통신 규칙

- 송신을 함으로써 통신을 개시하는 쪽을 MASTER 라 한다.
- 최대 패킷 길이는 ENQ, ETX, LRC 를 포함하여 250Bytes 이다
  - 포인트개수가 '999'까지라고 되어 있으나 실제 Packet 의 최대 크기는 250byte 이므로 이를 넘지 않는 범위안의 값만 사용가능하다. 250 바이트를 넘게될 경우, 제어기 내부적으로 모든 데이터를 버리고 통신을 다시 시작하게 된다.
- 패킷을 받는 측(수신측)은 LRC 가 틀릴 경우 NAK 를 보내고 일정 횟수 이상이면 RST 를 보낸다. 디폴트 횟수는 3 회이다. NAK 를 받은 측은 다시 이전 패킷을 보내야 한다.
- RST 는 통신 종료를 의미하며 RST 를 수신한 측은 즉시 통신을 종료하고 통신 대기점으로 복귀한다.
- ACK 는 한개의 통신 패킷을 잘 받았다는 Acknowledgement(인식)의 의미로 사용된다. 한개의 통신 패킷이란 다음과 같이 ENQ,DATA,ETX,LRC 로 이루어진 구조를 말한다.

ENQ	DATA	ETX	LRC

  - Proface 터치가 Master 가 되어 1:N(Sigma 가 N, RS422 에서 N 은 최대 32)으로 연결될 경우, ENQ 다음에 ASCII 2 바이트로 ID 를 삽입한다. 예를 들어 ID 가 0 인 경우 '0','0', 15 인 경우 '1','5'가 된다.

ENQ	ID	DATA	ETX	LRC

- 통신 패킷의 기본 구조는 다음과 같다.

송신측	ENQ	DATA	ETX	LRC			
수신측					ACK	ETX	LRC

  - ACK 대신 다음과 같이 ENQ 와 FLAG 를 보낸다.

송신측	ENQ	DATA	ETX	LRC						ACK
수신측					ENQ	FLAG	(DATA)	ET X	LRC	

예) 'MA' 프로토콜(현재 상태 읽기)

  - Proface 터치에서는 ENQ 다음에 ID 를 추가하여 다음과 같이 된다.

송신측	ENQ	ID	DATA	ET X	LRC							ACK
수신측						ENQ	ID	FLAG	(DATA)	ET X	LRC	

(DATA)는 DATA 가 프로토콜에 따라 없을 수도 있음을 나타냅니다.

- 주요 컨트롤 ASCII 문자
  - ENQ : 0x05
  - ETX : 0x03
  - ACK : 0x06
  - NAK : 0x15
  - RST : 0x12
- LRC 계산법(ENQ 를 제외하고 ENQ 다음 바이트부터 ETX 까지의 exclusive-OR)

ENQ	DATA	ETX	LRC

DATA 가 N 바이트라고 하면 아래와 같이 나타낼 수 있습니다.

LRC = DATA[0]^DATA[1]^....^DATA[N-1]^ETX

  - 만약 위 계산에 의한 LRC 값이 0 이면 LRC 는 ETX 로 합니다.
- ACK 와 NAK 를 보낼 때에는 ETX 와 LRC 를 보내지 않는다.
- 최대 패킷을 넘는 연속된 데이터는 다음과 같이 다음 패킷에 연속된 데이터를 계속 보내면 된다.

1) 호스트에서 데이터를 요구한 경우

호스트					ACK					ACK
컨트롤러	ENQ	DATA	ETX	LRC		ENQ	연속된 데이터	ETX	LRC	

2) 호스트에서 컨트롤러로 데이터를 전송하는 경우

ACK 대신 OK 패킷이 사용됩니다.

호스트	ENQ	DATA	ETX	LRC		ENQ	연속된 데이터	ETX	LRC	
컨트롤러					OK					OK

OK	ENQ	'0'	ETX	LRC

연속된 패킷의 종료는 다음과 같은 종료 패킷을 보냄으로써 수행됩니다.

1) 호스트에서 데이터를 요구한 경우

호스트					ACK
컨트롤러	ENQ	'4'	ETX	LRC	

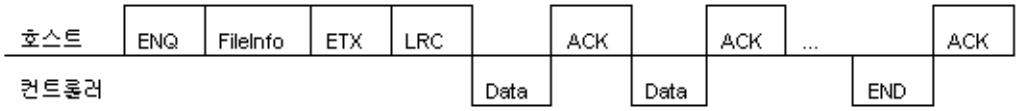
2) 호스트에서 컨트롤러로 데이터를 전송하는 경우

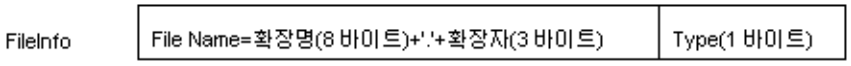
호스트	ENQ	'4'	ETX	LRC	
컨트롤러					ACK

* Proface 터치에서는 구현의 복잡성으로 위 형식을 사용하지 않는다.

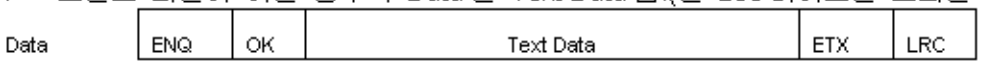
(연속된 패킷 예)

예 1) 파일 Upload(컨트롤러→호스트) 프로토콜('FH')

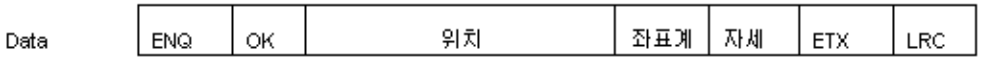




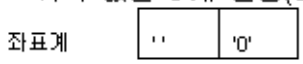
➢ 포인트 파일이 아닌 경우의 Data 는 Text Data 임.(단 250 바이트를 초과할 수 없음)



➢ 포인트 파일인 경우의 Data 는 아래 설명과 같음. 한번에 한 개씩 전송.

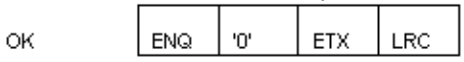


위의 값을 6 개 전송(6 축 데이터)



'0'=XY 좌표계, '1'=Joint 좌표계

'0'=스카라 오른손 형태, '1'=스카라 왼손 형태, 직교로봇인 경우에는 의미 없슴.



예 2) 파일 Download(호스트→컨트롤러) 프로토콜('FI')

목 차

1 프로토콜 일람표(기능별) 7

2 프로토콜 분류(이름별, 알파벳 순서) 9

3 FLAG 운용 방법 12

4 프로토콜 13

4.1 상태 보기 13

4.1.1 현재 상태 읽기 13

4.1.2 현재 속도 읽기 14

4.1.3 현재 위치 읽기 15

4.1.4 컨트롤러 에러 읽기(텍스트) 16

4.1.5 컨트롤러 에러 읽기(코드) 17

4.1.6 컨트롤러 운전 상태 읽기 18

4.1.7 위치 읽기(Proface 전용) 19

4.1.8 속도 읽기(Proface 전용) 20

4.2 모션 관련 21

4.2.1 지정된 위치로 이동하기 21

4.2.2 현 위치에서 지정된 값 만큼씩 이동하기 23

4.2.3 지정한 포인트 파일내의 위치로 이동하기 24

4.2.4 지정한 포인트 파일내의 위치로 이동하기(Proface 전용) 24

4.2.5 GPNT 를 이용한 절대 위치 이동(Proface 전용) 26

4.2.6 GPNT 를 이용한 상대 위치 이동(Proface 전용) 27

4.3 속도 쓰기 28

4.4 실행할 작업 관련 29

4.4.1 현재 운전 중인 파일 읽기 29

4.4.2 현재 운전 중인 작업 라인 읽기 30

4.4.3 운전할 프로그램 번호 설정하기(Proface 전용) 31

4.5 파일 관련 32

4.5.1 파일 UpLoad(컨트롤러 -> Host) 32

4.5.2 파일 DownLoad(Host -> 컨트롤러) 34

4.5.3 파일 존재 여부 검사 36

4.5.4 컨트롤러내에 있는 파일 정보 읽기 37

4.5.5 파일 삭제 39

4.5.6 파일 복사 40
4.5.7 파일 이름 변경 41
4.5.8 파일 아이디 변경 42
4.5.9 파일의 이름으로 ID 및 속해 있는 채널 알기 43
4.5.10 포인트 번호를 고정하여 포인트 파일 읽기(Controller→Host)(Proface 전용) 44
4.5.11 축번호를 고정하여 포인트 파일 읽기(Controller→Host)(Proface 전용) 45
4.5.12 포인트 번호를 고정하여 포인트 파일 쓰기(Host→Controller)(Proface 전용) 46
4.5.13 축번호를 고정하여 포인트 파일 쓰기(Host → Controller)(Proface 전용) 47
4.5.14 포인트 파일 번호 설정하기(Proface 전용) 48
4.5.15 설정한 포인트 파일 번호 읽기(Proface 전용) 49
4.5.16 파라미터 정의 파일 읽기 50
4.6 I/O 관련 52
4.6.1 지정한 개수만큼 접점 읽기 52
4.6.2 지정한 개수만큼 접점 쓰기 54
4.6.3 접점 1byte 읽기 56
4.6.4 접점 1byte 쓰기 57
4.6.5 접점 1bit 읽기 58
4.6.6 접점 1bit 쓰기 59
4.6.7 접점 명령 쓰기(Proface 전용) 60
4.6.8 접점 명령 읽기(Proface 전용) 62
4.6.9 지정한 개수만큼 접점 쓰기(Proface 전용) 63
4.6.10 다수의 접점 모니터링(PLC 편집기용) 64
4.6.11 모든 접점 모니터링(PLC 편집기용) 65
4.7 전역 변수 관련 66
4.7.1 지정한 개수 만큼 전역변수(GPNT,GINT,GFLT) 읽기 66
4.7.2 지정한 개수 만큼 전역변수(GPNT,GINT,GFLT) 쓰기 68
4.7.3 포인트 번호를 고정하여 지정한 축개수 만큼 GPNT 읽기(Proface 전용) 70
4.7.4 축 번호를 고정하여 지정한 포인트 개수 만큼 GPNT 읽기(Proface 전용) 71
4.7.5 포인트 번호를 고정하여 지정한 축개수 만큼 GPNT 쓰기(Proface 전용) 72
4.7.6 축 번호를 고정하여 지정한 포인트 개수 만큼 GPNT 쓰기(Proface 전용) 73
4.7.7 GINT/GFLT 쓰기(Proface 전용) 74
4.8 파라미터 관련 75
4.8.1 컨트롤러의 파라미터 버전 읽기 75
4.9 기타 76
4.9.1 기능 실패 원인 읽기 76

4.9.2 설정된 채널의 개수 및 ID 읽기 77
4.9.3 채널의 보유 축 수 읽기 78
개정 사항 79
4.10 3 차 수정본(2004.9) 79
4.10.1 개정사항 항목 추가 79
4.10.2 알파벳순 프로토콜 정리 표 추가 79
4.10.3 Proface 용 Protocol 추가 79
4.11 4 차 수정본(2004.10) 79
4.11.1 'QC'프로토콜 수정 79
4.11.2 포인트 파일 번호 설정 프로토콜 추가 79
4.11.3 Lamp 표시용으로 접점 명령을 읽을 수 있도록 함 79
4.11.4 포인트 파일과 전역워치변수에 대해서도 쓰기와 마찬가지로 포인트번호를 고정 또는 축 고정으로 하는 2 가지 모두 가능하도록 함 80
4.11.5 오타 수정 80
4.12 VER 5 (2005-04-01) 80
4.12.1 ~차 수정본을 Ver~ 표기방식으로 개정 80
4.12.2 'PM'프로토콜 추가 80
4.12.2.1 PLC Ladder 편집기에서 접점 모니터링에 사용 80
4.12.3 'PE'프로토콜 수정(ver 5.2) 80
4.12.4 오타 수정(ver 5.3, 2005-10-18) 80
4.12.5 4.6.2 'PW'프로토콜 오타 수정. (ver 5.3) 80
4.12.6 'FW'프로토콜 추가(ver 5.4) 80

1 프로토콜 일람표(기능별)

구 분	항 목	설명	Page
상태보기	Robot State	Error Exist, In Position, Job Run State, Origin Complete, Servo On/Off	13
	Read Velocity	제어기의 현재 속도 읽기	14
	Read Position	현재 로봇의 위치 읽기(encoder, joint, xyz)	15
	Read Error Text	제어기에 발생된 에러 정보 읽기(텍스트)	16
	Read Error Code	제어기에 발생된 에러 정보 읽기(Code)	17
	Read Operation State	제어기의 운전 상태를 읽음(Jog,Org,Run,Sv On)	18
Motion	Move with Data	사용자가 지정한 데이터를 이용하여 Joint, Linear, Arc, Circle 로 이동	21
	Move with Delta	현재 로봇의 위치에서 증가분 만큼 Joint, Linear 이동	23
	Move with File	지정한 포인트 파일 내의 점을 이용하여 Joint, Linear, Arc, Circle 로 이동	24
Operation	Velocity Write	속도를 Write	28
운전	Read Run Job Name	수행중인 작업이 바뀔 경우 파일 이름을 알수 있다	29
	Read Current Step	현재 수행중인 라인을 읽는다.	30
File	Read File	파일을 PC에 upload한다	32
	Write File	파일을 제어기에 download한다	34
	File 존재 검사	제어기에 파일이 있는지 검사한다.	36
	Read Dir	디렉토리 정보를 읽는다. 확장자로서는 *, PGM, SEQ,PNT	37
	File Delete	파일을 제어기로 부터 지운다	39
	File Copy	파일을 제어기 내에서 복사한다	40
	File Rename	파일을 제어기 내에서 rename한다	41
	Change File ID	파일의 ID를 변경한다.	42
	File ID 읽기	파일이 속해있는 채널의 ID와 파일의 ID를 읽는다.	43
I/O	접점 지정한 개수 읽기	지정한 개수 만큼 접점 읽기	52
	접점 지정한 개수 쓰기	지정한 개수 만큼 접점 쓰기	54
	접점 1byte 읽기	접점 1 byte 읽기	56
	접점 1byte 쓰기	접점 1 byte 쓰기	57
	접점 1bit 읽기	접점 1bit 읽기	58
	접점 1bit 쓰기	접점 1bit 쓰기	59
	다수 접점 읽기	불연속적인 다수의 접점을 읽습니다.	64
	모든 접점 읽기	모든 접점을 읽습니다.	65

전역변수	변수 지정한 개수 읽기	변수(GPNT,GINT,GFLT) 지정한 개수 만큼 읽기	66
	변수 지정한 개수 쓰기	변수(GPNT,GINT,GFLT) 지정한 개수 만큼 쓰기	68
파라미터	파라미터 버전 읽기	현재 제어기의 파라미터 버전을 읽는다.	75
	파라미터 정의파일 읽기	현재 제어기의 파라미터 정의 파일을 읽는다.	50
기 타	Get Execution Error	요구기능 실행 실패에 대한 에러를 받는다.	76
	Read Active Channel	설정된 채널의 개수 읽기	77
	Read Axes No	채널의 보유 축 수 읽기	78

2 프로토콜 분류(이름별, 알파벳 순서)

위치에 내용이 없는 항목은 DaWin_Pro(PC 용 원격제어 프로그램)에서만 사용하는 프로토콜이거나 이전의 PC 용 원격제어 프로그램(Hostpackage) 또는 기타 PLC 인터페이스에서 사용되는 프로토콜로서 본 매뉴얼에서는 설명되지 않습니다.

P에 표시된 항목은 Proface 터치에서 구현되어 있는 것입니다.

글자 1	글자 2	설 명	위 치	P
A	A	Controller Status Report		★
	C	Read Position(0-encoder,1-joint,2-XYZ)		
F	A	File Transfer(Controller → PC)		
	B	File Transfer(PC→ Controller)		
	C	Check If File Exists		
	D	Read All (Motion/Point/Sequence) Files		
	E	Delete File		
	F	Copy File		
	G	Rename		
	H	File Upload(Controller → PC)	4.5.1	
	I	File Download(PC → Controller)	4.5.2	
	J	Check If File Exists	4.5.3	
	K	Read All Files	4.5.4	
	L	Delete File	4.5.5	
	M	Copy File	4.5.6	
	N	Rename File	4.5.7	
	O	Change File ID	4.5.8	
	P	Read Point File with Fixed Point No(Controller→Host) (Proface Only)	4.5.10	★
	Q	Read Point File with Fixed Axis No(Controller→Host) (Proface Only)	4.5.11	★
	R	Read File & Channel ID	4.5.9	★
	S	Write Point File with Fixed Point No(Host→Controller)(Proface Only)	4.5.12	★
	T	Write Point File with Fixed Axis No(Host→Controller)(Proface Only)	4.5.13	★

	U	Set Point File No(Proface Only)	4.5.14	★
	V	Read Point File No(Proface Only)	4.5.15	★
	W	Read Parameter Configuration File		
K	C	Read Controller Type		
	E	Parameter Upload(Controller → PC)		
	F	Parameter Download(PC→ Controller)		
	G	Set Motion Program # To Run(Proface Only)	4.4.3	★
	H	Read Active Channel Number	4.9.2	
	I	Read Axes in One Channel	4.9.3	
	K	Read Current Channel ID		
	O	Read Last Byte Index of H/W Output Port		
	R	Read Running File Name	4.4.1	
	S	Read Current Step # in Running Program	4.4.2	
	V	Read Parameter Version	4.8.1	
M	A	Status Monitoring	4.1.1	★
	B	Speed Monitoring	4.1.2	
	C	Position Monitoring	4.1.3	
	D	Error Monitoring(Text)	4.1.4, 4.9.1	
	E	Error Monitoring(Code)	4.1.5	★
	F	Position Monitoring(Proface Only)	4.1.7	★
	G	Speed Monitoring(Proface Only)	4.1.8	★
	S	Read Controller Status	4.1.6	★
P	A	Write One Output Byte	4.6.4	★
	B	Write One Output Bit	4.6.6	★
	C	Read One Byte	4.6.3	★
	D	Read One Bit	4.6.5	★
	E	Write Command Group(Proface Only)	4.6.7	★
	F	Read Command Group(Proface Only)	4.6.8	★
	G	Write Multiple Byte(Proface Only)	4.6.9	★
	H	Read Whole Bit Information(For DaWin_LD)		
	M	Monitoring Multiple Bits for PLC Editor		
	R	Read Group Input	4.6.1	★
	W	Write Group Output	4.6.2	

Q	A	Read Global Point with Fixed Point No(Proface Only)	4.7.3	★
	B	Read Global Point with Fixed Axis No(Proface Only)	4.7.4	★
	C	Write Global Point Variables with Fixed Point No(Proface Only)	4.7.5	★
	D	Write Global Point Variables with Fixed Axis No(Proface Only)	4.7.6	★
	E	Write GInt/GFlt(Proface Only)	4.7.7	★
	R	Read Global Variables (단, Proface 에서는 GINT/GFLT 만 읽음, GPNT 를 읽을 때는 'QA'나 'QB' 프로토콜 사용)	4.7.1	★
	W	Write Global Variables	4.7.2	
S	B	Speed Override	4.3	★
V	A	Move with Direct Position Data	4.2.1	
	B	Incremental Move	4.2.2	
	C	Move with Point File Data	4.2.3	
	D	Move with Point File Data(Proface Only)	4.2.4	★
	E	Absolute Move with GPNT(Proface Only)	4.2.54.2.5	★
	F	Incremental Move with GPNT(proface Only)	4.2.6	★

3 FLAG 운용 방법

개요

HOST 에서 요구한 기능에 대하여 컨트롤러 응답 Packet 은 다음과 같은 FLAG 값을 갖는다.

FLAG – 0(H30) : 정상적인 통신이 이루어짐

HOST 에서 요구한 프로토콜을 잘 받았다는 의미이며, 프로토콜에 의해 정의된 정상적인 통신이 이루어 지고 있음을 나타냅니다.

FLAG – 1(H31) : 규정 데이터를 벗어난 경우

프로토콜에서 벗어난 기능을 요구 할 때 컨트롤러의 응답입니다.

- 없는 기능을 요구
- 규정 데이터 범위를 벗어난 경우
- 데이터의 Packet 길이가 틀린 경우

H31 을 받은 HOST 는 ACK 신호를 보내고 통신을 종료합니다.

FLAG – 2(H32) : 요구한 기능을 실패 했을 때

HOST 에서 요구한 기능을 실행하다가 실패했을 때 컨트롤러의 응답입니다.

• 선 작업을 설정을 하지 않고 수행을 요구한 경우(원점 수행이 안 되어 있는데, 프로그램을 수행하라고 했을 때) H32 를 받은 HOST 는 ACK 를 보내며 통신을 종료합니다. 이때 기능 실패의 원인을 알고 싶은 경우에는 " 기능실패 원인 알기" 프로토콜를 이용하여 원인을 알면 됩니다.

FLAG – 3(H33) : 현재 구현되어 있지 않은 기능 요구

현재 구현되어 있지 않은 기능을 요구했을 때의 컨트롤러 응답입니다.

H33 을 받은 측은 ACK 를 보내며 통신을 종료합니다.

FLAG – 4(H34) : 연속된 Packet 의 종료를 나타냄

Packet 이 한번으로 끝나지 않고 반복 될 때 Packet 의 종료를 나타냅니다. 제어기로부터 파일을 읽어오는 경우 파일의 길이에 따라 한 Packet 을 넘는 경우가 발생합니다. 만약 제어기가 파일을 보내주다가 파일을 다 보내게 되면 FLAG 에 H34 를 보내게 되므로 받는 측은 ACK 를 보내고 통신을 종료합니다.

4 프로토콜

4.1 상태 보기

4.1.1 현재 상태 읽기

개요

컨트롤러의 상태(동작 중 유무, 인포지션 유무, 알람 유무, 원점 완료 유무, 서보 온 유무)를 알 수 있습니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	종료	체크
프레임(예)	ENQ	**M**	**A**	1~4	ETX	LRC
HEX 값	H05	H4D	H41	H31~H34	H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	DATA1	DATA2	종료	체크
프레임(예)	ENQ	0	3	1	ETX	LRC
HEX 값	H05	H30	H33	H31	H03	

- 완료 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 채널 선택(1Bytes) : 현재 상태를 읽기를 원하는 채널 설정
- DATA1 : 1Bytes, BIT7~4 를 "0011"로 고정하여 ASCII 로 변환. 0x30='0', 0x3F='?'

BIT7 ~ BIT4	BIT3	BIT2	BIT1	BIT0
"0011"로 고정	Servo On	Cmd Fail	-	Any Alarm

- DATA2 : 1Bytes

BIT7 ~ BIT4	BIT3	BIT2	BIT1	BIT0
"0011"로 고정	INPOSITION	Running	Org OK	Alarm

* Alarm : 현재 채널의 Alarm 유무

* Any Alarm : 운전 중인 전채널의 Alarm 유무

4.1.2 현재 속도 읽기

개요

컨트롤러의 각 축의 현재 속도를 알 수 있습니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	종료	체크
프레임(예)	ENQ	**M**	**B**	1~4	ETX	LRC
HEX 값	H05	H4D	H42	H31~H34	H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0	각 축의 속도	ETX	LRC
HEX 값	H05	H30	H33	H03	

- 완료 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 채널 선택(1Bytes) : 현재 상태를 읽기를 원하는 채널 설정
- DATA (축당 10Bytes) : 해당 채널의 보유 축 수가 3 축이면 30Bytes 의 속도값 전송
  소수점을 제외하고 1000 을 곱한 값, RPM 단위

예)1 축의 현재 속도가 1500 RPM 이면

'0' '0' '0' '1' '5' '0' '0' '0' '0' '0' 을 전송한다.

4.1.3 현재 위치 읽기

개요

컨트롤러의 각 축의 현재 위치를 알 수 있습니다.(Encoder,Joint,XY)

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	Type	종료	체크
프레임(예)	ENQ	**M**	**C**	1~4	1	ETX	LRC
HEX 값	H05	H4D	H43	H31~H34	H31	H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0	각 축의 위치	ETX	LRC
HEX 값	H05	H30		H03	

- 완료 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 채널 선택(1Bytes) : 현재 상태를 읽기를 원하는 채널 설정
- Type(1Bytes) : Encoder(0),Joint(1),XY(2)
- DATA (축당 10Bytes) : 해당 채널의 보유 축 수가 3 축이면 30Bytes 의 위치값 전송
  - Encoder 의 경우 : 1 축의 현재 Encoder 값이 12345 이면
    '0' '0' '0' '0' '0' '1' '2' '3' '4' '5' 의 값을 전송한다
  - Joint 의 경우 : 1 축의 현재 위치값이 30.139 이면
    '0' '0' '0' '0' '0' '3' '0' '1' '3' '9' 의 값을 전송한다(소수점 제외, 1000 을 곱한 값)
    XY 의 경우 :

4.1.4 컨트롤러 에러 읽기(텍스트)

개요

상태 읽기에서 ERROR 비트가 설정 되었을 때 응답하며, ERROR 의 내용을 Text 로 전송한다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	종료	체크
프레임(예)	ENQ	**M**	**D**	1~4	ETX	LRC
HEX 값	H05	H4D	H44	H31~H34	H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	DATA		종료	체크
프레임(예)	ENQ	0	**ERROR TEXT**		ETX	LRC
HEX 값	H05	H30			H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

현재 제어기에 발생한 Error 의 내용을 Text 로 전송한다.

4.1.5 컨트롤러 에러 읽기(코드)

개요

상태 읽기에서 ERROR 비트가 설정 되었을 때 응답하며, ERROR 의 Code 를 전송한다

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	종료	체크
프레임(예)	ENQ	**M**	**E**	1~4	ETX	LRC
HEX 값	H05	H4D	H45	H31~H34	H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0	Error Code	ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- DATA(2bytes) : 현재 채널에 발생한 Error Code
  (Error Code 의 내용은 매뉴얼 참조)

4.1.6 컨트롤러 운전 상태 읽기

개요

컨트롤러의 현재 운전 상태를 읽는다.

파라미터 다운로드 전에 Host 에서 먼저 컨트롤러의 상태를 읽을 때 사용한다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		종료	체크
프레임(예)	ENQ	**M**	**S**	ETX	LRC
HEX 값	H05	H4D	H45	H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0	운전 상태	ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- DATA(4bytes) : 각 채널의 운전 상태(순서대로 채널 1 ~ 4)
- 채널 별 DATA 의 내용

BIT7 ~ BIT4	BIT3	BIT2	BIT1	BIT0
"0011"로 고정	Jog 중	Origin 중	Run 중	Servo On

4.1.7 위치 읽기(Proface 전용)

개요

컨트롤러의 현재 위치를 읽는다. 'MC' 프로토콜과는 달리 축시작번호와 축개수에 의해 DATA 의 크기가 결정될 수 있도록 하였다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	Type	축시작	축개수	종료	체크
프레임(예)	ENQ	**M**	**F**	'1'~'4'	'0'~'2'	'0'~'5'	'1'~'6'	ETX	LRC
HEX 값	H05	H4D	H46					H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0	각축의 위치	ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- Type : '0'-Encoder Pulse, '1'-Joint, '2'-XY
- DATA : 10 Bytes × 축의 개수
- 사용되지 않는 축에는 '0'의 값을 채워서 보냅니다. 예를 들어 X 기구부는 한 축을 가지고 있는데 2 개의 축을 요구하였을 경우 2 번째 축에 해당되는 데이터값에는 '0'이 채워집니다.

4.1.8 속도 읽기(Proface 전용)

개요

컨트롤러의 현재 속도를 읽는다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	축시작	축개수	종료	체크
프레임(예)	ENQ	**M**	**G**	'1'~'4'	'0'~'5'	'1'~'6'	ETX	LRC
HEX 값	H05	H4D	H47				H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0	각축의 속도	ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- DATA : 10 Bytes × 축의 개수
- 사용되지 않는 축에는 '0'의 값을 채워서 보냅니다. 예를 들어 X 기구부는 한 축을 가지고 있는데 2 개의 축을 요구하였을 경우 2 번째 축에 해당되는 데이터값에는 '0'이 채워집니다.

4.2 모션 관련

4.2.1 지정된 위치로 이동하기

개요

현위치에서 지정된 절대 위치로 이동합니다.
이동 방법은 4 가지(JOINT,LINEAR,ARC,CIRCLE)가 있고, 이동 형태는 2 가지(JOINT, XYZ)가 있습니다. 이동방법 중 JOINT/LINEAR 는 축별로 이동 위치가 1 개(10Bytes)면 되지만 ARC/CIRCLE 은 축별 이동위치는 2 개가 있어야 합니다.
- 컨트롤러의 현재 상태가 운전 중이 아닐 때 동작 합니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	TYPE1	TYPE2	DATA	종료	체크
프레임(예)	ENQ	**V**	**A**	1~4	0	0		ETX	LRC
HEX 값	H05	H56	H41	H31~H34	H30	H30		H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 채널(1Byte) : 이동을 원하는 축이 속해있는 채널
- TYPE1(1Byte) : JOINT(0), LINEAR(1), ARC(2), CIRCLE(3)
- TYPE2(1Byte) : JOINT(0), XYZ(1) – 단위: Degree(SCARA), mm(직교)
- DATA : JOINT/LINEAR 인 경우

- JOINT(10Bytes/축) : 1 축 DATA 가 –345.48, 2 축 DATA 가 200.45 이면

프레임(예)	-	0	0	0	3	4	5	4	8	0
ASCII 값	H2D	H30	H30	H30	H33	H34	H35	H34	H38	H30

프레임(예)	0	0	0	0	2	0	0	4	5	0
ASCII 값	H30	H30	H30	H30	H32	H30	H30	H34	H35	H30

- DATA 는 소수점을 제외하고 1000 을 곱한 값을 보냄(10 Bytes 중 앞 부분의 빈자리는 0(H30) 으로 채워서 보냄
- 부호가 음수인 DATA 인 경우 '-' 기호는 10Bytes 의 가장 앞에 위치

- DATA : ARC/CIRCLE 인 경우

첫번째 이동위치의 1 축 DATA 가 50.00, 2 축 DATA 가 50.00 이면

프레임(예)	0	0	0	0	0	5	0	0	0	0
ASCII 값	H30	H30	H30	H30	H30	H35	H30	H30	H30	H30

프레임(예)	0	0	0	0	0	5	0	0	0	0
ASCII 값	H30	H30	H30	H30	H30	H35	H30	H30	H30	H30

두번째 이동위치의 1 축 DATA 가 100.00, 2 축 DATA 가 50.00 이면

프레임(예)	0	0	0	0	1	0	0	0	0	0
ASCII 값	H30	H30	H30	H30	H31	H30	H30	H30	H30	H30

프레임(예)	0	0	0	0	0	5	0	0	0	0
ASCII 값	H30	H30	H30	H30	H30	H35	H30	H30	H30	H30

XYZ(10Bytes/축) : X 축 DATA 가 345.48 이면

프레임(예)	0	0	0	0	3	4	5	4	8	0
ASCII 값	H30	H30	H30	H30	H33	H34	H35	H34	H38	H30

4.2.2 현 위치에서 지정된 값 만큼씩 이동하기

개요

현위치에서 지정된 DATA만큼 더해서 지정된 방법으로 이동합니다.
이동 방법은 2가지(JOINT,LINEAR)가 있고, 이동 형태는 2가지(JOINT, XYZ)가 있습니다.
- 컨트롤러의 현재 상태가 운전 중이 아닐 때 동작 합니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	TYPE1	TYPE2	DATA	종료	체크
프레임(예)	ENQ	**V**	**B**	1~4	0	0		ETX	LRC
HEX값	H05	H56	H42	H31~H34	H30	H30		H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX값	H06

설명

- 채널(1Byte) : 이동을 원하는 축이 속해있는 채널
- TYPE1(1Byte) : PTP(0), LINEAR(1)
- TYPE2(1Byte) : JOINT(0), XYZ(1) – 단위: Degree(SCARA), mm(직교)
- DATA : 지정된 위치로 이동하기의 경우와 같음

- JOINT(10Bytes/축) : 1 축 DATA가 –345.48, 2 축 DATA가 200.45 이면

프레임(예)	-	0	0	0	3	4	5	4	8	0
ASCII 값	H2D	H30	H30	H30	H33	H34	H35	H34	H38	H30

프레임(예)	-	0	0	0	2	0	0	4	5	0
ASCII 값	H2D	H30	H30	H30	H32	H30	H30	H34	H35	H30

XYZ(10Bytes/축) : X 축 DATA가 345.48 이면

프레임(예)	0	0	0	0	3	4	5	4	8	0
ASCII 값	H30	H30	H30	H30	H33	H34	H35	H34	H38	H30

4.2.3 지정한 포인트 파일내의 위치로 이동하기

개요

포인트 파일 내에 저장된 좌표로 지정된 방법으로 이동합니다.
이동 방법은 2가지(JOUINT,LINEAR)가 있고, 이동 형태는 2가지(JOINT, XYZ)가 있습니다.
- 컨트롤러의 현재 상태가 운전 중이 아닐 때 동작 합니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	TYPE1	TYPE2	DATA1	DATA2	종료	체크
프레임(예)	ENQ	**V**	**C**	1~4	0	0	파일 ID	포인트	ETX	LRC
HEX 값	H05	H56	H43	H31~H34	H30	H30			H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 채널(1Byte) : 이동을 원하는 축이 속해있는 채널
- TYPE1(1Byte) : JOINT(0), LINEAR(1), ARC(2), CIRCLE(3)
- TYPE2(1Bytes) : JOINT(0), XYZ(1) – 단위: Degree(SCARA), mm(직교)
- DATA1(3Bytes) : 저장된 포이트 파일 ID
- DATA2(3~6Bytes) : 이동할 포인트 번호
  -> 이동 모션이 JOINT,LINEAR 의 경우 1개의 포인트 번호(3Bytes)
  -> 이동 모션이 ARC,CIRCLE 의 경우 경유점 1개, 종료점 1개 모두 2개의 포인트 번호(3Bytes)

4.2.4 지정한 포인트 파일내의 위치로 이동하기(Proface 전용)

개요

포인트 파일 내에 저장된 좌표로 지정된 방법으로 이동합니다.

- 컨트롤러의 현재 상태가 운전 중(원점/조그/모션실행/직접이동)이 아닐 때 동작합니다.
- 원점완료가 되지 않았을 때에도 에러가 발생합니다.(응답 Flag 가 Run Fail 발생)

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	TYPE1	DATA1	DATA2	DATA3	종료	체크
프레임(예)	ENQ	**V**	**D**	1~4	0	파일 ID	포인트번호1	포인트번호2	ETX	LRC
HEX 값	H05	H56	H43	H31~H34	H30				H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 채널(1Byte) : 이동을 원하는 축이 속해있는 채널
- TYPE1(1Byte) : PTP(0), LINEAR(1), ARC(2), CIRCLE(3)
- DATA1(2Bytes) : 저장된 포인트 파일 ID
- DATA2(3Bytes) : 이동할 포인트 번호 1
  -> 이동 모션이 JOINT,LINEAR 의 경우 1 개의 포인트 번호(3Bytes)
- DATA3(3Bytes) : 이동할 포인트 번호 2
  ➔ 이동 모션이 ARC,CIRCLE 의 경우에만 사용됨
  포인트 번호값이 'F''F''F'일때는 포인트 번호 입력 에러(Proface 에서 처리).

4.2.5 GPNT 를 이용한 절대 위치 이동(Proface 전용)

개요

GPNT 값을 읽어서 절대위치로 이동합니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	TYPE1	DATA1	DATA2	종료	체크
프레임(예)	ENQ	**V**	**E**	1~4	0	포인트번호1	포인트번호2	ETX	LRC
HEX 값	H05	H56	H45	H31~H34	H30			H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 채널(1Byte) : 이동을 원하는 축이 속해있는 채널
- TYPE1(1Byte) : PTP(0), LINEAR(1), ARC(2), CIRCLE(3)
- DATA1(3Bytes) : 이동할 GPNT 번호 1
  -> 이동 모션이 JOINT,LINEAR 의 경우 1 개의 포인트 번호(3Bytes)
- DATA2(3Bytes) : 이동할 GPNT 번호 2
  ➔ 이동 모션이 ARC,CIRCLE 의 경우에만 사용됨
  포인트 번호값이 'F''F''F'일때는 포인트 번호 입력 에러(Proface 에서 처리).

4.2.6 GPNT 를 이용한 상대 위치 이동(Proface 전용)

개요

GPNT 값을 읽어서 상대위치로 이동합니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	TYPE1	DATA1	DATA2	종료	체크
프레임(예)	ENQ	**V**	**F**	1~4	0	포인트번호1	포인트번호2	ETX	LRC
HEX 값	H05	H56	H46	H31~H34	H30			H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 채널(1Byte) : 이동을 원하는 축이 속해있는 채널
- TYPE1(1Byte) : PTP(0), LINEAR(1), ARC(2), CIRCLE(3)
- DATA1(3Bytes) : 이동할 GPNT 번호 1
  -> 이동 모션이 JOINT,LINEAR 의 경우 1 개의 포인트 번호(3Bytes)
- DATA2(3Bytes) : 이동할 GPNT 번호 2
  ➔ 이동 모션이 ARC,CIRCLE 의 경우에만 사용됨
  포인트 번호값이 'F''F''F'일때는 포인트 번호 입력 에러(Proface 에서 처리).

4.3 속도 쓰기

개요

로봇 이동(JOG/절대위치/상대위치) 전에 이동 속도를 설정합니다.
로봇 동작(RUN) 중에 속도를 가변 시킵니다.(Override)

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	DATA	종료	체크
프레임(예)	ENQ	**S**	**B**	1~4	0100	ETX	LRC
HEX 값	H05	H53	H42	H31~H34		H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 채널(1Byte) : 속도변경을 원하는 채널
- DATA(4Bytes) : 1 ~ 100 %

  예) 속도를 100% 로 변경하려면 '0' '1' '0' '0' 을 전송합니다.

  -> 최상위 1byte 는 항상 '0' 으로 설정해야 합니다.

4.4 실행할 작업 관련

4.4.1 현재 운전 중인 파일 읽기

개요

현재 설정 및 수행 중인 프로그램이름 및 ID 를 읽어 옵니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	종료	체크
프레임(예)	ENQ	**K**	**R**	1~4	ETX	LRC
HEX 값	H05	H4B	H52	H31~H34	H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	파일 ID	파일 이름	종료	체크
프레임(예)	ENQ	0		TEST.PGM	ETX	LRC
HEX 값	H05	H30			H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 채널(1Byte) : 운전 중인 파일이 속해있는 채널
- 파일 ID(2Bytes) : 파일 ID(00~99)
- 파일이름(12Bytes) : 8 자 의 파일 이름 + '.' + 파일의 종류 3Bytes

  ➔ 파일 이름이 TEST.PGM 이면 8 자의 파일이름에서 나머지 부준은 공백(H20)으로 채움

프레임(예)					T	E	S	T	.	P	G	M
ASCII 값	H20	H20	H20	H20	H54	H45	H53	H54	H2E	H50	H47	H4D

4.4.2 현재 운전 중인 작업 라인 읽기

개요

현재 채널에서 운전 중인 프로그램의 수행 라인을 읽어옵니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	종료	체크
프레임(예)	ENQ	**K**	**S**	1~4	ETX	LRC
HEX 값	H05	H4B	H53	H31~H34	H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	Line#	종료	체크
프레임(예)	ENQ	0		ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 채널(1Byte) : 운전 중인 파일이 속해있는 채널
- Line#(3Bytes) : 현재 수행 중인 라인 번호

  -> 현재 15 라인 수행 중이면 '0' '1' '5' 를 전송

4.4.3 운전할 프로그램 번호 설정하기(Proface 전용)

개요

실행하기를 원하는 모션 프로그램의 번호를 설정합니다. 설정한 후에는 RunCmd 접점을 이용하여 실행합니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	ID	종료	체크
프레임(예)	ENQ	**K**	**G**	'1'~'4'	'00'~'99'	ETX	LRC
HEX 값	H05	H4B	H47		H31~H34	H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

4.5 파일 관련

4.5.1 파일 UpLoad(컨트롤러 -> Host)

개요

컨트롤러에 있는 프로그램을 HOST 로 업 로드 합니다. 파일 내용이 250Bytes 를 넘는 경우 연속된 Package 방식이 적용됩니다. 파일 로드가 완료되면 컨트롤러는 완료 신호인 FLAG 를 4(H34)보냅니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		NAME	TYPE	종료	체크
프레임(예)	ENQ	**F**	**H**	A.PGM	0	ETX	LRC
HEX 값	H05	H46	H48		H30	H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	FILE CONTENT	종료	체크
프레임(예)	ENQ	0	SPD 10000	ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	연속 FILE CONTENT	종료	체크
프레임(예)	ENQ	0	LOOP 1	ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ACK	4	ETX	LRC
HEX 값	H06	H34	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- NAME(12Bytes) :
  - 프로그램 이름이 A.PGM 이면

프레임(예)	-	-	-	-	-	-	-	A	.	P	G	M
ASCII 값	H20	H20	H20	H20	H20	H20	H20	H41	H2E	H50	H47	H4D

- TYPE(1Byte) : 포인트 파일일 경우 적용. 프로그램 파일은 0(H30)으로 설정 합니다.
  포인트 파일일 경우 : JOINT(0), XYZ(1)
- FILE CONTENT : 파일의 내용

4.5.2 파일 DownLoad(Host -> 컨트롤러)

개요

HOST 에 있는 프로그램을 컨트롤러로 다운 로드 합니다. 파일 내용이 250Bytes 를 넘는 경우 연속된 Package 방식이 적용됩니다. 파일 다운 로드가 완료되면 HOST 는 완료 신호인 FLAG 를 4(H34)로 보냅니다.(한 라인의 끝은 라인 피드(H0A)을 사용합니다)

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	NAME	종료	체크
프레임(예)	ENQ	**F**	**I**	1~4	A.PGM	ETX	LRC
HEX 값	H05	H46	H49	H31~H34		H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작	FLAG	FILE CONTENT		종료	체크
프레임(예)	ENQ	0	SPD 10000	라인 피드	ETX	LRC
HEX 값	H05	H30		H0A	H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작	FLAG	연속 FILE CONTENT		종료	체크
프레임(예)	ENQ	0	LOOP 1	라인 피드	ETX	LRC
HEX 값	H05	H30		H0A	H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

• 응답 포맷 (**HOST** →컨트롤러)

	시작	FLAG	종료	체크
프레임(예)	ENQ	4	ETX	LRC
HEX 값	H05	H34	H03	

• 응답 포맷(컨트롤러 → **HOST**)

	시작	ACK	종료	체크
프레임(예)	ENQ	ACK	ETX	LRC
HEX 값	H05	H05	H03	

설명

• 채널(1Byte) : 파일이 저장될 채널 설정

• NAME(12Bytes) :

- 프로그램 이름이 A.PGM 이면

프레임(예)	-	-	-	-	-	-	-	A	.	P	G	M
ASCII 값	H20	H20	H20	H20	H20	H20	H20	H41	H2E	H50	H47	H4D

- 프로그램 이름이 A.PNT 이면

프레임(예)	-	-	-	-	-	-	-	A	.	P	N	T
ASCII 값	H20	H20	H20	H20	H20	H20	H20	H41	H2E	H50	H4E	H54

• FILE CONTENT : 파일의 내용

4.5.3 파일 존재 여부 검사

개요

컨트롤러에 해당 파일이 있는지 검사한다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		NAME	종료	체크
프레임(예)	ENQ	**F**	**J**	A.PGM	ETX	LRC
HEX 값	H05	H46	H4A		H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	DATA	DATA1	DATA2	종료	체크
프레임(예)	ENQ	0	0	채널 ID	파일 ID	ETX	LRC
HEX 값	H05	H30	H30			H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- NAME(12Bytes) :
  - 프로그램 이름이 A.PGM 이면

프레임(예)	-	-	-	-	-	-	-	A	.	P	G	M
ASCII 값	H20	H20	H20	H20	H20	H20	H20	H41	H2E	H50	H47	H4D

- DATA(1Byte) : 존재(1), 없음(0)
- DATA1(1Byte) : 해당 파일이 속해 있는 채널 ID(1 ~ 4)
- DATA2(2ByteS) : 해당 파일의 ID(00 ~ 99)

4.5.4 컨트롤러내에 있는 파일 정보 읽기

개요

컨트롤러에 있는 모든 프로그램을 보여 줍니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		EXTENSION	종료	체크
프레임(예)	ENQ	**F**	**K**	*.PGM	ETX	LRC
HEX 값	H05	H46	H4B		H03	

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0		ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H05

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0		ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H05

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	4	ETX	LRC
HEX 값	H05	H34	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- EXTENSION(12Bytes) : 프로그램(*.PGM), 포인트(*.PNT), 시퀀스(*.SEQ), 모든파일(*.*)

  - *.PGM 이면

프레임(예)	-	-	-	-	-	-	-	*	.	P	G	M
ASCII 값	H20	H20	H20	H20	H20	H20	H20	H2A	H2E	H50	H47	H4D

  - *.*이면

프레임(예)	-	-	-	-	-	-	-	*	.	*	-	-
ASCII 값	H20	H20	H20	H20	H20	H20	H20	H2A	H2E	H2A	H20	H20

- DATA(15Bytes) :

  NAME(12) + 채널 ID(1) + 파일 ID(2)

4.5.5 파일 삭제

개요
컨트롤러에 있는 파일을 삭제한다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		NAME	종료	체크
프레임(예)	ENQ	**F**	**L**	A.PGM	ETX	LRC
HEX 값	H05	H46	H4C		H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

4.5.6 파일 복사

개요

컨트롤러에 해당 파일을 복사한다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		NAME0	채널	NAME1	종료	체크
프레임(예)	ENQ	**F**	**M**	A.PGM	1~4	B.PGM	ETX	LRC
HEX 값	H05	H46	H4D		H31~H34		H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- NAME(12Bytes) :
  - 프로그램 이름이 A.PGM 이면

프레임(예)	-	-	-	-	-	-	-	A	.	P	G	M
ASCII 값	H20	H20	H20	H20	H20	H20	H20	H41	H2E	H50	H47	H4D

- 채널(1Byte) : 복사하려는 대상 채널 설정
  예) 파일 A.PGM 이 채널 1 에 속해있고 채널 2 에 복사하려면 채널 2 에 B.PGM 이 복사된다.

4.5.7 파일 이름 변경

개요

컨트롤러에 해당 파일이름을 변경한다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		NAME0	NAME1	종료	체크
프레임(예)	ENQ	**F**	**N**	A.PGM	B.PGM	ETX	LRC
HEX 값	H05	H46	H4E			H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

NAME0 파일이름을 NAME1 로 변경합니다.
NAME0 의 파일은 삭제 됩니다.

4.5.8 파일 아이디 변경

개요

컨트롤러에 해당 파일아이디를 변경한다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		index	ID 1	ID 2	종료	체크
프레임(예)	ENQ	**F**	**O**		1	2	ETX	LRC
HEX 값	H05	H46	H4E				H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 1 번 파일아이디를 2 번 으로 변경합니다.
- Index = 0 ; 모션 프로그램을 말합니다.
  Index = 1 ; 시컨스 프로그램을 말합니다.
  Index = 2 ; 포인트 파일을 말합니다.
- ID(2Bytes) :
  - 파일 ID 가 1 0 이면

프레임(예)	0	1
ASCII 값	H30	H31

4.5.9 파일의 이름으로 ID 및 속해 있는 채널 알기

개요

파일의 이름으로 ID 및 파일이 속해있는 채널을 알아 냅니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		파일 이름	종료	체크
프레임(예)	ENQ	**F**	**R**		ETX	LRC
HEX 값	H05	H46	H52		H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	DATA1	DATA2	종료	체크
프레임(예)	ENQ	0	채널 ID	파일 ID	ETX	LRC
HEX 값	H05	H30			H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 파일이름(12Bytes) : 8 자 의 파일 이름 + '.' + 파일의 종류 3Bytes
- 파일의 종류 : 모션 파일(PGM), 시퀀스 파일(SEQ), 포인트 파일(PNT)
- DATA1(1Byte) : 채널 ID(1~4)
- DATA2(2Byte) : 파일 ID(00~99)

➔ 프로그램 이름이 TEST.PGM 이면 8 자의 파일이름에서 나머지 부분은 공백(H20)으로 채움

프레임(예)					T	E	S	T	.	P	G	M
ASCII 값	H20	H20	H20	H20	H54	H45	H53	H54	H2E	H50	H47	H4D

4.5.10 포인트 번호를 고정하여 포인트 파일 읽기(Controller→Host)(Proface 전용)

개요

포인트 파일의 내용을 읽습니다.(포인트 시작번호 우선)

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		포인트번호	축시작	축개수	종료	체크
프레임(예)	ENQ	**F**	**P**	'000'~'999'	'0'~'6'	'1'~'7'	ETX	LRC
HEX 값	H05	H46	H50				H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0		ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 포인트 파일의 번호는 포인트 파일 설정 프로토콜을 사용합니다.
- DATA : 포인트 데이터 × 축의 개수
  - 123.456 의 데이터는 '0000123456'으로 전송한다.
- 7 번째 DATA 는 위치데이터가 아니라 정수데이터값이고 ASCII 숫자로 변환하여 전송한다.
  - 0x11(16 진수)의 데이터는 '0000000017'로 전송한다.
  - 최대 0xFFFFFFFF = '4294967295'까지의 범위를 갖는다.(unsigned 정수로 취급)
- 포인트 파일이 없거나 해당포인트가 없을 때에는 에러 Flag 를 보내지 않고 DATA 영역에 '0'을 채워 보냅니다.
- 축시작+축개수를 계산한 값이 7 보다 크면 프로토콜 에러가 발생합니다.
- 포인트번호가 0 에서 999 의 범위를 벗어나면 프로토콜 에러가 발생합니다.
- DATA 가 제어기 내부의 수신버퍼의 크기(290 여 바이트)를 초과하면 프로토콜에러가 발생합니다.

4.5.11 축번호를 고정하여 포인트 파일 읽기(Controller→Host)(Proface 전용)

개요

포인트 파일의 내용을 읽습니다. 축을 고정한 채로 여러 개의 포인트 번호에 대해서 읽습니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		축번호	포인트시작	포인트개수	종료	체크
프레임(예)	ENQ	**F**	**Q**	'0'~'6'	'000'~'999'	'1'~'256'	ETX	LRC
HEX 값	H05	H46	H51				H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0		ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 포인트 파일의 번호는 포인트 파일 설정 프로토콜을 사용합니다.
- DATA : 포인트 데이터 × 포인트의 개수
- 포인트 파일이 없거나 해당포인트가 없을 때에는 에러 Flag 를 보내지 않고 DATA 영역에 '0'을 채워 보냅니다.
- 포인트시작+포인트개수를 계산한 값이 256 보다 크면 프로토콜 에러가 발생합니다.
- 축번호가 0 에서 6 의 범위를 벗어나면 프로토콜 에러가 발생합니다.

4.5.12 포인트 번호를 고정하여 포인트 파일 쓰기(Host→Controller)(Proface 전용)

개요

포인트 파일의 내용을 변경합니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		포인트번호	축시작	개수	DATA	종료	체크
프레임(예)	ENQ	**F**	**S**	'000'~'999'	'0'~'6'	'1'~'7'		ETX	LRC
HEX 값	H05	H46	H53					H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 포인트 파일의 번호는 포인트 파일 설정 프로토콜을 사용합니다.
- DATA : 포인트 데이터 × 포인트의 개수
- 포인트 파일이 없거나 해당포인트가 없을 때에는 에러 Flag 를 보내지 않고 DATA 영역에 '0'을 채워 보냅니다.
- 축시작+축개수를 계산한 값이 7 보다 크면 프로토콜 에러가 발생합니다.
- 포인트번호가 0 에서 999 의 범위를 벗어나면 프로토콜 에러가 발생합니다.
- 제어기 내부의 메모리 용량 제한 등으로 파일을 만들 수 없는 경우에는 RunFail 에러 Flag 를 전송합니다.

4.5.13 축번호를 고정하여 포인트 파일 쓰기(Host → Controller)(Proface 전용)

개요

포인트 파일의 내용을 변경합니다. 축번호를 고정한 채로 여러 개의 포인트 번호에 대해서 변경합니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		축번호	포인트시작	포인트개수	DATA	종료	체크
프레임(예)	ENQ	F	T	'0'~'6'	'000'~'999'	'1'~'999'		ETX	LRC
HEX 값	H05	H46	H54					H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 포인트 파일의 번호는 포인트 파일 설정 프로토콜을 사용합니다.
- DATA : 포인트 데이터 × 포인트의 개수
- 포인트 파일이 없거나 해당포인트가 없을 때에는 에러 Flag 를 보내지 않고 DATA 영역에 '0'을 채워 보냅니다.
- 포인트시작+포인트개수를 계산한 값이 256 보다 크면 프로토콜 에러가 발생합니다.
- 축번호가 0 에서 6 의 범위를 벗어나면 프로토콜 에러가 발생합니다.

4.5.14 포인트 파일 번호 설정하기(Proface 전용)

개요

포인트 파일을 읽고 쓸 때 사용될 포인트 파일의 번호를 설정합니다

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		파일ID	종료	체크
프레임(예)	ENQ	**F**	**U**	'00'~'99'	ETX	LRC
HEX 값	H05	H46	H54		H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 포인트 파일을 읽고/쓸 때 사용되는 포인트 파일의 번호를 설정한다.

4.5.15 설정한 포인트 파일 번호 읽기(Proface 전용)

개요

포인트 파일의 내용을 변경할 때 사용되는 현재 설정되어 있는 포인트 파일의 번호를 제어기로부터 읽습니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		종료	체크
프레임(예)	ENQ	**F**	**V**	ETX	LRC
HEX 값	H05	H46	H55	H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0	'00'~'99'	ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 포인트 파일을 읽고/쓸 때 사용되는 포인트 파일의 번호를 설정한다.

4.5.16 파라미터 정의 파일 읽기

개요

파라미터의 종류와 설정값을 정의한 파라미터 파일을 제어기로부터 읽습니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		종류	종료	체크
프레임(예)	ENQ	**F**	**W**	'0'~'1'	ETX	LRC
HEX 값	H05	H46	H55	H31~H32	H03	

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0	Text 데이터(최대 250 바이트)	ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H05

데이터가 많을 경우 반복

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	4	ETX	LRC
HEX 값	H05	H34	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

더이상 데이터가 없을 경우

설명

- 시그마 제어기 버전 1.9.12 부터 사용가능합니다.
- 요구포맷에서 종류가 '0'일 경우에 Parameter.txt 파일을 '1'일 경우에 Option.txt 파일을 전송합니다.

4.6 I/O 관련

4.6.1 지정한 개수만큼 접점 읽기

개요

점점을 지정한 개수만큼 읽습니다. 읽어올 내용이 250Bytes 를 넘는 경우 연속된 Package 방식이 적용됩니다. 접점읽기가 완료되면 HOST 는 완료 신호인 FLAG 를 4(H34)로 보냅니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		시작위치	개수	종료	체크
프레임(예)	ENQ	**P**	**R**	'000'~'399'	'001'~'400'	ETX	LRC
HEX 값	H05	H50	H52			H03	

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0		ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H05

- 응답 포맷(컨트롤러 → **HOST**) : 연속되는 DATA 가 있을 때

	시작	FLAG	DATA(연속)	종료	체크
프레임(예)	ENQ	0		ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H05

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	4	ETX	LRC
HEX 값	H05	H34	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 시작위치(3Bytes) : 읽어올 접점의 시작 번지("000" ~ "339")
- 개수(3Bytes) : 읽어올 접점의 개수(Byte 단위)

  예) 100 번지부터 3Bytes 를 읽으려면

  시작위치 : '1' '0' '0'

  개수 : '0' '0' '3'
- DATA : 접점의 내용

  접점 1 byte 형식(4 접점 데이타/바이트) : 접점 데이타는 High 시에 1, Low 시에 0

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	0	1	1	접점 1	접점 2	접점 3	접점 4

  -> 전체 통신이 1 개의 패킷내에서 종료되면 연속되는 Data 는 송신할 필요 없음

  -> bit7 ~ bit4 는 0011 data 를 ASCII 로 변환하기 위해 고정

  예) B100 = 01011010(B100.0=0, .1=1, .2=0, .3=1, .4=1, .5=0, .6=1, .7=0)이면

  byte1(B100.0 ~ B100.3)

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	0	1	1	1	0	1	0

  byte2(B100.4 ~ B100.7)

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	0	1	1	0	1	0	1

4.6.2 지정한 개수만큼 접점 쓰기

개요

점점을 지정한 개수만큼 컨트롤러에 씁니다. 쓸 접점의 내용이 250Bytes 를 넘는 경우 연속된 Package 방식이 적용됩니다. 접점 쓰기가 완료되면 HOST 는 완료 신호인 FLAG 를 4(H34)로 보냅니다

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		시작위치	개수	종료	체크
프레임(예)	ENQ	**P**	**W**			ETX	LRC
HEX 값	H05	H50	H57			H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0		ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작	FLAG	DATA(연속)	종료	체크
프레임(예)	ENQ	0		ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** →컨트롤러)

	시작	FLAG	종료	체크
프레임(예)	ENQ	4	ETX	LRC

HEX 값	H05	H34	H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 시작위치(3Bytes) : 쓰기를 원하는 접점의 시작 번지("000" ~ "339")
- 개수(3Bytes) : 쓸 접점의 개수(Byte 단위)
- DATA : 접점의 내용(읽기의 경우와 동일 함)

4.6.3 접점 1byte 읽기

개요

점점의 내용을 1Byte 읽습니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		접점위치	종료	체크
프레임(예)	ENQ	**P**	**C**		ETX	LRC
HEX 값	H05	H50	H43		H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG		종료	체크
프레임(예)	ENQ	0		ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 시작위치(3Bytes) : 읽기를 원하는 접점 번지("000" ~ "339")
- DATA : 접점의 내용(읽기의 경우와 동일 함)

4.6.4 접점 1byte 쓰기

개요

점점의 내용을 1Byte 씁니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		접점위치	DATA	종료	체크
프레임(예)	ENQ	**P**	**A**			ETX	LRC
HEX 값	H05	H50	H41			H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 접점위치(3Bytes) : 쓰기를 원하는 접점 번지("000" ~ "339")
  ➔ H/W Input 접점은 쓸 수 없습니다.
- DATA : 접점의 내용(읽기의 경우와 동일 함)

4.6.5 접점 1bit 읽기

개요

점점의 내용을 1Bit 읽습니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		접점위치	Index	종료	체크
프레임(예)	ENQ	**P**	**D**			ETX	LRC
HEX 값	H05	H50	H44			H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0		ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 접점위치(3Bytes) : 읽기를 원하는 접점의 번지("000" ~ "339") – Byte 단위
- Index(1Byte) : 읽기를 원하는 접점의 Index ("0" ~ "7") – Bit 단위
- DATA(1Byte) : 읽어온 Bit 접점의 내용('0' ~ '1')

4.6.6 접점 1bit 쓰기

개요

점점의 내용을 1Bit 씁니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		접점위치	Index	DATA	종료	체크
프레임(예)	ENQ	**P**	**B**				ETX	LRC
HEX 값	H05	H50	H42				H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 접점위치(3Bytes) : 쓰기를 원하는 접점의 번지("000" ~ "339") – Byte 단위
- Index(1Byte) : 쓰기를 원하는 접점의 Index ("0" ~ "7") – Bit 단위
- DATA (1Byte): 쓸 Bit 접점의 내용('0' ~ '1')

4.6.7 접점 명령 쓰기(Proface 전용)

개요

점점을 사용한 각종 동작을 위한 명령 접점을 조작합니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		접점위치	Index	DATA	종료	체크
프레임(예)	ENQ	**P**	**E**	'00'~'15'	'0'~'7'	'0' or '1'	ETX	LRC
HEX 값	H05	H50	H45				H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 접점위치(2Bytes) : 쓰기를 원하는 접점의 번지("00" ~ "15") – Byte 단위
- Index(1Byte) : 쓰기를 원하는 접점의 Index ("0" ~ "7") – Bit 단위
- DATA (1Byte): 쓸 Bit 접점의 내용('0' ~ '1')

구분	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
B0	전 Ch 선택	Ch1 선택	Ch2 선택	Ch3 선택	Ch4 선택	모션실행	모션리셋	원점실행
B1	비상정지	정지	JogInch	JogXY	JogSpd4	JogSpd3	JogSpd2	JogSpd1
B2	사용안함	에러리셋	JogAxis6+	JogAxis5+	JogAxis4+	JogAxis3+	JogAxis2+	JogAxis1+
B3	포인트삭제	포인트복사	JogAxis6-	JogAxis5-	JogAxis4-	JogAxis3-	JogAxis2-	JogAxis1-
B4	사용안함	AllSvOn	Sv6On	Sv5On	Sv4On	Sv3On	Sv2On	Sv1On
B5	사용안함	AllSvOff	Sv6Off	Sv5Off	Sv4Off	Sv3Off	Sv2Off	Sv1Off
B6	사용안함	Mpg On	MpgAxis6	MpgAxis5	MpgAxis4	MpgAxis3	MpgAxis2	MpgAxis1

B7 ~ B15	다른 기능 추가시 사용예정

- 모션리셋은 프로그램을 처음부터 실행하고 싶을 때 사용합니다.(Rising Edge 에서 동작합니다.)
- MPG 기능 사용시, MPG 의 이동배율 및 XY/Joint 이동모드는 JogSpd1~4, JogMode 비트를 조그 수행할때와 같이 사용합니다.
- JogInch : 0 = Continuous 조그, 1 = Inching 조그
- JogXY : 0 = Joint 조그, 1 = XY 조그
- 여러명령접점이 동시에 들어오면 내부의 우선순위에 따라 처리되오니 한번에 한 개의 수행명령만 입력하십시오(예, 원점과 조그는 동시에 수행될 수 없습니다).
- 포인트복사, 포인트삭제는 1.6.15 에서 추가됨.(2005-05-10) 이 접점이 1 이 되면, GINT(233)에 저장되어 있는 값은 해당 작업이 적용되는 포인트 파일 번호를 나타냄. 복사시에는 원본은 포인트 파일 번호 설정 프로토콜에 의해서 설정된 포인트 파일 번호이고, 대상은 GINT(233)값을 번호로 가진 포인트 파일임. 삭제시에는 GINT(233)값을 번호로 가진 포인트 파일이 삭제됨. Rising Edge 일 때에 동작됨.

4.6.8 접점 명령 읽기(Proface 전용)

개요

점점을 사용한 각종 동작을 위한 명령 접점의 현재상태를 바이트 단위로 읽습니다(터치의 Lamp 용으로 사용).

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		접점시작위치	개수	종료	체크
프레임(예)	ENQ	**P**	**F**	'00'~'15'	'01'~'16'	ETX	LRC
HEX 값	H05	H50	H46			H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0		ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 접점시작위치(2Bytes) : 읽기를 원하는 접점(바이트)의 번지("00" ~ "15") – Byte 단위
- 개수(2Byte) : 읽기를 원하는 접점(바이트)의 시작번지로부터의 개수("01" ~ "16") – Byte 단위

4.6.9 지정한 개수만큼 접점 쓰기(Proface 전용)

개요

점점의 내용을 지정한 개수만큼 씁니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		시작위치	개수	DATA	종료	체크
프레임(예)	ENQ	**P**	**G**	'000'~'399'	'1'~'400'		ETX	LRC
HEX 값	H05	H50	H47				H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 시작위치(3Bytes) : 쓰기를 원하는 접점의 시작 번지("000" ~ "399")
- 개수(3Bytes) : 쓸 접점의 개수(Byte 단위)
- DATA : 접점의 내용(읽기의 경우와 동일 함)
- 바이트의 내용을 전송시에는 하위 nibble 부터 먼저 전송합니다. 그리고 DATA 값을 ASCII 문자로 변환하기 위해서 0x30 을 더하여 전송합니다.
  - B = 0x87 일 경우, '78'로 전송합니다.

4.6.10 다수의 접점 모니터링(PLC편집기용)

개요

접점의 내용을 지정한 개수만큼 읽습니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		비트개수	비트 정의 데이터 (비트개수만큼)	종료	체크
프레임(예)	ENQ	**P**	**M**	'00'~'99'		ETX	LRC
HEX 값	H05	H50	H4D			H03	

*비트 정의 데이터

바이트번호	비트번호
'0'~'399'	'0'~'7'

- 응답 포맷(컨트롤러 → **HOST**)

	시작	Data	종료	체크
프레임(예)	ENQ	설명 참조	ETX	LRC
HEX 값	H05		H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 응답 데이터의 내용

  접점의 내용이 0 일 경우, ASCII '0'(0x30)을, 1 일 경우, ASCII '1'(0x31)을 전송한다.

  요구포맷에서 요구한 개수만큼 데이터를 전송한다. 10 개의 접점 데이터를 요구하였을 경우 Data 의 내용은 10 바이트가 된다.

4.6.11 모든 접점 모니터링(PLC편집기용)

개요

모든 접점의 내용을 읽습니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		종료	체크
프레임(예)	ENQ	**P**	**H**	ETX	LRC
HEX 값	H05	H50	H4D	H03	

나머지 포맷은 연속 데이터 포맷을 따른다.

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0	접점 데이터(200 바이트씩)	ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H05

4 번 반복

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	4	ETX	LRC
HEX 값	H05	H34	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

더이상 데이터 가 없을 경우

설명

- DaWin_LD 에서 모든 접점의 내용을 읽을 때 사용된다.
- DATA 의 내용은 4.6.1 을 참조

4.7 전역 변수 관련

4.7.1 지정한 개수 만큼 전역변수(GPNT,GINT,GFLT) 읽기

개요

지정한 전역변수를 지정한 개수만큼 읽습니다. 읽어올 내용이 250Bytes 를 넘는 경우 연속된 Package 방식이 적용됩니다. 접점일기가 완료되면 HOST 는 완료 신호인 FLAG 를 4(H34)로 보냅니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		전역변수 종류	시작위치	개수	종료	체크
프레임(예)	ENQ	**Q**	**R**				ETX	LRC
HEX 값	H05	H51	H52				H03	

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0		ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H05

- 응답 포맷(컨트롤러 → **HOST**) : 연속되는 DATA 가 있을 때

	시작	FLAG	DATA(연속)	종료	체크
프레임(예)	ENQ	0		ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H05

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	4	ETX	LRC
HEX 값	H05	H34	H03	

• 응답 포맷 (**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H06

설명

- 전역변수 종류(1Bytes) : 읽어올 전역 변수의 종류
  GINT 의 경우 : 'I'
  GFLT 의 경우 : 'F'
  GPNT 의 경우 : 'P'
- 시작위치(3Bytes) : 읽어올 전역 변수의 시작 번지("000" ~ "339")
- 개수(3Bytes) : 읽어올 전역 변수의 개수(Byte 단위)
- DATA : 전역 변수의 내용
  GINT 변수(10bytes) : 앞은 '0'으로 채워서 보냄
  예) GINT[100]=12345 인 경우 '0''0''0''0''0''1''2''3''4''5' 를 송신 함
  GFLT 변수(10bytes) : 소수점 제외, 부호 포함, 앞은 '0'으로 채워서 보냄
  예) GFLT[100]=12.345 인 경우 '0''0''0''0''0''1''2''3''4''5' 를 송신 함
  예) GFLT[100]=-12.345 인 경우 '-''0''0''0''0''1''2''3''4''5' 를 송신 함
  GPNT 변수(10bytes * 6) : 소수점 제외, 부호 포함, 앞은 '0'으로 채워서 보냄
  ➔ GPNT 의 경우는 GFLT 와 동일하고 한 개의 GPNT 에 6 개의 Data 를 송신 함

4.7.2 지정한 개수 만큼 전역변수(GPNT,GINT,GFLT) 쓰기

개요

점점을 지정한 개수만큼 컨트롤러에 씁니다. 쓸 접점의 내용이 250Bytes 를 넘는 경우 연속된 Package 방식이 적용됩니다. 접점 쓰기가 완료되면 HOST 는 완료 신호인 FLAG 를 4(H34)로 보냅니다

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		전역변수 종류	시작위치	개수	종료	체크
프레임(예)	ENQ	**Q**	**W**				ETX	LRC
HEX 값	H05	H51	H57				H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0		ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** → 컨트롤러)

	시작	FLAG	DATA(연속)	종료	체크
프레임(예)	ENQ	0		ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷 (**HOST** →컨트롤러)

	시작	FLAG	종료	체크
프레임(예)	ENQ	4	ETX	LRC

HEX 값	H05	H34	H03	

- 응답 포맷(컨트롤러 → **HOST**)

	ACK
프레임(예)	ACK
HEX 값	H05

설명

- 전역변수 종류(1Bytes) : 쓸 전역 변수의 종류
  GINT 의 경우 : 'I'
  GFLT 의 경우 : 'F'
  GPNT 의 경우 : 'P'
- 시작위치(3Bytes) : 쓸 전역 변수의 시작 번지("000" ~ "339")
- 개수(3Bytes) : 쓸 전역 변수의 개수(Byte 단위)
- DATA : 전역 변수의 내용
  GINT 변수(10bytes) : 앞은 '0'으로 채워서 보냄
  예) GINT[100]=12345 로 변경을 원할 경우 '0''0''0''0''0''1''2''3''4''5' 를 송신 함
  예) GFLT[100]=12.345 인 경우 '0''0''0''0''0''1''2''3''4''5' 를 송신 함
  예) GFLT[100]=-12.345 인 경우 '-''0''0''0''0''1''2''3''4''5' 를 송신 함
  GPNT 변수(10bytes * 6) : 소수점 제외, 부호 포함, 앞은 '0'으로 채워서 보냄
  ➔ GPNT 의 경우는 GFLT 와 동일하고 한 개의 GPNT 에 6 개의 Data 를 송신 함

4.7.3 포인트 번호를 고정하여 지정한 축개수 만큼 GPNT 읽기(Proface 전용)

개요

지정한 포인트형 전역변수를 지정한 개수만큼 읽습니다. 한 개의 포인트에 대해서 지정한 축 개수만큼의 데이터를 읽습니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		포인트번호	축시작번호	개수	종료	체크
프레임(예)	ENQ	**Q**	**A**	'000'~'255'	'0'~'6'	'1'~'7'	ETX	LRC
HEX 값	H05	H51	H41				H03	

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0		ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H05

설명

- DATA : 포인트 데이터(10Bytes) × 축의 개수
- 축시작+축개수를 계산한 값이 7 보다 크면 프로토콜 에러가 발생합니다.
- 포인트번호가 0 에서 255 의 범위를 벗어나면 프로토콜 에러가 발생합니다.

4.7.4 축 번호를 고정하여 지정한 포인트 개수 만큼 GPNT 읽기(Proface 전용)

개요

지정한 포인트형 전역변수를 지정한 개수만큼 읽습니다. 한 개의 축번호에 대해서 지정된 수만큼의 여러 개의 포인트 번호값을 읽습니다..

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		축번호	포인트시작번호	개수	종료	체크
프레임(예)	ENQ	**Q**	**B**	'0'~'6'	'000'~'255'	'001'~'256'	ETX	LRC
HEX 값	H05	H51	H42				H03	

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0		ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H05

설명

- DATA : 포인트 데이터(10Bytes) × 축의 개수
- 포인트시작+개수를 계산한 값이 256 보다 크면 프로토콜 에러가 발생합니다.
- 축번호가 0 에서 6 의 범위를 벗어나면 프로토콜 에러가 발생합니다.

4.7.5 포인트 번호를 고정하여 지정한 축개수 만큼 GPNT 쓰기(Proface 전용)

개요

지정한 포인트형 전역변수를 지정한 개수만큼 변경합니다. 한 개의 포인트에 대해서 여러축의 데이터를 변경합니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		포인트번호	축시작번호	개수	DATA	종료	체크
프레임(예)	ENQ	**Q**	**C**	'000'~'255'	'0'~'6'	'1'~'7'		ETX	LRC
HEX 값	H05	H51	H43					H03	

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H05

설명

- DATA : 포인트 데이터(10Bytes) × 축의 개수
- 축시작+축개수를 계산한 값이 7 보다 크면 프로토콜 에러가 발생합니다.
- 포인트번호가 0 에서 255 의 범위를 벗어나면 프로토콜 에러가 발생합니다.

4.7.6 축 번호를 고정하여 지정한 포인트 개수 만큼 GPNT 쓰기(Proface 전용)

개요

지정한 포인트형 전역변수를 지정한 개수만큼 변경합니다. 여러 개의 포인트에 대해서 같은 축번호를 갖는 여러 개의 데이터를 변경합니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		축번호	포인트시작번호	개수	DATA	종료	체크
프레임(예)	ENQ	**Q**	**D**	'0'~'6'	'000'~'255'	'001'~'256'		ETX	LRC
HEX 값	H05	H51	H44					H03	

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H05

설명

- DATA : 포인트 데이터(10Bytes) × 축의 개수
- 포인트시작+개수를 계산한 값이 256 보다 크면 프로토콜 에러가 발생합니다.
- 축번호가 0 에서 6 의 범위를 벗어나면 프로토콜 에러가 발생합니다.

4.7.7 GINT/GFLT 쓰기(Proface 전용)

개요

요구포맷에 데이터를 포함하기 위하여 Proface 를 위해 새로 만든 프로토콜 입니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		종류	시작번호	개수	DATA	종료	체크
프레임(예)	ENQ	**Q**	**E**	'I' or 'F'	'000'~255	'001'~'256'		ETX	LRC
HEX 값	H05	H51	H45					H03	

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	종료	체크
프레임(예)	ENQ	0	ETX	LRC
HEX 값	H05	H30	H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작
프레임(예)	ACK
HEX 값	H05

설명

- 시작번호 : 3 Bytes
- 개수 : 3 Bytes

4.8 파라미터 관련

4.8.1 컨트롤러의 파라미터 버전 읽기

개요

현재 제어기에 탑재된 S/W의 파라미터 버전을 읽습니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		종료	체크
프레임(예)	ENQ	**K**	**V**	ETX	LRC
HEX값	H05	H4B	H56	H03	

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0	TEXT	ETX	LRC
HEX값	H05	H30		H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작	종료	체크
프레임(예)	ACK	ETX	LRC
HEX값	H05	H03	

설명

- DATA(5bytes) : 현재 제어기에 탑재된 S/W의 파라미터 버전

4.9 기타

4.9.1 기능 실패 원인 읽기

개요

프로토콜 요구 중 FLAG 가 4(H34)가 왔을 경우에는 기능 실패로써 그 원인을 알 수 있습니다. 텍스트 데이터로 최대 100 바이트 입니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		종료	체크
프레임(예)	ENQ	**M**	**D**	ETX	LRC
HEX 값	H05	H4B	H44	H03	

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0	TEXT	ETX	LRC
HEX 값	H05	H30		H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작	종료	체크
프레임(예)	ACK	ETX	LRC
HEX 값	H05	H03	

설명

- DATA : 기능 실행 실패에 대한 에러 원인 텍스트 데이터 입니다.

4.9.2 설정된 채널의 개수 및 ID 읽기

개요

현재 제어기에 설정된 Active 한 채널의 개수 및 ID 를 읽습니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		종료	체크
프레임(예)	ENQ	**K**	**H**	ETX	LRC
HEX 값	H05	H4B	H48	H03	

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	DATA0	DATA1	종료	체크
프레임(예)	ENQ	0	1	채널의 ID	ETX	LRC
HEX 값	H05	H30	H31		H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작	종료	체크
프레임(예)	ACK	ETX	LRC
HEX 값	H05	H03	

설명

- DATA0(1byte) : 설정된 채널의 개수
- DATA1(4byte) : 설정된 채널의 ID

예) 현재 제어기에 채널 1 과 채널 2 가 설정되어 있으면

DATA0 : '2' -> 2 개의 채널

DATA1 : "1200" -> 각 채널의 ID 는 1,2

4.9.3 채널의 보유 축 수 읽기

개요

현재 제어기에 설정된 Active 한 채널이 보유하고 있는 축 수를 읽습니다.

- 요구 포맷(**HOST** → 컨트롤러)

	시작	명령어		채널	종료	체크
프레임(예)	ENQ	**K**	**I**	1~4	ETX	LRC
HEX 값	H05	H4B	H49	H31~H34	H03	

- 응답 포맷 (컨트롤러 → **HOST**)

	시작	FLAG	DATA	종료	체크
프레임(예)	ENQ	0	4	ETX	LRC
HEX 값	H05	H30	H34	H03	

- 응답 포맷(**HOST** → 컨트롤러)

	시작	종료	체크
프레임(예)	ACK	ETX	LRC
HEX 값	H05	H03	

설명

- DATA(1byte) : 현재 제어기에 설정된 Active 한 채널이 보유하고 있는 축 수

개정 사항

4.10 3 차 수정본(2004.9)

4.10.1 개정사항 항목 추가

4.10.2 알파벳순 프로토콜 정리 표 추가

사용되지 않는 프로토콜 이름을 쉽게 파악하여 차후 프로토콜 추가시 편리하게 활용할 수 있도록 함.

4.10.3 Proface용 Protocol 추가

PE, PF
MF, MG
VD, VE, VF
KG,
FP, FQ, FS
QA, QB, QC, QD
이상 15개 항목 추가.

4.11 4 차 수정본(2004.10)

4.11.1 'QC'프로토콜 수정

Host->제어기 전송시 DATA를 포함하도록 함.

4.11.2 포인트 파일 번호 설정 프로토콜 추가

GP 프로토콜 구현상 작화시 Slave 의 데이터 어드레스 설정이 복잡해지면 구현이 어렵게 되므로 포인트 파일 읽기/쓰기 프로토콜에서 포인트 파일 번호를 설정하는 프로토콜을 따로 만듬.('FU','FV'추가)

4.11.3 Lamp표시용으로 접점 명령을 읽을 수 있도록 함.

'PF' 프로토콜 추가

4.11.4 포인트 파일과 전역위치변수에 대해서도 쓰기와 마찬가지로 포인트번호를 고정 또는 축 고정으로 하는 2가지 모두 가능하도록 함.

4.11.5 오타 수정

4.12 Ver 5 (2005-04-01)

4.12.1 ~차 수정본을 Ver~ 표기방식으로 개정

4.12.2 'PM'프로토콜 추가

4.12.2.1 PLC Ladder 편집기에서 접점 모니터링에 사용

4.12.3 'PE'프로토콜 수정(ver 5.2)

포인트 파일 복사, 삭제 기능 추가.

4.12.4 오타 수정(ver 5.3, 2005-10-18)

개요 부분 ETX와 (DATA)바뀐 것 정정, 연속된 패킷에 대한 설명 추가.

4.12.5 4.6.2 'PW'프로토콜 오타 수정. (ver 5.3)

마지막에 ACK 송신 부분이 잘못됨.

4.12.6 프로토콜 추가(ver 5.4)

(1) 'FW'프로토콜 추가
파라미터 정의 파일을 제어기에서 호스트 또는 DaWin_Pro로 전송

(2) 'PH'프로토콜 추가
모든 접점의 내용을 호스트 또는 DaWin_LD로 전송