Specification of Dyn2-series AC servo system

1. Introduction

This the complete document for the specification of Dyn2-series AC servo system.

2. Drive interface Dyn2-series

There are three connectors in C-Grid (molex),and two terminal blocks for Dyn2 Drive board.

JP1: Power supply of Drive(terminal block)for Dyn2,up to 900 Watts
Pin 1: Operation range +24~+48(Vdc), max 20(A),+60(Vdc).
Pin 2: Ground for +24~48(Vdc).

for Dyn2s, up to 150 watts
Pin 1: Operation range +24~+48(Vdc), max 6(A),+60(Vdc).
Pin 2: Ground for +24~+48(Vdc).

JP2:RS232 Port
Pin 1: Ground for +5(V)
Pin 2~4: NC
Pin 5: RS232 signal input, RxD, TTL/CMOS level.
Pin 6: RS232 signal output, TxD, TTL/CMOS level.
Pin 7: +5(V) output, <10(mA), generated in board.

In order to connect JP2 with PC's 232 port, an intermediate cable with RS232 level shift buffer is necessary.

JP3: position command port
Pin 1: Gnd
Pin 2: Motor free signal (low active) Pulse/Dir, CW/CCW mode, Analog Input 0~5(Vdc) for analog input mode.
Pin 3: CCW, Dir, serial data in(SI) for SPI mode(Optional).
Pin 4: CW, Pulse, serial clock input for SPI mode(Optional).
Pin 5: On position signal (low active) for CW/CCW mode, serial data output for SPI mode(Optional).

JP4: Sensor port
Pin 1: Gnd
Pin 2: Position information in PWM form.
Pin 3: Clock signal to sensor.
Pin 4: Serial data to sensor.
Pin 5: Serial data from sensor.
Pin 6: +5(V) to sensor.

JP5: Motor power line(terminal block)

Pin 1: A phase ,Red for DML motor ,RED pittman
Pin 2: B phase ,Blue for DML motor ,Orange pittman
Pin 3: C phase ,Yelow for DML motor,Brown pittman

!!!Note: Because of the motor's working state, the voltage presented in the terminals JP1 could be over +60Vdc, the re-generated surge protect capacitors (>4700uF) or external re-generation protect circuit is necessary to ensure the voltage be less than 60Vdc.
Over 60Vdc in JP1 could cause the Drive permanent damage.

IMPORTANT WARNNING!!!
Improper use or operation for the servo system could cause the danger, health hazards, injury even death, DMM technology corp. can not be held responsibility for the improper use or operation.

3.Using DmmDrv.exe, GUI interface tune up servo

DmmDrv provides a good GUI interface between PC and Drive by the connection through RS232 cable or USB cable.

No matter what position command input mode is, the RS232 port is all the time active for setting the servo constants, and drive configuration.

It is very easy and convenient to make motor have some movement such as step, sinusoidal, speed, while tuning up the servo.

By selecting the servo setting item after dmmdrv is launched, the servo setting dialog box will be displayed.

The Drive configuration and servo cons are stored in the EEPROM of Drive if the save button is pushed or parameters setting is issued through the serial communication.

WARNING!: Never use serial communication to set the Drive cons or configuration all the time in a rate of certain time period, that will cause EEPROM busy in writing all the time and short life, the guaranteed writing cycle for EEPROM is 1 million times.
Once a group of cons and configuration are set, try to use it until next necessary parameter change.

Since MaxAcl(Max acceleration) and MaxSpd(Max speed) for point to point movement through RS232 input mode are not stored in EEPROM, MaxAcl and MaxSpd can be changed without time limits.

3.1 Read servo parameters out from Drive before revising them

The Drive will use the default servo parameters or constants for the first time power on reset(POR), that default parameters will ensure the servo is at best performance for no inertia load and stable for some how small inertia load.

It is highly recommended that read all parameters out from Drive by using the Read button first, then revise them and save them back to Drive by click Save button. That is the best step to go for the first time user.

WARNING! : if the RS485/RS232 net is used to connect more than 1 Drive in a serial net, RS485/232 check box MUST be checked, other wise one Drive's servo cons will be saved to the other.

If there is only one Drive connected to the RS485/232 port,and RS485/232 be not checked, the Drive ID number can be assigned to this only one Drive, also you can read out any Drive's all servo cons including its ID number even you do not know what ID number that Drive has before the reading.

After every Drives all have their unique Drive ID which is started such as from 0 to 3, then those Drives can be connected together by a RS484/RS232 net.

3.2 tune up servo

As the inertia load is varying, the servo performance will vary as well for example there will be overshot for step response if the heavy load are mounted(even unstable if the load is too heavier).

There are four parameter for tune up servo, MainGain, SpeedGain, IntGain and TrqCons, all them takes value from 1~127.

MainGain:(1~127)

The main gain for the servo loop, usually be increased as the load increasing.

The bigger value of ManGain means wider frequency range of servo loop relatively.

SpeedGain:(1~127)

The speed gain for the servo loop, usually be increased as the load increasing.

The bigger value of speedGain means more narrow frequency range of servo loop relatively.

Physically, the heavier load should have lower dynamic ability, so the servo loop frequency range should be more narrow by using bigger value of speedGain.

If the speedGain is somehow big enough for big load, there will be some loud noise because the torque command is too coarse not smooth, the smaller Trqcons (see TrqCons) could be used to attenuate that noise.

IntGain:(1~127)

There is integrator in the servo loop to ensure the error between position command and real position be zero during the steady state, also that integrator will let servo have more ability to attenuate the outside disturbance torque.

The bigger value of IntGain, the more ability of the servo to attenuate the outside disturbance torque.

TrqCons :(1~127)

TrqCons is a first order filter constant, the bigger value menas wider frequncey range of that filter.

That filter can be expressed as : a / (S + a), here a = 26TrqCons, if TrqCons = 100, then a = 2600.

That filter is used to make the torque sent to torque loop more smooth especially for the heavier load when bigger SpeedGain is used.

If a very quick response servo with small load is desirable, the bigger value even the value 127 should be used to ensure the stability and dynamic performance.

3.3 Drive ID

Every Drive have an unique ID number, which can be assigned or read out by using ServoSetting dialog box **WHEN RS485NET BOX NOT CHECKED** and there is **only one Drive** connected to RS232 port.

The default ID number for every Drive is 0.

That ID number can be used for the connection by using the network of RS485.

Set mouse curser inside the ID edit box, input the ID number, the click the save button, ID number will be sent to the Drive with all other parameters. ID = 0 ~ 126.

WHEN RS485NET BOX is CHECKED and there are **more than one Drive** connected to RS485/232net,Only for the Drive with indicated ID number in the ServoSetting dialog box, its servo cons can be read out or saved.

3.4 On Position range

On position range is a value used for determining whether the motor have reached the commanded position or not. That on position range is selectable according to customer's requirement.

Suppose the Pset is the commanded position, and Pmotor is the real motor position, if |Pset - Pmotor|<=OnRange, it is said motor is on the commanded position, otherwise not. That OnRange is from 1 ~127. The real position on range is : OnRange * 360(deg)/16384.

Set mouse curser into the On Position edit box, input the on position value, then click the save button,On position value will be sent to the Drive with all other parameters.

3.5 Start form absolute zero or start from right now position

Because the 360(deg) range absolute sensor is used for the servo feedback control, it is possible for the Drive to start from right now position like any other increment encoder does, or start form the absolute zero even start from any customer assigned start position which is remembered in the sensor position 2 of bank (see. absolute sensor spec).

The default mode for servo start is from right now position like any other Drive does in which the incremental encoder is engaged in the servo control.

If the start from absolute zero check box is checked, the servo will shift little bit to start from absolute zero or from the saved position 2 after the Save button is clicked, that means a valid customer servo start point have been selected.

The start from absolute zero is used if there is no valid position is saved in sensor position 2 of bank.

The return to the absolute zero or specified position takes way of the most close routine, i.e. always takes the routine of less than 180(deg).

How to evaluate the servo is started form absolute zero?
a. Start servo by put +24V on.
b. Launch the GUI interface DmmDrv
c. Read all of parameters out from Drive by click read button
d. do not let Drive (or motor) have any type movement.
e. Check the start from absolute zero box
f. Click the save button, then motor will return to absolute zero every time no matter where the initial position is.

The specified servo start position 2 can be saved by using the dialog box of Compass under the item of AbsSensor in main menu. More details are described in the spec_Sensor.doc.

3.6 Select command input mode

There are four type of command input mode to select, RS232, Pulse/Dir, CW/CCW, analog and SPI(Optional).

Even Pulse/Dir CW/CCW,or analog is selected for position setting input, RS232 port is still active for setting servo constants and Drive configuration.

The selected mode will not be valid until the save button is clicked.

3.6.1 RS232 serial input mode

If RS232 input mode is selected, position, speed , torque ,even circular/linear interpolation command can be sent through the RS232 port according to the RS232 communication protocol described in section 4.

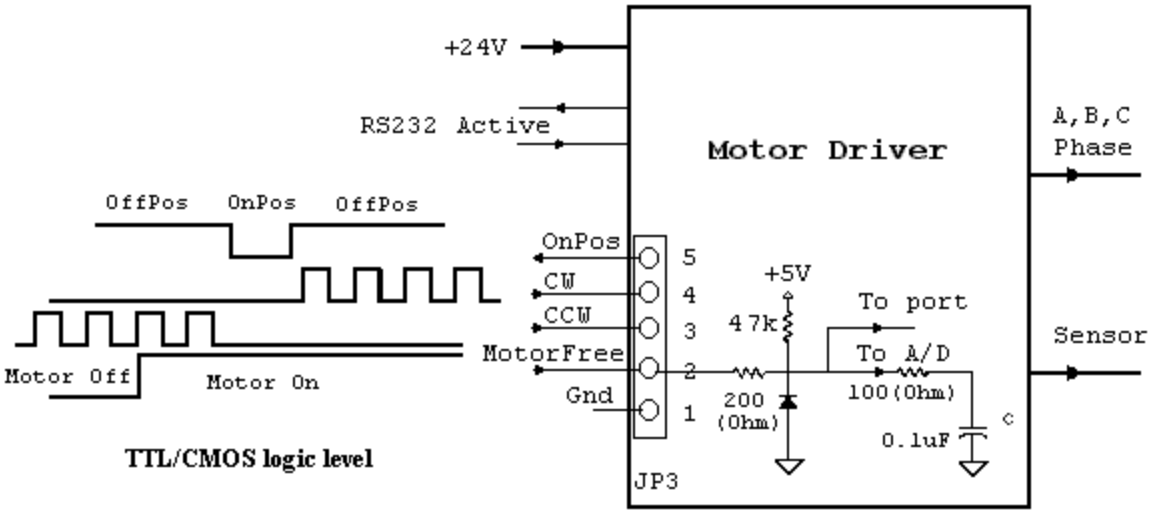
3.6.2 CW / CCW input mode

For CW/CCW input mode, the Pin2 and Pin4 of JP3 are used for sending CW and CCW pulse to Drive.

Also if |Pset - Pmotor|<=OnRange, Pin 5 = low, on position, otherwise Pin 5 = high, off position.

If Pin2 = low, motor will free, otherwise will be servo active even open.

For CW/CCW input Mode



For CW/CCW input, the up-edge of pulse is active for counting, also for CW input, motor will turn in the direction of CW from the view of motor mount side.

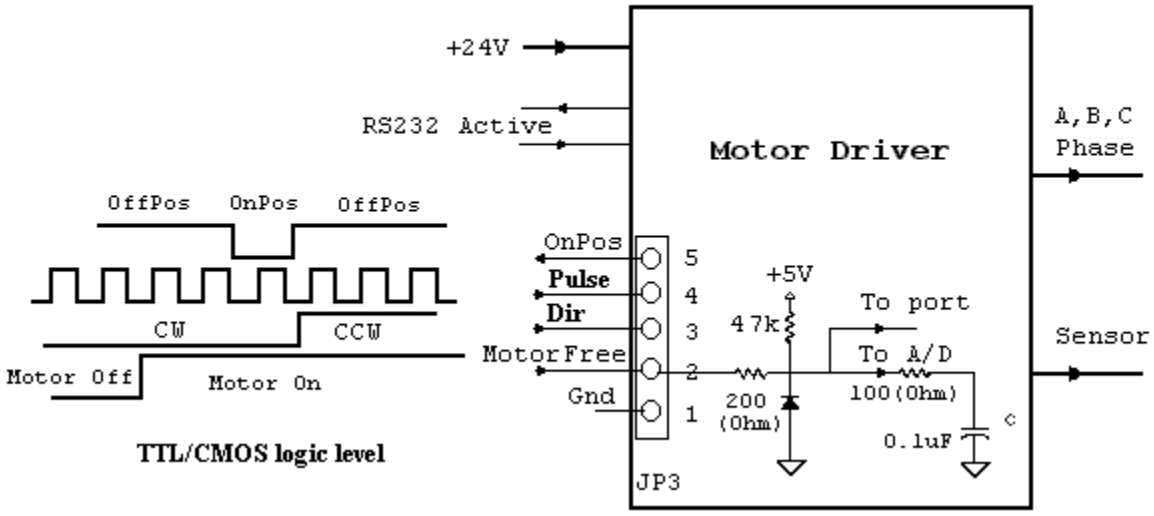
3.6.3 Pulse/Dir input mode

For Pulse/Dir input mode, the Pin2 and Pin4 of JP3 are used for sending direction and pulse to Drive.

Also if |Pset - Pmotor|<=OnRange, Pin 5 = low, on position, otherwise Pin 5 = high, off position.

If Pin2 = low, motor will free, otherwise will be servo active even open.

For Pulse/Dir input Mode



For CW/CCW input, the up-edge of pulse is active for counting, also for CW input, motor will turn in the direction of CW from the view of motor mount side.

3.6.4 SPI input mode(Optional)

The pin4 of JP3 used as clock input(into Drive) for SPI, the pin3 of JP3 is used for data input, and the pin5 of JP3 as data output.

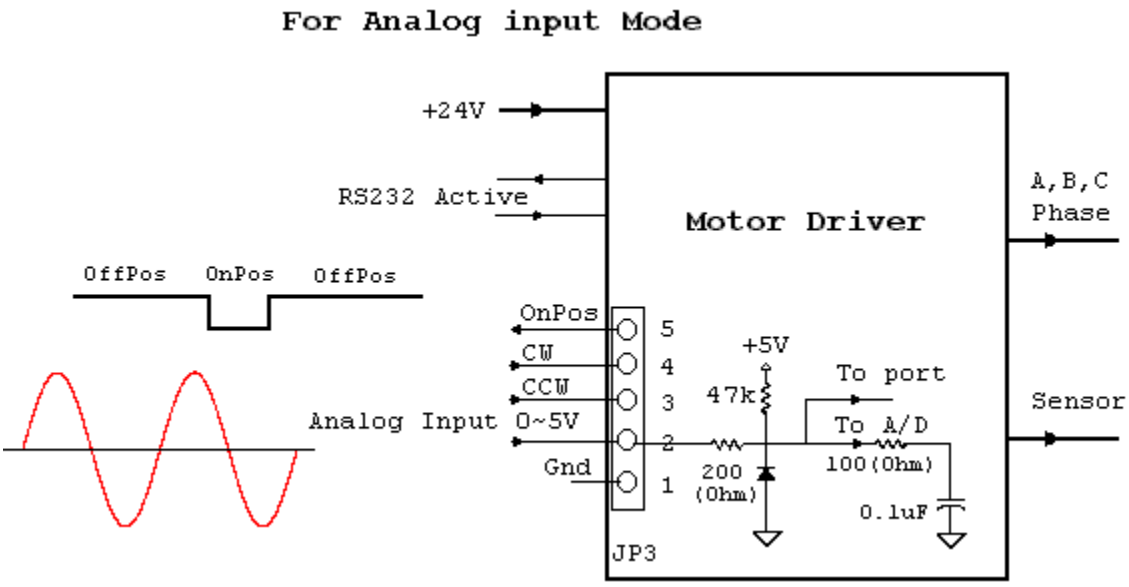
The every input byte, suppose it be SPI_in8, must be between -127 ~ +127, i.e. SPI_in8 = 0x81~0x7f.

The motor position will increase SPI_in8*4096/Gear_Number counts for every received byte.

The output byte is the content of Drive status register. This mode can be designed for special customer requirement.

3.6.5 Analog input mode

For Analog input mode, Pin 2 and Pin1(Gnd) are used.



For 5V analog input, motor will turn 90*4096 / Gear_Num(deg) in the CW direction in position servo mode if the pin4 of JP3 is high or open, otherwise will turn in CCW direction if the pin4 of JP3 is low.

Analog input is also used for speed / torque servo loop, it is possible to use a potentiometer between Pin2 and Pin1 of JP3 to control the motor position, speed, torque.

If the pin4 of JP3 is high or open, motor will turn in CW direction, otherwise will turn in CCW direction.

3.7 Servo mode

There are three servo modes for Dyn series servo system, position servo, speed servo, torque servo.

Postion servo mode

The position servo mode as described above, the motor position is feedbacked to the Drive for making position servo control, the motor position will follow the input position command.

Speed servo mode

The motor speed is feedacked into dirver to achieve a speed servo control loop. The Maingain * SpeedGain is used for the loop control gain.

The input mode can be RS232 or analog, even an external potentionmeter.

Torque servo mode

The motor will be controlled to generate a constant torque according to the torque command

The input mode can be RS232 or analog, even an external potentionmeter.

Note: There are two ways to turn the motor at a set speed, one is giving ramp position command to the servo in position servo mode, the other is giving a speed command in speed servo mode.

3.8 Test movement of motor

If the RS232 command input mode is selected, it is possible to have motor make some test movement such as small movement around zero, step, sinusoidal, constant speed.

It is possible and a very effective way to let motor have a type of movement while tune up the servo by revising/saving parameters.

3.9 Built in special motion

There are three types of special built in motion for Dyn series AC servo system, those motions can be easily used in RS232 mode to let CNC made up by Dyn seires cut a circle, circular arc, even oval shape.

Point to point movement

For the point to point movement through the RS232 command mode, a set machine instruction could be edited and be run in the GUI interface. Because the distance between the point and point could be very far away,the S-curve function embedded inside Drive microcontroller is automatically used.

Max acceleration and Max Speed are two important data for generating the DMM Tech S-Curve,also the acceleration in that S-curve is smoothed by DMM Tech filtering method.

i = interger part of (MaxSpd+3)*(MaxSpd+3)/16
Maximum Speed = i*3333.3*GearRatio(pps), Pulse Per Second
or Maximum Speed = i*12.21*GearRatio(rpm), 16384 pulse is one turn for senor. Here GearRatio = 4096 / Gear_Num.

Because MAxSpd is from 1 to 127, so Maximum Speed from 3333.3 to 3519965(pps), 12.21(rpm) to 12897(rpm) if GearRatio = 1.

Maximum Acceleration = MaxAcl*173611.1*GearRatio (ppss)
Or Maximum acceleration = MaxAcl*635.78*GearRatio (rpm/s)

If MaxAcl = 1, it will takes about 4seconds to speed up to 2500(rpm) if GearRatio = 1.

MaxSpd and MaxAcl are selectable through the GUI of DmmDrv.

Then for the point to point movement, the load will be accelerated with set maximum acceleration to the maximum speed and keep that speed for a while maybe then slowed down with set maximum acceleration to speed zero, everything is made smoothly.

Let Amax = Maximum Acceleration(ppss)
Vmax = Maximum Speed(pps)
S = Displacement(p)

Then the total time to finish the displacement S as:

Ttotal = V/Amax + S/V (Second)

Here V = minimum(sqrt(Amax*S) , Vmax)
that means V is the smallest value among sqrt(Amax*S) and Vmax.

Example: GearRatio = 1 or Gear_Num = 4096.
Suppose MaxSpd = 68,MaxAcl = 12 are set
Amax = 16666666(ppss)
Vmax = 1050000(pps), or 3845(RPM)

S = 884736(p), or motor shaft move 54(turn)

sqrt(Amax*S) = 3839999.9 > Vmax
So V = Vmax = 1050000(pps)

Ttotal = V/Amax + S/V = 1050000/16666666 + 884736/1050000
= 0.9056(Second) or 905.6(ms).

Host controller(PC or microcontroller or DSP) use absolute or relative position setting function send a destined position or a displacement from current motor position, the value of it will be from -134217728 to 134217727, then the Drive will start the S-curve motion itself, and set the bit5 of Drive status register be 1, means Drive is busy on S-curve now.

After the S-curve be finished, bit5 of Drive status register will be set be zero, means wait for next motion command.

By reading the Drive status register to detect its bit5 to make sure whether the cycle of S-curve is finished or not.

If a S-curve cycle has not be finished yet, and a new position command comes in, the current S-curve cycle will be abortioned and a new S-curve cycle be started from current motor position. In this case, if the position command uses the absolute function, then the motor will finally stopped at the assigned absolute position after the S-curve be finished. Other wise if the relative function is used, the motor will stopped

finally at the uncertain position for the Host controller.

X Y Z three axes coordinated linear motion

X axis must be Drive 0.
Y axis must be Drive 1.
Z axis must be Drive 2.

The linear coordinated motion can be made only by Drive 0 ,Drive 1 and Drive 2.

By using make linear line function and ID number 127(0x7f), send three data by using three packets. Suppose the motor current positions are (X0,Y0,Z0), the destination is (X1,Y1,Z1)

The data in first packet : X1 - X0
The data in second packet : Y1 - Y0
The data in third packet : Z1 - Z0
The data in the fourth packet: FeedRate,takes value 1~127

The speed for motor is about 1.526*FeedRate(Rpm) if the Gear_Number is 4096.

Four packets are necessary, even like Z1 -Z0 = 0, data 0 must be sent.

After dirver0,1,2 received above three data, they will make linear coordinated motion from (X0,Y0,Z0) to (X1,Y1,Z1).

During the linear motion, bit5 = 1, and then bit5 = 0 after the linear motion. Bit5 is the fifth bit of Drive status register.

By sending a relative or absolute position commands to Drive, the linear coordinated motion will be stopped.

X Y Z coordinated circular motion

X axis must be Drive 0.
Y axis must be Drive 1.
Z axis must be Drive 2.

The circular coordinated motion can be made only by Drive 0 and Drive 1.

By using make circular arc function and ID number 127(0x7f), send three data by using five packets. Suppose the motor current positions are (X0,Y0), the destination is (X1,Y1) also the circular center be (Xc,Yc).

The data in first packet : X0 - Xc
The data in decond packet : Y0 - Yc
The data in third packet : X1 - Xc
The data in fourth packet : Y1 - Yc
The data in fifth packet : Two Bytes (16bits)

PlaneNumber = 0,1,2. the 0 for X-Y plane,1 for Z-X and 2 for Y-Z, is expressed by one byte.

FeedRate = -127~127,except of 0, it takes one byte. If FeedRate is positive, it will make CW circular arc, otherwise make CCW circular arc.

Suppose PlaneNumber and FeedRate both are 8bits data(char).

TwoBytes = (PlaneNumber<<8) | FeedRate, the high byte for Plane number and the low byte for FeedRate.

The circle or arc could be made in X-Y,Z-X,or Y-Z plane, depends on the PlaneNumber.

After dirver0,1,2 received above five data, they will make circular coordinated motion from (X0,Y0) to (X1,Y1). if FeedRate >0, the clockwise direction, otherwise count clockwise direction.

During the circular motion, bit5 = 1, and then bit5 = 0 after the circular motion. Bit5 is the fifth bit of Drive status register.

By sending a relative or absolute position commands to Drive, the linear coordinated motion will be stopped.

suppose R0 = sqrt((X0-Xc)*(X0-Xc) + (Y0-Yc)*(Y0-Yc))
R1 = sqrt((X1-Xc)*(X1-Xc) + (Y1-Yc)*(Y1-Yc))
Make sure the difference of R0 and R1 is less or equal 1, otherwise the Drive can not find the final position of circular arc during circular motion.

sqrt((X0-X1)*(X0-X1) + (Y0-Y1)*(Y0-Y1)) > 4, means this function can not be used to make a whole circle, in order to make a whole circle, have to use this function two times.

-134217728 =< X0,Y0,X1,Y1,Xc,Yc <= 134217727

Also the radius of circle should be less than 134217727.

If the Drive 0 and Drive 1 have different Gear number, the real shape is oval, suppose the pitch of screwbolt is the same for XY axes.

Note: for the communication function,packets, please check section4 for the part of RS232 communication protocol.

3.10 Gear Number

Gear number is from 500 to 16384, default value is 4096.

Gear number provided a electrical gear ratio : 4096 / Gear_Num, from 0.25 ~ 8.192.

For example, if Gear number = 4096, the 16384 input counts from CW or CCW will turn motor exactly one turn.
If Gear number = 500, so 2000CWorCCW pulse will turn motor just one turn, motor seems used a 2000 pps optical encoder.

For analog input in position servo mode, the analog input is from 0 ~ 5(V) range, by using the Gear Number, 0~5(V) analog input can turn motor from 0 ~ 90 * 4096/Gear number (deg).

The gear number has the same effect on the serial Point to point movement or RS232 command input mode.

Gear number is only effective for position servo mode.

3.11 Default Value for servo cons and default mode

```
Position servo mode
RS232 position data input mode
Relative servo start position

MainGain = 16;
SpeedGain = 4;
IntGain = 24;
TrqCons = 127;
MaxSpd = 20;
MaxAcl = 8;
OnRange = 4;
Gear Number = 4096;
Drive_ID = 0;
Config = 0x00;
```

3.12 Drive configuration and status Register

Drive configuration such as RS232 data input or CW/CCW data input etc.. also the Drive now is normal or alarm, all of these are described by two register **Config** and **Status** which are stored inside Drive's EEPROM and can be read or set through RS232 communication.

Dirver status is a byte data, lower 7 bit valid for indicating the Drive status, is it in the state of servo, alarm, on position, or free.

Status = x b6 b5 b4 b3 b2 b1 b0

```
b0 = 0 : On position, i.e. |Pset - Pmotor| < = OnRange
b0 = 1 : motor busy, or |Pset - Pmotor|> OnRange

b1 = 0 : motor servo
b1 = 1 : motor free

b4 b3 b2 = 0 : No alarm
           1 : motor lost phase alarm, |Pset - Pmotor|>8192(steps), 180(deg)
           2 : motor over current alarm
           3 : motor overheat alarm, or motor over power
           4 : there is error for CRC code check, refuse to accept current command
         5~ 7 : TBD

b5 = 0 : means buit in S-curve,lieanr,circular motion wait for next motion
b5 = 1 : means buit in S-curve,lieanr,circular motion is busy on current motion
b6     : pin2 status of JP3,used for Host PC to detect CNC zero position or others
```

Drive configuration for communication mode, servo mode etc is expressed by a byte called

Config = x b6 b5 b4 b3 b2 b1 b0

```
b1 b0 = 0 : RS232 mode
```

```
          1 : CW,CCW mode
          2 : Pulse/Dir or (SPI mode Optional)
          3 : Anlog mode


   b2    = 0 : works as relative mode(default) like normal optical encoder
   b2    = 1 : works as absolute position system, motor will back to absolute zero
         or POS2(Stored in sensor)automatically after power on reset.

   b4 b3 = 0 : Position servo as default
           1 : Speed servo
           2 : Torque servo
           3 : TBD

   b5    = 0 : let Drive servo
   b5    = 1 : let Drive free, motor could be turned freely

   b6        : TBD
```

Config can be set as the customer requirement by DmmDrv GUI interface. The default Config = x0000000, RS232 communication mode, absolute position sensor works as relative mode, position servo, Drive active not free.
If the bit 5 of Config register be set to be 1, Drive will let motor free.

Status and Config are also called Status register and Config register.

4. RS232 communication protocol

The RS232 port is always active after power on for Dyn-series Drive, that active RS232 port could be used for reading and setting Drive parameters and status, also could be used for sending point to point position command if the RS232 mode is selected for position command input.

If the position command input mode is selected as CW/CCW mode, or SPI,Analog mode, the RS232 port is still active as mentioned above but it only can be used for reading and setting Drive parameters.

The RS232 port could be easily accessed by using the GUI interface DmmDrv.exe after the connection between PC and the Drive's RS232 port. This is the easiest way to tune up the servo and make some test movement.

The RS232 port could be accessed by other microcontroller, or DSP too if the sending and reading data by using the Dmm Drive's RS232 protocol.

The PC or DSP is working as Master and the Drive is always as slave.

Here are the detailed description for the RS232 communication protocol. Several Drive could be linked for a serial network.

4.1 Definition

Byte : consists of 8 bits,represented by b7b6b5b4b3b2b1b0 or b[7:0]. b7 is MSB and b0 is LSB, so called little endian.

Packet: consists of several bytes, expressed as

P = Bn Bn-1 Bn-2 .... B1 B0, length n+1

Bn is start byte, B0 is end byte, similar to the byte structure, Bn is MSB and B0 is LSB as little endian rule.

The integer n varys as the variation of packet length. Functionally, a packet could be expressed as

P = ID + packetLength + functioncode + data + checksum

```
                          ID   : takes one byte
packetLength + functioncode   : takes one byte
                        data   : takes one byte to three bytes
                    checksum   : takes one byte
```

Minimum packet is 4 bytes,packet length 4(n=3), 1 data byte.
Maximum packet is 7 bytes,packet length 7(n=6), 4 data byte.

Minimum packet length is 4, there is at least a data byte,for some function code, that data byte is meaning less, called dummy byte which can be set to any value(0~127) and do not affect the function of that packet.

The more details are described as following sections.

4.2 Features for the byte inside a packet

The start byte takes form of 0xxxxxxx, or MSB is 0, x for 0 or 1. Any other byte except of start byte takes the form of 1xxxxxxx, x coud be 0 or 1.

Most significant bit in a byte can be use for determining it is a packet's start byte or not.

4.3 Details for the byte in the packet

4.3.1 Start byte Bn

The MSB bit of start byte is always zero, the other seven bits are used for the Drive ID number which is from 0 ~ 126. The ID number can be assigned through the DmmDrv GUI interface.

ID number 127 is reserved for every Drive for the broadcasting purpose in other words, it is general ID number.

4.3.2 Bn-1 byte

The Bn-1 byte is used for representing the function and packet length.

Bn-1 = 1 b6 b5 b4 b3 b2 b1 b0

The bit b6 and b5 are for the length of packet, expressed as

b6	b5	packet length(=n+1)
0	0	4
0	1	5
1	0	6
1	1	7

The bit b4~b0 are used for the packet function, expressed as

Function(sent by master)	b[4:0]	Data (bytes)	Remarks
Set_Origin	0x00	1(dummy)	;set current position ;as zero ;
Go_Absolute_Pos	0x01	1~4	; ;
Make_LinearLine	0x02	1~4	; ;
Go_Relative_Pos	0x03	1~4	; ;
Make_CircularArc	0x04	1~4	; ;
Assign_Drive_ID	0x05	1	;Assign ID to Drive
Read_Drive_ID	0x06	1(dummy)	;
Set_Drive_Config	0x07	1	;One byte Config,see ;Config Register
Read_Drive_Config	0x08	1(dummy)	;Read Drive ;config Register
Read_Drive_Status	0x09	1(dummy)	;ask for Drive status
Turn_ConstSpeed	0x0a	1~3	
Square_Wave	0x0b	1~3	
Sin_Wave	0x0c	1~3	
SS_Frequency	0x0d	1~3	
Read_PosCmd32	0x0e	1~4	;Read Drive pos set
TBD	0x0f		
Set_MainGain	0x10	1	
Set_SpeedGain	0x11	1	
Set_IntGain	0x12	1	
Set_TrqCons	0x13	1	
Set_HighSpeed	0x14	1	;Set MaxSpd,1~127
Set_HighAccel	0x15	1	;Set MaxAcl,1~127
Set_Pos_OnRange	0x16	1	;if |Pset-Pmotor|<= ;OnRange, ;motor on Pos, OnRange ;1~127
Set_GearNumber	0x17	2	;Gear_Number ;500~16384
Read_MainGain	0x18	1(dummy)	
Read_SpeedGain	0x19	1(dummy)	
Read_IntGain	0x1a	1(dummy)	
Read_TrqCons	0x1b	1(dummy)	
Read_HighSpeed	0x1c	1(dummy)	
Read_HighAccel	0x1d	1(dummy)	
Read_Pos_OnRange	0x1e	1(dummy)	
Read_GearNumber	0x1f	1(dummy)	;Received:Gear_Number

Fuction(sent by Drive)

Not used		0x00 ~ 0x0a	
Is_MainGain		0x10	1
Is_SpeedGain		0x11	1
Is_IntGain		0x12	1
Is_TrqCons		0x13	1
Is_HighSpeed		0x14	1
Is_HighAccel		0x15	1
Is_Drive_ID	0x16	1	
Is_PosOn_Range		0x17	1
Is_GearNumber		0x18	2
Is_Status		0x19	1
Is_Config		0x1a	1
Is_PosCmd32		0x1b	1~4
TBD		0x1c~0x1f	

Depends on the function and data, the packet length will vary.

Bn-1 = 0x80 + (PacketLenght-4)*32 + FunctionCode, the FunctionCode could be Back_To_Origin, or Set_MainGain etc.

4.3.3 Bn-2 ~ B1 bytes

Bn-2 ~ B1 (n>2) are used for representing the data in the packet.

Bn-2 is MSByte, B1 is LSByte for the data and 7bits of a byte is used for containing the data.

n	data range	
3	-64 ~ 63	;Only B1 is used
4	-8192 ~ 8191	;Only B2 B1 are used
5	-1048576 ~ 1048575	;B3 B2 B1 are used
6	-134217728 ~ 134217727	;B4,B3 B2 B1 are used

Minimum packet length is 4, there is at least a data byte, for some function code, that data byte is meaning less, called dummy byte which can be set to any value(0~127) and do not affect the function that packet.

4.3.4 B0 Byte

B0 byte is used for check sum, which is calculated from Bn~B1 as

S = Bn + Bn-1 + Bn-2 +.... B1

B0 = 0x80 + Mod(S , 128), B0 = 0x80 + S - 128*[S/128],
B0 = 128 ~ 255

After receiving a packet, then calculate Temp = Mod(S , 128),

if Temp = B0 , there is no error, otherwise there is error during the packet transmission.

4.4 Example for the packet structure

Example 1: Make 3th axis motor right now position be the absolute zero position(= 0), ID = 3. One byte dummy data 0x00,PacketLenght = 4.

B3 = 0x03
B2 = 0x80 + (PacketLenght-4)*32 + Set_Origin =0x80 + 0x00=0x80
B1 = 0x80 + 0x00 = 0x80
S = B3 + B2 + B1 = 0x03 + 0x80 + 0x80 = 0x103
B0 = 0x80 + Mod(S,128) = 0x83

As shown in the appendix, by calling the subroutine : Send_Package(0x03,0), when Global_Func = (char)Set_Origin = 0x00. The code will generate above B3~B0.

The motor power on position is the default absolute zero position, or it is the position set by using set absolute zero function(0x00)

Example 2: Make 3th axis motor back to absolute zero position(= 0), ID = 3. Move to position 0 = 0x00, One byte data,PacketLenght = 4.

B3 = 0x03
B2 = 0x80 + (PacketLenght-4)*32 + Go_Absolute_Pos=0x80 + 0x01=0x81
B1 = 0x80 + 0x00 = 0x80
S = B3 + B2 + B1 = 0x03 + 0x81 + 0x80 = 0x104
B0 = 0x80 + Mod(S,128) = 0x84

Example 3: Make 3th axis motor move 120(steps) from right now position, ID = 3.

120 = 0x78 = 0x0111 1000 > 63,Two byte data, high 7bits 000 0000= 0x00, lower 7bits = 111 1000 = 0x78. And use function Go_Relative_Pos (=0x03),PacketLength = 5.

B4 = 0x03
B3 = 0x80 +(PacketLength-4)*32+Go_Relative_PosP = 0x80+0x03 = 0xa3
B2 = 0x80 + 0x00 = 0x80
B1 = 0x80 + 0x78 = 0xf8
S = B4 + B3 + B2 + B1 = 0x03 + 0xa3 + 0x80 + 0xf8 = 0x21e
B0 = 0x80 + Mod(S , 128) = 0x80 + 0x1e = 0x9e

Example 4: Make 3th axis motor move -120(steps) from right now position, ID = 3.

-120 = 0x88 = 0xff88 < -63,Two byte data.
0xff88 = 0x1111 1111 1000 1000, lower 7bits = 000 1000 = 0x08
higher 7bits = 0111 1111 = 0x7f
so use function Go_Relative_Pos(=0x03),PacketLength = 5.

B4 = 0x03;
B3 = 0x80 +(PacketLength-4)*32 + Go_Relative_Pos = 0x80 +0x04 =0xa3.

B2 = 0x80 + 0x7f = 0xff
B1 = 0x80 + 0x08 = 0x88
S = B4 + B3 + B2 + B1 = 0x03 + 0xa3 + 0xff + 0x88 = 0x22d
B0 = 0x80 + Mod(S , 128) = 0x80 + 0x2d = 0xad

Example 5: Make 2th axis motor turn at 60rpm, ID = 2.

speed is 60, onebyte data is enough, 60 = 0x3c
PacketLength = 4.

B3 = 0x02;
B2 = 0x80 +(PacketLength-4)*32 + Turn_ConstSpeed = 0x80 + 0x0a = 0x8a
B1 = 0x80 + 0x3c = 0xbc
S = B3 + B2 + B1 = 0x02 + 0x8a + 0xbc
B0 = 0x80 + Mod(S , 128) = 0xc8

Example 6: Make 2th axis motor turn at -60rpm, ID = 2.

speed is -60 = 0xc4 = 0x1100 0100 > -63, One byte data 7bits =
0x0100 0100 = 0x44
PacketLength = 4.

B3 = 0x02;
B2 = 0x80 +(PacketLength-4)*32 + Turn_ConstSpeed = 0x80+0x40+0x0a = 0x8a
B1 = 0x80 + 0x44 = 0xc4
S = B3 + B2 + B1 = 0x02 + 0x8a + 0xc4 = 0x150
B0 = 0x80 + Mod(S , 128) = 0x80 + 0x50 = 0xd0

Example 7: Make a line on X-Y Plane
Suppose right now position for three motors are(X0,Y0,Z0) = (0,0,0),
and the End point of straight line is (X1,Y1,Z1) = (100,200,0)

Always use General ID = 0x7f
The feedrate = 3, could be from 1~127
Global_Func = (char)Make_LinearLine = 0x02;

Then send four packets to the Drives as:
Send_Package(ID,X1 - X0), i.e. Send_Package(0x7f,100)
Send_Package(ID,Y1 - Y0), i.e. Send_Package(0x7f,200)
Send_Package(ID,Z1 - Z0), i.e. Send_Package(0x7f,0)
Send_Package(ID,FeedRate),i.e. Send_Package(0x7f,3)

After the X-Y-Z three Drives received all four packets, they will
start to move until the meet the end point of (X1,Y1,Z1). Three motors
will meet (X1,Y1,Z1) at the same time.

During the lienar or circular interpolation motion, the
Read_Drive_Status (=0x09) can used to read Drive's status register to check whether b5 = 0 or not, b5 = 0
means the coordinated motion be finished.

Send_Package(ID,Y1 - Y0) is the subroutine in the appendix, it will
generate a packet as above examples.

Example 8: Make a circular arc on X-Y Plane
Suppose right now position for three motors are(X0,Y0) = (0,0),

and the End point of arc is (X1,Y1) = (200,0) in CW direction.
It is easy to know the center of arc is (Xc,Yc) = (100,0)

The Feedrate = 1, could be from 1~127>0, because in CW direction otherwise be negative value.
The planeNumber = 0 because it is in X-Y plane
TwoBytes = (PlaneNumber<<8) | FeedRate = 0*256 + 1 = 1
use General ID = 0x7f
Global_Func = (char)Make_CircularArc = 0x04;

Then send five packets to the Drives as:
Send_Package(ID,X0 - Xc), i.e. Send_Package(0x7f,-100)
Send_Package(ID,Y0 - Yc), i.e. Send_Package(0x7f,0)
Send_Package(ID,X1 - Xc), i.e. Send_Package(0x7f,100)
Send_Package(ID,Y1 - Yc), i.e. Send_Package(0x7f,0)
Send_Package(ID,TwoBytes),i.e. Send_Package(0x7f,1)

After the X-Y-Z three Drives received all four packets, Only two of three motors will move and finally will meet (X1,Y1) at the same time.

During the lienar or circular interpolation motion, the Read_Drive_Status (=0x09) can used to read Drive's status register to check whether b5 = 0 or not, b5 = 0
means the coordinated motion be finished.

Two half arcs must be made for making a whole circle.

4.5 Motion through RS232 input mode in DmmDrv.exe Edit and RunningCode dialogue box

Use another interface in DmmDrv's Edit and RunningCode box, user can edit five text files, Program1~Program5.txt should be located in the same directory as DmmDrv.exe.

By using several simple button, user can edit, revise,save, the point to point movement files then use RunningCode to run it through RS232 input mode.

xxx: AMOV , axis , Absolute position

function : move to a specified absolute position
axis : Drive ID number
Absolute position : 0 ~ +-134217728

The line number xxx is from 0 to 999.

xxx: MOV , axis , move distance

function : move to a next position from current one
axis : Drive ID number
move distance : 0 ~ +-134217728

Note:if move distance is positive, motor turn in clockwise(CW) direction from the view of motor mountside, otherwise turn in the CCW direction.

xxx: GOTO , Line number

function : goto a specified line
Line number : Any line number inside the file

xxx: TIMER , number

function : delay a certain time
number : 1~999, delay time = number * 0.01(s)

xxx: SIN , axis , Amplitude

function : motor make sinusoidal movement
axis : Drive ID number
Amplitude : Sinusoidal Amplitude

xxx: SQUARE , axis , Amplitude

function : motor make step movement
axis : Drive ID number
Amplitude : Step Amplitude

xxx: FREQ , axis , Frequency

function : Setting Sin and Step frequency
axis : Drive ID number
Frequency : 1~30(Hz)

xxx: SPEED , axis , Speed Value

function : Make motor turning at a constant speed
axis : Drive ID number
Speed Value : -6000 ~ 6000(rpm)

xxx: MAXSPD , axis , MaxsPd value

function : Setting MaxSpd for S-curve
axis : Drive ID number
MaxSpd Value : 1~127

xxx: MAXACL , axis , MaxAcl value

function : Setting MaxAcl for S-curve
axis : Drive ID number
MaxAcl Value : 1~127

xxx: ASGNID , 127 , new ID

function : Setting a new ID to Drive(when only one Drive on the RS232port)
new ID : 0 ~ 126

Note : All of command like MOV,AMOV have to be in CAPITAL.

How to edit them?, double click any button like AMOV,or MOV, an instruction will appear in the current line box then use addin or insert or replace put it into program after revising the parameters.

Double click any line in the program, the clicked line will appear in the current line box then use replace put it back after revising parameters.

Install the DmmDrv.exe and program1~5.txt, copy all the files into a user directory.

Any other text file editor can be used for editing program1~5.txt(must be text file form), such as wordpad, or notepad from microsoft.

Please have a reference at G-Code Guide for the details of G-Code.

4.6 The RS232 communication format

Baud rate: 38400
Stop bit: 1
Odd/Even Verify bit: No

4.7 The RS232 signal level and intermediate RS232 cable

RxD and TxD RS232 signal from connector JP2 is TTL/CMOS level,

The intermediate cable is between PC's RS232 cable and JP2, There is a RS232 buffer inside intermediate cable connector, this buffer will shift TTL/CMOS and RS232 level.

5. RS485 network

Several Drives can be connected by RS485 after every Drive on the RS485 net have been designated an identical ID number.

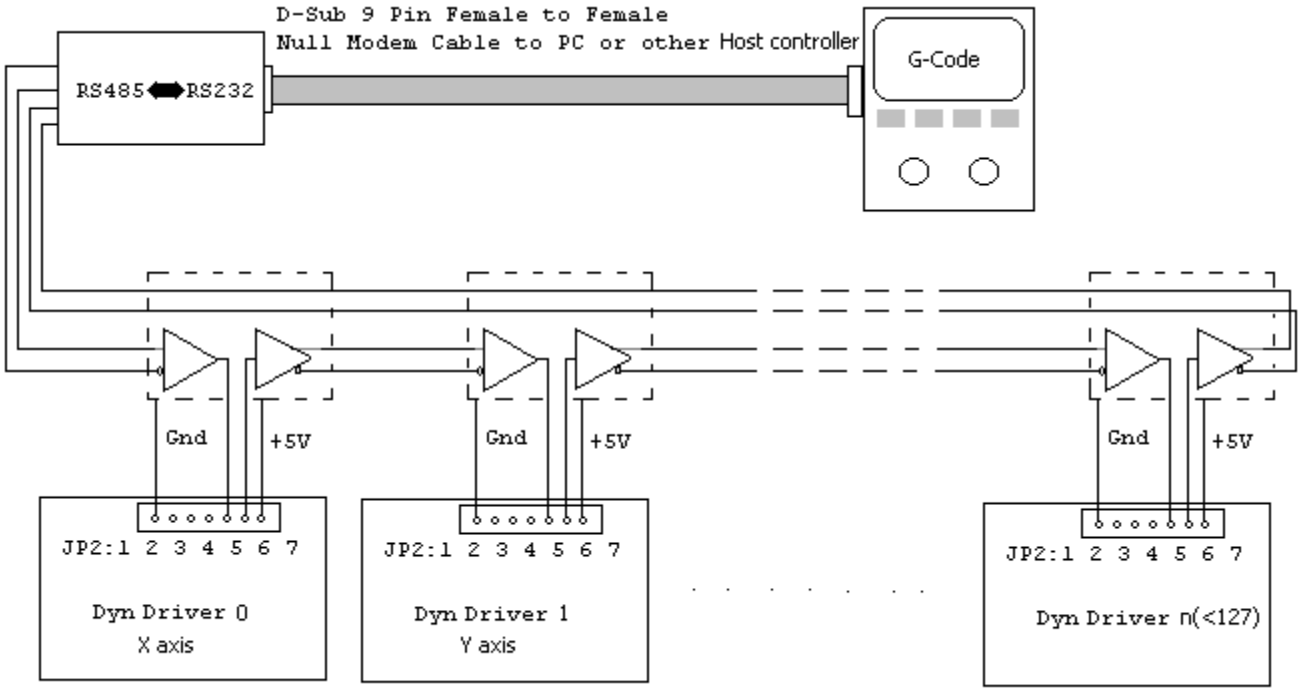
The RS485 check box must be checked if RS485 network is used which means there are at least two or more Drive on the net, then every Drive's servo mode, cons can be read or set according to the ID number on the servo setting dialog box. The ID number can not be assigned to a particular Drive if RS485 net is used.

The Drive's **ID CAN ONLY BE SET** when there is one Drive there, then assigned a new ID number to that Drive without check the RS485 check box.

The RS485 network is a serial network, if there is a packet in the network, one Drive will receive it first, if the packet's ID number is the same as the Drive's, that packet will be received by the Drive, otherwise that packet will be relayed to the next Drive.

Since the packet ID is contented in the first byte of the packet, when a Drive received the first byte of packet, it will make judgment for receiving it or relaying it right away, so the data flow on that serial RS485 net is very fast, efficient.

The RS485 net is shown as below, every Drive has a RS485NET node which contains a RS485 buffer such as LTC491.



Notice

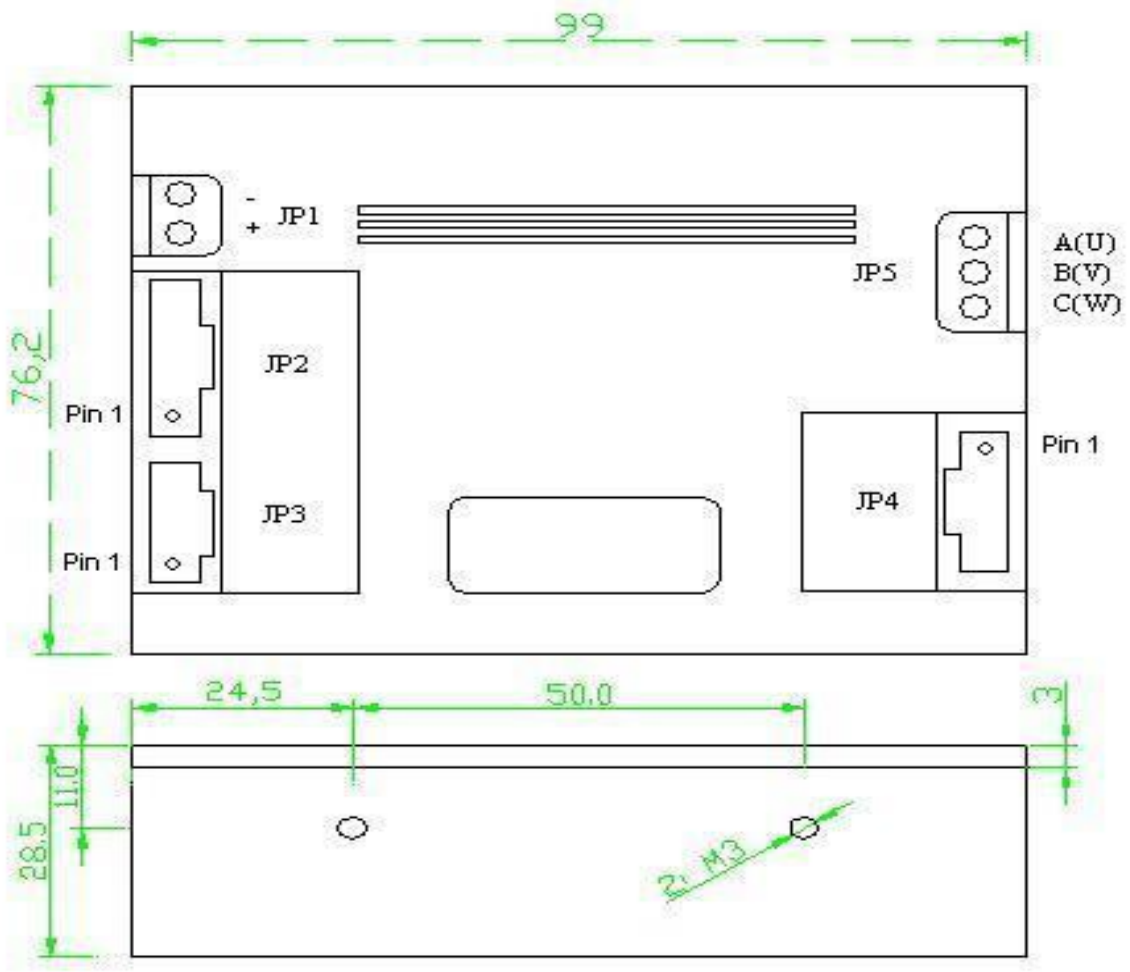
1 : If the RS485 or RS232 network is connected, the RS485/232 net checkbox must be checked when the servo cons are set into Drive,
so the servo cons will be saved into the specified Drive not save into several Drives.

2 : There should be ONLY one Drive connected on the RS232 port,
then the Drive's ID could be assigned correctly.

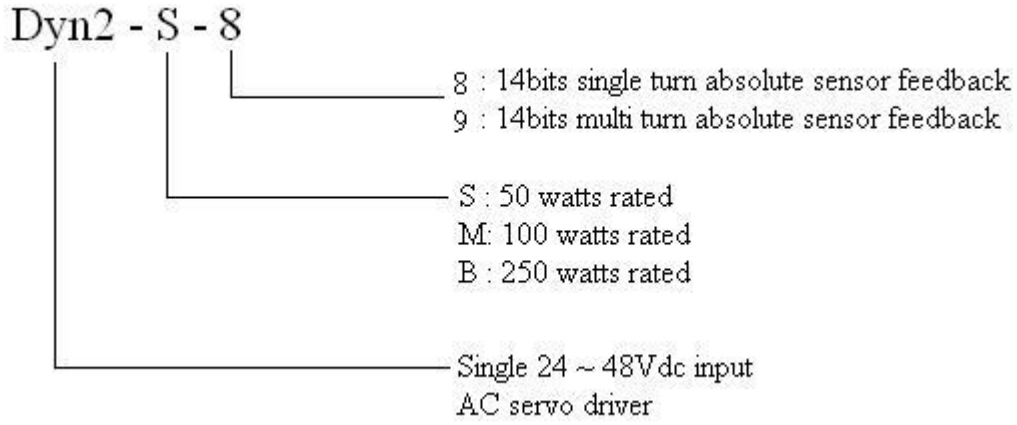
6. Environment

Operation temperature range	: -10 ~ 60°(C)
Maximum operation humidity	: 95RH%(no dew)
Storage temperature range	: -20 ~ 70°(C)
Maximum storage humidity	: 95RH%(no dew)

7. Dimension of Dyn2 Drive for driving 50~250Watts motor, 24~48Vdc



Dyn2-series AC servo order form



Appendix : C++ code for making packet of serial communication protocol

Note: in the description of RS232 communication protocol, the last byte of packet is always B0, but in the code of below, the first byte is always B0.

```
#define Go_Absolute_Pos             0x01
#define Is_AbsPos32                 0x1b
#define General_Read                0x0e

      char InputBuffer[256];              //Input buffer from RS232,
      char OutputBuffer[256];             //Output buffer to RS232,
      unsigned char InBfTopPointer,InBfBtmPointer;//input buffer pointers
      unsigned char OutBfTopPointer,OutBfBtmPointer;//output buffer pointers

      unsigned char Read_Package_Buffer[8],Read_Num,Read_Package_Length,Global_Func;

      long Motor_Pos32;

void DlgRun::ReadPackage()
{
      unsigned char c,cif;

      ReadRS232Port(); //     by Customer's codes
      while(InBfBtmPointer!=InBfTopPointer)
      {
            c = InputBuffer[Comm.InBfBtmPointer];
            InBfBtmPointer++;
            cif = c&0x80;
            if(cif==0)
            {
                  Read_Num = 0;
                  Read_Package_Length = 0;
            }

            if(cif==0||Read_Num>0)
            {
                  Read_Package_Buffer[Read_Num] = c;
                  Read_Num++;
                  if(Read_Num==2)
                  {
                        cif = c>>5;
                        cif = cif&0x03;
                        Read_Package_Length = 4 + cif;
                        c = 0;
                  }

                  if(Read_Num==Read_Package_Length)
                  {
                        Get_Function();
                        Read_Num = 0;
                        Read_Package_Length = 0;
                  }
            }

      }
}

void DlgRun::Get_Function(void)
{
      char ID,Function_Code,CRC_Check;

      ID = Read_Package_Buffer[0]&0x7f;
      Function_Code = Read_Package_Buffer[1]&0x1f;

      CRC_Check = 0;
      for(int i=0;i<Comm.Read_Package_Length-1;i++)
```

```
        CRC_Check += Read_Package_Buffer[i];
        CRC_Check ^= Read_Package_Buffer[Comm.Read_Package_Length-1];
        CRC_Check &= 0x7f;
       if(CRC_Check!= 0)
             //    MessageBox("There is CRC error!");

       switch(Function_Code){
       case  Is_AbsPos32:
             Motor_Pos32 = Cal_SignValue(Read_Package_Buffer);
             break;
       default:;
       }
}

/*Get position with sign                                              */
long DlgRun::Cal_SignValue(unsigned char One_Package[8])
{
       char Package_Length,OneChar,i;
       long Lcmd;
       OneChar = One_Package[1];
       OneChar = OneChar>>5;
       OneChar = OneChar&0x03;
       Package_Length = 4 + OneChar;

       OneChar = One_Package[2];           /*First byte 0x7f, bit 6 reprents sign
       */
       OneChar = OneChar << 1;
       Lcmd = (long)OneChar;               /*Sign extended to 32bits
             */
       Lcmd = Lcmd >> 1;
       for(i=3;i<Package_Length-1;i++)
       {
             OneChar = One_Package[i];
             OneChar &= 0x7f;
             Lcmd = Lcmd<<7;
             Lcmd += OneChar;
       }
       return(Lcmd);                              /*Lcmd : -2^27 ~ 2^27 - 1
                    */
}

//***************** Every Robot Instruction ******************
// Send a package with a function by Global_Func
// Displacement: -2^27 ~ 2^27 - 1
void DlgRun::Send_Package(char ID , long Displacement)
{
       unsigned char B[8],Package_Length,Function_Code;
       long TempLong;
       B[1] = B[2] = B[3] = B[4] = B[5] = (unsigned char)0x80;
       B[0] = ID&0x7f;
       Function_Code = Global_Func & 0x1f;

       TempLong = Displacement & 0x0fffffff;           //Max 28bits
       B[5] += (unsigned char)TempLong&0x0000007f;
       TempLong = TempLong>>7;
       B[4] += (unsigned char)TempLong&0x0000007f;
       TempLong = TempLong>>7;
       B[3] += (unsigned char)TempLong&0x0000007f;
       TempLong = TempLong>>7;
       B[2] += (unsigned char)TempLong&0x0000007f;
       Package_Length = 7;

       TempLong = Displacement;
       TempLong = TempLong >> 20;
       if(( TempLong == 0x00000000) || ( TempLong == 0xffffffff))
```

```
	{//Three byte data
		B[2] = B[3];
		B[3] = B[4];
		B[4] = B[5];
		Package_Length = 6;
	}

	TempLong = Displacement;
	TempLong = TempLong >> 13;
	if(( TempLong == 0x00000000) || ( TempLong == 0xffffffff))
	{//Two byte data
		B[2] = B[3];
		B[3] = B[4];
		Package_Length = 5;
	}

	TempLong = Displacement;
	TempLong = TempLong >> 6;
	if(( TempLong == 0x00000000) || ( TempLong == 0xffffffff))
	{//One byte data
		B[2] = B[3];
		Package_Length = 4;
	}

	B[1] += (Package_Length-4)*32 + Function_Code;

	//long x = Cal_SignValue(B);
	Make_CRC_Send(Package_Length,B);
}

void DlgRun::Make_CRC_Send(unsigned char Plength,unsigned char B[8])
{
	unsigned char Error_Check = 0;
	for(int i=0;i<Plength-1;i++)
	{
		OutputBuffer[OutBfTopPointer] = B[i];
		OutBfTopPointer++;
		Error_Check += B[i];
	}
	Error_Check = Error_Check|0x80;
	OutputBuffer[OutBfTopPointer] = Error_Check;
	OutBfTopPointer++;

	//Send to RS232 port  by customer's codes;
}


void main(void)
{
	/*Move motor 2 to the position of 321456*/
	char Axis_Num = 2;
	Global_Func = (char)Go_Absolute_Pos;
	long pos = 321456;
	Send_Package(Axis_Num,Pos);

	/*Below are the codes for reading the motor shaft 32bits absolute position */
	int i;
	InBfTopPointer = InBfBtmPointer = 0;//input buffer pointers
	OutBfTopPointer = OutBfBtmPointer = 0;//output buffer pointers

	for(i=0;i<8;i++)
		Read_Package_Buffer[i] = 0;

	Read_Num = Read_Package_Length = 0;
```

```
	//Read motor 32bits position
	char ID = 0; //Suppose read 0 axis motor
	Global_Func = General_Read;
	Send_Package(ID , Is_AbsPos32);

	while(i<10000) //10~20ms waiting
	{
		i++;
	}

	ReadPackage();		//Motor_Pos32 will be there

}
```