SERVO-300

3-axis servo motor control card

User Manual

Version 2.0 02/2001 Edition

Driver update : http://www.icpdas.com

Warranty: All products manufactured by ICP DAS are warranted against defective materials for one year from the date of delivery to the original purchaser

Warning: ICP DAS assumes no liability for damage consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for it's use, nor for any infringements of patents or other rights of third parties resulting from it's use.

Trademark
The names used for identification only maybe registered trademarks of their respective companies.

SERVO-300 --- 3-Axis Servo Motor Control Board

Servo-300 is a 3-axis, command-type, servo motor control board. The embedded CPU of servo-300 performs the motion command transferred from host-PC via a 2K bytes FIFO. It also sends the position and status to host-PC via another 2K bytes FIFO. This buffer provides time buffer and it is very suitable for windows operating system. This board provides DOS, windows 95 and windows NT drivers.

Features

- 3-axis V-command servo motor board
- Simulation / open loop / close loop mode.
- Command-type execution, 49 commands
- Embedded CPU
- Linear interpolation, circular interpolation
- Programmable trapezoidal speed profile.
- Programmable DDA cycle.
- Programmable direction configuration.
- Programmable 2 speed home return, home preset.
- 2500 points per axis pitch error compensation
- Home, forward, backward limit switches per axis.
- Limit switch auto-protection
- Hardware failure detection
- 8 digital input, 7 digital output
- Software / hardware emergency stop
- 2500Vrms optical couple
- DOS, Windows 95, Windows NT driver
- BCB, VB, Delphi, LabView driver
- Programmable limit switch normal state: N.O. (normal open) or N.C. (normal close).

Option

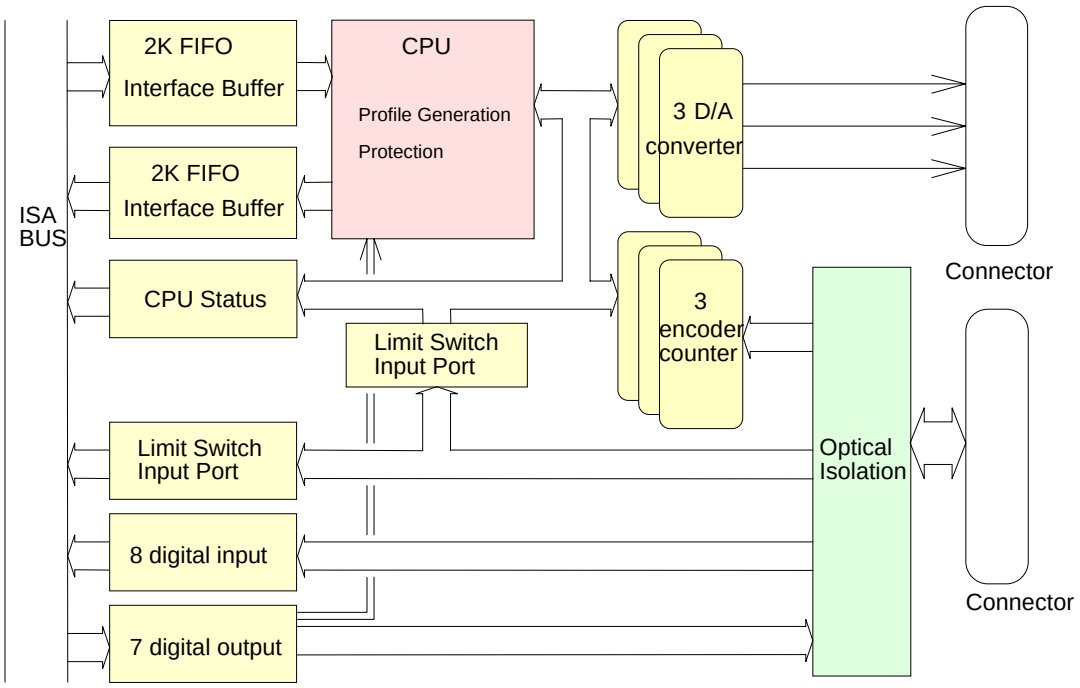
- DB-8R Daughter board --- 8 relay output, limit switch input board
- DB-200 Daughter board --- 3-axis encoder, analog output board.

Contents

1. Introduction

1.1 System Block Diagram

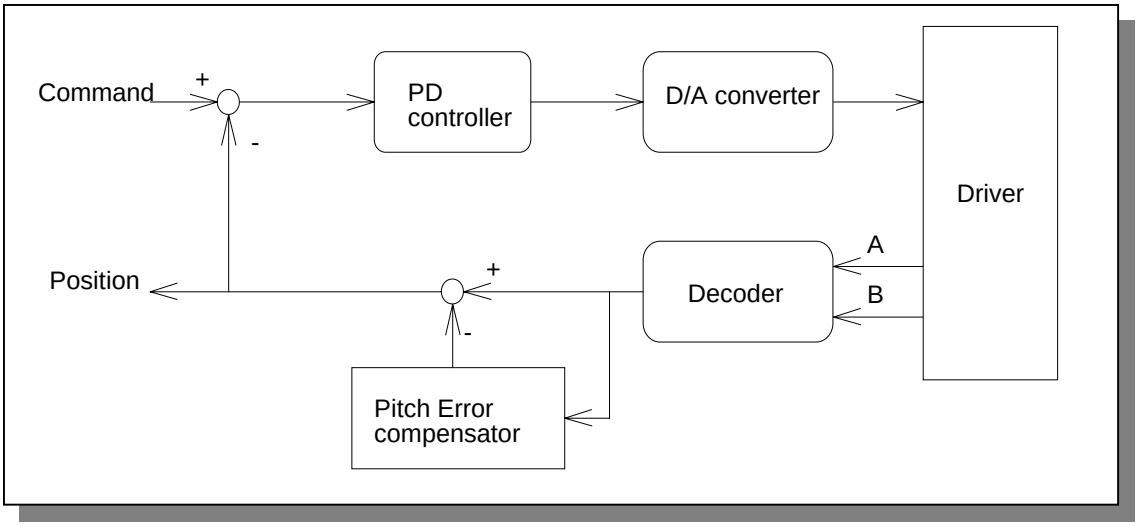
SERVO-300 is a microprocessor based and 3- axis V-command type Servo motor control board. It contains a 2K bytes-FIFO to receive motion command from host PC, and it also sends the position and status to host-PC via the other 2K bytes FIFO. The motion profile is generated by microprocessor. This microprocessor also handles auto-protection function. Each digital I/O supports 2500Vrms optical isolation.



Figure(1) block diagram of SERVO-300

1.2 The Control System

The sampling time is 3ms for 3-axis control system. The control loop includes a PD controller and a pitch error compensator. The gain parameter, Kp and Kd, of PD controller can be set from host PC. The pitch errors are recorded in a size 2500 table which can be set by user. The compensated pitch is 256 pulse, so the compensation range is 2500*256=640000 pulse. (The total length 800mm, if the screw pitch=5mm, encoder=4000pulse/rev.).



Figure(2) SERVO-300 control loop

Software DDA

For precise control and adjustable speed, the Software DDA will divide the command into several equal delta commands. The maximum command is 2040 pulse, so the maximum speed is

2040 / (0.003*DDA) * 60 / ENCODER.

For example, DDA=4, ENCODER=4000 pulse/rev.

delta command = 2040/4 = 508 pulse

speed = 2040/(0.003*4)*60/4000 = 2550 rpm

Therefore the maximum speed will be 2550 rpm while select DDA=4. In case of the user want to increase the maximum speed that the servo driver is available, the DDA value should be reduced. The maximum speed will be 5100 rpm if DDA=2. So, it is very important to select a proper DDA value for your system

1.3 The Operating Mode

For easily developing your system, SERVO-300 board provides three operating mode: simulation mode / open loop mode / close loop mode. Please refer to MSERVO3_SET_CONTROL_MODE() command.

Simulation mode

In simulation mode, the SERVO-300 control board will simulate the motion profile according to the motion command that received from host PC, and then the SERVO-300 will send the 3-axis positions back to host PC. The SERVO-300 control board will not output V-command to motor driver.

This mode is very easily used and efficient in the design phase. The simulation mode can be operated off machine. The user can debug and develop the software previously or at home. And if the user has the daughter board DB-8R, it can also simulate the digital input/output like as a machine.

Open loop mode

In open loop mode, user can directly output a constant voltage to servo driver, to let the motor rotate at constant speed. The constant value can be set by MSERVO3_CALV() command. This mode can be used to tune the velocity loop gain or to drive the inverter.

To tune velocity loop gain, user can set a small voltage for a short time. The position and velocity can be received from MSERVO3_GET_CARD (cardNo) command in a timer interrupt(~10ms).

Close loop mode

In close loop mode, the SERVO-300 performs the PD controller. It also executes the following function.

1. Pitch error compensation
2. Limit switch auto-protection
3.Hardware failure detection

.

The related information of the servo board can be received from MSERVO3_GET_CARD (cardNo) command using timer interrupt, please refer to chapter 3 software.

Software emergency stop

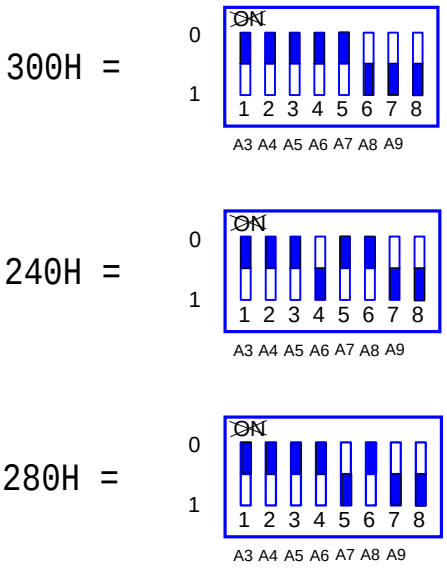
The servo command can be terminated from host-PC using software control. The user can use MSERVO3_STOP_ALL() command to terminate the servo commands being executed in SERVO-300 board. This command can clear all the commands pending in FIFO buffer.

2 Hardware

2.1 Address selection

The address is selected by A3~A9. There is a DIP switch on SERVO-300 board for address selection. The address can be selected as the following figures.

This address can be selected using MSERVO3_REGISTRATION( ) command. The MSERVO3_REGISTRATION( ) command has been described in chapter 3.



Figure(3) Address selection

2.2 Registers of SERVO-300 board

SERVO-300 has 6 registers, such as FIFO1 register, FIFO2 register, RSTFIFO1 register, DI register, DO register, MSC register.

(1) FIFO1 register (base + 0) (write only)

SERVO-300 driver will send motion command via this register. Please do not use this register to write anything, otherwise the SERVO-300 will not operate properly.

(2) DO register (base + 1) (write only)

MSB 7	6	5	4	3	2	1	0 LSB
don't use	DO6	DO5	DO4	DO3	DO2	DO1	DO0

MSB7 bit is reserved for special signal, please don't use it.

(3) RSTFIFO1 register (base + 2) (write only)

This register is used to reset FIFO1 for clear all of commands pending in the FIFO1 buffer.

(4) FIFO2 register (base + 0) (read only)

This register is used to receive the message coming from SERVO-300 board. This message includes SERVO-300 status, command position, actual position. Please refer to chapter 3 for more detail of message. Please don’t read this register, otherwise the message will lost.

(5) MSC register (base + 1) (read only)

MSB 7	6	5	4	3	2	1	0 LSB
/EMG	/Zstop	/Ystop	/Xstop	/F2HF	/F2FF	/F1FF	/F1EF

/Xstop, /Ystop, /Zstop:indicate which axis is stop, low active
/EMG : emergency switch, low active.
/F1EF: indicate FIFO1 is empty.
/F1FF: indicate FIFO1 is fully full.
/F2FF: indicate FIFO2 is fully full.
/F2HF: indicate FIFO2 is half full.

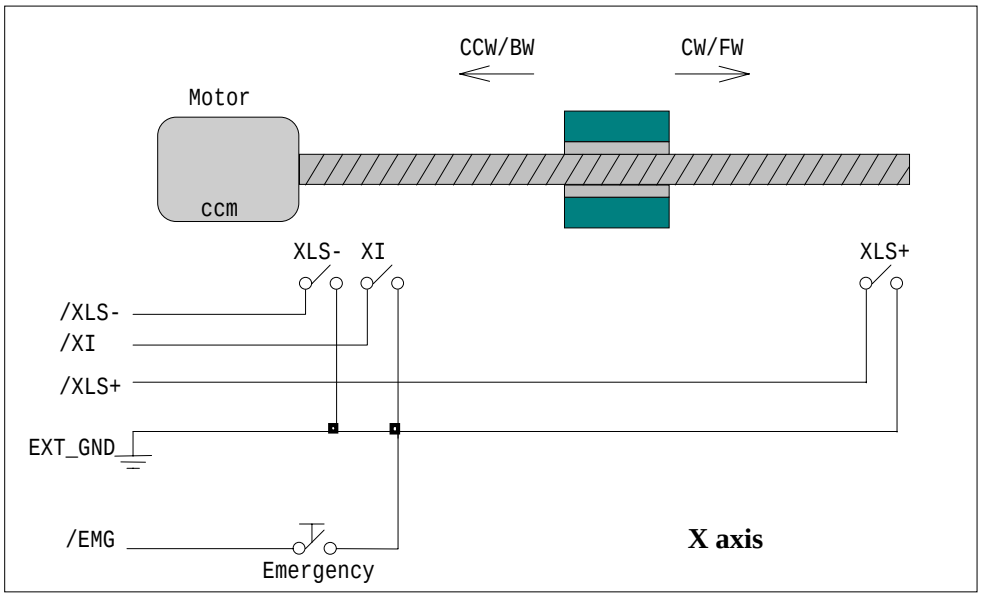
(6) DI register (base + 2) (read only)

MSB 7	6	5	4	3	2	1	0 LSB
di7	di6	di5	di4	Di3	di2	di1	di0

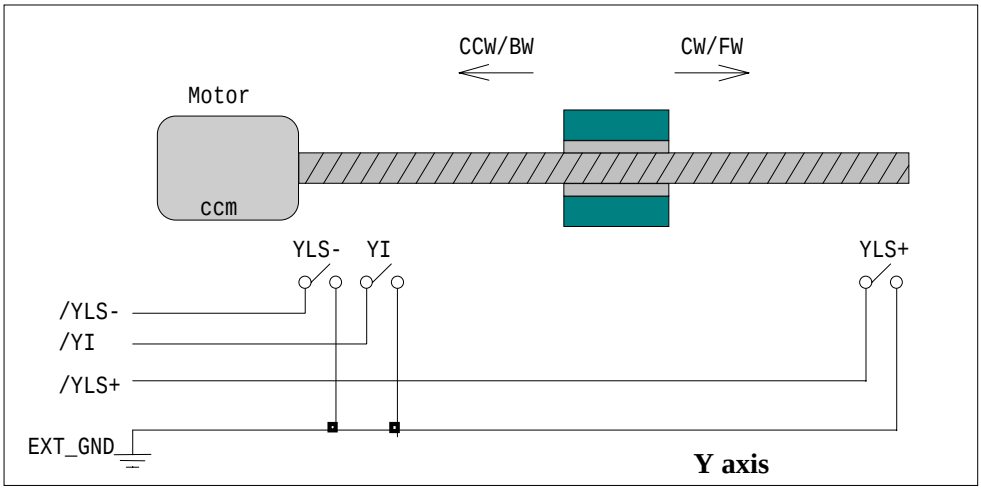
2.3 Hardware Configuration

2.3.1 Limit switch configuration

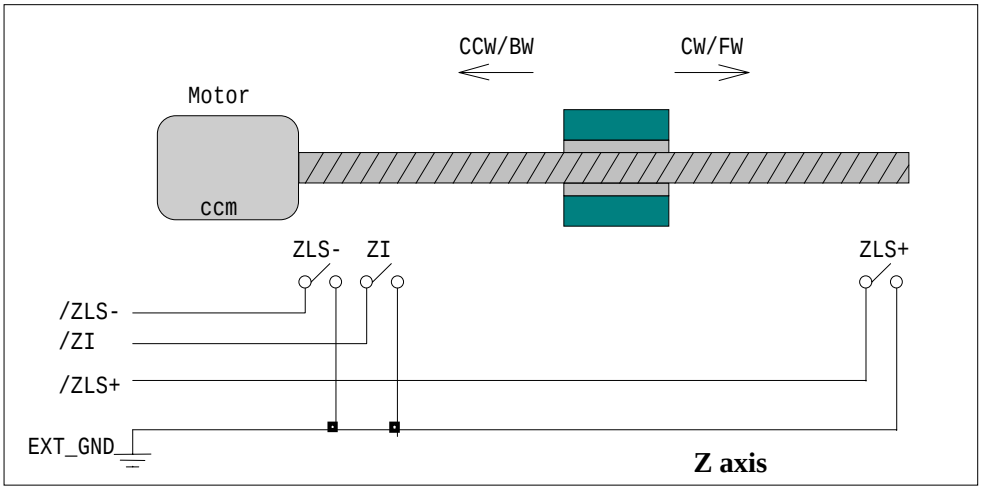
The profile generation and protection is executed by the CPU of SERVO-300 board, The limit switches must be configured as the following figure, otherwise the motion command won't work properly,.



Figure(4) Limit switch configuration of X axis



Figure(5) Limit switch configuration of Y axis



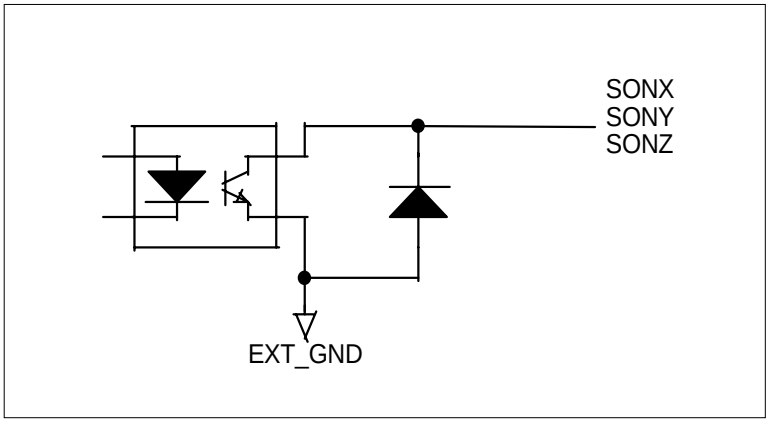
Figure(6) Limit switch configuration of Z axis

2.3.2 Direction configuration

Sometimes, the output direction of X-axis, Y-axis, Z-axis is not in desired direction due to motor connection or gear train. It is recommended to unify the output direction as shown in Figure(4)(5)(6). The CW/FW direction is defined as toward outside from motor and the CCW/BW direction is defined as toward inside to motor. The MSERVO3_SET_DEFDIR(cardNo, defdirX, defdirY, defdirZ) command provides parameters NORMAL_DIR (0) and REVERSE_DIR (1) to define the rotating direction of motor.

2.3.3 Turn Servo ON/OFF (Hold ON/OFF)

The MSERVO3_SET_SERVO_ON(cardNo, sonX, sonY, sonZ) command provides parameters ON (1) and OFF (0) to turn Servo ON or OFF. The internal circuitry of Servo-ON is sink-type connection as the following figure.



Figure(7) internal circuitry of Servo-ON signal

2.4 Auto-protection

SERVO-300 board supports a automatic protection system.

(a) When X(Y)(Z)-axis command is executed and the motor moves toward CW/FW direction, X(Y)(Z)-axis will immediately stop when XLS+(YLS+)(ZLS+) is touched. To release this protection, the X(Y)(Z)-axis must move toward CCW/BW direction.

(b) When X(Y)(Z)-aixs command is executed and the motor moves toward CCW/BW direction, X-axis will immediately stop when XLS-(YLS-)(ZLS-) is touched. To release this protection, the X(Y)(Z)-axis must move toward CW/FW direction.

(c) When any of the /EMG switches is touched, all motion command will be terminated and all motors will stop immediately. Meanwhile, the servo ON signal will be automatical turn off for rotating the shaft by manual. The servo ON signal will recover after released the /EMG switches.

2.5 Hardware failure detection

When any of the motor driver, encoder, connection or wire fail, SERVO-300 board will detect it and immediately terminate the motion command of each axis. The error status will be sent to host-PC (refer to chapter 3). And then, user must stop all the commands by MSERVO3_STOP_ALL(cardNo) and reset the system by MSERVO3_RESET_SYSTEM(cardNo) . After this procedure, the hardware error can be cleared. The parameter “error_range” can be used to adjust the failure condition. The “error_range” can set by MSTEP_SET_CONFIG() command.

2.6 Connection

(1) Pin assignment

Table(1) CN1 connector

pin name	pin number	description
VOUT1	1	analog outputof X axis
AGND	2	analog ground of X axis
VOUT2	3	analog output of Y axis
AGND	4	analog ground of Y axis
	5	Reserved
SON1	6	servo on signal of X axis
EXT_GND	7	external ground
SON2	8	servo on signal of Y axis
EXT_GND	9	external ground

Table(2) CN2 connector

pin name	pin number	description
VOUT3	1	analog output of Z axis
AGND	2	analog ground of Z axis
SON3	3	servo on signal of Z axis
VEXT	4	external voltage (apply 12~24V)
EXT_GND	5	external ground
/ZI	6	home index switch of Z axis, low active
/ZLS+	7	forward limit switch of Z axis, low active
/ZLS-	8	backward limit switch of Z axis, low active
	9	Reserved

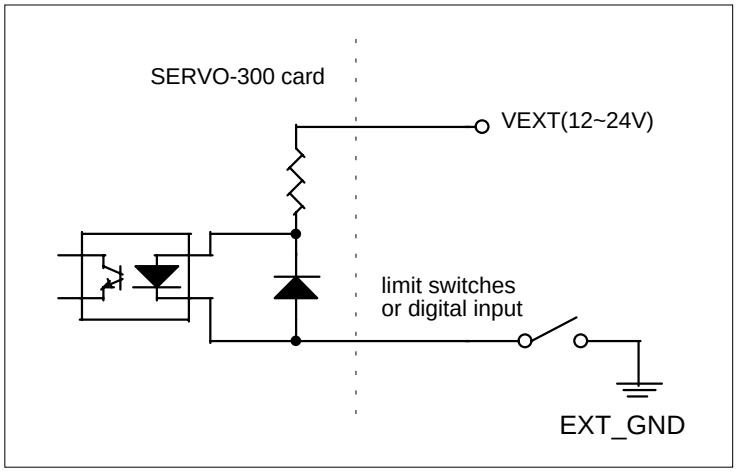
Table(3) CN3 connector

pin name	pin number	description
/XLS+	1	forward limit switch of X axis, low active
/XLS-	2	backward limit switch of X axis, low active
/YLS+	3	forward limit switch of Y axis, low active
/YLS-	4	backward limit switch of Y axis, low active
/XI	5	home index switch of X axis, low active
/YI	6	home index switch of Y axis, low active
/EMG	7	emergency input, low active
/IP1	8	digital input, low active
/IP2	9	digital input, low active
/IP3	10	digital input, low active
/IP4	11	digital input, low active
/IP5	12	digital input, low active
/IP6	13	digital input, low active
/IP7	14	digital input, low active
/IP8	15	digital input, low active
VEXT	16	external voltage (apply 12~24V)
/OP1	17	digital output, low active
/OP2	18	digital output, low active
/OP3	19	digital output, low active
/OP4	20	digital output, low active
/OP5	21	digital output, low active
/OP6	22	digital output, low active
/OP7	23	digital output, low active
	24	Reserved
EXT_GND	25	external ground

Table(4) CN4 connector

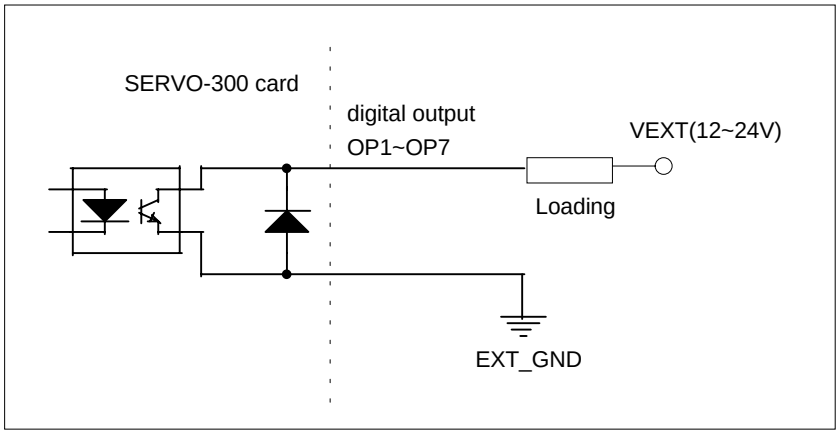
pin name	pin number	description
1A+	1	A+ input of X axis encoder
1A-	2	A- input of X axis encoder
1B+	3	B+ input of X axis encoder
1B-	4	B- input of X axis encoder
5V	5	encoder source (<50mA)
GND	6	encoder ground
1C+	7	C+ input of X axis encoder
1C-	8	C- input of X axis encoder
5V	9	encoder source (<50mA)
3A+	10	A+ input of Z axis encoder
3B+	11	B+ input of Z axis encoder
3C+	12	C+ input of Z axis encoder
	13	no used
2C-	14	C- input of Y axis encoder
2C+	15	C+ input of Y axis encoder
GND	16	encoder ground
5V	17	encoder source (<50mA)
2B-	18	B- input of Y axis encoder
2B+	19	B+ input of Y axis encoder
2A-	20	A- input of Y axis encoder
2A+	21	A+ input of Y axis encoder
GND	22	encoder ground
3A-	23	A- input of Z axis encoder
3B-	24	B- input of Z axis encoder
3C-	25	C- input of Z axis encoder

(2) The connection of limit switches and digital inputs



Figure(8)

(3) The connection of digital outputs



Figure(9)

(4) The connection of encoder

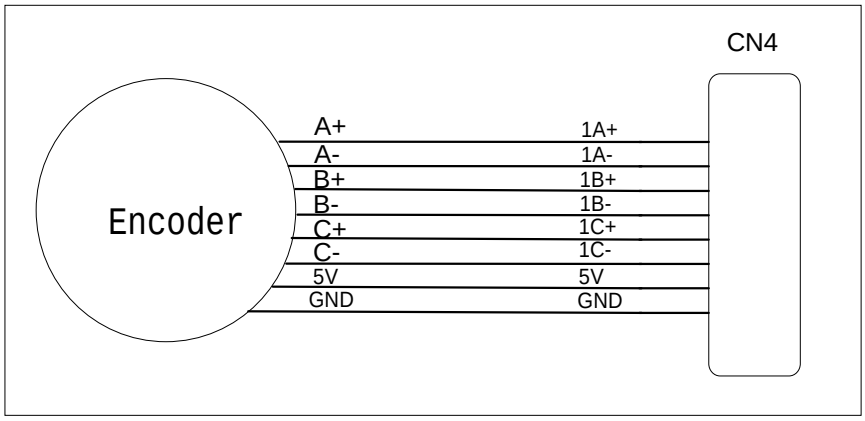


Figure (10) Connection between encoder and SERVO-300 card

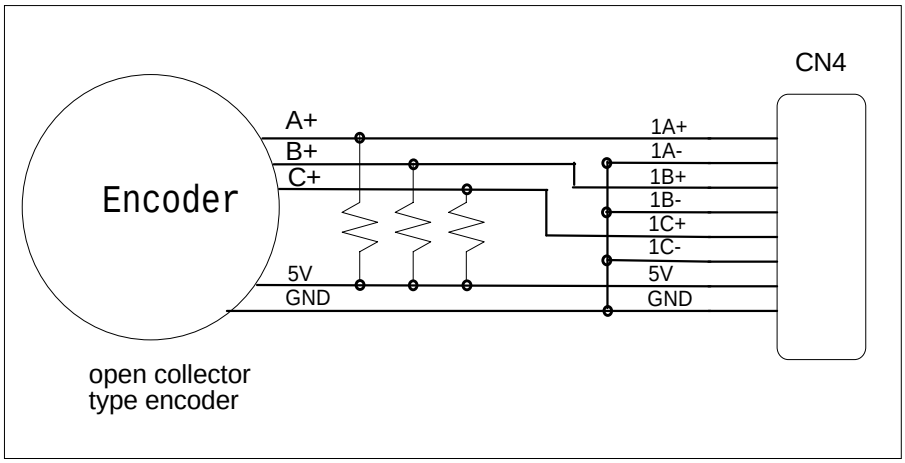
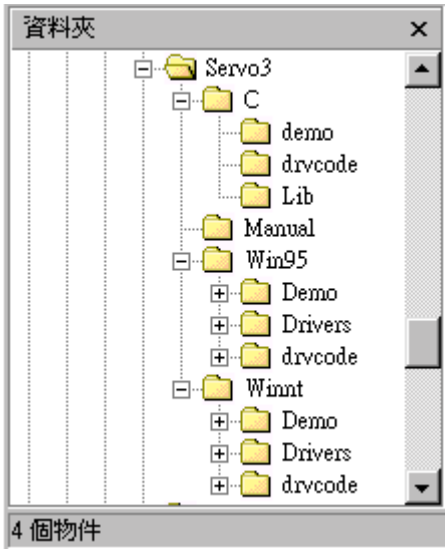


Figure (11) Connection of open-collector type encoder

3. Software

Directories



The software includes libraries and demonstrations of DOS(C,C++), windows 95 and windows NT.

3.1 The required software skeleton

To operate SERVO-300 board properly, the software require some process and a timer interrupt (10ms) to get the information transferred from SERVO-300 board.

The minimum software requirements:

(1) load VXD file (if windows application)
(2) SERVO-300 registration
(3) parameter setting
(4) motion command
(5) release VXD file (if windows application)
(6) 10ms timer interrupt

MSERVO3_INITIAL();	**(1). load VXD file**
exist=MSERVO3_REGISTRATION(CARD1,address);	**(2). registration** Select address and check it existed or not.
MSERVO3_RESET_SYSTEM(CARD1); MSERVO3_SET_CONTROL_MODE(CARD1,x_mode, y_mode, z_mode); MSERVO3_SET_VAR(CARD1, DDA, AD, LSP, HSP, arc_speed); MSERVO3_SET_DEFDIR(CARD1, x_dir, y_dir, z_dir); MSERVO3_SET_SERVO_ON(CARD1, x_son, y_son, z_son); MSERVO3_SET_CONFIG(CARD1, config, in_position, lag_error); MSERVO3_SET_X_CONTROLLER(CARD1, x_kp, x_kd); MSERVO3_SET_Y_CONTROLLER(CARD1, y_kp, y_kd); MSERVO3_SET_Z_CONTROLLER(CARD1, z_kp, z_kd);	**(3) parameter setting** reset SERVO-300 board set control mode set parameters of motion profile set direction set servo on set configuration and parameters set X axis controller set Y axis controller set Z axis controller
MSERVO3_BACK_HOME(CARD1, X_axis, home_speed, search_speed); MSERVO3_PULSE_MOVE(CARD1, X_axis, 50000, 1000); **...**	**(4) motion commands**
MSERVO3_RESET_SYSTEM(CARD1); MSERVO3_END();	reset SERVO-300 board **(5) release VXD file**

void __fastcall TMSERVO::Timer1Timer(TObject *Sender)
{
char str[20];
Timer1->Interval = 10; //10ms
 	
if (exist==YES)
{
card1.ip = MSERVO3_DI(CARD1);
card1.msc= MSERVO3_MSC(CARD1);
 	
MSERVO3_GET_CARD(CARD1);
 	
card1.sys =MSERVO3_GET_SYS(CARD1);
card1.ls =MSERVO3_GET_LIMIT(CARD1);
card1.p1 =MSERVO3_GET_P1(CARD1);
card1.XC =MSERVO3_GET_XC(CARD1);
card1.XP =MSERVO3_GET_XP(CARD1);
card1.YC =MSERVO3_GET_YC(CARD1);
card1.YP =MSERVO3_GET_YP(CARD1);
card1.ZC =MSERVO3_GET_ZC(CARD1);
card1.ZP =MSERVO3_GET_ZP(CARD1);
}
}
	(6) 10ms timer interrupt
	(demonstration for BCB)
	
	get digital input
	get limit switches
	
	get information from SERVO-300 board
	
	get X axis command
	get X axis position
	get Y axis command
	get Y axis position
	get Z axis command
	get Z axis position

3.2 Functions

Constants

```
#define YES    1
#define NO     0
#define ON     1
#define OFF    0
#define NORMAL_DIR   0
#define REVERSE_DIR  1
#define FW     0
#define BW     1
#define CW     0
#define CCW    1
#define X_axis   1
#define Y_axis   2
#define Z_axis   3
#define XY_plane  1
#define XZ_plane  2
#define YZ_plane  3
#define ENABLE_X    0x01
#define ENABLE_Y    0x02
#define ENABLE_Z    0x04
#define SIMU_MODE       0
#define TEST_MODE       1
#define CLOSE_MODE    2

#define READY 0
#define BUSY  1
```

3.2.1 Loading and unloading driver commands (only for windows)

(1) MSERVO3_INITIAL( )

To load VxD driver.

(2) MSERVO3_END( )

To release VxD driver.

3.2.2 Setting commands

(3) unsigned char MSERVO3_REGISTRATION(unsigned char cardNo, unsigned int address);

To select the address of board and check it exist or not. Ten SERVO-300 boards can be added in one system.

cardNo : board number 0~9.

address : select the address as well as hardware selected in chapter 2.

return NO : board not exist

YES : board exist

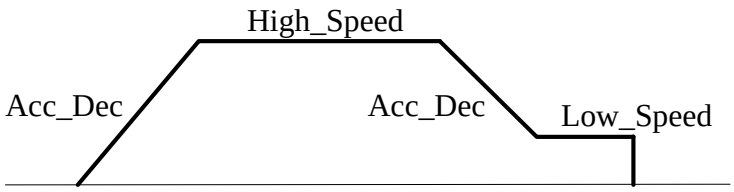
(4) MSERVO3_RESET_SYSTEM(unsigned char cardNo);

To reset SERVO-300 board.

cardNo : board number 0~9.

(5) MSERVO3_SET_VAR(unsigned char cardNo,
unsigned char set_DDA_cycle,
unsigned char set_Acc_Dec,
unsigned int set_Low_Speed,
unsigned int set_High_Speed,
unsigned int set_arc_speed);

cardNo : board number 0~9.
set_DDA_cycle : software DDA cycle.
set_Acc_Dec : accelerating/decelerating speed.
set_Low_speed : set end speed.
set_High_speed : set maximum speed.
set_arc_speed : set curve speed.



Restriction:

$1 \le DDA\_cycle \le 254$
$1 \le Acc\_Dec \le 200$
$1 \le Low\_Speed \le 200$
$Low\_Speed \le High\_Speed \le 2047$
$Arc\_Speed \le 2047$
default value
DDA_cycle = 8
Acc_Dec = 3
Low_Speed = 10
High_Speed = 500
Arc_Speed = 100

(6) MSERVO3_SET_DEFDIR(unsigned char cardNo,
unsigned char defdirX,
unsigned char defdirY,
unsigned char defdirZ);

Sometimes, the output direction of X-axis, Y-axis, Z-axis is not in desired direction due to motor connection or gear train. It is recommended to unify the output direction as shown in Figure(4)(5)(6). The CW/FW direction is defined as toward outside from motor and the CCW/BW direction is defined as toward inside to motor. MSERVO3_SET_DEFDIR( ) command provide parameters to define the rotating direction of motor.

cardNo : board number 0~9.
defdirX : X axis direction definition
defdirY : Y axis direction definition
defdirZ : Z axis direction definition
0 : NORMAL_DIR
1 : REVERSE_DIR

(7) MSERVO3_SET_CONTROL_MODE(unsigned char cardNo,
unsigned char x_mode,
unsigned char y_mode,
unsigned char z_mode);

To configure axis as simulation or open loop or close loop mode that described in chapter 2.
cardNo : board number 0~9.
x_mode : x axis control mode
y_mode : y axis control mode
z_mode : z axis control mode
where control mode 0 :SIMU_MODE (simulation mode)
1 :TEST_MODE (open loop mode)
2 :CLOSE_MODE (close loop mode)

(8) MSERVO3_SET_CONFIG(unsigned char cardNo,
unsigned char config,
unsigned char in_position,
unsigned int lag_error,
unsigned int error_range)

cardNo : board number 0~9.

config : set axis as enable or disable.

MSB 7	6	5	4	3	2	1	0 LSB
xx	xx	xx	xx	xx	Z enable	Y Enable	X enable

where X_enable, Y_enable, Z_enable

1 : enable

0 : disable

in_position (default 10) : define the maximum steady state error. When the position error is below this value and the motion command is completely executed, the SERVO-300 board will execute the next command.

lag_error (default 500) : define the maximum position error. When the position error is over this value, the SERVO-300 board will treat it as hardware failure.

error_range (default 500) : define the hardware failure detection range which means if the position error (command-position) > error_range and the speed is zero, it might be hardware failure, sleep or dis-connect. In this case, the X_ERR(Y_ERR, Z_ERR) and Error in sys register will indicate error.

(9) MSERVO3_SET_X_CONTROLLER(unsigned char cardNo,
unsigned int Kp,
unsigned int Kd);

cardNo : board number 0~9.

Kp : proportional control gain 0 < Kp < 65536

Kd : differential control gain 0<= Kd < 65536

(10) MSERVO3_SET_Y_CONTROLLER(unsigned char cardNo,
unsigned int Kp,
unsigned int Kd);

cardNo : board number 0~9.
Kp : proportional control gain 0 < Kp < 65536
Kd : differential control gain 0<= Kd < 65536

(11) MSERVO3_SET_Z_CONTROLLER(unsigned char cardNo,
unsigned int Kp,
unsigned int Kd);

cardNo : board number 0~9.
Kp : proportional control gain 0 < Kp < 65536
Kd : differential control gain 0<= Kd < 65536

(12) MSERVO3_SET_SERVO_ON(unsigned char cardNo,
unsigned char sonX,
unsigned char sonY,
unsigned char sonZ);

cardNo : board number 0~9.
sonX, sonY, sonZ : to turn servo signal ON/OFF
0 : servo off
1 : servo on

(13) MSERVO3_SET_ZERO(unsigned char cardNo, unsigned char axis);

to force set-position as zero.
cardNo : board number 0~9.
axis : can be X_axis, Y_axis or Z_axis.

(14) MSERVO3_PRESET_POSITION(unsigned char cardNo, unsigned char axis, long preset_position);

to pre-set the position in the SERVO-300 card.
cardNo : board number 0~9.
axis : can be X_axis, Y_axis or Z_axis.
preset_position : the desired pre-set position.

(15) MSERVO3_SET_PE(unsigned char cardNo,
unsigned char axis,
unsigned int index,
char p1,
char p2,
char p3,
char p4,
char p5);

The pitch error compensater of Servo-300 has a pitch error table. This table contains 2500 values, where the pitch is 256 pulse. So the maximum compensation range is
2500*256= 640000pulse.
cardNo : board number 0~9.
axis : selected axis.
index : the target offset of pitch error table.
p1,p2,p3,p4,p5 : to be filled value (-128~127)

3.2.3 Stop Commands

(16) MSERVO3_STOP(unsigned char cardNo, unsigned char axis);

To stop the motion command of selected axis

cardNo : board number 0~9.

axis : selected axis

(17) MSERVO3_DEC_STOP(unsigned char cardNo, unsigned char axis);

Decelerating to stop the selected axis's motor.

cardNo : board number 0~9.

axis : selected axis

(18) MSERVO3_STOP_ALL(unsigned char cardNo);

To stop motion command immediately, its function is the same as emergency stop by hardware.

cardNo : board number 0~9.

This command will clear the pending commands of the buffer, and terminate immediately all commands being executed in SERVO-300 board.

(19) MSERVO3_EMG_STOP(unsigned char cardNo);

This function is the same as MSERVO3_STOP_ALL(), but MSERVO3_EMG_STOP() only can be used in timer interrupt routine.

cardNo : card number 0~15.

This command will clear all of pending commands in the buffer, and immediately terminate all commands which is executing in SERVO-300 board.

3.2.4 Motion commands

(20) MSERVO3_BACK_HOME(unsigned char cardNo,
unsigned char axis,
unsigned char set_home_speed,
unsigned char set_search_speed);

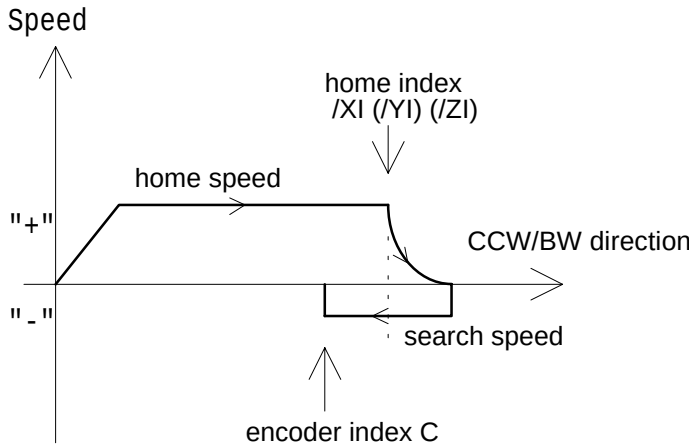
This command is used to move the motor toward CCW/BW direction at home speed at the beginning and then stop when home index switch /XI (/YI) (/ZI) is touched. Secondly, the motor will move toward CW/FW direction at search speed to find absolute zero. When /XI=1 and C=0, the motor stop and set position to zero. In general, the search speed should be set to 2~5. If the search speed is too large, the absolute point might be lost or lost accuracy. If search speed is set too small, it spends lots time.

cardNo : board number 0~9.

axis : selected axis.

0 < set_home_speed < 50

0 < set_search speed < 10



(21) MSERVO3_PULSE_MOVE(unsigned char cardNo,
unsigned char axis,
long pulseN,
unsigned int move_speed);

cardNo : board number 0~9.

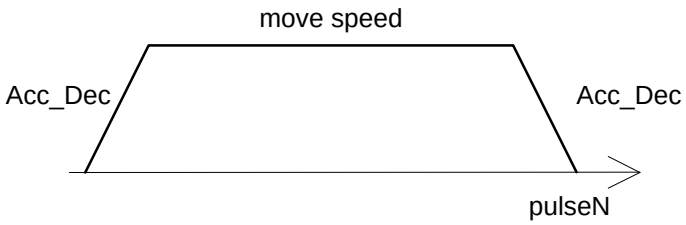
axis : selected axis.

pulseN : the distance to be moved.

when pulseN>0, move toward CW/FW direction

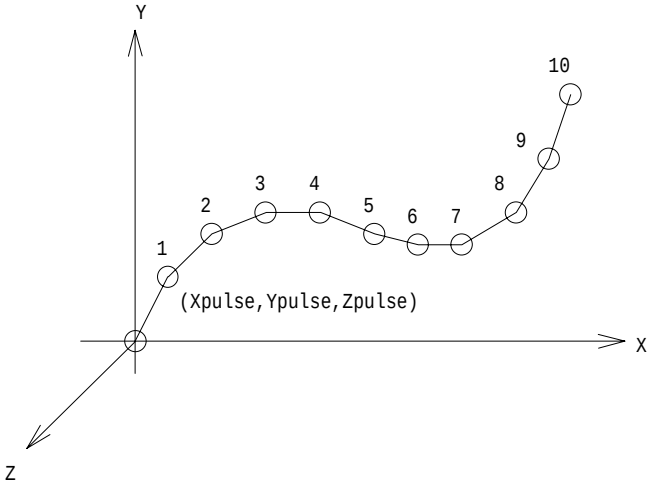
when pulseN<0, move toward CCW/BW direction

0 < move_speed <= 2040



(22) MSERVO3_INTP_PULSE(unsigned char cardNo,
int Xpulse,
int Ypulse,
int Zpulse);

This command will move a short distance (interpolation short line) in X-Y-Z space. This command supports user to generate an arbitrary space curve in X-Y-Z space.
cardNo : board number 0~9.
-2040 <= Xpulse, Ypulse, Zpulse <= 2040



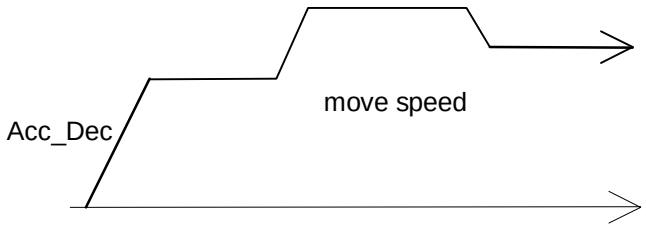
Example:

```
#define CARD1   1
  MSERVO3_INTP_PULSE(CARD1,20,20,2);
  MSERVO3_INTP_PULSE(CARD1,20,13,10);
  MSERVO3_INTP_PULSE(CARD1,20,7,10);
  MSERVO3_INTP_PULSE(CARD1,20,0,5);
  MSERVO3_INTP_PULSE(CARD1,15,-5,5);
```

(23)MSERVO3_CONSTANT_SPEED(unsigned char cardNo,
unsigned char axis,
unsigned char dir,
unsigned int move_speed);

This command will accelerate/decelerate the selected axis's motor to the "move_speed". This command can be continuously send to SERVO-300 to dynamicly change speed. The rotating motor can be stop by the command MSERVO3_STOP() or MSERVO3_DEC_STOP().

cardNo : board number 0~9.
axis : selected axis.
1 : X axis
2 : Y axis
3 : Z axis
dir : moving direction.
0 : CW
1 : CCW
0 < move_speed <= 2040

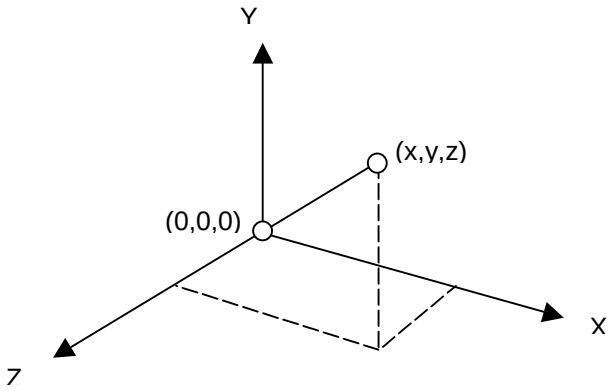


(24) MSERVO3_INTP_XYZ(unsigned char cardNo,
long x, long y, long z,
unsigned int speed);

This command will move a long distance interpolation line in X-Y-Z plane.
cardNo : board number 0~9.

$-2^{31}+1 \le \#x \le 2^{31}-1$
$-2^{31}+1 \le \#y \le 2^{31}-1$
$-2^{31}+1 \le \#z \le 2^{31}-1$

0 < speed <= 2040

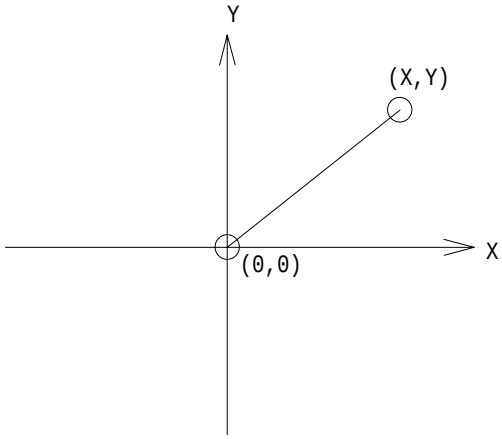


Example:

MSERVO3_INTP_XYZ(CARD1,2000,-3000,3333,1000);
MSERVO3_INTP_XYZ(CARD1,-500,200,200,500);

(25) MSERVO3_INTP_LINE(unsigned char cardNo,
long x,
long y,
unsigned int speed);

This command will move a long distance interpolation line in X-Y plane.
cardNo : board number 0~9.
$-2^{31}+1 \le \#x \le 2^{31}-1$
$-2^{31}+1 \le \#y \le 2^{31}-1$
0 < speed <= 2040



Example:
MSERVO3_INTP_LINE(CARD1,2000,-3000,1000);
MSERVO3_INTP_LINE(CARD1,-500,200,1000);

(26) MSERVO3_INTP_LINE01(unsigned char cardNo,
unsigned char plane,
long x,
long y,
unsigned int speed);

This command will move a long distance interpolation line in X-Y or X-Z or Y-Z plane.
plane :
1 : X-Y plane
2 : X-Z plane
3 : Y-Z plane

(27) MSERVO3_INTP_CIRCLE(unsigned char cardNo,
long x, long y,
unsigned char dir, unsigned int speed);

This command will generate an interpolation circle in X-Y plane. PC will automatically generate a trapezoidal speed profile of X-axis and Y-axis, and send these profile via of MSERVO3_INTP_PULSE( ) command.

cardNo : board number 0~9.

x, y : center point of circle related to present position.

dir : moving direction.

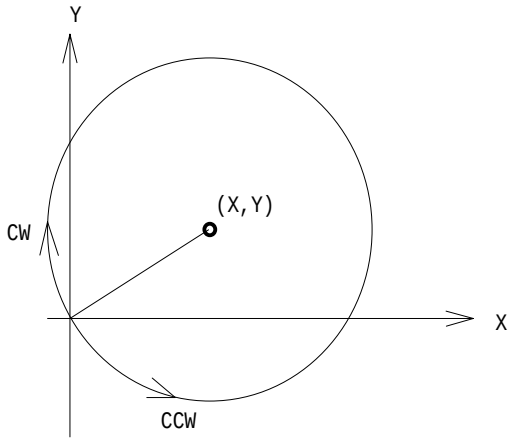
0 : CW

1 : CCW

$-2^{31}+1 \le \#x \le 2^{31}-1$

$-2^{31}+1 \le \#y \le 2^{31}-1$

0 < speed <= 2040



where radius = sqrt(X^2 + Y^2)

Example:

MSERVO3_INTP_CIRCLE(CARD1, 2000,-2000,CW,500);

(28) MSERVO3_INTP_CIRCLE01(unsigned char cardNo,
unsigned char plane, long x, long y,
unsigned char dir,
unsigned int speed);

This command will generate an interpolation circle in X-Y or X-Z or Y-Z plane.

plane :

1 : X-Y plane

2 : X-Z plane

3 : Y-Z plane

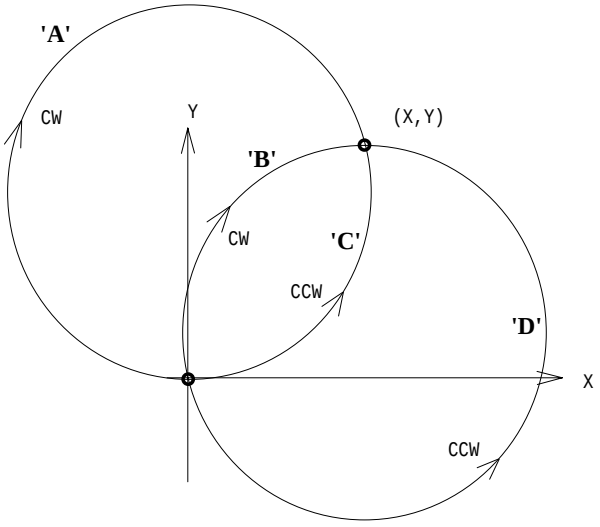
(29) MSERVO3_INTP_ARC(unsigned char cardNo,
long x, long y, long R,
unsigned char dir, unsigned int speed);

This command will generate an interpolation arc in X-Y plane. PC will automatically generate a trapezoidal speed profile of X-axis and Y-axis, and send these profile via MSERVO3_INTP_PULSE( ) command.
cardNo : board number 0~9.

x, y : end point of arc related to present position.
R : radius of arc.
if R>0 , the arc < 180 degree
if R<0 , the arc > 180 degree
dir : moving direction.
0 : CW
1 : CCW

R	dir	path of curve
R>0	CW	'B'
R>0	CCW	'C'
R<0	CW	'A'
R<0	CCW	'D'

0 < speed <= 2040



Restriction:

$$-2^{31}+1 \le \#x \le 2^{31}-1$$
$$-2^{31}+1 \le \#y \le 2^{31}-1$$
$$-2^{31}+1 \le \#R \le 2^{31}-1$$
$$R \ge \frac{\sqrt{x^2+y^2}}{2}$$

Example:

MSERVO3_INTP_ARC(CARD1,2000,-2000,2000,CW,500);

(30) MSERVO3_INTP_ARC01(unsigned char cardNo,
unsigned char plane, long x, long y,
long R, unsigned char dir,
unsigned int speed);

This command will generate an interpolation arc in X-Y or X-Z or Y-Z plane.

plane :

1 : X-Y plane
2 : X-Z plane
3 : Y-Z plane

3.2.5 Get information

To get the information transferred from SERVO-300 board, the user should construct a timer interrupt to poll information. The software skeleton has been described in chapter 3.1.

The MSERVO3_GET_CARD() command should be executed at the beginning and then to get information you need.

example:

```
void __fastcall TMSERVO::Timer1Timer(TObject *Sender)
{
char str[20];
Timer1->Interval = 10;    //10ms

if (exist==YES)
 {
   card1.ip = MSERVO3_DI(CARD1);
   card1.msc= MSERVO3_MSC(CARD1);

   MSERVO3_GET_CARD(CARD1);

   card1.sys   =MSERVO3_GET_SYS(CARD1);
   card1.ls    =MSERVO3_GET_LIMIT(CARD1);
   card1.p1    =MSERVO3_GET_P1(CARD1);
   card1.XC    =MSERVO3_GET_XC(CARD1);
   card1.XP    =MSERVO3_GET_XP(CARD1);
   card1.YC    =MSERVO3_GET_YC(CARD1);
   card1.YP    =MSERVO3_GET_YP(CARD1);
   card1.ZC    =MSERVO3_GET_ZC(CARD1);
   card1.ZP    =MSERVO3_GET_ZP(CARD1);
 }
}
```

(31) MSERVO3_GET_CARD(unsigned char cardNo)

This command uses timer interrupt to poll the information transferred from SERVO-300 board.

CardNo : board number 0~9.

(32) unsigned char MSERVO3_GET_SYS(unsigned char cardNo)

The sys register contains

MSB 7	6	5	4	3	2	1	0 LSB
Error	xx	xx	xx	xx	Z_ERR	Y_ERR	X_ERR

Error : indicate hardware failure error

X_ERR : indicate X axis hardware failure error

Y_ERR : indicate Y axis hardware failure error

Z_ERR : indicate Z axis hardware failure error

(33) unsigned char MSERVO3_GET_LIMIT(unsigned char cardNo)

The limit register contains

MSB 7	6	5	4	3	2	1	0 LSB
xx	xx	/ZLS-	/ZLS+	/YLS-	/YLS+	/XLS-	/XLS+

(34) unsigned char MSERVO3_GET_P1(unsigned char cardNo)

The P1 register contains

MSB 7	6	5	4	3	2	1	0 LSB
xx	C3	C2	C1	/ZI	/YI	/XI	/EMG

/EMG : emergency input, low active.

/XI, /YI, /ZI : indicate home index switch, low active.

C1, C2, C3 indicate the encoder index of X, Y, Z axis, respectively.
high active

(35) long MSERVO3_GET_XC(unsigned char cardNo)

Get the command position of X axis.

CardNo : board number 0~9.

(36) long MSERVO3_GET_XP(unsigned char cardNo)

Get the actual position of X axis.

CardNo : board number 0~9.

(37) long MSERVO3_GET_YC(unsigned char cardNo)

Get the command position of Y axis .

CardNo : board number 0~9.

(38) long MSERVO3_GET_YP(unsigned char cardNo)

Get the actual position of Y axis .

CardNo : board number 0~9.

(39) long MSERVO3_GET_ZC(unsigned char cardNo)

Get the command position of Z axis.

CardNo : board number 0~9.

(40) long MSERVO3_GET_ZP(unsigned char cardNo)

Get the actual position of Z axis

CardNo : board number 0~9.

3.2.6 Others

(41) unsigned char MSERVO3_DI(unsigned char cardNo)

To get the DI register

MSB 7	6	5	4	3	2	1	0 LSB
di7	di6	di5	di4	di3	di2	di1	di0

(42) MSERVO3_DO(unsigned char cardNo, unsigned char value)

To output DO port

MSB 7	6	5	4	3	2	1	0 LSB
don't use	DO6	DO5	DO4	DO3	DO2	DO1	DO0

(43) unsigned char MSERVO3_MSC(unsigned char cardNo)

To get the status of limit switch

MSB 7	6	5	4	3	2	1	0 LSB
/EMG	/Zstop	/Ystop	/Xstop	xx	xx	xx	xx

/Xstop, /Ystop, /Zstop : indicates which axis is stop, low active

/EMG : emergency switch, low active.

(44) MSERVO3_CALV(unsigned char cardNo,
unsigned char axis,
int value);

When this command is executed, the related axis goes to the open loop mode(TEST_MODE). The SERVO-300 board will directly output a constant voltage by D/A converter.
cardNo : board number 0~9.
axis : selected axis.
-2047(-10V) <= value <= 2047(10V)

(45) MSERVO3_WAIT_X(unsigned char cardNo)

To wait X-axis goes to the STOP state.

(46) MSERVO3_WAIT_Y(unsigned char cardNo)

To wait Y-axis goes to the STOP state.

(47) MSERVO3_WAIT_Z(unsigned char cardNo)

To wait Z-axis goes to the STOP state.

(48) unsigned char MSERVO3_IS_X_STOP(unsigned char cardNo)

To check whether X axis is STOP or not.
Return value 0 (NO) : not yet stop
1 (YES) : stop

(49) unsigned char MSERVO3_IS_Y_STOP(unsigned char cardNo)

To check whether Y axis is STOP or not.
Return value 0 (NO) : not yet stop
1 (YES) : stop

(50) unsigned char MSERVO3_IS_Z_STOP(unsigned char cardNo)

To check whether Z axis is STOP or not.
Return value 0 (NO) : not yet stop
1 (YES) : stop

(51) MSERVO3_SET_NC(unsigned char cardNo, unsigned char sw);

To set all of the following limit switches as N.C.(normal close) or N.O.(normal open). If set as N.O., those limit switches are active low. If

set as N.C., those limit switches are active high. The auto-protection will automatically change the judgement whatever it is N.O. or N.C..

Limit switches: XLS+, XLS-, YLS+, YLS-, ZLS+, ZLS-, EMG, XI, YI and ZI.

cardNo : card number 0~15.

sw: 0(NO) normal open (default).

1(YES) normal close.

3.2.7 New Interpolation command

The new driver provide a set of state-machine-type interpolation command including:

(52) MSERVO3_INTP_XYZ02(unsigned char cardNo,
long x, long y, long z,
unsigned int speed,
unsigned char acc_mode);

(53) MSERVO3_INTP_LINE02(unsigned char cardNo,
unsigned char plane,
long x,
long y,
unsigned int speed,
unsigned char acc_mode);

(54) MSERVO3_INTP_CIRCLE02(unsigned char cardNo,
unsigned char plane,
long x, long y,
unsigned char dir,
unsigned int speed,
unsigned char acc_mode);

(55) MSERVO3_INTP_ARC02(unsigned char cardNo,
unsigned char plane,
long x, long y, long R,
unsigned char dir,
unsigned int speed,
unsigned char acc_mode);

acc_mode: 0: enable acceleration and deceleration profile
1: disable acceleration and deceleration profile

These command can be set acc_mode=1 to disable the acceleration and deceleration profile.

(56) unsigned char MSERVO3_INTP_STOP()

These command is to compute the interpolation service. It will return READY(0) for interpolation command completed. And retrun BUSY(1) for not yet complete.

These 4 commands are state machine type command, they are only set parameters into the driver. The computing entity is in **MSERVO3_GET_CARD()** (only for windows) and **MSERVO3_INTP_STOP().** In windows application, when The MSERVO3_GET_CARD() command is

running in the timer interrupt routine by 10ms, it will help to calculate the interpolation service.
Both of DOS and windows application, User can directly call the **do {} while (MSERVO3_INTP_STOP()!=READY)** to execute the computing entity. The user can monitor something or waiting for keyboard input in the do loop. Therefore, The user has chance to do the software stop or monitor something.

DOS application example1

```
MSERVO3_INTP_XYZ02(CARD1,1000,1000,0,20,1);
do
{
   show_panel();
   if (kbhit())  chkey=bioskey(0);   //F7=0x4100
} while (   (chkey!= 0x4100)  &&  (MSERVO3_INTP_STOP()!=READY) );
if (chkey==0x4100) MSERVO3_STOP_ALL(CARD1);
```

DOS application example2

```
void TimerInterrupt(void)
{
    MSERVO3_GET_CARD(CARD1);
    show_panel();
    if (kbhit())       chkey=bioskey(0);   //F7=0x4100
}
 void test_intp(void)
 {
    MSERVO3_INTP_XYZ02(CARD1,1000,1000,0,20,1);
    do
    { } while (   (chkey!= 0x4100)  &&  (MSERVO3_INTP_STOP()!=READY) );
     if (chkey==0x4100) MSERVO3_STOP_ALL(CARD1);
}
```

Windows application example1

```
void __fastcall TMSERVO::Timer1Timer(TObject *Sender)
{
    Timer1->Interval = 10; //10ms
    MSERVO3_GET_CARD(CARD1);
    show_panel();
}
```

```
void __fastcall TMSERVO::IntpLineClick(TObject *Sender)
{
char str[20];

  if (    (MSERVO3_IS_X_STOP(CARD1)==NO)
       || (MSERVO3_IS_Y_STOP(CARD1)==NO)
       || (MSERVO3_IS_Z_STOP(CARD1)==NO))
  {
    Application->MessageBox(
      "Motor's rotating, can't execute this command",
      "Message Box",
      MB_DEFBUTTON1);
    return;
  };

  ltoa(x, str, 10);
  IntpLineDialog->Xpulse->Text = AnsiString(str);
  ltoa(y, str, 10);
  IntpLineDialog->Ypulse->Text = AnsiString(str);
  ltoa(speed, str, 10);
  IntpLineDialog->speed->Text = AnsiString(str);
  IntpLineDialog->SelectPlane->ItemIndex = plane-1;

  if (IntpLineDialog->ShowModal()==mrOk)
  {
    x= (long)IntpLineDialog->Xpulse->Text.ToInt();
    y= (long)IntpLineDialog->Ypulse->Text.ToInt();
    speed= (unsigned int)IntpLineDialog->speed->Text.ToInt();
    plane= (unsigned char)(IntpLineDialog->SelectPlane->ItemIndex + 1);
      //MSERVO3_INTP_LINE01(CARD1,plane,x,y,speed);
    MSERVO3_INTP_LINE02(CARD1,plane,x,y,speed,0);
    do {Application->ProcessMessages();}
    while (MSERVO3_INTP_STOP()!=READY);
  }
}
```

The example for wait stop command (DOS):

```
//test XYZ01, no acceleration
MSERVO3_INTP_XYZ02(CARD1,1000,1000,0,20,1);
do {} while (MSERVO3_INTP_STOP()!=READY);
MSERVO3_INTP_XYZ02(CARD1,2000,1000,0,22,1);
do {} while (MSERVO3_INTP_STOP()!=READY);
MSERVO3_INTP_XYZ02(CARD1,2000,2000,0,24,1);
do {} while (MSERVO3_INTP_STOP()!=READY);
MSERVO3_INTP_XYZ02(CARD1,1000,2000,0,26,1);
do {} while (MSERVO3_INTP_STOP()!=READY);

do {} while (MSERVO3_IS_X_STOP(CARD1)==NO);
delay(10000);

//test WAIT_X, WAIT_Y, WAIT_Z
MSERVO3_INTP_LINE02(CARD1, XY_plane, 10000,-10000,200,0);
do {} while (MSERVO3_INTP_STOP()!=READY);
MSERVO3_INTP_LINE02(CARD1, XY_plane, -10000, 10000,200,0);
do {} while (MSERVO3_INTP_STOP()!=READY);
MSERVO3_INTP_XYZ02(CARD1, 5000, -10000, -40000, 200, 0);
do {} while (MSERVO3_INTP_STOP()!=READY);

do {} while (MSERVO3_IS_X_STOP(CARD1)==NO);
do {} while (MSERVO3_IS_Y_STOP(CARD1)==NO);
do {} while (MSERVO3_IS_Z_STOP(CARD1)==NO);
MSERVO3_STOP_ALL(CARD1);
delay(10000);

MSERVO3_INTP_CIRCLE02(CARD1, XY_plane, 5000,-5000, CW, 200,
0);
do {} while (MSERVO3_INTP_STOP()!=READY);

do {} while (MSERVO3_IS_X_STOP(CARD1)==NO);
do {} while (MSERVO3_IS_Y_STOP(CARD1)==NO);
MSERVO3_STOP_ALL(CARD1);
```

4. Driver

DOS Driver (C, C++)

Item	File
Header file	mservo3.h
Library file	mservo3.lib
Example file	sv3tcc.prj(turbo C++)

Windows 95 Driver

Item	File
Header file	servo32.h
ImportLibrary file	servo32.lib bcservo.lib (only for Borland C++)
Dynamic Link Library	servo32.dll(copy to c:\windows)
VxD file	napdio.vxd(copy to c:\windows)
Example file	bcservo1.bpr(Borland C++ Builder) bcservo1.cpp main.cpp

Windows NT Driver

Item	File
Header file	servo32.h
ImportLibrary file	servo32.lib bcservo.lib (only for Borland C++)
Dynamic Link Library	servo32.dll
Driver	regdrv.bat napwnt.ini napwnt.sys regini.exe
Example file	bcservo1.bpr(Borland C++ Builder) bcservo1.cpp main.cpp

5. Example

5.1 DOS example

The execution file, SV3TCC.EXE/SV3BCC.EXE, is a testing program. It can let you fully understand the action of every command. The source files include SV3TCC.PRJ(SV3BCC.IDE), MAIN.CPP, MSERVO3.H and MSERVO3.LIB. The MAIN.CPP file provides several examples of MSERVO3 command set. If you have any questions about the command set, you can trace the MAIN.CPP source file.

The panel of SV3TCC.EXE has three areas :

(1) I/O information area

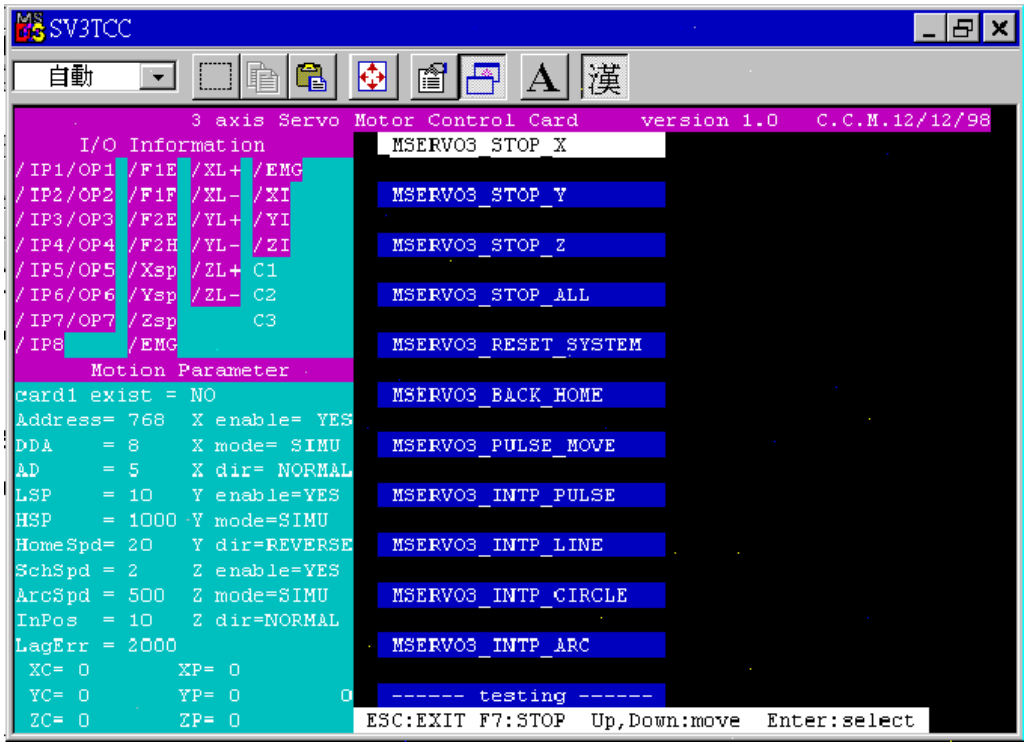
It indicates the status of limit switch, digital input/output and SERVO-300 board.

(2) motion parameter area

It shows the variables of motion parameter. It also shows the command position and actual position of each axis.

(3) Command area

You can select any command and execute it.



The panel of DOS example

5.2 Windows example

The bcservo1.exe (source file included) is an example of SERVO-300 board. It has windows95 and NT edition. If you have any question about SERVO-300 command set, you can trace the source file.

The panel of bcservo1.exe has three areas :

(1) I/O information area

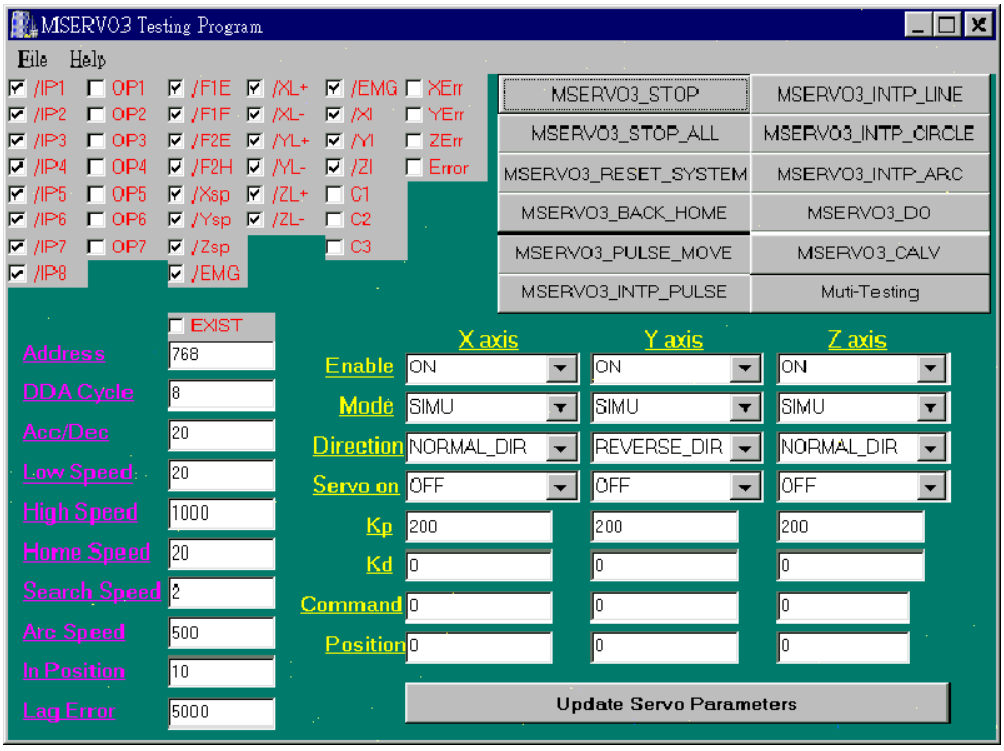
It indicates the status of limit switches, digital inputs/outputs and SERVO-300 board.

(2) Motion parameter area

It shows the variables of motion parameter. It also shows the command position and actual position of each axis. All parameters can be modified and updated by pressing the lower-right menu bar.

(3) Command area

You can select any command and execute it.

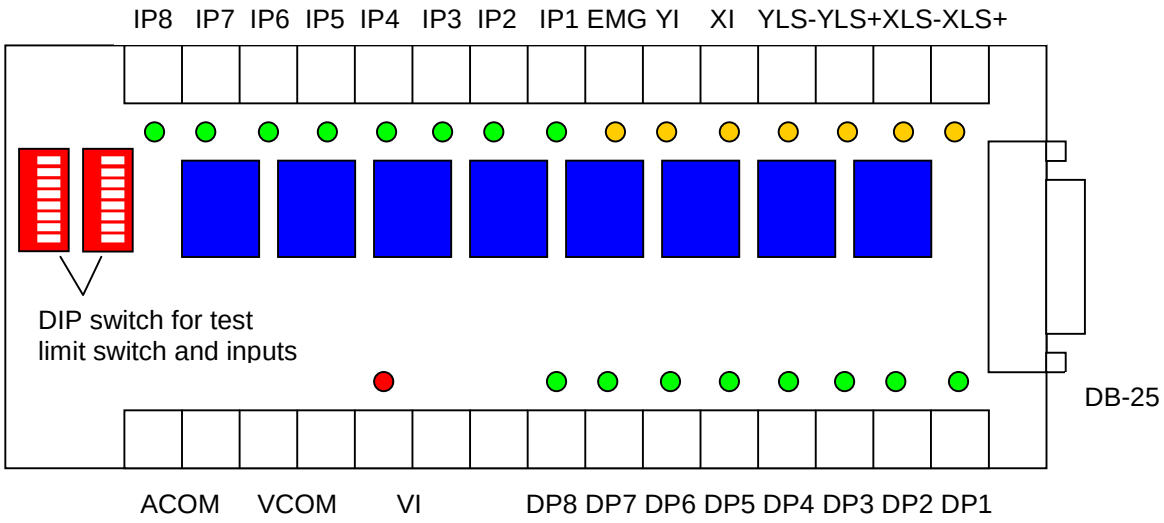


The panel of windows example

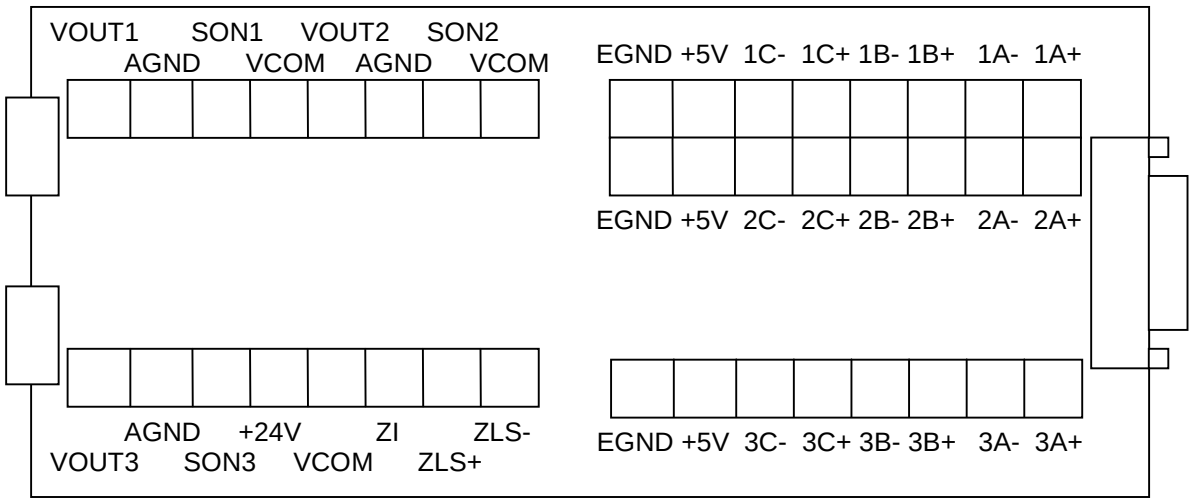
6. Application

The SERVO-300 can be applied in X-Y table control, robot, CNC PCB driller, CNC PCB router, CNC wire cutter, lathe and semiconductor equipment.

For easily set up a machine, there are two daughter boards DB-8R and DB-200 can be adopted. The DB-8R board is the connection board for limit switches, digital inputs/outputs. The DB-200 board is the connection board for servo driver that includes encoder signal A+,A-,B+,B-,C+,C- and voltage command. The following two figure is the outlook of DB-8R and DB-200.



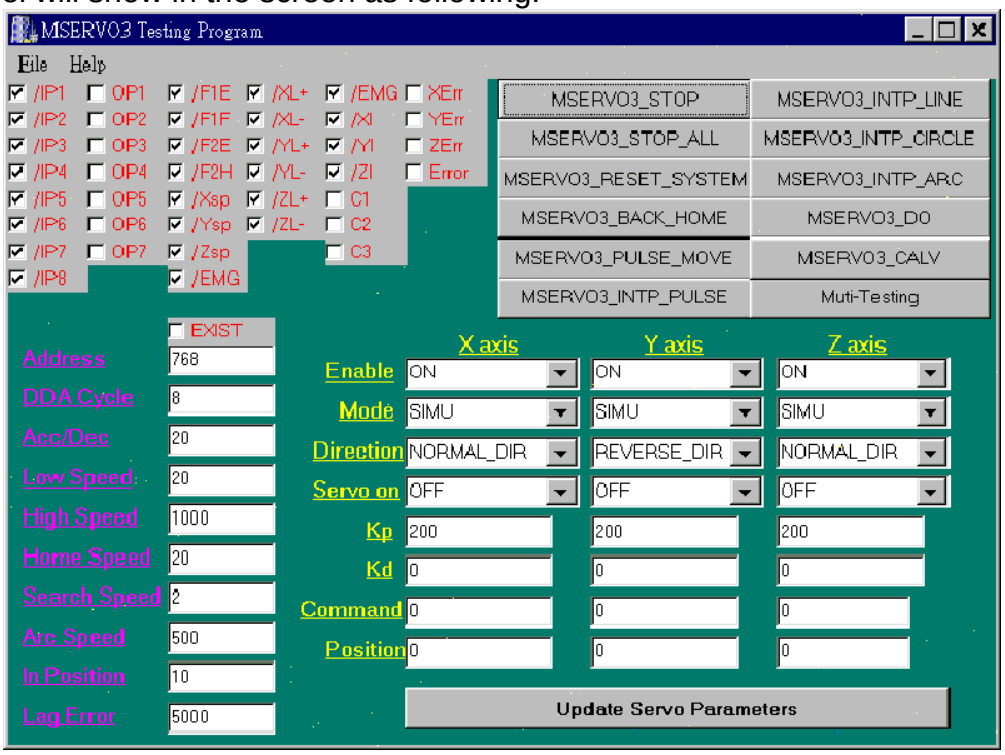
The DB-8R daughter board



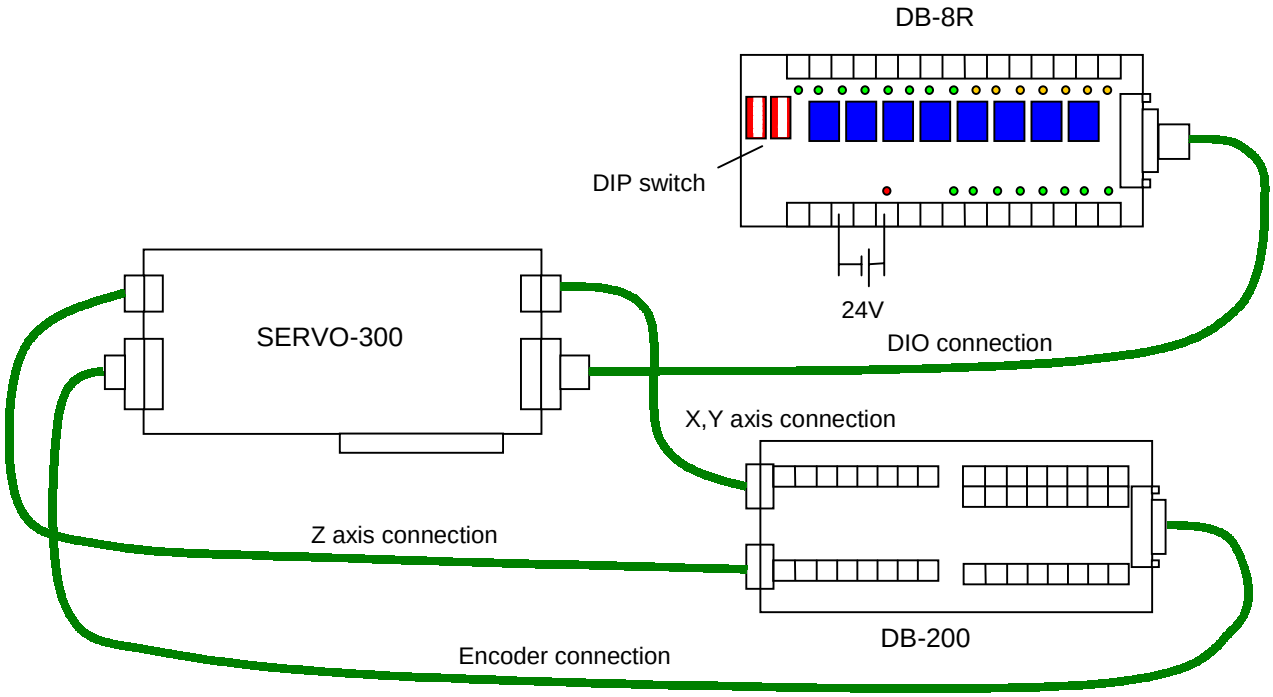
The DB-200 daughter board

6.1 Functional testing

If the user wants to verify the hardware and the function of SERVO-300, it can run the bcservo1.exe in windows95/98 or windows NT. The control panel will show in the screen as following.



For easily test, the hardware connect as the following diagram.



- Check the SERVO-300 card is existed or not
  The indicator “EXIST” will show the SERVO-300 card is existed or not. In case of card is not existed, all of the function will not able to be performed.

- Test digital inputs and limit switches
  First of all, the external power +24V shall be applied. While turn on the DIP switch on DB-8R daughter board, the corresponding LED on DB-8R and indicator in the control will light.

- Test digital outputs
  First of all, the external power +24V shall be applied. The user can press the button **MSERVO3_DO**, and turn on the DO then press OK button, the corresponding DO LED on DB-8R will light.

- Test encoder input
  The user can use a encoder or servo motor to test this item. It just connect the A+,A-,B+,B-,C+,C-,5V and GND to the daughter board DB-200, and then rotate the encoder or motor’s shaft by manual. The position will be shown on the panel. The index C also will be shown on panel when rotate slowly.

- Test analog output
  The test mode is open loop control that means output a constant voltage. The button **MSERVO3_CALV** provides test mode output function. The user can output the value between –2040(-9.96V) to 2040(+9.96V). The analog offset also can be calibrated by this mode when set the value as 0, then to tune the variable resister on the SERVO-300 board.

- Test servo on signal
  The servo ON/OFF switch right in the parameter area on the panel. The user can select ON or OFF, and press the update parameter button, the corresponding SON1-3 will act.

- Test motion command in simulation mode
  First of all, select the **SIMU** mode and then press the update parameter button. The user can execute the motion command such as **MSERVO3_PULSE_MOVE.** The simulated position will show on the control panel.