ADT-8948A1

4-axle Motion Controlling Card

Technical Manual

Shenzhen Adtech Digital Control Tech Co., Ltd.
Add：Building 36 Majialong Industrial Area Nanshan District,Shenzhen
Post Code: 518052
Tel：0755-26722719 （20 lines） Fax: 0755-26722718

Statement about the Copyright

Statement about the Trade Mark

Contents

Chapter I Introduction

☞About the product

ADT-8948A1 is a high performance 4-axis servo/stepping motion card based on PCI bus. One system can use up to 16 motion cards and control 64 servo/stepping motors. It supports PnP and the position has variable cyclic function. The speed and target position can be changed in real-time in the motion process. Advanced functions like continuous interpolation are available.

The pulse output mode is either signal pulse (pulse + direction) or double pulse (pulse + pulse). The maximum pulse frequency is 4MHz. With advanced technology, the frequency error is less than 0.1% even when the pulse output frequency is high.

The position is managed by two up/down counters. One is logical position counter used to manage inner pulse output and the other is used to receive outer pulse input. It is either A/B phase input signal of coder or grating scale, or input signal of up/down pulse. As actual position counter, its bits can reach 32 and the maximum range is -2,147,483,648~+2,147,483,647. The outer input can be used as handwheel input for common counting.

Provide servo interface signals, e.g. Z-phase signal of coder, in-position signal (INPOS), alarm signal (ALARM) and servo enable (SERVO ON).

Multiple control modes: external signal driving, position lock, automatically back to home, synchronized control, stepping interpolation, quantitative motion, continuous motion, home motion, multi-axis interpolation, arc interpolation, emergency stop. Interpolation is normally used for constant velocity motion or linear/S curve acceleration/deceleration (S curve acceleration/deceleration can't be used for arc interpolation).

External signal (handwheel or general input signal) driving can be either constant or continuous driving.

With position lock, you can lock the value of logical counter or actual position counter.

Automatically back to home can be in various modes. The home signals may be combined freely in certain mode. The modes can also be customized.

Synchronized control means that the motion axis acts in preset mode when the state of specified signal changes.

Steeping interpolation is to perform interpolation in single step, including command driving and external signal driving.

The interpolation is integrated with continuous interpolation function, i.e. input the interpolation data of next point in the interpolation process to ensure the continuity of pulse and optimize the performance of interpolation. The maximum interpolation speed is 2MHz.

The speed control can be constant linear speed (vector composite speed), constant velocity or linear/S-curve acceleration/deceleration. It may perform asymmetric linear acceleration/deceleration. Automatic deceleration and manual deceleration are optional. In constant driving process, it can prevent triangle wave caused by speed curve.

Each axis has two 32-bit compare registers, which are used for software limit.

Each axis has 8 input signals, including 2 positive/negative limit signals, 3 stop signals, 1 servo in-position signal, 1 servo alarm signal and 1 general input signal. Except limit signals, other signals can be used as general input signals by setting them as invalid. The 3 stop signals can be used to search for home signal, deceleration signal and Z-phase of coder. All digital input signals have integral filters. Eight filtering time constants are optional to avoid interference.

Provide DOS, WINDOWS95/98/NT/2000/XP and WINCE development libraries, and developing software with VC++, VB, BC++, C++builder, LabVIEW and Delphi is possible.

☞Features

- 32-bit PCI bus, PnP
- 4-axis servo/stepping motor control; each axis can control independently
- The frequency error of pulse input is less than 0.1%
- Maximum pulse output frequency is 4MHz
- Pulse output can be either single pulse (pulse + direction) or double pulse (pulse + pulse)
- All the 4 axes have position feedback input; 32-bit counting; maximum counting range: -2,147,483,648~+2,147,483,647
- Linear or S-curve acceleration/deceleration
- Asymmetric linear acceleration/deceleration
- Random 2-3 axes linear interpolation
- Random 2 axes arc interpolation
- Continuous interpolation is available; top driving speed: 2MHz
- Each axis has two 32-bit compare registers, which are used for position comparison between logical position counter and actual position counter, and software limit
- Receive signals from servo motor drive, e.g. coder Z-phase signal, in-position signal, alarm signal, etc
- Each axis has 3 STOP signals, which are used to search for home and Z-phase of coder
- The speed and target position can be changed in real-time in the motion process
- Read the logical position, real position, driving speed, acceleration and driving state in real-time in the motion process
- Position counter is integrated with variable cyclic function; the logical position counter and actual position counter are 32-bit up/down cyclic counters
- Each axis has 8-in/8-out digital I/O. Except two limit signals, all signals can be used as general I/O. Digital output can be used in signals of servo on and servo alarm reset
- The input port of every input signal is equipped with integral filter. You can select to activate/deactivate the filter of certain input signal. Select one from the 8 constants for

the filter time

- Constant linear speed mode is available
- Automatic back to home in various modes
- Position lock triggered by external signal
- Synchronized motion triggered by external signal
- Synchronized stop triggered by external signal
- Synchronized motion at specified position
- Synchronized stop at specified position
- Handwheel and external signal operation
- Stepping interpolation function
- Up to 16 motion cards can be used in one system

Supported operating systems: DOS, WINDOWS95/98/NT/2000/XP, WINCE

☞**Application**

⚐ **Multi-asix carving and milling system**
⚐ **Robot system**
⚐ **Coordinate measuring system**
⚐ **Numerical control system based on PC**

Chapter II Installing Hardware

☞Accessories

1．	ADT-8948A1	User Manual
2．	ADT-8948A1	4-axis PCI motion card
3．	ADT-8948A1	CD
4．	ADT-9162	2 terminal blocks
5．	D62GG	Two 62-core shielded cables
6．	DB64	One 64-core flat cable
7．	ADT_9164	One transition board

☞Install

1. Cut off the power supply of PC.
2. Open the back cover of computer case.
3. Select an idle PCI slot and insert the ADT-8948A1.
4. Make sure the golden finger of ADT-8948A1 is inserted into the slot completely and then fix the screw.
5. Connect one end of the D62GG cable to J1 interface of motion card and the other end to terminal block ADT_9162.
6. Check whether it is necessary to install J2 interface cable. To install J2 if necessary:

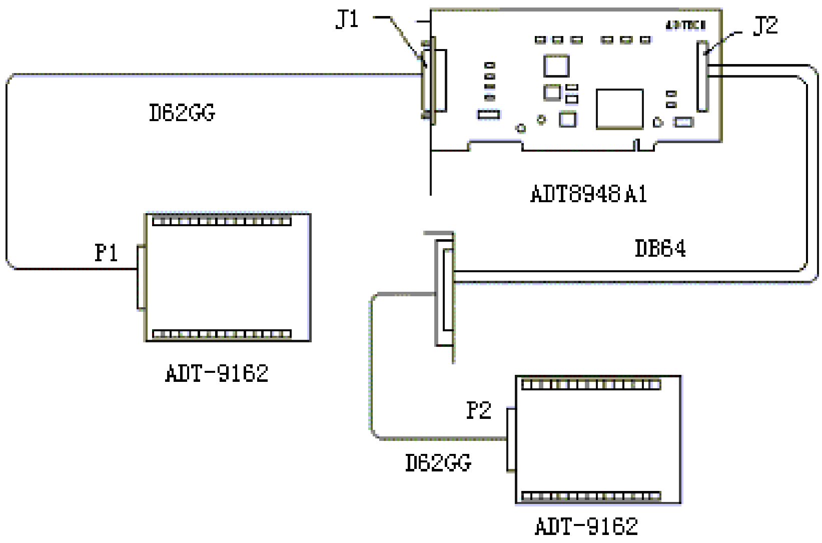
(1) Connect one end of DB64 to J2 of motion card and the other end to P2 of ADT_9164;

(2) Fix the ADT_9164 on the rear side of the enclosure;

(3) Connect D62GG to P2 of the transition board and ADT_9162.

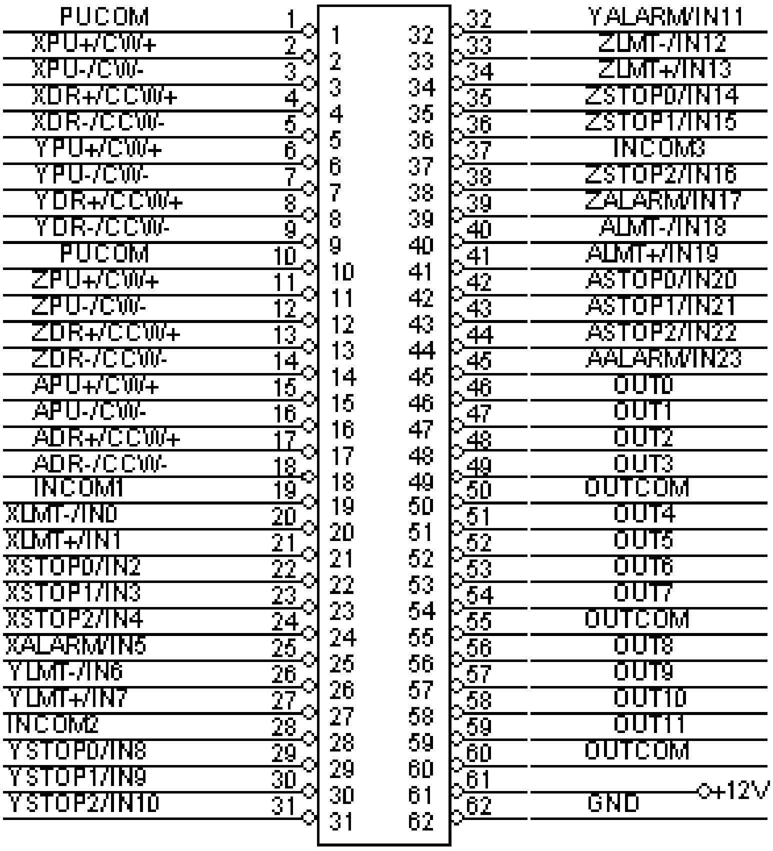
Chapter III Electric Connection

☞ Wiring Diagram



One ADT8948A1 card has two input/output interfaces: J1/P1, J2/P2 (62-pin socket).

☞J1 cable marker description

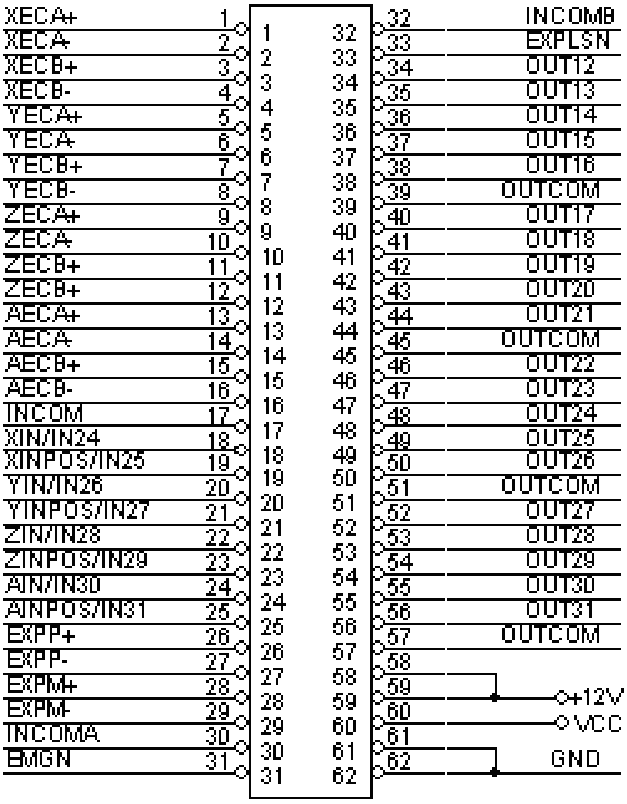


PUCOM	1	32	YALARM/IN11
XPU+/CW+	2	33	ZLMT-/IN12
XPU-/CW-	3	34	ZLMT+/IN13
XDR+/CCW+	4	35	ZSTOP0/IN14
XDR-/CCW-	5	36	ZSTOP1/IN15
YPU+/CW+	6	37	INCOM3
YPU-/CW-	7	38	ZSTOP2/IN16
YDR+/CCW+	8	39	ZALARM/IN17
YDR-/CCW-	9	40	ALMT-/IN18
PUCOM	10	41	ALMT+/IN19
ZPU+/CW+	11	42	ASTOP0/IN20
ZPU-/CW-	12	43	ASTOP1/IN21
ZDR+/CCW+	13	44	ASTOP2/IN22
ZDR-/CCW-	14	45	AALARM/IN23
APU+/CW+	15	46	OUT0
APU-/CW-	16	47	OUT1
ADR+/CCW+	17	48	OUT2
ADR-/CCW-	18	49	OUT3
INCOM1	19	50	OUTCOM
XLMT-/IN0	20	51	OUT4
XLMT+/IN1	21	52	OUT5
XSTOP0/IN2	22	53	OUT6
XSTOP1/IN3	23	54	OUT7
XSTOP2/IN4	24	55	OUTCOM
XALARM/IN5	25	56	OUT8
YLMT-/IN6	26	57	OUT9
YLMT+/IN7	27	58	OUT10
INCOM2	28	59	OUT11
YSTOP0/IN8	29	60	OUTCOM
YSTOP1/IN9	30	61	+12V
YSTOP2/IN10	31	62	GND

Cable marker	Symbol	Description
1	PUCOM	Positive port of internal +5V power supply; do not connect to external power supply; COM for common anode wiring
2	XPU+/CW+	X pulse signal +
3	XPU-/CW-	X pulse signal -
4	XDR+/CCW+	X direction signal +
5	XDR-/CCW-	X direction signal -
6	YPU+/CW+	Y pulse signal +
7	YPU-/CW-	Y pulse signal -
8	YDR+/CCW+	Y direction signal +
9	YDR-/CCW-	Y direction signal -
10	PUCOM	Positive port of internal +5V power supply; do not connect to external power supply; COM for common anode wiring
11	ZPU+/CW+	Z pulse signal +
12	ZPU-/CW-	Z pulse signal -
13	ZDR+/CCW+	Z direction signal +
14	ZDR-/CCW-	Z direction signal -
15	APU+/CW+	A pulse signal +
16	APU-/CW-	A pulse signal -
17	ADR+/CCW+	A direction signal +
18	ADR-/CCW-	A direction signal -
19	INCOM1	Common port of photoelectric coupling input (signals below), 12V-24V
20	XLMT-/IN0	X negative limit signal
21	XLMT+/IN1	X positive limit signal
22	XSTOP0/IN2	X origin signal 0; can be used as universal input signal
23	XSTOP1/IN3	X origin signal 1; can be used as universal input signal
24	XSTOP2/IN4	X origin signal 2; can be used as universal input signal
25	XALM/IN5	X servo alarm signal; can be used as universal input signal
26	YLMT-/IN6	Y negative limit signal
27	YLMT+/IN7	Y positive limit signal
28	INCOM2	Common port of photoelectric coupling input (signals below), 12V-24V
29	YSTOP0/IN8	Y origin signal 0; can be used as universal input signal
30	YSTOP1/IN9	Y origin signal 1; can be used as universal input signal
31	YSTOP2/IN10	Y origin signal 2; can be used as universal input signal
32	YALM/IN11	Y servo alarm signal; can be used as universal input signal
33	ZLMT-/IN12	Z negative limit signal
34	ZLMT+/IN13	Z positive limit signal
35	ZSTOP0/IN14	Z origin signal 0; can be used as universal input signal
36	ZSTOP1/IN15	Z origin signal 1; can be used as universal input signal
37	INCOM3	Common port of photoelectric coupling input (signals

		below), 12V-24V
38	ZSTOP2/IN16	Z origin signal 2; can be used as universal input signal
39	ZALM/IN17	Z servo alarm signal; can be used as universal input signal
40	ALMT-/IN18	A negative limit signal
41	ALMT+/IN19	A positive limit signal
42	ASTOP0/IN20	A origin signal 0; can be used as universal input signal
43	ASTOP1/IN21	A origin signal 1; can be used as universal input signal
44	ASTOP2/IN22	A origin signal 2; can be used as universal input signal
45	AALM/IN23	A servo alarm signal; can be used as universal input signal
46	OUT0	Digital output
47	OUT1	
48	OUT2	
49	OUT3	
50	OUTCOM	External power supply ground wire; common port of digital output OUT0-3
51	OUT4	Digital output
52	OUT5	
53	OUT6	
54	OUT7	
55	OUTCOM	External power supply ground wire; common port of digital output OUT4-7
56	OUT8	Digital output
57	OUT9	
58	OUT10	
59	OUT12	
60	OUTCOM	External power supply ground wire; common port of digital output OUT8-12
61	+12V	Positive port of internal +12V power supply; do not connect to external power supply
62	GND	Internal power supply ground wire

☞J2 cable marker description



Cable marker	Symbol	Description
1	XECA+	X-axis coder phase A input +
2	XECA-	X-axis coder phase A input -
3	XECB+	X-axis coder phase B input +
4	XECB-	X-axis coder phase B input -
5	YECA+	Y-axis coder phase A input +
6	YECA-	Y-axis coder phase A input -
7	YECB+	Y-axis coder phase B input+
8	YECB-	Y-axis coder phase B input -
9	ZECA+	Z-axis coder phase A input +
10	ZECA-	Z-axis coder phase A input -
11	ZECB+	Z-axis coder phase B input +
12	ZECB-	Z-axis coder phase B input -
13	AECA+	A-axis coder phase A input +
14	AECA-	A-axis coder phase A input -
15	AECB+	A-axis coder phase B input +
16	AECB-	A-axis coder phase B input -
17	INCOM	Common port of photoelectric coupling input (signals below), 12V-24V
18	XIN/IN24	X position lock signal; can be used as universal input signal
19	XINPOS/IN25	X servo in-position signal; can be used as universal input

		signal
20	YIN/IN26	Y position lock signal; can be used as universal input signal
21	YINPOS/IN27	Y servo in-position signal; can be used as universal input signal
22	ZIN/IN28	Z position lock signal; can be used as universal input signal
23	ZINPOS/IN29	Z servo in-position signal; can be used as universal input signal
24	AIN/IN30	A position lock signal; can be used as universal input signal
25	AINPOS/IN31	A servo in-position signal; can be used as universal input signal
26	EXPP+	Handwheel phase A input +, can be used as common port of universal input signal (IN32)
27	EXPP-	Handwheel phase A input -, can be used as universal input signal (IN32)
28	EXPM+	Handwheel phase B input +, can be used as common port of universal input signal (IN33)
29	EXPM-	Handwheel phase B input -, can be used as universal input signal (IN33)
30	INCOMA	Common port of photoelectric coupling input (signals below), 12V-24V
31	EMGN	Emergency stop input signal (IN34)
32	INCOMB	Common port of photoelectric coupling input (signals below), 12V-24V
33	EXPLSN	Stepping interpolation input signal
34	OUT12	Digital output
35	OUT13	
36	OUT14	
37	OUT15	
38	OUT16	
39	OUTCOM	External power supply ground wire; common port of digital output OUT12-16
40	OUT17	Digital output
41	OUT18	
42	OUT19	
43	OUT20	
44	OUT21	
45	OUTCOM	External power supply ground wire; common port of digital output OUT17-21
46	OUT22	Digital output
47	OUT23	
48	OUT24	
49	OUT25	
50	OUT26	
51	OUTCOM	External power supply ground wire; common port of digital

		output OUT22-26
52	OUT27	Digital output
53	OUT28	
54	OUT29	
55	OUT30	
56	OUT31	
57	OUTCOM	External power supply ground wire; common port of digital output OUT27-31
58	+12V	Potive port of internal +12V power supply; do not connect to external power supply
59	+12V	Potive port of internal +12V power supply; do not connect to external power supply
60	+5V	Potive port of internal +5V power supply; do not connect to external power supply
61	GND	Internal power supply ground wire
62	GND	Internal power supply ground wire

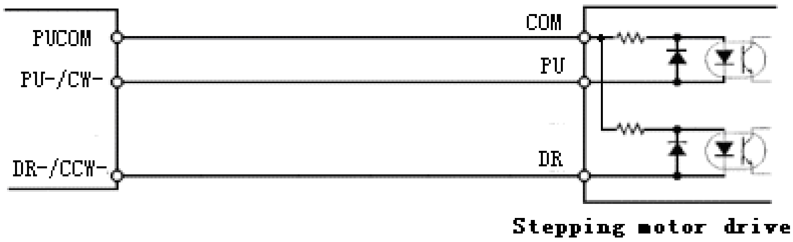
Note:

XIN, YIN, ZIN and AIN are control signals of position lock and synchronous control

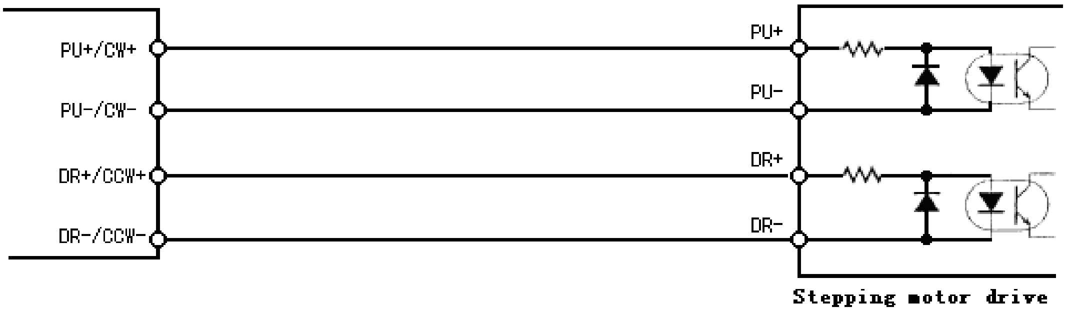
☞ Connection of pulse/direction output signals

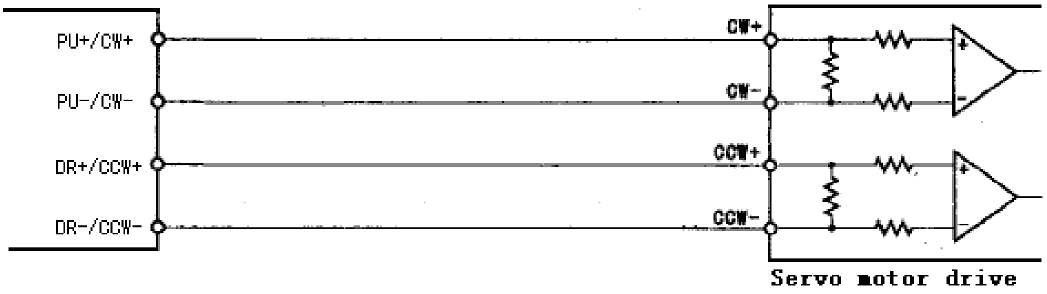
The pulse/direction signals of the motion card and wiring methods of servo drive and stepping drive are divided into differential mode and common anode mode.

The following figure shows the common anode wiring of pulse/direction:



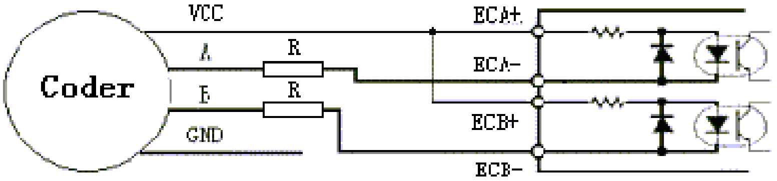
The following figure shows the differential wiring of pulse/direction (this mode is recommended for its strong anti-interference performance):





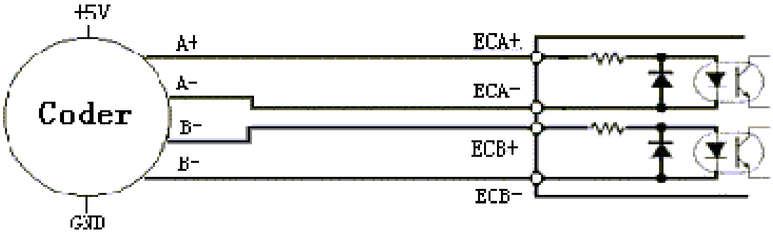
Note: See Appendix A for the wiring diagrams of stepping motor drive, common servo motor dirve and terminal board.

☞Connection of input signals of coder



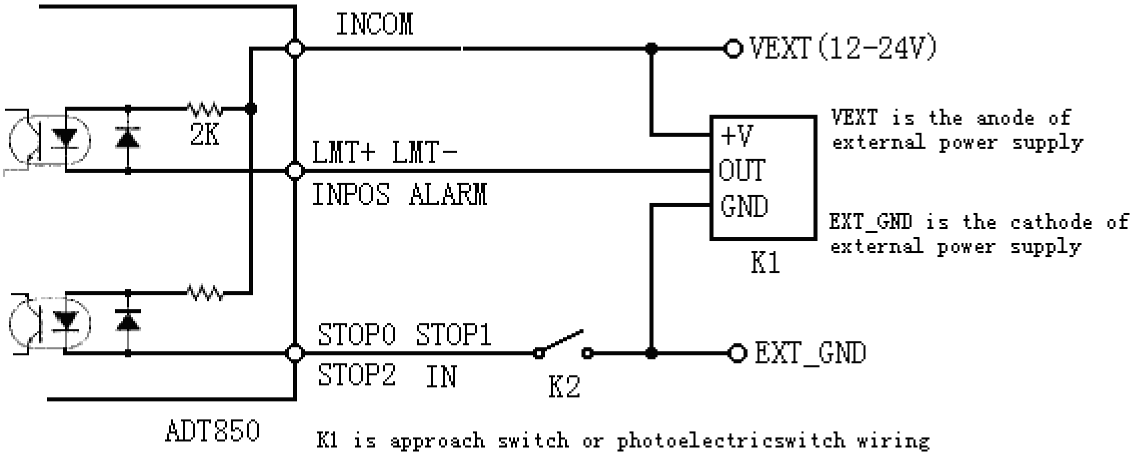
Wiring diagram of collector open-circuit output coder

+5V: R is invalid; +12V: R=1KΩ; +24V: R=2KΩ



Wiring diagram of differential drive output coder

☞Connection of digital input


INCOM
VEXT(12-24V)
2K
LMT+ LMT-
INPOS ALARM
+V
OUT
GND
K1
VEXT is the anode of external power supply
EXT_GND is the cathode of external power supply
STOP0 STOP1
STOP2 IN
K2
EXT_GND
ADT850

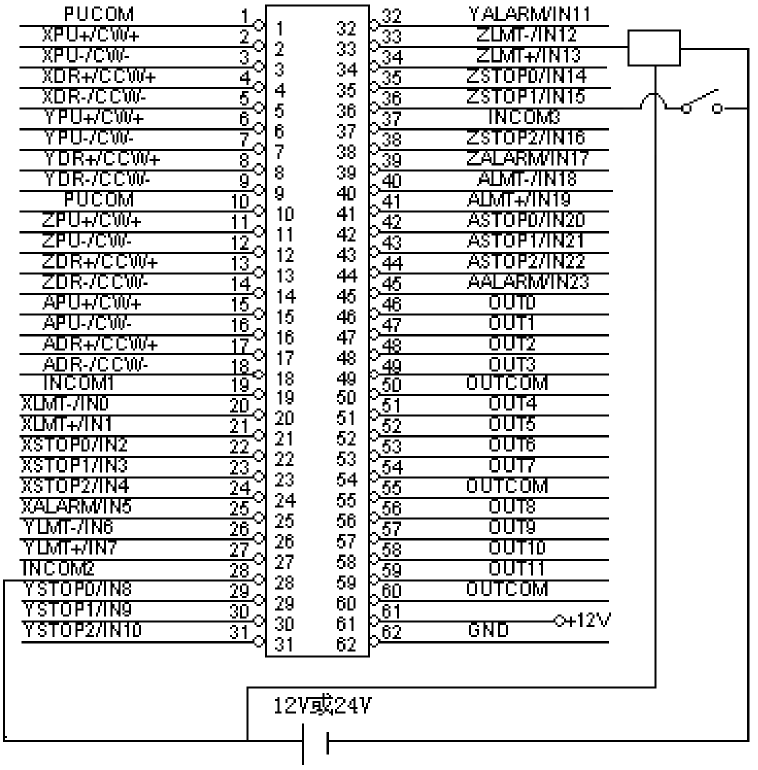
K1 is approach switch or photoelectricswitch wiring

K2 is common mechanical switch wiring

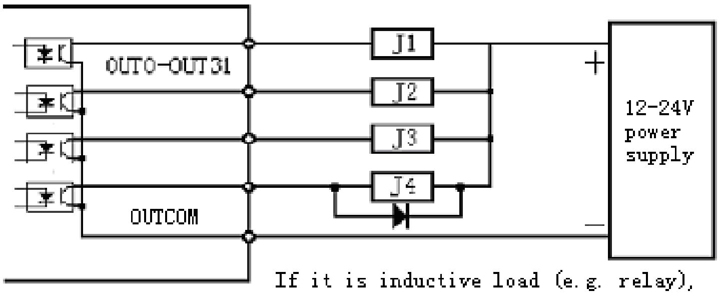
Note:

(1) To make the input signals valid, connect the "common photoelectric coupling port" of corresponding input signals (INCOM1, INCOM2, INCOM3, INCOM4, INCOMA, INCOMB) to the positive port of 12V or 24V power supply; connect one end of common switch or ground wire of approach switch to negative port (ground wire) of power supply; connect the other end of common switch or control end of approach switch to corresponding input port of terminal board.

(2) The following figure shows the wiring diagram of common switch and approach switch supplying "common photoelectric coupling port" with external power (take J1 terminal board for example).



☞Connection of digital output



If it is inductive load (e.g. relay), add freewheeling diodes on both ends of the load, as shown in J4.

Wiring modes of OUT0-OUT31

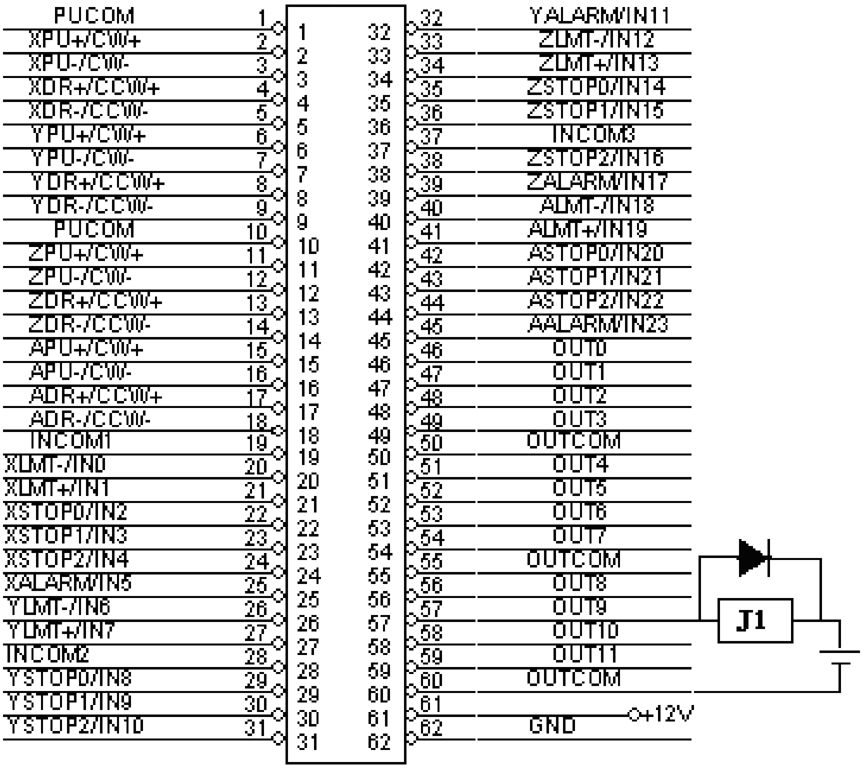
OUT0-OUT31 is photoelectric coupling isolation output; see the figure above for wiring mode; the output current of each line should be smaller than 100mA.

Note:

(1) To make the output signals valid, connect common output (OUTCOM) to negative end (ground wire) of external power supply; connect the ground wire (GND) of internal power supply to earth. Connect one end of relay coil to positive end of power supply and the other end to corresponding output of terminal board.

(2) Do not connect positive ends of external and internal power supply.

(3) The following figure shows the actual wiring diagram of relay with external power supply (take J1 terminal borad for example).



Chapter IV Installing Software

To use the ADT-8948A1 card in Win95/Win98/NT/Win2000/WinXP, you need to install driver first. The driver isn't necessary in DOS.

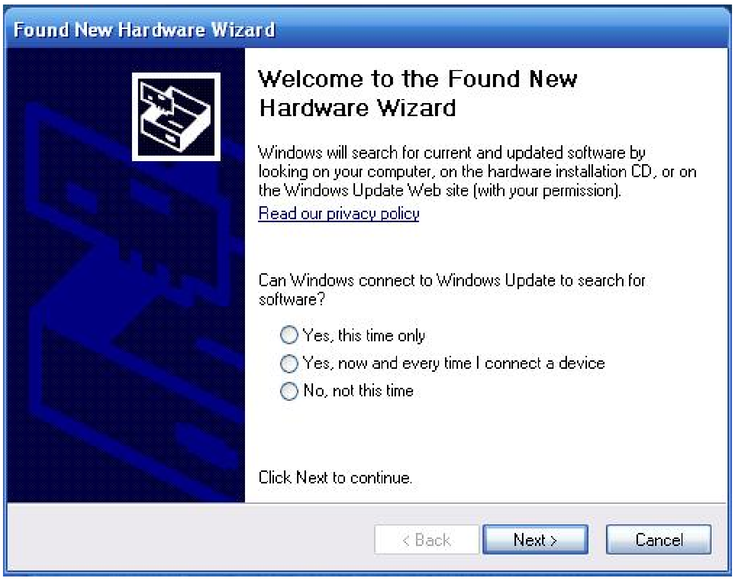
The following section shows the procedures of installing driver in WinXP. Refer to this part to install driver in other operating systems.

The driver (filename: adt8948.inf) of the motion card is in the folder "Development package\Drivers\Driver of motion card" in the disc.

☞ Install driver in WinXP

To install driver in WinXP:

1. First, the dialogue box "Found New Hardware Wizard" appears, as shown in the figure below:



Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission).

Read our privacy policy

Can Windows connect to Windows Update to search for software?

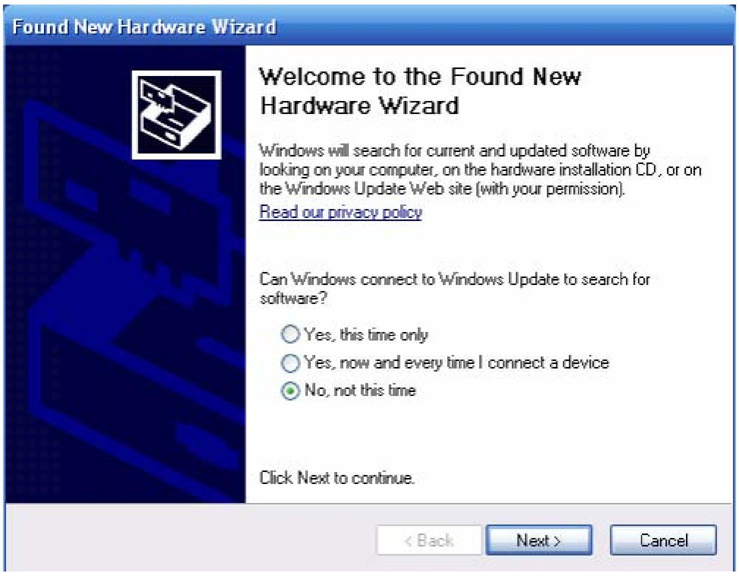
○ Yes, this time only

○ Yes, now and every time I connect a device

○ No, not this time

Click Next to continue.

Select the last option in the dialogue box (No, not this time), as shown in the figure below:



Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

Windows will search for current and updated software by looking on your computer, on the hardware installation CD, or on the Windows Update Web site (with your permission).

Read our privacy policy

Can Windows connect to Windows Update to search for software?

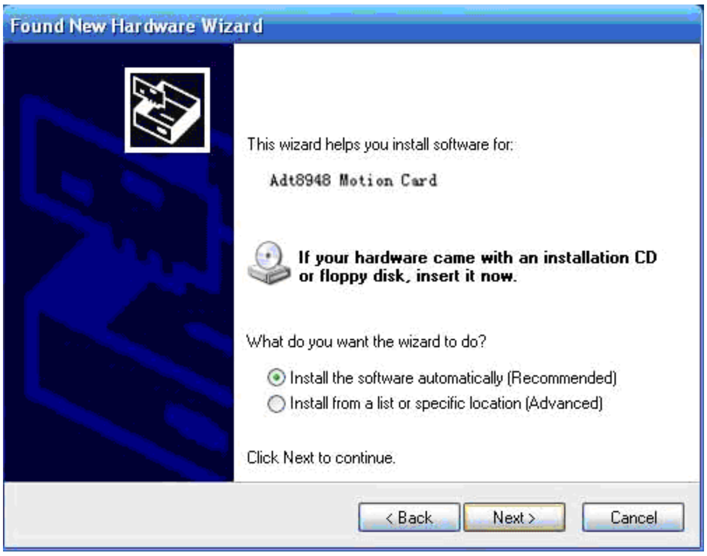
○ Yes, this time only

○ Yes, now and every time I connect a device

● No, not this time

Click Next to continue.

2. Click “Next” and the following dialogue box appears:



Found New Hardware Wizard

This wizard helps you install software for:

Adt8948 Motion Card

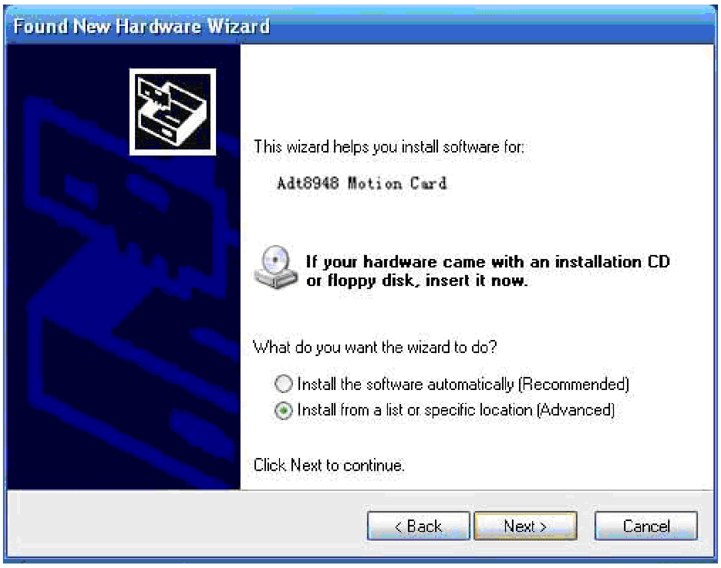
If your hardware came with an installation CD or floppy disk, insert it now.

What do you want the wizard to do?

● Install the software automatically (Recommended)
○ Install from a list or specific location (Advanced)

Click Next to continue.

Select “Install from a list or specific position (Advanced)”, as shown in the figure below:



Found New Hardware Wizard

This wizard helps you install software for:

Adt8948 Motion Card

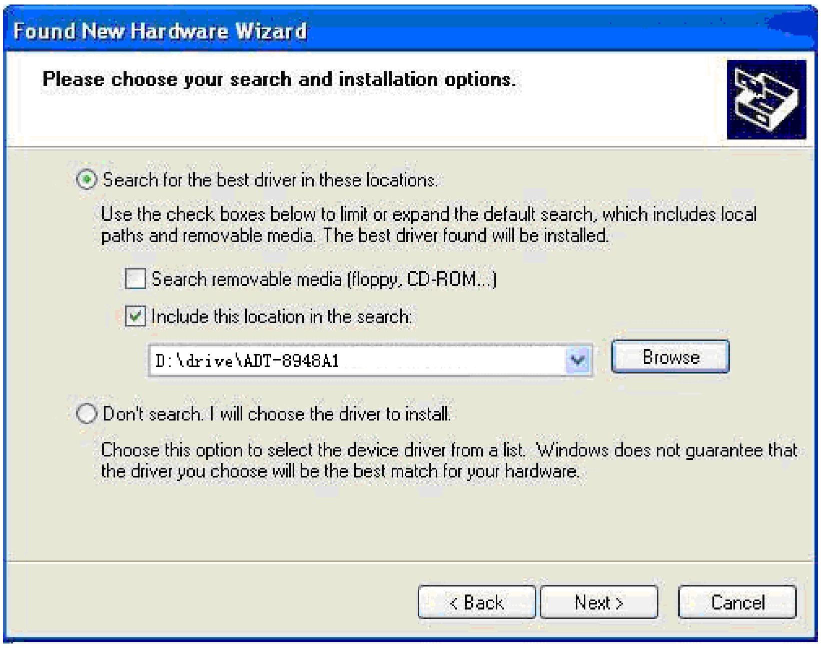
If your hardware came with an installation CD or floppy disk, insert it now.

What do you want the wizard to do?

○ Install the software automatically (Recommended)
● Install from a list or specific location (Advanced)

Click Next to continue.

3. Click “Next” and the following dialogue box appears:



Found New Hardware Wizard

Please choose your search and installation options.

● Search for the best driver in these locations.

Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.

□ Search removable media (floppy, CD-ROM...)

☑ Include this location in the search:

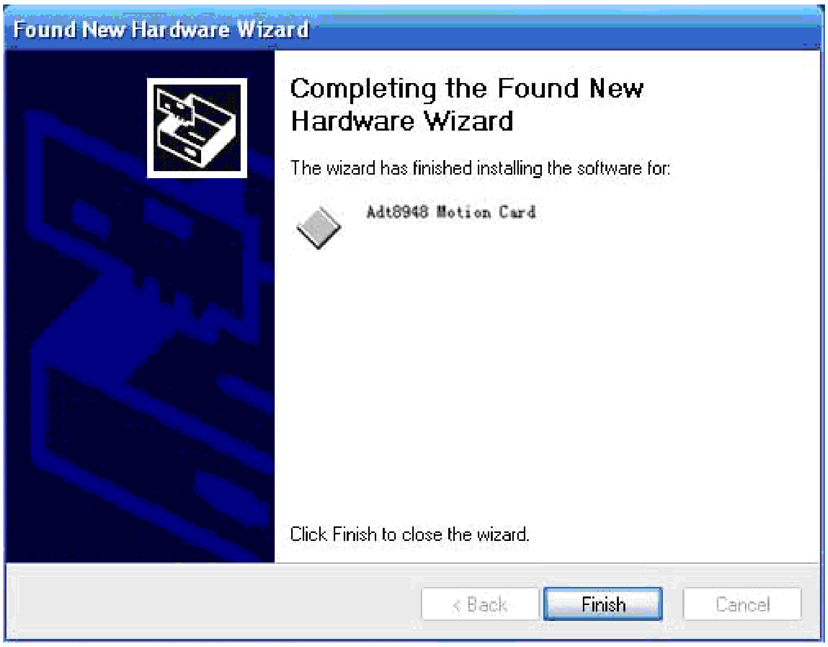
D:\drive\ADT-8948A1

○ Don't search. I will choose the driver to install.

Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.

Click “Browse” to select the position of driver.

4. Click “Next” to install. The following interface appears after installation:



Found New Hardware Wizard

Completing the Found New Hardware Wizard

The wizard has finished installing the software for:

Adt8948 Motion Card

Click Finish to close the wizard.

5. Click “Finish”.

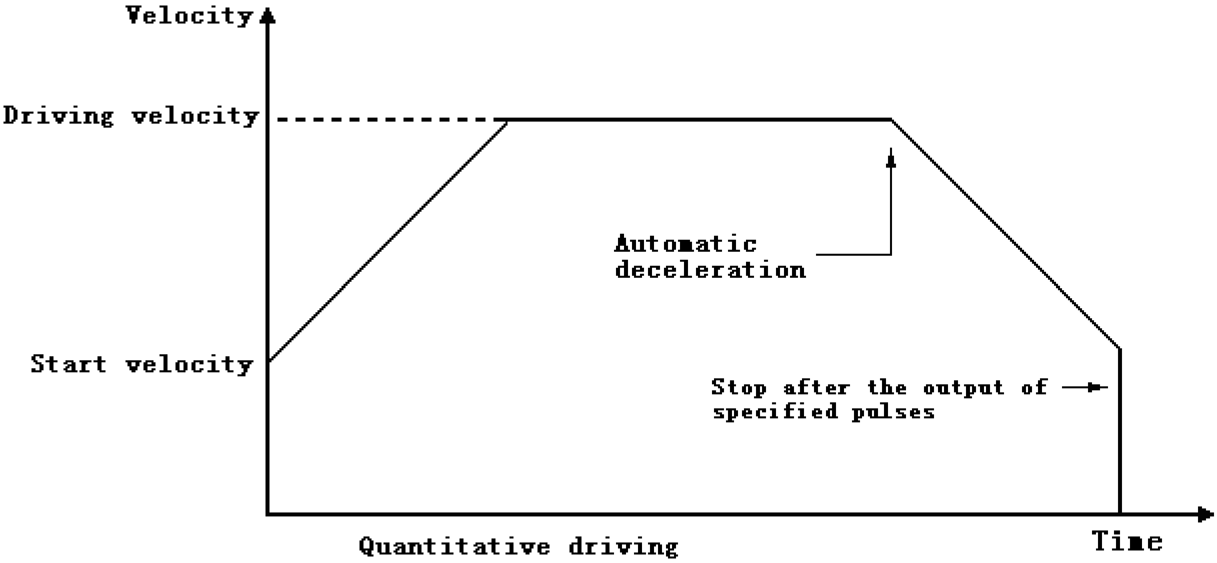
Chapter V Functions

☞Quantitative driving

Quantitative driving means to output pulse of specified amount in constant velocity or acceleration/deceleration. It is useful to move to specified position or execute specified action. The quantitative driving of acceleration/deceleration is shown in the following picture. Deceleration starts when left output pulses are less than accumulated acceleration pulses. The driving stops after the output of specified pulses.

Configure the following parameters to execute the quantitative driving of acceleration/deceleration:

a) Range R
b) Acceleration/deceleration A/D
c) Start velocity SV
d) Driving velocity V
e) Output pulse P



Acceleration/deceleration quantitative driving automatically decelerates from the deceleration point as shown in the picture above. Manual deceleration is also available. In the following conditions, the automatic deceleration point can’t be calculated accurately, thus the manual calculation is necessary:

- The velocity changes frequenctly in linear acceleration/deceleration quantitative driving.
- Perform arc and quantitative interpolation in acceleration/deceleration.

Change into manual deceleration mode and select deceleration point.

☞Continuous driving

In continuous driving, output driving pulse continuously until the high stop command or external stop signals are valid. It is useful in home searching, scanning and controlling of motor velocity.

Two stop commands are available: decelerated and sudden. Each axis has the three external signals STOP0, STOP1 and STOP2 for decelerated/sudden stop. Every signal can be set as valid/invalid electricity. STOP0, STOP1 and STOP2 signals are decelerated stop in acceleration/deceleration driving and sudden stop in constant velocity driving.

The application of continuous driving in home searching

Cinfigure home approach signal, home signal and coder phase Z signal to STOP0, STOP1 and STOP2. Set the valid/invalid and logical electricity of every signal of each axis. Acceleration/deceleration continuous driving is used in high speed searching. If the set valid signal is in activated electricity level, decelerated stop is used. Constant velocity continuous driving is used in low speed searching. If the set valid signal is in activated electricity level, sudden stop is used. To drive continuously in acceleration/deceleration, you need to configure same parameters as quantitative driving except output pulses.

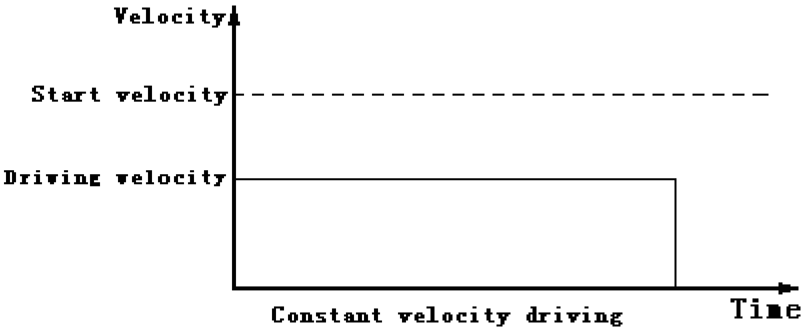
☞Velocity curve

3.1 Constant velocity driving

Constant velocity driving is to output driving pulses in constant speed. If the set driving velocity is lower than start velocity, there is only constant velocity driving. Only low velocity constant driving is necessary if you use home searching and coder phase Z signals and stop immediately when signals are searched.

Configure the following parameters to execute constant velocity driving:

- Range R
- Start velocity SV
- Driving velocity V



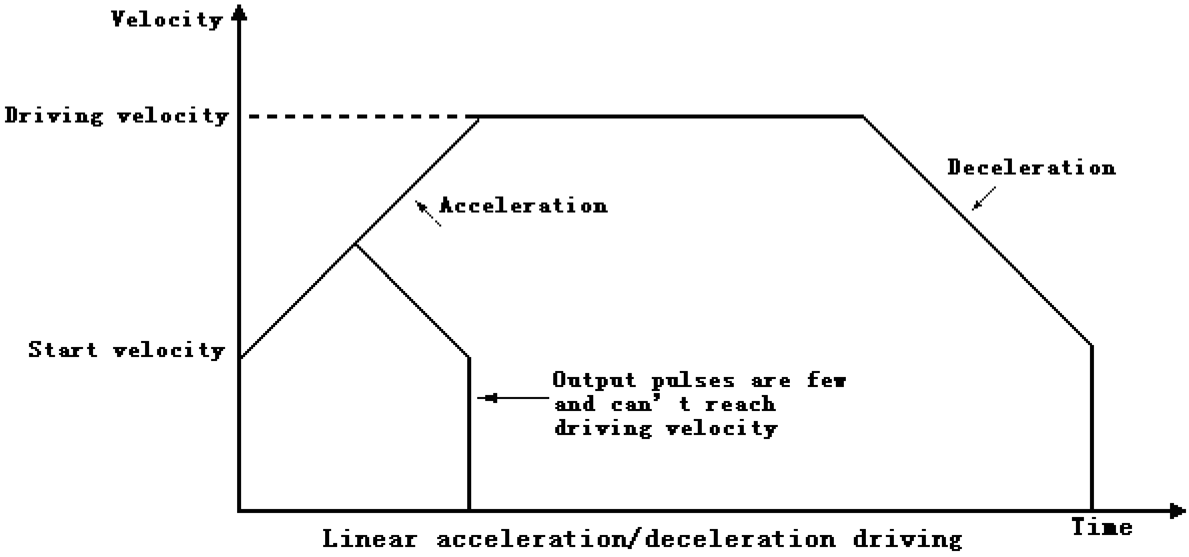
3.2 Linear acceleration/deceleration driving

Linear acceleration/deceleration driving is to accelerate from start velocity to specified driving velocity linearly.

In quantitative driving, the acceleration counter records the accumulated pulses of acceleration. If left output pulses are less than acceleration pulses, it will decelerate (automatically). In deceleration, it will decelerate to start velocity linearly in specified velocity.

Configure the following parameters to execute linear acceleration/deceleration driving:

- Range R
- Acceleration A Acceleration and deceleration
- Deceleration D Deceleration if they are set separately (if necessary)
- Start velocity SV
- Driving velocity V

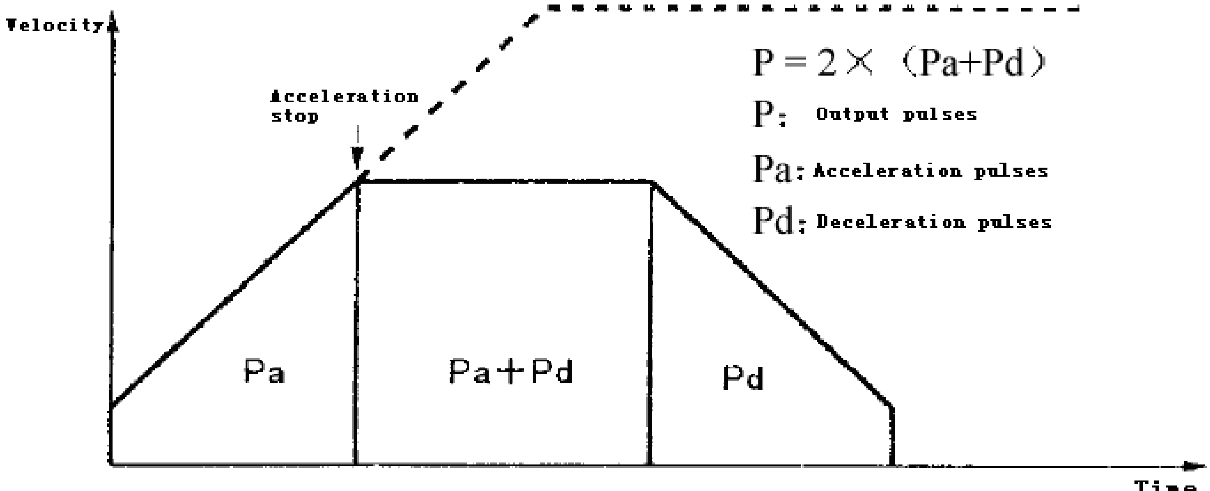


Linear acceleration/deceleration driving

☛ Triangle prevention in quantitative driving

In quantitative driving of linear acceleration/deceleration, if the output pulses are less than the required pulses to accelerate to driving velocity, triangle waves as shown in the picture will appear, and triangle prevention is activated in this case.

The triangle prevention function is to prevent triangle wave if the output pulses are less than required pulses in linear acceleration/deceleration quantitative driving. In acceleration, if the pulses consumed by acceleration and deceleration are more than 1/2 of total output pulses, acceleration stops and keeps in constant velocity field. Therefore, even if output pulses are less than 1/2 of output pulses, it is in constant velocity field.



Triangle prevention of linear acceleration/deceleration

3.3 Asymmetrical linear acceleration/deceleration driving

When an object is moved vertically, it has the load of acceleration of gravity; therefore, you'd better change the acceleration and deceleration in such asymmetrical linear acceleration/deceleration quantitative driving whose acceleration and deceleration are different. At this moment, you needn't to set manual deceleration point and automatic deceleration is used. Fig. 1 shows the example that acceleration is lower than deceleration and Fig. 2 shows the example that deceleration is lower than acceleration.

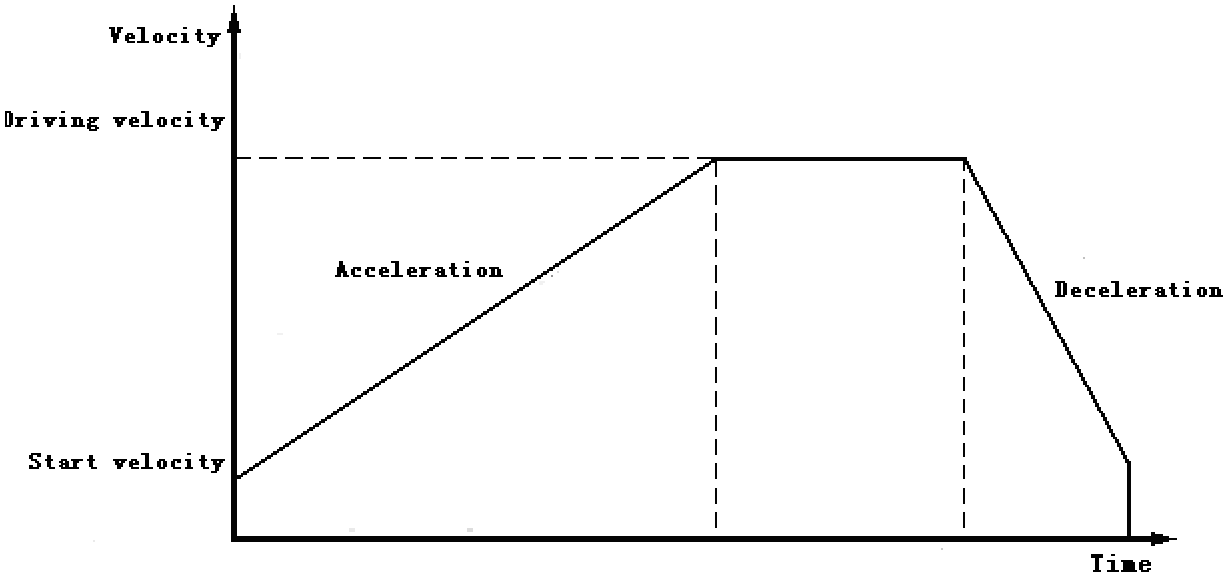


Fig. 1 Asymmetrical linear acceleration/deceleration driving
(acceleration < deceleration)

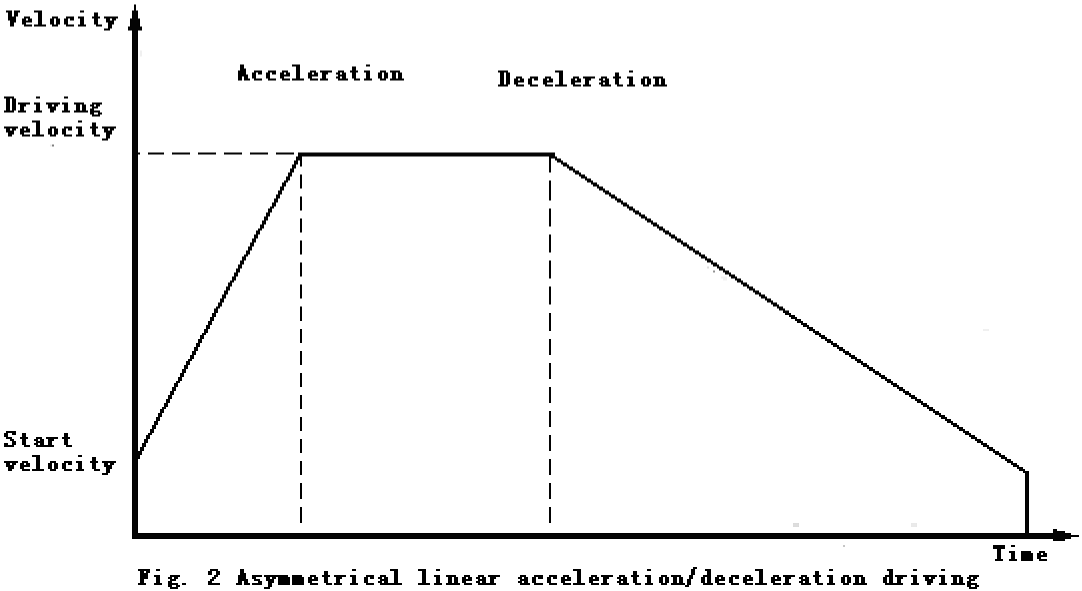


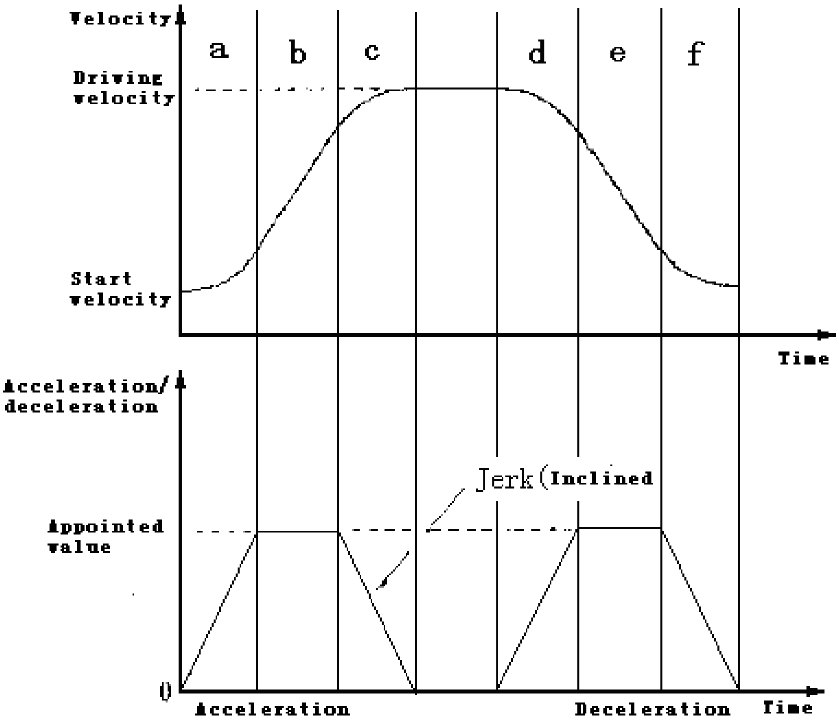
Fig. 2 Asymmetrical linear acceleration/deceleration driving (acceleration > deceleration)

Configure the following parameters like common linear acceleration/deceleration driving:

- Range R
- Acceleration A
- Deceleration D
- Start velocity SV
- Driving velocity V

3.4 S-curve acceleration/deceleration driving

When driving velocity accelerates or decelerates, the acceleration/deceleration can be increased



S-curve acceleration/ deceleration driving

When the driving accelerates, the acceleration increases from zero linearity to appointed value (A) in appointed rate (K). Therefore, this velocity curve becomes secondary parabola (a-zone). When the acceleration reaches this value (A), it will retain this value. At this moment, the velocity curve is in linear mode and the velocity is accelerating (b-zone). The acceleration tends to be 0 if the difference between target velocity (V) and current velocity is less than the velocity increased in corresponding time. The decreasing rate is same as increasing rate and decreases in appointed rate (K). At this moment, the velocity curve is in linear mode and the velocity is accelerating (c-zone). In this manual, the accelerating with partly fixed acceleration is called as partial S-curve accelerating.

On the other hand, b-zone will disappear if the difference between target velocity (V) and current velocity is less than the velocity increased in corresponding time before the acceleration reaches appointed value (A) in a-zone. The accelerating without fixed acceleration is called as complete S-curve accelerating.

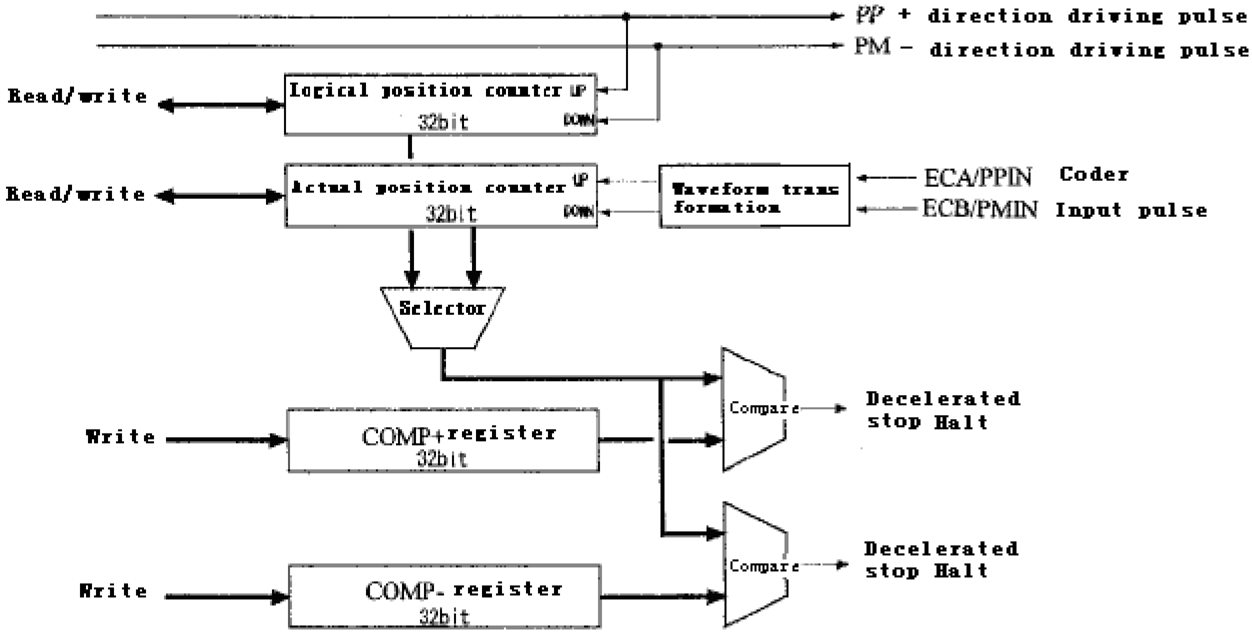
To perform S-curve acceleration/deceleration, you need to set the accelerating mode as S-curve and then configure the following parameters:

- Range R
- Change rate of acceleration/deceleration K
- Acceleration A
- Deceleration D (if necessary)
- Start velocity SV
- Driving velocity V

Precautions of performing S-curve acceleration/deceleration driving:

- Do not change the driving velocity when performing S-curve acceleration/deceleration quantitative driving.
- Do not drive are interpolation or continuous interpolation when performing S-curve acceleration/deceleration.

☞Position management



Block diagram of position management

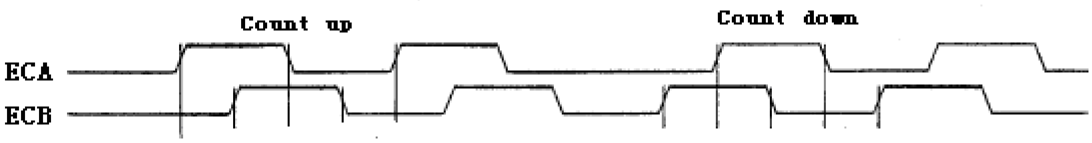
4.1 Logical position counter and actual position counter

Logical position counter is used to count the positive/negative output pulses of ADT8948A1 card. It counts up 1 after the output of one positive pulse and counts down 1 after the output of one negative pulse.

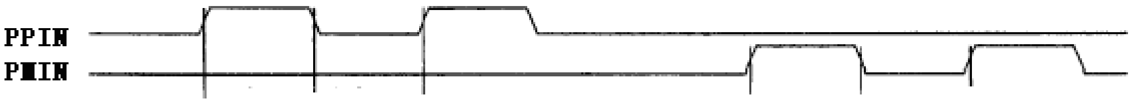
Actual position counter counts the input pulses from external coder. You can select the type of input pulse A/B phase signal or independent 2-pulse up/down counting signals. The counting direction can be customized.

The data of the two counters can be written or read at any time. The counting range is -2,147,483,648~+2,147,483,647.

■ 2-phase pulse input mode



■ Up/down pulse input mode



4.2 Compare register and software limit

Each axis has two 32-bit registers (COMP+ COMP-), which can compare size with logical position counter and actual position counter. You can customize the objects of the two compare registers as logical position counter or actual position counter. COMP+ register is mainly used to detect the upper limit of logical/actual position counter and COMP- register is mainly used to detect the lower limit.

When software limit is valid, deceleration stop is performed if the value of logical/actual position counter in the driving is bigger than the value of COMP+, and then only negative driving commands can be executed until the value of logical/actual position counter is smaller than the value of COMP+. Similarly, deceleration stop is performed if the value of ogical/actual position counter is smaller than the value of COMP-, and then only positive driving commands can be executed until the value of logical/actual position counter is bigger than the value of COMP-.

COMP+ register and COMP- register can be written at any time.

4.3 Variable circle of position counter

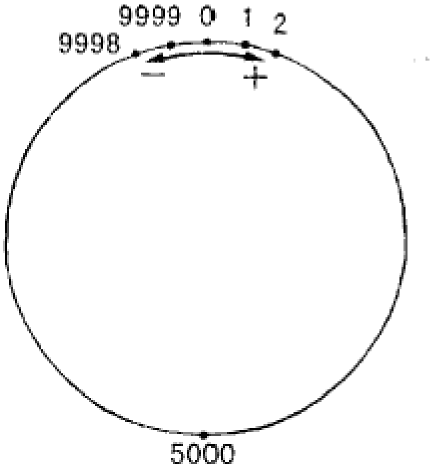
The logical position counter and actual position driver are 32-bit up/down cyclic counters. Therefore, if you count from the 32-bit maximum value FFFFFFFFh to + direction, the last counting will be 0; count from 0 to – direction, the last counting will be FFFFFFFFh. With variable circle function, you can customize the maximum value of the cyclic counter. If the position is not in linear but rolling motion, it is convenient to control position with this function. When variable circle function is activated, COMP+ register sets the maximum value of logical position counter and COMP- register sets the maximum value of actual position counter.

If X-axis is the rotating axis, supposing X-axis rotates one circle every 10,000 circles and variable corcle function is valid, set 9,999 on COMP+ register; if actual position counter is used at the same time, set 9,999 on COMP- register.

Then, the counting:

Count up to + direction: … → 9998 → 9999 → 0 → 1 …

Count down to - direction: … → 1 → 0 → 9999 → 9998 …



Maximum value 9,999 operation of variable circle of position counter

Then, the counting range is 0-9999 and you needn't to consider the calculation when the value is over 10,000.

Note

- You need to activate/deactivate the variable circle function of each axis, but can't activate/deactivate logical position counter and actual position counter separately.
- Software limit is invalid if variable circle function is activated.

☞Interpolation

ADT8948A1 card can perform linear interpolation of random 2-3 axes and arc interpolation of random 2 axes.

In interpolation driving process, the interpolation operation is performed in basic pulse time series of appointed axis. Before executing interpolation command, you need to set the start velocity and driving velocity of appointed axis.

■ Threshold-crossing error in interpolation

In interpolation driving, every driving axis can perform hardware limit and software limit. The interpolation driving will stop if the limit of any axis changes.

Note

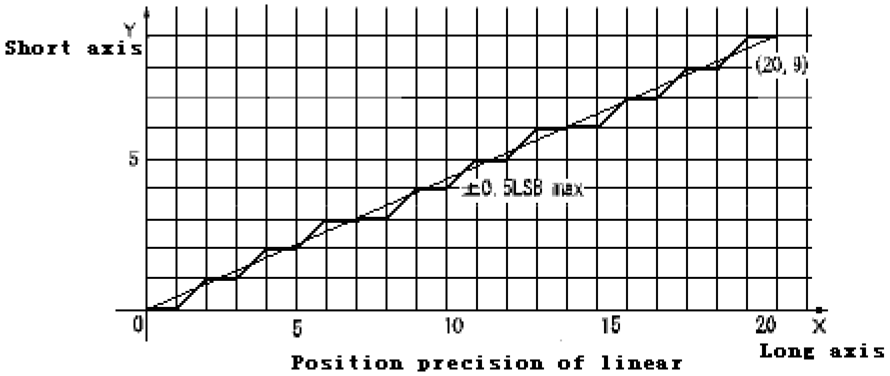
The interpolation will stop if the hardware limit or software limit in any direction (+/-) is activated in the arc interpolation process. Therefore, you must be careful when perform arc interpolation and can't leave limit area.

■ **In-position signal of servo motor**

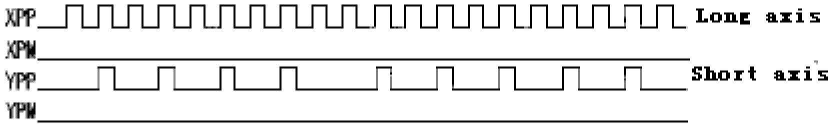
The interpolation driving will stop once the INPOS signal of each axis is valid. The INPOS signals of all axes are in effective electricity level when the interpolation driving is finished.

5.1 2-3 axes linear interpolation

The linear interpolation starts when the end coordinate relative to current position is set. The coordinate range of linear interpolation is 24-bit with symbols. The interpolation range is from current position of each axis to -8,388,607~+8,388,607.



Position precision of linear interpolation

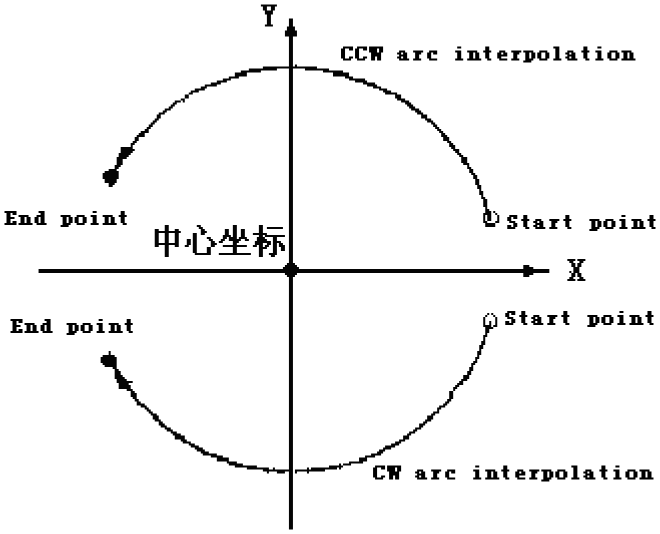


Example of end point (X: 20, Y: 9) driving pulse output

As shown in the picture above, the position precision of appointed line is ±0.5LSB in the whole interpolation range. The above picture also shows the example of pulse output of linear interpolation driving. In set end-point values, the axis with maximum absolute value is long axis and this axis always outputs pulses in the interpolation driving process. Other axes are short axes and they output pulses sometimes according to the result of linear interpolation operation.

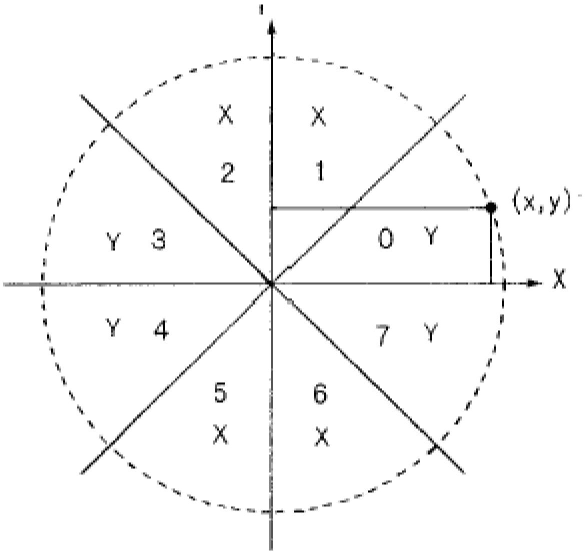
5.2 Arc interpolation

Set the center coordinates and end-point coordinates of the arc relative to the start point of current position, and then perform arc interpolation.



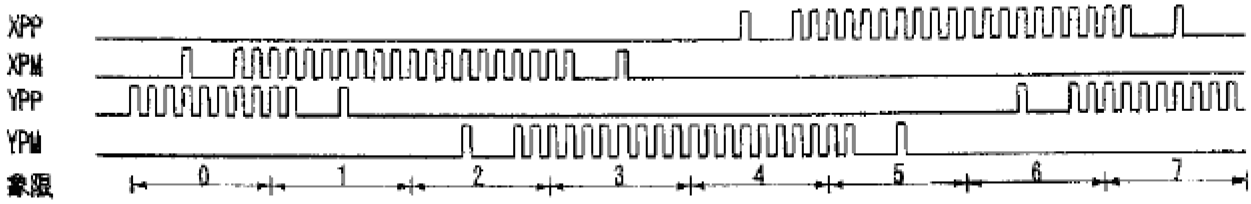
CW arc interpolation draws arc from current coordinates to end-point coordinates in clockwise direction around the center coordinates. CCW arc interpolation draws arc around the center coordinates in counterclockwise direction. It will draw a complete circle if the end point is (0, 0).

The arithmetic of arc interpolation is shown in the picture below. A plane is defined by X-axis and Y-axis. Devide it into 8 quadrants (0-7) around center coordinates. At the interpolation coordinates (X, Y) of quadrant 0, absolute value Y is always smaller than absolute value X. The axes with smaller absolute values are short axes. Quadrants 1, 2, 5 and 6 are X axes and quadrants 0, 3, 4 and 7 are Y axes. Short axes always output dricing pulses in these quadrants and long axes output pulses sometimes according to the result of arc interpolation operation.

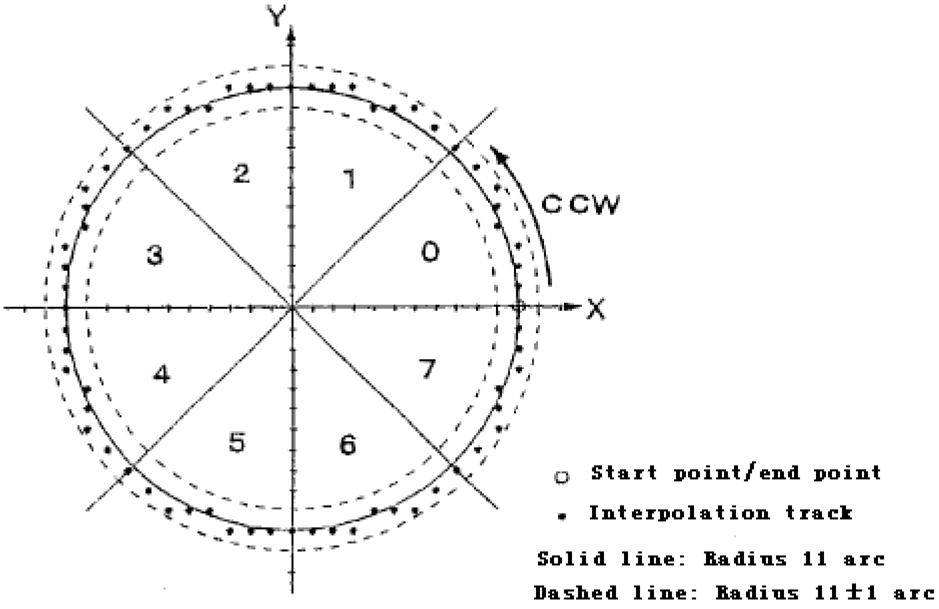


0-7 quadrants and short axis of arc interpolation arithmetic

The following examples show the output of a complete circle (take pulse output as an example).



Example of arc interpolation driving pulse output



Example of arc interpolation

■ Judgment of end-point

For arc interpolation, set the current coordinates as (0, 0) before starting interpolation driving, then, determine the radius according to the value of center coordinates and draw a circle. The error of arc algorithmic is one pulse within the range of interpolation driving. Therefore, the appointed end-point may be not on the track of the circle. When the arc interpolation enters the quadrant of end-point, it will stop if the value of end point is same to the value of short axis of end point.

5.3 Continuous interpolation

For motion card without continuous interpolation function, if you want to continue next interpolation after the previous interpolation, you have to check whether the previous interpolation is finished and then output the data of next interpolation. If the upper computer is too slow, or multi-task OS is run on the upper computer, an intermission, which has a adverse impact on effect and velocity of interpolation, occurs between interpolations.

ADT8948A1 card is intergrated with continuous interpolation function and thus this problem is solved. It can output the data of next interpolation before previous interpolation is finished. It can get excellent effect even if the computer is slow.

In continuous interpolation driving, read the writable state of continuous interpolation and interpolation driving state first, you can write the command of next interpolation if the interpolation hasn’t been finished and it is writable. Therefore, in all interpolation nodes, the time from the beginning to the end of continuous interpolation driving should be more than the time of setting the data of next interpolation node and sending command.

■ Errors in continuous interpolation

In continuous interpolation driving process, if invalid drivings like threshold-crossing error occur, it stops immediately at current interpolation node. At stopped interpolation node, the command is invalid although the data and command of next node still exist. In addition, you need to check error before sending interpolation command. If you haven't checked, the data and command will be invalid when error occurs and driving stops. If it is started from the second interpolation node at the bottom, you have to check. If any error is found, the circle of continuous interpolation should be disengaged.

If there is arc interpolation in continuous interpolation, the value of short axis of arc interpolation end point might deviate one pulse from actual value. To avoid accumulating the errors of every node, you have to confirm the end point of every arc interpolation first and then determine the mode of continuous interpolation.

5.4 Interpolation of acceleration/deceleration driving

The interpolation is usually driven in constant velocity. But ADT8948A1 card can perform interpolation in linear acceleration/deceleration driving or S-curve acceleration/deceleration driving (for linear interpolation only).

To realize acceleration/deceleration driving in continuous interpolation, you can use deceleration valid command and deceleration invalid command. In interpolation driving, deceleration valid command is used to make automatic or manual deceleration valid and deceleration invalid command is used to make them invalid. To run interpolation driving in acceleration/deceleration separately, you must select deceleration valid state before driving, otherwise, the deceleration valid command is invalid when you write it in the driving process.

■ Acceleration/deceleration driving of linear interpolation

In linear interpolation, linear acceleration/deceleration driving, S-curve acceleration/deceleration driving and automatic deceleration can be performed.

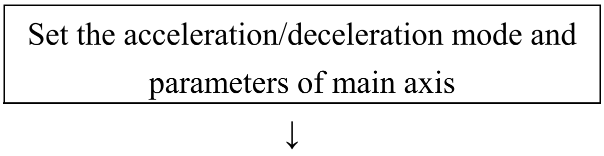
■ Acceleration/deceleration driving of arc interpolation

In bit mode arc interpolation, only manual decelerated linear acceleration/deceleration driving instead of S-curve acceleration/deceleration driving and automatic deceleration can be performed.

■ Acceleration/deceleration driving of continuous interpolation

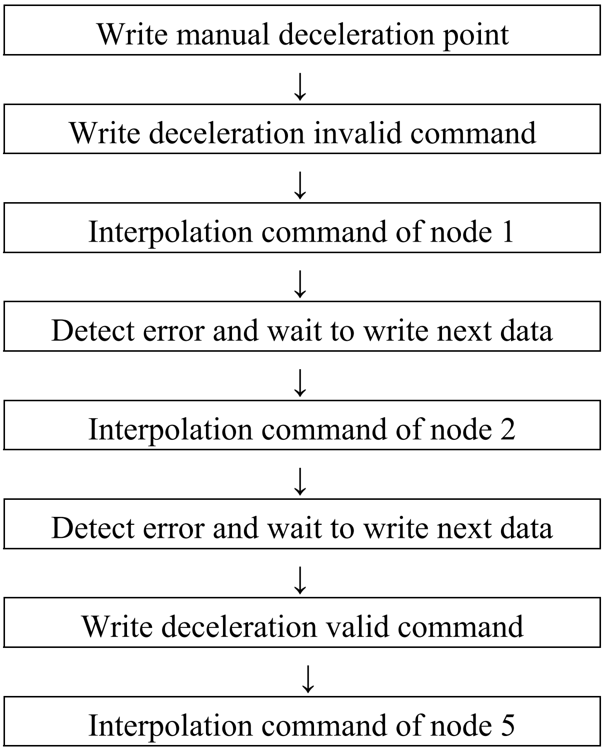
In continuous interpolation, only manual decelerated linear acceleration/deceleration driving instead of S-curve acceleration/deceleration driving and automatic deceleration can be performed. In continuous interpolation, you need to set manual deceleration point first. This manual deceleration point is set on the final node of deceleration and the value of basic oulse from X axis is also set. To perform continuous interpolation, deactivate interpolation deceleration first and then perform interpolation driving. At the final interpolation node to be decelerated, write allow deceleration command before writing interpolation command. Then, the deceleration is valid when the driving of final interpolation node starts. Deceleration starts when basic pulses from X axis of final interpolation node are bigger than the value of manual deceleration point.

For example, in the continuous interpolation from node 1 to node 5, the procedures are as follows if manual deceleration is performed on final node 5.



Set the acceleration/deceleration mode and parameters of main axis

↓


Write manual deceleration point
Write deceleration invalid command
Interpolation command of node 1
Detect error and wait to write next data
Interpolation command of node 2
Detect error and wait to write next data
Write deceleration valid command
Interpolation command of node 5

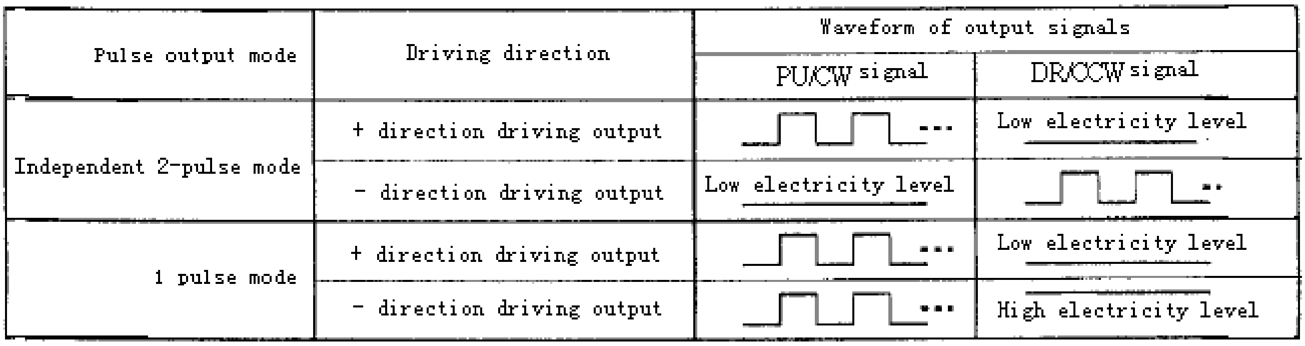
Set the manual deceleration point according to the value of basic pulses from node 5. For example, supposing that the deceleration consumes 2,000 pulses and total amount of basic pulses from node 5 is 5,000, set manual deceleration point as 5,000-2,000=3,000.

The deceleration should be performed within one node from start to end. The total basic pulses from X axis to Z axis of final interpolation point of decelerated stop should be more than the pulses consumed by deceleration.

☞Pulse output mode

The driving output pulse has two pulse output modes as shown in the figure below. In independent 2-pulse mode, PU/CW output driving pulse in positive driving and DR/CCW output driving pulse in negative driving. In single pulse mode, PU/CW output driving pulse and DR/CCW output direction signal.

脉冲/方向都是正逻辑设定时



Pulse output mode	Driving direction	Waveform of output signals	
		PU/CW signal	DR/CCW signal
Independent 2-pulse mode	+ direction driving output		Low electricity level
	- direction driving output	Low electricity level	
1 pulse mode	+ direction driving output		Low electricity level
	- direction driving output		High electricity level

☞Hardware limit signal

Hardware limit signals (LMT+, LMT-) are used to limit the input signals of positive and negative direction driving pulses. If the limit signals and their logical level are valid, you can select decelerated stop or sudden stop with command.

☞Signals corresponding to servo motor

The input signals connected to servo motor drive are INPOS signal and ALARM signal. You can activate/deactivate each signal and set the logical electricity level.

INPOS signal corresponds to the position end signal of servo motor. If the mode is valid, when a driving is finished, the waiting INPOS input signal is valid and the driving state is finished. ALARM input signal receives alarm signal from servo motor drive. It monitors the ALARM input signal if it is valid. If the signal is valid, the driving will be stopped immediately. You can read the status of the input signals for servo motor drive with universal I/O function. Universal output signal can be used to clear counter, reset counter or turn on servo.

☞Position lock

Realize hardware position lock function with the IN signal on each axis. With one lock signal, the current position (either logical or actual) of all axes can be locked.

The position lock is usefyl in mearsuring system.

☞Automatically back to home

1. Back to hone includes four steps:

(1) Approach stop0 signal in high velocity;
(2) Approach stop1 signal in low velocity;
(3) Approach stop2 signal in low velocity;
(4) Move from home to start point of processing in high velocity.

2. Process analysis

You can select whether perform the above four steps or not, as well as the direction of each step; you can also use same signal in different steps.

☞ External signal driving

External signal driving is the motion controlled by external signals (handwheel or switch). It is mainly used in the manual debugging of machines and provides a lot of convenience in teaching system.

To simplify the wiring, the motion card short connects the positive driving signals of the four axes and also short connects the negative driving signals of the four axes; therefore, only one signal cable of the coder is connected to the external interface of external singals.

☞ Stepping interpolation

Stepping interpolation is to perform interpolation in single step. When one step is finished, the next interpolation motion will be performed while waiting for the command or signal of next interpolation.

Stepping interpolation is mainly used in multi-axis manual debugging.

Chapter VI Basic Library Functions List of ADT-8948A1

Function catalog	Function name	Description	Page
Basic parameters	adt8948_initial	Initial card	33
	adt8948_end	End card	33
	set_stop0_mode	Home signal mode	33
	set_stop1_mode	Home signal mode	34
	set_stop2_mode	Home signal mode	34
	set_actualcount_mode	Actula position counter mode	34
	set_pulse_mode	Pulse mode	35
	set_limit_mode	Limit mode	35
	set_softlimit_mode1	Soft limit mode	36
	set_softlimit_mode2	Soft limit mode	36
	set_softlimit_mode3	Soft limit mode	36
	disable_soft_limit	Soft limit mode enabled	37
	enable_soft_limit	Disable soft limit mode	37
	set_inpos_mode	Servo in-position mode	37
	set_alarm_mode	Servo alarm mode	37
	set_ad_mode	Acceleration/deceleration mode	38
	set_dec1_mode	Asymmetric acceleration/deceleration setting	38
	set_dec2_mode	Deceleration mode	38
	set_circle_mode	Variable circle mode	38
	set_input_fiilter	Signal filtering	39
	set_filer_time	Signal filtering time	39
	set_lock_position	Set position lock mode	39
Driving status checking	get_status	Get axis driving status	40
	get_stopdata	Get error stop data	40
	get_inp_status	Get interpolation driving status	41

	get_inp_status2	Continuous interpolation writing status	41
	get_lock_status	Get lock status	41
	get_home_status	Get back-to-home status	42
	get_home_error	Get back-to-home error	42
Driving parameters setting	set_range	Set range	42
	set_acac	Set acceleration changing rate	43
	set_acc	Set acceleration	43
	set_dec	Set deceleration	44
	set_startv	Set start speed	44
	set_speed	Set driving speed	44
	set_command_pos	Set logical position counter	45
	set_actual_pos	Set actual position counter	45
	set_comp1	Set register	45
	set_comp2	Set register	46
	set_soft_limit	Set software limit value	46
	set_dec_pos	Set deceleration position	46
	set_vector_speed	Set speed mode	46
	set_home_mode	Set back-to-home mode	47
	set_symmetry_speed	Set symmetric acceleration/deceleration	47
	set_unsymmetry_speed	Set asymmetric acceleration/deceleration	48
Driving parameters checking	get_command_pos	Get logical position	48
	get_actual_pos	Get actual position	48
	get_speed	Get driving speed	49
	get_ad	Get acceleration	49
	get_lock_position	Get lock position	49
Version number	get_lib_vision	Get library functions version	49
Basic driving	pmove	Single axis quantitative driving	49
	continue_move	Continuous driving	50

	dec_stop	Decelerated stop	50
	sudden_stop	Sudden stop	50
	stop_axis	Stop axis	50
	stop_all	Stop all axes	51
	inp_move2	Two axes inperpolation	51
	inp_cw_arc	Clockwise arc interpolation	51
	inp_ccw_arc	Counterclockwise arc interpolation	51
	inp_move3	Three axes interpolation	52
	inp_dec_enable	Interpolation deceleration enabled	52
	inp_dec_disable	Interpolation deceleration disabled	52
	manual_pmove	Quantitative driving of external signals	53
	manual_continue	Continuous driving of external signals	53
	manual_disable	Disable external signals driving	53
	inp_step_command2	Two axes command stepping interpolation	53
	inp_step_command3	Three axes command stepping interpolation	54
	inp_step_move	Stepping interpolation driving command	54
	inp_step_signal2	Two axes signal stepping interpolation	54
	inp_step_signal3	Three axes signal stepping interpolation	54
	inp_step_stop	Stepping interpolation stop command	55
	home	Back to home	55
	clear_home_error	Clear back-to-home error	55
Synchronized function setting	set_in_move1	Single axis following signal moving	56
	set_in_move2	Two axes following signal moving	56
	set_in_move3	Three axes following signal moving	56

	set_in_stop1	Single axis following signal stop	57
	set_in_stop2	Two axes following signal stop	57
	set_in_stop3	Three axes following signal stop	57
	set_comp_pmove1	Position comparison single axis moving	58
	set_comp_pmove2	Position comparison two axes moving	58
	set_comp_pmove3	Position comparison three axes moving	58
	set_comp_stop1	Position comparison single axis stop	59
	set_comp_stop2	Position comparison two axes stop	59
	set_comp_stop3	Position comparison three axes stop	59
Composite driving	symmetry_relative_move	Single axis symmetric relative moving	60
	symmetry_absolute_move	Single axis symmetric absolute moving	60
	unsymmetry_relative_move	Single axis asymmetric relative moving	61
	unsymmetry_absolute_move	Single axis asymmetric absolute moving	61
	symmetry_relative_line2	Two axes symmetric linear interpolation relative moving	61
	symmetry_absolute_line2	Two axes symmetric linear interpolation absolute moving	62
	unsymmetry_relative_line2	Two axes asymmetric linear interpolation relative moving	62
	unsymmetry_absolute_line2	Two axes asymmetric linear interpolation absolute moving	62
	symmetry_relative_line3	Three axes symmetric linear interpolation relative moving	63
	symmetry_absolute_line3	Three axes symmetric linear interpolation absolute moving	63
	unsymmetry_relative_line3	Three axes asymmetric linear interpolation relative moving	64
	unsymmetry_absolute_line3	Three axes asymmetric linear interpolation absolute moving	64
	symmetry_relative_arc	Two axes symmetric arc interpolation relative moving	65
	symmetry_absolute_arc	Two axes symmetric arc interpolation absolute moving	65

	unsymmetry_relative_arc	Two axes asymmetric arc interpolation relative moving	66
	unsymmetry_absolute_arc	Two axes asymmetric arc interpolation absolute moving	66
Switch signals	read_bit	Read single input bit	66
	write_bit	Output single bit	67
	get_out	Get output status	67

Chapter VII Details of Basic Library Functions of ADT8948A1

☞Basic parameters settings

☞ Initialize ADT-8948A1 card

int adt8948_initial(void);

Function

Initialize motion card

Return value

Return value is the quantity of ADT-8948 cards in the system. Value 3 indiates that three cards have been installed and available card numbers are 0, 1 and 2;

Value 0 indicates that no ADT-8948A1 card has been installed;

Value -1 indicates that the drivers of ADT-8948A1 card haven't been installed

Relative default status after initialization:

Pulse output mode: pulse + direction
Feedback input is A/B phase coding pulse input with 4 times frequency doubling
STOP0, STOP1 and STOP2 are invalid
Limit signals LMT+ and LMT- are low electricity level valid, sudden stop
Software limit is invalid
Servo nINPOS signal is invalid
Servo nALARM signal is invalid
Available acceleration/deceleration modes are linear, symmetric and automatic.
The variable circle function of counter is invalid
Input filtering is invalid

Note

Initialization function is the premise to invoke other functions. You need to invoke this function first to confirm available cards and initialize certain parameters.

☞ Release ADT-8948A1 card

int adt8948_end(void);

Function

Release the resources of motion card

Return value 0:correct 1:wrong

☞ Set the mode of stop0 signal

int set_stop0_mode(int cardno, int axis, int value,int logic);

Function

Setup of STOP0 signal valid /invalid and logic electric level

Parameter

cardno	card number	
axis	axis number(1-4)	
value	0:invalid	1:valid
logic	0:low electric stop	1:high electric stop

Return value 0:correct 1:wrong

Default modes Signal is invalid，low electric stop

Notice

The way to stop rests on that it is A/D drive or uniform acceleration drive. For former it is A/D stop while for latter instant stop. STOP1 and STOP2 are just the same

☞ **Set the mode of stop1 signal**

int set_stop1_mode(int cardno, int axis, int value,int logic);

Function

Setup of STOP1 signal valid /invalid and logic electric level

Parameter

cardno	card number	
axis	axis number(1-4)	
value	0:invalid	1:valid
logic	0:low electric stop	1:high electric stop

Return value 0:correct 1:wrong

Default modes Signalinvalid，low electric stop

☞ **Set the mode of stop2 signal**

int set_stop2_mode(int cardno, int axis, int value,int logic);

Function

Setup of STOP2 signal valid /invalid and logic electric level

Parameter

Parameter

cardno	card number	
axis	axis number(1-4)	
value	0:invalid	1:valid
logic	0:low electric stop	1:high electric stop

Return value 0:correct 1:wrong

Default modes Signalinvalid，low electric stop

Notice

STOP2 signal can clear actual position counter when it is valid. Due to the delay of servo system or mechanical system, error may occur in home position if you clear the actual position counter with software after driving. With this function, you can get higher precision.

☞ **Set the working mode of actual position counter (coder input)**

int set_actualcount_mode(int cardno, int axis, int value,int dir,int freq);

Function

Set working mode of actual position counte

Parameter

cardno card number

axis axis number(1-4)

value Pulse input mode

0:A/B pulse input

1:Up/Down (PPIN/PMIN) pulse input

dir Counting direction

0: A leads B or PPIN pulse input up count

B leads A or PMIN pulse input down count

1:B leads A or PMIN pulse input up count

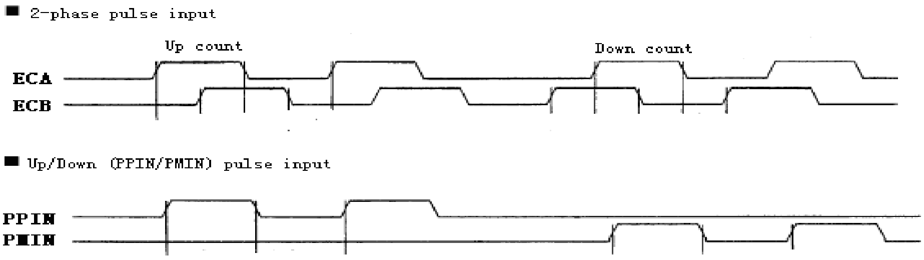
A leads B or PPIN pulse input down count

freq Frequency multiplication of A/B pulse input， Invalid for Up/Down pulse input

0: 4× 1: 2× 2: 1×

Return value 0:correct 1:wrong

Default modes A/B pulse input，direction: 0，Frequency multiplication: 4×



Most position feedback devices use coder or grating scale, so you need to select A/B phase pulse input mode. The precision of this mode can be improved with frequency doubling technology. You can select 4 times or 2 times, or disable frequency doubling. For 4 times frequency doubling, if a coder of 1000 pulse per circle is used, the counter value will increase by 4000 when it rotates one circle clockwise, i.e. the precision is increased by 4 times

☞ **Set working mode of output pulse**

int set_pulse_mode(int cardno, int axis, int value,int logic,int dir_logic);

Function

Set working mode of output pulse

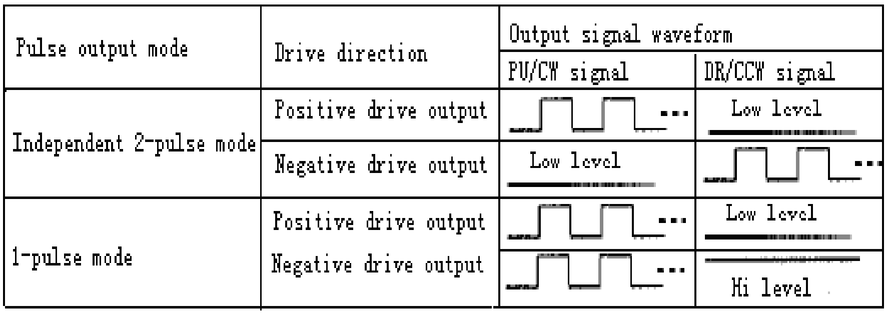
Parameter

cardno card number

axis axis number(1-4)

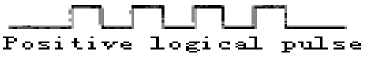
value 0:Pulse + Pulse 1:Pulse + Direction

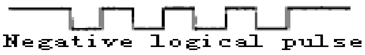
Pulse/direction are both of positive logic setting



Pulse output mode	Drive direction	Output signal waveform	
		PU/CW signal	DR/CCW signal
Independent 2-pulse mode	Positive drive output	┌┐┌┐...	Low level
	Negative drive output	Low level	┌┐┌┐...
1-pulse mode	Positive drive output	┌┐┌┐...	Low level
	Negative drive output	┌┐┌┐...	Hi level

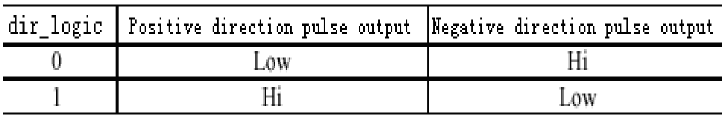
logic 0:Positive logical pulse 1:Negative logical pulse


Positive logical pulse


Negative logical pulse

dir-logic 0: Positive direction pulse output

1:Negative direction pulse output



dir_logic	Positive direction pulse output	Negative direction pulse output
0	Low	Hi
1	Hi	Low

Return value 0:correct 1:wrong

Default mode Pulse + direction; positive logical pulse; Positive logic of direction output signal

☞ Set the working mode of limit signal

int set_limit_mode(int cardno, int axis, int value, int logic);

Function

Set the mode of positive/negative limit input signal nLMT

Parameter

cardno card number

axis axis number(1-4)

value 0:sudden stop 1:decelerated stop0:sudden stop 1:decelerated stop

logic 0:low electric valid1:high electric valid

Return value 0:correct 1:wrong

Default modes sudden stop，low electric valid

Notice

The limit signal can’t be valid or invalid.

☞ Setting of COMP+ register as software limit

int set_softlimit_mode1(int cardno, int axis, int value);

Function

Set COMP + register as software limit

Parameter

cardno card number

axis axis number(1-4)

value 0:invalid 1:valid

Return value 0:correct 1:wrong

Default modes invalid

Notice

Software limit is always decelerated stop and the counting value may exceed the setting value. It is necessary to consider this point while setting range

☞ Setting of COMP- register as software limit

int set_softlimit_mode2(int cardno, int axis, int value);

Function

Set COMP - register as software limit

Parameter

cardno card number

axis axis number(1-4)

value 0:invalid 1:valid

Return value 0:correct 1:wrong

Default modes invalid

Notice

The same as above

☞ Comparison objects setting of COMP+/- registers

int set_softlimit_mode3(int cardno, int axis, int value);

Function

set COMP+/- registers as the compare objects of software limit

Parameter

cardno	card number	
axis	axis number(1-4)	
value	0:Logical position counter	1:Actual position counter

Return value 0:correct 1:wrong

Default modes Logical position counter

This function is the comparison object of setting software limit.

☞ Set software limit invalid

int disable_soft_limit(int cardno, int axis);

Function

set software limit function invalid

Parameter

cardno	card number
axis	axis number(1-4)

Return value 0:correct 1:wrong

Notice

This function is encapsulated with COMP+ and COMP- setting functions.

☞ Set software limit valid

int enable_soft_limit(int cardno, int axis, int value);

Function

set software limit function valid

Parameter

cardno	card number
axis	axis number(1-4)
value	compare objects (0: logical position counter;1:Actual position counter)

Return value 0:correct 1:wrong

Notice

This function is encapsulated with COMP+, COMP- and COMP+/- setting functions

☞ Setting of servo in-position signal nINPOS

int set_inpos_mode(int cardno, int axis, int value, int logic);

Function

Setting of servo in-position signal nINPOS

Parameter

cardno	card number	
axis	axis number(1-4)	
value	0:invalid	1:valid

logic	0:low electricvalid	1:high electricvalid

Return value 0:correct 1:wrong

Default modes invalid， low electricvalid

Notice

Do not select valid if nINPOS isn't connected to servo or stepping motor is used.

☞ **Setting of servo alarm signal nALARM**

int set_alarm_mode(int cardno, int axis, int value,int logic);

Function

Set the working mode of servo alarm signal nALARM

Parameter

cardno	card number	
axis	axis number(1-4)	
value	0:invalid	1:valid
logic	0:low electric valid	1:high electric valid

Return value 0:correct 1:wrong

Default modes invalid， low electric valid

Notice

Do not select valid if nALARM isn't connected to servo or stepping motor is used

☞ **Acceleration/deceleration setting**

int set_ad_mode(int cardno, int axis, int value);

Function

Select linear or S-curve acceleration/deceleration

Parameter

cardno	card number	
axis	axis number(1-4)	
value	0:linear A/D	1:S-curve A/D

Return value 0:correct 1:wrong

Default modes linear A/D

☞ **Asymmetric ladder acceleration/deceleration setting**

int set_dec1_mode(int cardno, int axis, int value);

Function

Select symmetric or asymmetric acceleration/deceleration

Parameter

cardno	card number
axis	axis number(1-4)
value	deceleration mode (0: symmetric deceleration, 1: asymmetric deceleration)

Return value 0:correct 1:wrong

Default modes symmetric acceleration/deceleration

☞ **Deceleration mode setting of acceleration/deceleration quantitative driving**

int set_dec2_mode(int cardno, int axis, int value);

Function

Set deceleration mode

Parameter

cardno card number
axis axis number(1-4)
value 0:automatic deceleration 1: manual deceleration

Return value 0:correct 1:wrong

Default modes Automatic deceleration

Notice

Automatic deceleration is used in most cases. To use manual deceleration, it is necessary to set deceleration point

☞ Setting of variable circle function of the counter

int set_circle_mode(int cardno, int axis, int value);

Function

Set the variable circle mode of counter

Parameter

cardno card number
axis axis number(1-4)
value 0:invalid 1:valid

Return value 0:correct 1:wrong

Default modes invalid

☞ Input signal filtering function setting

int set_input_filter(int cardno,int axis,int number,int value);

Function

Set the filtering function of input signal

Parameter

cardno card number
axis axis number
number Input types
1:LMT+、LMT-、STOP0、STOP1
2:STOP2
3:nINPOS、nALARM
4:nIN
Set the filtering state of the four types input signals above
value 0: Filtering invalid 1: Filtering valid

Return value 0:correct 1:wrong

Default modes invalid

☞ Constant setting of filtering time of input signals

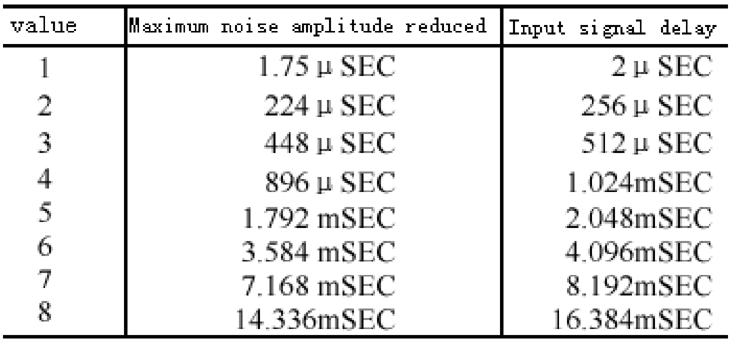
int set_filter_time(int cardno,int axis,int value);

Function

Set the filtering time constant of input signals

Parameter

cardno card number
axis axis number
value Range: 1-8, and the meanings are as follow:



value	Maximum noise amplitude reduced	Input signal delay
1	1.75 μ SEC	2 μ SEC
2	224 μ SEC	256 μ SEC
3	448 μ SEC	512 μ SEC
4	896 μ SEC	1.024mSEC
5	1.792 mSEC	2.048mSEC
6	3.584 mSEC	4.096mSEC
7	7.168 mSEC	8.192mSEC
8	14.336mSEC	16.384mSEC

Return value 0:correct 1:wrong

☞ **Set the working mode of position lock**

int set_lock_position(int cardno, int axis,int regi,int logical);

Function

Set the working mode of position lock

Parameter

cardno	card number
axis	axis number
regi	Counter type (0:logical position counter;1:Actual position counter)
logical	Electricity level logic of lock signals (0: low level lock; 1: high level lock)

Return value 0:correct 1:wrong

☞**Driving status checking**

☞ **Get the driving status of single axis**

int get_status(int cardno,int axis,int *value)

Function

Get the driving status of single axis

Parameter

cardno	card number
axis	axis number(1-4)
value	Index of driving status; the meanings are:

0: Driving stopped

Non-0: the value is in 2 bytes and their meanings are:

D0 is the lowest position and D15 is the highest position

D0: indicate the size comparison of logical/actual position counter and COMP+ register

1: logical/actual position counter >=COMP+ register

0: logical/actual position counter <COMP+ register

D1: indicate the size comparison of logical/actual position counter and COMP- register

1: logical/actual position counter >=COMP- register

0: logical/actual position counter <COMP- register

D2: In acceleration/deceleration driving, it is 1 in acceleration

D3: In acceleration/deceleration driving, it is 1 in constant speed

D4: In acceleration/deceleration driving, it is 1 in deceleration

D5: In S-curve acceleration/deceleration driving, it is 1 when acceleration/deceleration is increased

D6: In S-curve acceleration/deceleration driving, it is 1 when acceleration/deceleration is constant

D7: In S-curve acceleration/deceleration driving, it is 1 when acceleration/deceleration is decreased

D8-D15: Reserved

Return value 0:correct 1:wrong

Notice

If single axis driving command is executed, you can send next driving instruction to the axis when the driving of corresponding axis is stopped. Otherwise, previous driving instruction stops immediately and next instruction is executed.

☞ Get the error stop data of axes

int get_stopdata(int cardno,int axis,int *value)

Function

Get the error stop data of axes

Parameter

cardno	card number
axis	axis number(1-4)
value	Index of error status
	0: No error

Non-0: the value is in 2 bytes and their meanings are :

D0 is the lowest position and D15 is the highest position

D0: Stopped by STOP0

D1: Stopped by STOP1

D2: Stopped by STOP2

D3: Stopped by positive limit LMT+

D4: Stopped by negative limit LMT-

D5: Stopped by servo alarm

D6: COMP+ register limit driving stopped

D7: COMP- register limit driving stopped

D8-D15: Reserved

Return value 0:correct 1:wrong

☞ Get the driving status of interpolation

int get_inp_status(int cardno,int *value)

Function

Get the driving status of interpolation

Parameter

cardno	card number
value	Index of interpolation status
	0: Interpolation stopped 1: Interpolating

Return value 0:correct 1:wrong

Notice

If interpolation driving command is executed, you can send next driving instruction to the axis when the interpolation driving of corresponding axis is stopped. Otherwise, previous driving instruction stops immediately and next instruction is executed.

☞ Get the writable status of continuous interpolation

int get_inp_status2(int cardno,int *value)

Function

Get the writable status of continuous interpolation

Parameter

cardno	card number
value	Index of writing status 0: Unwritable 1: Writable

Return value 0:correct 1:wrong

Notice

If the driving is stopped, the status is 0. Threrfore, it is necessary to check whether error occurs in continuous interpolation process.

☞ Get lock status

int get_lock_status(int cardno, int axis, int *status)

Function

Get the status of position lock

Parameter

cardno	card number
axis	axis number(1-4)
status	Lock status (0: unlocked, 1: locked)

Return value 0:correct 1:wrong

☞ Get back-to-home status

int get_home_status(int cardno, int axis, int *status, int *err);

Function

Get back-to-home status

Parameter

cardno	card number
axis	axis number(1-4)
status	Whether driving is stopped (0:driving stopped, 1: moving)
err	Whether error occurs (0:correct,1:wrong)

Return value 0:correct 1:wrong

☞ Get back-to-home error

int get_home_error(int cardno, int axis ,int *err);

Function

Get back-to-home error information

Parameter

cardno	card number
axis	axis number(1-4)
err	Error marker 0:correct >0:wrong The meanings of first 7 digits of err:

D0: comp+ limit

D1: comp- limit

D2: LMT+ limit

D3: LMT- limit
D4: Servo alarm
D5: Emergency stop
D6: Z phase signal arrives in advance

Return value 0:correct 1:wrong

☞Moving parameter settings

The values of the following parameters are not fixed when the motion card is initialized. You need to set before using

☞ Range setting

int set_range(int cardno,int axis,long value);

Function

set range

Parameter

cardno	card number
axis	axis number(1-4)
value	range(8000000-16000)

Return value 0:correct 1:wrong

Notice

Range is the rate parameter that determines the speed, acceleration/deceleration and change rate of acceleration/deceleration. If the range is R, the formula to calculate M is: M = 8000000/R.

The effective range of driving speed, inialization speed and acceleration/deceleration is 1~8000. If required actual speed or acceleration/deceleration is higher than 8000, you need to set the range and adjust the magnification. If magnification is increased, the actual speed and acceleration/deceleration can be increased in same scale, but the resolution of speed and acceleration/deceleration becomes rough. Therefore, it is recommended that you set minimum value for magnification and maximum value for range within stated range of actual speed.

In engineering practice, the setting of reasonable range is the premise to get ideal actual speed curve. To calculate the range**:**

Step 1: Calculate the magnification (M) according to estimated maximum actual speed (Vmax) M=Vmax/8000 (8000 is the maximum speed);

Step 2: Calculate range (R) according to magnification: R = 8000000/M.

For example: if required maximum speed is 40KPPS, then, M = 40*1000/8000 = 5 and R = 8000000/5 = 1600000.

The range of R is 8000000-16000 and corresponding rate is 1-500.

You’d better set the range in the process of system initialization. Do not change the range in the moving process; otherwise, the speed may jump.

In a word, actual speed = set value of speed * magnification.

☞ Set the change rate of acceleration/deceleration

int set_acac(int cardno,int axis,long value);

Function

Set the change rate of acceleration/deceleration

Parameter

cardno	card number
axis	axis number(1-4)
value	range(1-65535)

Return value 0:correct 1:wrong

Notice

The change rate of acceleration/deceleration is the parameter that determines the change rate of acceleration and deceleration of S-curve acceleration/deceleration in unit time. If the value of acceleration/deceleration change rate is K, the actual value V (PPS/SEC^2) of acceleration/deceleration change rate is V = (62500000/K)*M = (62500000/K)*(8000000/R).

The range of acceleration/deceleration change rate is 1~65,535.

If the setting is as follow:

set_range(0,1,800000); the range is 800000 and magnification M=8000000/800000=10

set_acac(0,1,100); acceleration/deceleration change rate is 100;

Then, the actual change rate of acceleration/deceleration is (62500000/100)*10=6250000 PPS/SEC^2

☞ **Acceleration setting**

int set_acc(int cardno,int axis,long value);

Function

Set the value of acceleration

Parameter

cardno	card number
axis	axis number(1-4)
value	range(1-8000)

Return value 0:correct 1:wrong

Notice

This is the acceleration and deceleration parameter in linear acceleration/deceleration driving. In S-curve acceleration/deceleration driving, the acceleration and deceleration increase linearly from 0 to set acceleration value.

The set acceleration value is A and the actual acceleration is:

Actual acceleration (PPS/SEC) = A*125*M = A*125*(8000000/R)

The range of acceleration value is 1~8,000.

If the setting is as follow:

set_range(0,1,80000); the range is 800000 and magnification M=8000000/800000=10

set_acc(0,1,100);

The acceleration is:

100*125* (8000000/80000) =1250000 PPS/SEC

☞ **Deceleration setting**

int set_dec(int cardno,int axis,long value);

Function

Set the value of deceleration

Parameter

cardno	card number
axis	axis number(1-4)
value	D-value(1-8000)

Return value 0:correct 1:wrong

Notice

It is the deceleration parameter in linear acceleration/deceleration driving in acceleration/deceleration fixed mode. In S-curve acceleration/deceleration driving, the deceleration increases linearly from 0 to set deceleration value.

The deceleration value is D and the actual deceleration is:

Actual deceleration (PPS/SEC) = D*125*M=D*125*(8000000/R)

The range of deceleration value D is 1~8,000.

If the setting is as follow:

set_range(0,1,80000); the range is 800000 and magnification M=8000000/800000=10

set_dec(0,1,100);

The deceleration is:

100*125* (8000000/80000) =1250000 PPS/SEC

☞ **Start velocity setting**

int set_startv(int cardno,int axis,long value);

Function

Set the value of start velocity

Parameter

cardno	card number
axis	axis number(1-4)
value	range(1-8000)

Return value 0:correct 1:wrong

Notice

Start velocity is the velocity when the driving starts. The start velocity value is SV and the actual value of start velocity is: actual value of start velocity (PPS) = SV*M = SV*(8000000/R)

If the value of start velocity is higher than that of driving speed, the drving will be performed in driving speed constantly even if the acceleration/deceleration has been set.

☞ **Driving speed setting**

int set_speed(int cardno,int axis,long value);

Function

Set the value of driving speed

Parameter

cardno	card number
axis	axis number(1-4)
value	range(1-8000)

Return value 0:correct 1:wrong

Notice

Driving speed is the speed that reaches constant speed area in the moving process. The driving speed should be no lower than start velocity in principle. The driving speed is V and the actual value of driving speed is: actual value of driving speed (PPS) = V*M = V*(8000000/R).In ladder acceleration/deceleration or constant speed moving process, you can change the driving speed in real time. You can’t change the driving speed in the quantitative pulse driving process of S-curve acceleration/deceleration. Besides, if the speed is changed in acceleration area or deceleration area in the continuous driving process of S-curve acceleration/deceleration, you can't run correct S-curve. Please change the speed in constant speed area.

In the quantitative pulse driving of linear acceleration/deceleration, if the driving speed si changed frequently, the dragging driving in start velocity will probably occur in the deceleration at the end of output pulse

☞ **Logical position counter setting**

int set_command_pos(int cardno,int axis,long value);

Function

Set the value of logical position counter

Parameter

cardno	card number
axis	axis number(1-4)
value	range value(-2147483648~+2147483647)

Return value 0:correct 1:wrong

Notice

You can access the logical position counter in real time

☞ **Actual position counter setting**

int set_actual_pos(int cardno,int axis,long value);

Function

Set the value of actual position counter

Parameter

cardno	card number
axis	axis number(1-4)
value	range value(-2147483648~+2147483647)

Return value 0:correct 1:wrong

Notice

You can access the actual position counter in real time

☞ **COMP+ register setting**

int set_comp1(int cardno,int axis,long value);

Function

Set the value of COMP+ register

Parameter

cardno	card number
axis	axis number(1-4)
value	range value(-2147483648~+2147483647)

Return value 0:correct 1:wrong

Notice

You can access the COMP+ register in real time

☞ **COMP- register setting**

int set_comp2(int cardno,int axis,long value);

Function

Set the value of COMP- register

Parameter

cardno	card number
axis	axis number(1-4)
value	range value(-2147483648~+2147483647)

Return value 0:correct 1:wrong

Notice

You can access the COMP- register in real time

☞ Software limit setting

int set_soft_limit (int cardno,int axis,long value1, long value2);

Function

Set the value of software limit

Parameter

cardno	card number
axis	axis number(1-4)
value1	positive limit
value2	negative limit

Return value 0:correct 1:wrong

Notice

This function is encapsulated by COMP+ and COMP-.

☞ Manual deceleration point setting

Set the value of manual deceleration point

int set_dec_pos(int cardno,int axis,long value);

Function

Set the value of COMP+ register

Parameter

cardno	card number
axis	axis number(1-4)
value	range value(0~268435455)

Return value 0:correct 1:wrong

Notice

If manual deceleration mode is used, you need to set manual deceleration point first.Manual deceleration point = output pulses – pulses consumed by deceleration

☞ Set linear speed mode

int set_vector_speed(int cardno, int mode);

Function

Linear speed mode setting

Parameter

cardno	card number
mode	Speed mode

(0: do not use constant linear speed; 1: use constant linear speed)

Return value 0:correct 1:wrong

Notice

Linear speed indicates vector speed. Constant linear speed can guarantee the constant combination speed in interpolation

☞ Set back-to-home mode

int set_home_mode(int cardno, int axis,long speed,int logical0, int logical1, int logical2,int offset,int dir0, int dir1, int dir2,int offsetdir,int clear,long pulse);

Function

Set back-to-home mode of appointed axis

Parameter

cardno	card number
axis	axis number(1-4)
speed	Low speed (this value should be lower than start velocity of high speed)
logical0	stop0 electricity level logic (0:low electric stop, 1:high electric stop,-1:invalid)
logical1	stop1 electricity level logic (0:low electric stop, 1:high electric stop,-1:invalid)
logical2	stop2 electricity level logic (0:low electric stop, 1:high electric stop,-1:invalid)
offset	Offset symbol (0: do not offset from home, 1: offset from home)
dir0	Direction approaching stop0 (0: positive, 1: negative)
dir1	Direction approaching stop1 (0: positive, 1: negative)
dir2	Direction approaching stop2 (0: positive, 1: negative)
offsetdir	Offset direction (0: positive, 1: negative)
clear	Clear counter? (0: Yes, 1: No)
pulse	Offset pulses

Return value 0:correct 1:wrong

☞ **Set symmetric acceleration/deceleration**

int set_symmetry_speed(int cardno,int axis,long lspd,long hspd,double tacc,long vacc,int mode);

Function

Set the value of symmetric acceleration/deceleration

Parameter

cardno	card number
axis	axis number(1-4)
lspd	Start speed
hspd	Driving speed
tacc	Acceleration time
vacc	Acceleration change rate
mode	Acceleration mode (0: trapezoid 1: S-curve)

Return value 0:correct 1:wrong

Notice

This function is composed of the function that sets acceleration/deceleration mode and the functions that set start velocity, driving speed, acceleration and change rate of acceleration/deceleration.

☞ **Set asymmetric acceleration/deceleration**

int set_unsymmetry_speed(int cardno,int axis,long lspd,long hspd,double tacc,double tdec,long vacc,int mode);

Function

Set the value of asymmetric acceleration/deceleration

Parameter

cardno	card number
axis	axis number(1-4)
lspd	start speed
hspd	driving speed
tacc	acceleration time
tdec	deceleration time
vacc	acceleration change rate
mode	acceleration mode (0: trapezoid 1:S-curve)

Return value 0:correct 1:wrong

Notice

This function is composed of the function that sets acceleration/deceleration mode and the functions that set start velocity, driving speed, acceleration, deceleration and change rate of acceleration/deceleration.

☞Moving parameter checking

The following functions can be invoked at any moment.

☞ Get the logical position of axes

int get_command_pos(int cardno,int axis,long *pos);

Function

Get the logical position of axes

Parameter

cardno	card number
axis	axis number(1-4)
pos	Index of logical position value

Return value 0:correct 1:wrong

Notice

You can use this function to get the logical position of axes and it can represent the current position of axes if the motor is not out of step

☞ Get the actual position of axes (i.e. coder feedback value)

int get_actual_pos(int cardno,int axis,long *pos);

Function

Get the actual position of axis

Parameter

cardno	card number
axis	axis number(1-4)
pos	Index of actual position value

Return value 0:correct 1:wrong

Notice

You can use this function to get the actual position of axes and you can get the current position of axes even if the motor is out of step.

☞ Get the current driving speed of axes

int get_speed(int cardno,int axis,long *speed);

Function

Get the current driving speed of axes

Parameter

cardno	card number

axis axis number(1-4)

speed Index of current driving speed

Return value 0:correct 1:wrong

Notice

Value of actual speed = Getting speed*M = Getting speed*(8000000/R)

☞ Get the current acceleration of axes

int get_ad(int cardno,int axis,long *ad);

Function

Get the current acceleration of axes

Parameter

cardno	card number
axis	axis number(1-4)
ad	Index of current acceleration

Return value 0:correct 1:wrong

☞ Get lock position

int get_lock_position(int cardno,int axis,long *pos);

Function

Get the latest lock position

Parameter

cardno	card number
axis	axis number(1-4)
pos	Locked position

Return value 0:correct 1:wrong

☞Library function version checking

☞ Get the version of library function

int get_lib_vision(int *ver);

Function

Get the version number of library function

Parameter

ver Version number index (ver is a 4-digit integer with the first two digits are master version number and last two sigits are secondary version number)

Return value 0:correct 1:wrong

☞Basic driving

☞ Quantitative driving

int pmove(int cardno,int axis,long pulse);

Function

Single axis quantitative driving

Parameter

cardno	card number		
axis	axis number(1-4)		
pulse	output pulses	>0: Positive	<0: Negative
	range(-268435455~+268435455)		

Return value 0:correct 1:wrong

Notice

You need to set valid speed parameter before writing driving command.

☞ Continuous driving

int continue_move(int cardno,int axis,int dir);

Function

Single axis continuous driving

Parameter

cardno	card number		
axis	axis number(1-4)		
dir	Driving direction	0: Positive	1: Negative

Return value 0:correct 1:wrong

Notice

You need to set valid speed parameter before writing driving command.

☞ Driving decelerated stop

int dec_stop(int cardno,int axis);

Function

Stop current driving process in deceleration

Parameter

cardno	card number
axis	axis number(1-4)

Return value 0:correct 1:wrong

Notice

This command is decelerated stop in acceleration/deceleration driving, process and sudden stop in constant speed driving process.

☞ Driving sudden stop

int sudden_stop(int cardno,int axis);

Function

Stop current driving process immediately

Parameter

cardno	card number
axis	axis number(1-4)

Return value 0:correct 1:wrong

Notice

Stop pulse output immediately in acceleration/deceleration and constant speed driving process.

☞ Stop single axis

int stop_axis(int cardno,int axis,int mode);

Function

Stop the moving of singl axis in set mode.

Parameter

cardno	card number
axis	axis number(1-4)
mode	stop mode(0:sudden stop 1:decelerated stop)

Return value 0:correct 1:wrong

☞ Stop all axes

int stop_all(int cardno,int mode);

Function

Stop the moving of all axes of appointed card in set mode.

Parameter

cardno	card number
mode	stop mode(0:sudden stop 1:decelerated stop)

Return value 0:correct 1:wrong

☞ Two axes linear interpolation

int inp_move2(int cardno,**int axis1,int axis2,long pulse1,long pulse2);**

Function

Two axes linear interpolation

Parameter

cardno	card number
axis1	Interpolation axis number1(1-4，indicate :X、Y、Z、A)
axis2	Interpolation axis number2(1-4，indicate :X、Y、Z、A)
pulse1	Moving distance of axis1
pulse2	Moving distance of axis2

Return value 0:correct 1:wrong

Notice

The interpolation speed takes the speed of the axis with smaller axis number between axis1 and axis2 as the standard.

☞ CW arc interpolation

int inp_cw_arc(int cardno,**int axis1,int axis2,long x,long y,long i,long j);**

Function

Two axes CW arc interpolation

Parameter

cardno	card number
axis1	Interpolation axis number1(1-4，indicate :X、Y、Z、A)
axis2	Interpolation axis number2(1-4，indicate :X、Y、Z、A)
x,y	End point position of arc interpolation (relative to start point)
i,j	Circle center position of arc interpolation (relative to start point)

Return value 0:correct 1:wrong

Notice

The interpolation speed takes the speed of axis1

☞ CCW arc interpolation

int inp_ccw_arc(int cardno,int axis1,int axis2,long x,long y,long i,long j);

Function

Two axes CCW arc interpolation

Parameter

cardno	card number
axis1	Interpolation axis number1(1-4，indicate :X、Y、Z、A)

axis2 Interpolation axis number2(1-4，indicate :X、Y、Z、A)

x,y End point position of arc interpolation (relative to start point)

i,j Circle center position of arc interpolation (relative to start point

Return value 0:correct 1:wrong

Notice

The interpolation speed takes the speed of the axis with smallest axis number among axis1, axis2 and axis3 as the standard (axis1).

☞ **Three axes linear interpolation**

int inp_move3(int cardno,int axis1,int axis2,int axis3,long pulse1,long pulse2,long pulse3);

Function

Three axes linear interpolation

Parameter

cardno	card number
axis1	Interpolation axis number1(1-4，indicate :X、Y、Z、A)
axis2	Interpolation axis number2(1-4，indicate :X、Y、Z、A)
axis3	Interpolation axis number3(1-4，indicate :X、Y、Z、A)
pulse1	Moving distance of axis1
pulse2	Moving distance of axis2

Return value 0:correct 1:wrong

Notice

The interpolation speed takes the speed of the axis with smallest axis number among axis1, axis2 and axis3 as the standard (axis1).

☞ **Enable interpolation deceleration**

int inp_dec_enable(int cardno);

Function

Enable deceleration of interpolation

Parameter

cardno card number

Return value 0:correct 1:wrong

☞ **Disable interpolation deceleration**

int inp_dec_disable(int cardno);

Function

Disable deceleration of interpolation

Parameter

cardno card number

Return value 0:correct 1:wrong

Notice

This function and previous function are used in interpolation of acceleration/deceleration. Select Enable for single interpolation and select Disable at first and then select Enable at the last point for continuous interpolation. See the following examples.

☞ **Manual quantitative driving**

int manual_pmove(int cardno, int axis, long pulse);

Function

Quantitative driving of external signals

Parameter

cardno	card number
axis	axis number(1-4)
pulse	pulse

Return value 0:correct 1:wrong

Notice

(1) After the command is sent, the quantitative driving can be triggered when external signal (trigger signal) drops to low electricity level;

(2) In the driving process, the trigger signal isn't valid even if it drops to low electricity level;

(3) The trigger signal may be triggered with handwheel or normal switch.

☞ **Manual continuous driving**

int manual_continue(int cardno, int axis);

Function

Continuous driving of external signals

Parameter

cardno	card number
axis	axis number(1-4)

Return value 0:correct 1:wrong

Notice

(1) After the command is sent, the quantitative driving can be triggered when manual signal (trigger signal) drops to low electricity level;

(2) In the driving process, the trigger signal isn't valid even if it drops to low electricity level;

(3) The trigger signal may be triggered with handwheel or normal switch.

☞ **Disable manual driving**

int manual_disable(int cardno, int axis);

Function

Disable manual driving function

Parameter

cardno	card number
axis	axis number(1-4)

Return value 0:correct 1:wrong

Notice

If this function is used, the axis won't move even if there is manual trigger signal as long as manual quantitative driving and manual continuous driving instructions are not invoked.

☞ **Command type two axes stepping interpolation**

int inp_step_command2(int cardno,int axis1,int axis2,long pulse1,long pulse2);

Function

Set the data of two axes command stepping interpolation

Parameter

cardno	card number
axis1	axis number1
axis2	axis number2
pulse1	pulse of axis1

pulse2	pulse of axis 2

Return value 0:correct 1:wrong

☞ Command type three axes stepping interpolation

int inp_step_command3(int cardno,int axis1,int axis2,int axis3,long pulse1,long pulse2,long pulse3);

Function

Set the data of three axes command stepping interpolation.

Parameter

cardno	card number
axis1	axis number1
axis2	axis number2
axis3	axis number3
pulse1	pulse of axis 1
pulse2	pulse of axis 2
pulse3	pulse of axis 3

Return value 0:correct 1:wrong

☞ Command stepping interpolation driving

int inp_step_move(int cardno);

Function

Execute command stepping interpolation in single step.

Parameter

cardno	card number

Return value 0:correct 1:wrong

Notice

When you execute two axes or three axes command type stepping interpolation in single step, you can sned this command consecutively to execute stepping interpolation consecutively.

☞ Signal type two axes stepping interpolation

int inp_step_signal2(int cardno,int axis1,int axis2,long pulse1,long pulse2);

Function

Set the data of two axes signal type stepping interpolation.

Parameter

cardno	card number
axis1	axis number1
axis2	axis number2
pulse1	pulse of axis 1
pulse2	pulse of axis 2

Return value 0:correct 1:wrong

☞ Signal type three axes stepping interpolation

int inp_step_signal3(int cardno,int axis1,int axis2,int axis3,long pulse1,long pulse2,long pulse3);

Function

Set the data of three axes stepping interpolation

Parameter

cardno	card number

axis1	axis number1
axis2	axis number2
axis3	axis number3
pulse1	pulse of axis 1
pulse2	pulse of axis 2
pulse3	pulse of axis 3

Return value 0:correct 1:wrong

Notice

Stepping interpolation can be triggered when stepping interpolation signal is in low electricity level.

☞ **Stop stepping interpolation**

int inp_step_stop(int cardno,int axis);

Function

Stop executing stepping interpolation.

Parameter

cardno	card number
axis	axis number(1-4)

Return value 0:correct 1:wrong

Notice

Once the stepping interpolation driving is started, the driving is always active state before it is stopped normally. To stop stepping interpolation moving in advance, you need to use stepping interpolation stop command, rather than sudden or decelerated stop.

☞ **Automatic back-to-home**

int home(int cardno,int axis);

Function

Appointed axis executes back-to-home motion automatically

Parameter

cardno	card number
axis	axis number(1-4)

Return value 0:correct 1:wrong

Notice

You need to select valid mode before using automatic back-to-home function.

☞ **Clear back-to-home error**

int clear_home_error (int cardno, int axis);

Function

Clear back-to-home error

Parameter

cardno	card number
axis	axis number(1-4)

Return value 0:correct 1:wrong

☞**Synchronized function setting**

☞ **Signal type single axis follow moving setting**

int set_in_move1(int cardno,int axis,int axis1,long pulse,long pulse1,int logical,int mode);

Function

When the IN signal of active axis is changed into set logic, the driven axis1 follows to move. The motion of active axis depends on set mode.

Parameter

cardno	card number
axis	Active axis number
axis1	Driven axis number1
pulse	Active axis pulses
pulse1	Driven axis1 pulses
logical	Electricity level logic of IN signal (0:low electric 1:high electric)
mode	Running status of active axis (0: persistent,1: appointed pulse)

Return value 0:correct 1:wrong

☞ **Signal type two axes follow moving setting**

int set_in_move2(int cardno, int axis, int axis1,int axis2 ,long pulse,long pulse1,long pulse2,int logical,int mode);

Function

When the IN signal of active axis is changed into set logic, the driven axis1 and axis2 follow to move. The motion of active axis depends on set mode.

Parameter

cardno	card number
axis	Active axis number
axis1	Driven axis number1
axis2	Driven axis number2
pulse	Active axis pulses
pulse1	Driven axis1 pulses
pulse2	Driven axis2 pulses
logical	Electricity level logic of IN signal (0:low electric 1:high electric)
mode	Running status of active axis (0: persistent,1: appointed pulse)

Return value 0:correct 1:wrong

☞ **Signal type three axes follow moving setting**

int set_in_move3(int cardno, int axis, int axis1,int axis2 ,int axis3,long pulse,long pulse1,long pulse2,long pulse3,int logical,int mode);

Function

When the IN signal of active axis is changed into set logic, the driven axis1, axis2 and axis3 follow to move. The motion of active axis depends on set mode.

Parameter

cardno	card number
axis	active axis number
axis1	driven axis number1
axis2	driven axis number2
axis3	driven axis number3
pulse	Active axis pulses
pulse1	Driven axis1 pulses
pulse2	Driven axis2 pulses
pulse3	Driven axis3 pulses

logical	Electricity level logic of IN signal (0:low electric 1:high electric)
mode	Running status of active axis (0: persistent,1: appointed pulse)

Return value 0:correct 1:wrong

☞ Signal type single axis follow stopping setting

int set_in_stop1(int cardno,int axis,int axis1,int logical,int mode);

Function

When the IN signal of active axis is changed into set logic, the driven axis1 stops immediately. The motion of active axis depends on set mode.

Parameter

cardno	card number
axis	active axis number
axis1	driven axis number1
logical	Electricity level logic of IN signal (0:low electric 1:high electric)
mode	Running status of active axis (0: persistent,1: appointed pulse)

Return value 0:correct 1:wrong

☞ Signal type two axes follow stopping setting

int set_in_stop2(int cardno, int axis, int axis1,int axis2,int logical,int mode);

Function

When the IN signal of active axis is changed into set logic, the driven axis1 and axis2 stop immediately. The motion of active axis depends on set mode.

Parameter

cardno	card number
axis	active axis number
axis1	driven axis number1
axis2	driven axis number2
logical	Electricity level logic of IN signal (0:low electric 1:high electric)
mode	Running status of active axis (0: persistent,1: appointed pulse)

Return value 0:correct 1:wrong

☞ Signal type three axes follow stopping setting

int set_in_stop3(int cardno, int axis,int logical,int mode);

Function

When the IN signal of active axis is changed into set logic, the other three axes stop immediately. The motion of active axis depends on set mode.

Parameter

cardno	card number
axis	active axis number
logical	Electricity level logic of IN signal (0:low electric 1:high electric)
mode	Running status of active axis (0: persistent,1: appointed pulse)

Return value 0:correct 1:wrong

☞ Position comparison single axis follow moving setting

int set_comp_pmove1(int cardno, int axis, int axis1, long pulse, long pulse1, int regi, int term);

Function

When active axis reaches appointed position, driven axis1 moves immediately.

Parameter

cardno	card number
axis	active axis number
axis1	driven axis number1
pulse	Active axis pulses
pulse1	drive pulse of driven axis1
regi	Comparison register type (0:comp+;1:comp-)
term	Relational operator (0:>=,1:<)

Return value 0:correct 1:wrong

☞ Position comparison two axes follow moving setting

int set_comp_pmove2(int cardno, int axis, int axis1,axis2, long pulse, long pulse1,long pulse2, int regi, int term);

Function

When active axis reaches appointed position, driven axis1 and axis2 move immediately.

Parameter

cardno	card number
axis	active axis number
axis1	driven axis number1
axis2	driven axis number2
pulse	Active axis pulses
pulse1	Driven axis1 pulses
pulse2	Driven axis2 pulses
regi	Comparison register type (0:comp+;1:comp-)
term	Relational operator (0:>=,1:<)

Return value 0:correct 1:wrong

☞ Position comparison three axes follow moving setting

int set_comp_pmove3(int cardno, int axis, int axis1,axis2,int axis3, long pulse, long pulse1,long pulse2,long pulse3, int regi, int term);

Function

When active axis reaches appointed position, driven axis1, axis2 and axis3 move immediately.

Parameter

cardno	card number
axis	active axis number
axis1	driven axis number1
axis2	driven axis number2
axis3	driven axis number3
pulse	drive pulse of active axis
pulse1	drive pulse of driven axis1
pulse2	drive pulse of driven axis2
pulse3	drive pulse of driven axis2
regi	Comparison register type (0:comp+;1:comp-)
term	Relational operator (0:>=,1:<)

Return value 0:correct 1:wrong

Notice

Position comparison driving is to compare the current position (logical and actual) of active axis with comparison register. If they satisfy the condition of relational operator, the driven axis will run appointed pulses immediately.

☞ Position comparison single axis stop setting

```
int   set_comp_stop1(int cardno, int axis, int axis1, long pulse, int regi, int term,int
                     mode);
```

Function

When active axis reaches appointed position, driven axis1 moves immediately. The motion of active axis depends on set mode.

Parameter

cardno	card number
axis	active axis number
axis1	driven axis number1
pulse	drive pulse of active axis
regi	Comparison register type (0:comp+;1:comp-)
term	Relational operator (0:>=,1:<)
mode	The driving state of active axis (0: persistent,1:stop)

Return value 0:correct 1:wrong

☞ Position comparison two axes stop setting

```
int   set_comp_stop2(int cardno, int axis, int axis1,int axis2, long pulse, int regi, int
                     term,int mode);
```

Function

When active axis reaches appointed position, driven axis1 and axis2 move immediately. The motion of active axis depends on set mode.

Parameter

cardno	card number
axis	active axis number
axis1	driven axis number1
axis2	driven axis number2
pulse	drive pulse of active axis
regi	Comparison register type (0:comp+;1:comp-)
term	Relational operator (0:>=,1:<)
mode	The driving state of active axis (0: persistent,1:stop)

Return value 0:correct 1:wrong

☞ Position comparison three axes stop setting

```
int   set_comp_stop3(int cardno, int axis, long pulse, int regi, int term,int mode);
```

Function

When active axis reaches appointed position, driven axis1, axis2 and axis3 move immediately. The motion of active axis depends on set mode.

Parameter

cardno	card number
axis	active axis number
pulse	drive pulse of active axis
regi	Comparison register type (0:comp+;1:comp-)

term	Relational operator (0:>=,1:<)
mode	The driving state of active axis (0: persistent,1:stop)

Return value 0:correct 1:wrong

☞Composite driving

To provide convenience for the customers, we encapsulated composite driving functions in the basic library functions. These functions mainly integrate speed mode setting, speed parameter setting and motion functions, while absolute motion and relative motion are also considered.

☞ Single axis symmetric relative moving

**int symmetry_relative_move(int cardno, int axis, long pulse, long lspd ,long hspd,
double tacc, long vacc, int mode);**

Function

Refer to current position and perform quantitative motion in symmetric acceleration/deceleration.

Parameter

cardno	card number
axis	axis number(1-4)
pulse	pulse
lspd	low speed
hspd	high speed
tacc	time of acceleration (Unit: sec)
vacc	change rate of acceleration
mode	mode (trapezoid(0) or S-curve(1))

Return value 0:correct 1:wrong

☞ Single axis symmetric relative moving

**int symmetry_absolute_move(int cardno, int axis, long pulse, long lspd ,long hspd,
double tacc, long vacc, int mode);**

Function

Refer to current position and perform quantitative motion in symmetric acceleration/deceleration.

Parameter

cardno	card number
axis	axis number(1-4)
pulse	pulse
lspd	low speed
hspd	high speed
tacc	time of acceleration (Unit: sec)
vacc	change rate of acceleration
mode	mode (trapezoid(0) or S-curve(1))

Return value 0:correct 1:wrong

☞ Single axis asymmetric relative moving

**int unsymmetry_relative_move (int cardno, int axis, long pulse, long lspd ,long hspd,
double tacc, double tdec, long vacc, int mode);**

Function

Refer to current position and perform quantitative motion in asymmetric acceleration/deceleration.

Parameter

cardno	card number
axis	axis number(1-4)
pulse	pulse
lspd	low speed
hspd	high speed
tacc	time of acceleration (Unit: sec)
tdec	time of deceleration (Unit: sec)
vacc	change rate of acceleration/deceleration
mode	mode (trapezoid(0) or S-curve(1))

Return value 0:correct 1:wrong

☞ Single axis asymmetric absolute moving

int unsymmetry_absolute_move (int cardno, int axis, long pulse, long lspd ,long hspd, double tacc, double tdec, long vacc, int mode);

Function

Refer to the position of zero point and perform quantitative motion in asymmetric acceleration/deceleration.

Parameter

cardno	card number
axis	axis number(1-4)
pulse	pulse
lspd	low speed
hspd	high speed
tacc	time of acceleration (Unit: sec)
tdec	time of deceleration (Unit: sec)
vacc	change rate of acceleration
mode	mode (trapezoid(0) or S-curve(1))

Return value 0:correct 1:wrong

☞ Two axes symmetric linear interpolation relative moving

int symmetry_relative_line2(int cardno, int axis1, int axis2,long pulse1, long pulse2, long lspd ,long hspd, double tacc, long vacc, int mode) ;

Function

Refer to current position and perform linear interpolation in symmetric acceleration/deceleration.

Parameter

cardno	card number
axis1	axis number1
axis2	axis number2
pulse1	pulse of axis1
pulse2	pulse of axis2
lspd	low speed
hspd	high speed
tacc	time of acceleration (Unit: sec)
vacc	change rate of acceleration
mode	mode (trapezoid(0) or S-curve(1))

Return value 0:correct 1:wrong

☞ Two axes symmetric linear interpolation absolute moving

int symmetry_absolute_line2(int cardno, int axis1, int axis2,long pulse1, long pulse2,

long lspd ,long hspd, double tacc, long vacc, int mode) ;

Function

Refer to the position of zero point and perform linear interpolation in symmetric acceleration/deceleration.

Parameter

cardno	card number
axis1	axis number1
axis2	axis number2
pulse1	pulse of axis1
pulse2	pulse of axis2
lspd	low speed
hspd	high speed
tacc	time of acceleration (Unit: sec)
vacc	change rate of acceleration
mode	mode (trapezoid(0) or S-curve(1))

Return value 0:correct 1:wrong

☞ **Two axes asymmetric linear interpolation relative moving**

int unsymmetry_relative_line2 (int cardno, int axis1, int axis2, long pulse1, long pulse2, long lspd ,long hspd, double tacc, double tdec, long vacc, int mode) ;

Function

Refer to current position and perform linear interpolation in symmetric acceleration/deceleration.

Parameter

cardno	card number
axis1	axis number1
axis2	axis number2
pulse1	pulse of axis 1
pulse2	pulse of axis 2
lspd	low speed
hspd	high speed
tacc	time of acceleration (Unit: sec)
tdec	time of deceleration (Unit: sec)
vacc	change rate of acceleration
mode	mode (trapezoid(0) or S-curve(1))

Return value 0:correct 1:wrong

☞ **Two axes asymmetric linear interpolation absolute moving**

int unsymmetry_absolute_line2 (int cardno, int axis1, int axis2, long pulse1, long pulse2, long lspd ,long hspd, double tacc, double tdec, long vacc, int mode) ;

Function

Refer to the position of zero point and perform linear interpolation in symmetric acceleration/deceleration.

Parameter

cardno	card number
axis1	axis number1
axis2	axis number2
pulse1	pulse of axis 1
pulse2	pulse of axis 2

lspd	low speed
hspd	high speed
tacc	time of acceleration (Unit: sec)
tdec	time of deceleration (Unit: sec)
vacc	change rate of acceleration
mode	mode (trapezoid(0) or S-curve(1))

Return value 0:correct 1:wrong

☞ Three axes symmetric linear interpolation relative moving

int symmetry_relative_line3(int cardno, int axis1, int axis2, int axis3, long pulse1, long pulse2, long pulse3, long lspd ,long hspd, double tacc, long vacc, int mode);

Function

Refer to current position and perform linear interpolation in symmetric acceleration/deceleration.

Parameter

cardno	card number
axis1	axis number1
axis2	axis number2
axis3	axis number3
pulse1	pulse of axis 1
pulse2	pulse of axis 2
pulse3	pulse of axis 3
lspd	low speed
hspd	high speed
tacc	time of acceleration (Unit: sec)
vacc	change rate of acceleration
mode	mode (trapezoid(0) or S-curve(1))

Return value 0:correct 1:wrong

☞ Three axes symmetric linear interpolation absolute moving

int symmetry_absolute_line3(int cardno, int axis1, int axis2, int axis3, long pulse1, long pulse2, long pulse3, long lspd ,long hspd, double tacc, long vacc, int mode);

Function

Refer to the position of zero point and perform linear interpolation in symmetric acceleration/deceleration.

Parameter

cardno	card number
axis1	axis number1
axis2	axis number2
axis3	axis number3
pulse1	pulse of axis 1
pulse2	pulse of axis 2
pulse3	pulse of axis 3
lspd	low speed
hspd	high speed
tacc	time of acceleration (Unit: sec)
vacc	change rate of acceleration
mode	mode (trapezoid(0) or S-curve(1))

Return value 0:correct 1:wrong

☞ Three axes asymmetric linear interpolation relative moving

int unsymmetry_relative_line3(int cardno, int axis1, int axis2, int axis3, long pulse1, long pulse2, long pulse3, long lspd ,long hspd, double tacc, double tdec, long vacc, int mode);

Function

Refer to current position and perform linear interpolation in symmetric acceleration/deceleration.

Parameter

cardno	card number
axis1	axis number1
axis2	axis number2
axis3	axis number3
pulse1	pulse of axis1
pulse2	pulse of axis2
pulse3	pulse of axis3
lspd	low speed
hspd	high speed
tacc	time of acceleration (Unit: sec)
tdec	time of deceleration (Unit: sec)
vacc	change rate of acceleration
mode	mode (trapezoid(0) or S-curve(1))

Return value 0:correct 1:wrong

☞ Three axes asymmetric linear interpolation absolute moving

Int unsymmetry_absolute_line3(int cardno, int axis1, int axis2, int axis3, long pulse1, long pulse2, long pulse3, long lspd ,long hspd, double tacc, double tdec, long vacc, int mode);

Function

Refer to the position of zero point and perform linear interpolation in symmetric acceleration/deceleration.

Parameter

cardno	card number
axis1	axis number1
axis2	axis number2
axis3	axis number3
pulse1	pulse of axis1
pulse2	pulse of axis2
pulse3	pulse of axis3
lspd	low speed
hspd	high speed
tacc	time of acceleration (Unit: sec)
tdec	time of deceleration (Unit: sec)
vacc	change rate of acceleration
mode	mode (trapezoid(0) or S-curve(1))

Return value 0:correct 1:wrong

☞ Two axes symmetric arc interpolation relative moving

int symmetry_relative_arc(int cardno, int axis1, int axis2, long x, long y, long i, long j, int dir, long lspd ,long hspd, double tacc, long vacc, int mode);

Function

Refer to current position and perform arc interpolation in symmetric acceleration/deceleration.

Parameter

cardno	card number
axis1	axis number1
axis2	axis number2
x、y	Coordinates of arc end point (Refer to current point, that is starting point of arc)
i、j	Centre coordinate (Refer to current point, that is, starting point of arc)
dir	Moving direction (0-Clockwise,1-Anti-clockwise)
lspd	low speed
hspd	high speed
tacc	time of acceleration (Unit: sec)
vacc	change rate of acceleration
mode	mode (trapezoid(0) or S-curve(1))

Return value 0:correct 1:wrong

☞ Two axes symmetric arc interpolation absolute moving

int symmetry_absolute_arc(int cardno, int axis1, int axis2, long x, long y, long i, long j,
int dir, long lspd ,long hspd, double tacc, long vacc, int mode);

Function

Refer to the position of zero point and perform arc interpolation in symmetric acceleration/deceleration.

Parameter

cardno	card number
axis1	axis number1
axis2	axis number2
x、y	Coordinates of arc end point (Refer to current point, that is starting point of arc)
i、j	Centre coordinate (Refer to current point, that is, starting point of arc)
dir	Moving direction (0-Clockwise,1-Anti-clockwise)
lspd	low speed
hspd	high speed
tacc	time of acceleration (Unit: sec)
vacc	change rate of acceleration
mode	mode (trapezoid(0) or S-curve(1))

Return value 0:correct 1:wrong

☞ Two axes asymmetric arc interpolation relative moving

int unsymmetry_relative_arc(int cardno, int axis1, int axis2, long x, long y, long i, long j,
int dir, long lspd ,long hspd, double tacc, double tdec, long vacc, int mode);

Function

Refer to current position and perform arc interpolation in asymmetric acceleration/deceleration.

Parameter

cardno	card number
axis1	axis number1
axis2	axis number2

x、y	Coordinates of arc end point (Refer to current point, that is starting point of arc)
i、j	Centre coordinate (Refer to current point, that is, starting point of arc)
dir	Moving direction (0-Clockwise,1-Anti-clockwise)
lspd	low speed
hspd	high speed
tacc	time of acceleration (Unit: sec)
tdec	time of deceleration (Unit: sec)
vacc	change rate of acceleration
mode	mode (trapezoid(0) or S-curve(1))

Return value 0:correct 1:wrong

☞ Two axes asymmetric arc interpolation absolute moving

int unsymmetry_absolute_arc(int cardno, int axis1, int axis2, long x, long y, long i, long j, int dir, long lspd ,long hspd, double tacc, double tdec, long vacc, int mode);

Function

Refer to the position of zero point and perform arc interpolation in asymmetric acceleration/deceleration.

Parameter

cardno	card number
axis1	axis number1
axis2	axis number2
x、y	Coordinates of arc end point (Refer to current point, that is starting point of arc)
i、j	Centre coordinate (Refer to current point, that is, starting point of arc)
dir	Moving direction (0-Clockwise,1-Anti-clockwise)
lspd	low speed
hspd	high speed
tacc	time of acceleration (Unit: sec)
tdec	time of deceleration (Unit: sec)
vacc	change rate of acceleration
mode	mode (trapezoid(0) or S-curve(1))

Return value 0:correct 1:wrong

☞Switch quantity input/output

☞ Read single input bit

int read_bit(int cardno,int number);

Function

Get the status of single input bit

Parameter

cardno	card number
number	input bit (0-34)

The meanings of input bit number are as follow:

0:X LMT- 1:X LMT+
2:X STOP0 3:X STOP1
4:X STOP2 5:X ALARM
6:Y LMT- 7:Y LMT+
8:Y STOP0 9:Y STOP1

10:YSTOP2 11:Y ALARM
12:Z LMT- 13:Z LMT+
14:Z STOP0 15:Z STOP1
16:Z STOP2 17:Z ALARM
18:A LMT- 19:A LMT+
20:A STOP0 21:A STOP1
22:A STOP2 23:A ALARM
24:X IN 25:X INPOS
26:Y IN 27:Y INPOS
28:Z IN 29:Z INPOS
30:A IN 31: A INPOS
32:EXPP 33:EXPM
34:EMGN

Return value 0:low electric 1:high electric

☞ **Output single bit**

int write_bit(int cardno,int number,int value);

Function

Corresponding port performs output operation.

Parameter

cardno card number
number output bit(0-31)
value 0:low 1:high

Return value 0:correct 1:wrong

☞ **Get the status of output bit**

int get_out(int cardno, int number);

Function

Get the status of output port

Parameter

cardno card number
number Output bit (0-31)

Return value Get the current status of appointed port
0: OFF 1: ON -1:Parameterwrong

Chapter VIII Motion Control Function Library Guide

1. Introduction of ADT8948A1 functions library

ADT-8948A1 functions library is actually the interface for users to operate motion card.Users can control the motion card and perform corresponding functions by invoking interface functions.

The motion card provides motion functions library in DOS and DLL in Windows. The invoking methods of functions library in DOS and Windows are introduced below.

2. Invoking DLL in Windows

The DLL "adt8948.dll" in Windows is written with VC and locates in directory "Development kits\Drivers\DLL" in the CD. Common programing tools in Windows are: VB, VC, C++Builder, VB.NET, VC.NET, Delphi and configuration software LabVIEW.

2.1 Invoke in VC

(1) Create a new project;

(2) Copy the "adt8948.lib" and "adt8948.h" in "Development kits\VC" to the directory of new project;

(3) In the "File view" of working area of the new project, right click the mouse and select "Add Files to Project". In the dialogue box, select "Library Files(.lib)" in file type, search and select "adt8948.lib" and then click "OK". The static library is loaded.

(4) Add "#include 'adt8948.h'" to the declaration part of source program file or header file or global header file "StdAfx.h";

Then, you can invoke the functions in DLL.

Note: Use same method to invoke in VC.NET.

2.2 Invoke in VB

(1) Create a new project;

(2) Copy the "adt8948lib.bas" in "Development kits\VB" to the directory of new project;

(3) Select "Project\Add module" menu and select "Existing" tag in the dialogue box, then, search the "adt8948lib.bas" module file and click "Open".

Then, you can invoke the functions in DLL.

Note: Use same method to invoke in VB.NET.

2.3 Invoke in C++Builder

(1) Create a new project;

(2) Copy the "adt8948.lib" and "adt8948.h" in "Development kits\C++Builder" to the directory of new project;

(3) Select "Project\Add to Project" menu, and select "Library files(*.lib)" in the file type dialogue box, search and select "adt8948.lib" and then click "OK".

(4) Add #include "adt8948.h" to the declaration part of the program.

Then, you can invoke the DLL in the program.

2.4 Invoke in LabVIEW 8

(1) Create a new VI;

(2) Copy the "adt8948a1.llb" and "adt8948.dll" in "Development kits\LabVIEW" to the

directory of new project;

(3) Right click the mouse in the blank area of the program window and the Functions Palette popup. Select "Select a VI.." and select adt8948a1.llb file in the popup window, and then select desired library function in the "Select the VI to Open" window and drag it into the program window.

Then, you can invoke the DLL in the program.

3. Invoking library function in DOS

The functions library in DOS is written with Borland C3.1 and saved in "Development kits\BC" or "Development kits\C" of the CD. The library functions are in large model or huge model. It is suitable for standard C and Borland C3.1 or later.

To invoke functions library in Borland C:

(1) In Borland C, select "Project\Open Project" to create a new project;

(2) Copy the "8948dosh.lib" or "8948dosl.lib" and "adt8948.h" in "Development kits\BC" in the CD to the directory of new project;

(3) Select "Project\Add Item" and select "8948DOSH.LIB" or "8948dos.lib" in the dialogue box, and then click "Add";

(4) Add "#include 'adt8948.h'" declaration to the program file.

Then, you can invoke the library functions in the program.

4. Return values of library function and their meanings

To make sure that users can monitor the normal execution while using library functions, every library function in the library will return the result after execution. Users can check whether the function is invoked successfully according to the return value.

Except "int adt8948_initial(void)" and "int read_bit(int cardno, int number)", the return values of other functions are "0" (normal) and "1" (error).

The meanings of return values are introduced in the following table.

Function name	Return value	Meaning
adt8948_initial	-1	Port driver isn't installed
	-2	PCI slot fault
	0	Motion card isn't installed
	>0	Quantity of motion cards
read_bit	0	Low electricity level
	1	High electricity level
	-1	Card No. error or input bit exceeding limit
All other functions	0	Normal
	1	Error

Note: Return value 1 error is caused by the cardno or axis error in the processs of invoking library functions. The values of card number are 0, 1 and 2, counting form small to big. If there is only one card, the card number should be 0. The axis number should be 1, 2, 3 or 4.

Chapter IX Key Points of Motion Control Development

Most of the problems in the programming of the card are caused by the misunderstanding of the principle of the motion card. Refer to the following section for common problems.

☞Initialize the card

Invoke the adt8948_initial() function first, and make sure that the ADT8948A1 card has been installed properly. Then, set the pulse output mode and working mode of limit switch. These parameters should be set according to the PC configuration. Set only once when the program is initialized.

The “set_range” function is usually set according to maximum pulse frequency and do not change it after this. The ranges of different axes can be different. If the maximum frequency of X axis is 100K, the maximum value of “set_speed” is 8000, the magnification should be 100000/8000=12.5，the range R should be 8000000/12.5=640000, i.e.

set_range(cardno,1, 640000);

To output 100K frequency

set_speed(cardno,1,8000)

To output 10K frequency

set_speed(cardno,1,800)

Minimum output frequency is 1*12.5=12.5Hz

Confirm the R value according to maximum application frequency. Do not change the R value in the application process unless the minimum frequency can’t satisfy your requirement.

Note: Library function “adt8948_initial” is the “door” to ADT8948A1 card. The invoking of other functions is effective only when the motion card is initialized with this function.

☞Speed setting

2.1 Constant speed moving

The parameter configuration is simple. You just need to set the driving speed same as start speed.

Related functions:

set_startv

set_speed

Note: The actual speed is the result that function value multiplys magnification.

2.2 Symmetric linear acceleration/deceleration

This is a common used mode and you need to set start speed, driving speed, acceleration and automatic deceleration.

Related functions:

set_startv

set_speed

set_acc

set_ad_mode (set as linear acceleration/deceleration)
set_dec1_mode (set as symmetric mode)
set_dec2_mode (set as automatic deceleration)

Note: The actual acceleration is the result that acceleration function multiplys magnification and then multiplys 125.

2.3 Asymmetric linear acceleration/deceleration

This mode is mainly used in moving objects in vertical direction. The acceleration time is different from deceleration time and you need to set the value of deceleration.

Related functions:

set_startv
set_speed
set_acc
set_dec
set_ad_mode (set as linear acceleration/deceleration)
set_dec1_mode et as asymmetric mode)
set_dec2_mode (set as automatic deceleration)

2.4 S S-curve acceleration/deceleration

For certain modes with heavy load, S-curve acceleration is used to get better acceleration. In this case, you need to set the values of acceleration/deceleration. The calculation of acceleration/deceleration has a big influence on the shape of S-curve. Refer to the following examples.

Related functions:

set_startv
set_speed
set_acc
set_acac
set_ad_mode (set as S-curve acceleration/deceleration)
set_dec2_mode (set as automatic deceleration)

2.5 Manual deceleration

Manual deceleration is used only when automatic deceleration can't be used normally, e.g. acceleration/deceleration driving of arc interpolation and continuous interpolation. In this case, you need to calculate the manual deceleration point. Refer to the following examples.

2.6 Interpolation speed

For interpolation speed, you just need to set the parameter of first axis. The first axis is the axis1 in the interpolation function parameters of basic library function drivings.

The interpolation is usually driven in constant speed. Therefore, you just need to set start speed and driving speed. You can also use acceleration/deceleration interpolation if you set valid X axis parameter. However, S-curve interpolation can't be used in arc interpolation and continuous interpolation.

The interpolation deceleration is disabled by default. If acceleration/deceleration interpolation is used, there is only acceleration process in the driving. This is suitable for continuous interpolation. If single interpolation is used, you need to enable interpolation deceleration before driving.

The aforesaid settings are to exert the functions of the card. Actually, many functions are necessary only when the speed and effect requirements are very high. In this case, the setting is complicated but necessary.

☞STOP0, STOP1 and STOP2 signal

Every axis has STOP0, STOP1 and STOP2 signals, therefore, there are 12 STOP signals totally. These signals are mainly used in back-to-home operation. The back-to-home mode can use either one signal or several signals. Please note that this signal is decelerated stop. For high speed resetting, you can add one deceleration switch before home switch, i.e. use two STOP signals (one for home switch and the other for deceleration switch). You can also use one signal only. In this case, when the machine receives STOP signal, it stops in deceleration, then, moves to opposite direction in constant speed and stops when receives the signal again.

STOP2 has a special function, i.e. if the setting is valid, the actual position counter will be cleared by STOP2 if you use STOP2 signal to stop. This is to ensure that the value of the counter is 0 in home position. Other motion cards reset the counter with software when the driving is stopped. In servo driving process, even if the pulse output is stopped, it will move forward a little as there are still pulses acclumulated in the servo, thus the position error occurs. With the aforesaid method of the card, it is normal if the actual position isn't 0 after resetting. It isn't necessary to invoke function and reset.

☞Servo signal

Servo in-position signal and servo alarm signal are valid only when the signals have been connected. If servo in-position signal is activated before it is connected, the driving won't be able to stop, because the in-position signal is the symbol to stop driving.

Other servo signals, e.g. servo ON signal and clear alarm signal, can be driven with general output signal.

Chapter Ⅹ Examples of Motion Control Developing and Programming

Results of all motion control functions will be returned immediately. The motion is controlled by motion card when the drive command is sent, and at this point, upper computer software of the user not only could monitor the whole motion in real time, but also force the motion to stop.

Note: It is not allowed to send new drive command to the axis when it is working, otherwise, it will quit the last drive and carry out the next drive command.

Although there are many kinds of programming languages, they will all come to one regarding its nature. In all, there are “Three structures and one thought”. Three structures are sequence structure, cyclic structure and branch structure that are stressed in all programming languages, and one thought is the algorithm and module division that are used to complete the design job, which is the stress and difficulty of the whole program design.

In order to ensure the generality, normative, extensibility and convenient maintenance of a program, we put an eye on the project design and divide the following examples into several modules as shown below: Motion control module (Envelop the library function offered by motion card), function-realization module (Code snippet that fits the specific technics), monitoring module and stop-processing module.

We will briefly introduce the application of ADT-8948A1 card function library in VB and VC programming languages as below. For other programming languages, please refer to the example program of VB and VC.

☞Example program of VB

1.1 PREPARATION

(1) Create a new program and save it as “test.vbp”;

(1) Add “ADT8948.bas” module to the program according to the aforesaid method.

1.2 Motion control module

(1) Add a new module to the program and save it as “ctrlcard.bas”;

(2) Firstly, self-define the initialization function of motion card in the motion control module and initialize the library function that needed to be enveloped in initialization function;

(3) Continue to self-define related motion control function, such as speed setting function, single-axis motion function and interpolation function etc;

(4) source code of ctrcard.bas：

```
'****************** Motion control module *************************
   ' For developing an application system of great generality, extensibility and 'convenient maintenance easily and swiftly, we envelop all the library functions 'by category basing on the card function library
'************************************************************

Public Result As Integer          'return value
Const MULTIPLE = 5               ' ratio
Const MAXAXIS = 4               ' max axis number
```

```
'****************** Initialization function **************************
 ' This function contains library functions that are usually used in the initialization 'of the motion
   card; it is the base of invoking other functions, so it must be 'invoked first in the example
   program.Return value<=0 means the initialization 'is failed, while return value >0 means the
   initialization is successful.
'***********************************************************
Public Function Init_Card() As Integer
        Result = adt8948_initial                    ' Initialization function of the card
        If Result <= 0 Then
              Init_Card = Result
              Exit Function
        End If
        For i = 1 To MAXAXIS
              set_range 0, i, CLng(8000000 / 5)        ' Set the range, set the initial 'ratio to be 5
              set_command_pos 0, i, 0                   ' Reset logical counter
              set_actual_pos 0, i, 0                    ' Reset actual position counter
              set_startv 0, i, 1000                     ' Set the start velocity
              set_speed 0, i, 1000                      ' Set the driving speed
              set_acc 0, i, 625                         ' Set the acceleration
        Next i
        Init_Card = Result
End Function
'***************** Function of releasing control card *****************
This function contains the library function that releases the control card; it should be invoked at
the end of the program
**********************************************************
Public Function End_Board()
        Result = adt8948_end()
        End_Board = Result
End Function'
'****************** Set stop0 signal mode ******************'
      Set mode of stop0 input signal '
    Parameter:         axis number
                        0—ineffective    1—effective
                logic      0—low level effective    1—high level effective
      Defaule : ineffective
      Return    0：Correct 1：   Wrong
************************************************************
Public Function Setup_Stop0Mode(ByVal axis As Integer, ByVal Value As Integer, ByVal logic
As Integer) As Integer
        Setup_Stop0Mode = set_stop0_mode(0, axis, Value, logic)
End Function
'*******************Set stop1 signal mod*****************
    Set mode of stop1 input signal
      para：    axis—axis number
                 value      0—ineffective    1—effective
              logic     0—low level effective   1—high level effective
        Defaule : ineffective
        Return   0：Correct                      1：   Wrong
    ************************************************************
    Public Function Setup_Stop1Mode(ByVal axis As Integer, ByVal Value As Integer, ByVal logic As Integer) As Integer
        Setup_Stop1Mode = set_stop1_mode(0, axis, Value, logic)
    End Function
    '********************* Set stop2 signal mode ***************'
        Set mode of stop2 input signal
        para：  axis—axis number
               value     0—ineffective   1—effective
              logic     0—low level effective   1—high level effective
        Defaule : ineffective
        Return   0：Correct                      1：   Wrong
    ************************************************************
    Public Function Setup_Stop2Mode(ByVal axis As Integer, ByVal Value As Integer, ByVal logic As Integer) As Integer
        Setup_Stop2Mode = set_stop2_mode(0, axis, Value, logic)
    End Function

    '******************** Set the actual position counter *******************
cardno       Card number.
axis          Axis number（1-4）
value          Pulse input mode
0：             A/B pulse input        1：Up/Down (PPIN/PMIN) pulse input
dir          Counting direction
0：            A leads B or PPIN pulse input up count
             B leads A or PMIN pulse input down count
1：             B leads A or PMIN pulse input up count
              A leads B or PPIN pulse input down count
freq   Frequency multiplication of A/B pulse input，  Invalid for Up/Down pulse input
0: 4×        1: 2×             2: 1×
    Initialization status: A/B pulse input，direction: 0，Frequency multiplication: 4
    '************************************************************
    Public Function Actualcount_Mode(ByVal axis As Integer, ByVal Value As Integer, ByVal dir As Integer, ByVal freq As Integer) As Integer
        Result = set_actualcount_mode(0, axis, Value, dir, freq)
        Actualcount_Mode = Result
    End Function

    '********************** Set mode of pulse output *******************
    This function is used to set the working mode of pulse
    para：axis-axis number，  value-pulse mode 0—pulse+pulse 1—pulse + direction
      Return=0 correct，Return=1 wrong
     Default mode: Pulse + direction, with positive logic pulse
    Default pulse mode is Pulse + direction
```

```
        This program adopts default positive logic pulse and positive logic of direction output
signal
'**********************************************************
Public Function Setup_PulseMode(ByVal axis As Integer, ByVal Value As Integer) As Integer
        Setup_PulseMode = set_pulse_mode(0, axis, Value, 0, 0)
End Function
*********************set limit signal mode**********************'
      set the mode of nLMT signal input along positive/ negative direction
      para：   axis—axis number
                  value     0: sudden stop effective           1: decelerate stop effective
                  logic       0: low level effective  1: high level ineffective
      Default mode: Apply positive and negative limits with low level
      Return    0：Correct                         1：    Wrong
***********************************************************
Public Function Setup_LimitMode(ByVal axis As Integer, ByVal Value As Integer, ByVal logic
As Integer) As Integer
        Setup_LimitMode = set_limit_mode(0, axis, Value, logic)
End Function

*************** Set COMP + register as software limit ***************'
cardno          card number
axis       axis number（1-4）
value          0：ineffective                 1：effective
Return         0：Correct                     1：Wrong
Default mode：ineffective
Notice：Software position limiting always adopts acceleration to stop.
 Calculating value may be over set up value. Within setup sphere it must be considered.
'**********************************************************
Public  Function  Setsoft_LimitMode1(ByVal  axis  As  Integer,  ByVal  Value  As  Integer)  As
Integer
        Result = set_softlimit_mode1(0, axis, Value)
        Setsoft_LimitMode1 = Result
End Function

'************** Set COMP - register as software limit ***************'
cardno          card number
axis       axis number（1-4）
value          0：ineffective                    1：effective
Return          0：Correct                  1：Wrong
Default mode：ineffective
Notice：Software position limiting always adopts acceleration to stop.
 Calculating value may be over set up value. Within setup sphere it must be considered.
***********************************************************
Public  Function  Setsoft_LimitMode2(ByVal  axis  As  Integer,  ByVal  Value  As  Integer)  As
Integer
        Result = set_softlimit_mode2(0, axis, Value)
        Setsoft_LimitMode2 = Result
```

```
End Function
```

************** Set COMP+/-register as software limit **************'

cardno card number

axis axis number（1-4）

value 0：logic position counter 1：real position counter

Return 0：Correct 1：Wrong

Default mode : logic position counter

This function is of comparative object for setup of software limiting position.

****

```
Public Function Setsoft_LimitMode3(ByVal axis As Integer, ByVal Value As Integer) As Integer
        Result = set_softlimit_mode3(0, axis, Value)
        Setsoft_LimitMode3 = Result
End Function
```

'************ Setting of servo in-position signal nINPOS ***************

ardno card number

axis axis number（1-4）

value 0：ineffective 1：effective

logic 0:Effective when low electric level 1:Effective when low electric level

Return 0：Correct 1：Wrong

Default mode : ineffective, low electric level is effective

'***

```
Public Function Inpos_Mode(ByVal axis As Integer, ByVal Value As Integer, ByVal logic As Integer) As Integer
        Result = set_inpos_mode(0, axis, Value, logic)
        Inpos_Mode = Result
End Function
```

'************* Setting of servo alarm signal nALARM **************

cardno card number

axis axis number（1-4）

value 0：ineffective 1：effective

logic 0:Effective when low electric level1:Effective when low electric level

Return 0:Correct 1:Wrong

Default mode : ineffective, low electric level is effective

'***

```
Public Function Setup_AlarmMode(ByVal axis As Integer, ByVal Value As Integer, ByVal logic As Integer) As Integer
        Result = set_alarm_mode(0, axis, Value, logic)
        Setup_AlarmMode = Result
End Function
```

********************** Set the velocity module *********************

Check whether it is in constant speed or acceleration/deceleration according to the parameter value.

Set the range, which is the parameter that determines the ratio

Set the start velocity, driving velocity and acceleration of axis

Parameter：axis　Axis number.
startv　start velocity
speed　driving velocity
add　acceleration
dec　deceleration
ratio　ratio
mode　mode

Return=0 correct，Return=1 wrong

```
**********************************************************
Public Function Setup_Speed(ByVal axis As Integer, ByVal startv As Long, ByVal speed As Long, ByVal add As Long, ByVal Dec As Long, ByVal ratio As Long, ByVal mode As Integer) As Integer
   If (startv - speed >= 0) Then                    'constant-speed motion
        set_range 0, axis, 8000000 / ratio
        Result = set_startv(0, axis, startv / ratio)
        set_speed 0, axis, startv / ratio
    Else                                             'acceleration/ deceleration
            Select Case mode
        Case 0
            set_dec1_mode 0, axis, 0                 'Set symmetry type
            set_dec2_mode 0, axis, 0               'Set automatic deacceleration
            Result = set_range(0, axis, 8000000 / ratio)
            set_startv 0, axis, startv / ratio
            set_speed 0, axis, speed / ratio
            set_acc 0, axis, add / 125 / ratio
            set_ad_mode 0, axis, 0       Set as liner acceleration/deceleration
         Case 1
             set_dec1_mode 0, axis, 1                 ' Asymmetry
             set_dec2_mode 0, axis, 0             ' Set automatic deceleration
             Result = set_range(0, axis, 8000000 / ratio)
             set_startv 0, axis, startv / ratio
             set_speed 0, axis, speed / ratio
             set_acc 0, axis, add / 125 / ratio
             set_dec 0, axis, Dec / 125 / ratio
            set_ad_mode 0, axis, 0      'set liner acceleration/deceleration type
          Case 2
              Dim time As Double
              Dim addvar As Double
              Dim k As Long
             time = (speed - startv) / (add / 2)
             addvar = add / (time / 2)
             k = (62500000 / addvar) * ratio
             set_dec2_mode 0, axis, 0                 automatic deacceleration
             Result = set_range(0, axis, 8000000 / ratio)
```

```
                set_startv 0, axis, startv / ratio
                set_speed 0, axis, speed / ratio

                set_acc 0, axis, add / 125 / ratio
                set_acac 0, axis, k
                set_ad_mode 0, axis, 1            choose S-curve A/D type
                Setup_Speed = Result
           End Select
       End If
  End Function
  '******************** single-axis motion ************************
      ' This function is to drive the single axis
      ' para：   axis-axis number，value-pulse of motion
      ' Return=0 correct，Return=1 wrong
  '***********************************************************
  Public Function Axis_Pmove(ByVal axis As Integer, ByVal Value As Long) As Integer
      Result = pmove(0, axis, Value)
      Axis_Pmove = Result
  End Function
  '*************single-axis continuous motion ********************
      ' This function is to continuous drive the single axis
      ' axis-axis number，value-pulse of motion
      ' value-0:positive direction     1:negative direction
      ' Return=0 correct，Return=1 wrong
  '*******************************************************
  Public Function Axis_Cmove(ByVal axis As Integer, ByVal Value As Long) As Integer
      Result = continue_move(0, axis, Value)
      Axis_Cmove = Result
  End Function
  '**************two axes interpolation function ******************
     Two axes linear interpolation
Parameter
       cardno        card number
       axis1         Interpolation axis number1(1-4，indicate :X、Y、Z、A)
       axis2         Interpolation axis number2(1-4，indicate :X、Y、Z、A)
       pulse1          Moving distance of axis1
       pulse2          Moving distance of axis2
  Return value             0:correct               1:wrong
  Notice
  The interpolation speed takes the speed of the axis with smaller axis number between axis1
  and axis2 as the standard.
  '******************************************************
  Public Function Interp_Move2(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal value1
  As Long, ByVal value2 As Long) As Integer
      Result = inp_move2(0, axis1, axis2, value1, value2)
```

```
        Interp_Move2 = Result
End Function
'**************3-axis interpolation ************************
      Function:Three axes linear interpolation
Parameter
        cardno          card number
        axis1           Interpolation axis number1(1-4， indicate :X、 Y、 Z、 A)
        axis2           Interpolation axis number2(1-4， indicate :X、 Y、 Z、 A)
        axis3           Interpolation axis number3(1-4， indicate :X、 Y、 Z、 A)
        pulse1          Moving distance of axis1
        pulse2          Moving distance of axis2
Return value               0:correct                1:wrong
Notice
The interpolation speed takes the speed of the axis with smallest axis number among axis1,
axis2 and axis3 as the standard (axis1).
'*************************************************************
Public Function Interp_Move3(ByVal axis1 As Long, ByVal axis2 As Long, ByVal axis3 As
Long, ByVal value1 As Long, ByVal value2 As Long, ByVal value3 As Long) As Integer
        Result = inp_move3(0, axis1, axis2, axis3, value1, value2, value3)
        Interp_Move3 = Result
End Function
************** Clockwise CW arc interpolation function ***************
axis1,axis2          1: x    2:y    3:z    4:a
   x,y             End position of arc interpolation (relative to starting point)
  i,j       Center point position of arc interpolation (relative to starting point)
  return          0： Correct      1： False
***********************************************************
Public Function Interp_Arc(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal X As Long,
ByVal Y As Long, ByVal i As Long, ByVal j As Long) As Integer
          Interp_Arc = inp_cw_arc(0, axis1, axis2, X, Y, i, j)
End Function
*********** Counterclockwise CCW arc interpolation function *******
'axis1,axis2         1： X       2： Y       3： Z      4： A
 x,y       End position of arc interpolation (relative to starting point)
 i,j       Center point position of arc interpolation (relative to starting point)
return          0： Correct       1： False
*************************************************************
Public Function Interp_CcwArc(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal X As
Long, ByVal Y As Long, ByVal i As Long, ByVal j As Long) As Integer
          Interp_CcwArc = inp_ccw_arc(0, axis1, axis2, X, Y, i, j)
End Function
************* Setup of counter's variable circle function ***************
axis              axis number （1-4）
value             0: ineffective        1: effective
return          0： Correct       1： False
```

```
default mode: ineffective
*********************************************************
Public Function SetCircle_Mode(ByVal axis As Integer, ByVal Value As Integer) As Integer
        Result = set_circle_mode(0, axis, Value)
        SetCircle_Mode = Result
End Function
************* setup of wave filter time constant of input signal ***********
    number    input type
    1:          LMT +、LMT - 、STOP0、STOP1
    2:        STOP2
    3:          nINPOS、nALARM
    4:            nIN
      Set the filtering state of the four types input signals above
    value           0: Wave filter is ineffective                 1: Wave filter is effective
    default mode: ineffective
***********************************************************
Public Function Setup_InputFilter(ByVal axis As Integer, ByVal number As Integer, ByVal Value As Integer) As Integer
        Result = set_input_filter(0, axis, number, Value)
        Setup_InputFilter = Result
End Function
'************ setup of wave filter time constant of input signal *********
'axis          axis number（1-4）
'value         maximum noise scope deleted ,      delay of input signal
'*********************************************************
Public Function Setup_FilterTime(ByVal axis As Integer, ByVal Value As Integer) As Integer
        Result = set_filter_time(0, axis, Value)
        Setup_FilterTime = Result
End Function
'******************** set lockmode ********************
function:lock the logical position and real position for all axis
para:
     axis—reference axis
     regi—register mode      |0:logical position
                              |1:real position
     logical—level signal |0: from high to low
                              |1:from low to high
retutrn                 0：correct                   1：wrong
Note: Use IN signal of specific axis as the trigger signal
*********************************************************
Public Function Setup_LockPosition(ByVal axis As Integer, ByVal regi As Integer, ByVal logical As Integer) As Integer
      Result = set_lock_position(0, axis, regi, logical)
      Setup_LockPosition = Result
End Function
```

```
'****************** Stop the axis drive ********************
' This function is used to immediately stop or decelerate to stop the axis drive
para：axis-axis number、mode-stop mode(0－sudden stop, 1－decelerate stop)
Return=0 correct，Return=1 wrong
'**********************************************************
Public Function StopRun(ByVal axis As Integer, ByVal mode As Integer) As Integer
     If mode = 0 Then
          Result = sudden_stop(0, axis)
     Else
          Result = dec_stop(0, axis)
     End If
End Function
'****************** get status of motion *********************
 get status of single-axis motion or interpolation
 para：axis-axis number
       value-Indicator of motion status
        (0：Drive completed,Non-0: Drive in process)
       mode(0-single-axis motion，1－interpolation)
Return=0 correct，Return=1 wrong
***********************************************************
Public Function Get_MoveStatus(ByVal axis As Integer, Value As Integer, ByVal mode As Integer) As Integer
     If mode = 0 Then            ' status of single-axis motion
          GetMove_Status = get_status(0, axis, Value)
     Else                        ' status of interpolation
          GetMove_Status = get_inp_status(0, Value)
     End If
End Function
**************** get synchronous action state ******************
 function:get synchronous action state
 para:
      cardno          card number
      axis            axis number
      status—0|haven't run synchronous
              1|run synchronous
 retutrn           0：correct          1：wrong
 Note: This function could tell whether the position lock has been executed
**********************************************************/
Public Function Get_LockStatus(ByVal axis As Integer, status As Integer) As Integer
     Result = get_lock_status(0, axis, status)
     Get_LockStatus = Result
End Function
********************* home diving status**********************
unction:get the information of home position,judge the error message
pare：
```

```
        cardno          card number
        axis            axis number
         status—drive status 0|finish driving
                                  1|driving
        err—error message      0|correct
                                  1|wrong
    retutrn              0：correct              1：wrong
    ****************************************************************/
    Public Function Get_HomeStatus(ByVal axis As Integer, status As Integer, err As Integer) As
    Integer
         Result = get_home_status(0, axis, status, err)
         Get_HomeStatus = Result
    End Function
    ************ Get the error information of home position return *******
Function: Get the error information of home position return
err—error message
                  0|correct
              not 0|wrong
                  D0: comp+Limit
                  D1: comp-Limit
                  D2: LMT+Limit
                  D3: LMT-Limit
                  D4: Servo alarm
                  D5: Emergency stop
                  D6:Z-phase signal arrives ahead of time
    retutrn              0：correct              1：wrong
    *****************************************************************
    Public Function Get_HomeError(ByVal axis As Integer, err As Integer) As Integer
         Result = get_home_error(0, axis, err)
         Get_HomeError = Result
    End Function
    ******************deceleration enable********************
    This function is to enable the deceleration during the driving
    retutrn              0：correct              1：wrong
    ****************************************************************
    Public Function AllowDec() As Integer
         Result = inp_dec_enable(0)
         AllowDec = Result
    End Function
    ********************* deceleration disable **********************
    This function is to disable the deceleration during the driving
    retutrn              0：correct              1：wrong
    ****************************************************************
    Public Function ForbidDec() As Integer
         Result = inp_dec_disable(0)
         ForbidDec = Result
 End Function
 *************** Get the error stop information of axis ******************
This function is to get the stop information of axis
value: Index of error status    0：No error    1：Value of two bytes
 retutrn                0：correct               1：wrong
 ***********************************************************
 Public Function Get_ErrorInf(ByVal axis As Integer, Value As Integer) As Integer
         Result = get_stopdata(0, axis, Value)
         Get_ErrorInf = Result
 End Function
 ********** Get the status of continuous interpolation ***************
  This function is to get the writable status of continuous interpolation
   value：Index of interpolation status    0: Unwritable    1: Writable
 retutrn                0：correct               1：wrong
 ************************************************************
 Public Function Get_AllowInpStatus(Value As Integer) As Integer
         Result = get_inp_status2(0, Value)
         Get_AllowInpStatus = Result
 End Function
 ****************** Set the mode of deceleration ********************
 deceleration for symmetry/asymmetry/automatic/manual
 mode1: 0:symmetry acceleration/ deceleration
             1:asymmetry acceleration/ deceleration
 mode2: 0: automatic                 1:manual
 retutrn                0：correct               1：wrong
 **********************************************************
 Public Function Set_DecMode(ByVal axis As Integer, ByVal mode1 As Integer, ByVal mode2
 As Integer) As Integer
         Dim Result1 As Integer
         Dim Result2 As Integer
         Result1 = set_dec1_mode(0, axis, mode1)
         Result2 = set_dec2_mode(0, axis, mode2)
         Set_DecMode = Result1 & Result2
 End Function
 '**************** Set the mode of deceleration ******************
 '     This function is to set the symmetrical/dissymmetrical and automatic/manual deceleration
 '   retutrn                0：correct               1：wrong
 '**********************************************************
 Public Function Set_DecPos(ByVal axis As Integer, ByVal Value As Long, ByVal startv As
 Long, ByVal speed As Long, ByVal add As Long) As Integer
         Dim addtime As Double
         Dim DecPulse As Long
         addtime = (speed - startv) / add
         DecPulse = ((startv + speed) * addtime) / 2
```

```
        Result = set_dec_pos(0, axis, Value - DecPulse)
        Set_DecPos = Result
End Function
******************* Read the input point **********************
   This function is to read the single input point
   para：number- Input points (0 ~ 34)
  Return：0 — low level，1 — high level，-1 — error
***************************************************************
Public Function Read_Input(ByVal number As Integer) As Integer
        Read_Input = read_bit(0, number)
End Function
********************* Single point output function **************
 This function is to output single point signal
 Para: number- output port (0 ~ 31),
               value 0-low level、1—high level
  Return=0 correct，Return=1 wrong
***************************************************************
Public Function Write_Output(ByVal number As Integer, ByVal Value As Integer) As Integer
        Write_Output = write_bit(0, number, Value)
End Function
**************** Position counter setting ********************
 This function is to set the logical position and actual position
   para：axis- axis number,           pos- Value of position
   mode 0—Set the logical position,    Non-0—Set the actual position
  Return=0 correct，Return=1 wrong
**************************************************************
Public Function Setup_Pos(ByVal axis As Integer, ByVal pos As Long, ByVal mode As Integer)
As Integer
        If mode = 0 Then
                Result = set_command_pos(0, axis, pos)
        Else
                Result = set_actual_pos(0, axis, pos)
        End If
End Function
******************** COMP+register setting ****************
cardno              card number
axis       axis number
 value              range（-2147483648~+2147483647）
retutrn                  0：correct                   1：wrong
**************************************************************
Public Function Setup_Comp1(ByVal axis As Integer, ByVal Value As Long)
        Result = set_comp1(0, axis, Value)
        Setup_Comp1 = Result
End Function
****************** set COMP- register ********************
```

cardno card number

axis axis number

value range（-2147483648~+2147483647）

retutrn 0：correct 1：wrong

****

```
Public Function Setup_Comp2(ByVal axis As Integer, ByVal Value As Long)
    Result = set_comp2(0, axis, Value)
    Setup_Comp2 = Result
End Function
```

***************** Set the mode of fixed linear speed******************

Function: Set the mode of fixed linear speed

para:

cardno card number

mode—0| not use the fixed linear speed 1| Use the fixed linear speed

retutrn 0：correct 1：wrong

Note: Linear speed means vector speed; fixed linear speed could ensure the composite speed to be fixed during the interpolation

**/

```
Public Function Setup_VectorSpeed(ByVal mode As Integer) As Integer
    Result = set_vector_speed(0, mode)
    Setup_VectorSpeed = Result
End Function
```

******************* set home position mode ***********************

Function: Set the back-to-home mode of specific axis

para:

logical0—stop0|0: stop with low level
|1: stop with high level
|-1: ineffective

logical1—stop1|0: stop with low level
|1: stop with high level
|-1: ineffective

logical2—stop2|0: stop with low level
|1: stop with high level
|-1: ineffective

0ffset— 0| Do not offset from home 1| Offset from home

dir0—Direction |0:Positive |1:Negative

dir1—Direction |0:Positive |1:Negative

dir2—Direction |0:Positive |1:Negative

offsetdir—Offset direction |0:Positive |1:Negative

speed—Low search speed, it's required to be lower than start velocity of high speed

reset—Reset the counter?|0: Yes |1: No

return 0：correct 1：wrong

Note:

(1) Zero-return is divided into four steps:

|The first step: swiftly approach stop0 (logical0 close origin setup);

|The second step: slowly approach stop1 (logical1 origin setup);
|The third step: slowly approach stop2 (logical2 encoder Z-phase);
|The fourth step: Deviation range (For working origin);

(2) The above four steps can decide whether to be carried out by choosing among logical0, logical1、logical2 and offset

(3) Able to use a proximity switch to act as several signals

```
*************************************************************/
Public Function Setup_HomeMode(ByVal axis As Integer, ByVal speed As Long, ByVal
logical0 As Integer, ByVal logical1 As Integer, ByVal logical2 As Integer, ByVal offset As
Integer, ByVal dir0 As Integer, ByVal dir1 As Integer, ByVal dir2 As Integer, ByVal offsetdir As
Integer, ByVal clear As Integer, ByVal pulse As Long) As Integer
        Result = set_home_mode(0, axis, speed, logical0, logical1, logical2, offset, dir0, dir1, dir2,
offsetdir, clear, pulse)
        Setup_HomeMode = Result
End Function
'******************** Get the motion information *******************
```

'This function is to reflect a feedback of current logic position, actual position and operating speed of axis

para：axis-Axis number., LogPos-Logical position, ActPos-Actual position, Speed- driving speed

Return=0 correct，Return=1 wrong

```
'*********************************************************/
Public Function Get_CurrentInf(ByVal axis As Integer, LogPos As Long, ActPos As Long,
speed As Long) As Integer
        Result = get_command_pos(0, axis, LogPos)
                get_actual_pos 0, axis, ActPos
                get_speed 0, axis, speed
        Get_CurrentInf = Result
End Function
*********************get lock position*******************
```

Function: Get the locked position

para:

cardno	card number	
axis	axis number	
pos	lock position	
Return	0：Correct	1：Wrong

```
*************************************************************
Public Function Get_LockPosition(ByVal axis As Integer, pos As Long) As Integer
        Result = get_lock_position(0, axis, pos)
        Get_LockPosition = Result
End Function
'************* Function of library function's version number ***********
```

'Function: Get the version of library function

para:

ver-Version number (The first two numbers are the major version numbers, the last two numbers are secondary version numbers)

```
    Return        0：Correct                    1：Wrong
    '***********************************************************
    Public Function Get_LibVision(ver As Integer) As Integer
        Result = get_lib_vision(ver)
        Get_LibVision = Result
    End Function
    ************ Quantitative drive function of external signal ************
function: Quantitative drive function of external signal
para:
    cardno          card number
    axis            axis number
    pulse           pulse
Return        0：Correct                    1：Wrong
    Note: (1) Send out quantitative pulse, but the drive does not start immediately until the external signal level changes
         (2)Ordinary button and handwheel are acceptable.
    ***********************************************************/
    Public Function Manu_Pmove(ByVal axis As Integer, ByVal pulse As Long) As Integer
        Result = manual_pmove(0, axis, pulse)
        Manu_Pmove = Result

    End Function
    *************Continuous drive function of external signal ***********
function: Continuous drive function of external signal
para:
    cardno          card number
    axis            axis number
Return        0：Correct                    1：Wrong
    Note: (1) Send out fixed pulse, but the drive does not start immediately until the level of external signal changes
              (2) Ordinary button and handwheel are acceptable.
    ***********************************************************/
    Public Function Manu_Continue(ByVal axis As Integer) As Integer
        Result = manual_continue(0, axis)
        Manu_Continue = Result
    End Function
    **********Shut down the enabling of external signal drive ************
function: Shut down the enabling of external signal drive
para:
    cardno          card number
    axis            axis number
    Return        0：Correct                    1：Wrong
    ***********************************************************/
    Public Function Manu_Disable(ByVal axis As Integer) As Integer
        Result = manual_disable(0, axis)
        Manu_Disable = Result
    End Function
    '********* Command type two axes stepping interpolation setup ******
'Function:Set the data of two axes command stepping interpolation
para:
    The same with the 2-axis interpolation function
Return        0： Correct                1： Wrong
        Functional description: Send out data of stepping interpolation, the drive does not start but
    waiting for drive command function
    '***********************************************************
    Public Function Inp_Command2(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal pulse1
    As Long, ByVal pulse2 As Long) As Integer
        Result = inp_step_command2(0, axis1, axis2, pulse1, pulse2)
        Inp_Command2 = Result
    End Function
    ********* Command type three axes stepping interpolation setup *********
   Function: Set the data of three axes command stepping interpolation
   para:
    The same with three-axis interpolation function
     Return        0： Correct                1： Wrong
        Functional description: Send out data of stepping interpolation, the drive does not work but
    waiting for drive command function
    '***********************************************************
    Public Function Inp_Command3(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal axis3
    As Integer, ByVal pulse1 As Long, ByVal pulse2 As Long, ByVal pulse3 As Long) As Integer
        Result = inp_step_command3(0, axis1, axis2, axis3, pulse1, pulse2, pulse3)
        Inp_Command3 = Result
    End Function
    '************ Stepping interpolation command driving function ********
   function: Execute command stepping interpolation in single step
   para:
    cardno         card number
   Return        0： Correct                1： Wrong
        Functional description: Drive the stepping motion in the form of command according to the
    parameter of the set function
    '***********************************************************/
    Public Function Inp_StepMove() As Integer
        Result = inp_step_move(0)
        Inp_StepMove = Result
    End Function
    ********** Setting function of 2-axis signal stepping interpolation *******
Function: Set the data of 2-axis signal stepping interpolation
parameter:
    The same with the interpolation function of 2-axis
    Return        0： Correct                1： Wrong
```

Functional description: Send out data of stepping interpolation, the drive does not work but waiting for the external signal level to drop to low level

```
***************************************************************
Public Function Inp_Signal2(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal pulse1 As Long, ByVal pulse2 As Long) As Integer
        Result = inp_step_signal2(0, axis1, axis2, pulse1, pulse2)
        Inp_Signal2 = Result
End Function
******** Setting function of three-axis signal stepping interpolation ******
```

function: Set the data of three-axis stepping interpolation

para:

The same with the three-axis interpolation function

return value	0：correct	1：wrong

Functional description: Send out data of stepping interpolation, the drive does not work but waiting for the external signal level to drop to low level

```
***************************************************************
Public Function Inp_Signal3(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal axis3 As Integer, ByVal pulse1 As Long, ByVal pulse2 As Long, ByVal pulse3 As Long) As Integer
        Result = inp_step_signal3(0, axis1, axis2, axis3, pulse1, pulse2, pulse3)
                Inp_Signal3 = Result
End Function
******************** Stop stepping interpolation *******************
```

function: Stop the stepping interpolation

para:

cardno	card number	
axis	axis number	
return value	0：correct	1：wrong

Note: Axis that is in the state of stepping interpolation must carry out the stop command of stepping interpolation before going to other drives

```
'***************************************************************
Public Function Inp_Stop(ByVal axis As Integer) As Integer
        Result = inp_step_stop(0, axis)
        Inp_Stop = Result
End Function
******************** back-home drive function *********************
```

function: Back-home by following the set mode

para:

cardno	card number	
axis	axis number	
return value	0：correct	1：wrong

```
**************************************************************
Public Function Move_Home(ByVal axis As Integer) As Integer
        Result = home(0, axis)
        Move_Home = Result
End Function
```

```
*********** Clear back-to-home error information ***************
function: Clear error information of home
para:
        cardno card number
        axis axis number
return value    0: correct      1: wrong
'**************************************************************
Public Function Clear_HomeError(ByVal axis As Integer) As Integer
        Result = clear_home_error(0, axis)
        Clear_HomeError = Result
End Function
************** Single axis follow moving setting function *************
function: Function of IN simultaneous movement setup
para:
         axis—active axis number
       axis1—driven axis number1
       pulse—pulse of active axis
       pulse1—pulse of drive axis
       logical—Electricity level logic of IN signal (0:low electric 1:high electric)
mode—The active axis has moved? 0| Yes    1| No
       return value              0：correct              1：wrong
       Note: Use IN signal of active axis as trigger signal
**************************************************************
Public Function Set_InMove1(ByVal axis As Integer, ByVal axis1 As Integer, ByVal pulse As
Long, ByVal pulse1 As Long, ByVal logical As Integer, ByVal mode As Integer) As Integer
        Result = set_in_move1(0, axis, axis1, pulse, pulse1, logical, mode)
        Set_InMove1 = Result
End Function
**************2-axis follow moving setting function **************
function: Function of IN simultaneous movement setup
para:
       axis—active axis number
       axis1—driven axis number
       axis2—driven axis number
       pulse—pulse of active axis
       pulse1—pulse of drive axis1
       pulse2—pulse of drive axis2
       logical—level signal |0: From high to low          |1: From low to high
mode—The active axis has moved? 0| Yes    1| No
return value              0：correct              1：wrong
       Note: Use IN signal of active axis as trigger signal
**************************************************************
Public Function Set_InMove2(ByVal axis As Integer, ByVal axis1 As Integer, ByVal axis2 As
Integer, ByVal pulse As Long, ByVal pulse1 As Long, ByVal pulse2 As Long, ByVal logical As
Integer, ByVal mode As Integer) As Integer
```

```
        Result = set_in_move2(0, axis, axis1, axis2, pulse, pulse1, pulse2, logical, mode)
        Set_InMove2 = Result
    End Function
```

*************3-axis follow moving setting function *********

function: Function of IN simultaneous movement setup

para:

axis—active axis number

axis1—driven axis number1

axis2—driven axis number2

axis3—driven axis number3

pulse—pulse of active axis

pulse1—pulse of driven axis 1

pulse2—pulse of driven axis 2

pulse3—pulse of driven axis 3

logical—level signal |0: From high to low |1: From low to high

mode—The active axis has moved? 0| Yes 1| No

return value 0：correct 1：wrong

Note: Use IN signal of active axis as trigger signal

****

```
    Public Function Set_InMove3(ByVal axis As Integer, ByVal axis1 As Integer, ByVal axis2 As
    Integer, ByVal axis3 As Integer, ByVal pulse As Long, ByVal pulse1 As Long, ByVal pulse2 As
    Long, ByVal pulse3 As Long, ByVal logical As Integer, ByVal mode As Integer) As Integer
        Result = set_in_move3(0, axis, axis1, axis2, axis3, pulse, pulse1, pulse2, pulse3, logical,
    mode)
        Set_InMove3 = Result
    End Function
```

*************** Single axis follow stopping setting function **********

function: Set the IN synchronization action

para:

axis—active axis number

axis1—driven axis number1

logical—level signal |0: From high to low |1: From low to high

mode—active axis stop? |0:yes |1:no

return value 0: correct 1: wrong

Note:

Signal changes detected, dead axle has stopped and the driving status of drive axle can be set

****

```
    Public Function Set_InStop1(ByVal axis As Integer, ByVal axis1 As Integer, ByVal logical As
    Integer, ByVal mode As Integer) As Integer
        Result = set_in_stop1(0, axis, axis1, logical, mode)
        Set_InStop1 = Result
    End Function
```

*************2-axis follow stopping setting function *************

function: Set the IN synchronization action para:

axis—active axis number
axis1—driven axis number1
axis1—driven axis number2
logical—level signal |0: From high to low |1: From low to high
mode—active axis stop? |0:yes |1:no

return value 0：correct 1：wrong

Note:

Signal changes detected, dead axle has stopped and the driving status of drive axle can be set

```
***************************************************************
Public Function Set_InStop2(ByVal axis As Integer, ByVal axis1 As Integer, ByVal axis2 As
Integer, ByVal logical As Integer, ByVal mode As Integer) As Integer
        Result = set_in_stop2(0, axis, axis1, axis2, logical, mode)
        Set_InStop2 = Result
End Function
```

***********3-axis follow stopping setting function ***********

function: Set the IN synchronization action

para:

axis—active axis number
logical—level signal |0: From high to low |1: From low to high
mode—active axis stop? |0:yes |1:no

return value 0: correct 1: wrong

Note:

Signal changes detected, dead axle has stopped and the driving status of drive axle can be set

```
****************************************************************
Public Function Set_InStop3(ByVal axis As Integer, ByVal logical As Integer, ByVal mode As
Integer) As Integer
        Result = set_in_stop3(0, axis, logical, mode)
        Set_InStop3 = Result
End Function
```

******* Setting of single-axis drive when arriving at the target position *******

function: Setting of single-axis drive when arriving at the target position

para:

axis—active axis number
axis2—driven axis number
pulse1—pluse of active axis 1
pulse2—pluse of driven axis 2
regi—0|comp+ Select the compare register 1|comp-
term—0|>= Select the compare register 1|<

return value 0：correct 1：wrong

```
****************************************************************
Public Function Setup_Pmove1(ByVal axis As Integer, ByVal axis1 As Integer, ByVal pulse As
Long, ByVal pulse1 As Long, ByVal regi As Integer, ByVal term As Integer) As Integer
```

```
          Result = set_comp_pmove1(0, axis, axis1, pulse, pulse1, regi, term)
        Setup_Pmove1 = Result
    End Function
    ******** Setting of two-axis drive when arriving at the target position ********
     function: Setting of two-axis drive when arriving at the target position
     para:
        axis—active axis number
        axis1—driven axis number1
        axis2—driven axis number2
        pulse—pulse of active axis
        pulse1—pulse of driven axis1
        pulse2—pulse of driven axis2
        regi—0|comp+    Select the compare register          1|comp-
         term—0|>=    Select the compare register             1|<
    return value              0：correct                1：wrong
    ************************************************************
    Public Function Setup_Pmove2(ByVal axis As Integer, ByVal axis1 As Integer, ByVal axis2 As
    Integer, ByVal pulse As Long, ByVal pulse1 As Long, ByVal pulse2 As Long, ByVal regi As
    Integer, ByVal term As Integer) As Integer
        Result = set_comp_pmove2(0, axis, axis1, axis2, pulse, pulse1, pulse2, regi, term)
        Setup_Pmove2 = Result
    End Function
    ****** Setting of three-axis drive when arriving at the target position *********
function: Setting of three-axis drive when arriving at the target position
para:
        axis—active axis number
        axis1—driven axis number1
        axis2—driven axis number2
        axis2—driven axis number3
        pulse—pulse of active axis
        pulse1—pulse of driven axis1
        pulse2—pulse of driven axis 2
        pulse3—pulse of driven axis3
        regi—0|comp+    Select the compare register         1|comp-
         term—0|>=    Select the compare register            1|<
    return value              0：correct                1：wrong
    ************************************************************
    Public Function Setup_Pmove3(ByVal axis As Integer, ByVal axis1 As Integer, ByVal axis2 As
    Integer, ByVal axis3 As Integer, ByVal pulse As Long, ByVal pulse1 As Long, ByVal pulse2 As
    Long, ByVal pulse3 As Long, ByVal regi As Integer, ByVal term As Integer) As Integer
        Result = set_comp_pmove3(0, axis, axis1, axis2, axis3, pulse, pulse1, pulse2, pulse3, regi,
    term)
        Setup_Pmove3 = Result
    End Function
    ********** Setting of stopping drive when arriving at the target position ******
```

```
      function: Setting of stopping drive when arriving at the target position
      para:
              axis—active axis number
              axis1—driven axis number1
              pulse—target position of active axis
              regi—0|comp+    Select the compare register           1|comp-
                term—0|>=   Select the compare register               1|<
                mode—|0: active axis stop                         |1: active axis does not stop
       return value               0：correct                1：wrong
       ***************************************************************
       Public Function Setup_Stop1(ByVal axis As Integer, ByVal axis1 As Integer, ByVal pulse As
       Long, ByVal regi As Integer, ByVal term As Integer, ByVal mode As Integer) As Integer
               Result = set_comp_stop1(0, axis, axis1, pulse, regi, term, mode)
               Setup_Stop1 = Result
       End Function
       ****** Setting of stopping drive when arriving at the target position *********
function: Setting of stopping drive when arriving at the target position
para:
              axis—active axis number
              axis1—driven axis number1
              axis2—driven axis number2
              pulse—target position of active axis
              regi—0|comp+    Select the compare register           1|comp-
                term—0|>=   Select the compare register               1|<
                mode—|0: active axis stop                         |1: active axis does not stop
       return value               0：correct                1：wrong
       ***************************************************************
       Public Function Setup_Stop2(ByVal axis As Integer, ByVal axis1 As Integer, ByVal axis2 As
       Integer, ByVal pulse As Long, ByVal regi As Integer, ByVal term As Integer, ByVal mode As
       Integer) As Integer
               Result = set_comp_stop2(0, axis, axis1, axis2, pulse, regi, term, mode)
               Setup_Stop2 = Result
       End Function
       ******* Setting of stopping drive when arriving at the target position *******
      function: Setting of stopping drive when arriving at the target position
      para:
              axis—active axis number
              pulse—target position of active axis
              regi—0|comp+    Select the compare register             1|comp-
                term—0|>=   Select the compare register                    1|<
                mode—|0: active axis stop                         |1: active axis does not stop
       return value               0：correct                1：wrong
       ***************************************************************
       Public Function Setup_Stop3(ByVal axis As Integer, ByVal pulse As Long, ByVal regi As
       Integer, ByVal term As Integer, ByVal mode As Integer) As Integer
```

```
    Result = set_comp_stop3(0, axis, pulse, regi, term, mode)
    Setup_Stop3 = Result
End Function
******************* Composite drives ********************
' Note: The following functions are added for the convenience of customers
**************************************************

*************** Symmetrical relative movement of single-axis **********
function: Refer to the current position and perform quantitative movement in the symmetrical acceleration/deceleration
para:
      cardno -card number
      axis ---axis number
      pulse -- Pulse
      lspd --- Low speed
      hspd --- High speed
      tacc--- Time of acceleration (Unit: sec)
      vacc --- Change rate of acceleration/deceleration
      mode--- Mode (trapezoid(0) or S-curve(1))
return value 0: correct 1: wrong
**************************************************************
Public Function Symmetry_RelativeMove(ByVal axis As Integer, ByVal pulse As Long, ByVal lspd As Long, ByVal hspd As Long, ByVal tacc As Double, ByVal vacc As Long, ByVal mode As Integer) As Integer
    Result = symmetry_relative_move(0, axis, pulse, lspd, hspd, tacc, vacc, mode)
    Symmetry_RelativeMove = Result
End Function
********* Symmetrical absolute movement of single-axis *************
function: Refer to the position of zero point and perform quantitative movement in the symmetrical acceleration/deceleration
para:
      cardno -card number
      axis ---axis number
      pulse -- Pulse
      lspd --- Low speed
      hspd --- High speed
      tacc--- Time of acceleration (Unit: sec)
      vacc --- Change rate of acceleration/deceleration
      mode--- Mode (trapezoid(0) or S-curve(1))
return value 0: correct 1: wrong
***************************************************************
Public Function Symmetry_AbsoluteMove(ByVal axis As Integer, ByVal pulse As Long, ByVal lspd As Long, ByVal hspd As Long, ByVal tacc As Double, ByVal vacc As Long, ByVal mode As Integer) As Integer
    Result = symmetry_absolute_move(0, axis, pulse, lspd, hspd, tacc, vacc, mode)
```

```
        Symmetry_AbsoluteMove = Result
    End Function
```

*********** asymmetrical relative movement of single-axis *********

function: Refer to the current position and perform quantitative movement in the asymmetrical acceleration/deceleration

para:

cardno -card number
axis ---axis number
pulse -- Pulse
lspd --- Low speed
hspd --- High speed
tacc--- Time of acceleration (Unit: sec)
tdec--- Time of deceleration (Unit: sec)
vacc--- Change rate of acceleration/deceleration
mode--- Mode (trapezoid(0) or S-curve(1))

return value 0：correct 1：wrong

****

```
    Public Function Unsymmetry_RelativeMove(ByVal axis As Integer, ByVal pulse As Long, ByVal lspd As Long, ByVal hspd As Long, ByVal tacc As Double, ByVal tdec As Double, ByVal vacc As Long, ByVal mode As Integer) As Integer
        Result = unsymmetry_relative_move(0, axis, pulse, lspd, hspd, tacc, tdec, vacc, mode)

        Unsymmetry_RelativeMove = Result
    End Function
```

******** asymmetrical absolute movement of single-axis ********

function: Refer to the position of zero point and perform quantitative movement in the asymmetrical acceleration/deceleration

para:

cardno-card number
axis---axis number
pulse -- Pulse
lspd --- Low speed
hspd --- High speed
tacc--- Time of acceleration (Unit: sec)
tdec--- Time of deceleration (Unit: sec)
vacc--- Change rate of acceleration/deceleration
mode--- Mode (trapezoid(0) or S-curve(1))

return value 0：correct 1：wrong

****

```
    Public Function Unsymmetry_AbsoluteMove(ByVal axis As Integer, ByVal pulse As Long, ByVal lspd As Long, ByVal hspd As Long, ByVal tacc As Double, ByVal tdec As Double, ByVal vacc As Long, ByVal mode As Integer) As Integer
        Result = unsymmetry_absolute_move(0, axis, pulse, lspd, hspd, tacc, tdec, vacc, mode)
        Unsymmetry_AbsoluteMove = Result
    End Function
```

****** Relative movement of two-axis symmetrical linear interpolation *******

function: Refer to current position and perform linear interpolation in symmetrical acceleration/deceleration

para:

cardno-card number
axis1---axis number1
axis2---axis number2
pulse1-- pulse 1
pulse2-- pulse 2
lspd --- Low speed
hspd --- High speed
tacc--- Time of acceleration (Unit: sec)
vacc--- Change rate of acceleration/deceleration
mode--- Mode (trapezoid(0) or S-curve(1))

return value 0: correct 1: wrong

****

```
Public Function Symmetry_RelativeLine2(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal pulse1 As Long, ByVal pulse2 As Long, ByVal lspd As Long, ByVal hspd As Long, ByVal tacc As Double, ByVal vacc As Long, ByVal mode As Integer) As Integer
    Result = symmetry_relative_line2(0, axis1, axis2, pulse1, pulse2, lspd, hspd, tacc, vacc, mode)
    Symmetry_RelativeLine2 = Result
End Function
```

******* Two axes symmetric linear interpolation absolute moving ******

function: Refer to the position of zero point and perform linear interpolation in symmetrical acceleration/deceleration

para:

cardno -card number
axis1 ---axis number1
axis2 ---axis number2
pulse1—pulse of axis 1
pulse2-- pulse of axis 2
lspd --- Low speed
hspd --- High speed
tacc--- Time of acceleration (Unit: sec)
vacc--- Change rate of acceleration/deceleration
mode--- Mode (trapezoid(0) or S-curve(1))

return value 0: correct 1: wrong

****

```
Public Function Symmetry_AbsoluteLine2(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal pulse1 As Long, ByVal pulse2 As Long, ByVal lspd As Long, ByVal hspd As Long, ByVal tacc As Double, ByVal vacc As Long, ByVal mode As Integer) As Integer
    Result = symmetry_absolute_line2(0, axis1, axis2, pulse1, pulse2, lspd, hspd, tacc, vacc, mode)
    Symmetry_AbsoluteLine2 = Result
```

End Function

****** Two axes asymmetric linear interpolation relative moving *********

function: Refer to current position and perform linear interpolation in asymmetric acceleration/deceleration.

para:

cardno-card number

axis1---axis number1

axis2---axis number2

pulse1—pulse of axis 1

pulse2-- pulse of axis 2

lspd --- Low speed

hspd --- High speed

tacc--- Time of acceleration (Unit: sec)

tdec--- Time of deceleration (Unit: sec)

vacc--- Change rate of acceleration/deceleration

mode--- Mode (trapezoid(0) or S-curve(1))

return value 0: correct 1: wrong

****

Public Function Unsymmetry_RelativeLine2(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal pulse1 As Long, ByVal pulse2 As Long, ByVal lspd As Long, ByVal hspd As Long, ByVal tacc As Double, ByVal tdec As Double, ByVal vacc As Long, ByVal mode As Integer) As Integer

Result = unsymmetry_relative_line2(0, axis1, axis2, pulse1, pulse2, lspd, hspd, tacc, tdec, vacc, mode)

Unsymmetry_RelativeLine2 = Result

End Function

****** Two axes asymmetric linear interpolation absolute moving *****

function: Refer to the position of zero point and perform linear interpolation in asymmetric acceleration/deceleration

para:

cardno-card number

axis1---axis number1

axis2---axis number2

pulse1—pulse of axis 1

pulse2-- pulse of axis 2

lspd --- Low speed

hspd --- High speed

tacc--- Time of acceleration (Unit: sec)

tdec--- Time of deceleration (Unit: sec)

vacc--- Change rate of acceleration/deceleration

mode--- Mode (trapezoid(0) or S-curve(1))

return value 0: correct 1: wrong

****

Public Function Unsymmetry_AbsoluteLine2(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal pulse1 As Long, ByVal pulse2 As Long, ByVal lspd As Long, ByVal hspd As Long, ByVal tacc As Double, ByVal tdec As Double, ByVal vacc As Long, ByVal mode As Integer) As Integer

Result = unsymmetry_absolute_line2(0, axis1, axis2, pulse1, pulse2, lspd, hspd, tacc, tdec, vacc, mode)

Unsymmetry_AbsoluteLine2 = Result

End Function

****** Three axes symmetric linear interpolation relative moving ******

function: Refer to current position and perform linear interpolation in symmetric acceleration/deceleration

para:

cardno-card number
axis1---axis number1
axis2---axis number2
axis3---axis number3
pulse1-- pulse of axis 1
pulse2-- pulse of axis 2
pulse3-- pulse of axis 3
lspd --- Low speed
hspd --- High speed
tacc--- Time of acceleration (Unit: sec)
vacc--- Change rate of acceleration/deceleration
mode--- Mode (trapezoid(0) or S-curve(1))

return value 0: correct 1: wrong

****

Public Function Symmetry_RelativeLine3(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal axis3 As Integer, ByVal pulse1 As Long, ByVal pulse2 As Long, ByVal pulse3 As Long, ByVal lspd As Long, ByVal hspd As Long, ByVal tacc As Double, ByVal vacc As Long, ByVal mode As Integer) As Integer

Result = symmetry_relative_line3(0, axis1, axis2, axis3, pulse1, pulse2, pulse3, lspd, hspd, tacc, vacc, mode)

Symmetry_RelativeLine3 = Result

End Function

******* Three axes symmetric linear interpolation absolute moving *******

function: Refer to the position of zero point and perform linear interpolation in symmetric acceleration/deceleration.

para:

cardno-card number
axis1---axis number1
axis2---axis number2
axis3---axis number3
pulse1-- pulse of axis 1
pulse2-- pulse of axis 2
pulse3-- pulse of axis 3
lspd --- Low speed
hspd --- High speed

tacc--- Time of acceleration (Unit: sec)
vacc--- Change rate of acceleration/deceleration
mode--- Mode (trapezoid(0) or S-curve(1))

return value 0: correct 1: wrong

****

Public Function Symmetry_AbsoluteLine3(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal axis3 As Integer, ByVal pulse1 As Long, ByVal pulse2 As Long, ByVal pulse3 As Long, ByVal lspd As Long, ByVal hspd As Long, ByVal tacc As Double, ByVal vacc As Long, ByVal mode As Integer) As Integer

Result = symmetry_absolute_line3(0, axis1, axis2, axis3, pulse1, pulse2, pulse3, lspd, hspd, tacc, vacc, mode)

Symmetry_AbsoluteLine3 = Result

End Function

****** Three axes asymmetric linear interpolation relative moving *******

function: Refer to current position and perform linear interpolation in asymmetric acceleration/deceleration

para:

cardno-card number
axis1---axis number1
axis2---axis number2
axis3---axis number3
pulse1-- pulse of axis 1
pulse2-- pulse of axis 2
pulse3-- pulse of axis 3
lspd --- Low speed
hspd --- High speed
tacc--- Time of acceleration (Unit: sec)
tdec--- Time of deceleration (Unit: sec)
vacc--- Change rate of acceleration/deceleration
mode--- Mode (trapezoid(0) or S-curve(1))

return value 0: correct 1: wrong

****

Public Function Unsymmetry_RelativeLine3(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal axis3 As Integer, ByVal pulse1 As Long, ByVal pulse2 As Long, ByVal pulse3 As Long, ByVal lspd As Long, ByVal hspd As Long, ByVal tacc As Double, ByVal tdec As Double, ByVal vacc As Long, ByVal mode As Integer) As Integer

Result = unsymmetry_relative_line3(0, axis1, axis2, axis3, pulse1, pulse2, pulse3, lspd, hspd, tacc, tdec, vacc, mode)

Unsymmetry_RelativeLine3 = Result

End Function

****** Three axes asymmetric linear interpolation absolute moving ****

function: Refer to the position of zero point and perform linear interpolation in asymmetric acceleration/deceleration

para:

cardno-card number

axis1---axis number1
axis2---axis number2
axis3---axis number3
pulse1-- pulse of axis 1
pulse2-- pulse of axis 2
pulse3-- pulse of axis 3
lspd --- Low speed
hspd --- High speed
tacc--- Time of acceleration (Unit: sec)
tdec--- Time of deceleration (Unit: sec)
vacc--- Change rate of acceleration/deceleration
mode--- Mode (trapezoid(0) or S-curve(1))

****

```
Public Function Unsymmetry_AbsoluteLine3(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal axis3 As Integer, ByVal pulse1 As Long, ByVal pulse2 As Long, ByVal pulse3 As Long, ByVal lspd As Long, ByVal hspd As Long, ByVal tacc As Double, ByVal tdec As Double, ByVal vacc As Long, ByVal mode As Integer) As Integer
    Result = unsymmetry_absolute_line3(0, axis1, axis2, axis3, pulse1, pulse2, pulse3, lspd, hspd, tacc, tdec, vacc, mode)
    Unsymmetry_AbsoluteLine3 = Result
End Function
```

********* Two axes symmetric arc interpolation relative moving ********

function: Refer to current position and perform arc interpolation in symmetric acceleration/deceleration.

para:

cardno-card number
axis1---axis number1
axis2---axis number2
x、y---- Coordinates of arc end point (Refer to current point, that is, starting point of arc)
i、j---- Centre coordinate (Refer to current point, that is, starting point of arc)
dir----- Moving direction (0-Clockwise,1-Anti-clockwise)
lspd --- Low speed
hspd --- High speed
tacc--- Time of acceleration (Unit: sec)
vacc--- Change rate of acceleration/deceleration
mode--- Mode (trapezoid(0) or S-curve(1))

return value 0: correct 1: wrong

****

```
Public Function Symmetry_Relativearc(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal X As Long, ByVal Y As Long, ByVal i As Long, ByVal j As Long, ByVal dir As Integer, ByVal lspd As Long, ByVal hspd As Long, ByVal tacc As Double, ByVal vacc As Long, ByVal mode As Integer) As Integer
    Result = symmetry_relative_arc(0, axis1, axis2, X, Y, i, j, dir, lspd, hspd, tacc, vacc, mode)
    Symmetry_Relativearc = Result
End Function
```

****** Two axes symmetric arc interpolation absolute moving *******

function: Refer to the position of zero point and perform arc interpolation in symmetric acceleration/deceleration

para:

cardno-card number

axis1---axis number1

axis2---axis number2

x、y---- Coordinates of arc end point (Refer to current point, that is, starting point of arc)

i、j---- Centre coordinate (Refer to current point, that is, starting point of arc)

dir----- Moving direction (0-Clockwise,1-Anti-clockwise)

lspd --- Low speed

hspd --- High speed

tacc--- Time of acceleration (Unit: sec)

vacc--- Change rate of acceleration/deceleration

mode--- Mode (trapezoid(0) or S-curve(1))

return value 0: correct 1: wrong

****

Public Function Symmetry_AbsoluteArc(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal X As Long, ByVal Y As Long, ByVal i As Long, ByVal j As Long, ByVal dir As Integer, ByVal lspd As Long, ByVal hspd As Long, ByVal tacc As Double, ByVal vacc As Long, ByVal mode As Integer) As Integer

Result = symmetry_absolute_arc(0, axis1, axis2, X, Y, i, j, dir, lspd, hspd, tacc, vacc, mode)

Symmetry_AbsoluteArc = Result

End Function

********* Two axes asymmetric arc interpolation relative moving ******

function: Refer to current position and perform arc interpolation in asymmetric acceleration/deceleration

para:

cardno-card number

axis1---axis number1

axis2---axis number2

x、y---- Coordinates of arc end point (Refer to current point, that is, starting point of arc)

i、j---- Centre coordinate (Refer to current point, that is, starting point of arc)

dir----- Moving direction (0-Clockwise,1-Anti-clockwise)

lspd --- Low speed

hspd --- High speed

tacc--- Time of acceleration (Unit: sec)

tdec--- Time of deceleration (Unit: sec)

vacc--- Change rate of acceleration/deceleration

mode--- Mode (trapezoid(0) or S-curve(1))

return value 0: correct 1: wrong

****

Public Function Unsymmetry_RelativeArc(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal X As Long, ByVal Y As Long, ByVal i As Long, ByVal j As Long, ByVal dir As Integer, ByVal lspd As Long, ByVal hspd As Long, ByVal tacc As Double, ByVal tdec As Double, ByVal vacc As Long, ByVal mode As Integer) As Integer

Result = unsymmetry_relative_arc(0, axis1, axis2, X, Y, i, j, dir, lspd, hspd, tacc, tdec, vacc, mode)

Unsymmetry_RelativeArc = Result

End Function

********** Two axes asymmetric arc interpolation absolute moving *****

function: Refer to current position and perform arc interpolation in asymmetric acceleration/deceleration

para:

cardno-card number
axis1---axis number1
axis2---axis number2
x、y---- Coordinates of arc end point (Refer to current point, that is, starting point of arc)
i、j---- Centre coordinate (Refer to current point, that is, starting point of arc)
dir----- Moving direction (0-Clockwise,1-Anti-clockwise)
lspd --- Low speed
hspd --- High speed
tacc--- Time of acceleration (Unit: sec)
tdec--- Time of deceleration (Unit: sec)
vacc--- Change rate of acceleration/deceleration
mode--- Mode (trapezoid(0) or S-curve(1))

return value 0: correct 1: wrong

****

Public Function Unsymmetry_AbsoluteArc(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal X As Long, ByVal Y As Long, ByVal i As Long, ByVal j As Long, ByVal dir As Integer, ByVal lspd As Long, ByVal hspd As Long, ByVal tacc As Double, ByVal tdec As Double, ByVal vacc As Long, ByVal mode As Integer) As Integer

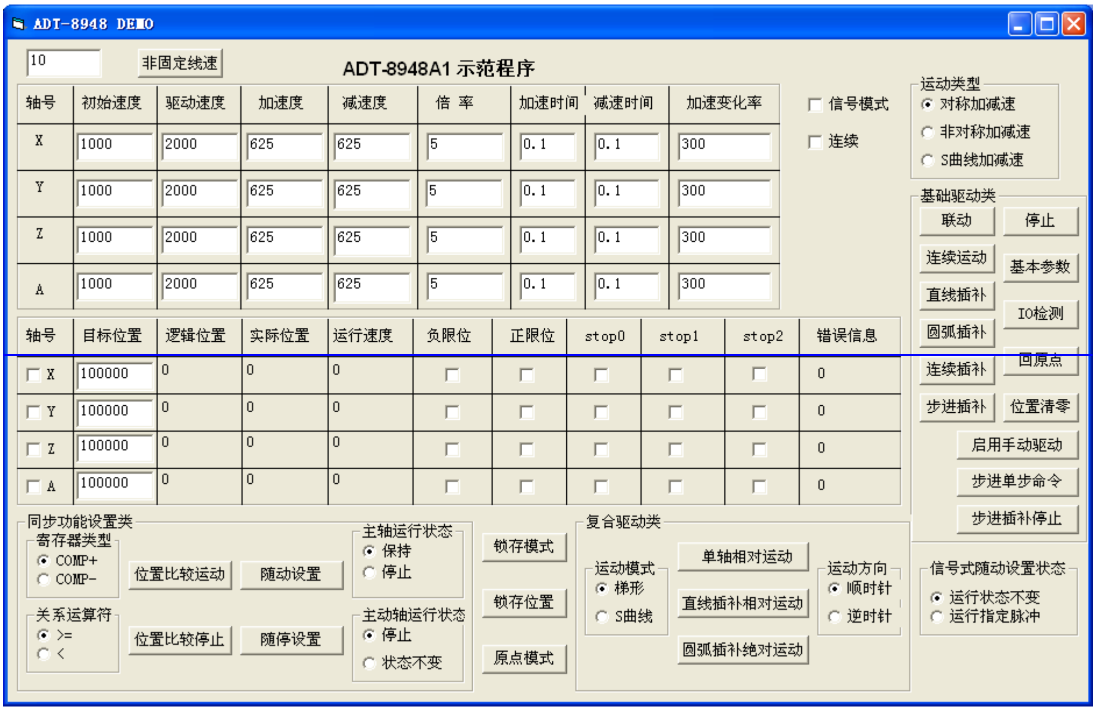
Result = unsymmetry_absolute_arc(0, axis1, axis2, X, Y, i, j, dir, lspd, hspd, tacc, tdec, vacc, mode)

Unsymmetry_AbsoluteArc = Result

End Function

1.3 Function-Realization-Module

1.31 Interface design



10

轴号	初始速度	驱动速度	加速度	减速度	倍 率	加速时间	减速时间	加速变化率
X	1000	2000	625	625	5	0.1	0.1	300
Y	1000	2000	625	625	5	0.1	0.1	300
Z	1000	2000	625	625	5	0.1	0.1	300
A	1000	2000	625	625	5	0.1	0.1	300

□ 信号模式
□ 连续

运动类型
● 对称加减速
○ 非对称加减速
○ S曲线加减速

轴号	目标位置	逻辑位置	实际位置	运行速度	负限位	正限位	stop0	stop1	stop2	错误信息
□ X	100000	0	0	0	□	□	□	□	□	0
□ Y	100000	0	0	0	□	□	□	□	□	0
□ Z	100000	0	0	0	□	□	□	□	□	0
□ A	100000	0	0	0	□	□	□	□	□	0

寄存器类型
● COMP+
○ COMP-

关系运算符
● >=
○ <

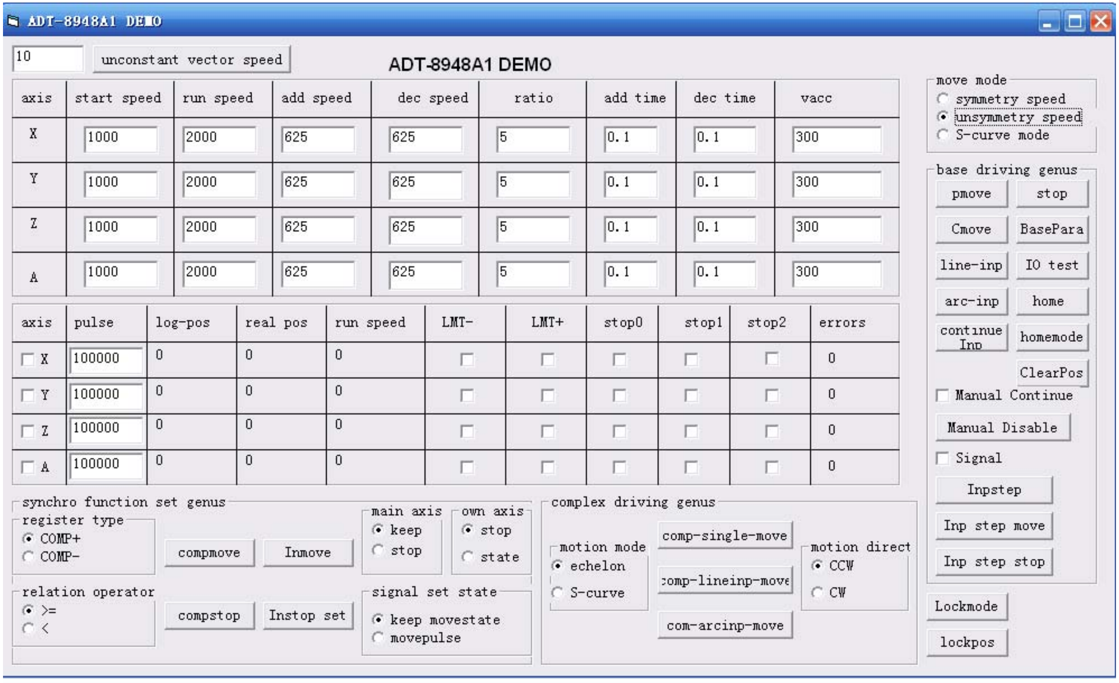
主轴运行状态
● 保持
○ 停止

主动轴运行状态
● 停止
○ 状态不变

运动模式
● 梯形
○ S曲线

运动方向
● 顺时针
○ 逆时针

信号式随动设置状态
● 运行状态不变
○ 运行指定脉冲



10

axis	start speed	run speed	add speed	dec speed	ratio	add time	dec time	vacc
X	1000	2000	625	625	5	0.1	0.1	300
Y	1000	2000	625	625	5	0.1	0.1	300
Z	1000	2000	625	625	5	0.1	0.1	300
A	1000	2000	625	625	5	0.1	0.1	300

move mode
○ symmetry speed
● unsymmetry speed
○ S-curve mode

□ Manual Continue
□ Signal

axis	pulse	log-pos	real pos	run speed	LMT-	LMT+	stop0	stop1	stop2	errors
□ X	100000	0	0	0	□	□	□	□	□	0
□ Y	100000	0	0	0	□	□	□	□	□	0
□ Z	100000	0	0	0	□	□	□	□	□	0
□ A	100000	0	0	0	□	□	□	□	□	0

register type
● COMP+
○ COMP-

relation operator
● >=
○ <

main axis
● keep
○ stop

own axis
● stop
○ state

signal set state
● keep movestate
○ movepulse

motion mode
● echelon
○ S-curve

motion direct
● CCW
○ CW

Note:

(1) Speed settings—Set the start velocity, driving velocity and acceleration of every axis; position settings—set the drive pulse of every axis; drive information—real-time display the logical position, actual position and operating speed of every axis.

(2) Driver objects—Select the driver object and confirm the axis that involved in linkage or interpolation;

(3) Linkage—to send single-axis drive command to all axis of selected driver object; interpolation—to send interpolation command to all axis of selected driver object; stop—stop the pulse output of all axis;

All the data above are in Pulse

1.3.2 Initial code of motion card is located in window initialization. Codes added by user are as follow:

```
Private Sub Form_Load()
    Init_Board                          ' initialize card
    m_nAddMode(0).Value = True     'motion type
    m_nMoveState(0).Value = True
    m_nCcwDir(0).Value = True
    m_nMoveType(0).Value = True
    m_nKeep(0).Value = True
    m_nStop1(0).Value = True
    m_nPcomp(0).Value = True
    m_nBigAmount(0).Value = True
    g_CheckVector = False
    g_CheckManual = False
    g_CompMove = False
    g_LineinpMove = False
    g_ArcMove = False
End Sub
```

1.3.3 Pmove code is in the message from the click of “AxisPmove” button, it will send out corresponding drive command according to the selected object, codes are as below:

```
        Private Sub AxisPmove_Click()
            For i = 1 To 4
                g_Ratio(i - 1) = CLng(m_nRatio(i - 1).Text)
            Next i
            If m_bX.Value = vbChecked Then
                Setup_Speed 1, m_nStartV(0).Text, m_nSpeed(0).Text, m_nAdd(0).Text, m_nDec(0).Text, m_nRatio(0).Text, nAddMode
                Axis_Pmove 1, m_nPulse(0).Text
            End If
            If m_bY.Value = vbChecked Then
                Setup_Speed 2, m_nStartV(1).Text, m_nSpeed(1).Text, m_nAdd(1).Text, m_nDec(1).Text, m_nRatio(1).Text, nAddMode
                Axis_Pmove 2, m_nPulse(1).Text
            End If
            If m_bZ.Value = vbChecked Then
                Setup_Speed 3, m_nStartV(2).Text, m_nSpeed(2).Text, m_nAdd(2).Text, m_nDec(2).Text, m_nRatio(2).Text, nAddMode
                Axis_Pmove 3, m_nPulse(2).Text
            End If
            If m_bA.Value = vbChecked Then
                Setup_Speed 4, m_nStartV(3).Text, m_nSpeed(3).Text, m_nAdd(3).Text, m_nDec(3).Text, m_nRatio(3).Text, nAddMode
                Axis_Pmove 4, m_nPulse(3).Text
            End If
    End Sub
```

1.3.4 Interpolation code is in the message from the click of “InterpMove” button, it will send out corresponding drive command according to the selected object, codes are as below:

```
Private Sub InterpMove_Click()
    For i = 1 To 4
        g_Ratio(i - 1) = CLng(m_nRatio(i - 1).Text)
    Next i
'************************* Interpolation ***************************
'           Random 2-3 axes interpolation
'*********************************************************
    If m_bX.Value = vbChecked And m_bY.Value = vbChecked And m_bZ.Value = vbChecked And
m_bA.Value <> vbChecked Then
        Setup_Speed 1, m_nStartV(0).Text, m_nSpeed(0).Text, m_nAdd(0).Text, m_nDec(0).Text,
m_nRatio(0).Text, nAddMode
        Interp_Move3 1, 2, 3, m_nPulse(0).Text, m_nPulse(1).Text, m_nPulse(2).Text
    ElseIf m_bX.Value = vbChecked And m_bY.Value = vbChecked And m_bZ.Value <>
vbChecked And m_bA.Value = vbChecked Then
        Setup_Speed 1, m_nStartV(0).Text, m_nSpeed(0).Text, m_nAdd(0).Text, m_nDec(0).Text,
m_nRatio(0).Text, nAddMode
        Interp_Move3 1, 2, 4, m_nPulse(0).Text, m_nPulse(1).Text, m_nPulse(3).Text
   ElseIf m_bX.Value = vbChecked And m_bY.Value <> vbChecked And m_bZ.Value = vbChecked
And m_bA.Value = vbChecked Then
        Setup_Speed 1, m_nStartV(0).Text, m_nSpeed(0).Text, m_nAdd(0).Text, m_nDec(0).Text,
m_nRatio(0).Text, nAddMode
        Interp_Move3 1, 3, 4, m_nPulse(0).Text, m_nPulse(2).Text, m_nPulse(3).Text
     ElseIf m_bX.Value <> vbChecked And m_bY.Value = vbChecked And m_bZ.Value =
vbChecked And m_bA.Value = vbChecked Then
        Setup_Speed 2, m_nStartV(1).Text, m_nSpeed(1).Text, m_nAdd(1).Text, m_nDec(1).Text,
m_nRatio(1).Text, nAddMode
        Interp_Move3 2, 3, 4, m_nPulse(1).Text, m_nPulse(2).Text, m_nPulse(3).Text
  ElseIf m_bX.Value = vbChecked And m_bY.Value = vbChecked And m_bZ.Value <> vbChecked
And m_bA.Value <> vbChecked Then
        Setup_Speed 1, m_nStartV(0).Text, m_nSpeed(0).Text, m_nAdd(0).Text, m_nDec(0).Text,
m_nRatio(0).Text, nAddMode
        Interp_Move2 1, 2, m_nPulse(0).Text, m_nPulse(1).Text
    ElseIf m_bZ.Value = vbChecked And m_bA.Value <> vbChecked And m_bX.Value =
vbChecked And m_bY.Value <> vbChecked Then
        Setup_Speed 1, m_nStartV(0).Text, m_nSpeed(0).Text, m_nAdd(0).Text, m_nDec(0).Text,
m_nRatio(0).Text, nAddMode
        Interp_Move2 1, 3, m_nPulse(0).Text, m_nPulse(2).Text
    ElseIf m_bZ.Value <> vbChecked And m_bA.Value = vbChecked And m_bX.Value =
vbChecked And m_bY.Value <> vbChecked Then
        Setup_Speed 1, m_nStartV(0).Text, m_nSpeed(0).Text, m_nAdd(0).Text, m_nDec(0).Text,
m_nRatio(0).Text, nAddMode
        Interp_Move2 1, 4, m_nPulse(0).Text, m_nPulse(3).Text
   ElseIf m_bZ.Value = vbChecked And m_bA.Value <> vbChecked And m_bX.Value <>
vbChecked And m_bY.Value = vbChecked Then
```

```
        Setup_Speed 2, m_nStartV(1).Text, m_nSpeed(1).Text, m_nAdd(1).Text, m_nDec(1).Text, m_nRatio(1).Text, nAddMode
        Interp_Move2 2, 3, m_nPulse(1).Text, m_nPulse(2).Text
   ElseIf m_bZ.Value <> vbChecked And m_bA.Value = vbChecked And m_bX.Value <> vbChecked And m_bY.Value = vbChecked Then
        Setup_Speed 2, m_nStartV(1).Text, m_nSpeed(1).Text, m_nAdd(1).Text, m_nDec(1).Text, m_nRatio(1).Text, nAddMode
        Interp_Move2 2, 4, m_nPulse(1).Text, m_nPulse(3).Text
   ElseIf m_bZ.Value = vbChecked And m_bA.Value = vbChecked And m_bX.Value <> vbChecked And m_bY.Value <> vbChecked Then
        Setup_Speed 3, m_nStartV(2).Text, m_nSpeed(2).Text, m_nAdd(2).Text, m_nDec(2).Text, m_nRatio(2).Text, nAddMode
        Interp_Move2 3, 4, m_nPulse(2).Text, m_nPulse(3).Text
    ElseIf m_bX.Value <> vbChecked And m_bY.Value <> vbChecked And m_bZ.Value <> vbChecked And m_bA.Value <> vbChecked Then
        MsgBox " Please select the interpolation axis ", , " Tips "
    Else
       MsgBox "The axis you choose are wrong,please choose again!", , " Tips "
    End If
End Sub
```

1.3.5 Continuous interpolation code is in the message from the click of “ConIntMove” button, here only perform four segments continuous interpolation of X and Y axis. Codes are as below:

```
Private Sub ConIntMove_Click()
   Dim nstate1 As Integer
   Dim nstate2 As Integer
   If nAddMode = 2 Then
      MsgBox " Not allowed to select S-curve acceleration/deceleration ", , " Tips "
       Exit Sub
   End If
   For i = 1 To 4
         g_Ratio(i - 1) = CLng(m_nRatio(i - 1).Text)
   Next i
   If m_bX.Value = vbChecked And m_bY.Value = vbChecked And m_bZ.Value <> vbChecked And m_bA.Value <> vbChecked Then
      Setup_Speed 1, m_nStartV(0).Text, m_nSpeed(0).Text, m_nAdd(0).Text, m_nDec(0).Text, m_nRatio(0).Text, nAddMode
      Set_DecMode   1, 0, 1
     Set_DecPos 1, m_nPulse(0).Text, m_nStartV(0).Text, m_nSpeed(0).Text, m_nAdd(0).Text
  Else
       MsgBox " Please select X-Y axis!", , " Tips "
       Exit Sub
   End If
      ForbidDec
    ************ The first segment *************
    Interp_Move2 1, 2, m_nPulse(0), m_nPulse(1)
    Do While True
       Get_ErrorInf 1, nstate1
     If nstate1 <> 0 Then GoTo err
        Get_AllowInpStatus nstate2
     If nstate2 <> 0 Then Exit Do
         DoEvents
     Loop
     ************* The second segment ************
     Interp_Move2 1, 2, m_nPulse(0), m_nPulse(1)
      Do While True
         Get_ErrorInf 1, nstate1
      If nstate1 <> 0 Then GoTo err
          Get_AllowInpStatus nstate2
      If nstate2 <> 0 Then Exit Do
         DoEvents
      Loop
     *********** The third segment ***********
      Interp_Move2 1, 2, m_nPulse(0), m_nPulse(1)
     Do While True
         Get_ErrorInf 1, nstate1
      If nstate1 <> 0 Then GoTo err
          Get_AllowInpStatus nstate2
      If nstate2 <> 0 Then Exit Do
         DoEvents
      Loop
     ************ The last segment ***********
      AllowDec
      Interp_Move2 1, 2, m_nPulse(0), m_nPulse(1)
      Do While True
         Get_ErrorInf 1, nstate1
      If nstate1 <> 0 Then GoTo err
        Get_MoveStatus 1, nstate2, 1
      If nstate2 = 0 Then Exit Do
          DoEvents
       Loop
       Exit Sub
err:
    MsgBox "error"
End Sub
```

1.3.6 Arc interpolation code is in the message from the click of “ArcInpMove” button, it will send out corresponding drive command according to the selected object, codes are as below:

```
Private Sub ArcInpMove_Click()
Dim Value As Long
       If nAddMode = 2 Then
       MsgBox " Not allow to select S-curve acceleration/deceleration!", , " Tips "
```

```
              Exit Sub
                  End If
        For i = 1 To 4
                  g_Ratio(i - 1) = CLng(m_nRatio(i - 1).Text)
        Next i
        '*******************X-Y arc interpolation ********************
        If m_bX.Value = vbChecked And m_bY.Value = vbChecked And m_bZ.Value <> vbChecked
And m_bA.Value <> vbChecked Then
                    Setup_Speed 1, m_nStartV(0).Text, m_nSpeed(0).Text, m_nAdd(0).Text,
m_nDec(0).Text, m_nRatio(0).Text, nAddMode
                    If m_nStartV(0).Text < m_nSpeed(0).Text Then
                          Set_DecMode 1, 0, 1
                          Value = (Sqr(m_nPulse(0).Text * m_nPulse(0).Text + m_nPulse(1).Text *
m_nPulse(1).Text) / 1.414) * 8
                          Set_DecPos 1, Value, m_nStartV(0).Text, m_nSpeed(0).Text, m_nAdd(0).Text
                    End If
                          Interp_Arc 1, 2, 0, 0, m_nPulse(0).Text, m_nPulse(1).Text
      '*******************X-Z arc interpolation ********************
      ElseIf m_bX.Value = vbChecked And m_bY.Value <> vbChecked And m_bZ.Value = vbChecked
And m_bA.Value <> vbChecked Then
                    Setup_Speed 1, m_nStartV(0).Text, m_nSpeed(0).Text, m_nAdd(0).Text,
m_nDec(0).Text, m_nRatio(0).Text, nAddMode
                    If m_nStartV(0).Text < m_nSpeed(0).Text Then
                          Set_DecMode 1, 0, 1
                          Value = (Sqr(m_nPulse(0).Text * m_nPulse(0).Text + m_nPulse(2).Text *
m_nPulse(2).Text) / 1.414) * 8
                          Set_DecPos 1, Value, m_nStartV(0).Text, m_nSpeed(0).Text, m_nAdd(0).Text
                    End If
                          Interp_Arc 1, 3, 0, 0, m_nPulse(0).Text, m_nPulse(2).Text
        '*******************X-A arc interpolation ********************
        ElseIf m_bX.Value = vbChecked And m_bY.Value <> vbChecked And m_bZ.Value <>
vbChecked And m_bA.Value = vbChecked Then
                Setup_Speed 1, m_nStartV(0).Text, m_nSpeed(0).Text, m_nAdd(0).Text, m_nDec(0).Text,
m_nRatio(0).Text, nAddMode
              If m_nStartV(0).Text < m_nSpeed(0).Text Then
                      Set_DecMode 1, 0, 1
                      Value = (Sqr(m_nPulse(0).Text * m_nPulse(0).Text + m_nPulse(3).Text *
m_nPulse(3).Text) / 1.414) * 8
                        Set_DecPos 1, Value, m_nStartV(0).Text, m_nSpeed(0).Text, m_nAdd(0).Text
              End If
                        Interp_Arc 1, 4, 0, 0, m_nPulse(0).Text, m_nPulse(3).Text
        '*******************Y-Z arc interpolation ********************
        ElseIf m_bX.Value <> vbChecked And m_bY.Value = vbChecked And m_bZ.Value =
vbChecked And m_bA.Value <> vbChecked Then
                    Setup_Speed 2, m_nStartV(1).Text, m_nSpeed(1).Text, m_nAdd(1).Text,
m_nDec(1).Text, m_nRatio(1).Text, nAddMode
                If m_nStartV(1).Text < m_nSpeed(1).Text Then
                     Set_DecMode 2, 0, 1
                     Value  =  (Sqr(m_nPulse(1).Text  *  m_nPulse(1).Text  +  m_nPulse(2).Text  *
m_nPulse(2).Text) / 1.414) * 8
                     Set_DecPos 2, Value, m_nStartV(1).Text, m_nSpeed(1).Text, m_nAdd(1).Text
                End If
                     Interp_CcwArc 2, 3, 0, 0, m_nPulse(1).Text, m_nPulse(2).Text              '
          '*******************Y-A arc interpolation ********************
          ElseIf  m_bX.Value  <>  vbChecked  And  m_bY.Value  =  vbChecked  And  m_bZ.Value  <>
vbChecked And m_bA.Value = vbChecked Then

               Setup_Speed 2, m_nStartV(1).Text, m_nSpeed(1).Text, m_nAdd(1).Text, m_nDec(1).Text,
m_nRatio(1).Text, nAddMode
               If m_nStartV(1).Text < m_nSpeed(1).Text Then
                    Set_DecMode 2, 0, 1
                     Value  =  (Sqr(m_nPulse(1).Text  *  m_nPulse(1).Text  +  m_nPulse(3).Text  *
m_nPulse(3).Text) / 1.414) * 8
                     Set_DecPos 2, Value, m_nStartV(1).Text, m_nSpeed(1).Text, m_nAdd(1).Text
                End If
                     Interp_CcwArc 2, 4, 0, 0, m_nPulse(1).Text, m_nPulse(3).Text              '
       '**************Z-A arc interpolation *************************
       ElseIf  m_bZ.Value  =  vbChecked  And  m_bA.Value  =  vbChecked  And  m_bX.Value  <>
vbChecked And m_bY.Value <> vbChecked Then
            Setup_Speed  3,  m_nStartV(2).Text,  m_nSpeed(2).Text,  m_nAdd(2).Text,  m_nDec(2).Text,
m_nRatio(2).Text, nAddMode
            If m_nStartV(2).Text < m_nSpeed(2).Text Then
                  Set_DecMode 3, 0, 1
                  Value  =  (Sqr(m_nPulse(2).Text  *  m_nPulse(2).Text  +  m_nPulse(3).Text  *
m_nPulse(3).Text) / 1.414) * 8
                  Set_DecPos 3, Value, m_nStartV(2).Text, m_nSpeed(2).Text, m_nAdd(2).Text
            End If
            Interp_CcwArc  3,  4,  m_nPulse(2).Text  *  2,  m_nPulse(3).Text  *  2,  m_nPulse(2).Text,
m_nPulse(3).Text              ' CCW arc interpolation
       ElseIf  m_bZ.Value  <>  vbChecked  And  m_bA.Value  <>  vbChecked  And  m_bX.Value  <>
vbChecked And m_bY.Value <> vbChecked Then
               MsgBox " Please select the interpolation axis ", , " Tips "
  Else
              MsgBox " Selected axis could not carry out the arc interpolation ", , " Tips "
                     End If
          End Sub
```

1.4 Monitoring module

Monitoring module is used to real-time get the driving information of all axes, show the motion information, and at the same time, control the new driving command from being sent during the driving. Monitoring module uses the timer messages to realize its function. Codes are as follow:

```
Private Sub Timer1_Timer()
    Dim nLogPos As Long                         'logic position
    Dim nActPos As Long                         'actual position
    Dim nSpeed As Long                          'driving speed
    Dim nstatus(4) As Integer               ' state of driving
    Dim InpStatus As Integer                ' state of interpolate
    Dim nStopData(4) As Integer                 'error informations
    Get_MoveStatus 1, InpStatus, 1
    For i = 1 To 4
         Get_CurrentInf i, nLogPos, nActPos, nSpeed
         m_nLogPos(i - 1).Caption = nLogPos
         m_nActPos(i - 1).Caption = nActPos
         m_nRunSpeed(i - 1).Caption = nSpeed * g_Ratio(i - 1)
         Get_MoveStatus i, nstatus(i - 1), 0
         Get_ErrorInf i, nStopData(i - 1)
         m_nStopData(i - 1).Caption = nStopData(i - 1)
**************'detect   limit、stop0、stop1 and stop2 signal***********
 'detect positive limit signal (XLMT+ : 0,YLMT+ :6,ZLMT+ :12,ALMT+ :18)
            If Read_Input((i - 1) * 6) = 0 Then
                 m_bPLimit(i - 1).Value = 1
             Else
                 m_bPLimit(i - 1).Value = 0
             End If
    ' detect negative limit signal (XLMT- : 1,YLMT- :7,ZLMT- :13,ALMT- :19)
             If Read_Input((i - 1) * 6 + 1) = 0 Then
                 m_bNLimit(i - 1).Value = 1
             Else
                m_bNLimit(i - 1).Value = 0
             End If
     'detect stop0(XSTOP0 : 2,YSTOP0 :8,ZSTOP0 :14,ASTOP0 :20)
             If Read_Input((i - 1) * 6 + 2) = 0 Then
                   m_bStop0(i - 1).Value = 1
             Else
                   m_bStop0(i - 1).Value = 0
             End If
    'detect stop1(XSTOP1 : 3,YSTOP1 :9,ZSTOP1 :15,ASTOP1 :21)
             If Read_Input((i - 1) * 6 + 3) = 0 Then
                   m_bStop1(i - 1).Value = 1
             Else
                   m_bStop1(i - 1).Value = 0
             End If
    'detect stop2(XSTOP2 : 4,YSTOP2 :10,ZSTOP2 :16,ASTOP2 :22)
             If Read_Input((i - 1) * 6 + 4) = 0 Then
                   m_bStop2(i - 1).Value = 1
             Else
```

```
                    m_bStop2(i - 1).Value = 0
                End If
           Next i          '
         If (nstatus(0) = 0 And nstatus(1) = 0 And nstatus(2) = 0 And nstatus(3) = 0) Or InpStatus = 0
Then
             AxisPmove.Enabled = True
             InterpMove.Enabled = True
             BaseparaSet.Enabled = True
             IOTest.Enabled = True
             ArcInpMove.Enabled = True
             ConIntMove.Enabled = True
             Continuemove.Enabled = True
             ClearPos.Enabled = True
         Else
             AxisPmove.Enabled = False
             InterpMove.Enabled = False
             BaseparaSet.Enabled = False
             ArcInpMove.Enabled = False
             IOTest.Enabled = False
             Continuemove.Enabled = False
             ConIntMove.Enabled = False
             ClearPos.Enabled = False
         End If
     End Sub
```

1.5 Stop module

Stop module is used to control the incident during the driving, which demands to stop the driving of all axes immediately. The code of stop module is in the messages from the click of stop button.

```
        Private Sub Stop_Click()
         For i = 1 To 4
             StopRun i, 0
         Next i
         g_WorkStatus = 0
     End Sub
```

☞Example program of VC

2.1 PREPARATION

(1) Create a new program and save it as “VCExample.dsw”;

(2) Load the static library “ADT8948.lib” to the program according to the aforesaid method;

2.2 Motion control module

(1) Add a new category in the program, save the header file as “CtrlCard.h” and the source file as “CtrlCard.cpp”;

(2) Firstly, self-define the initialization function of motion card in the motion control module and initialize the library function that needed to be enveloped in initialization function;

(3) Continue to self-define related motion control function, such as speed setting function, single-axis motion function and interpolation function;

(4) Codes of header file “CtrlCard.h” are as follow:

```
 # ifndef __ADT8948__CARD__
 # define __ADT8948__CARD__
 /********************** Motion control module *******************
     For developing an application system of great generality, extensibility and convenient
 maintenance easily and swiftly, we envelop all the library functions by category basing on the
 card function library
 ***************************************************************/
#define   MULTIPLE    5// ratio
#define   MAXAXIS    4       //max axis number
class CCtrlCard
{
public:
   int Setup_Stop0Mode(int axis, int value, int logic);
   int Setup_Stop1Mode(int axis, int value, int logic);
   int Setup_Stop2Mode(int axis, int value, int logic);
   int Actualcount_Mode(int axis,int value, int dir,int freq);
    int AllowDec();
    int Axis_Pmove(int axis ,long value);
   int Axis_Cmove(int axis ,long value);
    int Setsoft_LimitMode1(int axis, int value);
    int Setsoft_LimitMode2(int axis, int value);
    int Setsoft_LimitMode3(int axis, int value);
   int Setup_LimitMode(int axis, int value, int logic);
   int Setup_PulseMode(int axis, int value);
   int Setup_Comp1(int axis, long value);
    int Setup_Comp2(int axis, long value);
   int Setup_Pos(int axis, long pos, int mode);
    int SetCircle_Mode(int axis, int value);
   int Write_Output(int number, int value);
   int Read_Input(int number);
   int Get_CurrentInf(int axis, long &LogPos, long &ActPos, long &Speed);
   int Get_Status(int axis, int &value, int mode);
   int Get_AllowInpStatus( int &value);
   int Set_DecPos(int axis, long value, long startv, long speed, long add);
    int Set_DecMode(int axis, int mode1, int mode2);
   int Get_ErrorInf(int axis, int &value);
   int StopRun(int axis, int mode);
   int Interp_Move2(int axis1,int axis2, long value1, long value2);
    int Interp_Move3(int axis1,int axis2,int axis3,long value1, long value2, long value3);   int
Setup_Range(int axis, long value);
   int Interp_Arc(int axis1,int axis2, long x, long y, long i,long j);
   int Interp_CcwArc(int axis1,int axis2, long x, long y, long i,long j);
int End_Board();
int ForbidDec();
int Init_Board();
int Setup_Speed(int axis ,long startv ,long speed ,long  add ,long dec,long ratio,int mode);
int Inpos_Mode(int axis, int value, int logic);
int Setup_AlarmMode(int axis,int value,int logic);
int Setup_InputFilter(int axis,int number,int value);
int Setup_FilterTime(int axis,int value);
int Setup_LockPosition(int axis,int regi,int logical);
int Get_LockStatus(int axis,int &status);
int Get_HomeStatus(int axis,int &status,int&err);
int Get_HomeError(int axis,int&err);
int Setup_VectorSpeed(int mode);
int Setup_HomeMode(int axis,long speed,int logical0, int logical1, int logical2,int offset,int dir0, int dir1, int dir2,int offsetdir,int clear,long pulse);
int Get_LockPosition(int axis,long &pos);
int Get_LibVision(int &ver);
int Manu_Pmove(int axis, long pulse);
int Manu_Continue(int axis);
int Manu_Disable(int axis);
int Inp_Command2(int axis1,int axis2,long pulse1,long pulse2);
int Inp_Command3(int axis1,int axis2,int axis3,long pulse1,long pulse2, long pulse3);
int Inp_StepMove();
int Inp_Signal2(int axis1,int axis2,long pulse1,long pulse2);
int Inp_Signal3(int axis1,int axis2,int axis3,long pulse1,long pulse2, long pulse3);
int Inp_Stop(int axis);
int Move_Home(int axis);
int Clear_HomeError(int axis);
int Set_InMove1(int axis,int axis1,long pulse,long pulse1,int logical,int mode);
int Set_InMove2(int axis, int axis1,int axis2 ,long pulse,long pulse1,long pulse2,int logical,int mode);
int Set_InMove3(int axis,int axis1,int axis2,int axis3 ,long pulse,long pulse1,long pulse2, long pulse3,int logical,int mode);
int Set_InStop1(int axis, int axis1 ,int logical, int mode);
int Set_InStop2(int axis, int axis1,int axis2,int logical, int mode);
int Set_InStop3(int axis,int logical, int mode);
int Setup_Pmove1(int axis, int axis1, long pulse, long pulse1, int regi, int term);
int Setup_Pmove2(int axis, int axis1,int axis2, long pulse, long pulse1,long pulse2, int regi, int term);
int Setup_Pmove3(int axis, int axis1,int axis2, int axis3, long pulse, long pulse1,long pulse2,long pulse3, int regi, int term);
int Setup_Stop1(int axis,int axis1,long pulse,int regi,int term,int mode);
int Setup_Stop2(int axis,int axis1,int axis2,long pulse,int regi,int term,int mode);
int Setup_Stop3(int axis,long pulse,int regi,int term,int mode);
int Symmetry_RelativeMove(int axis, long pulse, long lspd ,long hspd, double tacc, long vacc, int
mode);
   int Symmetry_AbsoluteMove(int axis, long pulse, long lspd ,long hspd, double tacc, long vacc, int mode);
   int Unsymmetry_RelativeMove( int axis, long pulse, long lspd ,long hspd, double tacc, double tdec, long vacc, int mode);
   int Unsymmetry_AbsoluteMove(int axis, long pulse, long lspd ,long hspd, double tacc, double tdec, long vacc, int mode);
   int Symmetry_RelativeLine2(int axis1, int axis2, long pulse1, long pulse2, long lspd ,long hspd, double tacc, long vacc, int mode);
   int Symmetry_AbsoluteLine2(int axis1, int axis2, long pulse1, long pulse2, long lspd ,long hspd, double tacc, long vacc, int mode);
  int Unsymmetry_RelativeLine2(int axis1, int axis2, long pulse1, long pulse2, long lspd ,long hspd, double tacc, double tdec, long vacc, int mode);
   int Unsymmetry_AbsoluteLine2(int axis1, int axis2, long pulse1, long pulse2, long lspd ,long hspd, double tacc, double tdec, long vacc, int mode);
   int Symmetry_RelativeLine3(int axis1, int axis2, int axis3, long pulse1, long pulse2, long pulse3, long lspd ,long hspd, double tacc, long vacc, int mode);
   int Symmetry_AbsoluteLine3(int axis1, int axis2, int axis3, long pulse1, long pulse2, long pulse3, long lspd ,long hspd, double tacc, long vacc, int mode);
   int Unsymmetry_RelativeLine3(int axis1, int axis2, int axis3, long pulse1, long pulse2, long pulse3, long lspd ,long hspd, double tacc, double tdec, long vacc, int mode);
   int Unsymmetry_AbsoluteLine3(int axis1, int axis2, int axis3, long pulse1, long pulse2, long pulse3, long lspd ,long hspd, double tacc, double tdec, long vacc, int mode);
   int Symmetry_Relativearc(int axis1, int axis2, long x, long y, long i, long j, int dir, long lspd ,long hspd, double tacc, long vacc, int mode);
   int Symmetry_AbsoluteArc(int axis1, int axis2, long x, long y, long i, long j, int dir, long lspd ,long hspd, double tacc, long vacc, int mode);
   int Unsymmetry_RelativeArc(int axis1, int axis2, long x, long y, long i, long j, int dir, long lspd ,long hspd, double tacc, double tdec, long vacc, int mode);
   int Unsymmetry_AbsoluteArc(int axis1, int axis2, long x, long y, long i, long j, int dir, long lspd ,long hspd, double tacc, double tdec, long vacc, int mode);
   CCtrlCard();
   int Result;    // return value
};
# endif
```

(5) Codes of source file “CtrlCard.cpp” are as follow:

```
#include "stdafx.h"
#include "DEMO.h"
#include "CtrlCard.h"
#include "adt8948.h"
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
```

```
CCtrlCard::CCtrlCard()
{

}
/*********************** Initialization function *************************
       This function contains library functions that are usually used in the initialization of the motion
card; it is the base of invoking other functions, so it must be invoked first in the example program.
       Return value<=0 means the initialization is failed, while return value >0 means the initialization
is successful.
*********************************************************/
int CCtrlCard::Init_Board()
{
      Result = adt8948_initial() ;                 // Initialization function of the card
      if (Result <= 0) return Result;
       for (int i = 1; i<=MAXAXIS; i++)
      {
            set_range (0, i, 8000000 / 5);        // Set the range, set the initial ratio to be 5
      set_command_pos (0, i, 0);                  // Reset logical counter
            set_actual_pos (0, i, 0);              // Reset actual position counter
            set_startv (0, i, 100);                // Set the start velocity
            set_speed (0, i, 100);                 // Set the driving speed
            set_acc (0, i, 0);                  // Set the acceleration
            set_actualcount_mode(0,i,0,0,0);    // Set the working mode of actual position //counter
       }
      inp_dec_enable(0);
       return 1;
}
/*************** Function of releasing control card **********************
     This function contains the library function that releases the control card; it should be invoked at
the end of the program
*********************************************************/
int CCtrlCard::End_Board ()
{
      Result = adt8948_end();
      return Result;
}
/****************** Set stop0 signal mode *******************
     Set mode of stop0 input signal
     Parameter：  axis number
                  value    0－ineffective   1－effective
                logic    0－low level stop   1－high level stop
     Defaule : ineffective
     Return    0：Correct                         1：   Wrong
   *****************************************************/
int CCtrlCard::Setup_Stop0Mode(int axis, int value, int logic)
```

```
{
     Result = set_stop0_mode(0, axis, value ,logic);
     return Result;
}
/******************* Set stop1 signal mod ********************
    Set mode of stop1 input signal
    para:   axis—axis number
              value     0—ineffective   1—effective
             logic     0—low level stop   1—high level stop
       Defaule : ineffective
    Return    0:  Correct                        1:    Wrong
  *****************************************************/
int CCtrlCard::Setup_Stop1Mode(int axis, int value, int logic)
{
     Result = set_stop1_mode(0, axis, value, logic);
     return Result;
}

/***************** Set stop2 signal mode ********************
    Set mode of stop2 input signal
    para:   axis—axis number
              value     0—ineffective   1—effective
             logic     0—low level stop   1—high level stop
    Defaule : ineffective
    Return    0:  Correct                        1:    Wrong
*******************************************************/
int CCtrlCard::Setup_Stop2Mode(int axis, int value, int logic)
{
     Result = set_stop2_mode(0, axis, value, logic);
     return Result;
}
/******************** Set the actual position counter *******************
cardno         Card number.
axis              Axis number（1-4）
value             Pulse input mode
0:                A/B pulse input        1:  Up/Down (PPIN/PMIN) pulse input
dir             Counting direction
0:              A leads B or PPIN pulse input up count
                B leads A or PMIN pulse input down count
1:                B leads A or PMIN pulse input up count
                  A leads B or PPIN pulse input down count
freq    Frequency multiplication of A/B pulse input，  Invalid for Up/Down pulse input
0: 4×        1: 2×             2: 1×
Initialization status: A/B pulse input，direction: 0，Frequency multiplication: 4×
*******************************************************/
```

```
int CCtrlCard::Actualcount_Mode(int axis,int value, int dir,int freq)
{
     Result = set_actualcount_mode(0, axis, value,dir,freq);
     return Result;
}
/******************** Set mode of pulse output **********************
     This function is used to set the working mode of pulse
     para：axis-axis number，  value-pulse mode 0－pulse+pulse 1－pulse + direction
       Return=0 correct，Return=1 wrong
      Default mode: Pulse + direction, with positive logic pulse
    Default pulse mode is Pulse + direction
      This program adopts default positive logic pulse and positive logic of direction output signal
*********************************************************/
int CCtrlCard::Setup_PulseMode(int axis, int value)
{
     Result = set_pulse_mode(0, axis, value, 0, 0);
     return Result;
}
/*******************set limit signal mode*********************
    set the mode of nLMT signal input along positive/ negative direction
    para：  axis－axis number
            value     0: sudden stop effective          1: decelerate stop effective
            logic      0: low level effective  1: high level ineffective
    Default mode: Apply positive and negative limits with low level
    Return    0：Correct                      1：   Wrong
*********************************************************/
int CCtrlCard::Setup_LimitMode(int axis, int value, int logic)
{
     Result = set_limit_mode(0, axis, value,   logic);
     return Result;
}
/***************** Set COMP + register as software limit ***************
cardno          card number
axis       axis number（1-4）
value           0：ineffective                1：effective
Return        0：Correct                     1：Wrong
Default mode：ineffective
Notice：Software position limiting always adopts acceleration to stop.
Calculating value may be over set up value. Within setup sphere it must be considered.
*********************************************************/
int CCtrlCard::Setsoft_LimitMode1(int axis, int value)
{
      Result =   set_softlimit_mode1(0, axis, value);
     return Result;
}
```

```
/**************** Set COMP - register as software limit ****************
cardno        card number
axis      axis number（1-4）
value         0：ineffective                1：effective
Return        0：Correct                1：Wrong
Default mode：ineffective
Notice：Software position limiting always adopts acceleration to stop.
Calculating value may be over set up value. Within setup sphere it must be considered.
*********************************************************/
int CCtrlCard::Setsoft_LimitMode2(int axis, int value)
{
     Result =   set_softlimit_mode2(0, axis, value);
    return Result;
}
/**************** Set COMP+/-register as software limit ****************
cardno        card number
axis      axis number（1-4）
value         0：logic position counter      1：real position counter
Return       0：Correct                1：Wrong
Default mode : logic position counter
This function is of comparative object for setup of software limiting position.
*********************************************************/
int CCtrlCard::Setsoft_LimitMode3(int axis, int value)
{
     Result =   set_softlimit_mode3(0, axis, value);
    return Result;
}
/*********** Setting of servo in-position signal nINPOS ******************
ardno         card number
axis      axis number（1-4）
value         0：ineffective                1：effective
logic         0：Effective when low electric level        1：Effective when low electric level
Return        0：Correct                1：Wrong
Default mode :   ineffective, low electric level is effective
*********************************************************/
int CCtrlCard::Inpos_Mode(int axis, int value, int logic)
{
    Result = set_inpos_mode(0, axis,value,logic);
    return Result;
}
/**************** Setting of servo alarm signal nALARM *****************
cardno        card number
axis      axis number（1-4）
value         0：ineffective              1：effective
logic         0：Effective when low electric level    1：Effective when low electric level
```

```
Return          0：Correct                    1：Wrong
Default mode :    ineffective, low electric level is effective
*******************************************************/
int CCtrlCard:: Setup_AlarmMode(int axis,int value,int logic)
{
      Result = set_alarm_mode(0, axis,value,logic);
      return Result;
}
/*********************** Set the velocity module **********************
       Check whether it is in constant speed or acceleration/deceleration according to the parameter
value.
       Set the range, which is the parameter that determines the ratio
      Set the start velocity, driving velocity and acceleration of axis
     Parameter：axis      Axis number.
                startv       start velocity
               speed         driving velocity
                add        acceleration
                dec        deceleration
                ratio        ratio
                mode      mode
     Return=0 correct，Return=1 wrong
*******************************************************/
int CCtrlCard::Setup_Speed(int axis, long startv, long speed, long add ,long dec,long ratio,int mode)
{
      //constant-speed motion
      if (startv - speed >= 0)
      {
            Result = set_range(0, axis, 8000000/ratio);
            set_startv(0, axis, startv/ratio);
            set_speed (0, axis, startv/ratio);
      }
      else//acceleration/ deceleration
      {
            if (mode == 0)
            {
                        set_dec1_mode(0,axis,0);
                        set_dec2_mode(0,axis,0);
                        set_ad_mode(0,axis,0);
                  Result = set_range(0, axis, 8000000/ratio);
                              set_startv(0, axis, startv/ratio);
                              set_speed (0, axis, speed/ratio);
                  set_acc (0, axis, add/125/ratio);
            }
            else if(mode==1)
            {
```

```
            set_dec1_mode(0,axis,1);// Asymmetry
            set_dec2_mode(0,axis,0);// Set automatic deacceleration
             set_ad_mode(0,axis,0);
            Result = set_range(0, axis, 8000000/ratio);
            set_startv(0, axis, startv/ratio);
            set_speed (0, axis, speed/ratio);
            set_acc (0, axis, add/125/ratio);
            set_dec (0, axis, dec/125/ratio);
        }
        else if(mode==2)// choose S-curve acceleration/deceleration type
        {
            float time;
            float addvar;
            long k;
              time = (float)(speed-startv)/(add/2);
            addvar=add/(time/2);
            k=(long)(62500000/addvar)*ratio;
              set_dec2_mode(0,axis,0);
              set_ad_mode(0,axis,1);
            Result = set_range(0, axis, 8000000/ratio);
            set_startv(0, axis, startv/ratio);
            set_speed (0, axis, speed/ratio);
            set_acc (0, axis, add/125/ratio);
            set_acac (0, axis,k );
        }
    }
    return Result;
}
/********************** single-axis motion **********************
     This function is to drive the single axis
     para：  axis-axis number，value-pulse of motion
     Return=0 correct，Return=1 wrong
*********************************************************/
int CCtrlCard::Axis_Pmove(int axis, long value)
{
    Result = pmove(0, axis, value);
    return Result;
}
/**************** single-axis continuous motion ********************
     This function is to continuous drive the single axis
     para：  axis-axis number，value-pulse of motion
     value-0:positive direction    1:negative direction
     Return=0 correct，Return=1 wrong
*********************************************************/
int CCtrlCard::Axis_Cmove(int axis, long value)
```

```
{
     Result = continue_move(0, axis, value);
     return Result;
}
/****************two axes interpolation function ******************
   Function        two axes linear interpolation
   Parameter
           cardno        card number
           axis1         Interpolation axis number1(1-4，indicate :X、Y、Z、A)
           axis2         Interpolation axis number2(1-4，indicate :X、Y、Z、A)
           pulse1           Moving distance of axis1
           pulse2           Moving distance of axis2
     Return value              0:correct               1:wrong
     Notice
           The interpolation speed takes the speed of the axis with smaller axis number between
axis1 and axis2 as the standard.
*********************************************************/
int CCtrlCard::Interp_Move2(int axis1,int axis2, long value1, long value2)
{
     Result = inp_move2(0, axis1,axis2, value1, value2);
     return Result;
}
/******************* 3-axis interpolation *******************
     Function:Three axes linear interpolation
     Parameter
           cardno        card number
           axis1         Interpolation axis number1(1-4，indicate :X、Y、Z、A)
           axis2         Interpolation axis number2(1-4，indicate :X、Y、Z、A)
           axis3         Interpolation axis number3(1-4，indicate :X、Y、Z、A)
           pulse1        Moving distance of axis1
           pulse2        Moving distance of axis2
     Return value              0:correct               1:wrong
     Notice
The interpolation speed takes the speed of the axis with smallest axis number among axis1, axis2
and axis3 as the standard (axis1).
*********************************************************/
int CCtrlCard::Interp_Move3(int axis1,int axis2,int axis3,long value1, long value2, long value3)
{
     Result = inp_move3(0,axis1,axis2,axis3, value1, value2, value3);
     return Result;
}
/********** Clockwise CW arc interpolation function **************
  axis1,axis2      1：X     2:Y     3：Z    4:A
  x,y             End position of arc interpolation (relative to starting point)
  i,j       Center point position of arc interpolation (relative to starting point)
```

```
               return          0：Correct        1：False
*********************************************************/
int CCtrlCard::Interp_Arc(int axis1,int axis2, long x, long y, long i,long j)
{
     Result = inp_cw_arc(0,axis1,axis2,x,y,i,j);
     return Result;
}
/**********Counterclockwise CCW arc interpolation function ***************
axis1,axis2         1：X      2：Y      3：Z     4：A
x,y        End position of arc interpolation (relative to starting point)
  i,j        Center point position of arc interpolation (relative to starting point)
  return          0：Correct      1：False
*********************************************************/
int CCtrlCard::Interp_CcwArc(int axis1,int axis2, long x, long y, long i,long j)
{
     Result = inp_ccw_arc(0,axis1,axis2,x,y,i,j);
     return Result;
}
/************** Setup of counter's variable circle function *****************
         This function can set the any maximum value of the circle counter. If the position is not linear
motion but rotating motion, it is convenient to control the position by this function.
   return          0：Correct         1：False
*********************************************************/
int CCtrlCard::SetCircle_Mode(int axis, int value)
{
     Result = set_circle_mode(0,    axis, value);
     return Result;
}
/*************** setup of wave filter time constant of input signal *********
     value           0: Wave filter is ineffective                   1: Wave filter is effective
     default mode: ineffective
*********************************************************/
int CCtrlCard::Setup_InputFilter(int axis,int number,int value)
{
     Result = set_input_filter(0, axis, number, value);
     return Result;
}
/************* setup of wave filter time constant of input signal *************
axis           axis number（1-4）
value          maximum noise scope deleted ,      delay of input signal
*********************************************************/
int CCtrlCard::Setup_FilterTime(int axis,int value)
{
     Result = set_filter_time(0, axis, value);
     return Result;
```

```
}
/********************** set lockmode *******************
function:lock the logical position and real position for all axis
para:
       axis—reference axis
       regi—register mode      |0:logical position
                                 |1:real position
       logical—level signal |0: from high to low
                                  |1:from low to high
retutrn                0： correct                1： wrong
       Note: Use IN signal of specific axis as the trigger signal
***********************************************************
int CCtrlCard::Setup_LockPosition(int axis,int regi,int logical)
{
       Result = set_lock_position(0, axis, regi , logical);
       return Result;
}
/**************** Stop the axis drive **************************
        This function is used to immediately stop or decelerate to stop the axis drive
       para： axis-axis number、 mode-stop mode(0－sudden stop, 1－decelerate stop)
        Return=0 correct， Return=1 wrong
*******************************************************/
int CCtrlCard::StopRun(int axis, int mode)
{
       if(mode == 0)              // sudden stop
        {
                Result = sudden_stop(0, axis);
        }
        else                              // decelerate stop
        {
                Result = dec_stop(0, axis);
        }
        return Result;
}
/*************** get status of motion ***********************
        get status of single-axis motion or interpolation
        para： axis-axis number
                  value-Indicator of motion status
                          (0： Drive completed,Non-0: Drive in process)
              mode(0-single-axis motion， 1－interpolation)
        Return=0 correct， Return=1 wrong
*******************************************************/
int CCtrlCard::Get_Status(int axis, int &value, int mode)
{
       if (mode==0)                    // status of single-axis motion
```

```
            Result=get_status(0,axis,&value);
      else                              // status of interpolation
            Result=get_inp_status(0,&value);
      return Result;
}
/******************* get synchronous action state *****************
function:get synchronous action state
para:
      cardno              card number
      axis                axis number
      status—0|haven't run synchronous
                 1|run synchronous
retutrn              0：correct              1：wrong
Note: This function could tell whether the position lock has been executed
**************************************************************/
int CCtrlCard::Get_LockStatus(int axis,int &status)
{
          Result=get_lock_status(0,    axis, &status);
          return Result;
}
/************************ home diving status**********************
function:get the information of home position,judge the error message
pare：
      cardno            card number
      axis              axis number
       status—drive status 0|finish driving
                                1|driving
      err—error message     0|correct
                                1|wrong
retutrn              0：correct              1：wrong
**************************************************************/
int CCtrlCard::Get_HomeStatus(int axis,int &status,int &err)
{
          Result=get_home_status(0,    axis, &status,&err);
          return Result;
}
/********* Get the error information of home position return ************
Function: Get the error information of home position return
err—error message
                    0|correct
               not 0|wrong
                    D0: comp+Limit
                    D1: comp-Limit
                    D2: LMT+Limit
                    D3: LMT-Limit
```

```
                        D4: Servo alarm
                        D5: Emergency stop
                        D6:Z-phase signal arrives ahead of time
retutrn                 0：correct                1：wrong
**************************************************************/
int CCtrlCard::Get_HomeError(int axis,int &err)
{
          Result=get_home_error(0,    axis, &err);
         return Result;
}
/***********************deceleration enable***********************
          This function is to enable the deceleration during the driving
        retutrn                 0：correct                1：wrong
***************************************************************/
int CCtrlCard::AllowDec()
{
          Result=inp_dec_enable(0);
         return Result;
}
/****************deceleration disable ****************************
            This function is to disable the deceleration during the driving
      retutrn                 0：correct                1：wrong
****************************************************************/
int CCtrlCard::ForbidDec( )
{
      Result=inp_dec_disable(0);
      return Result;
}
/************** Get the error stop information of axis ********************
            This function is to get the stop information of axis
           value: Index of error status    0：No error    1：Value of two bytes
           retutrn                 0：correct                1：wrong
*****************************************************************/
int CCtrlCard::Get_ErrorInf(int axis, int &value)
{
      Result=get_stopdata(0,axis,&value);
      return Result;
}
/****************** Get the status of continuous interpolation *************
            This function is to get the writable status of continuous interpolation
            value：Index of interpolation status    0: Unwritable    1: Writable
           retutrn                 0：correct                1：wrong
****************************************************************/
int CCtrlCard::Get_AllowInpStatus( int &value)
{
       Result=get_inp_status2(0,&value);
       return Result;
}
/******************** Set the mode of deceleration ******************
              deceleration for symmetry/asymmetry/automatic/manual
       mode1: 0:symmetry acceleration/ deceleration
                   1:asymmetry acceleration/ deceleration
       mode2: 0: automatic                   1:manual
             retutrn                 0：correct                  1：wrong
******************************************************************/
int CCtrlCard::Set_DecMode(int axis, int mode1, int mode2)
{
       int result1,result2;
       result1=set_dec1_mode(0,axis,mode1);
       result2=set_dec2_mode(0,axis,mode2);
       Result=result1 && result2;
       return Result;
}
/*******************Set the mode of deceleration ****************
              This function is to set the symmetrical/dissymmetrical and automatic/manual deceleration
             retutrn                 0：correct                  1：wrong
******************************************************************/
int CCtrlCard::Set_DecPos(int axis, long value, long startv, long speed, long add)
{
       float addtime;
       long    DecPulse;       // Pulse cost in deceleration
       addtime=float(speed-startv)/add;
       DecPulse=long((startv+speed)*addtime)/2;
       Result=set_dec_pos(0,axis,value-DecPulse);
        return Result;
}
/**************** Read the input point ***************************
          This function is to read the single input point
          para：number- Input points (0 ~ 34)
          Return：0  －  low level，1  －  high level，-1  －  error
******************************************************************/
int CCtrlCard::Read_Input(int number)
{
       Result = read_bit(0, number);
       return Result;
}
/********************* Single point output function ********************
        This function is to output single point signal
      Para: number- output port (0 ~ 31),
                    value 0-low level、1－high level
        Return=0 correct，Return=1 wrong
*******************************************************************
int CCtrlCard::Write_Output(int number, int value)
{
       Result = write_bit(0, number, value);
       return Result;
}
/****************** Position counter setting ***********************
          This function is to set the logical position and actual position
          para：axis- axis number,          pos- Value of position
            mode 0－Set the logical position,    Non-0－Set the actual position
          Return=0 correct，Return=1 wrong
*******************************************************************/
int CCtrlCard::Setup_Pos(int axis, long pos, int mode)
{
       if(mode==0)
       {
              Result = set_command_pos(0, axis, pos);
       }
       else
       {
              Result = set_actual_pos(0, axis, pos);
       }
       return Result;
}
/******************** COMP+register setting ****************
        cardno          card number
       axis       axis number
        value            range（-2147483648~+2147483647）
       retutrn                 0：correct                   1：wrong
*****************************************************************/
int CCtrlCard::Setup_Comp1(int axis, long value)
{
       Result= set_comp1(0, axis, value);
        return Result;
}

/****************** set COMP- register **********************
            cardno              card number
           axis            axis number
            value          range（-2147483648~+2147483647）
         retutrn                 0：correct                   1：wrong
*******************************************************************/
int CCtrlCard::Setup_Comp2(int axis, long value)
{
       Result= set_comp2(0, axis, value);
        return Result;
}
/*************** Set the mode of fixed linear speed*************
Function: Set the mode of fixed linear speed
para:
       cardno card number
       mode—0| not use the fixed linear speed                  1| Use the fixed linear speed
retutrn                  0：correct                    1：wrong
       Note: Linear speed means vector speed; fixed linear speed could ensure the composite speed to
be fixed during the interpolation
**************************************************************/
int CCtrlCard::Setup_VectorSpeed(int mode)
{
       Result= set_vector_speed(0, mode);
        return Result;
}
/********************** set home position mode*********************
Function: Set the back-to-home mode of specific axis
para:
            logical0—stop0|0: stop with low level
                              |1: stop with high level
                              |-1: ineffective
            logical1—stop1|0: stop with low level
                              |1: stop with high level
                              |-1: ineffective
            logical2—stop2|0: stop with low level
                              |1: stop with high level
                              |-1: ineffective
             0ffset— 0| Do not offset from home                         1| Offset from home
            dir0—Direction |0:Positive                                  |1:Negative
            dir1—Direction |0:Positive                                  |1:Negative
            dir2—Direction |0:Positive                                  |1:Negative
            offsetdir—Offset direction |0:Positive                            |1:Negative
speed—Low search speed, it's required to be lower than start velocity of high speed
            reset—Reset the counter?|0: Yes                |1: No
return                  0：correct                    1：wrong
      Note:
             (1) Zero-return is divided into four steps:
                   |The first step: swiftly approach stop0 (logical0 close origin setup);
                   |The second step: slowly approach stop1 (logical1 origin setup);
                   |The third step: slowly approach stop2 (logical2 encoder Z-phase);
                   |The fourth step: Deviation range (For working origin);
               (2) The above four steps can decide whether to be carried out by choosing among logical0,
logical1、logical2 and offset
```

```
            (3) Able to use a proximity switch to act as several signals
*****************************************************************/
int CCtrlCard::Setup_HomeMode(int axis,long speed,int logical0, int logical1, int logical2,int
offset,int dir0, int dir1, int dir2,int offsetdir,int clear,long pulse)
{
      Result=set_home_mode(0,axis,speed,logical0,logical1,logical2,offset,dir0,dir1,dir2,offsetdir,clear
,pulse);
       return Result;
}
/********************* Get the motion information ********************
This function is to reflect a feedback of current logic position, actual position and operating speed of
axis
para：axis-Axis number., LogPos-Logical position, ActPos-Actual position, Speed- driving speed
Return=0 correct，Return=1 wrong
**********************************************************/
int CCtrlCard::Get_CurrentInf(int axis, long &LogPos, long &ActPos, long &Speed)
{
      Result = get_command_pos(0, axis, &LogPos);
       get_actual_pos (0, axis, &ActPos);
       get_speed (0, axis, &Speed);
      return Result;
}
/*********************get lock position***********************
Function: Get the locked position
para:
      cardno          card number
      axis            axis number
      pos              lock position
Return        0：Correct                 1：Wrong
*****************************************************************/
int CCtrlCard::Get_LockPosition(int axis,long &pos)
{
      Result= get_lock_position(0, axis, &pos);
       return Result;
}
/************* Function of library function's version number **************
Function: Get the version of library function
para:
        ver-Version number (The first two numbers are the major version numbers, the last two
numbers are secondary version numbers)
Return        0：Correct                 1：Wrong
*****************************************************************/
int CCtrlCard::Get_LibVision(int &ver)
{
      Result= get_lib_vision( &ver);
```

```
        return Result;
}
/************ Quantitative drive function of external signal ****************
function: Quantitative drive function of external signal
para:
       cardno          card number
       axis            axis number
       pulse           pulse
Return          0：Correct                1：Wrong
       Note: (1) Send out quantitative pulse, but the drive does not start immediately until the external
signal level changes
                (2)Ordinary button and handwheel are acceptable.
**************************************************************/
int CCtrlCard::Manu_Pmove(int axis, long pulse)
{
       Result= manual_pmove(0, axis, pulse);
        return Result;
}
/************* Continuous drive function of external signal **************
function: Continuous drive function of external signal
para:
       cardno          card number
       axis            axis number
Return          0：Correct                1：Wrong
       Note: (1) Send out fixed pulse, but the drive does not start immediately until the level of external
signal changes
                (2) Ordinary button and handwheel are acceptable.
**************************************************************/
int CCtrlCard::Manu_Continue(int axis)
{
       Result= manual_continue(0, axis);
        return Result;
}
/************** Shut down the enabling of external signal drive *************
function: Shut down the enabling of external signal drive
para:
       cardno          card number
       axis            axis number
Return          0：Correct                1：Wrong
**************************************************************/
int CCtrlCard::Manu_Disable(int axis)
{
       Result= manual_disable(0, axis);
        return Result;
}
```

```
/********** Command type two axes stepping interpolation setup **********
Function:Set the data of two axes command stepping interpolation
para:
      The same with the 2-axis interpolation function
Return        0：Correct                 1：Wrong
      Functional description: Send out data of stepping interpolation, the drive does not start but
waiting for drive command function
***************************************************************/
int CCtrlCard::Inp_Command2(int axis1,int axis2,long pulse1,long pulse2)
{
      Result= inp_step_command2(0, axis1,axis2,pulse1,pulse2);
       return Result;
}
/********* Command type three axes stepping interpolation setup ***********
Function: Set the data of three axes command stepping interpolation
para:
      The same with three-axis interpolation function
Return        0：Correct                 1：Wrong
      Functional description: Send out data of stepping interpolation, the drive does not work but
waiting for drive command function
***************************************************************/
int CCtrlCard::Inp_Command3(int axis1,int axis2,int axis3,long pulse1,long pulse2, long pulse3)
{
      Result= inp_step_command3(0, axis1,axis2,axis3,pulse1,pulse2,pulse3);
       return Result;
}

/********** Stepping interpolation command driving function ************
function: Execute command stepping interpolation in single step
para:
      cardno          card number
Return        0：Correct                 1：Wrong
      Functional description: Drive the stepping motion in the form of command according to the
parameter of the set function
***************************************************************/
int CCtrlCard::Inp_StepMove()
{
      Result= inp_step_move(0);
       return Result;
}
/*********** Setting function of 2-axis signal stepping interpolation ************
Function: Set the data of 2-axis signal stepping interpolation
parameter:
      The same with the interpolation function of 2-axis
Return        0：Correct                 1：Wrong
```

```
      Functional description: Send out data of stepping interpolation, the drive does not work but
waiting for the external signal level to drop to low level
******************************************************************/
int CCtrlCard::Inp_Signal2(int axis1,int axis2,long pulse1,long pulse2)
{
      Result= inp_step_signal2(0, axis1,axis2,pulse1,pulse2);
       return Result;
}
/********* Setting function of three-axis signal stepping interpolation **********
function: Set the data of three-axis stepping interpolation
para:
      The same with the three-axis interpolation function
return value                  0： correct                  1： wrong
      Functional description: Send out data of stepping interpolation, the drive does not work but
waiting for the external signal level to drop to low level
*****************************************************************/
int CCtrlCard::Inp_Signal3(int axis1,int axis2,int axis3,long pulse1,long pulse2, long pulse3)
{
      Result= inp_step_signal3(0,    axis1,axis2,axis3,pulse1,pulse2,pulse3);
       return Result;
}
/******************** Stop stepping interpolation ******************
function: Stop the stepping interpolation
para:
      cardno            card number
      axis              axis number
return value                  0： correct                  1： wrong
      Note: Axis that is in the state of stepping interpolation must carry out the stop command of
stepping interpolation before going to other drives
*****************************************************************/
int CCtrlCard::Inp_Stop(int axis)
{
      Result= inp_step_stop(0, axis);
       return Result;
}
/************* back-home drive function **********************
function: Back-home by following the set mode
para:
      cardno            card number
      axis              axis number
return value                  0： correct                  1： wrong
*****************************************************************/
int CCtrlCard::Move_Home(int axis)
{
      Result= home(0, axis);
```

```
        return Result;
}
/************** Clear back-to-home error information ********************
function: Clear error information of home
para:
       cardno           card number
       axis              axis number
return value                0:   correct                 1:   wrong
******************************************************************/
int CCtrlCard::Clear_HomeError(int axis)
{
       Result= clear_home_error(0, axis);
        return Result;
}
/************** Single axis follow moving setting function ***************
function: Function of IN simultaneous movement setup
para:
              axis—active axis number
              axis1—driven axis number1
              pulse—pulse of active axis
              pulse1—pulse of drive axis
              logical—Electricity level logic of IN signal (0:low electric 1:high electric)
       mode—The active axis has moved? 0| Yes     1| No
              return value                0:   correct                 1:   wrong
       Note: Use IN signal of active axis as trigger signal
*******************************************************************/
int CCtrlCard::Set_InMove1(int axis,int axis1,long pulse,long pulse1,int logical,int mode)
{
       Result= set_in_move1(0, axis,axis1,pulse,pulse1,logical,mode);
        return Result;
}
/************2-axis follow moving setting function ****************
function: Function of IN simultaneous movement setup
para:
              axis—active axis number
              axis1—driven axis number
              axis2—driven axis number
              pulse—pulse of active axis
              pulse1—pulse of drive axis1
              pulse2—pulse of drive axis2
              logical—level signal |0: From high to low          |1: From low to high
       mode—The active axis has moved? 0| Yes     1| No
return value                0:   correct                 1:   wrong
       Note: Use IN signal of active axis as trigger signal
*******************************************************************/
```

```
int CCtrlCard::Set_InMove2(int axis, int axis1,int axis2 ,long pulse,long pulse1,long pulse2,int
logical,int mode)
{
    Result= set_in_move2(0, axis,axis1,axis2,pulse,pulse1,pulse2,logical,mode);
     return Result;
}
/***********3-axis follow moving setting function ***************
function: Function of IN simultaneous movement setup
para:
        axis—active axis number
        axis1—driven axis number1
        axis2—driven axis number2
        axis3—driven axis number3
        pulse—pulse of active axis
        pulse1—pulse of driven axis 1
        pulse2—pulse of driven axis 2
        pulse3—pulse of driven axis 3
        logical—level signal |0: From high to low        |1: From low to high
        mode—The active axis has moved? 0| Yes    1| No
return value          0：correct           1：wrong
    Note: Use IN signal of active axis as trigger signal
*******************************************************************/
int CCtrlCard::Set_InMove3(int axis,int axis1,int axis2,int axis3 ,long pulse,long pulse1,long pulse2,
long pulse3,int logical,int mode)
{
    Result= set_in_move3(0, axis,axis1,axis2,axis3,pulse,pulse1,pulse2,pulse3,logical, mode);
     return Result;
}
/*************** Single axis follow stopping setting function **************
function: Set the IN synchronization action
para:
        axis—active axis number
        axis1—driven axis number1
        logical—level signal |0: From high to low        |1: From low to high
        mode—active axis stop? |0:yes                          |1:no
return value          0：correct           1：wrong
    Note:
        Signal changes detected, dead axle has stopped and the driving status of drive axle can be set
******************************************************************/
int CCtrlCard::Set_InStop1(int axis, int axis1 ,int logical, int mode)
{
    Result= set_in_stop1(0, axis,axis1,logical,mode);
     return Result;
}
/************* 2-axis follow stopping setting function ************
```

```
function: Set the IN synchronization action
para:
            axis—active axis number
            axis1—driven axis number1
            axis1—driven axis number2
            logical—level signal |0: From high to low      |1: From low to high
            mode—active axis stop? |0:yes                  |1:no
return value                0:  correct             1:  wrong
      Note:
            Signal changes detected, dead axle has stopped and the driving status of drive axle can be set
*******************************************************************/
int CCtrlCard::Set_InStop2(int axis, int axis1,int axis2,int logical, int mode)
{
      Result= set_in_stop2(0, axis,axis1,axis2,logical,mode);
       return Result;
}

/***************** 3-axis follow stopping setting function ****************
function: Set the IN synchronization action
para:
            axis—active axis number
            logical—level signal |0: From high to low      |1: From low to high
              mode—active axis stop? |0:yes                |1:no
return value                0:  correct             1:  wrong
Note:
            Signal changes detected, dead axle has stopped and the driving status of drive axle can be set
********************************************************************/
int CCtrlCard::Set_InStop3(int axis,int logical, int mode)
{
      Result= set_in_stop3(0, axis,logical,mode);
       return Result;
}
/******* Setting of single-axis drive when arriving at the target position *********
function: Setting of single-axis drive when arriving at the target position
para:
            axis—active axis number
            axis2—driven axis number
            pulse1—pluse of active axis 1
            pulse2—pluse of driven axis 2
            regi—0|comp+   Select the compare register          1|comp-
            –term—0|>=   Select the compare register                1|<
return value                0:  correct             1:  wrong
*******************************************************************/
int CCtrlCard::Setup_Pmove1(int axis, int axis1, long pulse, long pulse1, int regi, int term)
{
       Result= set_comp_pmove1(0, axis,axis1,pulse,pulse1,regi,term);
        return Result;
}
/********** Setting of two-axis drive when arriving at the target position ******
function: Setting of two-axis drive when arriving at the target position
para:
              axis—active axis number
              axis1—driven axis number1
              axis2—driven axis number2
              pulse—pulse of active axis
              pulse1—pulse of driven axis1
              pulse2—pulse of driven axis2
              regi—0|comp+    Select the compare register          1|comp-
                term—0|>=    Select the compare register              1|<
return value                 0：correct                  1：wrong
**************************************************************/
int CCtrlCard::Setup_Pmove2(int axis, int axis1,int axis2, long pulse, long pulse1,long pulse2, int regi, int term)
{
       Result= set_comp_pmove2(0, axis,axis1,axis2,pulse,pulse1,pulse2,regi,term);
        return Result;
}
/********* Setting of three-axis drive when arriving at the target position *********
function: Setting of three-axis drive when arriving at the target position
para:
              axis—active axis number
              axis1—driven axis number1
              axis2—driven axis number2
              axis2—driven axis number3
              pulse—pulse of active axis
              pulse1—pulse of driven axis1
              pulse2—pulse of driven axis 2
              pulse3—pulse of driven axis3
              regi—0|comp+    Select the compare register         1|comp-
                term—0|>=    Select the compare register             1|<
return value                 0：correct                  1：wrong
****************************************************************/
int CCtrlCard::Setup_Pmove3(int axis, int axis1,int axis2, int axis3, long pulse, long pulse1,long pulse2,long pulse3, int regi, int term)
{
          Result=set_comp_pmove3(0,axis ,axis1,axis2,axis3,pulse,pulse1,pulse2,pulse3,regi,term);
        return Result;
}
/******** Setting of stopping drive when arriving at the target position ********
function: Setting of stopping drive when arriving at the target position
```

```
para:
            axis—active axis number
            axis1—driven axis number1
            pulse—target position of active axis
            regi—0|comp+    Select the compare register           1|comp-
              term—0|>=    Select the compare register              1|<
              mode—|0: active axis stop                          |1: active axis does not stop
return value              0：correct                1：wrong
*****************************************************************/
int CCtrlCard::Setup_Stop1(int axis,int axis1,long pulse,int regi,int term,int mode)
{
      Result= set_comp_stop1(0, axis,axis1,pulse,regi,term,mode);
       return Result;
}
/****** Setting of stopping drive when arriving at the target position *********
function: Setting of stopping drive when arriving at the target position
para:
            axis—active axis number
            axis1—driven axis number1
            axis2—driven axis number2
            pulse—target position of active axis
            regi—0|comp+    Select the compare register           1|comp-
              term—0|>=    Select the compare register              1|<
              mode—|0: active axis stop                          |1: active axis does not stop
return value              0：correct                1：wrong
****************************************************************/
int CCtrlCard::Setup_Stop2(int axis,int axis1,int axis2,long pulse,int regi,int term,int mode)
{
      Result= set_comp_stop2(0, axis,axis1,axis2,pulse,regi,term,mode);
       return Result;
}
/******* Setting of stopping drive when arriving at the target position *********
function: Setting of stopping drive when arriving at the target position
para:
            axis—active axis number
            pulse—target position of active axis
            regi—0|comp+    Select the compare register             1|comp-
              term—0|>=    Select the compare register                   1|<
              mode—|0: active axis stop                          |1: active axis does not stop
return value              0：correct                1：wrong
*****************************************************************/
int CCtrlCard::Setup_Stop3(int axis,long pulse,int regi,int term,int mode)
{
      Result= set_comp_stop3(0, axis,pulse,regi,term,mode);
       return Result;
}
//****************** Composite drives *****************
// Note: The following functions are added for the convenience of customers
//**********************************************

/************ Symmetrical relative movement of single-axis ***************
function: Refer to the current position and perform quantitative movement in the symmetrical
acceleration/deceleration
para:
           cardno-card number
          axis---axis number
          pulse-- Pulse
          lspd--- Low speed
          hspd--- High speed
           tacc--- Time of acceleration (Unit: sec)
          vacc--- Change rate of acceleration/deceleration
          mode--- Mode (trapezoid(0) or S-curve(1))
return value                0：correct                1：wrong
****************************************************************/
int CCtrlCard::Symmetry_RelativeMove(int axis, long pulse, long lspd ,long hspd, double tacc, long
vacc, int mode)
{
      Result= symmetry_relative_move(0,axis, pulse, lspd ,hspd, tacc, vacc, mode);
       return Result;
}
/************** Symmetrical absolute movement of single-axis ***********
function: Refer to the position of zero point and perform quantitative movement in the symmetrical
acceleration/deceleration
  para:
                cardno -card number
                axis ---axis number
                pulse -- Pulse
                lspd --- Low speed
                hspd --- High speed
                tacc--- Time of acceleration (Unit: sec)
                vacc --- Change rate of acceleration/deceleration
                mode--- Mode (trapezoid(0) or S-curve(1))
  return value 0: correct     1: wrong
           ****************************************************************/
int CCtrlCard::Symmetry_AbsoluteMove(int axis, long pulse, long lspd ,long hspd, double tacc, long
vacc, int mode)
{
      Result= symmetry_absolute_move(0,axis, pulse, lspd ,hspd, tacc, vacc, mode);
       return Result;
}
```

```
/********* asymmetrical relative movement of single-axis ***********
function: Refer to the current position and perform quantitative movement in the asymmetrical
acceleration/deceleration
para:
          cardno -card number
          axis ---axis number
          pulse -- Pulse
          lspd --- Low speed
          hspd --- High speed
           tacc--- Time of acceleration (Unit: sec)
          tdec--- Time of deceleration (Unit: sec)
          vacc--- Change rate of acceleration/deceleration
          mode--- Mode (trapezoid(0) or S-curve(1))
return value                0：correct                1：wrong
***************************************************************/
int CCtrlCard::Unsymmetry_RelativeMove( int axis, long pulse, long lspd ,long hspd, double tacc,
double tdec, long vacc, int mode)
{
      Result= unsymmetry_relative_move(0,    axis, pulse, lspd ,hspd,tacc, tdec, vacc, mode);
       return Result;
}
/********** asymmetrical absolute movement of single-axis ************
function: Refer to the position of zero point and perform quantitative movement in the asymmetrical
acceleration/deceleration
para:
           cardno-card number
          axis---axis number
          pulse -- Pulse
          lspd --- Low speed
          hspd --- High speed
           tacc--- Time of acceleration (Unit: sec)
          tdec--- Time of deceleration (Unit: sec)
          vacc--- Change rate of acceleration/deceleration
          mode--- Mode (trapezoid(0) or S-curve(1))
return value                0：correct                1：wrong
**************************************************************/
int CCtrlCard::Unsymmetry_AbsoluteMove(int axis, long pulse, long lspd ,long hspd, double tacc,
double tdec, long vacc, int mode)
{
      Result= unsymmetry_absolute_move(0, axis, pulse, lspd ,hspd, tacc, tdec, vacc, mode);
       return Result;
}
/***** Relative movement of two-axis symmetrical linear interpolation ********
function: Refer to current position and perform linear interpolation in symmetrical
acceleration/deceleration
```

```
para:
           cardno-card number
          axis1---axis number1
          axis2---axis number2
          pulse1-- pulse 1
          pulse2-- pulse 2
          lspd --- Low speed
          hspd --- High speed
           tacc--- Time of acceleration (Unit: sec)
          vacc--- Change rate of acceleration/deceleration
          mode--- Mode (trapezoid(0) or S-curve(1))
return value                0：correct                  1：wrong
**************************************************************/
int CCtrlCard::Symmetry_RelativeLine2(int axis1, int axis2, long pulse1, long pulse2, long lspd ,long
hspd, double tacc, long vacc, int mode)
{
      Result= symmetry_relative_line2(0, axis1, axis2, pulse1,pulse2, lspd ,hspd,tacc, vacc,mode);
       return Result;
}
/****** Two axes symmetric linear interpolation absolute moving ********
function: Refer to the position of zero point and perform linear interpolation in symmetrical
acceleration/deceleration
para:
           cardno-card number
          axis1---axis number1
          axis2---axis number2
          pulse1—pulse of axis 1
          pulse2-- pulse of axis 2
          lspd --- Low speed
          hspd --- High speed
           tacc--- Time of acceleration (Unit: sec)
          vacc--- Change rate of acceleration/deceleration
          mode--- Mode (trapezoid(0) or S-curve(1))
return value                0：correct                  1：wrong
**************************************************************/
int CCtrlCard::Symmetry_AbsoluteLine2(int axis1, int axis2, long pulse1, long pulse2, long
lspd ,long hspd, double tacc, long vacc, int mode)
{
      Result= symmetry_absolute_line2(0, axis1, axis2, pulse1, pulse2, lspd ,hspd, tacc, vacc,mode);
       return Result;
}
/****** Two axes asymmetric linear interpolation relative moving ********
function: Refer to current position and perform linear interpolation in asymmetric
acceleration/deceleration.
para:
           cardno-card number
          axis1---axis number1
          axis2---axis number2
          pulse1—pulse of axis 1
          pulse2-- pulse of axis 2
           lspd --- Low speed
          hspd --- High speed
         tacc--- Time of acceleration (Unit: sec)
          tdec--- Time of deceleration (Unit: sec)
          vacc--- Change rate of acceleration/deceleration
          mode--- Mode (trapezoid(0) or S-curve(1))
return value              0：correct              1：wrong
*************************************************************/
int CCtrlCard::Unsymmetry_RelativeLine2(int axis1, int axis2, long pulse1, long pulse2, long
lspd ,long hspd, double tacc, double tdec, long vacc, int mode)
{
      Result= unsymmetry_relative_line2(0, axis1,axis2, pulse1, pulse2, lspd , hspd, tacc,  tdec,
vacc,   mode);
       return Result;
}
/******* Two axes asymmetric linear interpolation absolute moving *********
function: Refer to the position of zero point and perform linear interpolation in asymmetric
acceleration/deceleration
para:
            cardno-card number
            axis1---axis number1
            axis2---axis number2
            pulse1—pulse of axis 1
            pulse2-- pulse of axis 2
            lspd --- Low speed
            hspd --- High speed
            tacc--- Time of acceleration (Unit: sec)
            tdec--- Time of deceleration (Unit: sec)
            vacc--- Change rate of acceleration/deceleration
            mode--- Mode (trapezoid(0) or S-curve(1))
return value              0：correct              1：wrong
*************************************************************/
int CCtrlCard::Unsymmetry_AbsoluteLine2(int axis1, int axis2, long pulse1, long pulse2, long
lspd ,long hspd, double tacc, double tdec, long vacc, int mode)
{
      Result= unsymmetry_absolute_line2(0, axis1, axis2, pulse1, pulse2, lspd ,hspd, tacc, tdec, vacc,
mode);
       return Result;
}
/***** Three axes symmetric linear interpolation relative moving ******
```

```
function: Refer to current position and perform linear interpolation in symmetric
acceleration/deceleration
para:
            cardno-card number
            axis1---axis number1
            axis2---axis number2
            axis3---axis number3
            pulse1-- pulse of axis 1
            pulse2-- pulse of axis 2
            pulse3-- pulse of axis 3
            lspd --- Low speed
            hspd --- High speed
            tacc--- Time of acceleration (Unit: sec)
            vacc--- Change rate of acceleration/deceleration
            mode--- Mode (trapezoid(0) or S-curve(1))
return value            0：correct            1：wrong
*************************************************************/
int CCtrlCard::Symmetry_RelativeLine3(int axis1, int axis2, int axis3, long pulse1, long pulse2, long
pulse3, long lspd ,long hspd, double tacc, long vacc, int mode)
{
     Result= symmetry_relative_line3(0,axis1, axis2, axis3, pulse1, pulse2, pulse3, lspd ,hspd, tacc,
vacc, mode);
      return Result;
}
/******Three axes symmetric linear interpolation absolute moving *********
function: Refer to the position of zero point and perform linear interpolation in symmetric
acceleration/deceleration.
para:
            cardno-card number
            axis1---axis number1
            axis2---axis number2
            axis3---axis number3
            pulse1-- pulse of axis 1
            pulse2-- pulse of axis 2
            pulse3-- pulse of axis 3
            lspd --- Low speed
            hspd --- High speed
            tacc--- Time of acceleration (Unit: sec)
            vacc--- Change rate of acceleration/deceleration
            mode--- Mode (trapezoid(0) or S-curve(1))
return value            0：correct            1：wrong
*************************************************************/
int CCtrlCard::Symmetry_AbsoluteLine3(int axis1, int axis2, int axis3, long pulse1, long pulse2, long
pulse3, long lspd ,long hspd, double tacc, long vacc, int mode)
{
    Result= symmetry_absolute_line3(0, axis1, axis2,axis3,pulse1, pulse2,  pulse3,  lspd , hspd,
tacc,   vacc,   mode);
     return Result;
}
```

/****** Three axes asymmetric linear interpolation relative moving **********

function: Refer to current position and perform linear interpolation in asymmetric acceleration/deceleration

para:

cardno-card number
axis1---axis number1
axis2---axis number2
axis3---axis number3
pulse1-- pulse of axis 1
pulse2-- pulse of axis 2
pulse3-- pulse of axis 3
lspd --- Low speed
hspd --- High speed
tacc--- Time of acceleration (Unit: sec)
tdec--- Time of deceleration (Unit: sec)
vacc--- Change rate of acceleration/deceleration
mode--- Mode (trapezoid(0) or S-curve(1))

return value   0：correct   1：wrong

***/

```
int CCtrlCard::Unsymmetry_RelativeLine3(int axis1, int axis2, int axis3, long pulse1, long pulse2,
long pulse3, long lspd ,long hspd, double tacc, double tdec, long vacc, int mode)
{
    Result= unsymmetry_relative_line3(0, axis1, axis2, axis3, pulse1, pulse2, pulse3, lspd ,hspd, tacc,
tdec, vacc, mode);
     return Result;
}
```

/****** Three axes asymmetric linear interpolation absolute moving ******

function: Refer to the position of zero point and perform linear interpolation in asymmetric acceleration/deceleration

para:

cardno-card number
axis1---axis number1
axis2---axis number2
axis3---axis number3
pulse1-- pulse of axis 1
pulse2-- pulse of axis 2
pulse3-- pulse of axis 3
lspd --- Low speed
hspd --- High speed
tacc--- Time of acceleration (Unit: sec)
tdec--- Time of deceleration (Unit: sec)

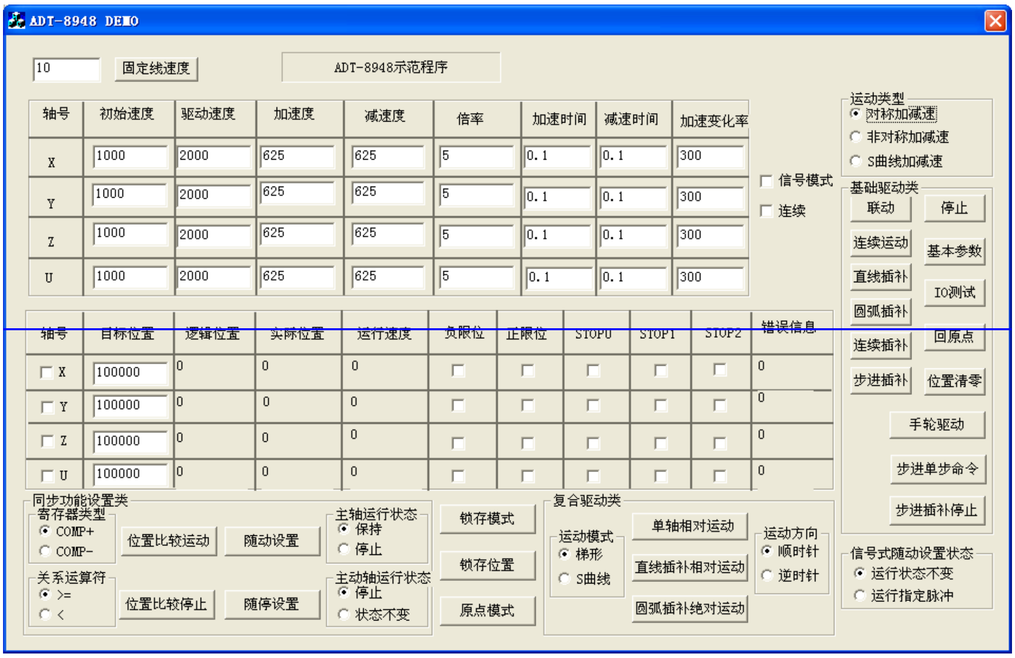
```
          vacc--- Change rate of acceleration/deceleration
          mode--- Mode (trapezoid(0) or S-curve(1))
*************************************************************/
int CCtrlCard::Unsymmetry_AbsoluteLine3(int axis1, int axis2, int axis3, long pulse1, long pulse2,
long pulse3, long lspd ,long hspd, double tacc, double tdec, long vacc, int mode)
{
    Result= unsymmetry_absolute_line3(0, axis1,   axis2,   axis3,   pulse1,   pulse2,   pulse3,
lspd , hspd,   tacc,   tdec,   vacc,   mode);
     return Result;
}
/********* Two axes symmetric arc interpolation relative moving ********
function: Refer to current position and perform arc interpolation in symmetric
acceleration/deceleration.
para:
        cardno-card number
        axis1---axis number1
        axis2---axis number2
        x、y---- Coordinates of arc end point (Refer to current point, that is, starting point of arc)
        i、j---- Centre coordinate (Refer to current point, that is, starting point of arc)
        dir----- Moving direction (0-Clockwise,1-Anti-clockwise)
        lspd --- Low speed
        hspd --- High speed
        tacc--- Time of acceleration (Unit: sec)
        vacc--- Change rate of acceleration/deceleration
        mode--- Mode (trapezoid(0) or S-curve(1))
return value          0：correct          1：wrong
*************************************************************/
int CCtrlCard::Symmetry_Relativearc(int axis1, int axis2, long x, long y, long i, long j, int dir, long
lspd ,long hspd, double tacc, long vacc, int mode)
{
    Result= symmetry_relative_arc(0, axis1,axis2,x,y,i,j, dir,   lspd , hspd,   tacc,   vacc,   mode);

     return Result;
}
/********** Two axes symmetric arc interpolation absolute moving **********
function: Refer to the position of zero point and perform arc interpolation in symmetric
acceleration/deceleration
para:
        cardno-card number
        axis1---axis number1
        axis2---axis number2
        x、y---- Coordinates of arc end point (Refer to current point, that is, starting point of arc)
        i、j---- Centre coordinate (Refer to current point, that is, starting point of arc)
        dir----- Moving direction (0-Clockwise,1-Anti-clockwise)
        lspd --- Low speed
```

```
          hspd --- High speed
          tacc--- Time of acceleration (Unit: sec)
          vacc--- Change rate of acceleration/deceleration
          mode--- Mode (trapezoid(0) or S-curve(1))
return value            0：correct            1：wrong
**********************************************************************/
int CCtrlCard::Symmetry_AbsoluteArc(int axis1, int axis2, long x, long y, long i, long j, int dir, long
lspd ,long hspd, double tacc, long vacc, int mode)
{
     Result= symmetry_absolute_arc(0, axis1,axis2,x,y,i,j,dir,lspd,hspd,tacc, vacc, mode);
      return Result;
}
/******* Two axes asymmetric arc interpolation relative moving ********
function: Refer to current position and perform arc interpolation in asymmetric
acceleration/deceleration
para:
          cardno-card number
          axis1---axis number1
          axis2---axis number2
          x、y---- Coordinates of arc end point (Refer to current point, that is, starting point of arc)
          i、j---- Centre coordinate (Refer to current point, that is, starting point of arc)
          dir----- Moving direction (0-Clockwise,1-Anti-clockwise)
          lspd --- Low speed
          hspd --- High speed
          tacc--- Time of acceleration (Unit: sec)
          tdec--- Time of deceleration (Unit: sec)
          vacc--- Change rate of acceleration/deceleration
          mode--- Mode (trapezoid(0) or S-curve(1))
return value            0：correct            1：wrong
**********************************************************************/
int CCtrlCard::Unsymmetry_RelativeArc(int axis1, int axis2, long x, long y, long i, long j, int dir, long
lspd ,long hspd, double tacc, double tdec, long vacc, int mode)
{
     Result= unsymmetry_relative_arc(0, axis1,axis2,x,y,i,j,dir,lspd,hspd,tacc, tdec, vacc, mode);
      return Result;
}
/****** Two axes asymmetric arc interpolation absolute moving *******
function: Refer to current position and perform arc interpolation in asymmetric
acceleration/deceleration
para:
          cardno-card number
          axis1---axis number1
          axis2---axis number2
          x、y---- Coordinates of arc end point (Refer to current point, that is, starting point of arc)
          i、j---- Centre coordinate (Refer to current point, that is, starting point of arc)
          dir----- Moving direction (0-Clockwise,1-Anti-clockwise)
          lspd --- Low speed
          hspd --- High speed
          tacc--- Time of acceleration (Unit: sec)
          tdec--- Time of deceleration (Unit: sec)
          vacc--- Change rate of acceleration/deceleration
          mode--- Mode (trapezoid(0) or S-curve(1))
return value              0：correct              1：wrong
*****************************************************************/
int CCtrlCard::Unsymmetry_AbsoluteArc(int axis1, int axis2, long x, long y, long i, long j, int dir,
long lspd ,long hspd, double tacc, double tdec, long vacc, int mode)
{
     Result= unsymmetry_absolute_arc(0, axis1,axis2,x,y,i,j,dir,lspd,hspd,tacc, tdec, vacc, mode);
      return Result;
}
```

2.3 Function-Realization-Module

2.3.1 Interface design



ADT-8948 DEMO

10 固定线速度 ADT-8948示范程序

轴号	初始速度	驱动速度	加速度	减速度	倍率	加速时间	减速时间	加速变化率
X	1000	2000	625	625	5	0. 1	0. 1	300
Y	1000	2000	625	625	5	0. 1	0. 1	300
Z	1000	2000	625	625	5	0. 1	0. 1	300
U	1000	2000	625	625	5	0. 1	0. 1	300

□ 信号模式
□ 连续

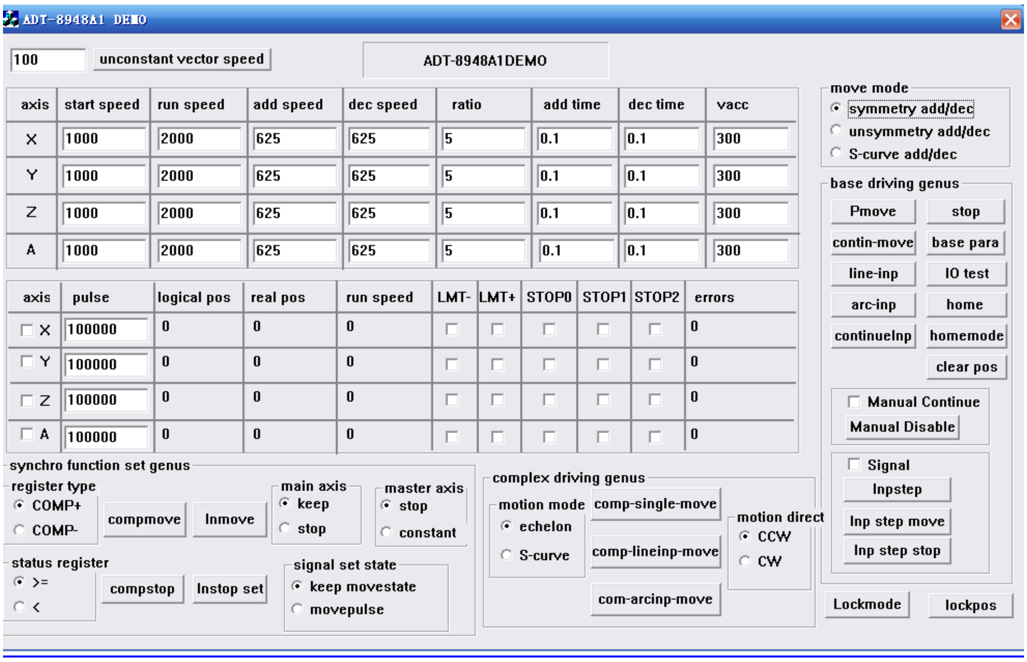
轴号	目标位置	逻辑位置	实际位置	运行速度	负限位	正限位	STOP0	STOP1	STOP2	错误信息
□ X	100000	0	0	0	□	□	□	□	□	0
□ Y	100000	0	0	0	□	□	□	□	□	0
□ Z	100000	0	0	0	□	□	□	□	□	0
□ U	100000	0	0	0	□	□	□	□	□	0

运动类型
● 对称加减速
○ 非对称加减速
○ S曲线加减速

同步功能设置类
寄存器类型
● COMP+
○ COMP-
关系运算符
● >=
○ <
主轴运行状态
● 保持
○ 停止
主动轴运行状态
● 停止
○ 状态不变

复合驱动类
运动模式
● 梯形
○ S曲线
运动方向
● 顺时针
○ 逆时针

信号式随动设置状态
● 运行状态不变
○ 运行指定脉冲



ADT-8948A1 DEMO

100 unconstant vector speed

ADT-8948A1DEMO

axis	start speed	run speed	add speed	dec speed	ratio	add time	dec time	vacc
X	1000	2000	625	625	5	0.1	0.1	300
Y	1000	2000	625	625	5	0.1	0.1	300
Z	1000	2000	625	625	5	0.1	0.1	300
A	1000	2000	625	625	5	0.1	0.1	300

axis	pulse	logical pos	real pos	run speed	LMT-	LMT+	STOP0	STOP1	STOP2	errors
☐ X	100000	0	0	0	☐	☐	☐	☐	☐	0
☐ Y	100000	0	0	0	☐	☐	☐	☐	☐	0
☐ Z	100000	0	0	0	☐	☐	☐	☐	☐	0
☐ A	100000	0	0	0	☐	☐	☐	☐	☐	0

move mode: ● symmetry add/dec ○ unsymmetry add/dec ○ S-curve add/dec

synchro function set genus — register type: ● COMP+ ○ COMP-; main axis: ● keep ○ stop; master axis: ● stop ○ constant; status register: ● >= ○ <; signal set state: ● keep movestate ○ movepulse

complex driving genus — motion mode: ● echelon ○ S-curve; motion direct: ● CCW ○ CW

☐ Manual Continue

☐ Signal

Note:

(1) Speed settings—Set the start velocity, driving velocity and acceleration of every axis; position settings—set the drive pulse of every axis; drive information—real-time display the logical position, actual position and operating speed of every axis.

(2) Driver objects—Select the driver object and confirm the axis that involved in linkage or interpolation;

(3) Linkage—to send single-axis drive command to all axis of selected driver object; interpolation—to send interpolation command to all axis of selected driver object;

stop—stop the pulse output of all axis;

All the data above are in Pulse

2.3.2 Initial code of motion card is located in window initialization. Codes added by user are as follow:

```
if (g_CtrlCard.Init_Board() <= 0){
    MessageBox( " fail to initialize card!");
   return false;
}
 else
   MessageBox ("It is applicable to motion card!");
//******* Set the default start velocity to be 1000*********
m_nStartvX = 1000;
m_nStartvY = 1000;
m_nStartvZ = 1000;
m_nStartvA = 1000;
//******** Set the default driving velocity to be 2000*******
m_nSpeedX   = 2000;
m_nSpeedY   = 2000;
m_nSpeedZ   = 2000;
m_nSpeedA   = 2000;
//******* Set the default acceleration to be 625*********
```

```
m_nAddX       = 625;
m_nAddY       = 625;
m_nAddZ       = 625;
m_nAddA       = 625;
//******** Set the default ratio to be 5**************
 m_nRatioX   = 5;
m_nRatioY   = 5;
m_nRatioZ   = 5;
m_nRatioA   = 5;
//********Set the default target position to be 100000****
m_nPulseX   = 100000;
m_nPulseY   = 100000;
m_nPulseZ   = 100000;
m_nPulseA   = 100000;
//******** Set the default deceleration to be 625**********
 m_nDecA=625;
m_nDecX=625;
m_nDecY=625;
m_nDecZ=625;
//********* Set the time of acceleration to be 0.1 (Unit: sec) *******
m_dAccX=0.1;
m_dAccY=0.1;
m_dAccZ=0.1;
m_dAccA=0.1;
//******* Set the time of deceleration to be 0.1 (Unit: sec) ********
m_dDecX=0.1;
m_dDecY=0.1;
m_dDecZ=0.1;
m_dDecA=0.1;
//******* Set the change rate of acceleration to be 300**********
m_nVaccX = 300;
m_nVaccA = 300;
m_nVaccY = 300;
m_nVaccZ = 300;
//******* Mode of motion (0: trapezoid, 1: S-shape curve) **********
m_nMoveType=0;
//******* Moving direction (0-clockwise, 1-anti-clockwise) ********
m_nCcwDir=0;
//***** Type of comparison register (0: comp+; 1: comp-)******
m_nPcomp = 0;
//********** Relation operator (0:>=,1:<)************
m_nBigAmount = 0;
//****** Running status of major axis (0: Hold, 1: Stop) *******
m_nKeep = 0;
m_nStop1 =0;
```

```
        m_nMoveState =0;
         UpdateData(FALSE);
        SetTimer(MAINTIMER,100,NULL);                // Start the timer
```

2.3.3 Pmove code is in the message from the click of linkage button, it will send out corresponding drive command according to the selected object, codes are as below:

```
          void CDEMODlg::OnButtonPmove()
        {
        UpdateData(TRUE);
        long Startv[]={m_nStartvX,m_nStartvY,m_nStartvZ,m_nStartvA};    //start speed
        long Speed[]={m_nSpeedX,m_nSpeedY,m_nSpeedZ,m_nSpeedA};          //driving speed
        long Dec[]={m_nDecX,m_nDecY,m_nDecZ,m_nDecA };           //deceleration
        long Add[] ={m_nAddX,m_nAddY,m_nAddZ,m_nAddA};           //acceleration
        long Ratio[]={m_nRatioX,m_nRatioY,m_nRatioZ,m_nRatioA};    //ratio
        if(m_bX )
        {
        g_CtrlCard.Setup_Speed(1, m_nStartvX, m_nSpeedX, m_nAddX, m_nDecX, m_nRatioX,
m_nAddMode);
                  g_CtrlCard.Axis_Pmove(1, m_nPulseX);

        }
        if(m_bY )
        {
                      g_CtrlCard.Setup_Speed(2, m_nStartvY, m_nSpeedY, m_nAddY, m_nDecY,m_nRatioY,
m_nAddMode);
                      g_CtrlCard.Axis_Pmove(2, m_nPulseY);

        }
        if(m_bZ )
        {
        g_CtrlCard.Setup_Speed(3, m_nStartvZ, m_nSpeedZ, m_nAddZ, m_nDecZ, m_nRatioZ,
m_nAddMode);

                       g_CtrlCard.Axis_Pmove(3, m_nPulseZ);
        }
        if(m_bA )
        {
        g_CtrlCard.Setup_Speed(4, m_nStartvA, m_nSpeedA, m_nAddA, m_nDecA, m_nRatioA,
m_nAddMode);
                g_CtrlCard.Axis_Pmove(4, m_nPulseA);

        }

}
```

2.3.4 Interpolation code is in the message from the click of interpolation button, it will send out corresponding drive command according to the selected object, codes are as below:

```
void CDEMODlg::OnButtonInpmove()
{
      UpdateData();
      long Startv[]={m_nStartvX,m_nStartvY,m_nStartvZ,m_nStartvA};
      long Speed[]={m_nSpeedX,m_nSpeedY,m_nSpeedZ,m_nSpeedA};
```

```
    long Add[]   ={m_nAddX,m_nAddY,m_nAddZ,m_nAddA};
    long Dec[]   ={m_nDecX,m_nDecY,m_nDecZ,m_nDecA};
    long Pulse[]={m_nPulseX,m_nPulseY,m_nPulseZ,m_nPulseA};
    long Ratio[]={m_nRatioX,m_nRatioY,m_nRatioZ,m_nRatioA};
//*********** Random 2-axes interpolation **********//
if(m_bX && m_bY && !m_bZ && !m_bA)              // X-Y interpolation
{
          g_CtrlCard.Setup_Speed(1,Startv[0],Speed[0],Add[0],Dec[0],Ratio[0],m_nAddMode);
          g_CtrlCard.Interp_Move2(1,2,    Pulse[0], Pulse[1]);
}
else if(m_bX && !m_bY && m_bZ && !m_bA)                 // X-Z interpolation
{
g_CtrlCard.Setup_Speed(1, Startv[0], Speed[0], Add[0], Dec[0],Ratio[0],m_nAddMode);
g_CtrlCard.Interp_Move2(1,3,    Pulse[0], Pulse[2]);
}
else if(m_bX && !m_bY && !m_bZ && m_bW)                 // X-A interpolation
{
g_CtrlCard.Setup_Speed(1, Startv[0], Speed[0], Add[0], Dec[0],Ratio[0],m_nAddMode);
g_CtrlCard.Interp_Move2(1,4,    Pulse[0], Pulse[3]);
}
else if(!m_bX && m_bY && m_bZ && !m_bA)                 // X-Z interpolation
{
g_CtrlCard.Setup_Speed(2, Startv[1], Speed[1], Add[1], Dec[1],Ratio[1],m_nAddMode);
g_CtrlCard.Interp_Move2(2,3,    Pulse[1], Pulse[2]);
}
else if(!m_bX && m_bY && !m_bZ && m_bA)                 // X-A interpolation
{
g_CtrlCard.Setup_Speed(2, Startv[1], Speed[1], Add[1], Dec[1],Ratio[1],m_nAddMode);
g_CtrlCard.Interp_Move2(2,4,    Pulse[1], Pulse[3]);
}
else if(!m_bX && !m_bY && m_bZ && m_bA)               // Z-A interpolation
{
g_CtrlCard.Setup_Speed(3, Startv[2], Speed[2], Add[2], Dec[2],Ratio[2],m_nAddMode);
g_CtrlCard.Interp_Move2(3,4, Pulse[2], Pulse[3]);
}
//***************** Three-axis interpolation ****************//
else if(m_bX && m_bY && m_bZ && !m_bA)                //X-Y-Z interpolation
{
g_CtrlCard.Setup_Speed(1, Startv[0], Speed[0], Add[0],Dec[0] ,Ratio[0],m_nAddMode);
g_CtrlCard.Setup_Speed(3, Speed[0], Speed[0], Add[2],Dec[2] ,Ratio[0],m_nAddMode);
g_CtrlCard.Interp_Move3( 1,2,3,Pulse[0], Pulse[1], Pulse[2]);
}
else if(m_bX && m_bY && !m_bZ && m_bA)                //X-Y-A interpolation
{
g_CtrlCard.Setup_Speed(1, Startv[0], Speed[0], Add[0], Dec[0] , Ratio[0], m_nAddMode);
g_CtrlCard.Setup_Speed(3, Speed[0], Speed[0], Add[2], Dec[2] , Ratio[0], m_nAddMode);
g_CtrlCard.Interp_Move3( 1,2,4,Pulse[0], Pulse[1], Pulse[3]);
}
else if(m_bX && !m_bY && m_bZ && m_bA)                //X-Z-A interpolation
{
g_CtrlCard.Setup_Speed(1, Startv[0], Speed[0], Add[0], Dec[0] , Ratio[0], m_nAddMode);
g_CtrlCard.Setup_Speed(3, Speed[0], Speed[0], Add[2], Dec[2] , Ratio[0], m_nAddMode);
g_CtrlCard.Interp_Move3( 1,3,4,Pulse[0], Pulse[2], Pulse[3]);
}
else if(!m_bX && m_bY && m_bZ && m_bA)                    //Y-Z-A interpolation
{
g_CtrlCard.Setup_Speed(2, Startv[1], Speed[1], Add[1], Dec[1] , Ratio[1], m_nAddMode);
g_CtrlCard.Setup_Speed(3, Speed[0], Speed[0], Add[2], Dec[2] , Ratio[0], m_nAddMode);
g_CtrlCard.Interp_Move3( 2,3,4,Pulse[1], Pulse[2], Pulse[3]);
}
else if(!m_bX && !m_bY && !m_bZ && !m_bA)
{
        MessageBox("Please select the interpolation axis!"," Tips ");
}
else
{
        MessageBox("Please select the two-axis interpolation!"," Tips ");
}
}
```

2.3.5 Continuous interpolation code is in the message from the click of continuous interpolation button, here only perform four segments continuous interpolation of X and Y axis. Codes are as below:

```
void CDEMODlg::OnContinueInp()
{
UpdateData(TRUE);
int status1,status2;
long startv[]={m_nStartvX,m_nStartvY,m_nStartvZ,m_nStartvA};
long Speed[]={m_nSpeedX,m_nSpeedY,m_nSpeedZ,m_nSpeedA};
long Dec[]   ={m_nDecX,m_nDecY,m_nDecZ,m_nDecA};
long Pos[]={m_nPulseX,m_nPulseY,m_nPulseZ,m_nPulseA};
long Add[]={m_nAddX,m_nAddY,m_nAddZ,m_nAddA};
long Ratio[]={m_nRatioX,m_nRatioY,m_nRatioZ,m_nRatioA};
if((m_nAddMode==0)||(m_nAddMode==1))
{
// It only performs continuous interpolation for X-Y axis here
if(m_bX && m_bY&&!m_bZ&&!m_bA)
{
}
else
{
         MessageBox("Please select X-Y axis!");
         return;
}
        g_WorkStatus=1;
         g_CtrlCard.Setup_Speed(1,startv[0],Speed[0],Add[0],Dec[0],Ratio[0],m_nAddMode);
// Acceleration/Deceleration
if(startv[0]<=Speed[0])
{
// The parameter value behind is 1, which means manual deceleration, because it can //only
decelerates manually
g_CtrlCard.Set_DecMode(1,0,1);
g_CtrlCard.Set_DecPos(1,Pos[0],startv[0],Speed[0],Add[0]);
```

```
}
// Interpolation deceleration disable response function
g_CtrlCard.ForbidDec();
//*********** The first segment **********
g_CtrlCard.Interp_Move2(1,2,Pos[0],Pos[1]);
while(true)
{
DoEvent();

g_CtrlCard.Get_ErrorInf(1,status1);

if(status1!=0 || g_WorkStatus==0)
goto err;

//status2:     0: Unwritable       1：   Writable
g_CtrlCard.Get_AllowInpStatus(status2);

if(status2!=0)break;

}
//********** The second segment ***********
g_CtrlCard.Interp_Move2(1,2,Pos[0],Pos[1]);

while(true)
{
DoEvent();

g_CtrlCard.Get_ErrorInf(1,status1);

if(status1!=0 || g_WorkStatus==0)
goto err;

//status2:    0: Unwritable       1：   Writable
g_CtrlCard.Get_AllowInpStatus(status2);

if(status2!=0)
break;
}

//*********** The third segment ***********
g_CtrlCard.Interp_Move2(1,2,Pos[0],Pos[1]);
while(true)
{
DoEvent();

g_CtrlCard.Get_ErrorInf(1,status1);

if(status1!=0 || g_WorkStatus==0)goto err;

g_CtrlCard.Get_AllowInpStatus(status2);
```

```
if(status2!=0)break;
}

//********** The last segment *********
g_CtrlCard.AllowDec();//Interpolation deceleration enable response function
g_CtrlCard.Interp_Move2(1,2,Pos[0],Pos[1]);
while(true)
{
     DoEvent();
     g_CtrlCard.Get_ErrorInf(1,status1);

   if(status1!=0 || g_WorkStatus==0) // g_ Report the error when WorkStatus is stop.
      goto err;
      g_CtrlCard.Get_Status(1,status2,1);
      if(status2==0)break;
  }
      g_WorkStatus=0;
      return;
  err:
      MessageBox("wrong!");
      return;
  }
  else
  {
      MessageBox("Not allowed to select S-curve acceleration/deceleration!");
  }
}
```

2.3.6 Arc interpolation code is in the message from the click of arc interpolation button, it will send out corresponding drive command according to the selected object, codes are as below:

```
void CDEMODlg::OnArcInp()
{
    UpdateData(TRUE);
    long startv[]={m_nStartvX,m_nStartvY,m_nStartvZ,m_nStartvA};
    long Speed[]={m_nSpeedX,m_nSpeedY,m_nSpeedZ,m_nSpeedA};
    long Add[]={m_nAddX,m_nAddY,m_nAddZ,m_nAddA};
    long Dec[]={m_nDecX,m_nDecY,m_nDecZ,m_nDecA};
    long Ratio[]={m_nRatioX,m_nRatioY,m_nRatioZ,m_nRatioA};
    long Pos[]={m_nPulseX,m_nPulseY,m_nPulseZ,m_nPulseA};
    long value;
    long double tmpvalue;
   if((m_nAddMode==0)||(m_nAddMode==1))
    {
       if( m_bX && m_bY && !m_bZ && !m_bA)
       {
    g_CtrlCard.Setup_Speed(1,startv[0],Speed[0],Add[0],Dec[0],Ratio[0],m_nAddMode);

          if(startv[0]<Speed[0]) // Acceleration/Deceleration
          {
              //****************************************//
```

```
                // The parameter value in the middle is 0, which means symmetrical //
                acceleration/deceleration
                // The last parameter value 1 means manual deceleration //
                //***************************************//
                g_CtrlCard.Set_DecMode(1,0,1);
                // Definition of X-Y axis distance pulse data type. If there is errors //in the definition
                of data type, it will cause malfunction.
                tmpvalue=double(Pos[0])*double(Pos[0])+double(Pos[1])*double(Pos[1]);
                value=long(sqrt(tmpvalue)/1.414)*8;
                g_CtrlCard.Set_DecPos(1,value,startv[0],Speed[0],Add[0]);
            }
            // When choosing X-Y axis, carry out clockwise interpolation
            g_CtrlCard.Interp_Arc(1,2,0,0,Pos[0],Pos[1]);
        }
         else if( m_bX && !m_bY && m_bZ && !m_bA)
        {
    g_CtrlCard.Setup_Speed(1,startv[0],Speed[0],Add[0],Dec[0],Ratio[0],m_nAddMode);
            if(startv[0]<Speed[0]) // Acceleration/Deceleration
            {
                g_CtrlCard.Set_DecMode(1,0,1);
                     tmpvalue= double(Pos[0])*double(Pos[0])+double(Pos[1])*double(Pos[1]);
                value=long(sqrt(tmpvalue)/1.414)*8;
                g_CtrlCard.Set_DecPos(1,value,startv[0],Speed[0],Add[0]);
            }
            // When choosing X-Z axis, carry out clockwise interpolation
            g_CtrlCard.Interp_Arc(1,3,0,0,Pos[0],Pos[1]);
        }
        else if( m_bX && !m_bY && !m_bZ && m_bA)
        {
    g_CtrlCard.Setup_Speed(1,startv[0],Speed[0],Add[0],Dec[0],Ratio[0],m_nAddMode);

            if(startv[0]<Speed[0]) // Acceleration/Deceleration
            {
                g_CtrlCard.Set_DecMode(1,0,1);
                    tmpvalue = double(Pos[0])*double(Pos[0])+double(Pos[1])*double(Pos[1]);
                value=long(sqrt(tmpvalue)/1.414)*8;
                g_CtrlCard.Set_DecPos(1,value,startv[0],Speed[0],Add[0]);
                }
            // When choosing X-A axis, carry out clockwise interpolation
            g_CtrlCard.Interp_Arc(1,4,0,0,Pos[0],Pos[1]);
        }
        else if(m_bZ && !m_bA&&!m_bX&&m_bY)
        {
    g_CtrlCard.Setup_Speed(2,startv[1],Speed[1],Add[1],Dec[1],Ratio[1],m_nAddMode);
            if(startv[2]<Speed[2]) // Acceleration/Deceleration
            {
                g_CtrlCard.Set_DecMode(2,0,1);
        tmpvalue = double(Pos[1])*double(Pos[1])+double(Pos[2])*double(Pos[2]);
                value=long(sqrt(tmpvalue)/1.414)*8;
                g_CtrlCard.Set_DecPos(2,value,startv[1],Speed[1],Add[1]);
            }
g_CtrlCard.Interp_CcwArc(2,3,Pos[1]*2,Pos[2]*2,Pos[1],Pos[2]);
        }
        else if(!m_bZ && m_bA&&!m_bX&&m_bY)
        {
    g_CtrlCard.Setup_Speed(2,startv[1],Speed[1],Add[1],Dec[1],Ratio[1],m_nAddMode);
            if(startv[2]<Speed[2]) // Acceleration/Deceleration
            {
                g_CtrlCard.Set_DecMode(2,0,1);
        tmpvalue = double(Pos[1])*double(Pos[1])+double(Pos[3])*double(Pos[3]);
                value=long(sqrt(tmpvalue)/1.414)*8;
    g_CtrlCard.Set_DecPos(2,value,startv[1],Speed[1],Add[1]);           }
            // When choosing Y-Z axis, carry out anti-clockwise arc interpolation
            g_CtrlCard.Interp_CcwArc(2,4,Pos[1]*2,Pos[3]*2,Pos[1],Pos[3]);
        }
        else if(m_bZ && m_bA&&!m_bX&&!m_bY)
        {
    g_CtrlCard.Setup_Speed(3,startv[2],Speed[2],Add[2],Dec[2],Ratio[2],m_nAddMode);
            if(startv[2]<Speed[2]) // Acceleration/Deceleration
            {
                g_CtrlCard.Set_DecMode(3,0,1);
        tmpvalue = double(Pos[2])*double(Pos[2])+double(Pos[3])*double(Pos[3]);
                value=long(sqrt(tmpvalue)/1.414)*8;
                g_CtrlCard.Set_DecPos(3,value,startv[2],Speed[2],Add[2]);           }
            // When choosing Z-A axis, carry out anti-clockwise arc interpolation
            g_CtrlCard.Interp_CcwArc(3,4,Pos[2]*2,Pos[3]*2,Pos[2],Pos[3]);
        }
        else if(!m_bX && !m_bY && !m_bZ && !m_bA)
        {
            MessageBox("Please select the interpolation axis!!");
        }
        else
        {
            MessageBox("Selected axis could not carry out the arc interpolation. Please select
two-axis interpolation!");
        }
    }
    else
    {
        MessageBox("Not allow to select S-curve acceleration/deceleration!");
    }
}
```

2.4 Monitoring module

Monitoring module is used to real-time get the driving information of all axes, show the motion information, and at the same time, control the new driving command from being sent during the driving. Monitoring module uses the timer messages to realize its function. Codes are as follow:

```
void CDEMODlg::OnTimer(UINT nIDEvent)
{
    long log,act,spd;
    int Stopdata[4];
UINT nID1[]={IDC_POS_LOGX,IDC_POS_LOGY,IDC_POS_LOGZ,IDC_POS_LOGA};
UINT nID2[]={IDC_POS_ACTX,IDC_POS_ACTY,IDC_POS_ACTZ,IDC_POS_ACTA};
UINT
nID3[]={IDC_RUNSPEED_X,IDC_RUNSPEED_Y,IDC_RUNSPEED_Z,IDC_RUNSPEED_A};
    UINT nID4[]={m_nRatioX,m_nRatioY,m_nRatioZ,m_nRatioA};
    UINT     nID5[]={IDC_STOPDATA_X,     IDC_STOPDATA_Y,     IDC_STOPDATA_Z,
IDC_STOPDATA_A};
    CStatic *lbl;
    CString str,stopinf;
    int status[4];
    for (int i=1; i<MAXAXIS+1; i++)
    {
        g_CtrlCard.Get_CurrentInf(i,log,act,spd);
        lbl=(CStatic*)GetDlgItem(nID1[i-1]);
        str.Format("%ld",log);
        lbl->SetWindowText(str);
        lbl=(CStatic*)GetDlgItem(nID2[i-1]);
        str.Format("%ld",act);
        lbl->SetWindowText(str);
        lbl=(CStatic*)GetDlgItem(nID3[i-1]);
        str.Format("%ld",spd*nID4[i-1]);
        lbl->SetWindowText(str);
        g_CtrlCard.Get_Status(i,status[i-1],0);
        g_CtrlCard.Get_ErrorInf(i, Stopdata[i-1]);
          stopinf.Format("%d",Stopdata[i-1]);
        lbl=(CStatic*)GetDlgItem(nID5[i-1]);
        lbl->SetWindowText(stopinf);
    }
    UINT nIDIN[]={    IDC_LIMIT_X,IDC_LIMIT_X2,
    IDC_STOP0_X,IDC_STOP1_X,    IDC_STOP2_X3,
      IDC_LIMIT_Y,IDC_LIMIT_Y2,
      IDC_STOP0_Y,IDC_STOP1_Y2, IDC_STOP2_Y3,
      IDC_LIMIT_Z,IDC_LIMIT_Z2,                      IDC_STOP0_Z,IDC_STOP1_Z,
      IDC_STOP2_Z2,
    IDC_LIMIT_A,IDC_LIMIT_A2,
    IDC_STOP0_A,IDC_STOP1_A ,   IDC_STOP2_A2 };
    int io[]={0,1,2,3,4,    6,7,8,9,10,    12,13,14,15,16,    18,19,20,21,22};
    CButton *btn;
    int value;
    for (i=0; i<20; i++)
    {
        value=g_CtrlCard.Read_Input(io[i]);                    // Read the signal
        btn=(CButton*)GetDlgItem(nIDIN[i]);
        btn->SetCheck(value==0?1:0);
    }
    //*******************************************************
    CDialog::OnTimer(nIDEvent);
}
```

2.5 Stop module

Stop module is used to control the incident during the driving, which demands to stop the driving of all axes immediately. The code of stop module is in the messages from the click of stop button.

```
    void CDEMODlg::OnButtonStop()
    {
        for (int i = 1; i<=MAXAXIS; i++)
        {
            g_CtrlCard.StopRun(i,0);
         }
        g_WorkStatus=0;
    }
```

Chapter XI Trouble shooting

☞ Motion card detection failed

If the motion card could not be detected during its usage, refer to the following methods to rule out the problem.

(1) Make sure to follow the installation instructions of control card to install the driver step by step. In Win2000 and WinXp, port driver must be installed to ensure that there are dynamic library file of control card in system folders (system32 or System);

(2) Check to see if the motion card is connected well with the slot. Re-insert the card or change the slot to test, besides, use the eraser to clean the golden finger of control card, then, test it again;

(3) Check the system device manager to see if there is any conflict between motion card and other hardware. Take off other cards, such as sound card and network card, when the PCI card is used; PC104 card is able to adjust the DIP switch and reset the base address, the base address used in the initialization of program must be the same with the actual base address;

(4) Check to see if there is any problem with the operation system, this could be done by re-installing other version of operation system;

Follow the above steps, if the motion card still can not be detected, it is recommended to change another motion card to test, so as to check whether the card is broken;

☞Malfunction of motor

Motor error appears when the motion card is working well. You could follow the following situations to clear the trouble.

(1) The motor does not work when the motion card sends out pulse

- ➢ Check whether the control card is connected well with the terminal board;
- ➢ Check whether the pulse and direction signal line of motor driver are correctly connected to the terminal board;
- ➢ Check whether the external power supply of servo driver is well connected;
- ➢ Check whether the servo driver and stepping motor driver are in the state of alarm, if so, check the reason according to the corresponding code of the alarm;
- ➢ Check whether the servo SON is well connected or whether the servo motor is in the state of magnetization etc;
- ➢ If it is the problem of servo motor, please check the control mode of driver, our control card supports “position control”;
- ➢ Check whether the motor or driver is broken

(2) There is abnormal scream when the stepping motor is running, and the motor shows obviously out of step.

- ➢ The speed of controller is too fast, please calculate the speed of motor, 10~15 rounds/sec is normal for stepping motor;
- ➢ Mechanical part is blocked or the resistance of machine is too big;
- ➢ The selected motor is not proper, please change for a high-torque motor;
- ➢ Please check the current and voltage of driver, set the current to be 1.2 times of rated current and supply voltage to be within rated range;
- ➢ Check the start velocity of controller, generally speaking, the start velocity is about 0.5~1

and time of acceleration/deceleration is above 0.1 sec;

(3) There are obvious vibration and noise when the servo and stepping motor work

- Position loop gain and speed loop gain of servo driver are too high, lower them on the condition that the positional accuracy allows;
- The rigidity of machine is too bad, adjust the structure;
- The selected stepping motor is not proper, please change for a high-torque motor;
- The speed of stepping motor is in resonance region of motor, please avoid the resonance region or enlarge the subdivision.

(4) Motor is not positioned well

- Check the screw pitch and the pulse/rev of motor to see if they are compliance with the set parameter of practical system, viz. pulse equivalent;
- If it is the problem of servo motor, then, increase the position loop gain and speed loop gain;
- Check the screw gap of the machine, test the screw opposite gap with dial indicator, adjust the screw if there is gap;
- If the position is not accurate due to the uncertain time and position, then, check to see if there is external interference signal;
- The model of motor is not proper that it shows vibration and out of step;

(5) Motor has no direction

- Check to see if there is any problem with the DR+ and DR-connection or if the connection is firm;
- Check to see if the pulse mode of the control card is compliance with actual driver mode, this card supports "Pulse+Direction" and "Pulse+Pulse";
- Check the stepper motor to see if the motor wire is broken or malfunction etc.

☞Switch quantity input error

If there is malfunction in detection of some input signals during the debugging and working of the system, check according to the following methods:

(1) No signal input

(1) Check the line according to the wiring diagram of normal switch and proximity switch mentioned before, to see if it is connected correctly, make sure the "optical coupler common" of input signal is connected with positive pole of inner/external power supply (+12V or 24V)

(2) Our input switch of I\O point is NPN type, if it is not that type, please check the type of switch and the wiring;

(3) Check whether the optical coupler is broken. When the line is well, and the input state does not change when the input point is in the state of turnoff and closure, use the universal meter to test the optical coupler to see if it is punctured, then, change the optical coupler to solve this problem;

(4) Check whether the 12V or 24V switching power supply is normal;

(5) The switch is broken；

(2) Signal disappear from time to time

- Check whether there is interference, test the signal state in I\O test; if there exist interference, then, add type 104 monolithic capacitor or use the shielding conductor etc.
- Obvious vibration or abnormal stop during the normal running，please check whether

there is interference in limit switch signal or the performance of limit switch is reliable;

- Check whether the external wiring connection is connected well;

(3) **Inaccurate back to home**

- The speed is too fast, slow down the speed of back to home;
- There is interference on external signal, please check the interference source;
- Direction of back to home is wrong;
- Installation position of back to home switch is improper or the switch is loosened;

(4) Invalid limit

- Test the limit switch in I\O test to see if it is in effect;
- The speed of manual/auto processing is too fast;
- There is interference on external signal, please check out the interference source;
- Manual direction error;
- Installation position of limit switch is improper or the switch is loosened;

☞ **Switch quantity output error**

Switch quantity output error can be checked out as follow:

(1) Output error

- Check whether the line is connected properly according to the aforesaid wiring diagram of output. Make sure the output common (earth wire) is connected with all the earth wires of power supply;
- Check whether the output device is broken;
- Check whether the optical coupler is broken. Use the universal meter to test whether the optical coupler is punctured, then, change the optical coupler to solve this problem;
- For safety: make sure to parallel connect the freewheeling diode when output and use the inductive load. The diode type is IN4007 or IN4001.

(2) Judgment of bad output

Cut off the outer connection on output points; instead, connect a pull-up resistor at around 10K on it to the power supply. Output earth wire should be connected to GND of power supply while the red test pen of universal meter connected to 12V positive electrode and black test pen connected to signal output end. At the same time, click the button of test menu by hand to check if there is output voltage. If there is voltage, check the outer line, otherwise, check whether the common port of card is well connected or the inner optical coupler is broken etc.

☞ Coder error

When there is error in the use of coder, follow the methods below to check out:

(1) Check the connection of coder. Make sure the connection is compliance with the differential or open-collector mode;

(2) Check the voltage of coder. The motion card usually receives +5V signal. If the coder is +12V or +24V, make sure to serial connect a 1K(+12V) resistance between A and B phase of coder and A and B phase of terminal board.

(3) Inaccurate counting of coder. Outer wiring connection of coder must use the shielded twisted pair (STP). The line should not be tied together with wire that with strong interfere such as strong electricity. They should be put separately at a distance of 30~50MM;