,
nmf_DOSetPins, nmf_DOSetTogPins

4.1.2. 출력상태 확인

nmf_DOGet	DO

장치에 연결된 모든 DO보드의 출력상태를 확인합니다.

구문

short WINAPI **nmf_DOGet** (short nNmfNo, short *pnOutStatus);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	pnOutStatus	OUT	확인할 출력상태 short 형 배열 포인터 (크기 128개 고정) 값 0: 출력 OFF 값 1: 출력 ON

주의사항

No	일람
1	지정되는 Pin번호는 『일괄 Pin번호 지정방식』을 참고하세요.
2	출력상태를 받는 배열 수량은 최대 128개로 고정됩니다.

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호

/*   장치가 연결된 상태 입니다.   */

// 출력 상태 확인하기
short nOutNum = 128;
short nOutStatus[128];
nRet = nmf_DOGet(nNmfNo, nOutStatus);
if ( nRet != nmf_R_OK )      return;      // 출력상태 확인 실패

for (int i=0; i< nOutNum; i++)
{
    if (nOutStatus [i] == 1)       printf("출력 d핀 ON상태", i);
    if (nOutStatus [i] == 0)       printf("출력 d핀 OFF상태", i);
}
```

관련 참조

nmf_DIGet, nmf_DOGet, nmf_DOSet, nmf_DOSetPin, nmf_DOSetTogPin,
nmf_DOSetPins, nmf_DOSetTogPins

4.2. 출력설정

4.2.1. 전체 출력설정

nmf_DOSet	DO

장치에 연결된 모든 DO보드의 출력상태를 설정합니다.

구문

short WINAPI **nmf_DOSet** (short nNmfNo, short *pnOutStatus);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	pnOutStatus	IN	설정할 출력상태 short 형 배열 포인터 (크기 128개 고정) 값 0: 출력 OFF 값 1: 출력 ON

주의사항

No	일람
1	입력상태를 받는 배열 수량은 최대 128개로 고정됩니다. 배열의 Index별로 해당Pin에 매칭되며 Pin번호는 1.2.3 일괄 Pin번호 지정방식을 참고하세요

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호

/*   장치가 연결된 상태 입니다.   */

// DO 출력 Pin 전체를 OFF설정
short nOutNum = 128;
short nOutStatus[128];
for (int i =0; i <nOutNum; i ++)        nOutStatus[i] = 0;
nRet = nmf_DOSet(nNmfNo, nOutStatus);
if ( nRet != nmf_R_OK )       return;       // 출력설정 실패
```

관련 참조

nmf_DIGet, nmf_DOGet, nmf_DOSet, nmf_DOSetPin, nmf_DOSetTogPin,
nmf_DOSetPins, nmf_DOSetTogPins

4.2.2. 단일 출력Pin 설정

nmf_DOSetPin	DO

장치에 연결된 모든 DO보드의 출력 Pin 중 하나를 선택하여 출력상태를 설정합니다.

구문

short WINAPI **nmf_DOSetPin** (short nNmfNo, short nPinNo, short nOutStatus);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinNo	IN	설정할 Pin 번호 (범위 : 전체 DO보드 출력 Pin 개수)
3	nOutStatus	IN	설정할 출력상태 값 0 : 출력 OFF 값 1 : 출력 ON

주의사항

No	일람
1	출력 범위(DO Board 수량과 전체 출력 Pin 개수)를 확인하여 주십시오.
2	지정되는 Pin번호는 『일괄 Pin번호 지정방식』을 참고하세요.

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호

/*   장치가 연결된 상태 입니다.   */

// DO 출력 Pin 12 번에 출력 ON 설정
nRet = nmf_DOSetPin(nNmfNo, 12, 1);
if ( nRet != nmf_R_OK )      return;      // 단일 Pin 출력설정 실패

// DO 출력Pin상태 읽기
short nOutStatus[128];
nRet = nmf_DOGet(nNmfNo, nOutStatus);
    if ( nOutStatus [12] == 1 )   printf("출력 Pin 12번 ON상태");
    if ( nOutStatus [12] == 0 )   printf("출력 Pin 12번 OFF상태");
}
```

관련 참조

nmf_DIGet, nmf_DOGet, nmf_DOSet, nmf_DOSetPin, nmf_DOSetTogPin,

nmf_DOSetPins, nmf_DOSetTogPins

4.2.3. 단일 출력Pin 반전

nmf_DOSetTogPin	DO

장치에 연결된 모든 DO보드의 출력 Pin 중 하나를 선택하여 출력상태를 반전(현재 상태가 출력 ON이면 OFF로, 출력 OFF이면 ON으로 변경)시킵니다.

구문

short WINAPI **nmf_DOSetTogPin** (short nNmfNo, short nPinNo);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinNo	IN	설정할 Pin 번호 (범위 : 전체 DO보드 출력 Pin 개수))

주의사항

No	일람
1	출력 범위(DO Board 수량과 전체 출력 Pin 개수)를 확인하여 주십시오.
2	지정되는 Pin번호는 1.2.3 일괄 Pin번호 지정방식을 참고하세요.

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호

/*   장치가 연결된 상태 입니다.   */

// DO 출력 Pin 12 번에 출력 반전 설정
nRet = nmf_DOSetTogPin(nNmfNo, 12);
if ( nRet != nmf_R_OK )      return;       // 단일 출력반전 실패

// DO 출력Pin상태 읽기
short nOutStatus[128];
nRet = nmf_DOGet(nNmfNo, nOutStatus);
    if ( nOutStatus [12] == 1 )   printf("출력 Pin 12번 ON상태");
    if ( nOutStatus [12] == 0 )   printf("출력 Pin 12번 OFF상태");
}
```

관련 참조

nmf_DIGet, nmf_DOGet, nmf_DOSet, nmf_DOSetPin, nmf_DOSetTogPin,
nmf_DOSetPins, nmf_DOSetTogPins

4.2.4. 다중 출력Pin 설정

nmf_DOSetPins	DO

장치에 연결된 모든 DO보드의 출력 Pin 중 여러 개를 선택하여 출력상태를 설정합니다.

구문

```
short WINAPI nmf_DOSetPins (short nNmfNo, short nPinCount, short *pnPinNo,
                            short *pnStatus);
```

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinCount	IN	설정할 Pin 개수 (범위 : 전체 DO보드 출력 Pin 개수))
3	pnPinNo	IN	선택할 Pin 번호 short 형 배열 포인터(최대 128개)
4	pnStatus	IN	설정할 출력상태 short 형 배열 포인터(최대 128개)

값	상태
0	출력 OFF
1	출력 ON

주의사항

No	일람
1	출력 범위(DO Board 수량과 전체 출력 Pin 개수)를 확인하여 주십시오.
2	지정되는 Pin번호는 『일괄 Pin번호 지정방식』을 참고하세요.

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호

/*   장치가 연결된 상태 입니다.   */

short nPinCnt = 3;
short nPinNo[3] = {0,};
short nStatus[3] = {0,};
// 출력 Pin 3개 상태 변경
nPinNo[0] = 4;       nStatus[0] = 1;       // DO 출력 PIN 4 번에 출력 ON
nPinNo[1] = 1;       nStatus[1] = 1;       // DO 출력 PIN 1 번에 출력 ON
nPinNo[2] = 7;       nStatus[2] = 0;       // DO 출력 PIN 7 번에 출력 OFF
nRet = nmf_DOSetPins(nNmfNo, nPinCnt, nPinNo, nStatus);
if ( nRet != nmf_R_OK )      return;       // 다중 출력 실패
```

관련 참조

nmf_DIGet, nmf_DOGet, nmf_DOSet, nmf_DOSetPin, nmf_DOSetTogPin,
nmf_DOSetPins, nmf_DOSetTogPins

4.2.5. 다중 출력Pin 반전

nmf_DOSetTogPins	DO

장치에 연결된 모든 DO보드의 출력 Pin 중 여러 개를 선택하여 출력상태를 반전(현재 상태가 출력 ON이면 OFF로, 출력 OFF이면 ON으로 변경)시킵니다.

구문

short WINAPI **nmf_DOSetTogPins** (short nNmfNo, short nPinCount, short *pnPinNo);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinCount	IN	설정할 Pin 개수 (범위 : 전체 DO보드 출력 Pin 개수)
3	pnPinNo	IN	선택할 Pin 번호 short 형 배열 포인터(최대 128개)

주의사항

No	일람
1	출력 범위(DO Board 수량과 전체 출력 Pin 개수)를 확인하여 주십시오.
2	지정되는 Pin번호는 『일괄 Pin번호 지정방식』을 참고하세요.

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호

/*   장치가 연결된 상태 입니다.   */

short nPinCnt = 3;
short nPinNo[3] = {4, 1, 7};        // 출력 Pin 3개 상태 변경
nRet = nmf_DOSetTogPins(nNmfNo, nPinCnt, nPinNo);
if ( nRet != nmf_R_OK )      return;       // 다중 출력반전 실패

// 출력 상태 확인하기
short nOutNum = 128;
short nOutStatus[128];
nRet = nmf_DOGet(nNmfNo, nOutStatus);

for (int i=0; i< nOutNum; i++){
     if (nOutStatus [i] == 1)        printf("DO=%d핀 ON상태", i);
     if (nOutStatus [i] == 0)        printf("DO=%d핀 OFF상태", i);
}
```

관련 참조

nmf_DIGet, nmf_DOGet, nmf_DOSet, nmf_DOSetPin, nmf_DOSetTogPin,

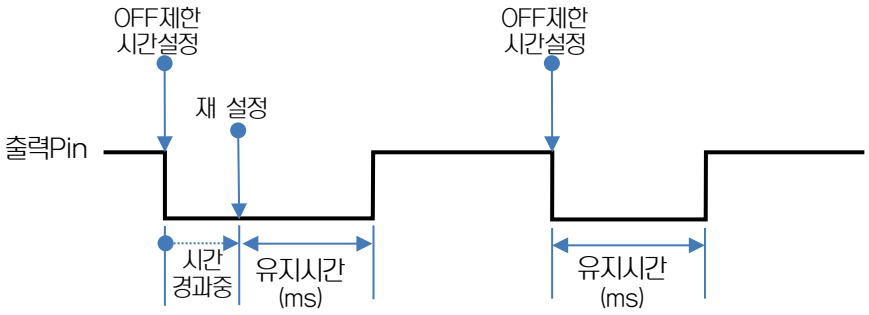
nmf_DOSetPins, nmf_DOSetTogPins

4.3. 출력유지 제한시간 설정

출력Pin의 출력 유지시간을 제한합니다.

예를들어, 출력Pin에 밸브(Valve)가 연결되어 ON을 한후 8초안에 OFF가 되어야 하는데,

통신이상이 발생하거나 상위 제어기(PC)가 Down되는 현상으로 인해 OFF를 하지 못하는 상황을 예방하기 위한 안전기능입니다. 유지 제한시간 경과 중에 다시 기능을 설정하면 시간초기화가 이루어 집니다.



4.3.1. 제한 설정

nmf_DOSetLimitTime	DO

출력상태 유지의 제한시간을 설정합니다.

구문

short WINAPI **nmf_DOSetLimitTime** (short nNmfNo, short nPinNo,
short nOutStatus, int lTime);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinNo	IN	지정될 Pin번호 (범위 : 0 ~ 127)
3	nOutStatus	IN	설정할 출력 값 / 상태 0 / 출력 OFF 1 / 출력 ON
4	lTime	IN	유지 제한시간을 ms단위로 설정하며, 지정할수 있는 최대 시간은 2,147,483,647ms입니다. 값 / 상태 0보다 큰 경우 / 지정된 ms단위의 제한시간을 사용 0과 같은 경우 / 제한시간을 해제

주의사항

No	일람
1	해당 함수로 설정된 출력값은 유지시간 경과 후, 반전됩니다.
2	지정되는 Pin번호는 『일괄 Pin번호 지정방식』을 참고하세요.

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호
short nPinNo = 8; // 출력 Pin번호

/*   장치가 연결된 상태 입니다.   */

// DO Pin을 ON한후 3000ms 이후 OFF로 설정
nRet = nmf_DOSetLimitTime(nNmfNo, nPinNo, 1, 3000);

// 80번 Pin OFF 설정 후, 10s뒤 ON 설정
nRet = nmf_DOSetLimitTime(nNmfNo, 80, 0, 10000);
// 127번 Pin ON 설정 후, 5.5s뒤 OFF 설정
nRet = nmf_DOSetLimitTime(nNmfNo, 127, 1, 5500);

// 상태확인
short nSet, nStatus, nOutStatus;
int lRemainTime;
nRet = nmf_DOGetLimitTime(nNmfNo, nPinNo, &nSet, &nStatus, &nOutStatus, &lRemainTime);
if ( nRet != nmf_R_OK )       return;
if ( nSet == 0 ){
    printf("설정안됨");
    return;
}
if ( nSet == 1 )       printf("ON으로 설정되어 시간경과 후, OFF됨");
if ( nSet == 2 )       printf("OFF으로 설정 되어 시간경과 후, ON됨");

// 남은 시간 표시
while (true){
    nRet = nmf_DOGetLimitTime(nNmfNo, nPinNo, &nSet, &nStatus, &nOutStatus, &lRemainTime);
    if ( nRet != nmf_R_OK )       break;
    if (nStatus == 0){
        printf("설정안됨");
        break;
    }
    if (nStatus == 1)      printf("제한시간 진행중 남은시간=%d", lRemainTime);
    if (nStatus == 2){
        printf("제한시간 종료");
        break;
    }
    if (nOutStatus == 0)      printf("OFF상태");
    if (nOutStatus == 1)      printf("ON상태");
}
```

관련 참조

nmf_DOSetLimitTime, nmf_DOGetLimitTime

4.3.2. 설정 확인

nmf_DOGetLimitTime	DO

설정된 출력유지 제한시간을 확인합니다.

구문

short WINAPI **nmf_DOGetLimitTime** (short nNmfNo, short nPinNo,
short *pnSet, short *pnStatus, short *pnOutStatus, int *plRemainTime);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinNo	IN	지정될 Pin번호 (범위 : 0 ~ 127)
3	pnSet	OUT	설정된 출력상태 값 0: 설정 안됨 값 1: 출력 ON으로 설정 값 2: 출력 OFF로 설정
4	pnStatus	OUT	현재 제한시간의 상태를 읽어올 short 형 포인터 변수 값 0: 설정 안됨 값 1: 제한시간 진행중 값 2: 제한시간 종료
5	pnOutStatus	OUT	현재 Pin 출력상태를 읽어올 short형 포인터 변수입니다. 값 0: 출력 OFF 값 1: 출력 ON
6	plRemainTime	OUT	제한시간이 진행중일 경우에 남은 시간을 읽어올 int형 포인터 변수입니다. (ms단위)

사용 예

nmf_DOSetLimitTime 항목을 참고하여 주십시오.

관련 참조

nmf_DOSetLimitTime, nmf_DOGetLimitTime

4.4. 보드지정 출력설정

4.4.1. 단일 출력Pin 설정

nmf_DOBrdSetPin	DO

지정된 DO보드의 Pin을 하나 선택하여 출력상태를 설정합니다.

구문

short WINAPI **nmf_DOBrdSetPin** (short nNmfNo, short nID, short nPinNo, short nOutStatus);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nID	IN	보드의 ID (범위: 연결된 DO보드의 ID (0~7))
3	nPinNo	IN	설정할 Pin 번호 (범위 : 0~ 15 Pin번호)
4	nOutStatus	IN	설정할 출력상태

값	상태
0	출력 OFF
1	출력 ON

주의사항

No	일람
1	설정된 DO 보드의 ID를 확인하여 주십시오.

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호
short nID = 1; // 출력보드 ID

/*   장치가 연결된 상태 입니다.   */
// DO 출력보드 1번의 Pin 0번 출력 ON 설정
nRet = nmf_DOBrdSetPin(nNmfNo, nID, 0, 1);

// DO 출력Pin상태 읽기
TNMF_ALL_STS  tStatus;
nRet = nmf_GetBrdAllStatus(nNmfNo, &tStatus);
if (tStatus.tDO[nID].nPinStatus[0] == 1)        printf("DO ID %d번 Pin 0번 ON상태", nID);
if (tStatus.tDO[nID].nPinStatus[0] == 0)        printf("DO ID %d번 Pin 0번 OFF상태", nID);
```

관련 참조

nmf_DOBrdSetPin, nmf_DOBrdSetPins, nmf_DOmBrdSetPins,

nmf_DOBrdSetTogPin, nmf_DOBrdSetTogPins, nmf_DOmBrdSetTogPins

4.4.2. 다중 출력Pin 설정

nmf_DOBrdSetPins	DO

지정된 DO보드의 Pin을 여러 개 선택하여 출력상태를 설정합니다.

구문

short WINAPI **nmf_DOBrdSetPins** (short nNmfNo, short nID, short nPinCount,
short *pnPinNo, short *pnOutStatus);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nID	IN	보드의 ID (범위: 연결된 DO보드의 ID (0~7))
3	nPinCount	IN	설정할 Pin 개수
4	pnPinNo	IN	설정할 Pin 번호 short 형 배열 포인터(최대 16개)
5	pnOutStatus	IN	출력상태 short 형 배열 포인터(최대 16개) 값 0: 출력 OFF 값 1: 출력 ON

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호
short nID = 1; // 출력보드 ID

/*   장치가 연결된 상태 입니다.   */

short nPinCnt = 3;
short nPinNo[3] = {0,};
short nStatus[3] = {0,};
// DO 보드 1번의 Pin 3개 출력상태 변경
nPinNo[0] = 4;        nStatus[0] = 1;        // 출력 Pin 4 번에 출력 ON
nPinNo[1] = 1;        nStatus[1] = 1;        // 출력 Pin 1 번에 출력 ON
nPinNo[2] = 7;        nStatus[2] = 0;        // 출력 Pin 7 번에 출력 OFF
nRet = nmf_DOBrdSetPins(nNmfNo, nID, nPinCnt, nPinNo, nStatus);

// DO 출력Pin상태 읽기
TNMF_ALL_STS tStatus;
nRet = nmf_GetBrdAllStatus(nNmfNo, &tStatus);
for (int i = 0; i < 16; i++){
    if (tStatus.tDO[nID].nPinStatus[i] == 1)        printf("DO ID %d번 Pin %d번 ON상태", nID, i);
    if (tStatus.tDO[nID].nPinStatus[i] == 0)        printf("DO ID %d번 Pin %d번 OFF상태", nID, i);
}
```

관련 참조

nmf_DOBrdSetPin, nmf_DOBrdSetPins, nmf_DOmBrdSetPins,

nmf_DOBrdSetTogPin, nmf_DOBrdSetTogPins, nmf_DOmBrdSetTogPins

4.4.3. 다중 보드 출력 설정

nmf_DOmBrdSetPins	DO

DO보드 및 출력Pin을 여러 개 선택하여 출력상태를 설정합니다.

구문

short WINAPI **nmf_DOmBrdSetPins** (short nNmfNo, short nBrdCount, short *pnID,
int *plOnPins, int *plOffPins);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nBrdCount	IN	설정할 DO 보드의 개수(최대 8개)
3	pnID	IN	보드의 ID short 형 배열 포인터 (범위: 연결된 DO보드의 ID (0~7))
4	plOnPins	IN	int 형 출력 On Mask (16Bit : 0~0xFFFF) 배열 포인터
5	plOffPins	IN	int 형 출력 Off Mask (16Bit : 0~0xFFFF) 배열 포인터

주의사항

No	일람
1	배열의 크기는 설정 DO 보드의 수량과 같습니다.
2	On, Off Mask는 중복되지 않도록 주의하여 주십시오.

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호

/*   장치가 연결된 상태 입니다.   */

// 장치와 연결된 출력보드 수량은 총 4개 // DO 보드 ID 0, 1, 2, 3 // 그중 3개만 출력 설정
short nBrdCount = 3;
short nBrdID[3] = {1, 2, 3};
int lOnPins[3] = {0x0F00, 0x00F0, 0x000F};
int lOffPins[3] = {0xF000, 0x0F00, 0x00F0};
nRet = nmf_DOmBrdSetPins(nNmfNo, nBrdCount, nBrdID, lOnPins, lOffPins);
```

관련 참조

nmf_DOBrdSetPin, nmf_DOBrdSetPins, nmf_DOmBrdSetPins,
nmf_DOBrdSetTogPin, nmf_DOBrdSetTogPins, nmf_DOmBrdSetTogPins

4.4.4. 단일 출력Pin 반전

nmf_DOBrdSetTogPin	DO

지정된 DO보드의 Pin을 하나 선택하여 출력상태를 반전(현재 상태가 출력 ON이면 OFF로, 출력 OFF이면 ON으로 변경)시킵니다.

구문

short WINAPI **nmf_DOBrdSetTogPin** (short nNmfNo, short nID, short nPinNo);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nID	IN	보드의 ID (범위: 연결된 DO보드의 ID (0~7))
3	nPinNo	IN	설정할 Pin 번호 (최대 16개)

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호
short nID = 1; // 출력보드 ID

/*   장치가 연결된 상태 입니다.   */

// DO 출력보드 1번의 Pin 0번 출력 반전 설정
nRet = nmf_DOBrdSetTogPin(nNmfNo, nID, 0);

// DO 출력Pin상태 읽기
TNMF_ALL_STS tStatus;

nRet = nmf_GetBrdAllStatus(nNmfNo, &tStatus);
if (tStatus.tDO[nID].nPinStatus[0] == 1)        printf("DO ID %d번 Pin 0번 ON상태", nID);
if (tStatus.tDO[nID].nPinStatus[0] == 0)        printf("DO ID %d번 Pin 0번 OFF상태", nID);
```

관련 참조

nmf_DOBrdSetPin, nmf_DOBrdSetPins, nmf_DOmBrdSetPins,

nmf_DOBrdSetTogPin, nmf_DOBrdSetTogPins, nmf_DOmBrdSetTogPins

4.4.5. 다중 출력Pin 반전

nmf_DOBrdSetTogPins	DO

지정된 DO보드의 Pin을 여러 개 선택하여 출력상태를 반전(현재 상태가 출력 ON이면 OFF로, 출력 OFF이면 ON으로 변경)시킵니다.

구문

short WINAPI **nmf_DOBrdSetTogPins** (short nNmfNo, short nID, short nPinCount,
short *pnPinNo);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nID	IN	보드의 ID (범위: 연결된 DO보드의 ID (0~7))
3	nPinCount	IN	설정할 Pin 개수
4	pnPinNo	IN	설정할 Pin 번호 short 형 배열 포인터(최대 16개)

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호
short nID = 1; // 출력보드 ID

/*   장치가 연결된 상태 입니다.   */

short nPinCnt = 3;
short nPinNo[3] = {4, 1, 7};        // 출력을 반전할 PinNo
// DO 보드 1번의 Pin 3개 출력상태 변경
nRet = nmf_DOBrdSetTogPins(nNmfNo, nID, nPinCnt, nPinNo);

// DO 출력Pin상태 읽기
TNMF_ALL_STS tStatus;
nRet = nmf_GetBrdAllStatus(nNmfNo, &tStatus);
for (int i = 0; i < 16; i++){
    if (tStatus.tDO[nID].nPinStatus[i] == 1)        printf("DO ID %d번 Pin %d번 ON상태", nID, i);
    if (tStatus.tDO[nID].nPinStatus[i] == 0)        printf("DO ID %d번 Pin %d번 OFF상태", nID, i);
}
```

관련 참조

nmf_DOBrdSetPin, nmf_DOBrdSetPins, nmf_DOmBrdSetPins,
nmf_DOBrdSetTogPin, nmf_DOBrdSetTogPins, nmf_DOmBrdSetTogPins

4.4.6. 다중 보드 출력 반전

nmf_DOmBrdSetTogPins	DO

DO보드 및 출력Pin을 여러 개 선택하여 출력상태를 반전(현재 상태가 출력 ON이면 OFF로, 출력 OFF이면 ON으로 변경)시킵니다.

구문

short WINAPI **nmf_DOmBrdSetTogPins** (short nNmfNo, short nBrdCount, short *pnID, int *plPins);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nBrdCount	IN	설정할 DO 보드의 개수(최대 8개)
3	pnID	IN	보드의 ID short 형 배열 포인터 (범위: 연결된 DO보드의 ID (0~7))
4	plPins	IN	int 형 출력 반전 Mask (16Bit : 0~0xFFFF) 배열 포인터

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호

/*   장치가 연결된 상태 입니다.   */

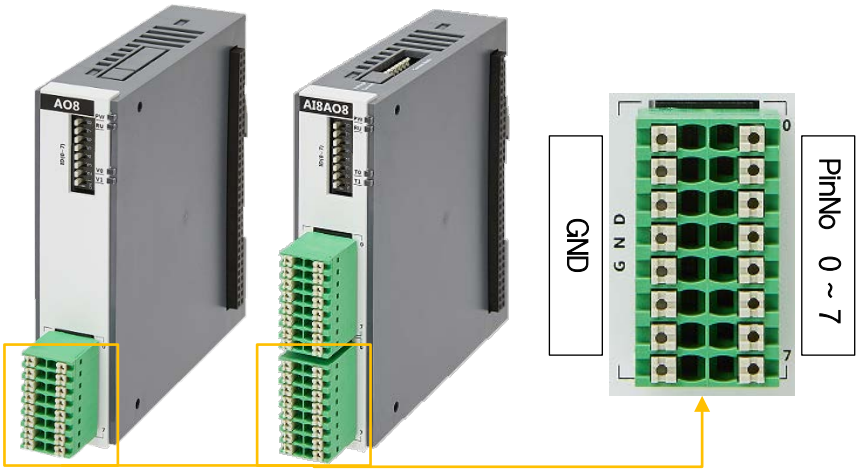
// 장치와 연결된 출력보드 수량은 총 4개 // DO 보드 ID 0, 1, 2, 3 // 그중 3개만 출력 설정
short nBrdCount = 3;
short nBrdID[3] = {1, 2, 3};
int lPins[3] = {0x0F00, 0x00F0, 0x000F};
nRet = nmf_DOmBrdSetTogPins(nNmfNo, nBrdCount, nBrdID, lPins);
```

관련 참조

nmf_DOBrdSetPin, nmf_DOBrdSetPins, nmf_DOmBrdSetPins,
nmf_DOBrdSetTogPin, nmf_DOBrdSetTogPins, nmf_DOmBrdSetTogPins

5.AO (Analog Output)

아날로드 출력기능을 지원하는 AO, AIO 입니다. 각 8 Pin, 8개의 보드를 증착하여 최대 64 Pin의 운영이 가능합니다. 상세 사양은 H/W 매뉴얼을 확인하여 주십시오.



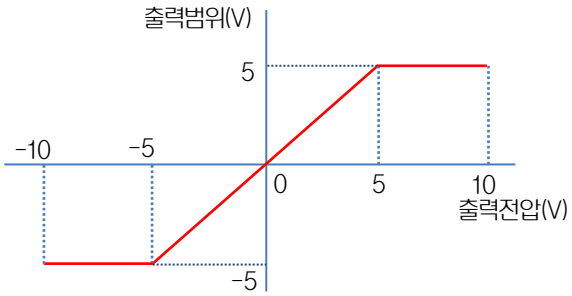
관련함수

No	명령	설명
출력설정 (일괄 Pin번호 지정방식)		
1	nmf_AOSetPin	아날로그 출력 Pin 중 하나를 선택하여 출력을 설정합니다.
2	nmf_AOSetPins	아날로그 출력 Pin을 여러 개 선택하여 출력을 설정합니다.
출력설정 (보드 지정)		
3	nmf_AOBrdSetPin	보드별 아날로그 출력 Pin을 선택하여 출력을 설정합니다.
4	nmf_AOBrdSetPins	보드별 여러 개의 아날로그 출력 Pin을 선택하여 출력을 설정합니다.
5	nmf_AOmBrdSetPins	보드와 아날로그 출력 Pin들을 여러 개 선택하여 출력을 설정합니다.

5.1. 알아두기

5.1.1. 출력범위

AO보드는 각 Pin별로 출력범위를 설정할 수 있으며, **최대 ±10.86V의 출력범위**를 제공하고 있습니다.
제한된 범위 내의 안전한 사용을 위하여, 장치에서 허용하실 최대 범위를 결정하시고 제어하여 주십시오.
아래와 같이, 출력 전압을 범위보다 초과하여 설정하는 경우, 범위를 넘겨 출력하지 않습니다.



<그림 5-1> ±5V의 출력범위를 지정하는 경우

5.1.2. 출력 예시

```
short nRet;
short nNmfNo   = 11;    // 장치 번호
short nBrdID   = 3;     // AO보드 ID
short nPinNo   = 2;     // Pin 번호

/*   장치가 연결된 상태 입니다.   */

// 보드가 여러 개 장착되어 있는 경우, 해당 AO보드들을 정렬하여 아날로그 출력을 설정합니다.
// 10번 Pin의 출력을 5V 설정, 출력범위 ±10V
nRet = nmf_AOSetPin(nNmfNo, 10, 4, 5);
if ( nRet == nmf_R_OK ){
    printf("정상적으로 출력하였습니다.");
}

// AO보드 3번의 Pin 2번에 +7.5V를 설정, 출력범위 ±5V
nRet = nmf_AOBrdSetPin(nNmfNo, nBrdID, nPinNo, 7.5);
if ( nRet == nmf_R_AO_RANGE_OVER ){
    // 단일 Analog 출력설정 실패, 출력범위를 초과하였습니다.
    printf("출력 범위 초과");
}

// AO보드 3번의 Pin들 전압을 각 1, 2, 3, 4V 로 설정
// AO보드 5번의 Pin들 전압을 각 -10, -5, 5, 10V 로 설정
short   nPinCnt = 8;
// 배열 순서에 따라 보드와 Pin번호가 맞아야 합니다.
short   nAOList[8] = {3, 3, 3, 3, 5, 5, 5, 5};  // Pin 수량에 따른 해당 보드의 ID
short   nPinList[8] = {4, 5, 6, 7, 0, 1, 2, 3};  // Pin 수량에 따른 출력 Pin 번호
short   nRangeList[8] = {3, 3, 3, 3, 3, 3, 3, 3};   // Pin 수량에 따른 출력범위 ±5V
double  dValue[8] = {1, -2, 3, 4, -10, -4, 5, 8};   // 각 Pin의 출력 전압값
nRet = nmf_AOmBrdSetPins(nNmfNo, nPinCnt, nAOList, nPinList, nRangeList, dValue);
if ( nRet == nmf_R_AO_RANGE_OVER ){
    // 다중보드 출력설정 실패, 출력범위를 초과하였습니다.
    printf("출력 범위 초과");
}

// 출력 상태 읽기
TNMF_ALL_STS tStatus;
```

```
nRet = nmf_GetBrdAllStatus(nNmfNo, &tStatus);
for (int i = 0; i < nPinCnt; i++)
{
    printf("AO ID %d번 Pin %d번 DA Value = %d", nAOList[i], nPinList[i], tStatus.tAO[nID].lDAValue[i]);
    printf("AO ID %d번 Pin %d번 Volt = %d", nAOList[i], nPinList[i], tStatus.tAO[nID].dVolt[i]);
}
```

5.2. 출력설정

5.2.1. 단일 출력Pin 설정

nmf_AOSetPin	AO

장치에 연결된 모든 AO보드의 출력 Pin 중 하나를 선택하여 출력 전압을 설정합니다.

구문

short WINAPI **nmf_AOSetPin** (short nNmfNo, short nPinNo, short nRange, double dValue);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinNo	IN	설정할 Pin 번호 (범위 : 전체 AO Pin 개수)
3	nRange	IN	출력 범위
4	dValue	IN	출력 전압값

nRange 값:

값	범위	값	범위
0	0~5V	3	±5V
1	0~10V	4	±10V
2	0~10.86V	5	±10.86V

주의사항

No	일람
1	지정되는 Pin번호는 『일괄 Pin번호 지정방식』을 참고하세요.

사용 예

출력설정 예시 항목을 참고하여 주십시오.

관련 참조

nmf_AOSetPin, nmf_AOSetPins, nmf_AOBrdSetPin, nmf_AOBrdSetPins, nmf_AOmBrdSetPins

5.2.2. 다중 출력Pin 설정

nmf_AOSetPins	AO

장치에 연결된 모든 AO보드의 출력 Pin 중 여러 개를 선택하여 출력 전압을 설정합니다.

구문

short WINAPI **nmf_AOSetPins** (short nNmfNo, short nPinCount, short *pnPinNo,
short *pnRange, double *pdValue);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinCount	IN	설정할 Pin 개수 (범위 : 전체 AO Pin 개수)
3	pnPinNo	IN	설정할 Pin 번호 short 형 배열 포인터 (범위 : 전체 AO Pin 개수)
4	pnRange	IN	출력 범위 short 형 배열 포인터 (범위 : 전체 AO Pin 개수)
5	pdValue	IN	double 형 출력 전압값 배열 포인터 (범위 : 전체 AO Pin 개수)

pnRange 값:

값	범위	값	범위
0	0~5V	3	±5V
1	0~10V	4	±10V
2	0~10.86V	5	±10.86V

주의사항

No	일람
1	지정되는 Pin번호는 『일괄 Pin번호 지정방식』을 참고하세요.

사용 예

출력설정 예시 항목을 참고하여 주십시오.

관련 참조

nmf_AOSetPin, nmf_AOSetPins, nmf_AOBrdSetPin, nmf_AOBrdSetPins, nmf_AOmBrdSetPins

5.3. 보드지정 출력설정

5.3.1. 단일 출력Pin 설정

nmf_AOBrdSetPin	AO

지정된 AO보드의 Pin을 하나 선택하여 출력 전압을 설정합니다.

구문

short WINAPI **nmf_AOBrdSetPin** (short nNmfNo, short nID, short nPinNo, short nRange, double dValue);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nID	IN	보드의 ID (범위: 연결된 AO보드의 ID (0~7))
3	nPinNo	IN	설정할 Pin 번호 (0~7)
4	nRange	IN	출력 범위
5	dValue	IN	출력 전압값

nRange 출력 범위

값	범위	값	범위
0	0~5V	3	±5V
1	0~10V	4	±10V
2	0~10.86V	5	±10.86V

주의사항

No	일람
1	AO보드 ID를 확인하여 주십시오.

사용 예

출력설정 예시 항목을 참고하여 주십시오.

관련 참조

nmf_AOSetPin, nmf_AOSetPins, nmf_AOBrdSetPin, nmf_AOBrdSetPins, nmf_AOmBrdSetPins

5.3.2. 다중 출력Pin 설정

nmf_AOBrdSetPins	AO

지정된 AO보드의 Pin을 여러 개 선택하여 출력 전압을 설정합니다.

구문

short WINAPI **nmf_AOBrdSetPins** (short nNmfNo, short nID, short nPinCount,
short *pnPinNo, short *pnRange, double *pdValue);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nID	IN	보드의 ID (범위: 연결된 AO보드의 ID (0~7))
3	nPinCount	IN	설정할 Pin 개수 (최대 8개)
4	pnPinNo	IN	설정할 Pin 번호 short 형 배열 포인터 (최대 8개)
5	pnRange	IN	출력 범위 short 형 배열 포인터 (최대 8개)
6	pdValue	IN	double 형 출력 전압값 배열 포인터 (최대 8개)

pnRange 값:

값	범위	값	범위
0	0~5V	3	±5V
1	0~10V	4	±10V
2	0~10.86V	5	±10.86V

주의사항

No	일람
1	AO보드 ID를 확인하여 주십시오.
2	배열 최대크기(8)는 출력보드의 Pin수(8) 입니다.

사용 예

출력설정 예시 항목을 참고하여 주십시오.

관련 참조

nmf_AOSetPin, nmf_AOSetPins, nmf_AOBrdSetPin, nmf_AOBrdSetPins, nmf_AOmBrdSetPins

5.3.3. 다중 보드 출력 설정

nmf_AOmBrdSetPins	AO

AO보드와 Pin을 여러 개 선택하여 출력을 설정합니다.

구문

short WINAPI **nmf_AOmBrdSetPins** (short nNmfNo, short nPinCount, short *pnID,
short *pnPinNo, short *pnRange, double *pdValue);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinCount	IN	설정할 Pin 수량 (최대 64개)
3	pnID	IN	보드의 ID short 형 배열 포인터 (최대 64개) (범위 : 연결된 AO보드의 ID (0~7))
4	pnPinNo	IN	설정할 Pin 번호 short 형 배열 포인터 (최대 64개)
5	pnRange	IN	출력 범위 short 형 배열 포인터 (최대 64개)
6	pdValue	IN	double 형 출력 전압값 배열 포인터 (최대 64개)

pnRange 값:

값	범위	값	범위
0	0~5V	3	±5V
1	0~10V	4	±10V
2	0~10.86V	5	±10.86V

주의사항

No	일람
1	AO보드 ID를 확인하여 주십시오.
2	배열 최대크기(64)는 최대 아날로그 입력보드 수량(8) X 각 입력보드의 Pin수(8) 입니다.

사용 예

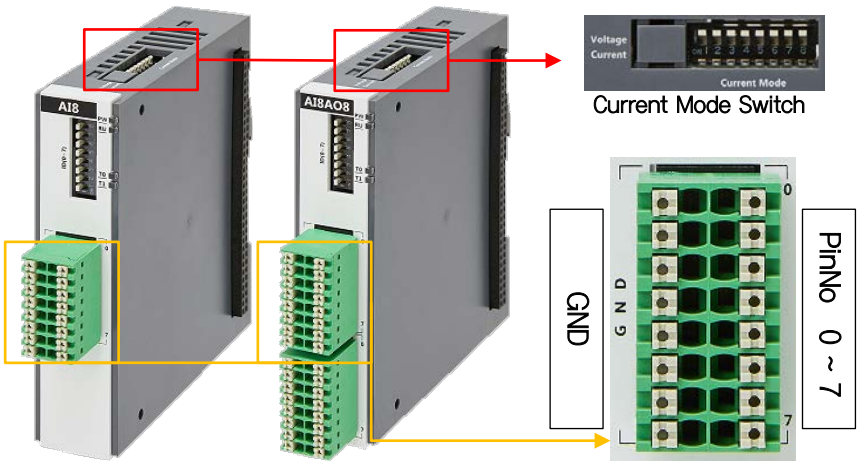
출력설정 예시 항목을 참고하여 주십시오.

관련 참조

nmf_AOSetPin, nmf_AOSetPins, nmf_AOBrdSetPin, nmf_AOBrdSetPins, nmf_AOmBrdSetPins

6.AI (Analog Input)

아날로드 입력기능을 지원하는 AI, AIO 입니다. 각 8 Pin, 8개의 보드를 증착하여 최대 64 Pin의 운영이 가능합니다. 상세 사양은 H/W 매뉴얼을 확인하여 주십시오.



관련함수

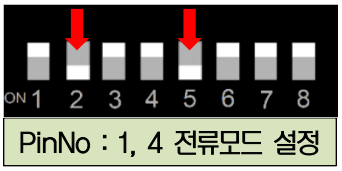
No	명령	설명
	입력범위 설정	
1	nmf_AIBrdSetRange	아날로그 입력범위를 설정합니다.
	입력방식 설정 및 정지 (일괄 Pin번호 지정방식)	
2	nmf_AISmplStartPin	아날로그 입력 Pin 중 하나를 선택하여 입력 방식을 설정합니다.
3	nmf_AISmplStartPins	아날로그 입력 Pin을 여러 개 선택하여 입력 방식을 설정합니다.
4	nmf_AISmplStopPin	아날로그 입력 Pin 중 하나를 선택하여 입력 상태 를 정지합니다.
5	nmf_AISmplStopPins	아날로그 입력 Pin을 여러 개 선택하여 입력 상태 를 정지합니다.
	입력방식 설정 및 정지 (보드 지정)	
6	nmf_AIBrdSmplStartPin	보드를 지정하고 단일 Pin을 선택하여 입력상태를 설정합니다.
7	nmf_AIBrdSmplStartPins	보드를 지정하고 Pin을 여러 개 선택하여 입력상태를 설정합니다.
8	nmf_AImBrdSmplStartPins	보드와 Pin을 여러 개 선택하여 입력상태를 설정합니다.
9	nmf_AIBrdSmplStopPin	보드를 지정하고 단일 Pin을 선택하여 입력상태를 정지시킵니다.
10	nmf_AIBrdSmplStopPins	보드를 지정하고 Pin을 여러 개 선택하여 입력상태를 정지시킵니다.
11	nmf_AImBrdSmplStopPins	보드와 Pin을 여러 개 선택하여 입력상태를 정지시킵니다.
	Sampling Rate 확인	
12	nmf_AISmplRateGetPin	입력 Pin중 하나에 실제 적용된 Sampling rate를 확인합니다.
13	nmf_AISmplRateGetPins	여러 Pin에 실제 적용된 Sampling rate를 확인합니다.
14	nmf_AIBrdSmplRateGetPin	보드를 지정하고 단일 Pin 적용된 Sampling rate를 확인합니다.
15	nmf_AIBrdSmplRateGetPins	보드를 지정하고 Pin을 여러 개 선택하여 적용된 Sampling rate를 확인합니다.
16	nmf_AImBrdSmplRateGetPins	보드와 Pin을 여러 개 선택하여 적용된 Sampling rate를 확인합니다.
	메모리 설정 및 확인	
17	nmf_AISmplExMemSet	아날로그 입력을 받을 메모리를 설정합니다. (일괄 Pin번호 지정방식)
18	nmf_AIBrdSmplExMemSet	보드를 선택하여 아날로그 입력을 받을 메모리를 설정합니다.
19	nmf_AIBrdSmplExMemGet	메모리 설정상태를 확인합니다.
	샘플링 데이터 확인	
20	nmf_AIBrdSmplValueGetPin	샘플링되어 저장된 데이터를 확인합니다.

6.1. 알아두기

6.1.1. 입력방식

AI, AIO 보드의 상단에는 각 Pin별로 설정이 가능한 전압/전류 모드 Switch가 있습니다.





PinNo : 1, 4 전류모드 설정

Pin 번호 = Switch 번호 - 1

별도의 설정없이 물리적 배선이 이루어진 상태에서 입력이 있다면, 전체상태를 확인하여, AI보드의 각 Pin에 입력되는 전압/전류 값을 확인할 수 있습니다.

그 외의 형태로 Sampling 기능을 사용할 때, AI 샘플링 설정함수를 통하여, 각 Pin의 설정을 수행하게 됩니다.

일반적으로 전압/전류를 확인하고자 하시는 경우, 전체상태 - AI 상태정보를 참고하여 주십시오.

6.1.2. Sampling (샘플링)

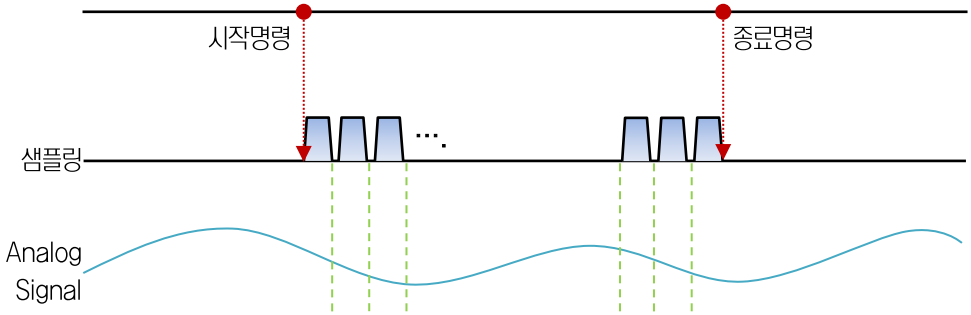
기본적인 입력값 확인이 아닌, 일정 시간동안 입력되는 전압/전류의 변화를 확인하고자 하신다면, Sampling 기능을 설정하실 수 있습니다.

샘플링 기능 사용 시 아래의 순서로 설정을 진행하여 주십시오.

No	제어 방식	함수
1	입력범위 설정	nmf_AIBrdSetRange
2	외부 메모리 설정 (필요시에만)	nmf_AIBrdExMemSet, nmf_AIBrdExMemSet, nmf_AIBrdExMemGet
3	입력 Pin 설정 시작 및 정지	nmf_AISmplStartPin, nmf_AISmplStartPins, nmf_AIBrdSmplStartPin, nmf_AIBrdSmplStartPins, nmf_AImBrdSmplStartPins, nmf_AISmplStopPins, nmf_AIBrdSmplStopPin, nmf_AIBrdSmplStopPins, nmf_AImBrdSmplStopPins
4	입력 확인	nmf_AIBrdSmplValueGetPin

6.1.2.1. 상위 명령에 의한 동작

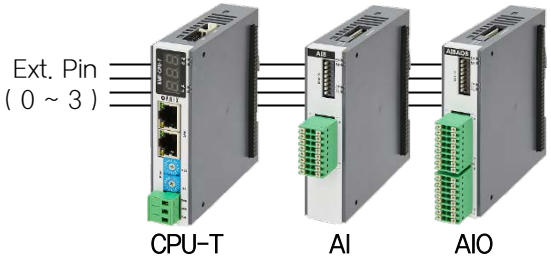
사용자가 임의의 시점에 시작명령(함수수행)을 전달하면 샘플링이 시작되며 할당된 저장공간에 데이터를 보관합니다. 시작명령은 각 AI보드 입력Pin에 샘플링을 설정하는 시점을 의미합니다.

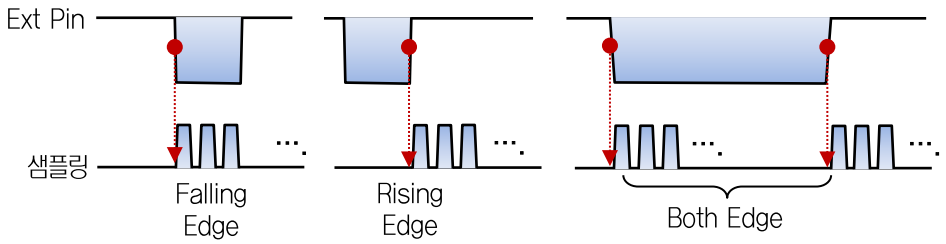


<그림 6-1> 임의시점 샘플링

6.1.2.2. 외부 입력신호(Trigger) 연동

샘플링을 시작하는 신호로 External Pin 0 ~ 3번 을 통해 입력받은 신호를 연동할 수 있습니다.



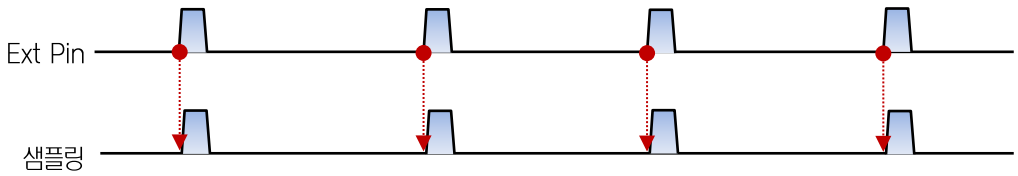


<그림 6-2> External Pin 입력에 의한 샘플링 시작

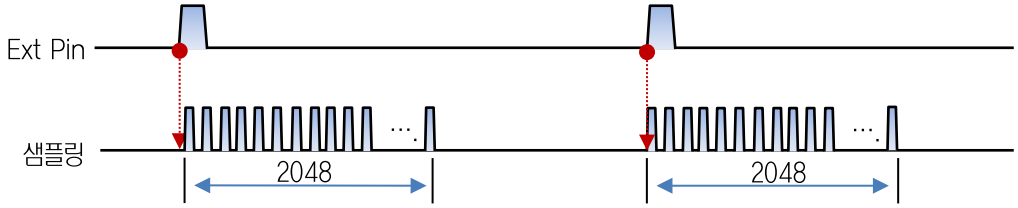
각 Pin 설정에 따라서 Falling, Rising, Both Edge를 기준으로 External Pin에서 입력이 확인되면, 샘플링을 시작하게 됩니다.

6.1.2.3. Sampling Rate

Sampling Rate는 입력변화를 확인하는 주파수로써, 초당 몇 번의 확인을 수행하는지 그 속도를 설정하는 값입니다.



<그림 6-3> Sampling Rate가 0인 경우



<그림 6-4> Sampling Rate가 0 보다 큰 경우

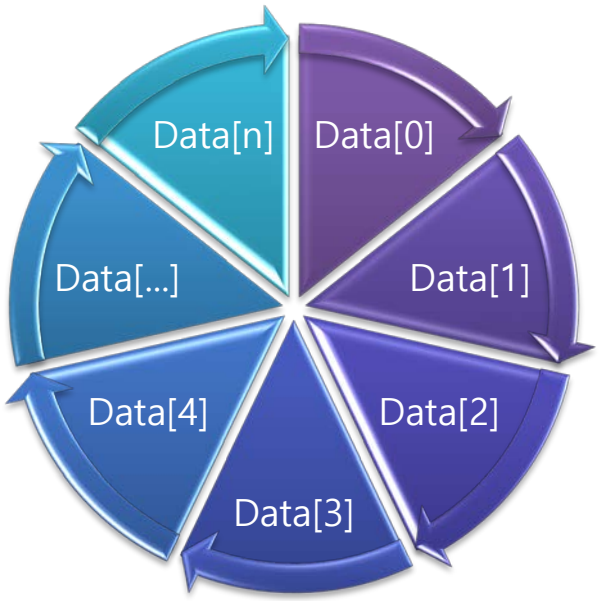
그림과 같이 값이 클수록 단위시간 당 입력을 확인하는 수량이 증가하게 됩니다.
0 ~ 100000 Hz 범위의 값이 적용될 수 있으며, 실제적으로 적용되는 값은 정수연산으로 인해 차이가 발생할 수 있습니다. 실적용값을 확인하는 함수를 통해 적용값을 확인하여 편차를 예측할 수 있습니다.

6.1.3. 메모리 관리

6.1.3.1. 데이터 처리 형태

샘플링 기능을 통해 아날로그 입력값을 확인하기 위해서는 별도의 공간(메모리)을 이용하여 확인된 변화값이 저장되어야 합니다. AI 보드의 각 Pin은 개별 메모리가 존재하며 각 2048개의 크기를 갖습니다.

최근 데이터의 갱신을 위해 자료구조 상, **FIFO(First In First Out) 형태의 원형큐를 구성**하여 입력값이 저장될 때, 메모리가 포화상태에 이르러 더 이상 저장을 할 수 없다면, 메모리에 저장된 가장 마지막 값은 최근 값으로 갱신됩니다.



<그림 6-5> 원형큐를 적용한 데이터 관리

6.1.3.2. 외부 메모리 할당

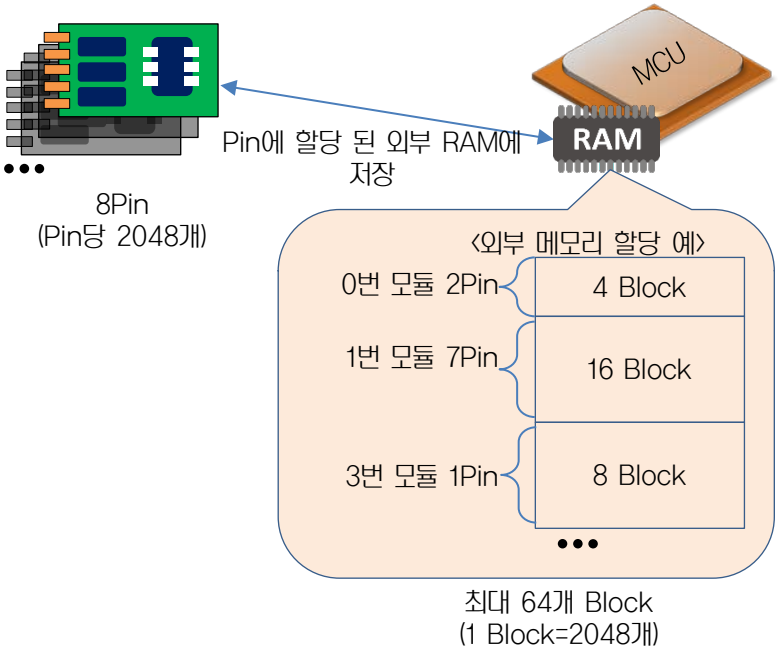
샘플링 데이터가 저장되는 메모리는 다음과 같습니다.

채널 메모리	AI 보드의 각 Pin에 기본으로 할당된 2048개의 공간
외부 메모리	CPU보드에 존재하는 추가 저장공간 (64 Block, 1Block 당 2048개)

외부 메모리를 할당하게 되면 데이터 처리형태는 동일하며 저장공간의 크기만 증가시키게 됩니다.

AI입력 Pin에 적용가능한 최대 메모리 크기는

채널 메모리 (2048) + 외부 메모리 (2048 x 64) = 133120 개 가 됩니다.



<그림 6-6> 메모리 할당의 예

6.1.3.3. 데이터의 확인

앞서 설명된 원형큐의 처리형태로 인하여, 저장된 데이터를 확인할 때는 **설정된 크기보다 한 개 적은 크기로 값이 응답**됩니다. 예를들어 2048개의 채널 메모리만 할당되었을 때, 메모리 포화상태에서 값을 확인하게 되면 응답되는 수량은 2047개가 되며, 최대 133120개의 메모리가 할당되었을 때, 포화상태에서 응답되는 수량은 133119개가 됩니다. 데이터 확인 시, 응답수량을 확인하시기 바랍니다.

샘플링 데이터 확인 함수의 예를 참고하여 주십시오.

6.2. 범위 설정

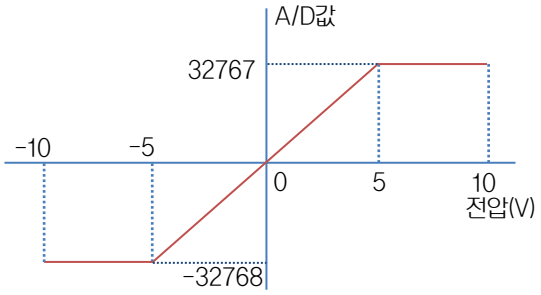
nmf_AIBrdSetRange	AI

아날로그 입력보드의 입력범위를 설정합니다.

설정된 AI보드의 입력 8Pin에서 받는 모든 입력에 적용됩니다.

입력범위를 ±5V로 설정하고 ±10V를 입력하면 아래 그림처럼 ±5V의 범위에 제한되어 입력받게 됩니다.

전압과 전류 둘 다 통용되는 함수입니다.



〈그림 6-7〉 입력범위에 따른 전압 입력상태

구문

short WINAPI **nmf_AIBrdSetRange** (short nNmfNo, short nID, short nRange);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nID	IN	보드의 ID (범위: 연결된 AI보드의 ID (0~7))
3	nRange	IN	아날로그 입력 범위 전압(V) 측정 시 값 / 입력 제한 범위 0 / ±5V 1 / ±10V 전류(mA) 측정 시 값 / 입력 제한 범위 0 / ±20mA 1 / ±40mA

주의사항

No	일람
1	입력 범위 이상의 전압은 제한 최대범위값으로 확인됩니다.

사용 예

```
#define MAX_BRD             8   // 최대 보드 수량 (CPU 장치 하나 당)
#define BRD_VS_AI_PINS      8   // AI 보드의 핀 개수

short   nRet;
short   nNmfNo = 11;    // 장치번호
```

```
/*   장치가 연결된 상태 입니다.   */

short    nBrdID = 1;       // 보드 ID
short    nRange = 0;     // 입력제한 ±5V
nRet = nmf_AIBrdSetRange(nNmfNo, nBrdID, nRange);
if (nRet != nmf_R_OK)        return;            // 입력제한 설정에 실패

// 입력 Pin 0번에 샘플링주기 1KHz, 함수 명령으로부터 시작하여 메모리가 포화될 때까지(Overwrite 안함)
// 아날로그 입력 샘플링을 시작
nRet = nmf_AISmplStartPin(nNmfNo, nBrdID, 0, 1000, 0, 0, 1);
if (nRet != nmf_R_OK)        return;

// 외부메모리 사용을 설정하며,
// 1번 보드의 다른 3개의 아날로그 입력 핀을 External Pin과 연계하여 샘플링 시작 설정
short    nPinCount = 3;
short    nID[1] = 1;
short    nBlock[3] = {8, 8, 8};     // 8 Block을 설정하는 경우 2048(외부)*8 + 2048(내부) 의 크기를 가짐.
short    nPinList     [BRD_VS_AI_PINS] = {5, 6, 7};
int      lRate        [BRD_VS_AI_PINS] = {500, 1000, 2000};
short    nTrgType     [BRD_VS_AI_PINS] = {1, 2, 3};          // Rising, Falling, Both
short    nTrgPin      [BRD_VS_AI_PINS] = {1, 2, 3};
short    nMemFullOvW  [BRD_VS_AI_PINS] = {0, 0, 1};
nRet = nmf_AIBrdSmplExMemSet (nNmfNo, nPinCount, nID, nPinList, nBlock);
nRet = nmf_AImBrdSmplStartPins (nNmfNo, nPinCount, nBrdID, nPinList, lRate, nTrgType, nTrgPin,
                                nMemFullOvW);
if (nRet != nmf_R_OK)        return;

// 아날로그 입력 중

TNMF_ALL_STS  tAllStatus;
nRet = nmf_GetBrdAllStatus(nNmfNo, &tAllStatus);
if (tAllStatus.tAI[nBrdID].nStatus == 1)              printf("AI 샘플링 수행중");
if (tAllStatus.tAI[nBrdID].nMemExists == 1)        printf("AI 샘플링 데이터가 메모리에 존재함");

printf("A/D    : %d", tAllStatus.tAI[nBrdID].lValue);      // 함수 반환 직전에 확인된 최근 값
printf("전압값 : %lf", tAllStatus.tAI[nBrdID].dVolt);      // lValue 변환값
printf("전류값 : %lf", tAllStatus.tAI[nBrdID].dCurrent);  // lValue 변환값

// 샘플링 정지
nPinCount = 4;
nPinList[BRD_VS_AI_PINS] = {0, 5, 6, 7};
nRet = nmf_AIBrdSmplStopPin(nNmfNo, nPinCount, nBrdID, nPinList);
```

관련 참조

nmf_AIBrdSetRange, nmf_AISmplStartPins, nmf_AIBrdSmplStartPin, nmf_AIBrdSmplStartPins, nmf_AImBrdSmplStartPins, nmf_AISmplStopPin, nmf_AISmplStopPins, nmf_AIBrdSmplStopPin, nmf_AIBrdSmplStopPins, nmf_AImBrdSmplStopPins, nmf_AISmplExMemSet, nmf_AIBrdSmplExMemSet, nmf_AIBrdSmplExMemGet, nmf_AIBrdSmplValueGetPin

6.3. 샘플링 설정

6.3.1. 단일 Pin 시작

nmf_AISmplStartPin	AI

장치에 연결된 모든 AI보드의 Pin 중 하나를 선택하여 Sampling 기능을 설정하고 시작합니다.

구문

short WINAPI **nmf_AISmplStartPin** (short nNmfNo, short nPinNo, int lRate, short nTrgType, short nExtPin, short nMemFullOvW);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinNo	IN	설정할 Pin 번호 (범위 : 전체 AI Pin 개수)
3	lRate	IN	샘플링 주기 (0~100000 Hz)
4	nTrgType	IN	샘플링 트리거 타입 값 0: 임의시점, 값 1: Rising, 값 2: Falling, 값 3: Both
5	nExtPin	IN	연계 External Pin 번호 (범위 : 0 ~ 3)
6	nMemFullOvW	IN	메모리 포화상태 시, Overwrite 여부 값 0: Overwrite, 값 1: Overwrite 안함

주의사항

No	일람
1	지정되는 Pin번호는 『일괄 Pin번호 지정방식』을 참고하세요.
2	nTrgType 인자의 "임의시점"은 함수명령이 장치로 전달되는 때를 의미합니다. 임의시점으로 선택시 nExtPin 번호는 의미를 갖지 않습니다.

사용 예

nmf_AIBrdSetRange 항목에 작성된 예시를 참고하여 주십시오.

관련 참조

nmf_AIBrdSetRange, nmf_AISmplStartPins, nmf_AIBrdSmplStartPin, nmf_AIBrdSmplStartPins, nmf_AImBrdSmplStartPins, nmf_AISmplStopPin, nmf_AISmplStopPins, nmf_AIBrdSmplStopPin, nmf_AIBrdSmplStopPins, nmf_AImBrdSmplStopPins, nmf_AISmplExMemSet, nmf_AIBrdSmplExMemSet, nmf_AIBrdSmplExMemGet, nmf_AIBrdSmplValueGetPin

6.3.2. 다중 Pin 시작

nmf_AISmplStartPins	AI

장치에 연결된 모든 AI보드의 Pin 중 여러 개를 선택하여 Sampling 기능을 설정하고 시작합니다.

구문

short WINAPI **nmf_AISmplStartPins** (short nNmfNo, short nPinCount, short *pnPinNo,
int *plRate, short *pnTrgType, short *pnExtPin, short *pnMemFullOvW);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinCount	IN	설정할 Pin 개수 (범위 : 전체 AI Pin 개수)
3	pnPinNo	IN	설정할 Pin 번호 short 형 배열 포인터
4	plRate	IN	샘플링 주기 (0~100000 Hz) int 형 배열 포인터
5	pnTrgType	IN	샘플링 트리거 타입 short 형 배열 포인터
6	pnExtPin	IN	연계 External Pin 번호 (범위 : 0 ~ 3) short 형 배열 포인터 (배열 최대크기 : 전체 AI Pin 개수)
7	pnMemFullOvW	IN	메모리 포화상태 시, Overwrite 여부 short 형 배열 포인터 (배열 최대크기 : 전체 AI Pin 개수)

pnTrgType 값:

값	시점	값	시점
0	임의시점	2	Falling
1	Rising	3	Both

pnMemFullOvW 값:

값	설명	값	설명
0	Overwrite	1	Overwrite 안함

주의사항

No	일람
1	지정되는 Pin번호는 『일괄 Pin번호 지정방식』을 참고하세요.
2	배열 최대크기(전체 AI Pin 개수)는 보드들의 전체 AI Pin 개수 입니다.

사용 예

nmf_AIBrdSetRange 항목에 작성된 예시를 참고하여 주십시오.

관련 참조

nmf_AIBrdSetRange, nmf_AISmplStartPin, nmf_AIBrdSmplStartPin, nmf_AIBrdSmplStartPins, nmf_AImBrdSmplStartPins, nmf_AISmplStopPin, nmf_AISmplStopPins, nmf_AIBrdSmplStopPin, nmf_AIBrdSmplStopPins, nmf_AImBrdSmplStopPins, nmf_AISmplExMemSet, nmf_AIBrdSmplExMemSet, nmf_AIBrdSmplExMemGet, nmf_AIBrdSmplValueGetPin

6.3.3. 단일 Pin 정지

nmf_AISmplStopPin	AI

장치에 연결된 모든 AI보드의 Pin 중 하나를 선택하여 Sampling 기능을 정지합니다.

구문

short WINAPI **nmf_AISmplStopPin** (short nNmfNo, short nPinNo);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinNo	IN	설정할 Pin 번호 (범위 : 전체 AI Pin 개수)

주의사항

No	일람
1	지정되는 Pin번호는 『일괄 Pin번호 지정방식』을 참고하세요.

사용 예

nmf_AIBrdSetRange 항목에 작성된 예시를 참고하여 주십시오.

관련 참조

nmf_AIBrdSetRange, nmf_AISmplStartPin, nmf_AISmplStartPins, nmf_AIBrdSmplStartPin, nmf_AIBrdSmplStartPins, nmf_AImBrdSmplStartPins, nmf_AISmplStopPins, nmf_AIBrdSmplStopPin, nmf_AIBrdSmplStopPins, nmf_AImBrdSmplStopPins, nmf_AISmplExMemSet, nmf_AIBrdSmplExMemSet, nmf_AIBrdSmplExMemGet, nmf_AIBrdSmplValueGetPin

6.3.4. 다중 Pin 정지

nmf_AISmplStopPins	AI

장치에 연결된 모든 AI보드의 Pin 중 여러 개를 선택하여 Sampling 기능을 정지합니다.

구문

short WINAPI **nmf_AISmplStopPins** (short nNmfNo, short nPinCount, short *pnPinNo);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinCount	IN	설정할 Pin 수량 (범위 : 전체 AI Pin 개수)
3	pnPinNo	IN	설정할 Pin 번호 short 형 배열 포인터 (범위 : 전체 AI Pin 개수)

주의사항

No	일람
1	지정되는 Pin번호는 『일괄 Pin번호 지정방식』을 참고하세요.
2	배열 최대크기(전체 AI Pin 개수)는 보드들의 전체 AI Pin 개수 입니다.

사용 예

nmf_AIBrdSetRange 항목에 작성된 예시를 참고하여 주십시오.

관련 참조

nmf_AIBrdSetRange, nmf_AISmplStartPin, nmf_AISmplStartPins, nmf_AIBrdSmplStartPin, nmf_AIBrdSmplStartPins, nmf_AImBrdSmplStartPins, nmf_AISmplStopPin, nmf_AIBrdSmplStopPin, nmf_AIBrdSmplStopPins, nmf_AImBrdSmplStopPins, nmf_AISmplExMemSet, nmf_AIBrdSmplExMemSet, nmf_AIBrdSmplExMemGet, nmf_AIBrdSmplValueGetPin

6.4. 보드지정 샘플링 설정

6.4.1. 단일 Pin 시작

nmf_AIBrdSmplStartPin	AI

AI보드를 선택하여 단일 Pin에 대한 Sampling 기능을 설정하고 시작합니다.

구문

short WINAPI **nmf_AIBrdSmplStartPin** (short nNmfNo, short nID, short nPinNo, int lRate, short nTrgType, short nExtPin, short nMemFullOvW);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nID	IN	보드의 ID (범위 : 연결된 AI보드의 ID (0~7))
3	nPinNo	IN	설정할 Pin 번호 (범위 : 0~7 Pin)
4	lRate	IN	샘플링 주기 (0~100000 Hz)
5	nTrgType	IN	샘플링 트리거 타입 값 0 : 임의시점 값 1 : Rising 값 2 : Falling 값 3 : Both
6	nExtPin	IN	연계 External Pin 번호 (범위 : 0 ~ 3)
7	nMemFullOvW	IN	메모리 포화상태 시, Overwrite 여부 값 0 : Overwrite 값 1 : Overwrite 안함

주의사항

No	일람
1	nTrgType 인자의 "임의시점"은 함수명령이 장치로 전달되는 때를 의미합니다. 임의시점으로 선택시 nExtPin 번호는 의미를 갖지 않습니다.

사용 예

nmf_AIBrdSetRange 항목에 작성된 예시를 참고하여 주십시오.

관련 참조

nmf_AIBrdSetRange, nmf_AISmplStartPin, nmf_AISmplStartPins, nmf_AIBrdSmplStartPins, nmf_AImBrdSmplStartPins, nmf_AISmplStopPin, nmf_AISmplStopPins, nmf_AIBrdSmplStopPin, nmf_AIBrdSmplStopPins, nmf_AImBrdSmplStopPins, nmf_AISmplExMemSet, nmf_AIBrdSmplExMemSet, nmf_AIBrdSmplExMemGet, nmf_AIBrdSmplValueGetPin

6.4.2. 다중 Pin 시작

nmf_AIBrdSmplStartPins	AI

AI보드를 선택하여 여러 개의 Pin에 대한 Sampling 기능을 설정하고 시작합니다.

구문

short WINAPI **nmf_AIBrdSmplStartPins** (short nNmfNo, short nID, short nPinCount,
short *pnPinNo, int *plRate, short *pnTrgType, short *pnExtPin,
short *pnMemFullOvW);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nID	IN	보드의 ID (범위 : 연결된 AI보드의 ID (0~7))
3	nPinCount	IN	설정할 Pin 개수 (최대 8개)
4	pnPinNo	IN	설정할 Pin 번호 short 형 배열 포인터 (배열 최대크기 : 8개)
5	plRate	IN	샘플링 주기 (0~100000 Hz) 배열 포인터 (배열 최대크기 : 8개)
6	pnTrgType	IN	샘플링 트리거 타입 배열 포인터 (배열 최대크기 : 8개) 값 0 : 임의시점, 값 1 : Rising, 값 2 : Falling, 값 3 : Both
7	pnExtPin	IN	연계 External Pin 번호 (범위 : 0 ~ 3) 배열 포인터 (배열 최대크기 : 8개)
8	pnMemFullOvW	IN	메모리 포화상태 시, Overwrite 여부 배열 포인터 (배열 최대크기 :8개) 값 0 : Overwrite, 값 1 : Overwrite 안함

주의사항

No	일람
1	배열 최대크기(8)는 입력보드의 Pin수(8) 입니다.
2	연결되어있는 AI보드의 ID를 확인하여 주십시오.

관련 참조

nmf_AIBrdSetRange, nmf_AISmplStartPin, nmf_AISmplStartPins, nmf_AIBrdSmplStartPin, nmf_AImBrdSmplStartPins, nmf_AISmplStopPin, nmf_AISmplStopPins, nmf_AIBrdSmplStopPin, nmf_AIBrdSmplStopPins, nmf_AImBrdSmplStopPins, nmf_AISmplExMemSet, nmf_AIBrdSmplExMemSet, nmf_AIBrdSmplExMemGet, nmf_AIBrdSmplValueGetPin

6.4.3. 다중 보드 시작

nmf_AImBrdSmplStartPins	AI

AI보드와 Pin을 여러 개 선택하여 Sampling 기능을 설정하고 시작합니다.

구문

short WINAPI **nmf_AImBrdSmplStartPins** (short nNmfNo, short nPinCount, short *pnID, short *pnPinNo, int *plRate, short *pnTrgType, short *pnExtPin, short *pnMemFullOvW);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinCount	IN	설정할 Pin 개수 (범위 : 전체 AI Pin 개수)
3	pnID	IN	보드의 ID short 형 배열 포인터 (배열 최대크기 : 64개) (범위 : 연결된 AI보드의 ID (0~7))
4	pnPinNo	IN	설정할 Pin 번호 short 형 배열 포인터 (배열 최대크기 : 64개)
5	plRate	IN	샘플링 주기 (0~100000 Hz) 배열 포인터 (배열 최대크기 : 64개)
6	pnTrgType	IN	샘플링 트리거 타입 배열 포인터 (배열 최대크기 : 64개)

값	시점	값	시점
0	임의시점	2	Falling
1	Rising	3	Both

7	pnExtPin	IN	연계 External Pin 번호 (범위 : 0 ~ 3) 배열 포인터 (배열 최대크기 : 64개)
8	pnMemFullOvW	IN	메모리 포화상태 시, Overwrite 여부 배열 포인터 (배열 최대크기 : 64개)

값	설명	값	설명
0	Overwrite	1	Overwrite 안함

주의사항

No	일람
1	배열 최대크기(64)는 최대 아날로그 입력보드 수량(8) X 각 입력보드의 Pin수(8) 입니다.
2	연결되어있는 AI보드의 ID를 확인하여 주십시오.

관련 참조

nmf_AIBrdSetRange, nmf_AISmplStartPin, nmf_AISmplStartPins, nmf_AIBrdSmplStartPin, nmf_AIBrdSmplStartPins, nmf_AISmplStopPin, nmf_AISmplStopPins, nmf_AIBrdSmplStopPin, nmf_AIBrdSmplStopPins, nmf_AImBrdSmplStopPins, nmf_AISmplExMemSet, nmf_AIBrdSmplExMemSet, nmf_AIBrdSmplExMemGet, nmf_AIBrdSmplValueGetPin

6.4.4. 단일 Pin 정지

nmf_AIBrdSmplStopPin	AI

AI보드를 선택하여 단일 Pin에 대한 Sampling 기능을 정지합니다.

구문

short WINAPI **nmf_AIBrdSmplStopPin** (short nNmfNo, short nID, short nPinNo);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nID	IN	보드의 ID (범위: 연결된 AI보드의 ID (0~7))
3	nPinNo	IN	설정할 Pin 번호 (범위 : 0 ~ 7 Pin)

주의사항

No	일람
1	연결되어있는 AI보드의 ID를 확인하여 주십시오.

사용 예

nmf_AIBrdSetRange 항목에 작성된 예시를 참고하여 주십시오.

관련 참조

nmf_AIBrdSetRange, nmf_AISmplStartPin, nmf_AISmplStartPins, nmf_AIBrdSmplStartPin, nmf_AIBrdSmplStartPins, nmf_AImBrdSmplStartPins, nmf_AISmplStopPin, nmf_AISmplStopPins, nmf_AIBrdSmplStopPins, nmf_AImBrdSmplStopPins, nmf_AISmplExMemSet, nmf_AIBrdSmplExMemSet, nmf_AIBrdSmplExMemGet, nmf_AIBrdSmplValueGetPin

6.4.5. 다중 Pin 정지

nmf_AIBrdSmplStopPins	AI

AI보드를 선택하여 여러 개의 Pin에 대한 Sampling 기능을 정지합니다.

구문

short WINAPI **nmf_AIBrdSmplStopPins** (short nNmfNo, short nID, short nPinCount,
short *pnPinNo);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nID	IN	보드의 ID (범위 : 연결된 AI보드의 ID (0~7))
3	nPinCount	IN	설정할 Pin 수량 (최대 8개)
4	pnPinNo	IN	설정할 Pin 번호 short 형 배열 포인터 (최대 8개)

주의사항

No	일람
1	배열 최대 크기(8)는 입력보드의 Pin수(8) 입니다.

사용 예

nmf_AIBrdSetRange 항목에 작성된 예시를 참고하여 주십시오.

관련 참조

nmf_AIBrdSetRange, nmf_AISmplStartPin, nmf_AISmplStartPins, nmf_AIBrdSmplStartPin, nmf_AIBrdSmplStartPins, nmf_AImBrdSmplStartPins, nmf_AISmplStopPin, nmf_AISmplStopPins, nmf_AIBrdSmplStopPin, nmf_AImBrdSmplStopPins, nmf_AISmplExMemSet, nmf_AIBrdSmplExMemSet, nmf_AIBrdSmplExMemGet, nmf_AIBrdSmplValueGetPin

6.4.6. 다중 보드 정지

nmf_AImBrdSmplStopPins	AI

AI보드와 Pin을 여러 개 선택하여 Sampling 기능을 정지합니다.

구문

short WINAPI **nmf_AImBrdSmplStopPins** (short nNmfNo, short nPinCount, short *pnID, short *pnPinNo);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinCount	IN	설정할 Pin 수량
3	pnID	IN	보드의 ID short 형 배열 포인터 범위: 0 ~ 7 최대 수량: 64
4	pnPinNo	IN	설정할 Pin 번호 short 형 배열 포인터 (최대 64개) (크기 : 선택보드 수량 X nPinCount)

주의사항

No	일람
1	배열 최대 크기(64)는 최대 아날로그 입력보드 수량(8) X 각 입력보드의 Pin수(8) 입니다

사용 예

nmf_AImBrdSetPins 항목에 작성된 예시를 참고하여 주십시오.

관련 참조

nmf_AIBrdSetRange, nmf_AISmplStartPin, nmf_AISmplStartPins, nmf_AIBrdSmplStartPin, nmf_AIBrdSmplStartPins, nmf_AImBrdSmplStartPins, nmf_AISmplStopPin, nmf_AISmplStopPins, nmf_AIBrdSmplStopPin, nmf_AIBrdSmplStopPins, nmf_AISmplExMemSet, nmf_AIBrdSmplExMemSet, nmf_AIBrdSmplExMemGet, nmf_AIBrdSmplValueGetPin

6.5. Sampling Rate

아날로그 입력보드의 입력 설정 시, 반영되는 Sampling Rate 값은 제어 적용을 위한 정수연산으로 인하여, 실제 입력값과는 다른 값이 제어에 반영될 수 있습니다. 그로인해 확인 시, 입력한 값과 편차가 발생할 수 있으며, 장치에 적용되는 실제 Sampling Rate를 확인하여, 입력값의 변동 편차를 예측할 수 있습니다.

6.5.1. 단일 Pin 확인

nmf_AISmplRateGetPin	AI

AI보드의 단일 입력Pin을 선택하여 실제 Sampling Rate를 확인합니다.

구문

short WINAPI **nmf_AISmplRateGetPin**(short nNmfNo, short nPinNo, int *plRetSmplRate);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinNo	IN	설정할 Pin 번호 (범위 : 전체 AI Pin 개수)
3	plRetSmplRate	OUT	Sampling Rate를 읽어올 int 형 포인터 변수

주의사항

No	일람
1	지정되는 Pin번호는 『일괄 Pin번호 지정방식』을 참고하세요.

사용 예

```
#define MAX_BRD             8   // 최대 보드 수량 (CPU 장치 하나 당)
#define BRD_VS_AI_PINS      8   // AI 보드의 핀 개수

short   nRet;
short   nNmfNo = 11;    // 장치번호
short   nBrdID = 1;     // 보드 ID
short   nPinNo = 1;     // Pin 번호

/*   장치가 연결된 상태 입니다.   */

// AI 보드의 모든 Pin에 Sampling Rate를 다르게 설정
short   nPinCount = 8;
short   pnPinList   [BRD_VS_AI_PINS] = {0, 1, 2, 3, 4, 5, 6, 7};
int     plRate      [BRD_VS_AI_PINS] = {0, 500, 1000, 2000, 20000, 50000, 99000, 100000};
short   pnTrgType   [BRD_VS_AI_PINS] = {0};         // Trigger 사용 안함.
short   pnTrgPin    [BRD_VS_AI_PINS] = {0};         // Trigger 사용 안함.
short   pnMemFullOvW [BRD_VS_AI_PINS] = {0};        // Overwrite 수행
nRet = nmf_AIBrdSmplStartPins(nNmfNo, nPinCount, nBrdID, pnPinList, plRate, pnTrgType,
                              pnTrgPin, pnMemFullOvW);
if (nRet != nmf_R_OK)       return;

// 각 Pin에 실제 적용된 Sampling Rate 값을 확인
```

```
// 단일보드에 대한 전체 Pin 값 확인
int lRetSmplRate[BRD_VS_AI_PINS] = {0};
nRet = nmf_AIBrdSmplRateGetPins(nNmfNo, nBrdID, nPinCount, pnPinList, lRetSmplRate);
if (nRet != nmf_R_OK)        return;            // 값 확인 실패

for (int i = 0; i < BRD_VS_AI_PINS; i++)
{
    printf("%d 번 Pin에 적용된 실제 SmplRate : %d", i, lRetSmplRate[i]);
    // 예로 적용된 99,000 Hz를 SamplingRate로 설정하게 되면,
    // 내부 정수 연산으로 인하여 99,010 Hz가 실제 적용되는 값이 됩니다.
    // 따라서, lRetSmplRate[6] 의 값은 99010으로 반환됩니다.
}
```

관련 참조

nmf_AISmplRateGetPins, nmf_AIBrdSmplRateGetPin, nmf_AIBrdSmplRateGetPins, nmf_AImBrdSmplRateGetPins , nmf_AIBrdSmplValueGetPin

6.5.2. 다중 Pin 확인

nmf_AISmplRateGetPins	AI

AI보드의 입력Pin을 여러 개 선택하여 실제 Sampling Rate를 확인합니다.

구문

```
short WINAPI nmf_AISmplRateGetPins(short nNmfNo, short nPinCount,
                short *pnPinNo, int *plRetSmplRate);
```

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinCount	IN	설정할 Pin 개수 (범위 : 전체 AI Pin 개수)
3	pnPinNo	IN	설정할 Pin 번호 short 형 배열 포인터
4	plRetSmplRate	OUT	Sampling Rate를 읽어올 int 형 배열 포인터 (크기 : nPinCount)

주의사항

No	일람
1	지정되는 Pin번호는 『일괄 Pin번호 지정방식』을 참고하세요.

사용 예

nmf_AISmplRateGetPin 항목에 작성된 예시를 참고하여 주십시오.

관련 참조

nmf_AISmplRateGetPin, nmf_AIBrdSmplRateGetPin, nmf_AIBrdSmplRateGetPins, nmf_AImBrdSmplRateGetPins, nmf_AIBrdSmplValueGetPin

6.5.3. 단일보드 Pin 확인

nmf_AIBrdSmplRateGetPin	AI

AI보드를 선택하고 단일 Pin의 실제 Sampling Rate를 확인합니다.

구문

short WINAPI **nmf_AIBrdSmplRateGetPin**(short nNmfNo, short nID, short nPinNo,
int *plRetSmplRate);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nID	IN	보드의 ID (범위: 연결된 AI보드의 ID (0~7))
3	nPinNo	IN	설정할 Pin 번호 (범위 : 0~7 Pin)
4	plRetSmplRate	OUT	Sampling Rate를 읽어올 int 형 포인터 변수

주의사항

No	일람
1	연결되어있는 AI보드의 ID를 확인하여 주십시오.

사용 예

nmf_AISmplRateGetPin 항목에 작성된 예시를 참고하여 주십시오.

관련 참조

nmf_AISmplRateGetPin, nmf_AISmplRateGetPins, nmf_AIBrdSmplRateGetPins, nmf_AImBrdSmplRateGetPins, nmf_AIBrdSmplValueGetPin

6.5.4. 단일보드 다중Pin 확인

nmf_AIBrdSmplRateGetPins	AI

AI보드를 선택하고 여러 Pin의 실제 Sampling Rate를 확인합니다.

구문

short WINAPI **nmf_AIBrdSmplRateGetPins**(short nNmfNo, short nID, short nPinCount,
short *pnPinNo, int *plRetSmplRate);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nID	IN	보드의 ID (범위 : 연결된 AI보드의 ID (0~7))
3	nPinCount	IN	설정할 Pin 수량 (최대 8개)
4	pnPinNo	IN	설정할 Pin 번호 short 형 배열 포인터 (최대 8개)
5	plRetSmplRate	OUT	Sampling Rate를 읽어올 int 형 배열 포인터 (크기 : nPinCount)

주의사항

No	일람
1	배열 최대크기(8)는 입력보드의 Pin수(8) 입니다.
2	연결되어있는 AI보드의 ID를 확인하여 주십시오.

사용 예

nmf_AISmplRateGetPin 항목에 작성된 예시를 참고하여 주십시오.

관련 참조

nmf_AISmplRateGetPin, nmf_AISmplRateGetPins, nmf_AIBrdSmplRateGetPin, nmf_AImBrdSmplRateGetPins, nmf_AIBrdSmplValueGetPin

6.5.5. 다중보드 다중Pin 확인

nmf_AImBrdSmplRateGetPins	AI

AI보드와 Pin을 여러 개 선택하여 실제 Sampling Rate를 확입합니다.

구문

short WINAPI **nmf_AImBrdSmplRateGetPins**(short nNmfNo, short nPinCount,
short *pnID, short *pnPinNo, int *plRetSmplRate);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinCount	IN	설정할 Pin 수량 (최대 64개)
3	pnID	IN	보드의 ID short 형 배열 포인터 범위: 0 ~ 7 최대 수량: 64
4	pnPinNo	IN	설정할 Pin 번호 short 형 배열 포인터 (최대 64개)
5	plRetSmplRate	OUT	Sampling Rate를 읽어올 int 형 배열 포인터 (크기 : nPinCount) (최대 64개)

주의사항

No	일람
1	배열 최대크기(64)는 최대 아날로그 입력보드 수량(8) X 각 입력보드의 Pin수(8) 입니다.

사용 예

nmf_AISmplRateGetPin 항목에 작성된 예시를 참고하여 주십시오.

관련 참조

nmf_AISmplRateGetPin, nmf_AISmplRateGetPins, nmf_AIBrdSmplRateGetPin, nmf_AIBrdSmplRateGetPins, nmf_AIBrdSmplValueGetPin

6.6. 외부 메모리 설정

6.6.1. 전체 설정

nmf_AISmplExMemSet	AI

장치에 연결된 모든 AI보드의 Pin들 중 여러 개를 선택하여 외부 메모리 사용을 설정합니다.

구문

short WINAPI **nmf_AISmplExMemSet** (short nNmfNo, short nPinCount,
short *pnPinNo, short *pnMemBCount);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinCount	IN	설정할 Pin 수량 (Brd 8개, 각 8 Pin으로 최대 64개)
3	pnPinNo	IN	설정할 Pin 번호 short 형 배열 포인터 (배열 최대크기 : 64개) (범위 : 0~7 Pin)
4	pnMemBCount	IN	Pin에 할당되는 외부 Memory Block의 개수 배열 포인터 (0 ~ 64 Block) :: 1Block = 2048개(4KByte) 0 설정 시, 외부 Memory 사용 안함

주의사항

No	일람
1	지정되는 Pin번호는 『일괄 Pin번호 지정방식』을 참고하세요.
2	메모리 블록을 0으로 설정하게 되면 외부 메모리를 사용하지 않습니다.

사용 예

nmf_AIBrdSmplValueGetPin 항목에 작성된 예시를 참고하여 주십시오.

관련 참조

nmf_AIBrdSetRange, nmf_AISmplStartPin, nmf_AISmplStartPins, nmf_AIBrdSmplStartPin, nmf_AIBrdSmplStartPins, nmf_AImBrdSmplStartPins, nmf_AISmplStopPin, nmf_AISmplStopPins, nmf_AIBrdSmplStopPin, nmf_AIBrdSmplStopPins, nmf_AImBrdSmplStopPins, nmf_AIBrdSmplExMemSet, nmf_AIBrdSmplExMemGet, nmf_AIBrdSmplValueGetPin

6.6.2. 보드지정 설정

nmf_AIBrdSmplExMemSet	AI

AI보드와 Pin번호를 개별 선택하여 외부 메모리 사용을 설정합니다.

구문

short WINAPI **nmf_AIBrdSmplExMemSet** (short nNmfNo, short nPinCount, short *pnID, short *pnPinNo, short *pnMemBCount);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nPinCount	IN	설정할 Pin 수량 (Brd 8개, 각 8 Pin으로 최대 64)
3	pnID	IN	보드의 ID short 형 배열 포인터 (배열 최대크기 : 64개) (범위 : 연결된 AO보드의 ID (0~7))
4	pnPinNo	IN	설정할 Pin 번호 short 형 배열 포인터 (배열 최대크기 : 64개) (범위 : 0~7 Pin)
5	pnMemBCount	IN	Pin에 할당되는 외부 Memory Block의 개수 배열 포인터 (0 ~ 64 Block) :: 1Block = 2048개(4KByte) 0 설정 시 외부 Memory 사용 안함

주의사항

No	일람
1	메모리 블록을 0으로 설정하게 되면 외부 메모리를 사용하지 않습니다.

사용 예

nmf_AIBrdSmplValueGetPin 항목에 작성된 예시를 참고하여 주십시오.

관련 참조

nmf_AIBrdSetRange, nmf_AISmplStartPin, nmf_AISmplStartPins, nmf_AIBrdSmplStartPin, nmf_AIBrdSmplStartPins, nmf_AImBrdSmplStartPins, nmf_AISmplStopPin, nmf_AISmplStopPins, nmf_AIBrdSmplStopPin, nmf_AIBrdSmplStopPins, nmf_AImBrdSmplStopPins, nmf_AISmplExMemSet, nmf_AIBrdSmplExMemGet, nmf_AIBrdSmplValueGetPin

6.6.3. 보드지정 설정 확인

nmf_AIBrdSmplExMemGet	AI

AI보드의 외부 메모리 설정상태를 확인합니다.

내부 메모리를 사용하는 Pin은 처리되지 않으며, 외부 메모리를 사용하는 Pin에 대해서 확인됩니다.

구문

short WINAPI **nmf_AIBrdSmplExMemGet** (short nNmfNo, short *pnRetPinCount,
short *pnRetID, short *pnRetPinNo,
short *pnRetMemBCount, short *pnRetMemBFill);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	pnRetPinCount	OUT	외부 메모리가 할당된 Pin의 개수를 받을 포인터 변수
3	pnRetID	OUT	할당된 보드의 ID short 형 배열 포인터 (배열 최대크기 : 64개)
4	pnRetPinNo	OUT	설정할 Pin 번호 short 형 배열 포인터 (배열 최대크기 : 64개)
5	pnRetMemBCount	OUT	각 Pin에 설정된 외부 메모리의 Block 수량(0~64 block)을 받을 short 형 배열 포인터 (배열 최대크기 : 64개)
6	pnRetMemBFill	OUT	각 Pin에 설정된 외부 메모리 Block 중 데이터가 채워진 Block 수량을 받을 short 형 배열 포인터 (배열 최대크기 : 64개)

주의사항

No	일람
1	배열 최대크기(64)는 최대 아날로그 입력보드 수량(8) X 각 입력보드의 Pin수(8) 입니다.
2	pnRetPinCount 의 범위를 넘겨 입력받는 배열의 값은 의미를 갖지 않습니다.

사용 예

nmf_AIBrdSmplValueGetPin 항목에 작성된 예시를 참고하여 주십시오.

관련 참조

nmf_AIBrdSetRange, nmf_AISmplStartPin, nmf_AISmplStartPins, nmf_AIBrdSmplStartPin, nmf_AIBrdSmplStartPins, nmf_AImBrdSmplStartPins, nmf_AISmplStopPin, nmf_AISmplStopPins, nmf_AIBrdSmplStopPin, nmf_AIBrdSmplStopPins, nmf_AImBrdSmplStopPins, nmf_AISmplExMemSet, nmf_AIBrdSmplExMemSet, nmf_AIBrdSmplValueGetPin

6.7. 데이터 확인

AI 보드에 입력설정 및 메모리를 할당하여 내부에 저장된 샘플링 데이터를 확인합니다.
메모리 할당크기에 따라 확인될 수 있는 데이터의 수량은 달라집니다.

6.7.1. 샘플링 데이터

nmf_AIBrdSmplValueGetPin	AI

AI 보드를 통해 입력받은 샘플링 데이터를 확인합니다..

구문

short WINAPI **nmf_AIBrdSmplValueGetPin** (short nNmfNo, short nID, short nPinNo,
short nType, PTNMFBRDAISMPL ptAISmpl, int lBufNum, double *pdCalcValue);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nID	IN	보드의 ID (범위: 연결된 AI 보드의 ID (0~7))
3	nPinNo	IN	확인할 Pin 번호 (범위 : 0 ~ 7 Pin)
4	nType	IN	확인할 아날로그 데이터 설정값 / 확인 데이터 0 / 전압 (V) 1 / AD Value 2 / 전류 (A)
5	ptAISmpl	OUT	AI 샘플링 정보 구조체 포인터 자세한 설명은 TNMF_BRD_AI_SPL 구조체 참조
6	lBufNum	IN	읽어올 데이터 버퍼 개수 (0으로 설정하는 경우 전체 가져오기)
7	pdCalcValue	OUT	AI 샘플링 데이터을 받을 double 형 배열 포인터 AI 보드 단일 Pin 내부 채널메모리 = 2048 개 외부메모리 할당 시, 2048개 x 64Block = 131072 개 (최대 배열 수량 133120개) ※ 외부메모리 할당 수량에 따라 배열크기가 변동됩니다.

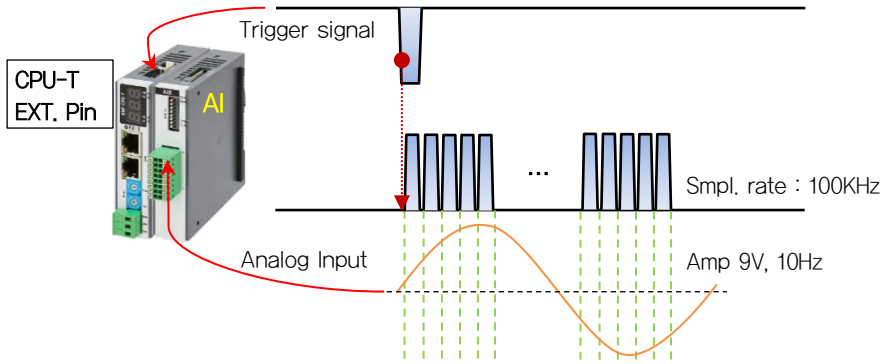
주의사항

No	일람
1	lBufNum 값을 0으로 설정 시, 수량 제한없이 전체 값을 확인합니다.
2	샘플링에 할당된 외부메모리의 크기에 따라 확인가능한 데이터 수량이 달라지므로, 설정에 유의하여 주십시오.
3	원형큐의 데이터관리 방식으로 인해 할당된 저장크기보다 하나 적게 응답됩니다.

사용 예

```
#define MAX_BRD                 8   // 최대 보드 수량 (CPU 장치 하나 당)
#define BRD_VS_AI_PINS          8   // AI 보드의 핀 개수

short   nRet;
short   nNmfNo = 11;    // 장치번호
short   nBrdID = 1;     // 보드 ID
short   nPinNo = 0;     // AI 보드의 Pin번호
/*   장치가 연결된 상태 입니다.   */
```



CPU-T 의 External Pin을 통한 Trigger 신호를 받아 입력값을 SampiIng 하여 그 변화를 확인

```
// 외부메모리 사용을 설정하여, 저장공간을 할당함
short   nPinCnt = 1;
short   pnID[1] = nBrdID;
short   pnPinNo[1] = nPinNo;
short   pnBlock[1] = {64};   // 64 Block을 설정하는 경우 2048(외부) X 64 + 2048(내부) 의 크기를 가짐.
nRet = nmf_AIBrdSmplExMemSet (nNmfNo, nPinCnt, pnID, pnPinNo, pnBlock);

// 입력 Pin 0번에 주기 100KHz, Ext. Pin을 통한 Trigger 신호(Falling)를 이용하여
// 메모리가 포화될 때까지 (Overwrite 안함) Sampling 설정
short   nTrgMode = 2;       // 1=Trg Rising, 2=Trg Falling, 3=Trg Both
short   nEPinNo = 0;        // 연동할 Ext. Pin번호
short   nMemFullOvW = 0;  // Overwrite 하지 않고, 포화상태 유지
nRet = nmf_AIBrdSmplStartPin(nNmfNo, nBrdID, nPinNo, 100000, nTrgMode, nEPinNo, nMemFullOvW);

// 아날로그 신호 입력 중 // Trigger 신호가 Ext Pin으로 입력되면 샘플링을 수행함.

// 샘플링 상태 확인
TNMF_ALL_STS  tStatus;
nRet = nmf_GetBrdAllStatus(nNmfNo, &tStatus);
if (tStatus.tAI[nBrdID].nStatus == 1)          printf("AI 샘플링 수행중");
if (tStatus.tAI[nBrdID].nMemExists == 1)       printf("AI 샘플링 데이터가 메모리에 존재함");
if (tStatus.tAI[nBrdID].nMemFullFlag == 1)     printf("메모리가 포화상태");

// 샘플링 데이터 포화상태가 확인되어 저장된 데이터 확인
TNMF_BRD_AI_SMPL    tSmplData;
short   nType = 0;      // 0=전압, 1=AD Value, 2=전류
int     lBufNum = 0;    // 전체확인
double  pdValue[133120] = {0};   // 샘플링 데이터를 받을 배열 (최대크기 : 133120)
nRet = nmf_AIBrdSmplValueGetPin(nNmfNo, nBrdID, nPinNo, nType, &tSmplData, lBufNum, pdValue);
if (nRet != nmf_R_OK)       return;  // 샘플링 데이터 확인에 실패하였습니다.

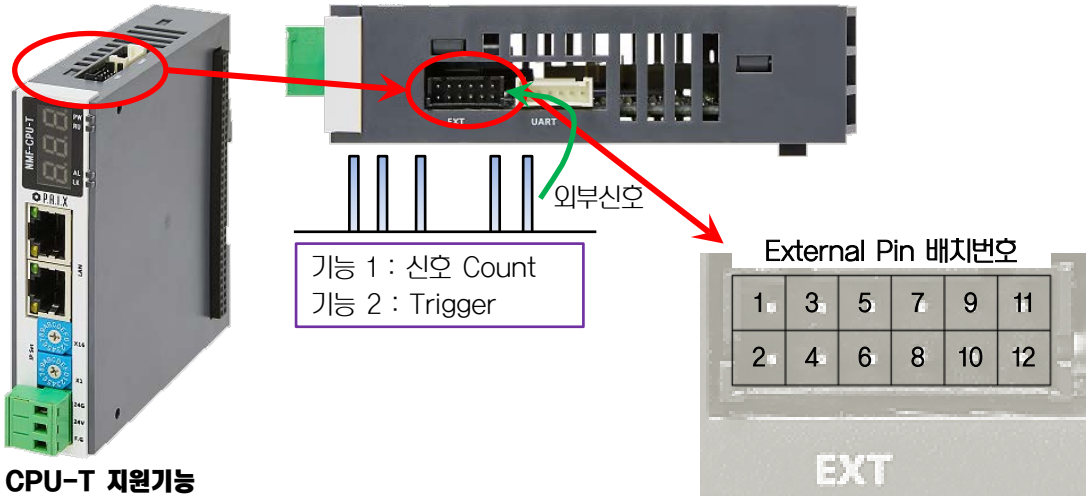
printf("응답수량 : %d", tSmplData.lRespSize);  // 응답된 수량을 확인 (원형큐 형태로 인한 -1 개)
for (int i = 0; i < tSmplData.lRespSize; i++)
{
    printf("Smpl Data %d : %lf", i, pdValue[i]);    // Sampling Data 확인 // 약 1.3s 의 입력변화가 확인됨.
}
```

관련 참조

nmf_AIBrdSetRange, nmf_AISmplStartPin, nmf_AISmplStartPins, nmf_AIBrdSmplStartPin, nmf_AIBrdSmplStartPins, nmf_AImBrdSmplStartPins, nmf_AISmplStopPin, nmf_AISmplStopPins, nmf_AIBrdSmplStopPin, nmf_AIBrdSmplStopPins, nmf_AImBrdSmplStopPins, nmf_AISmplExMemSet, nmf_AIBrdSmplExMemSet, nmf_AIBrdSmplExMemGet, nmf_AISmplRateGetPin, nmf_AISmplRateGetPins, nmf_AIBrdSmplRateGetPin, nmf_AIBrdSmplRateGetPins, nmf_AImBrdSmplRateGetPins

7.External Pin

CPU-T 보드에 적용된 External Pin을 사용하여 외부 입력신호를 받아 특정 동작을 수행할 수 있습니다.
입력신호의 수량을 확인(Count)하거나, AI 보드를 사용할 때 Trigger 신호로 이용하여 샘플링의 시작신호로 사용할 수 있습니다.



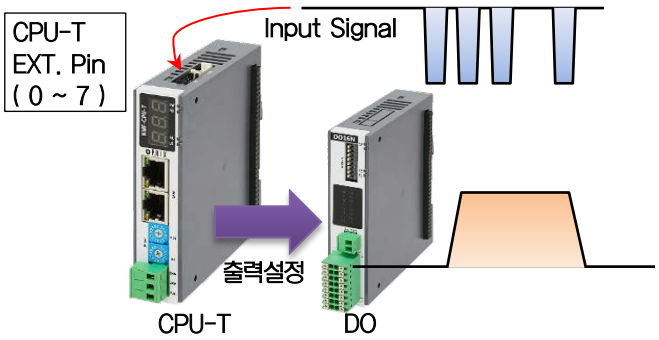
CPU-T 지원기능

External Pin 에서 Count와 Trigger가 적용되는 부분은 다음과 같습니다.

No	기능	Count	Trigger	No	기능	Count	Trigger
1	Ext Pin 0	O	O	2	Ext Pin 1	O	O
3	Ext Pin 2	O	O	4	Ext Pin 3	O	O
5	COM1-5V			6	COM1-24V		
7	Ext Pin 4	O	X	8	Ext Pin 5	O	X
9	Ext Pin 6	O	X	10	Ext Pin 7	O	X
11	COM2-5V			12	COM2-24V		

하드웨어 적인 배치번호와는 다르게 본 문의 External Pin 번호는 0번에서 7번으로 사용합니다.
별도의 동작에 대한 상세스펙은 하드웨어 매뉴얼을 참고하여 주십시오.

7.1. Count



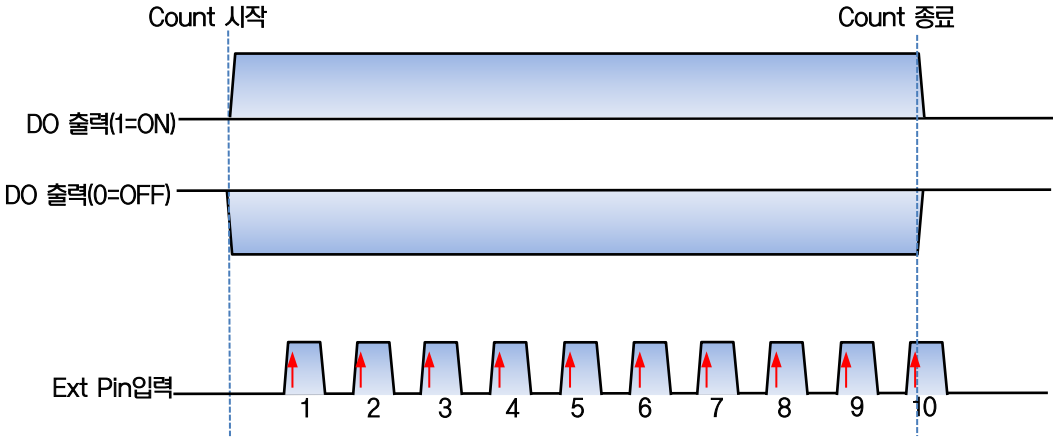
Count는 CPU-T 보드에서 사용 가능한 External Pin을 사용하여 외부 신호를 입력받아, 그 횟수를 확인하는 기능입니다. 각 External Pin으로 입력받은 신호의 지정된 기준에 따라 횟수가 증가되며, 시작과 종료 시점에 DO보드를 연계하여 On/Off 기능을 구성할 수 있습니다.

관련함수

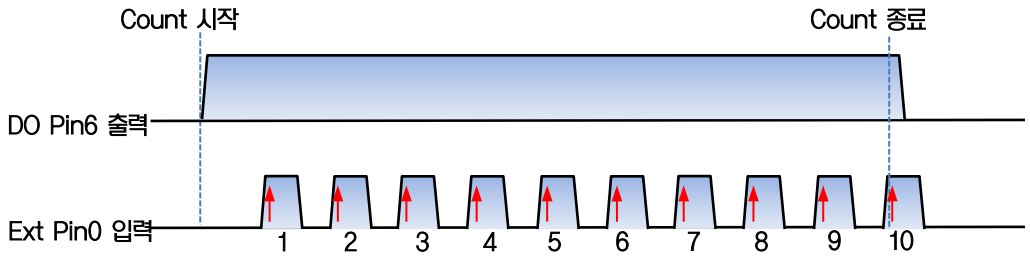
No	명령	설명
1	nmf_ExtCountStart	External Pin을 이용한 Count를 시작합니다.
2	nmf_ExtCountStop	설정된 Count 기능의 동작을 정지합니다.
3	nmf_ExtCountGetInfo	현재 Count 기능의 정보를 확인합니다.
4	nmf_ExtCountGetTime	시간측정 모드의 경우, 측정된 시간값을 확인합니다.
5	nmf_ExtCountFilterSet	Count 입력의 필터를 설정합니다.
6	nmf_ExtCountFilterGet	설정된 필터를 확인합니다.

7.1.1. 알아두기

Count 기능은 앞서 설명한 내용과 같이, 외부 신호의 입력횟수를 확인하는 기능으로써 DO보드의 출력을 함께 연계하여 사용할 수 있습니다.



예를 들어, 목표 Count를 10개 설정하고, DO 6번 Pin을 출력ON 하고, External Pin 0번을 통해 Rising Edge 신호 10개를 확인하여, 다시 DO 6번 Pin의 출력을 OFF 하는 경우, 아래 그림과 같은 동작을 확인할 수 있습니다.



실제 DO출력의 반전은 동시에 되는 것이 아닌 Edge 신호의 처리시간 이후 동작하게 됩니다.

7.1.1.1. 동작구성

Count는 아래의 동작으로 구성됩니다.



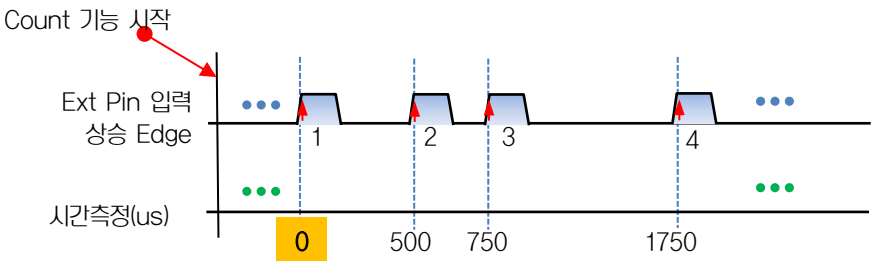
External Pin 0~2번은 최대 2KHz(약 0.5ms)의 저속으로 동작되며, 3~7번은 최대 4MHz(약 0.25us)의 고속동작을 지원합니다. 각 Pin의 입력확인은 개별 동작속도에 의해 수행되며, 고속동작이 지원되는 3~7번 Pin에서는 필터를 적용하여 상황에 따른 노이즈 필터링이 가능합니다.

이후 사용자의 확인 요구시점(데이터 확인 함수 사용)에 입력 횟수(32bit)와 Timer 동작시간(64bit)을 획득하게 됩니다.

7.1.1.2. 추가기능

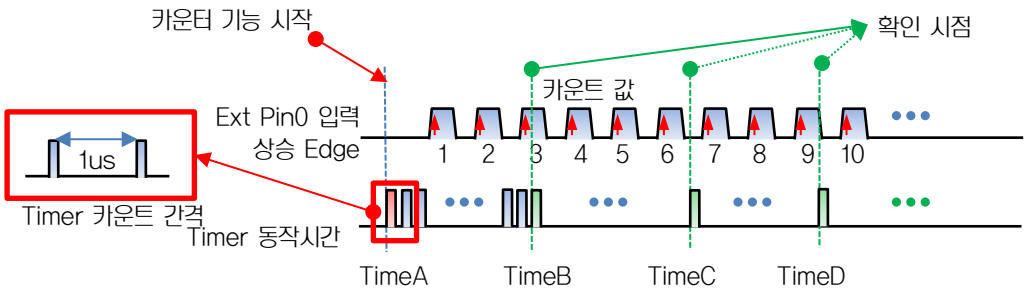
(1) 카운트 당 시간 측정 모드(Time Stamp)

일반적인 입력횟수를 확인하는 기능 외에 추가적으로, Count 기능 시작시, 설정하는 Edge(Rising, Falling, Both)를 기준으로 하여, External Pin에서 입력이 이루어질 때마다 해당 시간을 측정할 수 있습니다.
기능 시작 후, 처음으로 기준 Edge가 인식된 시점을 0으로 설정하며, 측정된 시간 데이터를 FIFO메모리를 통해(최대 1,024개) 읽을 수 있는 기능입니다.
시간 측정 모드로 적용시 모든 External Pin은 저속 동작하게 되며, 측정간격(Interval)에 따라 입력시간을 측정하게 됩니다. 측정 간격에 대한 조정이 필요하시다면 당사로 문의하여 주십시오.



(2) 변위량 확인

입력횟수와 Timer 동작시간을 확인하고 그 값을 이용하여 속도와 같은 단위시간 당 변위량을 구할 수 있습니다.



그림과 같이 카운트 획득 시점에 따라 확인된 값을 이용하여 구간별 속도(pulse/s)를 확인할 수 있습니다.
(Timer의 카운트 간격은 1us 단위로 속도의 Sec로 맞추기 위해 1000000의 역수를 계산)

```
TimeB – TimeA구간속도 = (카운트 값(TimeB) – 카운트 값(TimeA)) / ((TimeB – TimeA) / 1000000)
TimeC – TimeB구간속도 = (카운트 값(TimeC) – 카운트 값(TimeB)) / ((TimeC – TimeB) / 1000000)
TimeD – TimeC구간속도 = (카운트 값(TimeD) – 카운트 값(TimeC)) / ((TimeD – TimeC) / 1000000)
```

사용 예

```
short nRet;
short nNmfNo = 11;                  // 장치번호
double dSecConv = 1000000;   // uSec 단위를 Sec 단위로 계산해 주기 위한 값 (역수로 계산)

// 장치와의 네트워크 연결 상태를 확인합니다.
nRet = nmf_PingCheck(nNmfNo, 192, 168, 0, 200);
if ( nRet != nmf_R_OK )  return;      // 네트워크 연결 상태를 확인할 수 없습니다.
// 장치와 연결을 시도합니다.
nRet = nmf_Connect(nNmfNo, 192, 168, 0);
if ( nRet != nmf_R_OK )  return;      // 장치 연결 실패

TNMF_ALL_STS  tAllStatus;
long lPreCount;                          // Previous 카운트
long long llPreTime;                   // Previous 시간
short nSpeed;                            // 변위량(속도 pps)
```

```
while(1)
{
 :
// 일정한 시간 간격으로 반복 확인
nRet = nmf_GetBrdAllStatus(nNmfNo, &tAllStatus);
// Ext 4번 Pin 카운트 및 측정 시간으로 변위량(속도 = 현재 카운트 - 지난 카운트/(현재 시간 - 지난 시간)) 계산
nSpeed = tAllStatus.dwCountValue[4] - lPreCount / ((tAllStatus.ddwCountTime[4] - llPreTime) / dSecConv);

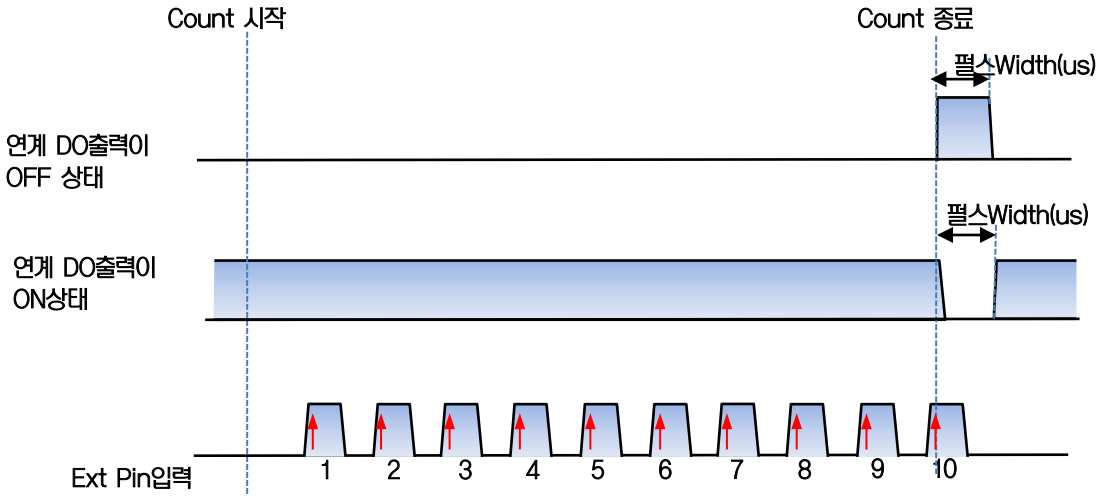
printf("Ext Pin4 속도 = %dpps", nSpeed);

// 변위량(속도)를 계산하기 위해 카운트 및 측정 시간을  Previous변수에 기입
lPreCount = tAllStatus.dwCountValue;
llPreTime = tAllStatus.ddwCountTime;
 :
}

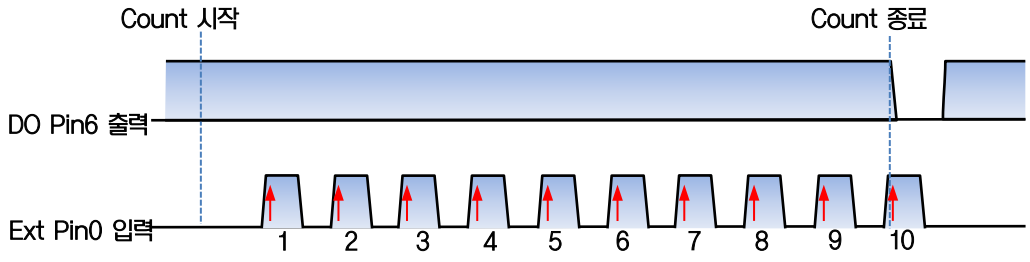
// 장치 연결을 해제합니다.
nmf_Disconnect(nNmfNo);
```

(3) DO 상태 Pulse 기능

nmf_ExtCountStartEx에서 추가된 기능으로 DO 연계 기능 중 DO 상태를 Pulse로 설정할 경우입니다. 구동 방식은 Count시작에 관계 없이 연계 DO의 출력의 상태를 유지하며, Count 종료 시 설정된 펄스Width(us)만큼 반전 후 원상태로 됩니다.



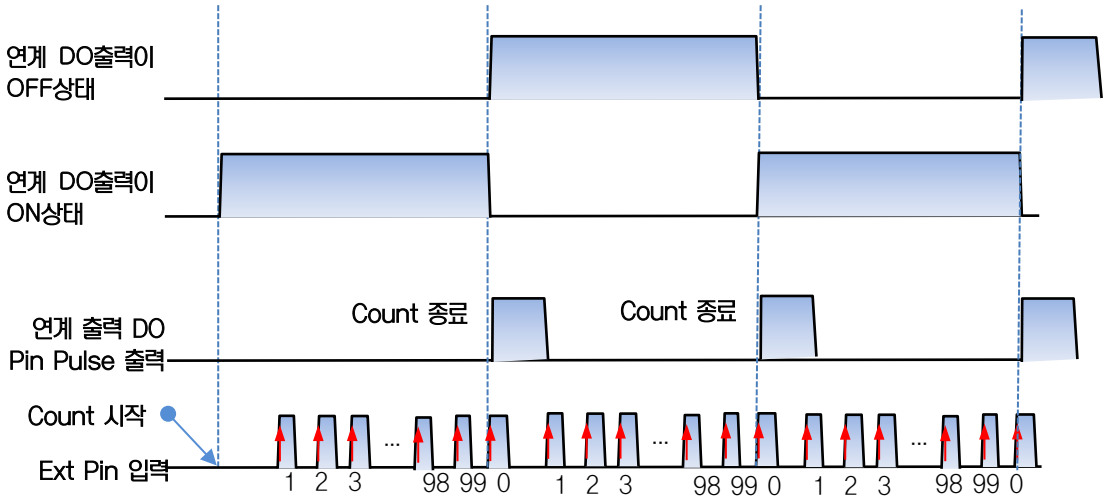
예를 들어, DO 6번 Pin은 On 상태고, 목표 Count를 10개 설정하고, External Pin 0번을 통해 Rising Edge 신호 10개를 확인하여, DO 6번이 Pulse 출력이 발생하는 경우, 아래 그림과 같은 동작을 확인할 수 있습니다.



(4) Count 반복 모드

현재 Count가 목표 Count에 다다를 경우 Count 기능을 완료하지 않으며, 현재 Count가 Reset 되어 0부터 다시 Count를 하는 반복기능입니다.

예로 목표 Count가 100일 경우 100에 다다를 경우 0으로 Reset이 되어 다시 Count를 합니다.



7.1.2. 기능 시작

nmf_ExtCountStart	CPU-T

외부 입력 신호의 Count 기능을 시작합니다.

구문

short WINAPI **nmf_ExtCountStart** (short nNmfNo, short nEPinNo, short nAddMode,
short nEdge, short nDO_ID, short nDO_PinNo, short nDO_On,
unsigned int dwCount);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nEPinNo	IN	Count 기능의 External Pin 번호 (범위: 0 ~ 7)
3	nAddMode	IN	추가 기능을 BitMask로 설정합니다. Bit설정 / 내 용 0 : None Bit 0 : 시간 측정 모드 수행여부 기능 동작 여부는 nmf_ExtCountGetTime 함수를 참고하여 주십시오
4	nEdge	IN	Count 기준 Edge설정 값 / Count 기준 Edge 0 : 상승 1 : 하강 2 : 상승+하강 
5	nDO_ID	IN	연계할 DO보드 ID (범위 : nDO_PinNo의 설명을 참조)
6	nDO_PinNo	IN	기능과 연계하여 사용할 DO보드의 Pin 번호 보드ID (nDO_ID) / Pin 번호 선택범위 0 보다 작은경우 : 연계기능 사용안함 0 ~ 7 : 각 출력보드의 Pin번호 (0~7) 7보다 큰 경우 : [일괄 Pin번호 지정방식] 참고 ※ Pin번호가 -1이면 DO연계 기능 사용하지 않습니다. ※ Pin번호를 일괄적으로 지정하는 경우, **적용가능한 Pin번호의 범위는 연결된 출력보드 수 X Pin수량(16)** 입니다.
7	nDO_On	IN	명령 전달과 동시에 연계된 DO Pin에 설정할 출력상태입니다. 목표 Count값(dwCount)에 도달시 출력상태는 반전됩니다. 값 / 출력 상태 0 : 출력 OFF 1 : 출력 ON
8	dwCount	IN	목표 Count 수량을 입력합니다. (1 ~ 4294967295(32Bit)) 목표수량에 도달 시 Count 기능은 정지합니다. 0을 입력하는 경우, 무제한으로 동작합니다.

주의사항

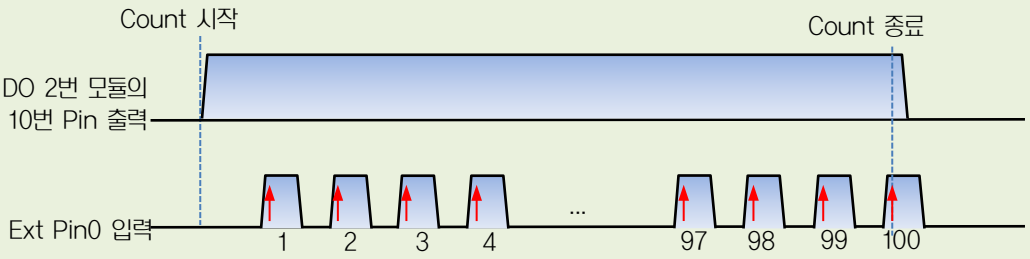
No	일람
1	nDO_PinNo 인자가 0보다 작은(-1)값을 갖는 경우 디지털 출력제어와 연계되는 기능을 사용하지 않습니다. 즉, 명령 시점으로부터 External Pin 입력신호의 수량을 Count하는 기능으로만 사용됩니다. 이 경우, nDO_ID, nDO_On 인자값 또한 의미를 갖지 않으며, 동작되지 않습니다.
2	dwCount 인자로 설정된 목표 수량에 도달 시, Count 기능은 정지합니다.
3	Count 동작시간을 0으로 초기화합니다.

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호

/*   장치가 연결된 상태 입니다.   */

short nEPinNo = 1;          // Count를 적용할 External Pin 번호
short nEdge = 0;            // Count 적용 시점 Rising
short nAddMode = 0x01;          // 시간측정 모드 사용
short nDO_ID = 2;           // 기능을 연계할 DO보드의 ID
short nDO_PinNo = 10;       // 기능을 연계할 DO보드의 PinNo
short nDO_On = 1;           // 시작될 때의 DO Pin출력 ON
uint dwCount = 100;         // 목표 수량 (목표수량 도달 시, 연계된 DO 출력은 반전됩니다.)
nRet = nmf_ExtCountStart(nNmfNo, nEPinNo, nAddMode, nEdge,
                         nDO_ID, nDO_PinNo, nDO_On, dwCount);
if ( nRet != nmf_R_OK )      return;       // Count 기능 설정 실패
```




```
// External Pin에 설정된 Count 기능의 상태정보
TNMF_EXTC_INFO tExtCntInfo;

// Count된 횟수 확인
nRet = nmf_ExtCountGetInfo(nNmfNo, &tExtCntInfo);
printf("Ext Pin 0번의 Count 동작여부 : %d", tExtCntInfo.nEnable[nEPinNo]);
printf("Ext Pin 0번의 Count 수량 : %d", tExtCntInfo.dwCurCount[nEPinNo]);
printf("Ext Pin 0번의 Count 동작시간 : %d", tExtCntInfo.ddwCountTime);

// Count 기능 해제
nRet = nmf_ExtCountStop(nNmfNo, nEPinNo);
if ( nRet != nmf_R_OK )      return;       // Count 해제 실패
```

관련 참조

nmf_ExtCountStart, nmf_ExtCountStartEx, nmf_ExtCountGetInfo, nmf_ExtCountGetTime, nmf_ExtCountStop, nmf_ExtCountFilterSet, nmf_ExtCountFilterGet

7.1.3. 기능 시작(확장)

nmf_ExtCountStartEx	CPU-T

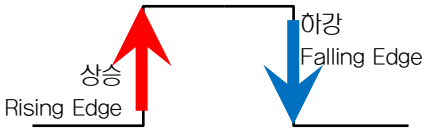
DO 연계에서 DO 상태를 Pulse 형태로 출력 하는 기능이 추가 됐습니다.

구문

short WINAPI **nmf_ExtCountStartEx**(short nNmfNo, short nEPinNo, short nAddMode,
short nEdge, short nDO_ID, short nDO_PinNo, short nDO_On,
short nDO_Width, unsigned int dwCount);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nEPinNo	IN	Count 기능의 External Pin 번호 (범위: 0 ~ 7)
3	nAddMode	IN	추가 기능을 BitMask로 설정합니다. Bit설정 / 내 용 0 : None Bit 0 : 시간 측정 모드 수행여부 기능 동작 여부는 nmf_ExtCountGetTime 함수를 참고하여 주십시오 Bit 1 : Count 반복 모드 수행 여부
4	nEdge	IN	Count 기준 Edge설정 값 / Count 기준 Edge 0 / 상승 1 / 하강 2 / 상승+하강 
5	nDO_ID	IN	연계할 DO보드 ID (범위 : nDO_PinNo의 설명을 참조)
6	nDO_PinNo	IN	기능과 연계하여 사용할 DO보드의 Pin 번호 보드ID (nDO_ID) / Pin 번호 선택범위 0 보다 작은경우 / 연계기능 사용안함 0 ~ 7 / 각 출력보드의 Pin번호 (0~7) 7보다 큰 경우 / [일괄 Pin번호 지정방식] 참고 ※ Pin번호가 -1이면 DO연계 기능 사용하지 않습니다. ※ Pin번호를 일괄적으로 지정하는 경우, **적용가능한 Pin번호의 범위는 연결된 출력보드 수 X Pin수량(16)** 입니다.
7	nDO_On	IN	명령 전달과 동시에 연계된 DO Pin에 설정할 출력상태입니다. 출력 OFF, ON은 dwCount값에 도달시 출력상태는 반전됩니다. 값 / 출력 상태 0 / 출력 OFF 1 / 출력 ON 2 / 출력 Pulse 출력 Pulse는 Count 구동 시작 시 현재 DO 상태를 출력하며, dwCount값 도달시 Pulse 형태로 DO 상태를 반전을 합니

			다.
8	nDO_Width	IN	nDO_On이 출력 Pulse 모드 일 경우 Pulse의 Width를 설정합니다. Pulse Width는 us단위로 1~2000까지 설정 가능합니다 ※ 실제로 출력되는 펄스는 물리적으로 내부 회로적인 이유로 인해 오차가 발생합니다. 오차는 1~20us에선 약 2배이며 20us 이상에선 10us정도의 오차가 적용됩니다. 이를 감안하여 설정하시길 바랍니다.
9	dwCount	IN	목표 Count 수량을 입력합니다. (1 ~ 4294967295(32Bit)) 목표수량에 도달 시 Count 기능은 정지합니다. 0을 입력하는 경우, 무제한으로 동작합니다.

주의사항

No	일람
1	nDO_PinNo 인자가 0보다 작은(-1)값을 갖는 경우 디지털 출력제어와 연계되는 기능을 사용하지 않습니다. 즉, 명령 시점으로부터 External Pin 입력신호의 수량을 Count하는 기능으로만 사용됩니다. 이 경우, nDO_ID, nDO_On 인자값 또한 의미를 갖지 않으며, 동작되지 않습니다.
2	dwCount 인자로 설정된 목표 수량에 도달 시, Count 기능은 정지합니다.
3	Count 동작시간을 0으로 초기화합니다.

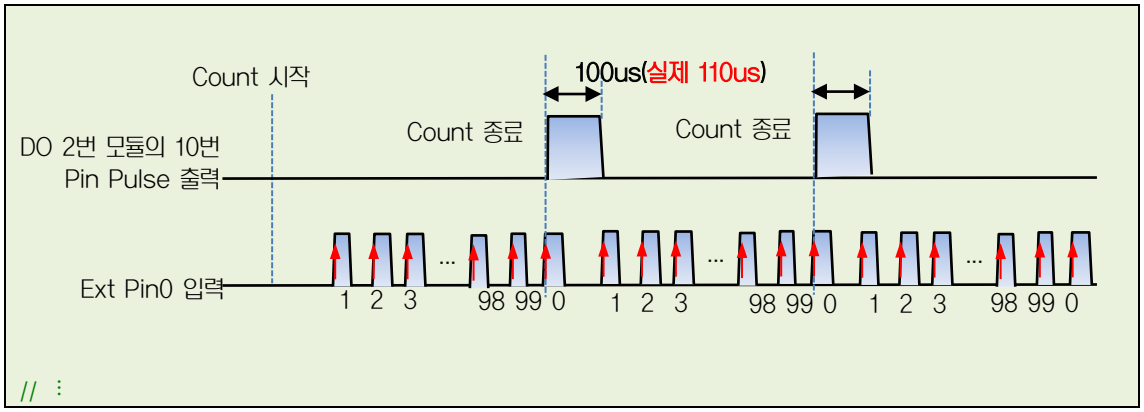
사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호

/*   장치가 연결된 상태 입니다.   */

short nEPinNo = 1;       // Count를 적용할 External Pin 번호
short nEdge = 0;         // Count 적용 시점 Rising
short nAddMode = 0x03; // 시간측정과 Count 반복 모드 사용
short nDO_ID = 2;        // 기능을 연계할 DO보드의 ID
short nDO_PinNo = 10;  // 기능을 연계할 DO보드의 PinNo
short nDO_On = 2;        // DO Pin출력을 Pulse 형으로 출력
uint dwCount = 100;     // 목표 수량 (목표수량 도달 시, 연계된 DO 출력은 반전됩니다.)
short nDO_Width = 100; // Pulse Width를 100us 설정

nRet = nmf_ExtCountStartEx(nNmfNo, nEPinNo, nAddMode, nEdge,
                           nDO_ID, nDO_PinNo, nDO_On, nDO_Width, dwCount);
if ( nRet != nmf_R_OK )      return;       // Count 기능 설정 실패
```



관련 참조

```
// ⋮
```

nmf_ExtCountStart, nmf_ExtCountStartEx, nmf_ExtCountGetInfo, nmf_ExtCountGetTime, nmf_ExtCountStop, nmf_ExtCountFilterSet, nmf_ExtCountFilterGet

7.1.4. 기능 정지

nmf_ExtCountStop	CPU-T

Count 기능을 정지(해제)합니다.

구문

short WINAPI **nmf_ExtCountStop** (short nNmfNo, short nEPinNo);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nEPinNo	IN	Count 기능의 External Pin 번호 (범위: 0 ~ 7)

주의사항

No	일람
1	Count의 모든 기능이 정지됩니다.

사용 예

nmf_ExtCountStart 항목에 작성된 예시를 참고하여 주십시오.

관련 참조

nmf_ExtCountStart, nmf_ExtCountStartEx, nmf_ExtCountGetInfo, nmf_ExtCountGetTime, nmf_ExtCountStop, nmf_ExtCountFilterSet, nmf_ExtCountFilterGet

7.1.5. 상태 확인

nmf_ExtCountGetInfo	CPU-T

설정된 Count의 상태정보를 확인합니다.

설정상태 및 현재 입력된 데이터의 정보를 구조체를 통해 쉽게 확인할 수 있습니다.

구문

short WINAPI **nmf_ExtCountGetInfo** (short nNmfNo, PTNMFEXTCINFO ptExtCInfo);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	ptExtCInfo	OUT	External Pin에 적용된 Count 기능의 구조체 포인터 자세한 설명은 TNMF_EXTC_INFO 구조체 참조

주의사항

No	일람
1	구조체 내부 Count 수량값은 32bit의 크기를 가지며, Count 동작시간은 64bit의 크기를 갖습니다.
2	Count 동작시간은 함수 명령시점의 Timer(1us 단위)값을 획득합니다.

사용 예

nmf_ExtCountStart 항목에 작성된 예시를 참고하여 주십시오.

관련 참조

nmf_ExtCountStart, nmf_ExtCountStartEx, nmf_ExtCountGetInfo, nmf_ExtCountGetTime, nmf_ExtCountStop, nmf_ExtCountFilterSet, nmf_ExtCountFilterGet

7.1.6. 측정 시간 확인

nmf_ExtCountGetTime	CPU-T

Count 기능 설정 시, 시간 측정모드를 적용하여 측정된 값을 확인합니다.

구문

short WINAPI **nmf_ExtCountGetTime** (short nNmfNo, short nEPinNo,
PTNMFCMTSTS ptCMT, short nBufNum, unsigned int *pdwRetBuf);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nEPinNo	IN	Count 기능의 External Pin 번호 (범위: 0 ~ 7)
3	ptCMT	OUT	측정된 시간정보 구조체 포인터 자세한 설명은 TNMF_CMT_STS 구조체 참조
4	nBufNum	IN	배열(pdwRetBuf)의 개수(최대 1024개)
5	pdwRetBuf	OUT	측정된 시간을 읽어올 배열 포인터 (us)

주의사항

No	일람
1	측정시간은 FIFO 형태의 내부 메모리에 최대 1024개가 저장되며, 최대량을 초과하면 값을 갱신하지 않습니다.
2	측정시간은 Count 기능 시작 시, 설정하는 Edge를 기준으로 처음 입력되는 순간이 초기화 값(0)을 갖습니다. 상세내용은 『시간 측정모드』를 참고하여 주십시오.

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호

// External Pin 0번의 카운트(설정된 Edge)당 측정 시간 확인
TNMF_CMT_STS tCmtSts;
unsigned int dwRetTime[1024];
nRet = nmf_ExtCountGetTime(nNmfNo, 0, &tCmtSts, 0, &dwRetTime);
for (int i = 0; i < 1024; i++)
{
    printf("측정시간[i] = %d", dwRetTime[i]);
}
```

관련 참조

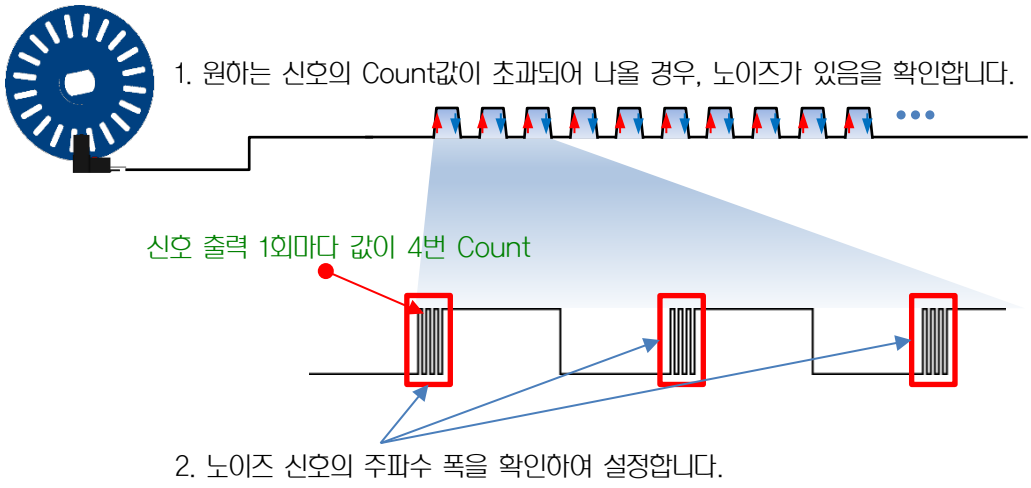
nmf_ExtCountStart, nmf_ExtCountStartEx, nmf_ExtCountGetInfo, nmf_ExtCountGetTime, nmf_ExtCountStop, nmf_ExtCountFilterSet, nmf_ExtCountFilterGet

7.1.7. 입력필터 설정

nmf_ExtCountFilterSet	CPU-T

Count 기능을 사용 중 요구치보다 많은 값이 확인된다면, 채터링이나 노이즈 성분의 신호가 함께 Count 되는 경우입니다. 이 때, 필터를 적용하여 해당 신호들의 Count를 최소화 할 수 있습니다.

해당 기능은 External Pin 3~7 번에만 설정하실 수 있습니다.



구문

short WINAPI **nmf_ExtCountFilterSet** (short nNmfNo, short nWriteFlash, short *pnFilter);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nWriteFlash	IN	Filter 값의 Flash 저장여부 값 0: 저장하지 않음 값 1: 저장 ※ Flash에 저장하기 위해서는 일정시간이 요구됩니다. (약 1초) 해당 시간에 다른 명령 시, 응답지연이 발생할 수 있습니다. ※ 저장된 값은 장비 재시작 이후에도 적용됩니다.
3	pnFilter	IN	External Pin(3~7)에 적용될 Filter Index 배열(5) Filter 값은 아래의 표를 참고하여 주십시오. 설정 가능한 Filter 값은 Pin에 따라 차이가 있습니다.

Filter 표

Index	Filter 주파수	Index	Filter 주파수
0	필터 없음	9	393000 (2.545us)
1	3270000 (0.306us)	10	327000 (3.059us)
2	2450000 (0.409us)	11	273000 (3.663us)
3	1630000 (0.614us)	12	204000 (4.902us)
4	1400000 (0.715us)	13	163000 (6.135us)
5	983000 (1.018us)	14	138000 (7.247us)
6	756000 (1.323us)	15	104000 (9.616us)
7	614000 (1.629us)	16 (Ext 4,5만 적용)	83000 (12.049us)
8	517000 (1.935us) 초기값	17 (Ext 4,5만 적용)	69000 (14.493us)
		18 (Ext 4,5만 적용)	52000 (19.231us)

주의사항

No	일람
1	External Pin의 전원은 DC5V와 DC24V 으로 나뉘어 있습니다. 특히 DC24V 전원에서는 많은 채터링이 발생될 수 있으므로, DC24V 사용시, 500000Hz (2us)보다 큰 값의 필터를 권장합니다.
2	장비(오실로스코프 등)의 부재로 노이즈 신호의 확인이 어려운 경우, Count 가능한 최고 속도를 기준으로 값을 설정하여 주십시오
3	Count 기능의 시간 측정모드를 적용 시, 고속 Count의 수행이 불가하며 Filter는 적용되지 않습니다.

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호

/*   장치가 연결된 상태 입니다.   */

short nWriteFlash = 1;  // 설정한 필터를 Flash에 저장합니다.
short nFilters[5] = {};    // 필터 값

// External Pin 입력의 필터 설정
nRet = nmf_ExtCountFilterSet(nNmfNo, nWriteFlash, nFilters);
if ( nRet != nmf_R_OK )      return;      // Filter 설정 실패

// Flash 저장시간 소요 및 장치 재시작

// 설정된 필터값 확인
nRet = nmf_ExtCountFilterGet(nNmfNo, nFilters);
if ( nRet != nmf_R_OK )      return;      // Filter 설정 실패
```

관련 참조

nmf_ExtCountStart, nmf_ExtCountStartEx, nmf_ExtCountGetInfo, nmf_ExtCountGetTime, nmf_ExtCountStop, nmf_ExtCountFilterSet, nmf_ExtCountFilterGet

7.1.8. 설정필터 확인

nmf_ExtCountFilterGet	CPU-T

External Pin에 설정된 필터 길이(시간)을 확인합니다.

구문

short WINAPI **nmf_ExtCountFilterGet** (short nNmfNo, short *pnRetFilter);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	pnRetFilter	OUT	필터 길이를 받을 배열 포인터 (5)

주의사항

No	일람
1	Filter Index 값이 확인되며, 설정값은 nmf_ExtCountFilterSet 항목을 참고하여 주십시오.

사용 예

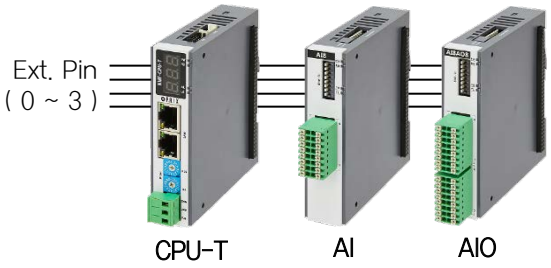
nmf_ExtCountFilterSet 항목에 작성된 예시를 참고하여 주십시오.

관련 참조

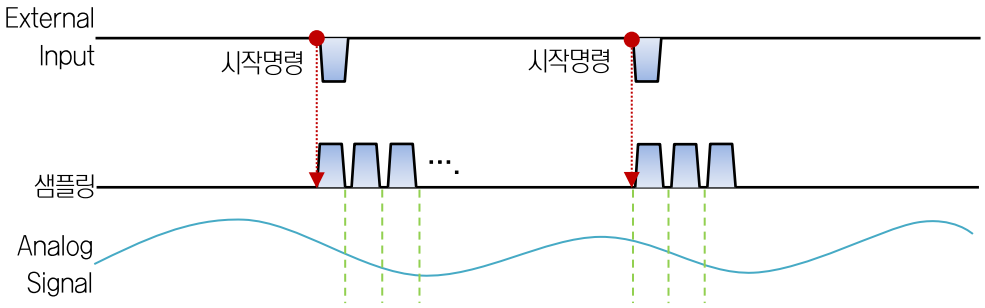
nmf_ExtCountStart, nmf_ExtCountStartEx, nmf_ExtCountGetInfo, nmf_ExtCountGetTime, nmf_ExtCountStop, nmf_ExtCountFilterSet, nmf_ExtCountFilterGet

7.2. Trigger

어떤 행위의 시발점 역할을 수행하는 것을 Tigger라고 합니다. CPU-T 보드의 External Pin에서 입력받은 신호를 해당 기능으로 사용할 수 있으며 External Pin 0~3번에서 지원하고 있습니다.



아래와 같이 특정 시점에 외부 입력이 들어오게 되면, AI 보드의 아날로그 입력을 샘플링하도록 구성합니다.



<그림 7-1> 외부입력에 의한 AI Sampling 연동

하나의 Pin에서 입력받는 신호는 개별 Pin에 등록되거나, 모든 Pin에 등록되어 사용될 수 있습니다.
설정은 [AI] - [입력설정], [보드지정 입력설정] 을 참고하여 주십시오.

8.특수기능

8.1. 시리얼 통신

장치에 내장된 시리얼 통신 포트를 이용하여 기타 장치와 통신이 가능하도록 구성하는 기능입니다.

관련함수

No	명령	설명
1	nmf_SerialSetBypass	시리얼 통신의 사용을 설정합니다.
2	nmf_SerialSetCfg	시리얼 통신환경을 설정합니다.
3	nmf_SerialWrite	장치로 전송받은 데이터를 확인합니다.
4	nmf_SerialRead	데이터를 전송합니다.

8.1.1. 사용설정

nmf_SerialSetBypass	Public

시리얼 통신을 수행할 수 있도록 기능을 활성화 합니다.

구문

short WINAPI **nmf_SerialSetBypass** (short nNmfNo, short nMode);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nMode	IN	기능 활성화 설정 설정값 / 기능 0 / 비활성화 1 / 활성화

주의사항

No	일람
1	장치 재시작 이후 비활성화됩니다. 기능 재사용 시 다시 활성화 하여 주십시오.

관련 참조

nmf_SerialSetBypass, nmf_SerialSetCfg, nmf_SerialWrite, nmf_SerialRead

8.1.2. 환경설정

nmf_SerialSetCfg	Public

시리얼 통신을 위한 환경설정을 수행합니다. 장치 내 EEPROM에 저장되어 유지됩니다.

구문

short WINAPI **nmf_SerialSetCfg** (short nNmfNo, short nBaud, short nData, short nStop, short nParity);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nBaud	IN	통신속도 설정값 / BaudRate 0 / 9,600 1 / 19,200 2 / 38,400
3	nData	IN	데이터 비트 수 설정값 7로 고정 됨 (8Bit)
4	nStop	IN	정지 비트 수 설정값 / StopBit 0 / 1 1 / 2
5	nParity	IN	Parity bit 설정값 / Parity Bit 0 / None 1 / Odd 2 / Even

주의사항

No	일람
1	기본설정 [Baudrate = 38400, DataBit = 8, StopBit = 1, ParityBit = None]
2	변경된 값은 즉시 반영되며, 장치 재시작 후에도 유지됩니다.
3	데이터 비트는 8Bit 로 사용이 고정됩니다.

관련 참조

nmf_SerialSetBypass, nmf_SerialSetCfg, nmf_SerialWrite, nmf_SerialRead

8.1.3. 데이터 송신

nmf_SerialWrite	Public

함수명령을 통해 전달받은 데이터를 Serial 통신으로 전송합니다.

구문

short WINAPI **nmf_SerialWrite** (short nNmfNo, short nLen, char *szData);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	nLen	IN	전송할 데이터의 바이트 수 (최대 1024 bytes)
3	szData	IN	전송할 내용의 char 형 배열 포인터

주의사항

No	일람
1	

관련 참조

nmf_SerialSetBypass, nmf_SerialSetCfg, nmf_SerialWrite, nmf_SerialRead

8.1.4. 데이터 수신

nmf_SerialRead	Public

Serial 통신으로 수신된 데이터를 확인합니다.

구문

short WINAPI **nmf_SerialRead** (short nNmfNo, short *pnReadLen, char *szpRetData);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	pnReadLen	OUT	수신된 데이터를 반환받을 배열의 크기
3	szpRetData	OUT	수신된 데이터를 반환받을 배열 포인터

주의사항

No	일람
1	

관련 참조

nmf_SerialSetBypass, nmf_SerialSetCfg, nmf_SerialWrite, nmf_SerialRead

8.2. 연결 지원

외부에서의 내부 네트워크 연결을 통하여 장치를 제어하기 위해서는 포트 포워딩이 필요합니다.
포트 포워딩은 IP주소마다 할당된 포트번호 중, 외부에서 특정포트로 접속했을 때, 해당 IP로 연결시키는 기능입니다.

8.2.1. 포트 변경

nmf_NetPortNumSet	Public

장치와 연결되는 포트번호를 설정합니다. 기본 1000으로 설정된 포트번호를 변경합니다.

구문

short WINAPI **nmf_NetPortNumSet** (short nNmfNo, int lPortNum);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	lPortNum	IN	접속지원을 위해 변경하고자 하는 포트 번호

주의사항

No	일람
1	이 함수는 장치의 포트번호를 변경하는 것이 아닙니다.
2	외부 네트워크에서의 접속을 위한 기능입니다. 내부 네트워크로 한정하여 사용되는 경우에는 필요하지 않습니다.

9.정보확인

9.1. 전체 상태 확인

nmf_GetBrdAllStatus	Public

장치의 모든 정보를 획득합니다.

구문

short WINAPI **nmf_GetBrdAllStatus** (short nNmfNo, PTNMFALLSTS ptRetAllSts);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	ptRetAllSts	OUT	상태를 받을 구조체 포인터 자세한 설명은 TNMF_ALL_STS 구조체 참조

주의사항

No	일람
1	본 함수로 장치의 모든 정보를 읽을 수 있으므로 사용 빈도가 가장 높습니다.

사용 예

```
short nRet;
short nNmfNo = 11; // 장치번호

// 장치와의 네트워크 연결 상태를 확인합니다.
nRet = nmf_PingCheck(nNmfNo, 192, 168, 0, 200);
if ( nRet != nmf_R_OK )  return;      // 네트워크 연결 상태를 확인할 수 없습니다.
// 장치와 연결을 시도합니다.
nRet = nmf_Connect(nNmfNo, 192, 168, 0);
if ( nRet != nmf_R_OK )  return;      // 장치 연결 실패

TNMF_ALL_STS  tAllStatus;
nRet = nmf_GetBrdAllStatus(nNmfNo, &tAllStatus);
// Type 별 보드 개수
printf("DI Type %d개 연결", tAllStatus.nCntDIBrd);
printf("DO Type %d개 연결", tAllStatus.nCntDOBrd);

// 보드 상태확인
printf("DO 보드 0번의 0번Pin 출력상태 : %d", tAllStatus.tDO[0].nPinStatus[0]);
printf("DI 보드 1번의 7번Pin 입력상태 : %d", tAllStatus.tDI[1].nPinStatus[7]);
printf("AO 보드 2번의 5번Pin 출력전압값 : %0.3f", tAllStatus.tAO[2].dVolt[5]);
printf("AI 보드 3번의 4번Pin 입력전압값: %0.3f", tAllStatus.tAI[3].dVolt[4]);

// 장치 연결을 해제합니다.
nmf_Disconnect(nNmfNo);
```

9.2. 장치 구성

nmf_GetCompo	Public

연결된 장치의 세부 구성정보를 읽어 옵니다. 보드 연결수량 및 보드의 형태를 확인합니다.

구문

short WINAPI **nmf_GetCompo** (short nNmfNo, PTNMFCOMPO ptCompo);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	nNmfNo	IN	장치의 IP번호
2	ptCompo	OUT	상태를 받을 구조체 포인터 자세한 설명은 TNMF_COMPO 구조체 참조

사용 예

```
#define MAX_BRD      8          // 최대 보드 개수
#define BRD_NONE     0          // 보드 없음
#define BRD_DO16     1          // DO16 보드 Index
#define BRD_DI16     2          // DI16 보드 Index
#define BRD_AI8      3          // AI8 보드 Index
#define BRD_AO8      4          // AO8 보드 Index
#define BRD_AI8AO8   5          // AI8AO8 보드 Index
#define BRD_DO8      6          // DO8 보드 Index
#define BRD_DI8      7          // DI8 보드 Index
#define BRD_DI8DO8   8          // DI8DO8 보드 Index

short nRet;
short nNmfNo = 11; // 장치번호

/*   장치가 연결된 상태 입니다.   */

TNMF_COMPO  tCompo;
nRet = nmf_GetCompo(nNmfNo, &tCompo);
if (nRet == nmf_R_OK)
{
    // 연결된 보드 수량
    printf("전체 보드 수량 : %d\n", tCompo.nTotalCntBrd);
    printf("DO16 %d개 연결\n ", tCompo.nCntBrd[BRD_DO16]);
    printf("DI16 %d개 연결\n ", tCompo.nCntBrd[BRD_DI16]);
    printf("AI8 %d개 연결\n ", tCompo.nCntBrd[BRD_AI8]);
    printf("AO8 %d개 연결\n ", tCompo.nCntBrd[BRD_AO8]);
    printf("AO8AO8 %d개 연결\n ", tCompo.nCntBrd[BRD_AI8AO8]);


    // Pin 수량
    printf("DO Pin = %d개\n ", tCompo.nCntBrd[BRD_DO16] * 16);
    printf("DI Pin = %d개\n ", tCompo.nCntBrd[BRD_DI16] * 16);
    printf("AI Pin = %d개\n ", tCompo.nCntBrd[BRD_AI8] * 8 + tCompo.nCntBrd[BRD_AI8AO8] * 8);
    printf("AO Pin = %d개\n ", tCompo.nCntBrd[BRD_AO8] * 8 + tCompo.nCntBrd[BRD_AI8AO8] * 8);
    // 예) AI8AO8 보드는 AI Pin와, AO Pin이 둘다 있으므로.

    for (int i = 0; i < MAX_BRD; i++)
```

```
    {
        // nType의 값은 CPU 보드와 연결된 보드들의 형태를 뜻하며, 배열번호는 보드ID와 같습니다.
        switch (tCompo.nType[i])
        {
            case BRD_DO16: // DO16
                printf("ID %d 보드형태 : DO16 \n", i);
                break;
            case BRD_DI16: // DI16
                printf("ID %d 보드형태 : DI16 \n", i);
                break;
            case BRD_AI8: // AI8
                printf("ID %d 보드형태 : AI8 \n", i);
                break;
            case BRD_AO8: // AO8
                printf("ID %d 보드형태 : AO8 \n", i);
                break;
            case BRD_AI8AO8: // AI8AO8
                printf("ID %d 보드형태 : AI8AO8 \n", i);
                break;
            default:
                printf("ID %d 보드형태 확인불가 \n");
                break;
        }
    }
}

// 장치 연결을 해제합니다.
nmf_Disconnect(nNmfNo);
```

관련 참조

TNMF_COMPO, nmf_GetCompo, nmf_GetCompoList

9.3. 장치 리스트

nmf_GetCompoList	Public

연결된 IP대역의 모든 장치의 구성정보를 읽어 옵니다.

구문

short WINAPI **nmf_GetCompoList** (short *pnIP, short nListCount, PTNMFCOMPO ptCompo);

☞ 함수 반환값은 여기를 참고하세요

인자

No	명칭	IN/OUT	설명
1	pnIP	IN	연결된 네트워크 대역의 UDP 요청 주소 (4자리) 장치 네트워크가 192.168.0 대역에 연결되었다면 192.168.0.255 값을 적용하여 주십시오.
2	nListCount	IN	요청 수량
3	ptCompo	OUT	상태를 받을 구조체 배열 포인터 자세한 설명은 TNMF_COMPO 구조체 참조

주의사항

No	일람
1	반환값이 양수인 경우 응답된 장치 수량을 의미합니다. 최대값은 요청수량(nListCount)와 같습니다.
2	본 함수는 내부적으로 UDP 프로토콜을 사용하므로 방화벽 해제가 필요합니다. 보안 설정 해제 항목을 참고하여 주십시오.

사용 예

```
short nDevCnt;
// 장치 리스트는 내부적으로 UDP 검색을 사용하기에 IP마지막 번호는 255를 고정으로 사용하여 주십시오.
short nIP[4] = {192, 168, 0, 255}; // IP 번호

/*   장치가 연결된 상태 입니다.   */

short nListCnt = 10;          // 검색요청 수량
TNMF_COMPO  tCompo[10]; // 배열 수량은 대역에 연결된 장치 수량이나 검색요청 수량
nDevCnt = nmf_GetCompoList(nIP, 10, tCompo);
if (nDevCnt > 0){
    printf("검색된 장치 수량 : %d\n", nDevCnt);
}
else {
    printf("장치가 검색되지 않습니다.");
}
```

관련 참조

TNMF_COMPO, nmf_GetCompo, nmf_GetCompoList

10. 구조체

NMF는 데이터의 효율적인 구분 및 통신효율을 위하여 내부 구성 및 상태정보를 구조체 형태로 관리하고 있습니다. 구조체는 각 언어 지원파일에 선언되어 있습니다.

10.1. 구성 정보

장치의 구성 정보 구조체입니다.

```
#define MAX_BRD         8           // 최대 보드 개수
...
typedef struct _TNMF_COMPO
{
    short       nIP[4];             // 장치의 IP주소
    short       nModelPart;         // 제품 모델 타입 정보 (0=NMF, ...)
    short       nType[MAX_BRD];     // 보드형태 ( 0=없음, 1=DO16, 2=DI16, 3=AI8, 4=AO8,
                                    // 5=AI8AO8, 6=DO8, 7=DI8, 8=DI8DO8, 9~49=예약)
    short       nCntBrd[50];        // 형태별 보드 개수(Max.8) 배열 번호는 보드 형태 값
                                    // 예) AI8 = 2개, DO16 = 3개일 경우 nCntBrd[1] = 3,
                                    // nCntBrd[3] = 2가 됨.
    short       nTotalCntBrd;       // 연결된 전체 보드 개수 (Max. 8)
    short       nReserved;          // 예약
} TNMF_COMPO, *PTNMFCOMPO;
```

주의사항

No	일람
1	nCntBrd의 배열 Index 9부터는 Reserved 입니다.

관련 참조

nmf_GetCompo,

10.2. 전체 상태 정보

장치의 전체 상태를 확인할 수 있는 정보 구조체 입니다.

CPU보드와 연결된 각 보드들의 정보를 확인할 수 있습니다. 보드에 설정된 ID는 배열의 번호가 되며, 설정되지 않은 번호는 값이 확인되지 않습니다.

```
#define MAX_BRD         8           // 최대 보드 개수
...
typedef struct _TNMF_ALL_STS
{
    short                   nCntDIBrd;      // DI 보드 개수
    short                   nCntDOBrd;      // DO 보드 개수
    short                   nCntAIBrd;      // AI 보드 개수
    short                   nCntAOBrd;      // AO 보드 개수
    TNMF_BRD_DIO_STS        tDI [MAX_BRD];  // DI 데이터
    TNMF_BRD_DIO_STS        tDO[MAX_BRD];   // DO 데이터
    TNMF_BRD_AI_STS         tAI [MAX_BRD];  // AI 데이터
    TNMF_BRD_AO_STS         tAO[MAX_BRD];   // AO 데이터
    TNMF_EXT_STS            tExtSts;        // External Pin 정보
} TNMF_ALL_STS, *PTNMFALLSTS;
```

주의사항

No	일람
1	AIO 보드가 연결되는 경우, AI, AO 보드가 모두 수량이 증가합니다.

관련 참조

nmf_GetCompo, nmf_GetBrdAllStatus

10.3. DIO 상태 정보

Digital 입출력상태 정보 구조체 입니다.

```
#define BRD_VS_DIO_PINS        16       // DIO 보드의 핀 개수
...
typedef struct _TNMF_BRD_DIO_STS
{
    short       nID;                    // 보드 ID
    short       nReserved;              // 예약
    short       nPinStatus    [BRD_VS_DIO_PINS];
                // DI : 0=입력OFF, 1=입력ON
                // DO : 0=출력OFF, 1=출력ON
} TNMF_BRD_DIO_STS, *PTNMFBRDDIOSTS;
```

관련 참조

nmf_GetBrdAllStatus

10.4. AO 상태 정보

Analog 출력상태 구조체입니다.

```
#define BRD_VS_AO_PINS        8       // AO 보드의 핀 개수
...
typedef struct _TNMF_BRD_AO_STS
{
    short       nID;                  // 보드 ID
    short       nReserved;            // 예약
    short       nRange    [BRD_VS_AO_PINS];  // AO보드의 각 Pin별 출력범위
                                      // 설정값 (0 : 0~5V,  1 : 0~10V,  2 : 0~10.86V
                                      //         3 : ±5V,  4 : ±10V,  5 : ±10.86V)
    int         lDAValue  [BRD_VS_AO_PINS];
                // 설정범위(nRange)에 따른 최근 출력 전압의 디지털 값
                // (0,1,2 = 0~65535, 3,4,5 = -32768 ~ 32767 )
    double      dVolt     [BRD_VS_AO_PINS];       // 설정된 전압값
} TNMF_BRD_AO_STS, *PTNMFBRDAOSTS;
```

관련 참조

nmf_GetBrdAllStatus

10.5. AI 상태 정보

Analog 입력상태 구조체입니다.

Analog 입력값을 나타내는 구조체 멤버 lADValue(AD 값), dVolt(전압), dCurrent(전류)는 별도로 선택 없이 확인 됩니다.

```
#define BRD_VS_AI_PINS          8       // AI 보드의 핀 개수
...
typedef struct _TNMF_BRD_AI_STS
{
    short       nID;            // 보드 ID
    short       nStatus         [BRD_VS_AI_PINS];   // 샘플링 상태
                // (0=미실행, 1=트리거 입력 대기중, 2=실행 중,
                // 3=Read중 샘플링 대기 중, 4=사용자 정지, 5=메모리 Full 정지)
    short       nTrgType        [BRD_VS_AI_PINS];   // 샘플링 트리거 타입
                // (0=임의 시점, 1=Trg Rising, 2=Trg Falling 3=Trg Both)
    short       nExtPin         [BRD_VS_AI_PINS];   // 샘플링에 트리거를 사용하는 경우
                                                    // External Pin 번호 (0 ~ 3)
    short       nRange;         // AI의 입력 범위 (0=±5V, 1=±10V)
    int         lRate           [BRD_VS_AI_PINS];   // AD 샘플링 주기 (0 ~ 100000 Hz)
    short       nMemFullOvW     [BRD_VS_AI_PINS];   // 메모리 Overwrite 여부
                // (0=Ovewrite안함(샘플링 정지), 1=Overwrite(샘플링 정지안함)
    short       nMemFullFlag    [BRD_VS_AI_PINS];   // 메모리 Full Flag (0=Full X, 1=Full O)
    short       nMemExists      [BRD_VS_AI_PINS];   // 메모리의 상태
                                                    // (0=데이터 없음, 1=데이터 있음)
    int         lADValue        [BRD_VS_AI_PINS];
                // 설정된 입력 범위에 따른 Pin에 입력된 전압의 디지털 값 (-32768 ~ 32767)
    double      dVolt           [BRD_VS_AI_PINS];   // Pin에 입력된 전압 값
    double      dCurrent        [BRD_VS_AI_PINS];   // Pin에 입력된 전류 값
} TNMF_BRD_AI_STS, *PTNMFBRDAISTS;
```

관련 참조

nmf_GetBrdAllStatus

10.6. AI Sampling 정보

AI보드에서 Smpling 기능을 설정한 후, 확인되는 Sampling정보 구조체 입니다.

전체상태 확인함수로 확인할 수 없으며, 전용 함수를 참고하여 주십시오.

```
typedef struct _TNMF_BRD_AI_SMPL
{
    short       nID;                // 보드 ID
    short       nReserved;          // 예약
    short       nStatus;            // 샘플링 상태
                // (0=미실행, 1 = 트리거 입력 대기중, 2=실행 중,
                // 3 = Read중 샘플링 대기 중, 4=사용자 정지, 5=메모리 Full 정지)
    short       nTrgType;           // 샘플링 트리거 타입
                // (0=임의 시점, 1=ExtTrg Rising, 2=ExtTrg Falling 3=ExtTrg Both)
    short       nExtPin;            // 샘플링에 트리거를 사용하는 경우
                                    // External Pin 번호 (0 ~ 3)
    short       nRange;             // AI 보드의 입력 범위 (0=±5V, 1=±10V)
    int         lRate;              // AD 샘플링 주기 (0 ~ 100000 Hz)
    short       nMemFullOvW;        // 메모리 Overwrite 여부
                // (0=Ovewrite안함(샘플링 정지), 1=Overwrite(샘플링 정지안함)
    short       nMemFullFlag;       // 메모리 Full Flag (0=Full X, 1=Full O)
    int         lMemCntData;        // 데이터로 채워져 있는 메모리 개수 (최대 133120개)
    int         lMemCntEmpty;       // 비어 있는 메모리 개수 (최대 133120개)
    int         lRespSize;          // 응답 데이터 개수 (최대 133120개)
                // 원형큐로 운영되는 데이터 관리방식으로 인해
                // 확인되는 데이터 수량은 할당크기보다 1개의 적습니다.
} TNMF_BRD_AI_SMPL, *PTNMFBRDAISMPL;
```

관련 참조

nmf_AIBrdSmplValueGetPin

10.7. External Pin 정보

CPU-T 보드에 적용된 외부 입력 신호의 설정 정보 구조체 입니다.

외부 입력신호는 Count 기능 및 AI 보드와 연계한 입력 Trigger로의 사용이 가능합니다.

```
#define EXT_IN_PINS          8        // External Pin 개수
...
typedef struct _TNMF_EXT_STS
{
    short           nStatus        [EXT_IN_PINS];    // Ext. Pin의 입력상태
    short           nCountEnable   [EXT_IN_PINS];    // Ext. Pin의 Count 기능 사용 여부
    unsigned int    dwCountValue[EXT_IN_PINS];       // Count 기능 사용 시, Count(횟수)
    unsigned long long   ddwCountTime;               // Count 동작시간(us)
} TNMF_EXT_STS, *PTNMFEXTSTS;
```

관련 참조

nmf_GetBrdAllStatus

10.8. Count 상태 정보

Count 기능의 시간측정모드를 사용하는 경우 측정된 데이터 구조체 입니다.

전체상태 확인함수로 확인할 수 없으며, 전용 함수를 참고하여 주십시오.

```
#define EXT_IN_PINS          8       // External Pin 개수

typedef struct _TNMF_EXTC_INFO
{
    unsigned long long      ddwCountTime;       // Count 동작시간(us)
    short           nReserved0;       // 예약0
    short           nReserved1;       // 예약1
    short           nStatus[EXT_IN_PINS];    // Count의 해당 External Pin 입력상태
    short           nEdge[EXT_IN_PINS];      // 설정 신호 Edge(0=Rising, 1=Falling, 2=Both)
    short           nAddMode[EXT_IN_PINS];    // 추가 기능 여부
                                              // Bit 0=시간 측정 모드, 0:사용안함, 1:사용
                                              // Bit 1=Count 반복 모드, 0:사용안함, 1:사용
    short           nEnable[EXT_IN_PINS];           // Count 사용 여부
    short           nDO_ID[EXT_IN_PINS];            // 연계 DO보드 ID
    short           nDO_PinNo[EXT_IN_PINS];         // 연계 DO보드의 Pin No
    short           nDO_PinStatus[EXT_IN_PINS];     // 연계 DO보드의 Pin 상태
    unsigned int    dwSetCount[EXT_IN_PINS];        // 설정 Count 값 (DO 연계에 적용)
    unsigned int    dwCurCount[EXT_IN_PINS];        // 현재 Count 값
    short           nInterval[EXT_IN_PINS];         // 측정간격(us)
    unsigned int    dwMTime[EXT_IN_PINS];           // 측정시간
    short           nEReserved0[EXT_IN_PINS];       // Ext Pin 별 예약0
    short           nEReserved1[EXT_IN_PINS];       // Ext Pin 별 예약1
} TNMF_EXTC_INFO, *PTNMFEXTCINFO;
```

관련 참조

nmf_ExtCountGetInfo,

10.9. Count 측정시간 정보

External Pin을 이용한 Count 기능을 통해 확인된 측정 시간의 데이터 구조체입니다.

전체상태 확인함수로 확인할 수 없으며, 전용 함수를 참고하여 주십시오.

```
typedef struct _TNMF_CMT_STS
{
    short          nStatus;        // 해당 External Pin의 입력상태
    short          nEdge;          // 입력신호 기준 Edge(0=Rising, 1=Falling, 2=Both)
    short          nAddMode;       // 추가 기능 여부
                                   // Bit 0=시간 측정 모드, 0:사용안함, 1:사용
                                   // Bit 1=Count 반복 모드, 0:사용안함, 1:사용
    short          nEnable;        // Count 사용여부
    unsigned int   dwValue;        // Count 횟수
    short          nInterval;      // 측정간격(us)
    short          nMemCntData;    // 데이터로 채워져 있는 메모리 개수 (최대 1024개)
    short          nMemCntEmpty;   // 비어 있는 메모리 개수 (최대 1024개)
    short          nRespSize;      // 응답 데이터 개수 (최대 1024개)
    short          nReserved0;     // 예약0
    short          nReserved1;     // 예약1
} TNMF_CMT_STS, *PTNMFCMTSTS;
```

관련 참조

nmf_ExtCountGetTime,

(주) 파익스
서울시 구로구 구로동222-31번지 G+코오롱디지털타워 505,506호
Tel : (02) 833-6663
Fax : (02) 833-6667

PRECISION
AUTOMATION
INDUSTRIAL
EXPERT

이 설명서의 내용은 예고없이 변경될 수 있으므로 구입 시 반드시 확인하시기 바랍니다.
매뉴얼의 해당 저작권법을 준수하는 것은 사용자의 책임 입니다. 저작원에서의 권리와는 별도로, 이 설명서의 어떠한 부분도 ㈜파익스의 명시적인 서면 승인 없이는 다른 목적으로 복제되거나, 검색 시스템에 저장 또는 도입되거나 전송될 수 없습니다.