Communication on RS485

MD co., Ltd

1. Introduction

Communication protocol(RS485) on the BLDC/DC motor controller.

This protocol is referenced by SAE J1587(automotive diagnostic protocol standard)

■ MD series BLDC motor controller(OP->Option)

Name	Volt	Curr-ent(A)	RS 485	TTL 232	CAN	ENC	PULSE	RC	CLUTCH	POW_SW
MD50	DC12~24	3								
MD50C	DC12~24	3.5	○			○				
PNT50	DC12~24	3x2	○			○				
MD100	DC12~24	7								
MD200	DC12~48	10	○			○				○
MD200T	DC12~48	10x2	○			○	○		○	○
MD400	DC12~48	20	○	○		○	○	○	○	○
MD400T	DC12~48	20x2	○	○	○	○	○	○	○	○
MD500S	DC12~48	20	○			○	○	○	○	○
MD750	DC24~72	30	○	○	○	○	○	○	○	○
MD750T	DC24~72	30x2	○	○	○	○	○	○	○	○
MD1K	DC12~48	50	○	○	○	○	○	○	○	○
MD2K	DC24~48	100	○	○	○	○		○	○	○
MDA200	AC110~220	1.5	○(Op)	○		○				
MDA400	AC110~220	2.5	○(Op)	○	○(Op)	○				
MDA400C	AC110~220	2.5	○	○		○	○	○		
MDA1K	AC110~220	5	○	○	○	○	○	○	○	

- PULSE : speed control by 0~500kpps pulse input
- RC : Wireless RC input
- ENC : Encoder input available(A,B,Z)
- CAN : CAN communication, extended mode only
- TTL232 : TTL level RS232 (G, Rx, Tx, 5VDC)
- CLUTCH : To drive electric brake, or clutch(G, Vpp)
- RS485 : RS485 communication(G, 485+, 485-)
- POW_SW : Power switch

2. Contents

2.1 Packet structure(communication with motor controller, 183)

Header		ID Number	Parameter ID	Data number	Data	Check sum
RMID	TMID	ID	PID	NUM	DATA	CHK
183(const)	Variable	ID of motor controller	Variable	Variable	(1~n Bytes)	Variable

- RMID : Receiving Machine ID number, 183(motor controller, constan
- TMID : Transmitting Machine ID number(variable)

MID Types	MID	Remark
MAIN_CTR	128	Control board using MPU
MMI	172	PC, Touch screen, etc
MOT_CTR	183	Motor controller(by Nextec)
BRIDGE_CTR	172	Connecting controller MMI<->BRIDGE<->BDLC_CTR

- ID: Identification Number of each RMID(Motor controller Broadcasting ID is 254, 0xfe)
- PID : Parameter IDentification number
- CHK : Check Sum
- Header : PC->CTR(183, 172), CTR->PC(172, 183)
  BRIDGE->CTR(183, 172), CTR->BRIDGE(172, 183)

Data bytes on the PID range

PID Numer	0~127	128~191	192~253
Data bytes	1 byte	2 bytes	N data bytes

- Data bytes : RMID, TMID, ID, PID, Data number, data, ..., CHK.
- How to make check-sum :

  - Sending:
    BYTE byChkSend, byCHK;
    byChkSend = RMID+TMID+ID+PID+Data number + Data..;
    byCHK = ~byChkSen +1

  - Receiving:
    BYTE byChkRecv;
    byChkRecv = RMID+TMID+ID+PID+Data number + Data..+CHK;
    If byChkRecv is zero then the packet is normal
    Order of data byte : Low-byte first.

Default settings

- Controller responses when the data is requested from the master.
- Want to send command to every controller, Use the Broadcasting ID, 254(0xfe)
- Default baudrate :

AC controller(MDA200, MDA400 only) : 8 data bits, 1 stop bit, no parity, 9600bps

else : 8 data bits, 1 stop bit, no parity, 19200bps

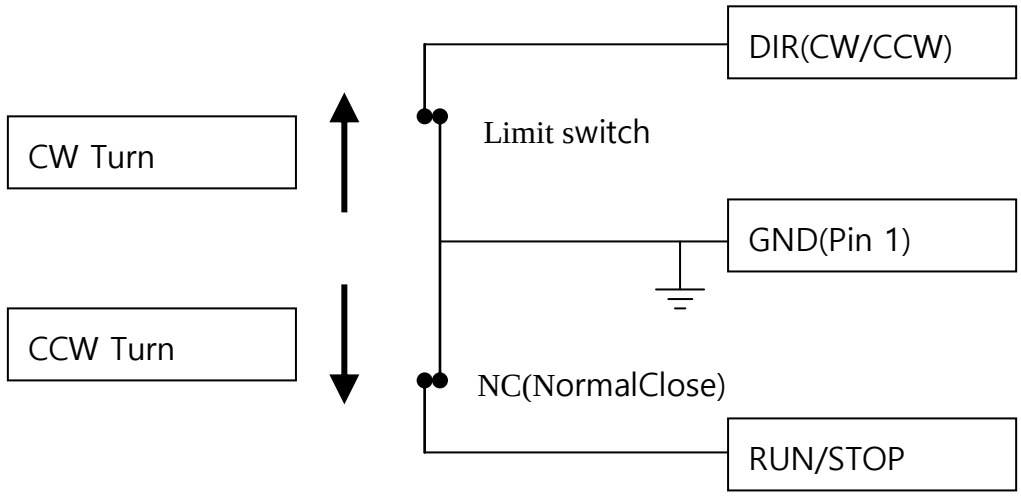
2.2 Driving condition when communition used

If you use communication command, then the controller uses the CTRL signals as limit switchs for system safety.

Set the DIR(CW/CCW) and START/STOP signals to ON(connected to GND) to drive motor

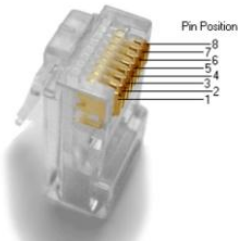
The relationship of the moving direction and the signal(Here, X is don't care)

Reference drection	**CTRL connector(no. 6 and no. 8 input signal)**		**Motor condition**
	DIR(CW/CCW)	**START/STOP**	
CW	**ON**	X	Driving
	OFF	X	Stop
CCW	X	**ON**	Driving
	X	OFF	Stop



Wiring condition of the moving direction and the limit

2.3 Specification on communication connector

Connector	Pin	Notation	Contents	Remark
RS485 YeonHo Elec. SMAW250-03	1 2 3	G 485+ 485-	RS485 connector(Option)	SMH250-03
CAN SMAW250-04	1 2	CAN_H CAN_L	Option	SMH250-02
TTL232(MDTS) MOLEX, 5267-04	1,2 3,4	G, RxTTL, TxTTL, 5VDC	TTL level RS232 signal 9600bps	MDTS related MOLEX, 5267-04
COM2(RJ45) (HA-108-NENL) (T568B connection)	1	Gnd	Ground	
	2	5V	5VDC, Power to external MDTS	
	3	RxD	TTL232 RxD(green/white)	
	4	485-	RS485-(blue)	
	5	485+	RS485+(blue/white)	
	6	TxD	TTL232 TxD(green)	
	7	CAN_H	CAN HIGH(brown/white)	
	8	CAN_L	CAN LOW(brown)	

2.4 PID(Parameter IDentification Number)

-R : Read only(request data by PID, PID_REQ_PID_DATA)
-W : Writing parameter (write to the flash memory)
-C : Command
-0xaa(170) : write check byte
-0xfe(254) : ID ALL(broadcasting ID)
-0x55 : default setting
-0x77 : PID gain writing(for developer)
-Condition : the data is requested and controlled from PC or MMI(MID, 172)

-1 Byte data(PID: 0~127)

PID	Type	PID Name/Remark	Contents of data bytes(Range)	Variable type Default value
1	R	PID_VERSION	x : version(13 -> 1.3) 172, 183, ID, 1, 1, x, CHK	BYTE
3	C	PID_DEFAULT_SET Default setting	Data : 0x55(CHECK) 183, 172, ID, 3, 1, 0x55, CHK	BYTE 0x55
4	**C**	**PID_REQ_PID_DATA Request data**	**R_PID : 0~253, wanted PID number. Want to get PID data, use this command. 183, 172, 1, 4, 1, PID, CHK**	**BYTE**
5	C	PID_TQ_OFF Stop naturally	Stop motor naturally, data don't care(x). 183, 172, ID, 5, 1, x, CHK	BYTE
6	C	PID_BRAKE Erectric brake	Stop motor urgently(electric braking mode) 183, 172, ID, 6, 1, x, CHK	BYTE
7	R	PID_ACK	Return received PID(RcvPID) number 172, 183, ID, 7, 1, RcvPID, CHK	BYTE
10	C	PID_COMMAND CMD_TQ_OFF CMD_BRAKE CMD_MAIN.. BC_ON CMD_MAIN_BC_OFF CMD_ALARM_RESET CMD_POSI_RESET CMD_MONITOR_BC_ON CMD_MONITOR_BC_OFF CMD_IO_MONITOR_ON CMD_IO_MONITOR_OFF CMD_FAN_ON	Contents on CMD number 2 : Tq-off, motor free state 4 : Erectric brake 5 : PID_MAIN_DATA broadcasting ON 6 : broadcasing OFF 8 : Reset alarm 10 : Position reset(set position to zero) 11 : PID_MONITOR broadcasting ON 12 : Broadcasting off 13 : PID_IO_MONITOR BC ON 14 : PID_IO_MONITOR BC OFF 15 : Fan ON(motor cooling fan)	

		CMD_FAN_OFF CMD_CLUTCH_ON CMD_CLUTCH_OFF CMD_TAR_VEL_OFF CMD_SLOW_START_OFF CMD_SLOW_DOWN_OFF CMD_CAN_RESEND_ON, CMD_CAN_RESEND_OFF CMD_MAX_TQ_OFF CMD_ENC_OFF CMD_LOW_SPEED_LIMIT_OF F, ... HIGH.. CMD_SPEED_LI._OFF CMD_CURVE.. OFF CMD_STEP.. OFF CMD_UICOM_OFF CMD_UICOM_ON CMD_MAX_RP._OFF CMD_HALL_TY..OFF CMD_LOW..POT_OFF CMD_HIGH.POT_OFF CMD_MAIN..BC_ON2 CMD_MAIN..BC_OFF2 CMD_MONIT..BC_ON2 CMD_MONIT..BC_OFF2 CMD_IO_MONIT..BC_ON2 CMD_IO_MONIT..BC_OFF2	16 : Fan OFF 17 : Mechanical brake(clutch) ON 18 : Mechanical breka OFF 20 : Erase target vel, set by PID_TAR_VEL 21 : Erase target slow/start value 22 : Erase target slow/down vaule 23 : Send CAN data to RS485 serial port. 24 : Turn off resending of CAN data 25 : Erase target limit load(max. current) 26 : Cancel the use of encoder sensor. 27 : Cancel the set of low speed limit. 28 : Cancel the set of high speed limit. 29 : Cancel the set of low/high speed limits. 31 : Cancel set of curve fitting func. 32 : Cancel step input mode 44 : I/O control(ctrl 11pin cnt) available 45 : I/O control disable(when comm. is used) 46 : Cancel max. speed set by DIP SW 47 : Cancel set of motor hall type 48 : Cancel set of low limit of POT input 49 : Cancel set of high limit of POT input 50 : PID_MAIN_DATA, BC ON for 2nd motor 51 : PID_MAIN_DATA, BC OFF for 2nd motor 52 : PID_MONITOR, BC ON for 2nd motor 53 : PID_MONITOR, BC OFF for 2nd motor 54 : PID_IO_MONITOR, BC ON for 2nd motor 55 : PID_IO_MONITOR, BC OFF for 2nd motor 183, 172, ID, 10, 1, CMD, CHK	
12	C	PID_ALARM_RESET Reset alarm	Data don't care(x) 183, 172, ID, 12, 1, x, CHK	BYTE
13	C	PID_POSI_RESET Reset position, Motor position to zero	183, 172, ID, 13, 1, x, CHK	BYTE
14	C	PID_MAIN_BC_STATE Request broadcasting of PID_MAIN_DATA	DATA, 1 : PID 193 broadcasting on 0 : broasdcasting off 183, 172, ID, 14, 1, DATA, CHK	BYTE 0
15	C	PID_MONITOR_BC_STATE Request BC on/off of PID_MONITOR	DATA, 1 : PID 196 broadcasting on 0 : broasdcasting off 183, 172, ID, 15, 1, DATA, CHK	BYTE 0

<table>
<tr><th>PID</th><th>Type</th><th>PID Name/Remark</th><th>Contents of data bytes/ Range</th><th>Variable type
Default value</th></tr>
<tr><td>16</td><td>R/W</td><td>PID_INV_SIGN_CMD
Inverse of moving direction
Used to the left, right, two wheel driving system.</td><td>DATA
PC sending data , 1000 then the controller gets inversed data -1000
1 : Inverse of reference direction
0 : Don’t use inverse sign(normal command)
183, 172, ID, 16, 1, DATA, CHK</td><td>BYTE
0</td></tr>
<tr><td>17</td><td>R/W</td><td>PID_USE_LIMIT_SW
Safety limit switch func.</td><td>DATA
1 : CTRL connector on, 6,7 is used as limit switchs
0 : Cancel limit switch function.
183, 172, ID, 17, 1, DATA, CHK</td><td>BYTE
1</td></tr>
<tr><td>18</td><td>R/W</td><td>PID_INV_SIGN_CMD2</td><td>DATA
1 : 2nd motor command direction inversed.
0 : 2nd motor normal sign.
183, 172, ID, 18, 1, DATA, CHK</td><td>BYTE
0
PNT50
MD750T 적용</td></tr>
<tr><td>19</td><td>R/W</td><td>PID_INV_ALARM
Inverse the sign of alarm signal(output)</td><td>DATA
1 : Inverse the alarm signal on/off status
0 : Normal signal
183, 172, ID, 19, 1, DATA, CHK</td><td>BYTE
0</td></tr>
<tr><td>21</td><td>R/W</td><td>PID_HALL_TYPE
Set the poles of BLDC motor</td><td>DATA
Set the number of poles of motor
<table>
<tr><td>Pole</td><td>4</td><td>8</td><td>10</td><td>12</td><td>2</td><td>6</td></tr>
<tr><td>No.</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr>
</table>
an and over the no. 5 must be set by comm. input then, the pole number is following,
poles of motor = DL * 2
183, 172, ID, 21, 1, DATA, CHK</td><td>BYTE
0</td></tr>
<tr><td>24</td><td>R/W</td><td>PID_ZERO_VEL_CTRL
Control zero velocity</td><td>DATA
0 : If speed input is zero, don't control(free)
1 : Control zero speed
183, 172, ID, 24, 1, DATA, CHK</td><td>BYTE
0</td></tr>
<tr><td>25</td><td>R/W</td><td>PID_INPUT_TYPE
Set the user input type</td><td>DATA(Input type)
0 : Analog input(0~5V) or PWM(>5Khz)
1 : Joystick(0~2.5~5), center(2.5V)
2 : Pulse(2~500Kpps)
3 : RC servo(>250hz), 1.05~1.95ms</td><td>BYTE
0</td></tr>
</table>

			5 : Step(1~7 step input) Set the speed steps up to 7 step. The step input number is set by following BIT0 : INT_SPEED, CTRL No. 2 BIT1 : RUN/BRAKE, CTRL No. 7 BIT2 : START/STOP, CTRL No. 8 StepInputNumber = BIT0 + BIT1x2 + BIT2x4 183, 172, ID, 25, 1, DATA, CHK	

■ User input type(PID_INPUT_TYPE, CTRL Pin no. 10 or PULSE_IN, Pin no. 2)

MODE	Contents/Connector	Range			비고(digit)
		Input	Speed(position)	Center	
0	Analog voltage/CTRL	0~5VDC	0~max.	-	Default setting
	PWM/CTRL	Duty cycle	0~max.	-	More than 5Khz
1	Joystick voltage/CTRL	0~5VDC	-max.~+max.	2.5VDC	deadzone:2~3VDC
2	Pulse/PULSE_IN	5~500kpps	0~max.		
3	RC(pulse)(>250Hz) /PULSE_IN	1.05~1.95ms	min-center-max	1.5ms	deadzone:1.4~1.6ms
5	Step(digital input)/CTRL BIT0:INT_SPEED BIT1:RUN/BRAKE BIT2:START/STOP	0~7	Set value	-	0 : Stop the motor 1~7 : drive motor by set value.

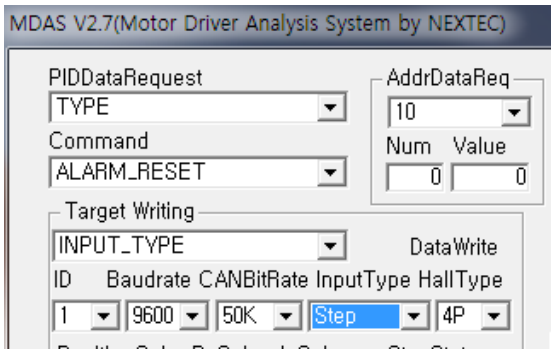
■ Step input

7steps speed setting

STEP INPUT(CTRL connector)				Default setting(%)
No.	INT_SPEED	RUN/BRKAKE	START/STOP	Percentage of max. speed
0	OFF	OFF	OFF	0(stop condition)
1	ON	OFF	OFF	14
2	OFF	ON	OFF	28
3	ON	ON	OFF	42
4	OFF	OFF	ON	57
5	ON	OFF	ON	71
6	OFF	ON	ON	85
7	ON	ON	ON	100

■ Input mode setting.

-Select wanted input type in the InputType combo box
-Select INPUT_TYPE in DataTypes combo box
-Press WriteData button.



PID	Type	PID Name/Remark	Contents of data bytes/ Range	Variable type Default value
30	C	PID_PRESET_SAVE Save preset position	DATA Preset number(address, 1~20) Set current position to the preset address 183, 172, ID, 31, 1, DATA, CHK	BYTE
31	C	PID_PRESET_RECALL Go to the recalled preset position	DATA Preset number(address, 1~20) Recall the saved preset data and move to that position.(position control) 183, 172, ID, 32, 1, DATA, CHK	BYTE
34	R	PID_CTRL_STATUS Control status	DATA BIT0 : ALARM, (1-> alarm status, 0->normal) BIT1 : CTRL_FAIL, Speed control fail BIT2 : OVER_VOLT, Over voltage BIT3 : OVER_TEMP, Over temperature BIT4 : OVER_LOAD, Overload BIT5 : HALL_FAIL, Hall sensor or encoder fail BIT6 : INV_VEL, Motor speed inversed BIT7 : STALL, motor not moved 184, 183, ID, 34, 1, DATA, CHK	BIT 0

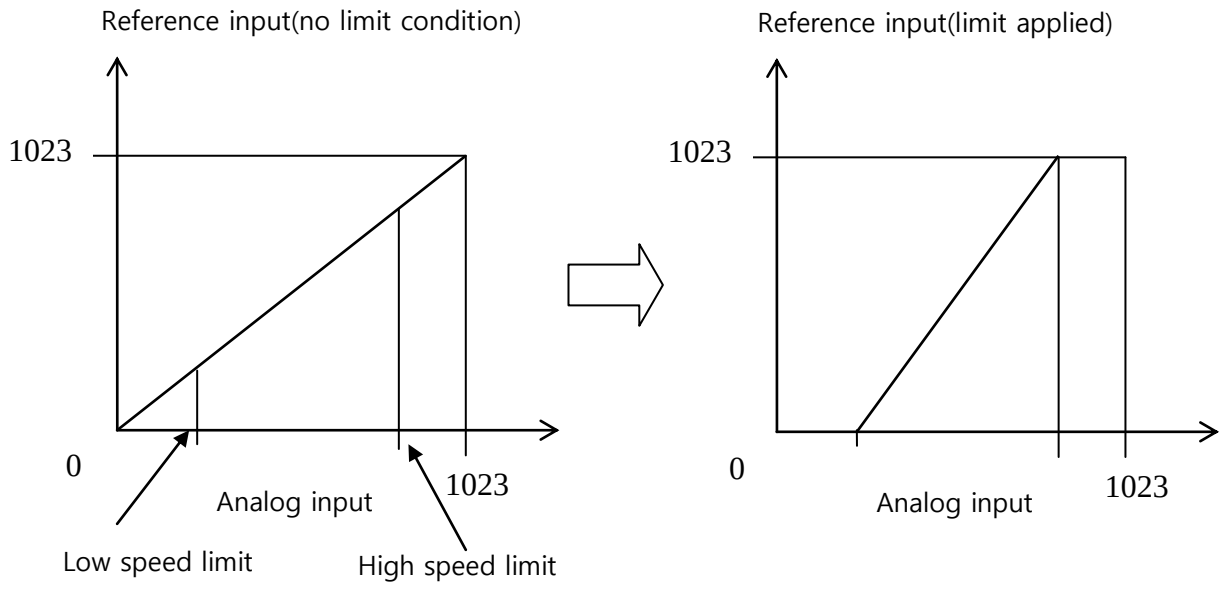
PID	Type	PID Name/Remark	Contents of data bytes/ Range	Variable type
36	R/W	PID_START_INV_SIGN START/STOP signal inverse	DATA 1 : Inverse START/STOP signal(HIGH->ON) 0 : Normal signal(LOW->ON) 183, 184, ID, 36, 1, DATA, CHK	BYTE 0
37	R/W	PID_RUN_INV_SIGN RUN/BRAKE signal inverse	DATA 1 : Inverse RUN/BRAKE signal(HIGH->ON) 0 : Normal signal(LOW->ON) 183, 184, ID, 37, 1, DATA, CHK	BYTE 0
38	R/W	PID_REGENERATION Regeneration function use or not	DATA 1 : Regen. function ON, 0 : Do not regeneration action(when the motor is turned by external load, then not resist to sustain the reference speed. 183, 184, ID, 38, 1, DATA, CHK	BYTE 0
40	R/W	PID_LIMIT_STOP_COND Set the stop condition when limit switch is ON	DATA Stop status when limit switch is ON(CW/CCW and STAR/STOP) 1 : BRAKE(electric braking) 0 : Natural stop(Free, TqOff) 183, 184, ID, 40, 1, DATA, CHK	BYTE 0
41	R/W	PID_TQ_LIMIT_SW Use the current level as limit switchs	DATA 1 : Over 90% of max. set current, stop the motor, 0 : Not use current limit switch function 183, 184, ID, 41, 1, DATA, CHK	BYTE 0
44	R/W	PID_TQ_CTRL Current(torque) control	DATA 1 : Current control(Torque) 0 : Speed control 183, 184, ID, 44, 1, DATA, CHK	BYTE 0
45	R/W	PID_BLUETOOTH	DATA 1 : Bluetooth commucation, 0 : don't care 183, 184, ID, 44, 1, DATA, CHK	BYTE 0
100	C	PID_START_STOP If there is a target speed(by PID180) then control like as START/STOP, DIR I/O act.	DATA 0 : STOP, 1 : CCW direction driving 2 : CW direction driving Refer to PID_COM_TAR_SPED(180) Target speed is set by PID_COM_TAR_SPEED 183, 184, ID, 100, 1, DATA, CHK	BYTE 0

- 2 Bytes data(PID: 128~192)

PID	Type	PID Name/Remark	Contents of data bytes/ Range	Variable type Default value
130	**C**	**PID_VEL_CMD** **Velocity command** **(unit : rpm)**	**Velocity command** **Speed(rpm) = (D1 | D2<<8)** **Speed>0, CCW direction** **Speed<0, CW direction** **183, 172, ID, 130, 2, D1, D2, CHK**	**INT**
131	C	PID_VEL_CMD2 Velocity command used more than 25,000rpm	High speed command RPM = (D1 | D2<<8) x 10 Unit : 10rpm 183, 172, ID, 131, 2, D1, D2, CHK	INT
133	W	PID_ID ID setting	Write command(0xaa) ID : 1~253 : setting ID 183, 172, 254, 133, 2, 0xaa, ID, CHK	BYTE 1
134	**C**	**PID_OPEN_VEL_CMD** **Open-loop control**	**D1, D2: Open-loop velocity** **Range : -1023~1023** **183, 172, ID, 134, 2, D1, D2, CHK**	**INT**
135	W	PID_BAUDRATE Baudrate setting(RS485)	Set the baudrate, BAUD 1 : 9600bps, 2 : 19200bps 3 : 38400bps, 4 : 57600bps 5 : 115200bps 183, 172, ID, 135, 2, 0xaa, BAUD, CHK	BYTE 1:AC power 2:DC power
137	W	PID_ECAN_BITRATE Set CAN bitrate	Set the CAN bitrate(bits/s), BIT_RATE 1 : 50k, 2 : 100k 3 : 250k, 4 : 500k, 5 : 1M 183, 172, ID, 137, 2, 0xaa, BIT_RATE, CHK	BYTE 1
138	R	PID_INT_RPM_DATA Motor speed (16bits data)	Motor speed : DATA1, DATA2 RPM = (D1 | D2<<8) 183, 172, ID, 138, 2, D1, D2, CHK	INT
139	R	PID_TQ_DATA Current(0.1A)	Return current(unit 0.1A) 10->1A, 15->1.5A Current = (D1 | D2<<8)/10 172, 183, ID, 139, 2, D1, D2, CHK	INT

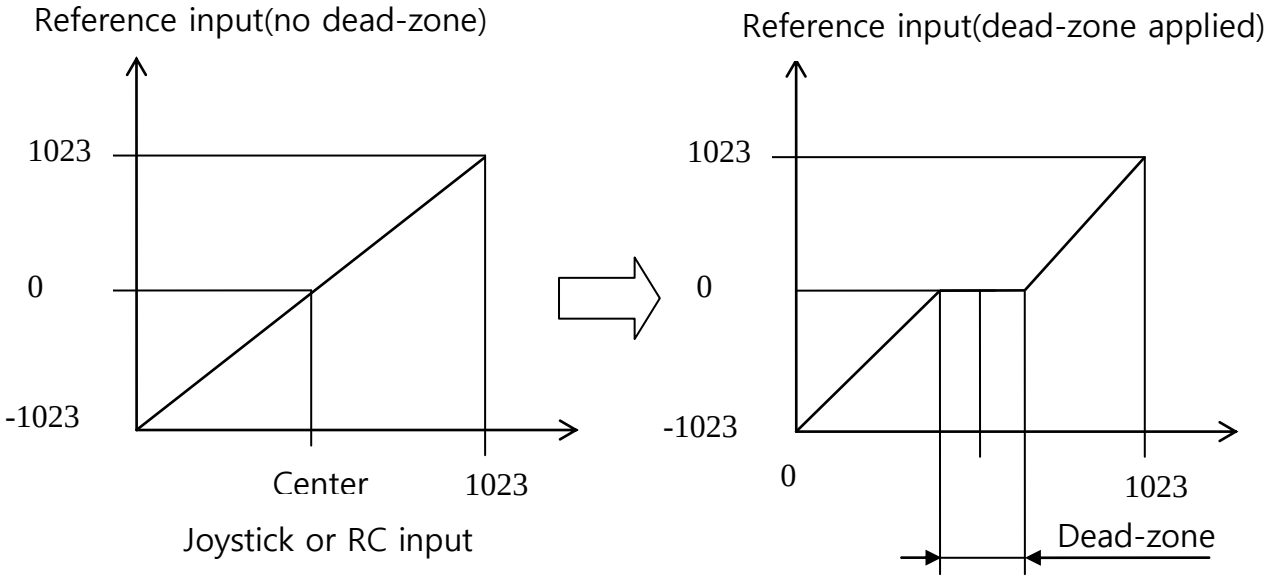
PID	Type	PID Name/Remark	Contents of data bytes/ Range	Variable type Default value
140	C	PID_TQ_CMD Torque(current) control	Target current input : D1, D2 Range of input = (D1 | D2<<8), -1023~1023 184, 183, ID, 140, 2, D1, D2, CHK	INT
143	R	PID_VOLT_IN Supply voltage	Return supply voltage Unit of DC power : 0.1V Unit of AC power : 1V(internal DC voltage) Voltage = D1 | (D2<<8) 172, 183, ID, 143, 2, D1, D2, CHK	INT
146	R/W	PID_CCW_PHASE_OFFSET CCW phase lead angle	Phase lead angle, CCW_PO(0~30deg) 183,172,ID,146,2,0xaa,CCW_PO,CHK	BYTE
147	R/W	PID_CW_PHASE_OFFSET CW phase lead angle	Phase lead angle, CCW(0~30deg) 183,172,ID,147,2,170,CW_PO,CHK	BYTE
149	R/W	PID_RETURN_TYPE Set return data type When command received, Controller send PID data set by RET_TYPE.	RET_TYPE 0 : No return 1 : PID_MONITOR 2 : PID_ACK(return received PID) 3 : PID_IO_MONITOR 183, 172, ID, 149, 2, 170, RET_TYPE, CHK	BYTE 0
152	R/W	PID_OVER_MODULATION Set over-modulation func.	OVER_MOD 1 : Over modulation applied(noise incresed but output power and effiency incresed) 0 : normal modulation 183,172, ID,152,2,170,OVER_MOD,CHK	BYTE 0
153	R/W	PID_SLOW_START Set the SlowStart variable (not use internal volume)	Value of SlowStart(0~1023->0~15s delay) Applied when the ref. speed increased SS = (D1 | D2<<8) 183, 172, ID, 153 ,2, D1, D2, CHK	INT 0
154	R/W	PID_SLOW_DOWN Set the SlowDown variable	Value of SlowDown(0~15s) Applied when the ref. speed decreased. SD = (D1 | D2<<8) 183, 172, ID, 154, 2, D1, D2, CHK	INT 0

PID	Type	PID Name/Remark	Contents of data bytes/ Range	Variable type Default value
155	R/W	PID_TAR_VEL Use fixed set speed, no use internal volume(SPEED_IN)	Replace internal volume, SPEED_IN TAR_VEL = Data, 0~MaxRPM(about 5000rpm) 183, 172, ID, 155, 2, D1, D2, CHK	INT 0
156	R/W	PID_ENC_PPR Set encoder pulse	When the encoder pulse is set. The controller uses encoder signals as a speed sensing input. ENC_PULSE = (D1 | D2<<8) 183, 172, ID, 156, 2, D1, D2, CHK	INT 0
157	R/W	PID_LOW_SPEED_LIMIT Set the low value of analog input(lower then this value is same as zero input)	Setting range is 0~512 If the input is lower than LOW_LIMIT This input is treated as a zero LOW_LIMIT = (D1 | D2<<8) 183, 172, ID, 154, 2, D1, D2, CHK	INT 0
158	R/W	PID_HIGH_SPEED_LIMIT Set the higher value of analog input	Setting range is 512~1023 Higher than this HIGH_LIMIT is same to HIGH_LIMIT, HIGH_LIMIT = (D1 | D2<<8) 183, 172, ID, 158, 2, D1, D2, CHK	INT 0



Reference input vs. input voltage by SPEED_LIMIT

PID	Type	PID Name/Remark	Contents of data bytes/ Range	Variable type Default value
160	R/W	PID_PWM_OUT Out PWM of OUT3 terminal When DC motor control	At DMD series controller, Out PWM output (0~1023) PWM_OUT = (D1 | D2<<8) 183, 184, ID, 160, 2, D1, D2, CHK	BYTE 0
162	R/W	PID_DEAD_ZONE Refer to the below figure.	Setting range is 0~1023 Applied at J/S and RC input mode. DeadZone = (D1 | D2<<8) 183, 172, ID, 162, 2, D1, D2, CHK	INT
164	R	PID_REQ_PID_DATA2 Used to request below PID PID_CURVE_PT PID_STEP_INPUT PID_PRESET_DATA	Request PID data with a parameter REQ_PID : Requested PID PARAM PID_CURVE_PT : PT number(1,2,3,4) PID_STEP_INPUT : Step number(1~7) PID_PRESET_DATA : Preset number(1~20) 183, 172, ID, 164, 2, REQ_PID, PARAM, CHK	BYTE BYTE



Reference input by dead-zone setting(Joystic and RC input)

PID	Type	PID Name/설명	Contents of data bytes/ Range	Variable type Default value
166	R	PID_REF_RPM Reference velocity	REF_RPM : reference velocity of motor REF_RPM = (D1 | D2<<8), CW(-), CCW(+) 172, 183, ID, 166, 2, D1, D2, CHK	INT
167	R/W	PID_PV_GAIN P-gain of position control	PV_GAIN = (D1 | D2<<8) 183, 172, ID, 167, 2, D1, D2, CHK	INT
168	R/W	PID_P_GAIN P-gain for velocity control	P_GAIN : Proportional gain for velocity control P_GAIN = (D1 | D2<<8) 183, 172, ID, 168, 2, D1, D2, CHK	INT
169	R/W	PID_I_GAIN I-gain for velocity control	I_GAIN : Integral gain for velocity control I_GAIN = (D1 | D2<<8) 183, 172, ID, 169, 2, , D1, D2, CHK	INT
170	R/W	PID_TQ_P_GAIN P-gain for tq(current) control	TQ_P_GAIN : Proportional gain for tq. control TQ_P_GAIN = (D1 | D2<<8) TqErr = Reference torque – motor torque Output = TQ_P_GAIN*TqErr/GAIN_FACTOR 183, 172, ID, 170, 2, D1, D2, CHK	INT
171	R/W	PID_IN_POSITION Resolution for the position control	IN_POSITION : resolution for position control Ex) IN_POSITION->10, the error less than 10, stop the control action. IN_POSITION = (D1 | D2<<8) 183, 172, ID, 167, 2, D1, D2, CHK	INT
174	C	PID_PNT_TQ_OFF Stop two motors for the controller which drive two motors.	D1, D2 : Tq off condition on motor1, 2 1->TqOff(stop), 0->Don't care D3 : Return data type 0 : no return data 1 : return PID_PNT_MONITOR 2 : return PID_PNT_MAIN_DATA 183, 172, ID, 174, 3, D1, D2, D3, CHK	BIT, BIT PNT50 MD750T
175	C	PID_PNT_BRAKE Brake command	D1, D2 : Brake condition on motor1, 2 1->Brake , 0->Don't care D3 : Return data type 0 : no return data 1 : return PID_PNT_MONITOR 2 : return PID_PNT_MAIN_DATA 183, 172, ID, 175, 3, D1, D2, D3, CHK	BIT, BIT PNT50 MD750T

176	R/W	PID_TAR_POSI_VEL Max. speed in position control	D1, D2 : max. speed in position control TAR_SPEED = (D1 | D2<<8) rpm 183, 184, ID, 176, 2, D1, D2, CHK	WORD
177	R/W	PID_TQ_I_GAIN Integral gain in torque control	D1, D2 : I-gain in torque control TQ_I_GAIN = (D1 | D2<<8) 183, 184, ID, 177, 2, D1, D2, CHK	WORD
178	R/W	PID_POSI_SS	Slow/Start(SS, 0~1023) in position control 183, 184, ID, 178, 2, D1, D2, CHK	INT
179	R/W	PID_POSI_SD	Slow/Dow(SD, 0~1023) in position control 183, 184, ID, 179, 2, D1, D2, CHK	INT

- N Bytes data(PID: 193~240)

PID	Type	PID Name/Remark	Contents of data bytes/ Range	Variable type Default value
180	C/R	PID_COM_TAR_SPEED Refer to PID_START_STOP, 100	D1, D2 : Target speed(rpm) TAR_SPEED(rpm)= (D1 | D2<<8) 183, 184, ID, 180, 2, D1, D2, CHK	WORD
183	**R/W**	**PID_FUNC_CMD_TYPE If the function type is set(more than 1) then the motor used by start/stop only. (option)**	**D1, D2 : Special function type Refer to the function below explanation FUNC_TYPE = D1, D2 don't care 0 : NONE 1 : SPEED 2 : POSITION 3 : SPEED_MONENTARY 4 : SPEED_ADD Refet to the following explanation 183, 184, ID, 183, 2, D1, x, CHK**	**WORD**
184	**W**	**PID_FUNC_CMD If function type is, then user can use start/stop signal and this PID also (option)**	**D1, D2 : Run the function set by PID 183 Like as control input signal START/STOP Refer to the function below explanation D1 : 1 or 1, D2 don't care 0 : OFF, more than 1 : ON 183, 184, ID, 184, 2, D1, D2, CHK**	**WORD**
193	R	PID_MAIN_DATA Main data	Data : 16 bytes(D1~D16) D1,2 : Speed(rpm) D3,4 : Current (0~1023, 0.1A unit) D5 : Control type. 0 : Tq-off(stop) 1 : Speed control(closed-loop control) 2 : Position control 3 : Open-loop control D6,7 : Reference speed(rpm) D8,9 : Control output D10: Status of controller(refer to below.) D11,12,13,14 : Motor position D15 : Brake duty(regernarative resistor, 0~255) D16 : Temperature(0~100deg) If there is no temp. sensor D16 will be zero. 172, 183, ID, 193, 16, D1, ..., D16, CHK	 INT INT BYTE INT LONG BYTE BYTE

PID	Type	PID Name/Remark	Contents of data bytes/ Range	Variable type
194	R	PID_IO_MONITOR Used when speed in/out range is bigger than 327676 rpm	Data : 17bytes D1,2 : motor speed(rpm) Speed(rpm) = (D1 | ((int)D2<<8))x10 D3,4 : current (0~1023, 0.1A unit) D5 : status on controller D6 : input signal(CTRL, ENC) BIT0 : INT_SPEED BIT1,2 : ALARM_RESET, DIR(CW/CCW) BIT3,4 : RUN/BRAKE, START/STOP BIT5,6 : ENC_B, ENC_A D7,8 : external speed input(0~1023) D9 : 8PIN dip_sw input(0~255) D10 : hall sensor signal(Hu, Hv, Hw) D11 : BIT0~3 : controller types 1:MDA400C, 2:MDA1K, 3:MDA2K 4:MD750, 5:MDA200, 10:MD400 D12 : etc(switch input) BIT0 : DIP_SW_STEP BIT1 : DIP_SW_INV_ALM BIT2 : TEMP. switch(MDA400) D13,14 : input voltage(0.1Vunit, 240->24.0V) VoltIn(volt) = (D13 | ((int)D14<<8))x10 D15: Slow/Start(SS, 0~100) D16:Slow/Down(SD,0~100) D17:Load(current limit, 0~100) 184, 183, ID, 194, 17, D1, ..., D17, CHK	INT INT BIT BIT INT BIT BYTE

PID	Type	PID Name/Remark	Contents of data bytes/ Range	Variable type
196	**R**	**PID_MONITOR**	**Data : 11 Bytes** **D1,2 : Speed(rpm)** **D3,4 : Current(0~1023, 0.1A unit)** **D5,6 : Controller output(0~1023)** **D7 : Status of controller** **D8,9,10,11 : Position** **D12(reserved)** **172, 183, ID, 196, 12, D1,..., D12, CHK**	**INT** **INT** **INT** **BIT** **LONG**
197	R	PID_POSI_DATA Motor position	Data : 4 bytes D1,2,3,4 : Position of motor 172, 183, ID, 197, 4, D1, D2, D3, D4, CHK	LONG
198	R	PID_RPM_DATA Motor speed	Data : 4 bytes D1,2,3,4 : Motor speed(rpm) 172, 183, ID, 198, 4, D1, D2, D3, D4, CHK	LONG
200	R	PID_MAIN_DATA2 MAIN_DATA of motor2	Same with PID193, but data is on the motor2 MAIN_DATA of motor2	PNT50 MD750T
201	R	PID_IO_MONITOR MONITOR of motor2	Same with PID196, but data is on the motor2 MONITOR of motor2	PNT50 MD750T
202	R	PID_ IO_IO_MONITOR IO_MONITOR of motor2	Same with PID194, but data is on the motor2 IO_MONITOR of motor2	PNT50 MD750T
203	R/W	PID_ GAIN Control gain	Data : 6 bytes D1,2 : proportional gain of positon control D3,4 : proportional gain of speed control(P) D5,6 : integral gain of speed control(I) 172, 183, ID, 203, 6, D1, ..D6, CHK	INT INT INT
205	R	PID_TYPE Type of controller	Data : max. 20 bytes (character) If the type is MD1K-V1.0, then n is 9(number) 172, 183, ID, 205, n, D1, ..Dn, CHK	CHAR

PID	Type	PID Name/Remark	Contents of data bytes/ Range	Variable type
206	C	PID_PNT_POSI_CMD Position control for two-motor-driver (MDT controller, PNT50, MD200T, MD400T, MD750T)	Data : 15Bytes D1 : ID of controller1 When two motor conroller used (like PNT50..) D1 : 1:Enable, 0:Disable of motor1 D2,3,4,5 : reference position of motor1 D6,7 : max. velocity on motor1 D8 : ID of controller2 At two motor conroller :enable(1) or disable (0) D9,10,11,12 : reference position of motor2 D13,14 : max. velocity on motor2 D15: return data type 0(no return), 1(PNT_MONITOR), 2(PNT_MAIN_DATA) 183, 172, ID, 206, 15, D1, D2, .., D15, CHK	BYTE LONG BYTE LONG
207	C	PID_PNT_VEL_CMD Velocity control for two-motor-driver (MDT controller, PNT50, MD200T, MD400T, MD750T)	Data : 7Bytes D1 : ID of controller1, 0:disable D2,3 : reference velocity on motor1(rpm) D4 : ID of controller2, 0:disable D5,6 : reference velocity on motor2(rpm) D7: return data type 0(no return), 1(PNT_MONITOR), 2(PNT_MAIN_DATA) 183, 172, ID, 207, 6, D1, D2, .., D6, D7, CHK	BYTE INT BYTE INT
208	C	PID_PNT_OPEN_VEL_CMD Open-loop control for two-motor-driver (PNT50, MD750T,MD400T)	Data : 7Bytes D1 : ID of controller1, 0:disable D2,3 : output on motor1(-1023~1023) D4 : ID of controller2, 0:disable D5,6 : output on motor2(-1023~1023) D7: return data type 0(no return), 1(PNT_MONITOR), 2(PNT_MAIN_DATA) 183, 172, ID, 208, 6, D1, D2, .., D7, CHK	BYTE INT BYTE INT

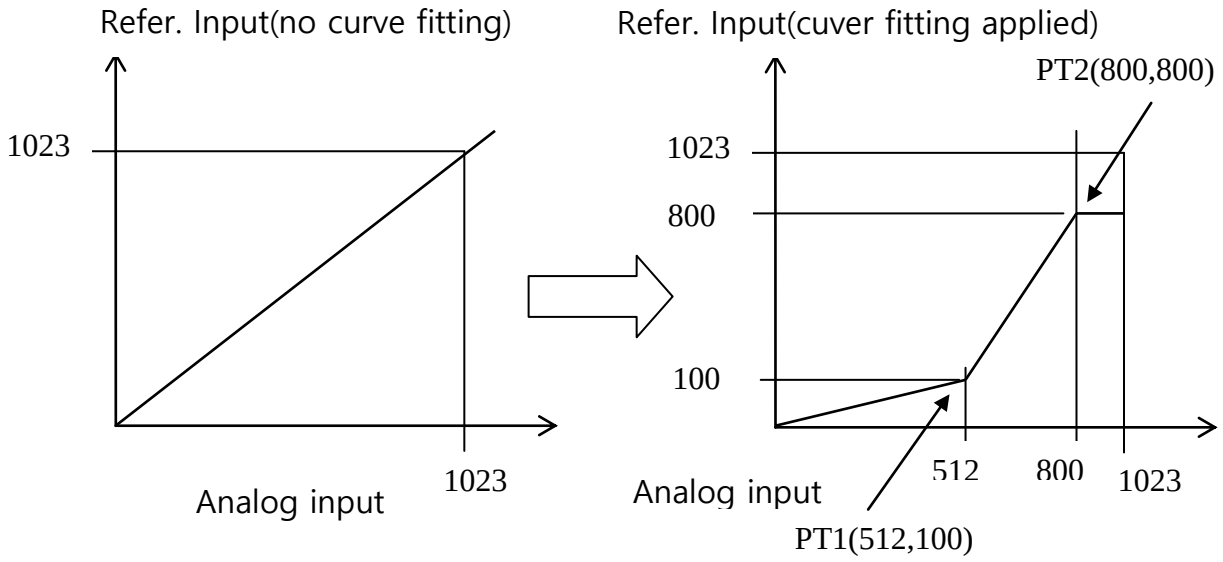
PID	Type	PID Name/Remark	Contents of data bytes/ Range	Variable type
209	C	PID_PNT_TQ_CMD Torque control for two-motor-driver	Data : 7Bytes D1 : ID of controller1, 0:disable D2,3 : reference tq. on motor1(-1023~1023) D4 : ID of controller2, 0:disable D5,6 : reference tq. on motor2(-1023~1023) D7: return data type 0(no return), 1(PNT_MONITOR), 2(PNT_MAIN_DATA) 183, 172, ID, 209, 6, D1, D2, .., D7, CHK	BYTE INT BYTE INT PNT50 MD750T
210	R	PID_PNT_MAIN_DATA MAIN_DATA for two-motor-driver	Data : 18 Bytes D1,2 : speed(rpm) of motor1 D3,4 : current of motor1(0.1A unit, 100->10A) D5 : status bit of motor1 D6,7,8,9 : position of motor1 D10,11: speed(rpm) of motor2 D12,13 : current of motor2 (0.1A unit, 100->10A) D14 : status bit of motor2 D15,16,17,18 : position of motor2 172, 183, ID, 210, 18, D1,..., D11, D18, CHK	INT INT BIT LONG PNT50 MD750T

-**Status bits**

BIT	Name	Contents
0	ALARM	Alarm detected, system abnormal.
1	CTRL_FAIL	The motor speed is larger than 30% of reference speed during 15s
2	OVER_VOLT	Supply voltage is over the set max. voltage.
3	OVER_TEMP	More than 65℃, At 55~65 deg, the output is limited proportionally.
4	OVER_LOAD	Detect more than set max. current over 4s. or 150% of max current is detected.(urgent alarm)
5	HALL_FAIL	Failed hall sensor, detected hall value is 0, or 7
6	INV_VEL	Sign of motor speed is not same with output sign.(inverse signal detect)
7	STALL	The speed is zero but the output is, more than 2s.

PID	Type	PID Name/Remark	Contents of data bytes/ Range	Variable type Default value
211	R/W	PID_MAX_LOAD Set the max. current	Data : 2Bytes D1,2 : Max. current available. 172, 183, ID, 211, 2, D1, D2, CHK	INT
215	C	PID_PNT_INC_POSI_CMD Increment position control for two controllers or two motor driving controller	Data : 11Bytes D1 : ID of controller1, 0:disable D2,3,4,5 : Relative position on motor1 D6 : Enable on motor2, 0:disable D7,8,9,10 : Relative position on motor2 D11: return data type 0(no return), 1(PNT_MONITOR), 2(PNT_MAIN_DATA) 183, 184, ID, 215, 15, D1, D2, .., D11, CHK	BYTE LONG BYTE LONG PNT50 MD750T MD400T
217	C	PID_POSI_SET Set the motor position	Data : 4 bytes D1,2,3,4 : (long type position data) Change the motor position by set value. 183, 172, ID, 217, 4, D1, D2, D3, D4, CHK	LONG
218	C	PID_POSI_SET2 Set the motor2 position	Data : 4Bytes D1,2,3,4 : set position on motor2 Change motor position by the set position. 183, 172, ID, 218, 4, D1, D2, D3, D4, CHK	LONG PNT50 MD750T
219	C	PID_POSI_VEL_CMD Position control with max. target speed.	Data : 6 bytes D1,2,3,4 : Ref. position D5,6 : max. control speed(rpm) If the speed is zero, then use half of max. 183, 172, ID, 219, 6, D1, .., D6, CHK	LONG INT
220	C	PID_INC_POSI_VEL_CMD Position command	Data : 6bytes D1,2,3,4 : reference position(incremental) D5,6 : max. speed(rpm) Current motor position is 100 And the ref. Position is 200, then motor go to the position 300 183, 184, ID, 219, 6, D1, .., D6, CHK	LONG INT
221	R/W	PID_MAX_RPM Max. speed(rpm)	Data : 2 bytes D1,2 : max. speed(rpm) 172, 183, ID, 221, 2, D1, D2, CHK	INT

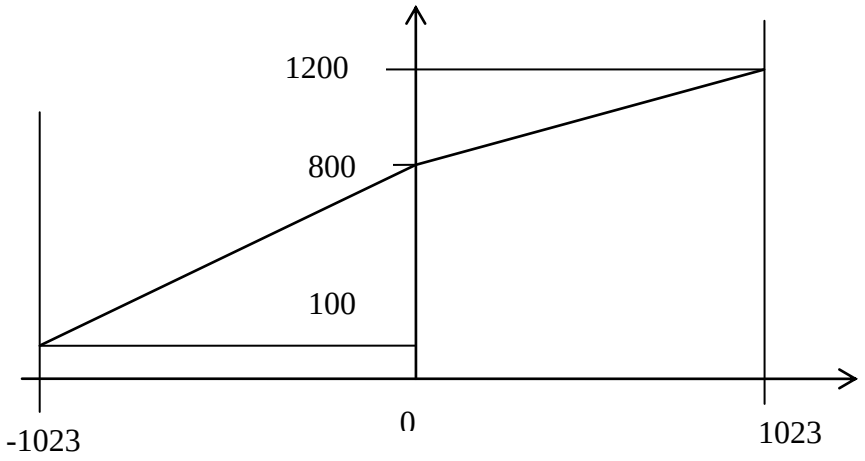
PID	Type	PID Name/Remark	Contents of data bytes/ Range	Variable type
222	R/W	PID_SPEED_LIMIT Refer to above fig. (Low/high speed limit)	Data : 4 bytes Limit speed input(analog input) D1,2 : low speed limit(0~512). D3,4 : high speed limit(512~1023) 183, 172, ID, 222, 4, D1, D2, D3, D4, CHK	 INT INT
225	R/W	PID_STEP_INPUT Set the step speed.	Data : 3 bytes Set the step input(7 steps) D1 : step number(1~7) D2,3 : speed(rpm) BIT0 : CTRL, Pin no. 2, INT_SPEED BIT1 : CTRL, Pin no. 7, RUN/BRAKE BIT2 : CTRL, Pin no.8, START/STOP Speed step(1~7, step 0 is zero velocity, stop) SpeedStepNumber = BIT0 + BIT1x2 + BIT2x4 183, 172, ID, 225, 4, D1, D2, D3, CHK	 BYTE INT
226	R/W	PID_CURVE_PT Set the curve fitting points	Data : 5 bytes D1 : Number of curvefitting point(1~4), Positive axis point(1:PT1, 2:PT2) Negative axis point(3:PT3, 4:PT4) D2,3 : point x(0~1023) must be>0 D4,5 : point y(0~1023), must be>0 183, 184, ID, 226, 5, D1, D2, D3, CHK	BYTE INT INT



Reference input vs. input voltage by curve fitting(PT3,4 are negative values)

PID	Type	PID Name/Remark	Contents of data bytes/ Range	Variable type Default value
227	R	PID_PRESET_DATA	Data : 5Bytes Return preset data(position) according to the preset number. D1 : preset number(1~20) D2,3,4,5 : Set preset data(position) 183, 172, ID, 227, 5, D1, .., D5, CHK	 BYTE LONG
230	R	PID_REF_POSI	Target(reference) position 183, 184, ID, 230, 4, D1, .., D4, CHK	LONG
231	R/W	PID_POSI_MIN_LIMIT	Data : 4Bytes Min. reference position to the min. input D1,2,3,4 : position 183, 184, ID, 231, 4, D1, .., D4, CHK	LONG
232	R/W	PID_POSI_CEN	Data : 4Bytes Reference center position to the zero input D1,2,3,4 : position 183, 172, ID, 232, 4, D1, .., D4, CHK	LONG
233	R/W	PID_POSI_MAX_LIMIT	Data : 4Bytes Max. refence position to the max. input. D1,2,3,4 : position 183, 172, ID, 233, 4, D1, .., D4, CHK	LONG

Reference position



Input range(volume, J/S, Etc)

Reference position to the values, MIN_LIMIT(100), CEN(800), MAX_LIMIT(1200)

PID	Type	PID Name/Remark	Contents of data bytes/ Range	Variable type Default value
238	R	PID_CAN_RESEND Resend CAN comm data to RS485 data(packet)	Data : 12bytes D1~4 : Extended ID D5~12 : CAN data(PID, Data...), 최대 8Bytes 172, 183, ID, 238, DataNum, D1, ..., Dn, CHK	LONG BYTE...
239	**R/W**	**PID_FUNC_SPEED** **With target speed the motor run for a run time(delay) by START/STOP signal (option)**	**Data : 4bytes** **D1,D2 : Target speed(rpm)** **D3,D4 : Delay(run time, 0.1s unit), 10->1s** **Target speed = D2*256 + D1** **183, 184, ID, 239, 4, D1, ..., D4, CHK**	**INT** **INT**
242	C	PID_PNT_INC_POSI_ VEL_CMD	Data : 15bytes D1 : 1(enable control on motor1), 0(don't control, there is no effect) D2~D5 : position incremental on motor1 D6,D7 : max. speed on motor1 with posi. ctrl. D8 : control enable on motor2 D9~D12: position incremental on motor2 D13,D14: max. speed on motor2 with posi. ctrl. D15: return data type(0, 1, 2) 0(no return), 1(PID_PNT_MONITOR) 2(PID_PNT_MAIN_DATA) 183, 184, ID, 242, 15, D1, ..., D15, CHK	INT
243	R/W	PID_POSI_CMD	Data : 4bytes D1~D4 : Target position D1(LSB), D4(MSB) 183, 184, ID, 243, 4, D1, ..., D4, CHK	LONG
244	R/W	PID_INC_POSI_CMD	Data : 4bytes D1~D4 : Incremental target position Target posi. = Current posi. + Incremental Posi. 183, 184, ID, 244, 4, D1, ..., D4, CHK	LONG

PID	Type	PID Name/Remark	Contents of data bytes/ Range	Variable type
246	R/W	PID_PNT_POSI_CMD	Data : 11bytes D1 : ID1(Command enable), 0(Disable) D2~D5 : Reference position of ID1 D6 : ID2(Command enable), 0(Disable) D7~D10 : Reference position of ID2 D11 : Controller ID which wanted PID_MONITOR(return data) 183, 184, ID, 246, 2, D1, ..., D11, CHK	LONG
250	R/W	PID_FUNC_POSI With target speed the motor move to the target position(option)	Data : 4bytes Target speed(rpm) = D2*256 + D1 Target position(motor poles*3/rev) = D4*256 + D3 183, 184, ID, 250, 4, D1, ..., D4, CHK	INT INT
251	R/W	PID_INC_POSI_CMD2	Data : 4bytes D1~D4 : Incremental target position on motor2 Target posi. = Current posi. + Incremental Posi. 183, 184, ID, 251, 4, D1, ..., D4, CHK	LONG

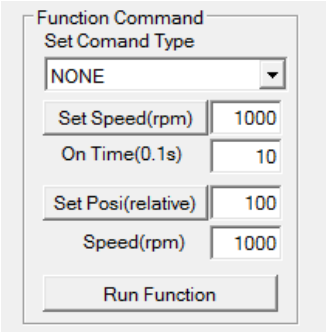
■ Function types(PID_FUNC_CMD_TYPE)

START/STOP : control input signal of controller, refer to each controller spec.

Related PID(PID_FUNC_CMD_TYPE, PID_FUNC_CMD, PID_FUNC_SPEED, PID_FUNC_POSI)

Special function types are 4 kinds, following

TYPE_SPEED/ TYPE_SPEED_MOMENTARY/ TYPE_SPEED_ADD/TYPE_POSI.

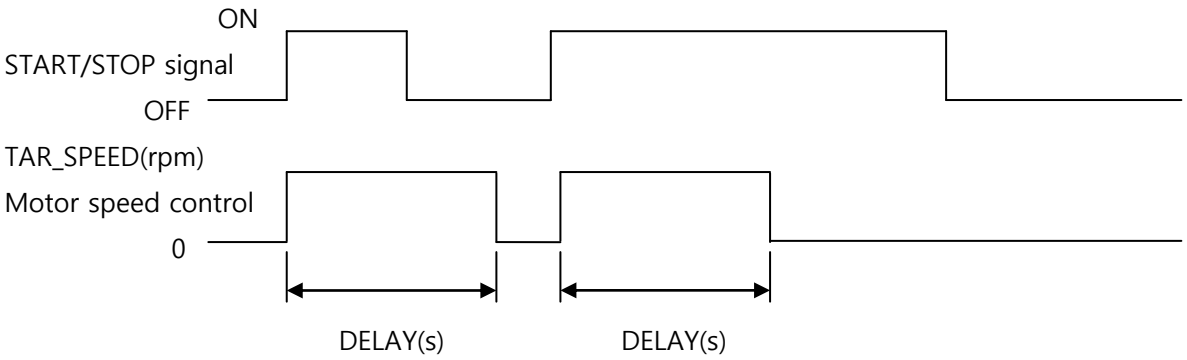


refer to the control panel of MDAS

- Function type, SPEED(1)

Motor moves when the signal of START/STOP is ON from OFF set by PID_FUNC_SPEED(target speed, and on time(delay))

X-axis is time(s)

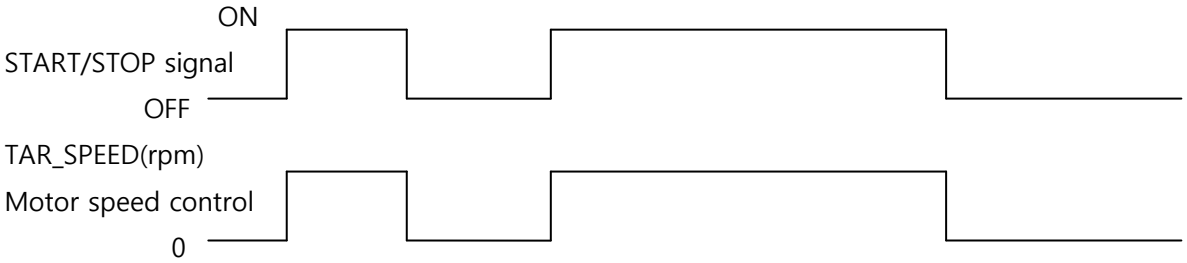


Here, the target speed and delay is set by PID_FUNC_SPEED(target speed(TAR_SPEED), and delay(DELAY))

- Function type, SPEED_MOMENTARY(3)

Motor moves when the signal of START/STOP is ON from OFF set by PID_FUNC_SPEED(target speed, and on time(delay))

Refer to following diagram.

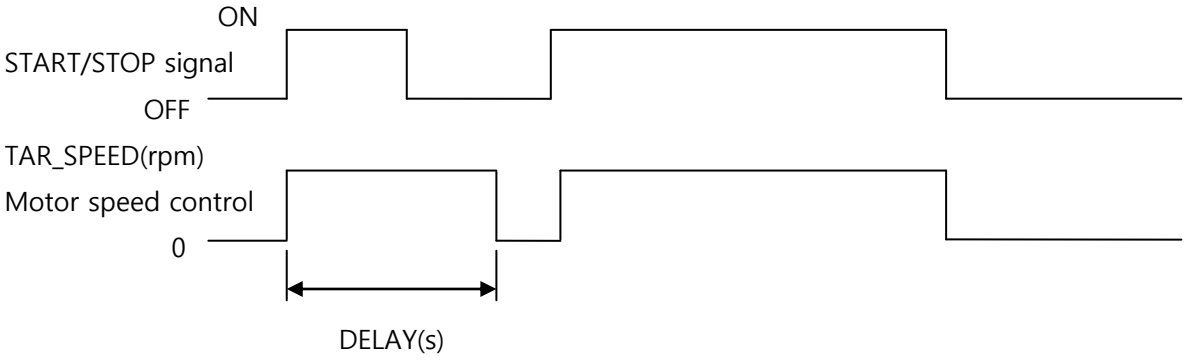


Here, the target speed only applied, on time is ignored.

- Function type, SPEED_ADD(4)

Motor moves when the signal of START/STOP is ON from OFF set by PID_FUNC_SPEED(target speed, and on time(delay))

Refer to following diagram.

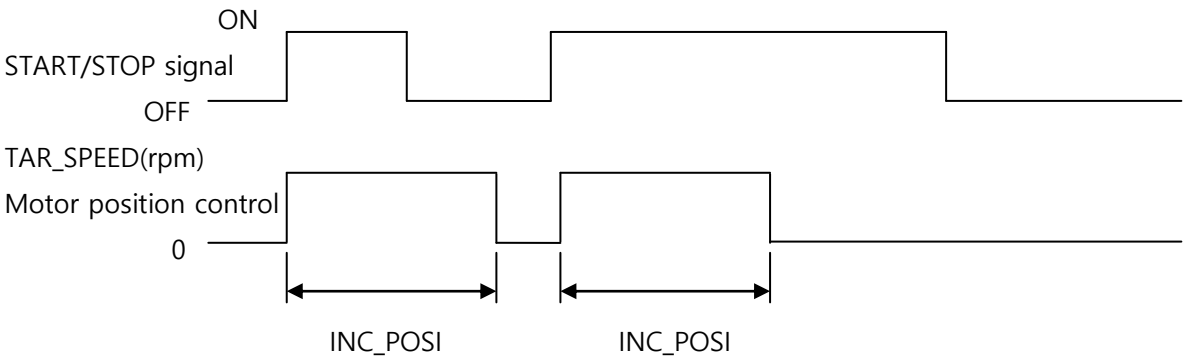


If the signal of START/STOP is ON then motor run continuously regardless of delay(DELAY)

- Function type, POSI(2)

Motor moves to the target position when the signal of START/STOP is ON from OFF

X-axis is motor position.



Here, the target speed(TAR_SPEED) and incremental position(INC_POSI) are set by PID_FUNC_POSI(250)

And the START/STOP signal is raplaced by PID_FUNC_CMD(184) when user want to use comm. only

About PID_PNTxxx command, user can handle two controllers concurrently, by using these kind of PID.

Example) If one has ID 1, and the other has ID 2 then, user send packet with ID 0xfe(Broadcasting ID), and at the ID1, write 1, and ID2 write 2 then the controller ID1, 2 will be response that PID_PNTxxx command.

2.4 PID HEADER

```
// General ID definition
#define ID_ALL                  0xfe
#define ID_WRITE_CHK            0xaa
#define ID_DEFALUT_CHK          0x55        // Default setting(write)
#define ID_DEVELOPER_CHK        0x77

//////////////////////////////////////////////////////
// Command : RMID, TMID, ID, PID, Data number, Data.., CHK
//////////////////////////////////////////////
// PID one-byte data : PID 0~127
#define PID_DEFAULT_SET         3
#define PID_REQ_PID_DATA        4
#define PID_TQ_OFF              5
#define PID_BRAKE               6
#define PID_ACK                 7

//////////////////////////////////////
#define PID_COMMAND             10

#define CMD_TQ_OFF              2
#define CMD_BRAKE               4
#define CMD_MAIN_DATA_BC_ON     5
#define CMD_MAIN_DATA_BC_OFF    6
#define CMD_ALARM_RESET         8
#define CMD_POSI_RESET          10
#define CMD_MONITOR_BC_ON       11
#define CMD_MONITOR_BC_OFF      12
#define CMD_IO_MONITOR_BC_ON    13
#define CMD_IO_MONITOR_BC_OFF   14
#define CMD_FAN_ON              15
#define CMD_FAN_OFF             16
#define CMD_CLUTCH_ON           17
#define CMD_CLUTCH_OFF          18
#define CMD_TAR_VEL_OFF         20
#define CMD_SLOW_START_OFF      21
#define CMD_SLOW_DOWN_OFF       22
#define CMD_CAN_RESEND_ON       23
```

```
#define CMD_CAN_RESEND_OFF          24
#define CMD_MAX_LOAD_OFF            25
#define CMD_ENC_PPR_OFF             26
#define CMD_LOW_SPEED_LIMIT_OFF     27
#define CMD_HIGH_SPEED_LIMIT_OFF    28


#define PID_ALARM_RESET             12
#define PID_POSI_RESET              13
#define PID_MAIN_BC_STATUS          14
#define PID_MONITOR_BC_STATUS       15
#define PID_INV_SIGN_CMD            16
#define PID_USE_LIMIT_SW            17
#define PID_INV_ALARM               18
#define PID_HALL_TYPE               19
#define PID_INPUT_TYPE              25
#define PID_PRESET_SAVE             30
#define PID_PRESET_RECALL           31

// PID two-byte data : PID 128 ~ 192
#define PID_VEL_CMD                 130
#define PID_VEL_CMD2                131
#define PID_ID                      133
#define PID_OPEN_VEL_CMD            134
#define PID_BAUD_RATE               135     // 9600, 19200, 38400, 57600 , 115200
#define PID_ECAN_BITRATE            137                 // 50K,100K,250K,500K,1M
#define PID_INT_RPM_DATA            138
#define PID_TQ_DATA                 139

#define PID_VOLT_IN                 143
#define PID_CCW_PHASE_OFFSET        146
#define PID_CW_PHASE_OFFSET         147

// 0 no return, 1:Monitor, 2:Ack return
#define PID_RETURN_TYPE             149
#define RETURN_TYPE_MONITOR          1
#define RETURN_TYPE_ACK              2
#define RETURN_TYPE_IO_MONITOR       3
```

```
#define PID_TQ_PO                   150
#define PID_OVER_MODULATION         152
#define PID_SLOW_START              153
#define PID_SLOW_DOWN               154
#define PID_TAR_VEL                 155
#define PID_ENC_PPR                 156
#define PID_LOW_SPEED_LIMIT         157
#define PID_HIGH_SPEED_LIMIT        158
#define PID_SLOW_START_DOWN         159
#define PID_DEAD_ZONE               162
#define PID_READ_ADDR               163
#define PID_REQ_PID_DATA2           164
```

// PID N-byte data : PID 193 ~ 240

```
#define PID_MAIN_DATA               193
#define PID_IO_MONITOR              194
#define PID_MONITOR                 196
#define PID_POSI_DATA               197
#define PID_RPM_DATA                198
#define PID_VEL_GAIN                202
#define PID_VEL_GAIN2               203
#define PID_TYPE                    205
#define PID_PNT_POSI_VEL_CMD            206
#define PID_PNT_VEL_CMD                 207
#define PID_PNT_OPEN_VEL_CMD            208
#define PID_PNT_TQ_CMD                  209
#define PID_PNT_MAIN_DATA               210
#define PID_MAX_LOAD                    211
#define PID_LIMIT_TQ                    212
#define PID_PNT_INC_POSI_CMD            215

#defien PID_POSI_SET                    217
#define PID_POSI_SET2                   218

#define PID_POSI_VEL_CMD            219
#define PID_INC_POSI_VEL_CMD        220 // Incremental posi. cmd.
#define PID_MAX_RPM                 221
#define PID_SPEED_LIMIT                 222
```

```
#define PID_MIN_RPM                223
#define PID_SPEED_LIMIT2           224
#define PID_STEP_INPUT             225     // No, input.
#define PID_CURVE_PT               226     // No. PtX(int), PtY(int)
#define PID_PRESET_DATA            227     // only position.

#define PID_POSI_MIN_LIMIT         231
#define PID_POSI_CEN               232
#define PID_POSI_MAX_LIMIT         233
#define PID_TIME                   234
#define PID_CAN_RESEND             238
#define PID_FUNC_SPEED             239
#define PID_PHASE_OFFSET           241
#define PID_POSI_CMD               243
#define PID_INC_POSI_CMD           244
#define PID_WRITE_ADDR             245
#define PID_PNT_POSI_CMD           246
#define PID_FUNC_POSI              250
```

2.5 Sample program.

```
// Test the packet using check-sum in InBuf
short IsChkSumOK(BYTE *byArray, short nPacketSize)
{
        short i;
        BYTE cbySum;

        cbySum = 0;
        for(i=0; i<nPacketSize; i++) {
            cbySum += *(byArray + i);
          }
        if(cbySum==0) return 1;
        else return 0;
}

// From the input array, return the chksum
BYTE GetCheckSum(short nPacketSize, BYTE *byArray)
{
        BYTE byTmp=0;
        short i;

        for(i=0; i<nPacketSize; i++) byTmp += *(byArray+i);
        return (~byTmp + 1);
}

short GetMainData(BYTE byData[])
{
        BLDC.nRPM = Byte2Int(byData[0], byData[1]);
        BLDC.wTq = Byte2Int(byData[2], byData[3]);
        BLDC.byType = byData[4];              //Control type
        BLDC.nRefVel = Byte2Int(byData[5], byData[6]);
        BLDC.nOut = Byte2Int(byData[7], byData[8]);
        BLDC.byStatus = byData[9];;
        BLDC.nPosi = Byte2LInt(byData[10], byData[11], byData[12], byData[13]);
        BLDC.byBrakeDuty = byData[14];
        BLDC.byTemp = byData[15];
        return 1;
}
```

```
// Make interger from two bytes
short Byte2Int(BYTE byLow, BYTE byHigh)
{
	return (byLow | (short)byHigh<<8);
}

// Make long type data from four bytes
int Byte2LInt(BYTE byData1, BYTE byData2, BYTE byData3, BYTE byData4)
{
	return((int)byData1 | (int)byData2<<8 | (int)byData3<<16 | (int)byData4<<24);
}

typedef struct {
	BYTE byLow;
	BYTE byHigh;
} IByte;

typedef struct {
	BYTE byData1;
	BYTE byData2;
	BYTE byData3;
	BYTE byData4;
} LByte;

// Get the low and high byte from interger
IByte Int2Byte(short nIn)
{
	IByte Ret;

	Ret.byLow = nIn & 0xff;
	Ret.byHigh = nIn>>8 & 0xff;
	return Ret;
}
```

3 History

VERSION	DATE	CONTENTS
V3.6	2013.05.09	PID number 10, 16, 137, 144, 150, 152, 153, 154, 155, 238, 241 added
V3.7	2013.05.16	PID number 17, 131, 149, 194, 238 added.
V3.8	2013.08.02	CMD_MAX_LOAD_OFF,25 CMD_ENC_PPR_OFF,26 CMD_LOW_SPEED_LIMIT_OFF,27 CMD_HIGH_SPEED_LIMIT_OFF, 28 added PID number 19, 21, 156, 157, 158, 203 added Write funn. Added of PID211, PID_MAX_LOAD
V3.9	2014.04.17	PID number 19, 21, 25, 30, 31, 159, 162, 163, 164, 203, 222, 225, 226, 227 added
V4.0	2016.01.14	MID defined. PID added.
V4.1	2016.09.20	PID added on torque(current) control
V4.2	2017.01.22	PID_PNT... added.(two motor controller or to drive two motor controller Concurrently(PNT50, MD750T, MD400T) PID_CW_PHASE_OFFSET, ..CCW_PHASE_OFFSET replaced PID_PHASE_OFFSET(241)
V4.3	2017.10.11	Special function related PID added(139, 250, 183, 184)
V4.4	2019.07.16	PID_PNT... related protocol changed(with return data type definition) RJ45 pin map is changed(our controller changed it's RJ45 signal type for the optimum noise filtering, so refer to the page 5 RS584- and TTL232TxD signal is changed each other.

- The end -