Communication protocol on CAN

NEXTEC

1. Introduction

Communication protocol(CAN) on the BLDC motor controller.

2. Contents

2.1 Packet structure(Extended mode only)

Header(29bits)				Data(8bytes)							
NC	RMID	TMID	ID	PID	D1	D2	D3	D4	D5	D6	D7
5bits	1byte	1byte	1byte	1byte	1byte	1byte	1byte	1byte	1byte	1byte	1byte

- NC(bit24~28) : Not used(Upper 5bits is not used).
- RMID(bit16~23) : Receiving Machine ID(183, BLDC motor controller).
- TMID(bit8~15) : Transmitting Machine ID(184, user controller).
- ID(bit0~7): Each controller ID(0~253, Broadcasting ID: 254).
- PID : Parameter IDentification number.
- The header is 183, 184 when the user sends data to the motor controller or 184, 183 when receiving data.

Data bytes on the PID

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

Examples

Packet configuration : 5bits(0), RMID, TMID, ID, PID, D1, ...,D7.

Default settings

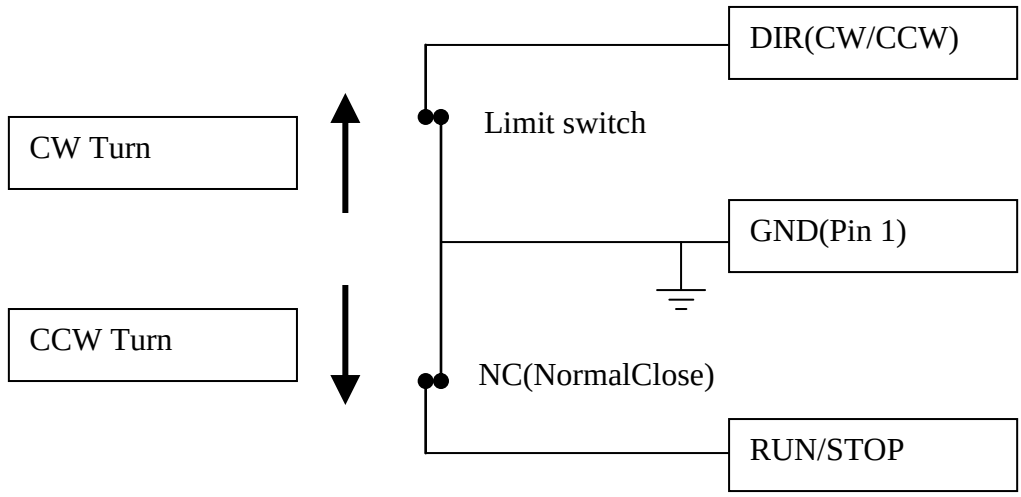
- When sending data bytes : Uses structure to send low byte of data first.
- Only when the request command is, the related information is responded.
- The response characteristic of the motor can be changed by using each corresponding parameter.
- When sending commands to all motors at the same time, use Broadcasting ID 254 (0xfe).
- If the request data in Broadcasting, system does not respond in order to avoid data collision.
- The header (RMID, TMID) are opposed when transported from the controller (184, 183).
- Default Baudrate (speed): **50kbps**

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 on the moving direction and the limit.

2.3 PID(Parameter IDentification Number)

-R : Read only(Request data by PID, PID_REQ_PID_DATA).

-W : Parameter change(Writing).

-C : Command.

-0xaa(170) : Write check byte(Additional data for security when writing).

-0xfe(254) : ID ALL(Used when sending commands to all controllers at the same time).

-0x55 : Default setting.

-0x77 : PID Gain writing(for developer).

-1 Byte data(PID: 0~127)

PID	Type	PID Name/Remark	Contents of data byte PID, D1, D2, D3, D4, D5, D6, D7	Variable type Default value
3	C	PID_DEFAULT_SET. Default setting	Data : 0x55(CHECK) Set variable as initial value at default setting. 3, 0x55, x, x, x, x, x, x	BYTE 0x55
4	**C**	**PID_REQ_PID_DATA Data request**	**0~253: Requested PID number 4, R_PID, x, x, x, x, x, x**	**BYTE**
5	C	PID_TQ_OFF Stop naturally	TQ_OFF Stops the motor naturally 5, x, x, x, x, x, x, x	BYTE
6	C	PID_BRAKE Erectric brake	Regardless of the data, the motor is electrically stopped quickly 6, x, x, x, x, x, x, x	BYTE
10	C	PID_COMMAND 1byte command bar	Contents on CMD number. 5 : PID_MAIN_DATA broadcasting ON 6 : PID_MAIN_DATA broadcasting OFF 8 : Reset alarm 10 : Position reset(set position to zero) 11 : PID_MONITOR broadcasting ON 12 : PID_MOINTOR broadcasting OFF 13 : PID_IO_MINITIOR broadcasting ON 14 : PID_IO_MONITOR broadcasting OFF 15 : Fan ON(motor cooling fan) 16 : Fan OFF 17 : Mechanical brake(clutch) ON 18 : Mechanical brake OFF 20 : Erase target speed, 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 10, CMD, x, x, x, x, x, x	

<table>
<tr><th>PID</th><th>Type</th><th>PID Name/Remark</th><th>Contents of data bytes
PID, D1, D2, D3, D4, D5, D6, D7</th><th>Variable type
Default value</th></tr>
<tr><td>12</td><td>C</td><td>PID_ALARM_RESET
Reset alarm</td><td>Release the alarm state of the controller
12, x, x, x, x, x, x, x</td><td>BYTE</td></tr>
<tr><td>13</td><td>C</td><td>PID_POSI_RESET
Reset position, Position->0</td><td>Reset motor position to 0
13, x, x, x, x, x, x, x</td><td>BYTE</td></tr>
<tr><td>14</td><td>C</td><td>PID_MAIN_BC_STATE
Request broadcasting of PID_MAIN_DATA</td><td>DATA
1 : PID 193 broadcasting on
0 : Broasdcasting off
14, DATA, x, x, x, x, x, x</td><td>BYTE
0</td></tr>
<tr><td>15</td><td>C</td><td>PID_MONITOR_BC_STATE
Request broadcasting of PID_MONITOR</td><td>DATA
1 : PID 196 broadcasting on
0 : Broasdcasting off
15, DATA, x, x, x, x, x, x</td><td>BYTE
0</td></tr>
<tr><td>16</td><td>R/W</td><td>PID_INV_SIGN_CMD
Inverse of moving direction.</td><td>DATA
PC data(1000)->Controller gets(-1000).
1 : Inverse of reference direction
0 : Don’t use inverse sign(normal command).
16, DATA, x, x, x, x, x, x.</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 PIN, 6,8(DIR, START/STOP) is used as limit switchs
0 : Cancel limit switch function
17, DATA, x, x, x, x, x, x</td><td>BYTE
1</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(ALARM ON->HIGH LEVEL)
19, DATA, x, x, x, x, x, x</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>
and over the no. 5 is input then, the pole is following, poles = DATA * 2
21, DATA, x, x, x, x, x, x</td><td>BYTE
0</td></tr>
</table>

- 2 Bytes data(PID: 128~192)

PID	Type	PID Name/설명	Contents of data byte PID, D1, D2, D3, D4, D5, D6, D7	Variable type Default value
130	**C**	**PID_VEL_CMD** **Speed command** **(Unit : rpm)**	**Speed command.** **Speed(rpm) = (D1 | D2<<8)** **Speed>0, CCW direction** **Speed<0, CW direction** **130, D1, D2, x, x, x, x, x**	**INT**
133	W	PID_ID ID setting	Write command(0xaa) ID : 1~253 : Setting ID 133, 0xaa, ID, x, x, x, x, x	BYTE, BYTE ID_ALL(0xfe)
134	**C**	**PID_OPEN_VEL_CMD** **Open-loop control**	**D1, 2: Open-Loop speed** **Range : -1023~1023(D1 | D2<<8)** **134, D1, D2, x, x, x, x, x**	**INT**
135	W	PID_BAUDRATE Baudrate setting(RS485)	Set the baudrate, BAUD 1 : 9600bps, 2 : 19200bps 3 : 38400bps, 4 : 57600bps 5 : 115200bps 135, 0xaa, BAUD, x, x, x, x, x	BYTE 1 or 2
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 137, 0xaa, BIT_RATE, x, x, x, x, x	BYTE 1
138	R	PID_INT_RPM_DATA Motor speed (16bits data)	Motor speed: DATA1, DATA2 RPM = (D1 | D2<<8) 138, D1, D2, x, x, x, x, x	INT
139	R	PID_TQ_DATA Current(0.1A)	Return current(unit 0.1A) 10->1A, 15->1.5A Current = (D1 | D2<<8)/10 139, D1, D2, x, x, x, x, x	INT
143	R	PID_VOLT_IN Supply voltage	Return supply voltage. 0.1V unit, 12->1.2V 143, D1, D2, x, x, x, x, x	INT

149	R/W	PID_RETURN_TYPE Set return data type When command received, Controller send PID data set by RET_TYPE	Determine the type of data that is returned when the controller receives commands but PID_REQ_PID_DATA RET_TYPE 0 : No return 1 : PID_MONITOR 2 : PID_ACK(return received PID) 3 : PID_IO_MONITOR 149, 170, RET_TYPE, x, x, x, x, x	BYTE 0
153	R/W	PID_SLOW_START Set the Slow/Start (not use internal volume)	Value of Slow/Start(0~1023->0~15s delay) Applied when the ref. speed is increased SS = (D1 | D2<<8) 153, D1, D2, x, x, x, x, x	BYTE 0
154	R/W	PID_SLOW_DOWN Set the Slow/Down variable	Value of Slow/Down(0~1023->0~15s). Applied when the ref. speed is decreased SD = (D1 | D2<<8) 154, D1, D2, x, x, x, x, x	BYTE 0
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~Max. RPM(about 5000rpm) 155, D1, D2, x, x, x, x, x	BYTE
156	R/W	PID_ENC_PPR Set encoder pulse per revolution(PPR)	When the encoder pulse is set, the controller uses encoder signals as a speed sensing input ENC_PULSE = (D1 | D2<<8) 156, D1, D2, x, x, x, x, x	BYTE 0
157	R/W	PID_LOW_SPEED_LIMIT Set the min. 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(0), this input is treated as a zero LOW_LIMIT = (D1 | D2<<8) 157, D1, D2, x, x, x, x, x	BYTE 0
158	R/W	PID_HIGH_SPEED_LIMIT Set the max. higher value of analog input	Setting range is 512~1023 Higher than this HIGH_LIMIT(1023) is the same to HIGH_LIMIT, HIGH_LIMIT = (D1 | D2<<8) 158, D1, D2, x, x, x, x, x	BYTE 0

- N Bytes data(PID: 193~240)

PID	Type	PID Name/Remark	Contents of data byte PID, D1, D2, D3, D4, D5, D6, D7	Variable type Default value
193	R	PID_MAIN_DATA Main data	Data : 7Bytes D1 : Status of controller(bit), D2,3 : Speed(rpm) D4,5 : Current (0~1023, 0.1A unit) D6,7: Control output(0~1023) 193, D1, D2, D3, D4, D5, D6, D7	BIT INT INT INT
194	R	PID_IO_MONITOR Monitor for in/out data of controller	Data : 7Bytes D1,2 : Speed(unit: 10rpm) Speed(rpm) = (D1 | (D2<<8))x10 D3,4 : Current(0~1023, 0.1A unit) D5 : Status of controller D6 : Input signal of controller. BIT0 : INT_SPEED BIT1 : ALARM_RESET BIT2 : DIR(CW/CCW) BIT3 : RUN/BRAKE BIT4 : START/STOP BIT5 : ENC_B BIT6 : ENC_A D7 : Internal speed volume(0~255) 194, D1, ..., D7	Mainly used when driving at more than 30,000 rpm
196	**R**	**PID_MONITOR** **Monitor data**	**Data : 7 Bytes** **D1 : Status of controller(bit),** **(Refer to status BIT data)** **D2,3 : Speed(rpm)** **D4,5,6,7 : Motor position(long type)** **196, D1, D2, D3, D4, D5, D6, D7**	**BIT** **INT** **LONG**
197	R	PID_POSI_DATA Motor position	Data : 4 Bytes D1,2,3,4 : Motor position(Home counting unit, If encoder is used, encoder counting number) POSI = (D1 | D2<<8 | D3<<16 | D4<<32) 197, D1, D2, D3, D4, x, x, x	LONG
198	R	PID_RPM_DATA Motor speed(rpm)	Returen Motor speed(rpm unit) RPM = (D1 | D2<<8) 198, D1, D2, x, x, x, x, x	INT

PID	Type	PID Name/Remark	Contents of data byte PID, D1, D2, D3, D4, D5, D6, D7	Variable type Default value
203	R/W	PID_VEL_GAIN Applying a gain value more than 255	Data : 6 Bytes D1,2 : : Proportional gain of positon control(P) D3,4 : Proportional gain of speed control(P) D5,6 : Integral gain of speed control(I) 213, D1, D2, D3, D4, D5, D6, x	INT
205	R	PID_TYPE Type of controller	Data : Within 7Bytes(Sent as a Character value) 205, D1, D2, D3, D4, D5, D6, D7	BYTE
211	R	PID_MAX_LOAD Set maximum current value	Data : 2Bytes D1,2: Set maximum current value of motor(Torque) 211, D1, D2, x, x, x, x, x	INT
217	C	PID_POSI_SET Setting position (Set user position)	Data : 4Bytes D1,2,3,4 : (Long type position data) Change the motor position value of the controller 217, D1, D2, D3, D4, x, x, x	LONG

Status BIT

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 byte** **PID, D1, D2, D3, D4, D5, D6, D7**	**Variable type** **Default value**
219	**C**	**PID_POSI_VEL_CMD** **Position control with max. target speed.**	**Data : 6Bytes.** **D1,2,3,4 : Ref. Position** **D5,6 : Max. control speed(rpm)** **If the speed is zero, then use half of max.** **219, D1, .., D6, x**	**LONG** **INT**
221	R	PID_MAX_RPM Set max. speed(rpm)	Data : 2Bytes D1,2 : Max. Speed(rpm) 221, D1, D2, x, x, x, x, x	
234	R	PID_TIME Motor driving time.	Data : 4Bytes D1,2,3,4 : Operating time. Transfer to the cumulative motor rotation time. 234, D1, D2, D3, D4, x, x, x.	LONG

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_MONITOR2_BC_ON      13
#define CMD_MONITOR2_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

// 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_MONITOR2        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

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

#define PID_MAIN_DATA 193
#define PID_MONITOR2 194
#define PID_MONITOR 196
#define PID_POSI_DATA 197
#define PID_VEL_GAIN 203
#define PID_TYPE 205
#define PID_MAX_LOAD 211
#defien PID_POSI_SET 217
#define PID_POSI_VEL_CMD 219
#define PID_MAX_RPM 221
#define PID_VEL_GAIN_W 231
#define PID_TIME 234
#define PID_CAN_RESEND 238
#define PID_PHASE_OFFSET 241

2.5 Sample program

```
// 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;
}

// Get the data bytes from long type data
LByte LInt2Byte(int nIn)
{
```

```
	LByte Ret;

	Ret.byData[0] = nIn & 0xff;
	Ret.byData[1] = nIn>>8 & 0xff;
	Ret.byData[2] = nIn>>16 & 0xff;
	Ret.byData[3] = nIn>>24 & 0xff;
	return Ret;
}

int EPutNByteData(BYTE byRMID, BYTE byID, BYTE byPID, BYTE byDataNum, BYTE byArray[])
{
		DWORD dwID;
		BYTE i, byData[10];

		for(i=0; i<8; i++) byData[i] = 0;
		dwID = byID;
		dwID |= (DWORD)MY_MID<<8;
		dwID |= (DWORD)byRMID<<16;
		byData[0] = byPID;

		if(byDataNum>7) byDataNum = 7;
		for(i=0; i<byDataNum; i++) byData[i+1] = byArray[i];

		WriteECanMsg(dwID, 8,  byData);
		return 1;
}
```

```
BYTE SetState(void)
{
        BYTE byData;

        byData = Ctrl.fgAlarm & 0x01;
        byData |= ((Ctrl.fgCtrlFail & 0x01)<<1);
        byData |= ((Ctrl.fgOverVolt || Ctrl.fgUnderVolt)<<2);
        byData |= ((Ctrl.fgOverTemp & 0x01)<<3);
        byData |= ((Ctrl.fgOverLoad || Ctrl.fgShort)<<4);
        byData |= ((Ctrl.fgHallFail & 0x01)<<5);
        byData |= ((Ctrl.fgInvVel & 0x01)<<6);
        byData |= ((Ctrl.fgZeroVelLoad & 0x01)<<7);
        return byData;
}

short GetMonitor(BYTE byData[])
{
          // Status
        BLDC.byAlarm = (byData[0] & 0x01);
        BLDC.byCtrlFail = (byData[0]>>1) & 0x01;
        BLDC.byOverVolt = (byData[0]>>2) & 0x01;
        BLDC.byOverTemp = (byData[0]>>3) & 0x01;
        BLDC.byOverLoad = (byData[0]>>4) & 0x01;
        BLDC.byHallFail = (byData[0]>>5) & 0x01;
        BLDC.byInvVel = (byData[0]>>6) & 0x01;
        BLDC.byStall = (byData[0]>>7) & 0x01;

        BLDC.nRPM = Byte2Int(byData[1], byData[2]);
        BLDC.lPosi = Byte2Long(byData[3], byData[4], byData[5], byData[6]);
          return 1;
}
```

3 History

VERSION	DATE	CONTENTS
V1.0	2013.05.10	Draw up the first specification for CAN communication.
V1.1	2013.05.17	PID17 : Add PID_USE_LIMIT_SW. Add I/O relationship of CTRL connector when used as communication specification. Fix PID_CAN_RESEND structure of PID238 (delete data number). PID 131, PID_VEL_CMD2. PID 194, PID_MONITOR2.

V1.2	2013.08.02	CMD_MAX_LOAD_OFF,25.
		CMD_ENC_PPR_OFF,26.
		CMD_LOW_SPEED_LIMIT_OFF,27.
		Add CMD_HIGH_SPEED_LIMIT_OFF,28.
		PID19, PID_INV_ALARM.
		PID21, PID_HALL_TYPE.
		PID156, PID_ENC_PPR.
		PID157, PID_LOW_SPEED_LIMIT.
		PID158, PID_HIGH_SPEED_LIMIT.
		Add PID203, PID_VEL_GAIN2.
		Write function of PID211, PID_MAX_LOAD.

V1.2a	2017.07.07	PID 139, revise current calculation formula PID 149, Change MONITOR2 to IO_MONITOR PID 153, Change the range of slow start value from 0 to 1000 to 0 to 1023 PID 154, Change the range of slow start value from 0 to 1000 to 0 to 1023 PID 194, Change MONITOR2 to IO_MONITOR PID 202, VEL_GAIN deleted and as PID203 replaced. Change the reference temperature value of OVER_TEMP to 65 degrees among status BITs

- The end -