CCU+ 사용자 설명서

삼성PLC N plus시리즈용
컴퓨터 커뮤니케이션 유니트

NX700 PLC (NX-CCU+)
N700 PLC (CPL7463)

SAMSUNG | Rockwell Automation

사용하시기 전에

■ 제품의 설치, 운전, 보수, 점검에 앞서 반드시 이 매뉴얼을 읽고, 기기에 대한 지식, 안전정보, 기타 주의사항을 모두 숙지하신 후 사용하시기 바랍니다.

■ 원하시는 제품인지를 확인하시고, 파손이나 빠진 부품이 없는지 확인하시기 바랍니다.

안전에 관한 주의사항

반드시 지켜 주십시오.

■ 본 취급설명서에 있는 '경고' 또는 '주의' 사항이 명기된 곳을 참조로 하여 안전하게 사용하시길 바랍니다.

경우에 따라 '⚠ 주의' 표시를 한 곳에서도 고장의 원인이 될 수도 있으므로 주의해 사용하시기 바랍니다.

⚠ 경고 경고 표시
잘못 취급하였을 경우 사용자가 사망 또는 중상을 입는 위험상태가 발생할 것으로 예상되는 경우
● 인체사고나 중대한 손해로 확대될 것으로 예측되는 용도로 사용하실 경우에는 고장시에도 사고가 발생하지 않고 안전하게 작동하도록 외부회로를 구성하여 안전대책을 세워 주십시오. ● 부착이나 배선작업, 보수점검 및 청소시 외부전원을 반드시 차단하시고, 통전중에는 절대 전원단자를 만지지 마십시오. 감전의 우려가 있습니다. ● 연소성 가스가 있는 곳에서는 사용하지 마십시오. 폭발의 원인이 됩니다.

⚠ 주의

주의 표시

잘못 취급하였을 경우 사용자가 상해를 입거나 또는 물적 손해가 발생하는 위험상태가 발생할 것으로 상정되는 경우

- 제품규격에 명기된 사용환경의 기준 이내에서 사용이나 보관 하시고, 설치작업시 이물질이 제품속으로 들어가지 않도록 주의해 주시기 바랍니다.
  - 고온, 고습, 많은 먼지, 염분, 금속입자, 부식성 가스, 인화성 가스, 솔벤트, 연마류, 직사광선 등에 노출하지 마십시오.
  - 진동이 있거나 다른 물체에 부딪치지 않도록 해야 합니다.
  화재, 손상, 오동작 또는 노화의 원인이 됩니다.
- 통신케이블은 전원케이블과 분리된 통로로 배선하시고, 통로를 200mm이상 간격을 두어 설치 바랍니다. 잡음이나 노이즈가 야기되어 오동작이 생길 수 있습니다.
- 비상정지, 인터록 회로는 외부회로에서 구성해 주십시오.
- 정격사양, 환경등의 사양범위 이외에서는 사용하지 마십시오.
  이상발열이나 고장의 원인이 됩니다.
- 분해나 개조하지 마시고, 제품수리는 서비스 센터를 이용해 주시기 바랍니다.
  고장의 원인이 되기도 합니다.
- 전류가 흐르고 있는 동안에는 단자를 만지지 마십시오.
  감전의 우려가 있습니다.
- 제품 및 매뉴얼에 명기된 용도로만 사용하시기 바랍니다.
- 제품을 폐기할 때는 산업폐기물 기준에 따르십시오.

목 차

제 1 장

제품의 개요

1.1 CCU+의 개요

CCU+ (Plus 시리즈용 컴퓨터 커뮤니케이션 유니트)는 N plus 계열의 CPU 통신단자에서 제공되는 RS-232C 또는 RS-485 통신방식, 통신 Protocol, 결선도 및 기능이 동일합니다. 즉, 컴퓨터와 CCU+ 프로토콜을 이용하여 1:1 또는 1:N의 Multi-drop 양방향 통신이 가능하며, CPU에서 제공되는 통신 Port가 부족하거나 확장할 경우 CCU+모듈을 설치하여 각종 주변기기를 쉽게 사용할 수 있습니다.

1.2 CCU+ 사용에 따른 제한사항

CCU+ 유니트는 다음과 같은 사용상의 제한이 있으므로 주의하시기 바랍니다.

1) CCU+ 는 N plus 계열의 CPU가 장착된 베이스에 사용 가능합니다.
2) 링크 유니트를 포함하여 3대까지 설치하여 사용이 가능합니다.
3) CPU가 장착된 기본 베이스에 설치가 가능합니다.
4) CCU+가 장착된 슬롯에는 입출력 번지를 할당받지 않으며, 임의로 지정을 할 수 없습니다. 번지를 강제로 설정시 CPU에서 Error가 발생합니다.

사용 가능한 CPU종류

제 품 명		CCU+ 유니트 모델명	CCU+ 유니트 사용수	비 고
PLC 시리즈	CPU 유니트			
NX700	NX-CPU700	사용 불가	-	NX-CCU 필요
	NX-CPU750A NX-CPU750B NX-CPU750C NX-CPU750D	사용 불가	-	NX-CCU 필요
	NX-CPU700p	NX-CCU+	링크유니트를 포함 3대까지 사용가능	
NX70 PLC	NX70-CPU70	사용 불가	-	CCU 유니트 필요 (CPL9462)
	NX70-CPU750			
	NX70-CPU70p1	사용 불가	-	
	NX70-CPU70p2	사용 불가	-	V-up후 출시예정
N700 PLC	CPL7211A CPL6210A CPL6210B	사용 불가	-	CCU 유니트 필요 (CPL7462)
	CPL7215A	CPL7463	링크유니트를 포함 3대까지 사용가능	

1.3 CCU+ 유니트의 설치환경

⚠ 주의 **이런 환경은 피해야 합니다. …**

1. 주변온도가 0~55℃를 넘는장소
2. 직사광선에 직접 노출된 장소
3. 습도가 30~85%를 넘는 장소
4. 전자부품에 영향을 주는 화공약품을 취급하는 장소
5. 지나치게 먼지, 염분이 많은 장소
6. 주변에 고전압, 강한자장, 강한전자파가 있는 장소
7. 충격, 진동이 심한 장소

⚠ 주의 **모듈을 본체에 장착 할 때의 순서**

1. 컴퓨터 커뮤니케이션 유니트(CCU+)의 통신포트에 통신케이블을 접속합니다.
2. 컴퓨터 커뮤니케이션 유니트(CCU+)에 접속된 외부기기의 전원을 ON시킵니다.
3. PLC 기본베이스에 전원을 ON시킵니다.

⚠ 주의 **모듈을 본체에서 분리 할 때의 순서**

1. PLC 주전원을 먼저 차단합니다.
2. 컴퓨터 커뮤니케이션 유니트(CCU+)에 접속된 외부기기의 전원을 차단시킵니다.
3. 통신케이블을 분리합니다.

⚠ 주의 **CCU+ 유니트의 오동작을 방지 요령**

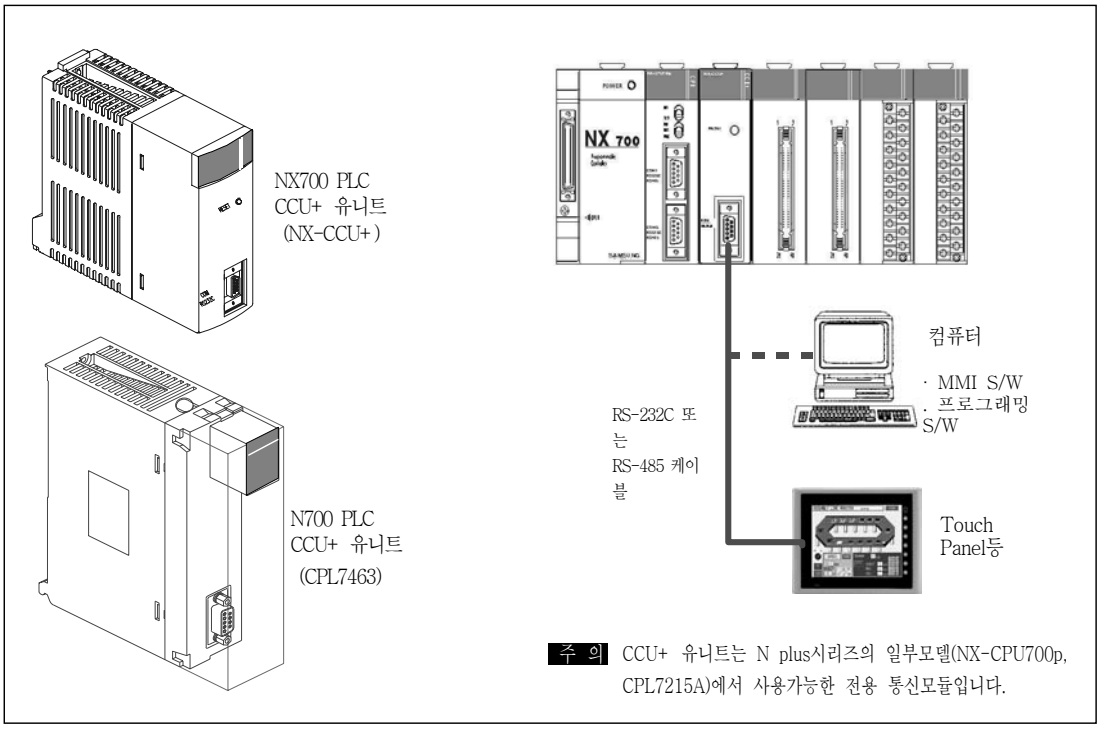
1· CCU+ 유니트를 실장하거나 뽑아 낼때는 전원을 OFF 시킨 상태에서 하십시오.
2· CCU+ 유니트를 마더보드에 확실하게 고정시켜 사용하십시오.
3· 배선시에 유니트 내부에 케이블조각이나 오염물 등이 들어가지 않도록 주의해 주십시오.
4· 유니트 밑면에 있는 커넥터 (마더보드 접속용)는 직접 손으로 만지지 마십시오. 접촉불량이나 정전기등으로 인한 부품파괴의 원인이 됩니다.
5· CCU+ 유니트의 케이스는 사출(성형수지)로 되어 있으므로 낙하나 충격을 주지 마십시오

⚠ 주의 **PLC의 오동작을 방지 요령**

1. 주변에 대용량의 기기/고전압/강한 자장이 있을 때는 PLC전원 입력단에 절연트랜스와 필터를 연결하여 깨끗한 전원을 사용합니다.
2. PLC본체 접지와 다른 장비 접지는 분리하고 반드시 3종 접지합니다.
3. 특히, PLC 본체에서 제공하는 외부 24V 전원은 정격에 맞게 사용해야 합니다. 그렇지 않으면 에러의 원인이 됩니다.
4. PLC 명령어를 충분히 이해하여 프로그램의 실수가 없도록 합니다.
5. 정기적으로 장비, 배선상태등을 점검하는 습관을 갖습니다.

1.4 CCU+의 특징

CCU+ 유니트에 내장된 Protocol을 이용하여 다음과 같이 다양한 기능을 수행하는 시리얼 통신용 제품입니다.



주 의 CCU+ 유니트는 N plus시리즈의 일부모델(NX-CPU700p, CPL7215A)에서 사용가능한 전용 통신모듈입니다.

1) PLC ↔ 컴퓨터간 RS232C 또는 RS485 통신

CCU+에 내장된 포트를 통해 RS-232C 또는 RS485통신을 별도의 장치 없이 스위치 상태에 따라 전환할 수 있으며, 1:1 또는 1:N의 Multi-drop 통신이 가능합니다.

2) N plus Protocol의 2단계/4단계 통신지원

CPU 통신 Port와 같이 2단계 및 4단계 통신 Protocol을 동시에 지원하며, 한번에 250바이트 까지 데이터를 송수신 할 수 있습니다.

3) 다양한 주변기기와 통신

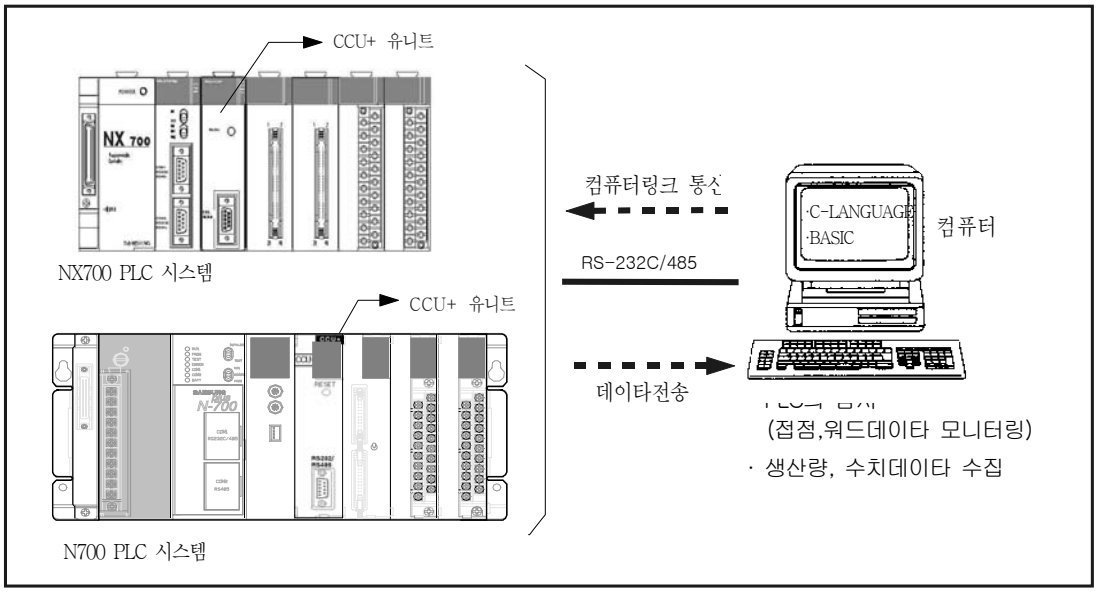
PC에서 운영되는 프로그래밍 S/W(WinGPC)로 Upload 및 Download를 할 수 있으며, 터치 판넬을 이용한 모니터링/제어, PC의 MMI S/W를 통한 각종 모니터링장치와 접속하여 상태감시 및 제어를 할 수 있습니다.

제 2 장

시스템 구성 및 사양

2 시스템 구성 및 사양

2.1 시스템 구성



2.2 시스템 사양

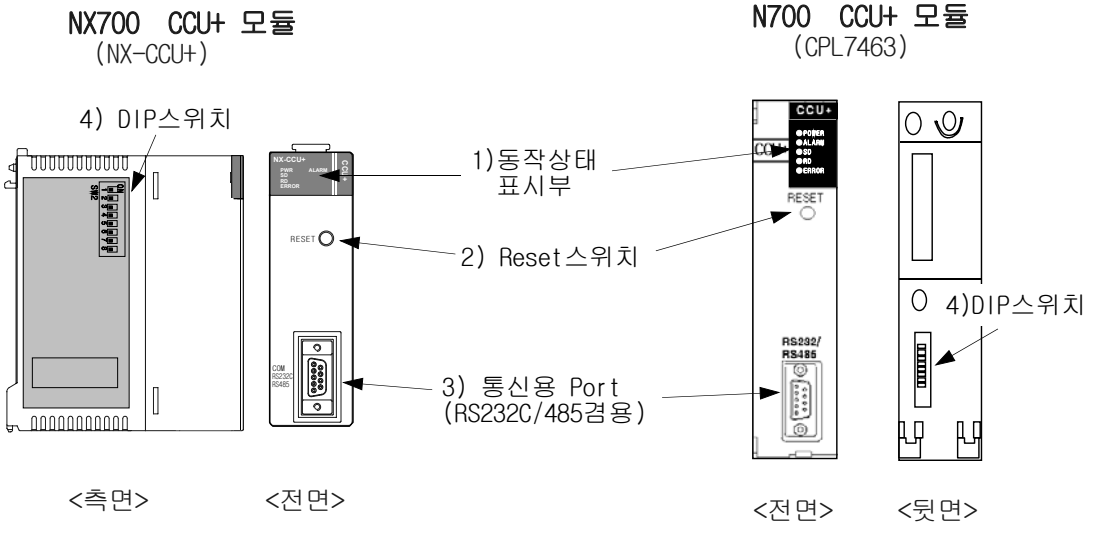
2.2.1 일반사양

항 목		사 양
주변온도	사용 온도	0 ~ 55℃
	보존 온도	-25 ~ 70℃
주변습도	사용 습도	30 ~ 80% RH (단, 이슬이 없을때)
	보존 습도	30 ~ 80% RH (단, 이슬이 없을때)
내 진 동		10 ~ 55Hz 1掃引/ 1분간 , 복진폭 0.75mm , X,Y,Z 각 방향 10분간
내 충 격		최대충격 가속도 및 인가시간 15g/ 11ms, X, Y, Z 방향 각 3회
내 노이즈성		1500Vp-p 펄스폭 50ns, 1μs (노이즈 시뮬레이터에 의함)
사 용 환 경		부식성 가스가 없을것 , 먼지가 심하지 않을것
점유 I/O 점수		점유하지 않음
최대 사용수		NX700 PLC (NX-CPU700p), N700 PLC(CPL7215A) 구성에서 1개만 장착 가능합니다.

2.2.2 통신 제어사양

항 목	사 양
인터페이스	RS-232C/RS-485 1포트
전송 속도	4800/ 9600/ 19200/38400 bps (CCU+ 유니트에 있는 DIP 스윗치로 조정)
통신 방식	반 2중 비동기방식 (Half DUPLEX Asynchronous)
동기 방식	Polling 방식
전송 코드	Binary (HEX)
전송데이타 포맷	데이타 길이 : 8bit
	STOP BIT : 1bit
	Parity : 없음 (NO Parity)
에러 체크 방식	CRC-16 방식
데이타 전송 순서	Byte 단위로 하위 바이트 부터 전송
전송 케이블	Twisted Pair Cable

2.3 각부의 명칭



2.4 표시장치의 기능

2.4.1 동작상태표시 기능

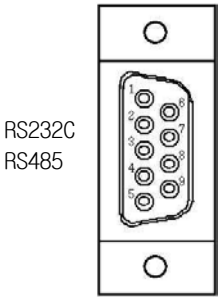
표시 명칭	점등(ON)상태 기능	점멸(깜박임)상태 기능	소등(OFF)상태 기능
PWR	동작중 표시	-	전원 OFF, 미작동상태
SD	-	데이터 송신중	송신 데이터가 없음
RD	-	데이터 수신중	수신 데이터가 없음
ERROR	통신이상 발생	-	정상 상태
ALARM	유니트 이상 (Reset스위치로 초기화)	-	정상 상태

2.4.2 RESET 스위치 기능

- Reset 스위치는 CCU+모듈에서 ALARM 발생시 CCU+ 모듈 초기화 기능을 가지고 있습니다.

2.4.3 통신 Port 기능

- 통신 Port (RS232C/485 지원용)



CCU+ 통신Port
(9Pin Female)

핀번호	신호	내 용
1	-	-
2	SD (TXD)	SEND DATA (RS232C)
3	RD (RXD)	RECEIVE DATA (RS232C)
4	-	-
5	SG (GND)	SIGNAL GROUND
6	485-	RS-485 -
7	485+	RS-485 +
8	-	-
9	Vcc	Vcc (+5VDC)

* 커넥터의 방향에 따라 핀 번호가 달라질 수 있습니다.

- 결선도 1

PLC(CCU+)측
(9Pin D-Sub 수컷)

신호	핀번호
FG	1
SD (TXD)	2
RD (RXD)	3
-	4
SG (GND)	5
485-	6
485+	7
-	8
Vcc	9

PC측 커넥터
(9Pin D-Sub 암컷)

핀번호	신호
1	FG
2	RD (RXD)
3	SD (TXD)
4	DTR
5	SG (GND)
6	DSR
7	RTS
8	CTS
9	

< PC용 통신 케이블 결선도 >
(RS232C/485겸용 케이블)(CPL5530, 5531)

- 결선도 2

PLC(CCU+)측
(9Pin D-Sub 수컷)

신호	핀번호
FG	1
SD (TXD)	2
RD (RXD)	3
-	4
SG (GND)	5
485-	6
485+	7
-	8
Vcc	9

PC측 커넥터
(9Pin D-Sub 암컷)

핀번호	신호
1	FG
2	RD (RXD)
3	SD (TXD)
4	DTR
5	SG (GND)
6	DSR
7	RTS
8	CTS
9	

< PC용 통신 케이블 결선도 >
(RS232C 전용 케이블)- Flow 제어 없는 3선식결선

- 결선도 3

PLC(CCU+)측
(9Pin D-Sub 수컷)

신호	핀번호
FG	1
SD (TXD)	2
RD (RXD)	3
-	4
SG (GND)	5
485-	6
485+	7
-	8
Vcc	9

PC측 커넥터
(25Pin D-Sub 암컷)

핀번호	신호
1	FG
2	SD (TXD)
3	RD (RXD)
4	RTS
5	CTS
6	DSR
7	SG (GND)
8	CD
20	DTR

< PC용 통신 케이블 결선도 >
(RS232C 전용 케이블)- Flow 제어 없는 3선식 결선

- 결선도 4

PLC(CCU+)측
(9Pin D-Sub 수컷)

신호	핀번호
FG	1
SD (TXD)	2
RD (RXD)	3
-	4
SG (GND)	5
485-	6
485+	7
-	8
Vcc	9

RS-485 기기

	485-
	485+

< RS-485 통신 케이블 결선도 >

- 결선도 5

PLC(CCU+)측
(9Pin D-Sub 수컷)

신호	핀번호
FG	1
SD (TXD)	2
RD (RXD)	3
-	4
SG (GND)	5
485-	6
485+	7
-	8
Vcc	9

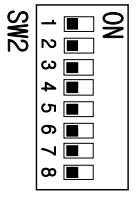
RS-422 기기

	신호
RD-	422- Data수신
RD+	422+ Data수신
SD-	422- Data송신
SD+	422+ Data송신

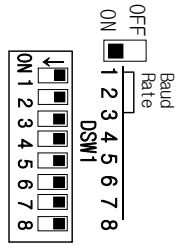
< RS-422 통신 케이블 결선도 >

2.4.3 DIP 스위치 설정기능

- DIP 스위치를 통해 통신속도, 통신방식등을 조정할 수 있습니다.



NX700 CCU+ 모듈
(NX-CCU+)



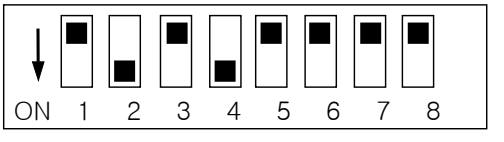
N700 CCU+ 모듈
(CPL7463)

스위치 번호 (주1)						기 능	
1 2 3	4	5	6	7	8	구분	상세기능
0 0 0						통신속도 조정 (Baud Rate)	38400 bps
1 0 0							19200 bps
0 1 0							9600 bps
1 1 0							4800 bps
0 0 1 1 0 1 0 1 1 1 1 1							2400 bps 1200 bps 600 bps 300 bps
	0 1					Data 길이	7 Bit 8 Bit
		0 1				Parity 체크	없음 있음(Parity 설정 영향)
			0 1			Parity 설정 주2)	기수(ODD) 패리티 우수(EVEN) 패리티
				0 1		Stop Bit 길이	1 Stop Bit 2 Stop Bit
					0 1	제어신호, 주3)	CTS,CD 신호 사용안함 CTS,CD 신호 사용

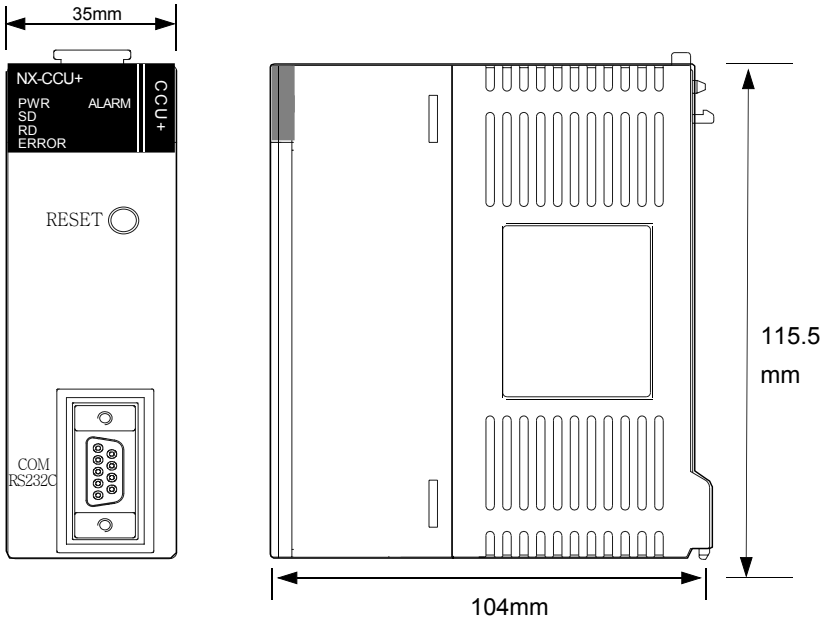
주1) 스위치 번호의 "0"은 OFF 상태를 말하며, "1"은 ON상태를 말합니다.
주2) Parity 설정은 'Parity 체크'가 '없음'(0)으로 설정되어 있으면 무효로 처리됩니다.
주3) 제어신호에 관해서는 CTS, CD를 무효/유효를 설정할 수 있지만,「3선식 FLOW 제어없는 방식」을 사용시에는 '0'으로 해 주십시오.

예) 아래와 같은 통신상태 설정의 경우 스위치 상태표시 예제입니다.

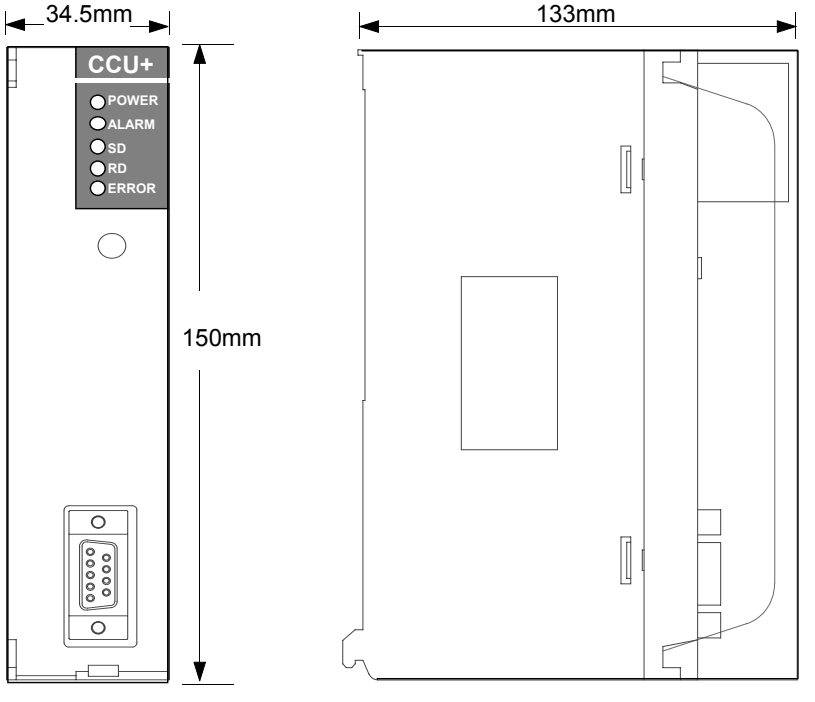
- Baud-Rate : 9600bps
- 데이타 길이 : 8bit
- 페리티 체크 : 없음
- STOP Bit 길이 : 1bit
- 제어 신호 : 무효



2.5 외형 치수도



NX700 PLC용 CCU+ 모듈 (NX-CCU+)



N700 PLC용 CCU+ 모듈 (CPL7463)

2.6 설치 및 번지 지정

2.6.1 CPU 유니트에 따른 사용제한

제 품 명		CCU+ 유니트 모델명	CCU+ 유니트 사용수	비 고
PLC 시리즈	CPU 유니트			
NX700	NX-CPU700	사용 불가	-	NX-CCU 필요
	NX-CPU750A NX-CPU750B NX-CPU750C NX-CPU750D	사용 불가	-	NX-CCU 필요
	NX-CPU700p	NX-CCU+	링크유니트를 포함 3대까지 사용가능	
NX70 PLC	NX70-CPU70	사용 불가	-	CCU 유니트 필요 (CPL9462)
	NX70-CPU750			
	NX70-CPU70p1	사용 불가	-	-
	NX70-CPU70p2	사용 불가	-	V-up후 출시예정
N700 PLC	CPL7211A CPL6210A CPL6210B	사용 불가	-	CCU 유니트 필요 (CPL7462)
	CPL7215A	CPL7463	링크유니트를 포함 3대까지 사용가능	

2.6.2 장착 위치에 따른 제한

- CCU+ 유니트는 기본 베이스측에만 장착 가능합니다.
- CCU+ 유니트는 신뢰도를 높이기 위해, 가능한 한 CPU 유니트 근처부터 장착할 것을 권합니다.

2.6.3 입출력 번지 지정

- CCU+유니트를 장착하면 자동적으로 CCU 모듈로 인식하며 입출력 번지는 할당받지 않습니다.
- 구형의 CPU 버젼일 경우 자동으로 인식하지 못할 경우가 있으므로 Upgrade를 실시한 후 사용하시기 바랍니다.

⚠ 주의

- N plus 계열의 CPU에서는 통신과 관련된 Link, Remote, CCU 모듈의 입출력 번지는 할당받지 않으므로 주의하시기 바랍니다.
- 이러한 모듈의 경우 강제 할당을 하면 정상적인 작동이 되지 않습니다.

제 3 장

통신규약 및 절차

3 통신규약 및 절차

3.1 통신개요

3.1.1 특징

CCU+ 및 N plus 시리즈의 통신은 CPU의 프로그램 제어, CPU상태제어, 입출력제어를 위해 PC용 프로그래밍 S/W (GPC5, WGPC 등), 휴대용 프로그래머와 완벽한 통신환경이 구축되어 있으며, 사용자의 편의에 따라 PLC를 효율적으로 제어할 수 있도록 통신환경이 공개되어 있습니다. 사용자는 아래에서 기술하는 통신규약 및 절차에 따라 주변통신기기 및 컴퓨터로서 원거리(RS-232C방식으로 약15m이내 또는 RS-485방식으로 최대 1.2Km이내)에 설치된 PLC를 용이하게 제어할 수 있습니다.

3.1.2 통신 제어 사양

통신을 위한 기본사양은 인터페이스 및 전송속도를 조정할 수 있으며, 기타 통신방식은 표와 같이 정의되어 있습니다.

항 목	사 양
인터페이스	RS-232C/RS-485 1포트
전송 속도	4800/ 9600/ 19200/38400 bps (CCU+ 유니트에 있는 DIP 스윗치로 조정)
통신 방식	반 2중 비동기방식 (Half DUPLEX Asynchronous)
동기 방식	Polling 방식
전송 코드	Binary (HEX)
전송데이타 포맷	데이타 길이 : 8bit
	STOP BIT : 1bit
	Parity : 없음 (NO Parity)
에러 체크 방식	CRC-16 방식
데이타 전송 순서	Byte 단위로 하위 바이트 부터 전송
전송 케이블	Twisted Pair Cable

3.2 통신 규약(Protocol)

3.2.1 통신 규약의 개요

N plus계열의 CPU는 통신을 위한 Protocol을 2가지 방법으로 제공합니다.
Master(PC등)에서 통신절차를 Q(Query), QA(Query Acknowledge), RR(Response Request), R(Response)등을 거쳐 완성시키는 4단계 통신이 있고, Master에서 Q를 보내면 즉시 R로 회신을 하는 2단계 통신을 모두 지원합니다. 이 통신의 단계는 기능코드에 따라 달라집니다.

3.2.2 통신 규약의 단계

1단계 Q	Query

통신을 원하는 PLC의 고유번호와 Function Code를 설정하여 주변장치(Master)에서 PLC로 보내는 신호

2단계 QA	Query Acknowedge

해당 고유번호의 PLC가 주변장치(Master)로부터 Q신호를 정상적으로 수신했다는 인식신호로써 PLC에서 주변기기(Master)로 회신하는 신호.

3단계 RR	Response Request

주변장치(Master)가 PLC에서 보낸 QA신호를 정상적으로 인색했다는 인식신호로써 주변장치에서 PLC로 보내는 신호.
QA 신호가 정상적인 경우에 보내는 신호.

4단계 R	Response

PLC는 주변장치가 보낸 RR을 정상적으로 수신했을 경우 주변장치와의 통신이 원활하다고 판단하여 주변장치에서 최초에 보낸 Q에 실려 있었던 Function Code에 대한 처리를 마치고 그 결과를 주변장치로 보내주는 신호. PLC에서 R을 보냄으로써 한 Function Code에 대한 통신 사이클을 종료한다.

3.2.3 통신 지연시간의 대응

PLC는 Q나 RR을 수신한 뒤 일정시간내에 반드시 다음 신호를 되돌려 주게 된다. 그러나 통신선로 이상, CRC값, 통신속도 불일치 등으로 PLC가 주변장치부로터 신호를 받지 못하는 경우가 발생할 수도 있다. 이때 주변장치는 Q나 RR에 대한 응답을 기다리다가 3초 이내에 응답이 없으면 통신에 이상이 생긴 것으로 간주하고 다시 Q나 RR을 송신하여야 한다.

3.2.4 PLC CPU의 고유번호

N plus 계열의 PLC CPU 에는 항상 고유번호를 가지고 있다. 이것은 CPU가 별도의 장치없이 RS-485 통신망을 구성할 수 있게 된다. 통신망에 연결된 모든 장치들은 각각의 통신을 위한 고유번호가 할당되어져야 하며 통신시에 그 고유번호와 통신이 이루어져야 한다.
N plus Series는 고유번호를 0~191까지 사용자가 임의로 설정하여 사용할 수 있다.

그러나, 한번 지정된 번호를 다른 CPU모듈의 고유번호로 중복 지정하여 통신망을 형성하면 안된다. 하나의 CPU모듈과 하나의 주변장치로 연결된 경우에 고유번호는 통상적으로 0, 1 또는 255로 지정되지만 여러대의 CPU모듈이 하나의 통신망에 연결되어 있으면 서로 다른 ID를 사용하여야 한다.
단, 고유번호를 255로 지정하면 모든 CPU에서 회신을 할 수 있는 상태가 되며, ID를 지정하지 않는 것과 동일하다. 즉, 통신망에 연결된 모든 CPU모듈과 통신하도록 할당된 번호이다. 그러나, N-plus Series는 어느 순간에 두대 이상의 CPU모듈과 동시에 통신을 할수 없으므로 255번 ID를 동시에 두대 이상의 CPU모듈의 ID로 지정하여 사용하면 통신에러가 발생한다. 두대 이상의 CPU모듈과 통신하고자 할 때는 반드시 다른 ID를 사용하여야 한다.

고유번호는 S/W(GPC5,WinGPC등) 또는 주변장치로 지정할 수 있으며, 래더 프로그램으로도 지정할 수 있고, Link 모듈을 사용할 경우에는 Link모듈의 ID번호를 따르게 된다.

3.3 통신 규약의 단계

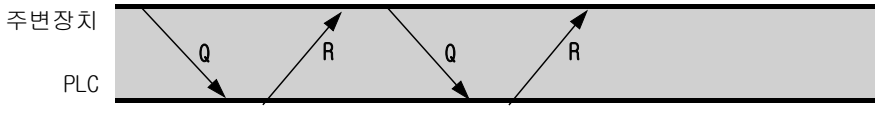
3.3.1 통신 단계

N plus CPU에서 2단계와 4단계 통신을 지원하고 있으며, 2단계와 4단계 통신의 구별은 Q(Query) Frame에 기능코드(Function Code)를 구별하여 전송함으로써 쉽게 구현할 수 있습니다. 2단계통신이나 4단계통신일지라도 반복되는 기능에 대해서는 2단계로 통신을 수행할 수 있습니다. 반복되는 기능이란 Query로 송신했던 프래임을 다시 수행하고자 할 경우로 RR 신호만을 반복적으로 송신하여 결과를 수신하는 것을 말한다. 이것은 4단계 통신에서 빠른 데이터 수신을 위해 사용할 때 자주 이용된다.

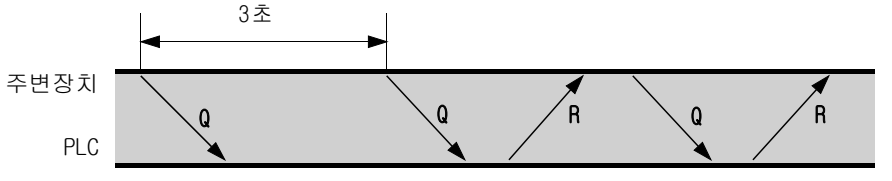
3.3.2 2단계 통신방법

2단계 통신은 N plus CPU에서 지원되며, 유사한 기종의 SPC시리즈에는 제공이 되지 않습니다.
이 방식을 사용하면 기본적으로 Q신호 송신에 대한 R신호 회신이 이루어져 데이터를 빠르게 교환할 수 있습니다.
즉, 2단계 구성 : Q(1단계) → R(2단계)
반복기능 코드 : Q(1단계) → R(2단계) → RR(1단계) → R(2단계) .. 등으로 구성할 수 있다.
단, 반복기능 코드는 일부 CPU에서 제공되지 않으며, Q신호로 전송하면 동일한 속도가 가능하다.

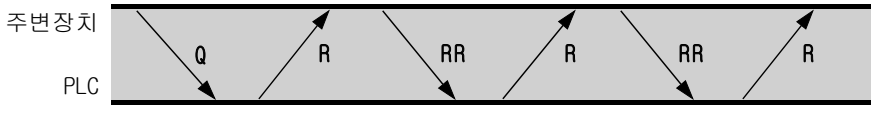
1) 통신에러가 없는 경우



2) R이 수신되지 않는 경우



3) 반복되는 Function Code에 대한 응답



3.3.3 4단계 통신방법

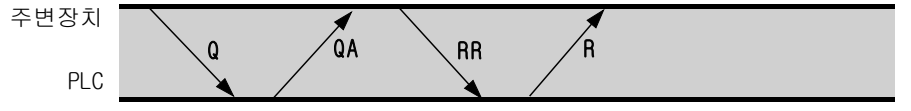
4단계 통신은 N plus CPU 및 SPC시리즈에도 제공이 되는 Protocol이며, 기본적으로 Q → QA → RR → R 의 4단계로 구성됩니다. 여기서 RR신호를 이용하여 반복되는 기능코드로 전송하면 2단계로 통신이 이루어 집니다.

즉, 4 단계 구성 : Q(1단계) → QA(2단계) → RR(3단계) → R(4단계)

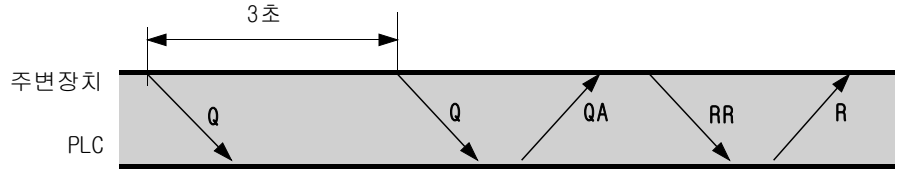
반복기능 코드 : Q(1단계) → QA(2단계) → RR(3단계) → R(4단계) → RR(1단계) → R(2단계)..

등으로 구성하여 제한적으로 빠른 회신을 얻을 수도 있다.

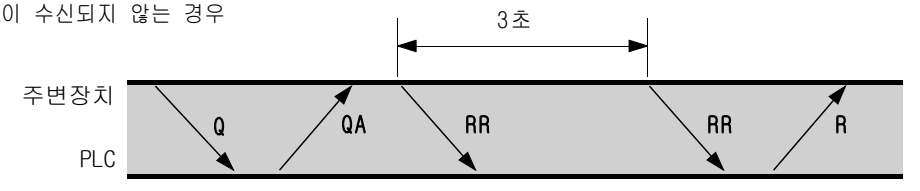
1) 통신에러가 없는 경우



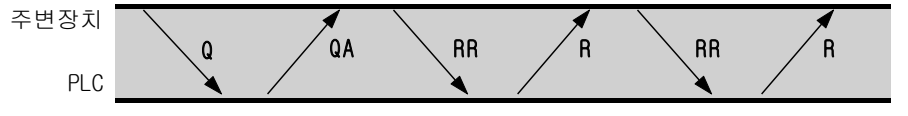
2) QA가 수신되지 않는 경우



3) R이 수신되지 않는 경우



4) 반복되는 Function Code에 대한 응답



3.4 통신 기능코드(Function Code)

3.4.1 쿼리(Query)에 실리는 기능코드

● 각각의 기능코드는 1바이트로 구성되며 PLC에서 정상적으로 쿼리(Q)신호를 수신했을때, R신호의 프레임에는 쿼리에서 보낸 기능코드에 $80(Hex)를 더하여 보낸다.
예를들면, PLC의 워드값을 읽는 FC(기능코드)인 $23을 Q신호를 통해 전송하면, R로 회신을 할때 $A3 ($80을 더한 값)의 FC로 회신이 된다.
2단계통신과 4단계의 통신의 차이점은 이 기능코드에 따라 다르게 회신하며, 쿼리나 Response 의 기능코드가 $20(Hex)의 차이가 있습니다.

● 주변기기는 자신이 보낸 쿼리를 PLC가 정상적으로 수신하였는지를 여부를 R 프레임의 Q Function Code를 검사하여 확인할 수 있다.

3.4.2 통신 기능코드

주) $표시는 Hex(16진수)표기를 나타냄

통신 기능	쿼리(Q)의 Function Code		Response(R)의 Function Code		비 고
	2단계	4단계	2단계	4단계	
비트 읽기	$21	$01	$A1	$81	상세설명 참조
비트 쓰기	$22	$02	$A2	$82	〃
워드 읽기	$23	$03	$A3	$83	〃
워드 쓰기	$24	$04	$A4	$84	〃
비트/워드 혼용 읽기	$25	$05	$A5	$85	〃
비트/워드 혼용 쓰기	$26	$06	$A6	$86	〃
프로그램 읽기	$27	$07	$A7	$87	
프로그램 쓰기	$28	$08	$A8	$88	
명령어 읽기	$29	$09	$A9	$89	
명령어 변경	$2A	$0A	$AA	$8A	
오퍼랜드 변경	$2B	$0B	$AB	$8B	
명령어 삽입	$2C	$0C	$AC	$8C	
명령어 삭제	$2D	$0D	$AD	$8D	
명령어 찾기	$2E	$0E	$AE	$8E	
오퍼랜드 찾기	$2F	$0F	$AF	$8F	
프로그램 전부/부분 삭제	$20	$10	$A0	$90	
NO SERVICE	$00	$00	$00(Hex)	$00(Hex)	

참고

● 통신프레임의 비트/워드 번지지정은 절대번지 방식으로 사용하므로 절대번지 변환표를 참조바랍니다.

● 프로그램 읽기, 쓰기등 기타 명기되지 않은 기능코드에 대한 자세한 내용은 당사 기술부서로 연락 바랍니다.

● Query란 ?
사전적인 의미로써 질문, 의문, 질문부호를 나타내며 통신에서 사용될 때는 사용자나 응용프로그램이 상대측에 어떠한 정보를 요구하는 것을 말합니다.

3.5 CPU의 절대 번지

■ 통신포트를 이용한 PLC 레지스터를 제어하거나 LDR, DLDR, STO, DSTO명령에서 레지스터의 번지를 간접지정 하여 프로그램 처리를 할 경우 절대번지를 이용합니다.

구분	레지스터 번지	절대번지(워드)	
		dec.	hex.
외부입출력	R0000	0000	$0000
	R0001	0001	$0001
	R0002	0002	$0002
	...	...	...
	R0126	0126	$007E
	R0127	0127	$007F
링크접점	L0000	0128	$0080
	L0001	0129	$0081
	L0002	0130	$0082
	...	...	...
	L0062	0190	$00BE
	L0063	0191	$00BF
내부접점	M0000	0192	$00C0
	M0001	0193	$00C1
	...	...	...
	M0063	0255	$00FF
	M0064	0256	$0100
	...	...	...
	M0127	0319	$013F
정전유지접점	K0000	0320	$0140
	K0001	0321	$0141
	K0002	0322	$0142
	...	...	...
	K0126	0446	$01BE
	K0127	0447	$01BF
내부특수접점	F0000	0448	$01C0
	F0001	0449	$01C1
	...	...	...
	F0014	0462	$01CE
	F0015	0463	$01CF

구분	레지스터 번지	절대번지(워드)	
		dec.	hex.
T/C 접점	TC00~15	0464	$01D0
	TC16~31	0465	$01D1
	...	...	...
	TC240~255	0479	$01DF
데이타영역	W0000	0512	$0200
	W0001	0513	$0201
	...	...	...
	W1024	1536	$0600
	...	...	...
	W2046	2558	$09FE
	W2047	2559	$09FF
T/C 설정치	SV000	2560	$0A00
	SV001	2561	$0A01
	...	...	...
	SV255	2815	$0AFF
T/C 현재치	PV000	2816	$0B00
	PV001	2817	$0B01
	...	...	...
	PV255	3071	$0BFF
상태표시	SR000	3072	$0C00
	SR001	3073	$0C01
	...	...	...
	SR256	3328	$0D00
	...	...	...
	SR511	3583	$0DFF
확장데이타영역 **주1)**	W3072	3584	$0E00
	...	...	...
	W4096	4608	$1200
	...	...	...
	W5119	5631	$15FF

주1) 확장 데이터 영역은 NX-CPU700P기종에 적용된 확장된 2048워드의 절대번지입니다.

참고)

- 통신기능에서 사용하는 **비트절대번지**는 워드절대번지와 해당워드의 비트번호 (0~15 or $0~$F)로 구성됩니다.

1워드표시

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
워드 절대번지												비트번호			

예) K127.12 내부출력에 대한 비트절대번지는 $1BFC(Hex)입니다.
("워드 절대번지=$01BF" + "비트번호=$C" = $1BFC)

3.6 에러 체크 방식

3.6.1 CRC-16

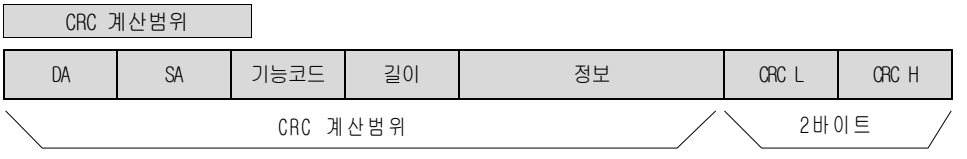
● 통신의 신뢰성 확보를 위해 에러를 체크하여 오류를 최소화 해야 합니다.
N plus 에서는 CRC-16방식(Cyclic Redundancy Checking, 순환중복검사)으로 에러를 점검하여 마지막 2Byte에 연결하여 전송한다.

● CRC 는 통신 프레임(Frame)이 에러없이 전송되었느지 여부를 확인하기 위하여 송신 프레임에 붙이는 일종의 체크섬 코드(Check Sum Code)정보로써 2바이트로 구성된다.

● 송/수신측은 수신데이타를 1바이트 송/수신할 때마다 CRC를 계산하여야 하므로 통신프로그램 작성시 CRC계산에 많은 시간이 소요되므로 통신속도 개선 및 통신에러를 방지하기 위해서는 이 부분의 속도를 향상시키도록 염두해 두어야 한다.

3.6.2 CRC-16의 계산

● CRC-16 은 통신 메세지의 전 블록(DA부터 정보의 마지막 데이터까지)를 직렬로 연결시켜 이 데이터를 정해진 17비트의 2진수(1 1000 000 000 0101)로 나눗셈한 나머지 2Byte값 입니다.



□ C++로 작성한 CRC-16계산 프로그램 예제 1

```
/*--------------------------------------------------------------------*/
// CRC-16 Routine with traditional method
// method 1. No need CRC table
/*--------------------------------------------------------------------*/
void CRC16_Test(int count)
{
    int i;

    Crc = 0xffff;
    for(i=0; i<count; i++)
    {
        CrcOld = Crc;
        Crc = Crc ^ (Data[i] & 0x00FF);
        for(int bit=0; bit<8; bit++)
        {
            if((Crc & 0x0001) == 0x0001)
                Crc = (Crc>>1) ^ 0xA001;
            else
                Crc = Crc>>1;
        }
        PRINT_CRC_JOBPROCESS(i, Data[i]);
    }
}
```

□ C++로 작성한 CRC-16계산 프로그램 예제 2 (Table을 이용한 예)

```
/*--------------------------------------------------------------------*/
// CRC-16 Routine with CrcTable
// method 2. Need CRC table
/*--------------------------------------------------------------------*/
void CRC16_TestFast(int count)
{
    int i;
    unsigned int Temp;

    Crc = 0xffff;
    for(i=0; i<count; i++)
    {
        CrcOld = Crc;
        Temp = Crc ^ Data[i];
        Crc = (Crc>>8) ^ Crc16Table[Temp&0x00ff];
        PRINT_CRC_JOBPROCESS(i, Data[i]);
    }
}
```

Table 파일 (CrcTest.h)

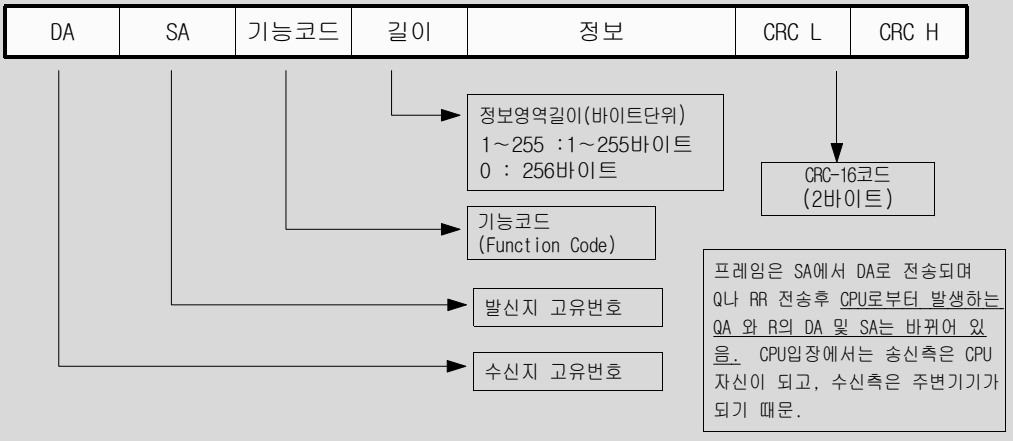
```
unsigned int Crc, CrcOld;

const unsigned int Crc16Table[256] =
{
    0x0000,0xc0c1,0xc181,0x0140,0xc301,0x03c0,0x0280,0xc241,
    0xc601,0x06c0,0x0780,0xc741,0x0500,0xc5c1,0xc481,0x0440,
    0xcc01,0x0cc0,0x0d80,0xcd41,0x0f00,0xcfc1,0xce81,0x0e40,
    0x0a00,0xcac1,0xcb81,0x0b40,0xc901,0x09c0,0x0880,0xc841,
    0xd801,0x18c0,0x1980,0xd941,0x1b00,0xdbc1,0xda81,0x1a40,
    0x1e00,0xdec1,0xdf81,0x1f40,0xdd01,0x1dc0,0x1c80,0xdc41,
    0x1400,0xd4c1,0xd581,0x1540,0xd701,0x17c0,0x1680,0xd641,
    0xd201,0x12c0,0x1380,0xd341,0x1100,0xd1c1,0xd081,0x1040,
    0xf001,0x30c0,0x3180,0xf141,0x3300,0xf3c1,0xf281,0x3240,
    0x3600,0xf6c1,0xf781,0x3740,0xf501,0x35c0,0x3480,0xf441,
    0x3c00,0xfcc1,0xfd81,0x3d40,0xff01,0x3fc0,0x3e80,0xfe41,
    0xfa01,0x3ac0,0x3b80,0xfb41,0x3900,0xf9c1,0xf881,0x3840,
    0x2800,0xe8c1,0xe981,0x2940,0xeb01,0x2bc0,0x2a80,0xea41,
    0xee01,0x2ec0,0x2f80,0xef41,0x2d00,0xedc1,0xec81,0x2c40,
    0xe401,0x24c0,0x2580,0xe541,0x2700,0xe7c1,0xe681,0x2640,
    0x2200,0xe2c1,0xe381,0x2340,0xe101,0x21c0,0x2080,0xe041,
    0xa001,0x60c0,0x6180,0xa141,0x6300,0xa3c1,0xa281,0x6240,
    0x6600,0xa6c1,0xa781,0x6740,0xa501,0x65c0,0x6480,0xa441,
    0x6c00,0xacc1,0xad81,0x6d40,0xaf01,0x6fc0,0x6e80,0xae41,
    0xaa01,0x6ac0,0x6b80,0xab41,0x6900,0xa9c1,0xa881,0x6840,
    0x7800,0xb8c1,0xb981,0x7940,0xbb01,0x7bc0,0x7a80,0xba41,
    0xbe01,0x7ec0,0x7f80,0xbf41,0x7d00,0xbdc1,0xbc81,0x7c40,
    0xb401,0x74c0,0x7580,0xb541,0x7700,0xb7c1,0xb681,0x7640,
    0x7200,0xb2c1,0xb381,0x7340,0xb101,0x71c0,0x7080,0xb041,
    0x5000,0x90c1,0x9181,0x5140,0x9301,0x53c0,0x5280,0x9241,
    0x9601,0x56c0,0x5780,0x9741,0x5500,0x95c1,0x9481,0x5440,
    0x9c01,0x5cc0,0x5d80,0x9d41,0x5f00,0x9fc1,0x9e81,0x5e40,
    0x5a00,0x9ac1,0x9b81,0x5b40,0x9901,0x59c0,0x5880,0x9841,
    0x8801,0x48c0,0x4980,0x8941,0x4b00,0x8bc1,0x8a81,0x4a40,
    0x4e00,0x8ec1,0x8f81,0x4f40,0x8d01,0x4dc0,0x4c80,0x8c41,
    0x4400,0x84c1,0x8581,0x4540,0x8701,0x47c0,0x4680,0x8641,
    0x8201,0x42c0,0x4380,0x8341,0x4100,0x81c1,0x8081,0x4040
};
```

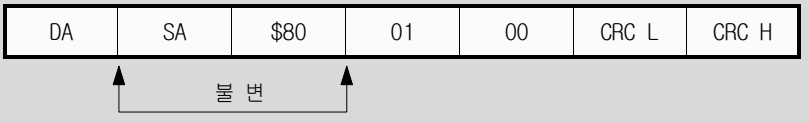
3.7 통신 프레임의 구조

● 기능코드는 2,4단계통신을 기준으로 Query 및 Response Frame의 예제를 설명하였습니다.

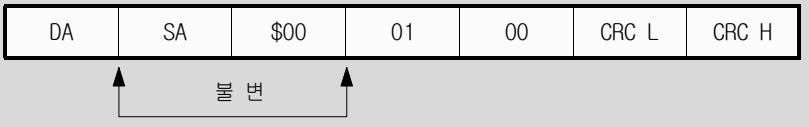
Q(Query) 및 R(Response) 프레임



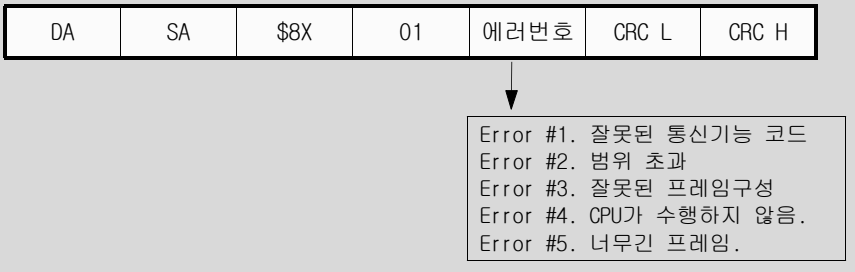
QA(Query Acknowledge) 프레임



RR(Response Request) 프레임



에러발생시 R(Response) 프레임



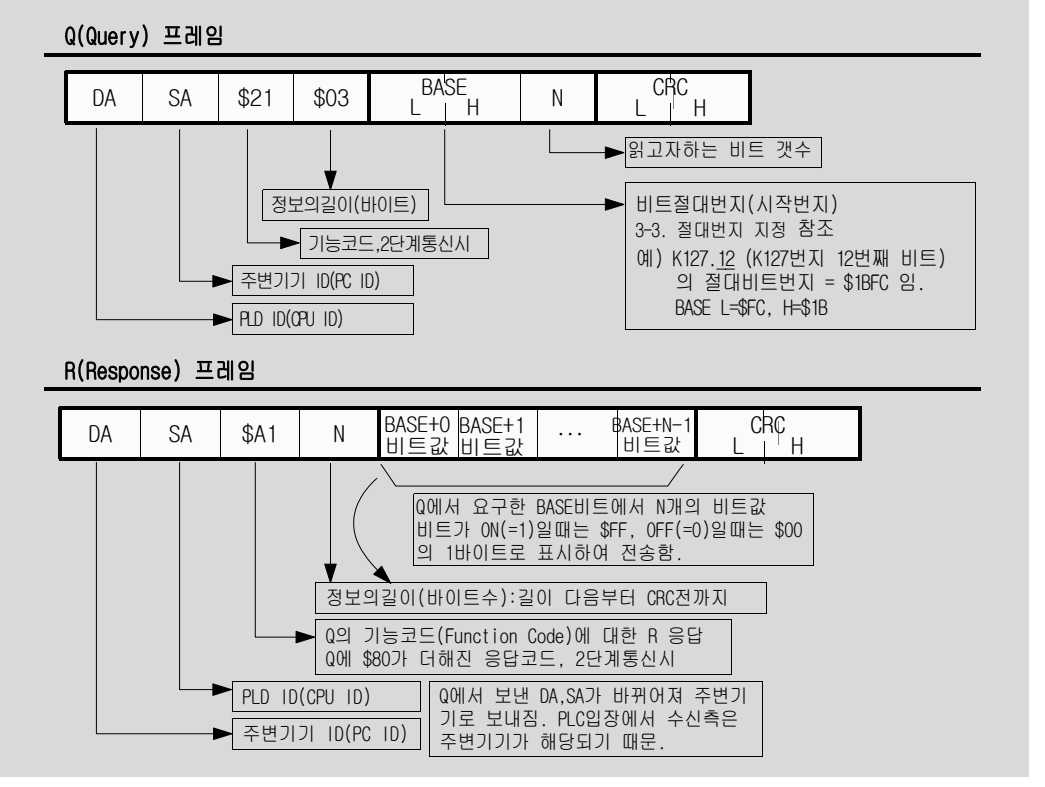
● 여기서 DA는 목적지의 ID번호, SA는 보내는 곳의 ID번호를 뜻하며, 먼저 보내는 Q의 경우 DA는 PLC의 ID번호이며, SA는 PC의 ID번호이다. 다시 회신하는 QA는 DA가 PC이며, PLC는 SA가 된다. 그러므로 주고받는 PLC의 ID번호에 맞게 프로그램을 해야 한다. 각각의 영역별로 1바이트씩(2 Digits) 값이 표시되어 있고, 정보 영역은 지정한 바이트에 따라 길이가 달라질 수 있다.

● 길이는 바로뒤에 있는 정보영역의 길이를 말하며, 00~FF까지 바이트 단위를 말하며, 256바이트까지 설정할 수 있다.

3.8 통신 Protocol 예제

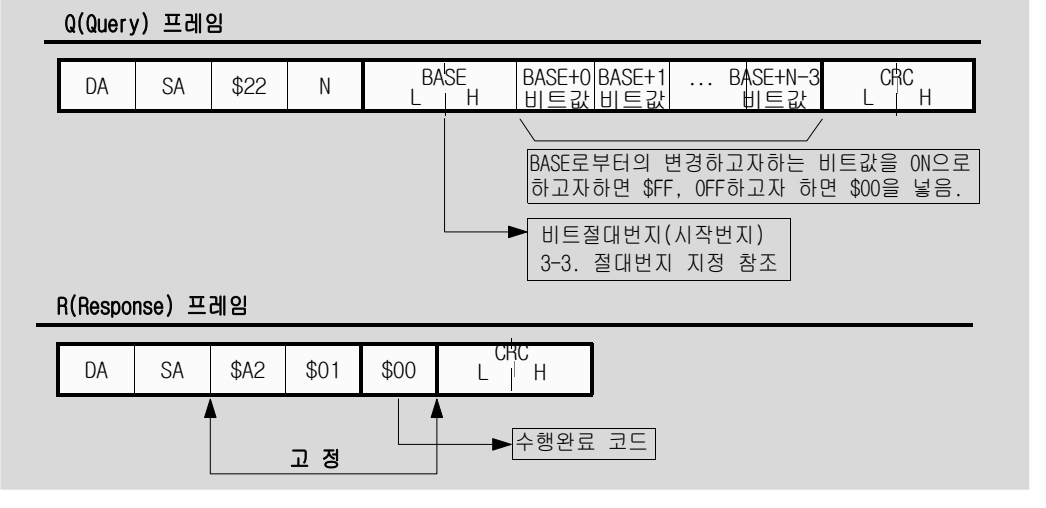
3.8.1 비트읽기

- 절대번지로 지정된 비트(R, L, M, K, F, TC)의 내용을 읽음.
- 연속하는 N개의 비트내용(ON/ OFF)를 읽을 수 있음.



3.8.2 비트쓰기

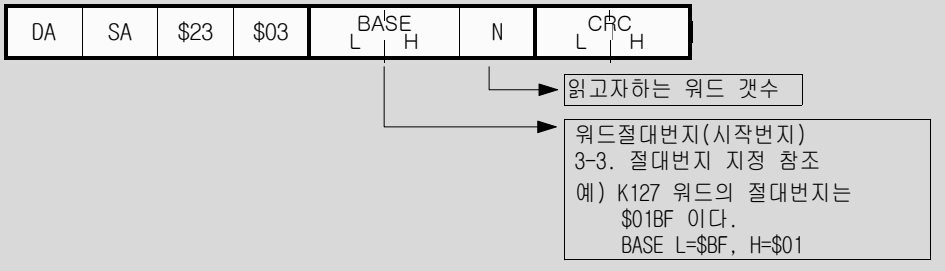
- 절대번지로 지정된 비트(R, L, M, K, F, TC)의 내용을 변경함. ON/ OFF를 변경함.
- 연속하는 여러개의 비트 내용를 변경할 수 있음.



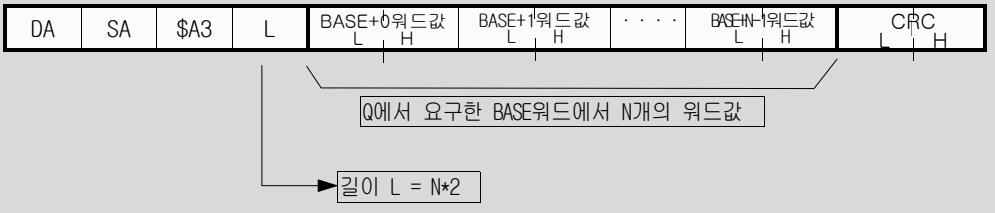
3.8.3 워드 읽기

- 절대번지로 지정된 워드(R, L, M, K, F, W)의 내용을 읽음.
- 연속하는 N개의 워드 내용을 읽을 수 있음.

Q(Query) 프레임



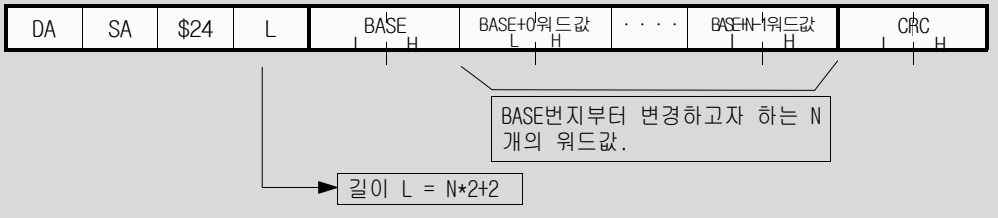
R(Response) 프레임



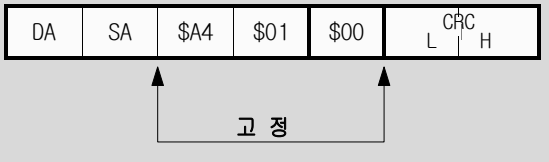
3.8.4 워드 쓰기

- 절대번지로 지정된 워드(R, L, M, K, F, W)의 내용을 변경함.
- 연속하는 N개의 워드 내용을 변경할 수 있음.

Q(Query) 프레임

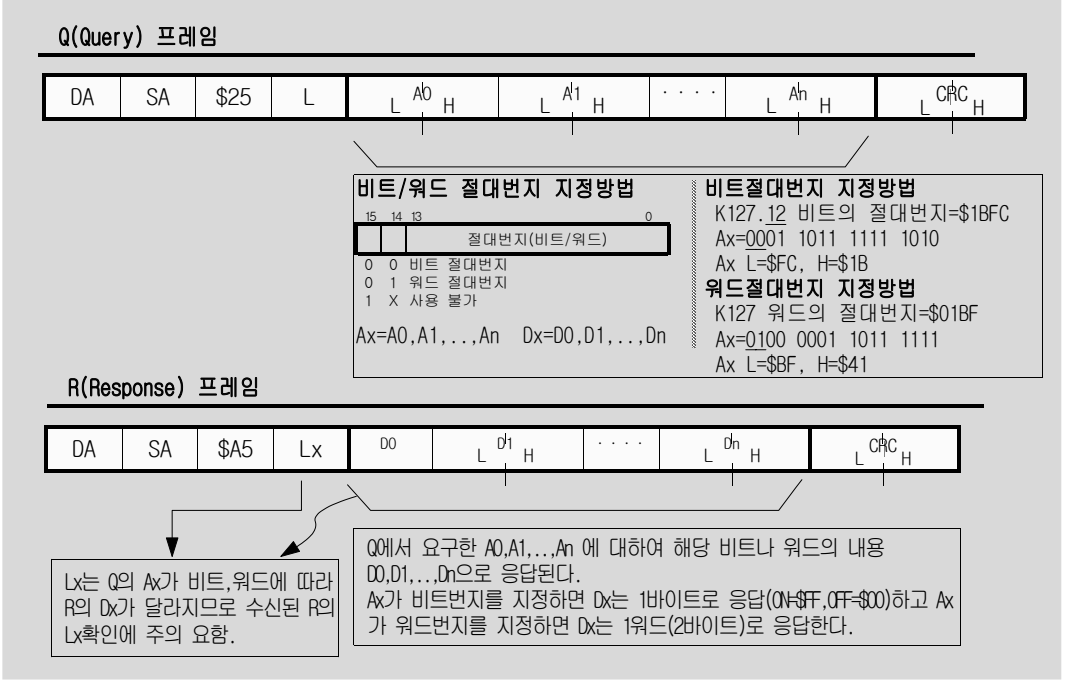


R(Response) 프레임



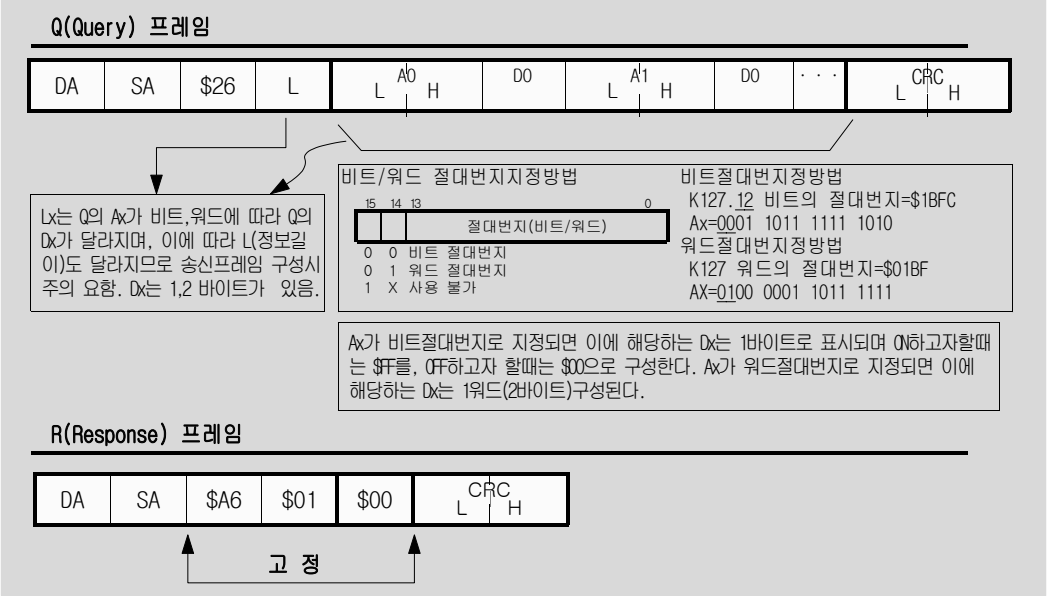
3.8.5 비트/워드 혼용하여 읽기

- 절대번지로 지정된 비트나 워드내용를 읽음.
- 비트와 워드를 순서없이 혼용하여 읽을수 있음.



3.8.6 비트/워드 혼용하여 쓰기

- 절대번지로 지정된 비트나 워드내용를 변경함.
- 비트와 워드를 순서없이 혼용하여 변경할수 있음.



3.9 통신 프로그램 예제

- 본 예제는 사용자가 직접 통신프로그램을 작성을 위한 예제입니다.
- 본 예제는 C++로 작성된 통신 프로그램이며, Visual Basic 또는 다른 프로그램 예제는 문의 바랍니다.

프 로 그 램	설 명
`` // // NpDemo.cpp : Defines the entry point for the console application. // // #include "NpDemo.h" #include <stdio.h> #include <conio.h> int main(int argc, char* argv[]) { printf("***Wn"); printf("* N-plus series PLC Communication Test Program *Wn"); printf("***Wn"); //-----Serial port initialization printf("WnReady serial port init... "); if(OnlineConnection(1)) { printf("Port open success!Wn"); } else { CloseHandle(hPort); printf("Port open error!Wn"); } //-----COMMUNICATION TEST if(Test_Read()!=CS_END) printf("Test_Read() failed!Wn"); DsplPacket(); if(Test_Write()!=CS_END) printf("Test_Write() failed!Wn"); DsplPacket(); //-----End process if(hPort!=NULL) { if(CloseHandle(hPort)) printf("WnPort close success!Wn"); else printf("WnPort close failed!Wn"); } return 0; } /*--*/ // N-plus READ_WORD TEST // R0 : 1 word read /*--*/ BYTE Test_Read() { int i=0; Tx_Data.Data[i++] = 0xFF; Tx_Data.Data[i++] = 0xE1; Tx_Data.Data[i++] = FC_READ_WORD_2STEP; Tx_Data.Data[i++] = 0x03; Tx_Data.Data[i++] = 0x00; Tx_Data.Data[i++] = 0x00; Tx_Data.Data[i++] = 0x01; CRC16(Tx_Data.Data, i++); i--; Tx_Data.Data[i++] = LOBYTE(Crc); Tx_Data.Data[i++] = HIBYTE(Crc); Tx_Data.Size = i; RRA_LengthCalc(FC_READ_WORD_2STEP); printf("WnREAD_WORD Function Code TestWn"); return StateMachine(); } /*--*/ // N-plus READ_WORD TEST // M0~M4 : 5 word write /*--*/ ```	COM1포트 접속 포트 접속이 비정상적인 경우 포트를 닫고 종료 테스트용 워드단위 읽기 함수 호출 결과출력 테스트용 워드단위 쓰기 함수 호출 결과출력 테스트후 사용했던 포트를 닫아 주어 다른 응용프로그램에서 사용할 수 있도록 한다 받는 측 (PLC) ID 지정 보내는 측 (PC) ID 지정 2단계 통신용 워드읽기 통신 코드 정보 길이 지정 읽기 시작 어드레스L. 절대번지로 지정한다 읽기 시작 어드레스H 읽고자 하는 워드 개수 CRC-16 계산 CRC결과L CRC결과H PLC로부터 수신될 개수 계산 실제 데이터 송수신부 호출

프 로 그 램	설 명
84 BYTE Test_Write() 85 { 86 int i=0, j, WordNumber=5; 87 88 Tx_Data.Data[i++] = 0xFF; 89 Tx_Data.Data[i++] = 0xE1; 90 Tx_Data.Data[i++] = FC_WRITE_WORD_2STEP; 91 Tx_Data.Data[i++] = WordNumber*2+2; 92 Tx_Data.Data[i++] = LOBYTE(0xc0); //0xc0 is M0 address 93 Tx_Data.Data[i++] = HIBYTE(0xc0); 94 95 for(j=0; j<WordNumber; j++) 96 { 97 Tx_Data.Data[i+j*2+0] = rand(); 98 Tx_Data.Data[i+j*2+1] = rand(); 99 } 100 i += WordNumber*2; 101 CRC16(Tx_Data.Data, i++); 102 i--; 103 Tx_Data.Data[i++] = LOBYTE(Crc); 104 Tx_Data.Data[i++] = HIBYTE(Crc); 105 Tx_Data.Size = i; 106 107 RRA_LengthCalc(FC_WRITE_WORD_2STEP); 108 printf("WnWRITE_WORD Function Code TestWn"); 109 return StateMachine(); 110 } 111 113 /*--*/ 114 // Query Acknwoledgement 115 /*--*/ 116 bool Q_Process() 117 { 118 BOOL bRet; 119 DWORD dwWritten, dwErr; 120 COMSTAT CommState; 121 122 printf("Q : "); 123 ClearCommError(hPort, &dwErr, &CommState); 124 PurgeComm(hPort, MY_PURGE_EVENT); 125 bRet = WriteFile(hPort, 126 Tx_Data.Data, 127 Tx_Data.Size, 128 &dwWritten, 129 NULL); 130 131 if(!bRet) return false; 132 133 while(dwWritten!=Tx_Data.Size); 134 135 return true; 136 } 137 138 /*--*/ 139 // Request Response Acknwoledgement 140 /*--*/ 141 bool RRA_Process() 142 { 143 BOOL bRet; 144 DWORD dwReaded=0, dwRdTotal=0, dwErr, StartTime; 145 COMSTAT CommState; 146 147 printf("RRA: "); 148 ClearCommError(hPort, &dwErr, &CommState); 149 PurgeComm(hPort, MY_PURGE_EVENT); 150 StartTime = GetTickCount(); 151 152 while((StartTime+1000)>GetTickCount()) 153 { 154 bRet = ReadFile(hPort, 155 (Rx_Data.Data+dwRdTotal), 156 Rx_Data.Size-dwRdTotal, 157 &dwReaded, 158 NULL); 159 if(!bRet) return false; 160 if(dwReaded!=0) 161 { 162 dwRdTotal+=dwReaded; 163 if(dwRdTotal==Rx_Data.Size) 164 return true; 165 } 166 } 167 168 return false; 169 }	받는 측 (PLC) ID 지정 보내는 측 (PC) ID 지정 2단계 통신용 워드쓰기 통신 코드 정보 길이 지정 쓰기 시작 어드레스L. 절대번지로 지정한다 쓰기 시작 어드레스H 실제 쓰고자 하는 값L 대입 실제 쓰고자 하는 값H 대입 쓰고자 하는 워드 개수 CRC-16 계산 CRC결과L CRC결과H PLC로부터 수신될 개수 계산 실제 데이터 송수신부 호출 새로운 데이터를 송수신하기 위하여 포트가 이전에 에러가 발생했는지, 쓰레기값들이 있는지 검사하고 이를 클리어 함. 포트와 데이터 포인터, 개수를 지정하고 WriteFile을 호출하면 포트로 데이터가 출력됨. 이때 출력된 개수를 dwWritten 로 리턴해주므로, 이것을 검사하면 실제 데이터가 정상적으로 출력되었는지 확인 할 수 있음. PLC로 데이터 전송(Q)후 1000ms 이내에 응답이 없을경우 통신 무시. PLC에서 송신하는 데이터를 PC에서 수신함. Rx_Data.Size=읽고자하는 총갯수는 앞서 결정됨. 1000ms 동안 수신되는 데이터를 계속 읽어내어 원하는 개수만큼 데이터을 얻어 냄.

프 로 그 램	설 명
/*---*/ // Win32 Serial Communication Port Open // method 1. No need CRC table /*---*/	
bool OnlineConnection(int PortNumber) {	포트 접속 루틴
char cpName[256]; DCB CommDCB; BOOL fRet;	
hPort = NULL; sprintf(cpName,"COM%d", PortNumber);	포트이름 지정(95/98과 NT/2000/XP의 경우 포트이름 지정방법이 다르므로 API 설명서를 참조한다.)
hPort = CreateFile(cpName, GENERIC_READ|GENERIC_WRITE, 0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);	본예제에서는 NON_OVERRAPPED 비동기식 방식을 선택하였음. 쓰레드를 사용하여 통신프로그램을 구현할경우 오버랩드 타입으로 지정하여야 함.
if(hPort == INVALID_HANDLE_VALUE) return FALSE;	
// set buffer size fRet = SetupComm(hPort, MAX_FRAME_LEN, MAX_FRAME_LEN); if(!fRet) return FALSE;	지정된 포트가 사용할 버퍼크기를 지정.
// device clear PurgeComm(hPort, MY_PURGE_EVENT); if(!fRet) return FALSE;	포트상태 초기화.
// Comm timeout set COMMTIMEOUTS CommTimeOuts; CommTimeOuts.ReadIntervalTimeout = 1; CommTimeOuts.ReadTotalTimeoutMultiplier = 5; CommTimeOuts.ReadTotalTimeoutConstant = 100; CommTimeOuts.WriteTotalTimeoutMultiplier = 5; CommTimeOuts.WriteTotalTimeoutConstant = 100; fRet = SetCommTimeouts(hPort, &CommTimeOuts); if(!fRet) return FALSE;	통신 타임아웃값을 설정. 자세한 내용은 API 설명서를 참조한다.
// DCB Block Set CommDCB.DCBlength = sizeof(DCB) ; fRet = GetCommState(hPort, &CommDCB) ; if(!fRet) return FALSE; CommDCB.BaudRate = CBR_9600; CommDCB.Parity = NOPARITY; // Parity Setting CommDCB.ByteSize = 8; // Data Size Setting CommDCB.StopBits = ONESTOPBIT; // Stop Bit Setting fRet = SetCommState(hPort, &CommDCB); if(!fRet) return FALSE;	보드레이트, 패리티비트, 스톱비트등 포트 파라미터 설정. 재설정시 까지 현재 상태 유지됨. 보드레이트는 가변가능. 고정: Plus 통신규약 임 고정: Plus 통신규약 임 고정: Plus 통신규약 임
return TRUE; } /*---*/ // CRC-16 Routine with traditional method // method 1. No need CRC table /*---*/	
void CRC16(BYTE *src, int count) { int i; Crc = 0xffff; for(i=0; i<count; i++) { CrcOld = Crc; Crc = Crc ^ (*(src+i) & 0x00FF); for(int bit=0; bit<8; bit++) { if((Crc & 0x0001) == 0x0001) Crc = (Crc>>1) ^ 0xA001; else Crc = Crc>>1; } } }	CRC 테이블에 의존하지 않은 고전적인 CRC-16 계산 루틴
/*---*/ // Recieve length calcuration /*---*/	
void RRA_LengthCalc(BYTE fc) { switch(fc) {	Q 전송시 PLC로부터 수신될 개수는 결정되는데 이것을 계산하기 위한 루틴임.

프 로 그 램	설 명
257 case FC_READ_WORD_2STEP: 258 Rx_Data.Size = 6+Tx_Data.Data[6]*2; //6=4(header)+2(Crc) 259 break; 260 261 case FC_WRITE_WORD_2STEP: 262 Rx_Data.Size = 7; //7=4(header)+1(info)+2(Crc) 263 break; 264 265 default: 266 break; 267 } 268 269 }	PLC 가 송신하게 될 데이터 내용과 길이는 통신규약을 참조한다. PLC 가 송신하게 될 데이터 내용과 길이는 통신규약을 참조한다.
270 271 /*----------*/ 272 // N-plus protocol establish 273 /*----------*/ 274 BYTE StateMachine() 275 {	통신 프로토콜 구현부
276 bool bRet; 277 278 //-----Q 279 bRet = Q_Process(); 280 if(!bRet) { printf("TX error!Wn"); 281 return CS_ERR; 282 } 283 printf("%d bytes transmit success!Wn", Tx_Data.Size); 284 285 //-----RRA 286 bRet = RRA_Process(); 287 if(!bRet) { printf("%d bytes received error!Wn", Rx_Data.Size); 288 return CS_ERR; 289 } 290 printf("%d bytes received success!Wn", Rx_Data.Size); 291 292 //-----packet verify 293 bRet = VerifyFrame(); 294 if(!bRet) { printf("Received packet informaton invalid!Wn"); 295 return CS_ERR; 296 } 297 printf("Received packet informaton success!Wn"); 298 299 return CS_END; 300 }	Q 과정 RRA 과정 수신데이터 검사 과정
301 302 /*----------*/ 303 // Packet verify 304 /*----------*/ 305 bool VerifyFrame() 306 {	수신데이터 검사부 프레임의 끝 2바이트에 실려있는 CRC값과 수신된 데이터의 CRC16값을 계산한 결과가 일치하면 수신된 데이터가 유효하다고 볼수 있음.
307 DWORD rxCrc; 308 309 310 rxCrc = (DWORD)(Rx_Data.Data[Rx_Data.Size-2] + 311 Rx_Data.Data[Rx_Data.Size-1]*0x100); 312 CRC16(Rx_Data.Data, Rx_Data.Size-2); 313 314 if((Tx_Data.Data[0]==Rx_Data.Data[1]) && 315 ((Tx_Data.Data[2]+0x80)==Rx_Data.Data[2]) && 316 (Crc==rxCrc)) 317 return true; 318 else 319 return false; 320 }	또한, 수신된 데이터가 Q에서 지정한 PLC로부터 온 데이터 인지를 RRA의 헤더부를 검사하면 알수 있다. (RS-485 통신을 하는 경우, 여러대의 PLC가 연결되어 있으므로 반드시 보낸쪽 ID를 검사하여야 한다)
321 322 323 /*----------*/ 324 // Sent and Received packet data display 325 /*----------*/ 326 void DsplPacket() 327 {	
328 int i; 329 330 printf("TxPacket:"); 331 for(i=0; i<Tx_Data.Size; i++) 332 printf("%02X ",Tx_Data.Data[i]); 333 printf("WnRxPacket:"); 334 335 for(i=0; i<Rx_Data.Size; i++) 336 printf("%02X ",Rx_Data.Data[i]); 337 printf("Wn"); 338 }	송수신 데이터 표시부

● 통신 프로그램 예제 (Header File)

프 로 그 램	설 명
`` 1 // 2 // NpDemo.h : NpDemo source Header file 3 // 7 // 8 9 #include <windows.h> 10 11 //-----DEFINITION PART------------------------- 12 const unsigned int MAX_FRAME_LEN = 262; 13 const unsigned int MY_PURGE_EVENT = PURGE_TXABORT | PURGE_RXABORT | 14 PURGE_TXCLEAR | PURGE_RXCLEAR; 15 const BYTE CS_RDY = 0; 16 const BYTE CS_ING = 1; 17 const BYTE CS_END = 2; 18 const BYTE CS_ERR = 3; 19 20 const BYTE FC_READ_BIT_2STEP = 0x21; 21 const BYTE FC_WRITE_BIT_2STEP = 0x22; 22 const BYTE FC_READ_WORD_2STEP = 0x23; 23 const BYTE FC_WRITE_WORD_2STEP = 0x24; 24 const BYTE FC_READ_BITWORD_2STEP = 0x25; 25 const BYTE FC_WRITE_BITWORD_2STEP = 0x26; 26 27 struct Tx_RxData 28 { 29 public: 30 BYTE Data[MAX_FRAME_LEN]; 31 WORD Size; 32 }; 33 34 35 //-----VARIABLE PART--------------------------- 36 unsigned int Crc, CrcOld; 37 HANDLE hPort; 38 Tx_RxData Tx_Data,Rx_Data; 39 40 //-----FUNCTION PART--------------------------- 41 bool OnlineConnection(int PortNumber); 42 void CRC16(BYTE *src, int count); 43 bool Q_Process(); 44 bool RRA_Process(); 45 void RRA_LengthCalc(BYTE fc); 46 BYTE StateMachine(); 47 BYTE Test_Read(); 48 BYTE Test_Write(); 49 bool VerifyFrame(); 50 void DsplPacket(); ```	**262: N-plus 통신 규약상 한번에 송수신되는 최대 바이트 수임.** **통신 준비 상태** **통신중인 상태** **통신 완료 상태** **통신 에러 상태** **2단계통신용 비트 읽기 코드** **2단계통신용 비트 쓰기 코드** **2단계통신용 워드 읽기 코드** **2단계통신용 워드 쓰기 코드** **2단계통신용 비트/워드 혼합 읽기 코드** **2단계통신용 비트/워드 혼합 쓰기 코드** **데이터 수신을 위한 구조체 선언**