ADT-856

6-axis Motion Control Card User's Reference Manual

28 August 2009

Rev 2.0

Adtech (Shenzhen) CNC Technology Co., Ltd
Add：Building 36 Majialong Industrial Area Nanshan District， Shenzhen
Post Code: 518052
Tel：+86 755 2609 9116 Fax: +86 755 2609 2719
E-mail: export@adtechen.com Technical support: support@adtechen.com

Copyrights

This User Manual contains proprietary information held by Adtech (Shenzhen) CNC Technology Co., Ltd (“Adtech” hereafter); stimulation, copy, photocopy or translation into other languages to this User Manual shall be disallowed unless otherwise approved by Adtech.

Meanwhile, Adtech doesn’t provide any kind of warranty, expression on standing or implication. Adtech and its staffs are not liable for any direct/ indirect information disclosure, economic loss or progress termination caused by this User Manual and the product information inside.

All the contents in this User Manual may be changed without any notice.

Trademark

All the product names introduced in this User Manual are only for identification purpose, while they may belong to other various trademarks or copyrights, such as:

※ INTEL and PENTIUM are trademarks of INTEL Company;
※ WINDOWS and MS-DOS trademarks of MICROSOFT Company;
※ ADT-8940 is the trademark of Adtech;
※ Other trademarks belong to their corresponding registered companies.

Version Upgrading Record

Version	Revised in	Descriptions
V2.0	2009/08/28	The fourth version

Remark: The three digits in the version number respectively mean:

□ □ □

Hardware version number Major version number Minor version number

Contents

Chapter 1 General information

INTRODUCTION

ADT856 Card is a kind of high-performance 4-axis servo/ stepping control card based on PCI bus and supporting Plug & Play, while one system can support up to 16 control cards and control up to 64 lines of servo/ stepping motors.

Pulse output method may be single pulse (pulse + direction) or double pulse (pulse+pulse), with the maximum pulse frequency of 2MHz. Advanced technologies are applied to ensure the frequency tolerance is less than 0.1% despite of high output frequency.

It supports 2-4 axis of linear interpolation, with the maximum interpolation speed of 1MHz.

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.

Speed can be set as contstant speed or trapezoidal acceleration/ deceleration.

Hardware caching features with a large-capacity.

I / O response time of about 500μs.

Position management is realized through two up/ down counters, one used to manage logical positions of internally driven pulse output, and the other used to receive external input, with encoder or grating ruler inputted through A/ B phase as the input signal.

Counters are up to 32 digits, specially, the range is 2,147,483,648~+2,147,483,647.
The system also provides DOS/WINDOWS95/98/NT/2000/XP/WINCE development libraries and enable software development in VC++, VB, BC++, LabVIEW, Delphi, and C++Builder.

☞ FEATURES

- 32-bit PCI bus, PnP
- 6-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 6 axes have code 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
- 2-6 axes linear interpolation
- CW and CCW circular-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/ deceleration 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
- Up to 16 motion cards can be used in one system
- Supported operating systems: DOS,WINDOWS95/98/NT/2000/XP, WINCE

APPLICATIONS

- Multi-axis engraving system
- Robot system
- Coordinate measurement system
- PC-based CNC system

Chapter 2 Hardware installation

☞ PARTS

1. ADT-856 User Manual (this manual)
2. ADT-856 6-axis PCI bus high-performance motion control card
3. ADT-856F1
4. ADT-856 user CD
5. ADT-9137 37-pin signal connecting plate,4 pcs（Optional）
6. ADT-D37 4 pc （Optional）
7. ADT-DB37 2 pc
8. DB40 40-pin cable 1pc
9. DB26 26-pin cable 1pc

☞ INSTALLATION

(1) Switch off the computer power supply (for ATX supply case, switch off the overall power)

(2) Open the back cover of the computer case

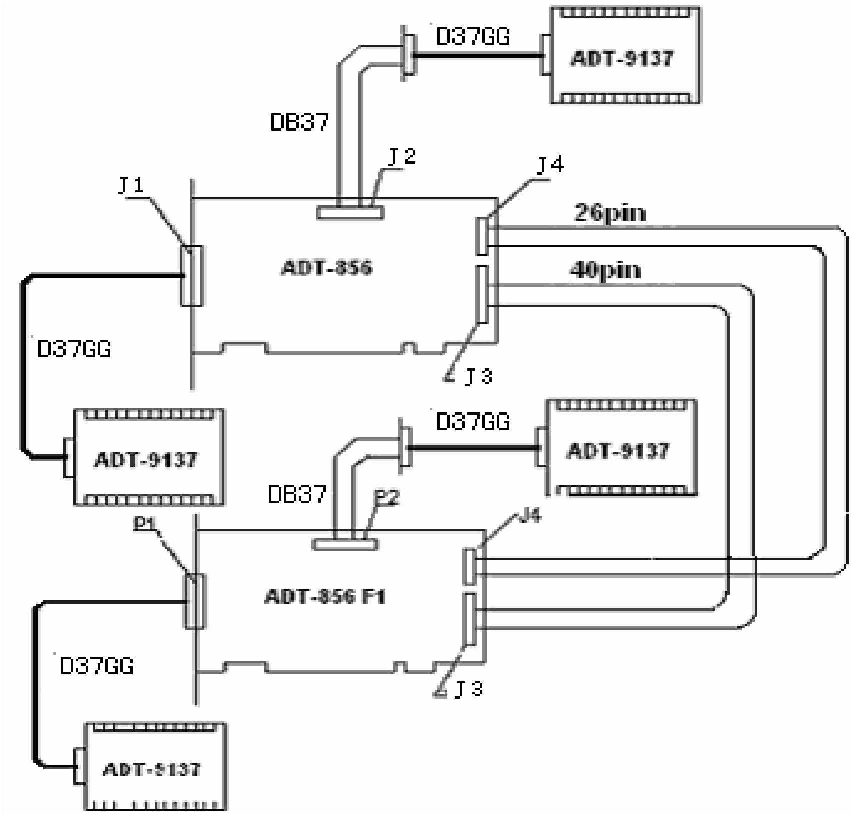
(3) Insert ADT-856 into an available PCI slot

(4) Ensure the golden finger of ADT-856 has been fully inserted the slot and then fasten card with screws

(5) Check whether it is necessary to install ADT-856F1

(6) Check whether it is necessary to install J2,P1,P2 interface cable

Chapter 3 Electrical connection



There are four input/ output interfaces inside an ADT856 card, whereby J1,J2 is for 37-pin socket , J3 is for 40-pin, and J4 is for 26-pin.

J1 is the signal cable for pulse output of X, Y, Z axis, switch amount input and switch amount output (OUT0-OUT5); **J2** is the signal cable for pulse output of A, B, C axis, switch amount input and switch amount output (OUT6-OUT11); **J3** is the signal cable for encoder input and switch amount input of X, Y, Z and A, B, C axis; **J4** is switch amount input and switch amount output (OUT12-OUT31).

There are four input/ output interfaces inside an ADT856F1 card, whereby P1,P2 is for 37-pin socket , J3 is for 40-pin, and J4 is for 26-pin. ADT856F1 is the adapter for

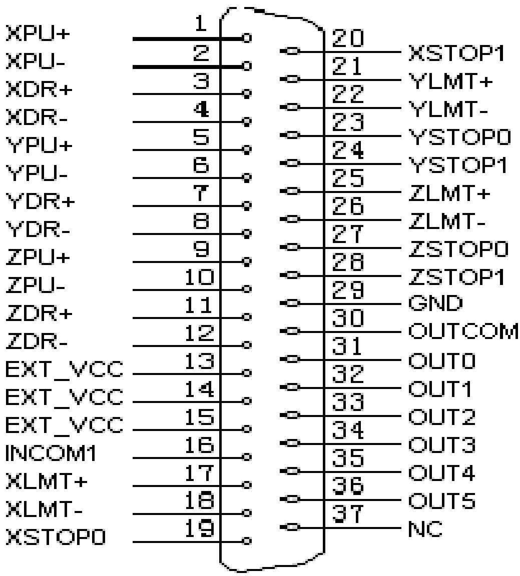
ADT856 to connect to J3, J4.

P1 is the signal cable for INPOS,ALARM,STOP2 and IN0 of X, Y, Z ,A ,B, C axis, switch amount output (OUT12-OUT19); **P2** is the signal cable for encoder input and switch amount output (OUT20-OUT31); **J3** ,**J4** is connected with the ADT-856 port, which are connect to J3,J4 on the ADT856.

REMARK: **The general-purpose input, output signals of J3 and J4 are not added optocoupler isolation, Isolation measures should be added if it is directly connected to the input and output. Otherwise, if it is burned outside the voltage does not fall within the scope of the warranty. ADT-856F1 card adapter through the post P1 and P2 of the general-purpose input and output signals have been added Optocoupler isolation.**

Signals are defined as follows:

J1 line

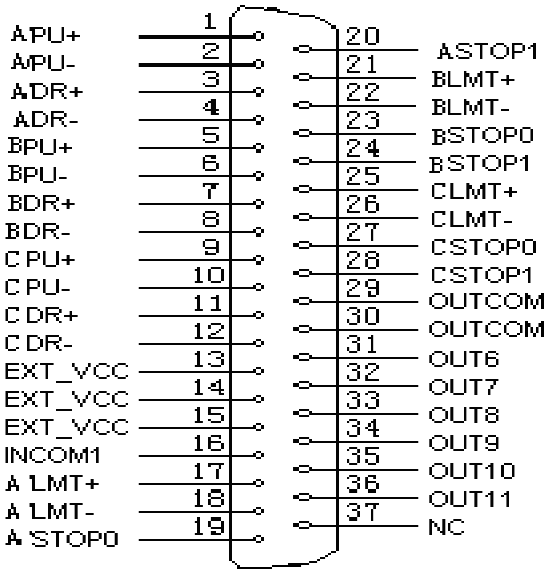


XPU+	X pulse signal +
XPU-	X pulse signal -
XDR+	X direction signal +

XDR-	X direction signal -
YPU+	Y pulse signal +
YPU-	Y pulse signal -
YDR+	Y direction signal +
YDR-	Y direction signal -
ZPU+	Z pulse signal +
ZPU-	Z pulse signal -
ZDR+	Z direction signal +
ZDR-	Z direction signal -
EXT_VCC	Positive port of internal +5V power supply; do not connect to external power supply
EXT_VCC	Positive port of internal +5V power supply; do not connect to external power supply
EXT_VCC	Positive port of internal +5V power supply; do not connect to external power supply
INCOM1	Common port of photoelectric coupling input (signals below)
XLMT+	X positive limit signal
XLMT-	X negative limit signal
XSTOP0	X home signal 0; can be used as universal input signal
XSTOP1	X home signal 1; can be used as universal input signal
YLMT+	Y positive limit signal
YLMT-	Y negative limit signal
YSTOP0	Y home signal 0; can be used as universal input signal
YSTOP1	Y home signal 1; can be used as universal input signal
ZLMT+	Z positive limit signal
ZLMT-	Z negative limit signal
ZSTOP0	Z home signal 0; can be used as universal input signal
ZSTOP1	Z home signal 1; can be used as universal input signal
GND	Internal power supply ground wire
OUTCOM	Common ground wire of output point of switching quantity

OUT0	Digital output
OUT1	
OUT2	
OUT3	
OUT4	
OUT5	
NC	untapped

J2 line

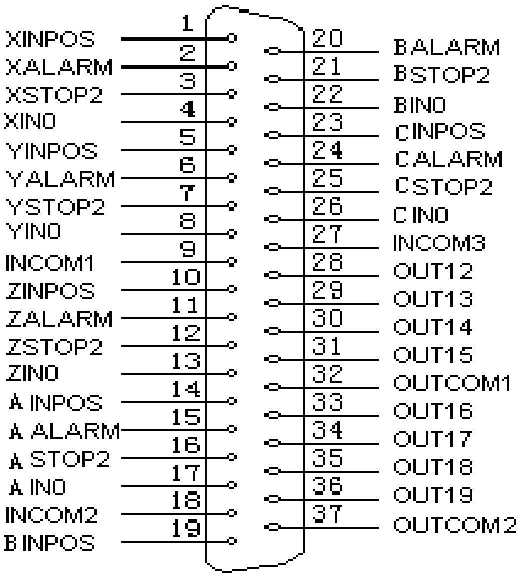


APU+	A pulse signal +
APU-	A pulse signal -
ADR+	A direction signal +
ADR-	A direction signal -
BPU+	B pulse signal +
BPU-	B pulse signal -
BDR+	B direction signal +

BDR-	B direction signal -
CPU+	C pulse signal +
CPU-	C pulse signal -
CDR+	C direction signal +
CDR-	C direction signal -
EXT_VCC	Positive port of internal +5V power supply; do not connect to external power supply
EXT_VCC	Positive port of internal +5V power supply; do not connect to external power supply
EXT_VCC	Positive port of internal +5V power supply; do not connect to external power supply
INCOM1	Common port of photoelectric coupling input (signals below)
ALMT+	A positive limit signal
ALMT-	A negative limit signal
ASTOP0	A home signal 0; can be used as universal input signal
ASTOP1	A home signal 1; can be used as universal input signal
BLMT+	B positive limit signal
BLMT-	B negative limit signal
BSTOP0	B home signal 0; can be used as universal input signal
BSTOP1	B home signal 1; can be used as universal input signal
CLMT+	C positive limit signal
CLMT-	C negative limit signal
CSTOP0	C home signal 0; can be used as universal input signal
CSTOP1	C home signal 1; can be used as universal input signal
OUTCOM	Common output
OUTCOM	
OUT6	Digital output
OUT7	
OUT8	
OUT9	

OUT10	
OUT11	
NC	untapped

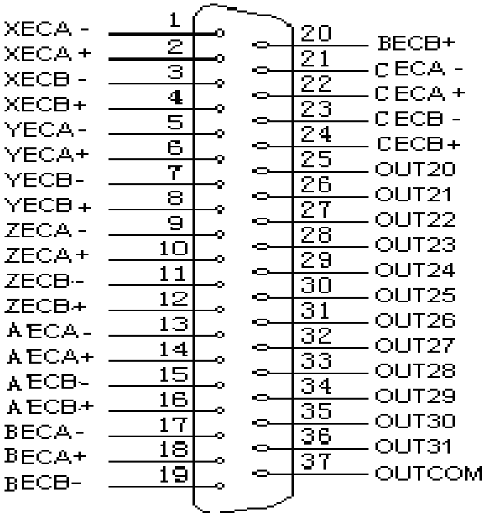
P1 line



XINPOS	X servo in-position signal; can be used as universal input signal
XALARM	X servo alarm signal;
XSTOP2	X home signal 2; can be used as universal input signal
XIN0	X input signal 0;can be used as universal input signal
YINPOS	Y servo in-position signal; can be used as universal input signal
YALARM	Y servo alarm signal;
YSTOP2	Y home signal 2; can be used as universal input signal
YIN0	Y input signal 0;can be used as universal input signal
INCOM1	Common port of photoelectric coupling input (signals above)
ZINPOS	Z servo in-position signal; can be used as universal input signal

ZALARM	Z servo alarm signal;
ZSTOP2	Z home signal 2; can be used as universal input signal
ZIN0	Z input signal 0;can be used as universal input signal
AINPOS	A servo in-position signal; can be used as universal input signal
AALARM	A servo alarm signal;
ASTOP2	A home signal 2; can be used as universal input signal
AIN0	A input signal 0;can be used as universal input signal
INCOM2	Common port of photoelectric coupling input (signals above)
BINPOS	B ervo in-position signal; can be used as universal input signal
BALARM	B servo alarm signal;
BSTOP2	B home signal 2; can be used as universal input signal
BIN0	B input signal 0;can be used as universal input signal
CINPOS	C ervo in-position signal; can be used as universal input signal
CALARM	C servo alarm signal;
CSTOP2	C home signal 2; can be used as universal input signal
CIN0	C input signal 0;can be used as universal input signal
INCOM3	Common port of photoelectric coupling input (signals above)
OUT12	Digital output
OUT13	
OUT14	
OUT15	
OUTCOM1	Common output
OUT16	Digital output
OUT17	
OUT18	
OUT19	
OUTCOM2	Common output

P2 line



XECA—	X-axis coder phase A input -
XECA+	X-axis coder phase A input +
XECB—	X-axis coder phase B input -
XECB+	X-axis coder phase B input +
YECA—	Y-axis coder phase A input -
YECA+	Y-axis coder phase A input +
YECB—	Y-axis coder phase B input -
YECB+	Y-axis coder phase B input +
ZECA—	Z-axis coder phase A input -
ZECA+	Z-axis coder phase A input +
ZECB—	Z-axis coder phase B input -
ZECB+	Z-axis coder phase B input +
AECA—	A-axis coder phase A input -
WECA+	A-axis coder phase A input +
AECB—	A-axis coder phase B input -
AECB+	A-axis coder phase B input +
BECA—	B-axis coder phase A input -

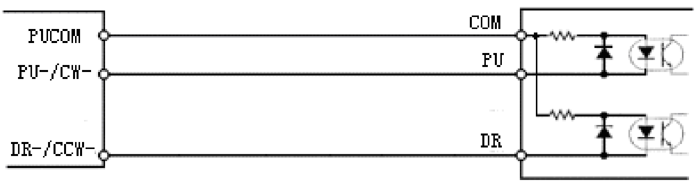
BECA+	B-axis coder phase A input +
BECB－	B-axis coder phase B input -
BECB+	B-axis coder phase B input +
CECA－	C-axis coder phase A input -
CECA+	C-axis coder phase A input +
CECB－	C-axis coder phase B input -
CECB+	C-axis coder phase B input +
OUT20	Digital output
OUT21	
OUT22	
OUT23	
OUT24	
OUT25	
OUT26	
OUT27	
OUT28	
OUT29	
OUT30	
OUT31	
OUTCOM	Digital output

☞ CONNECTION FOR PULSE/ DIRECTION INPUT SIGNAL

Pulse output is in differential output.

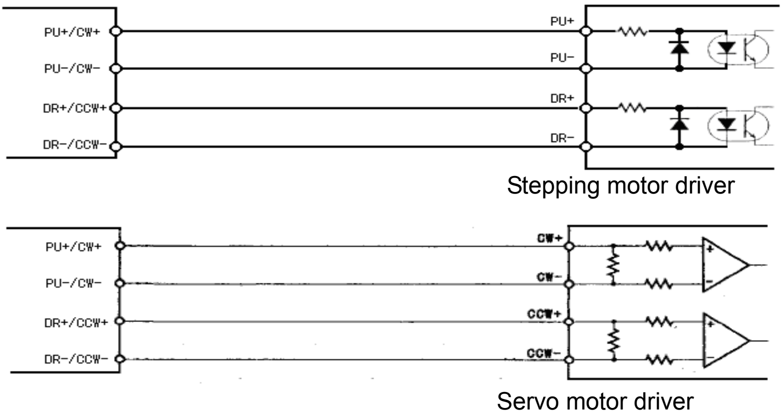
May be conveniently connected with a stepping/ servo driver

The following figure shows open-collector connection between pulse and direction.



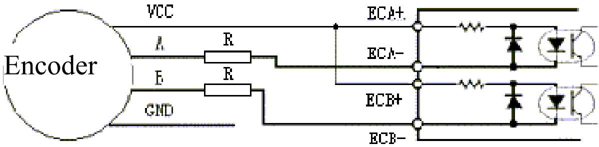
Stepping motor driver

The following figure shows differential-output connection between pulse and direction signals; this method is recommended as it is differential connection with strong resistance to disturbance.

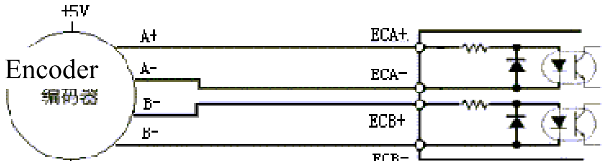


Remark: Refer to Appendix A for wiring maps of stepping motor drivers, normal servo motor driver and terminal panel.

☞ CONNECTION FOR ENCODER INPUT SIGNAL



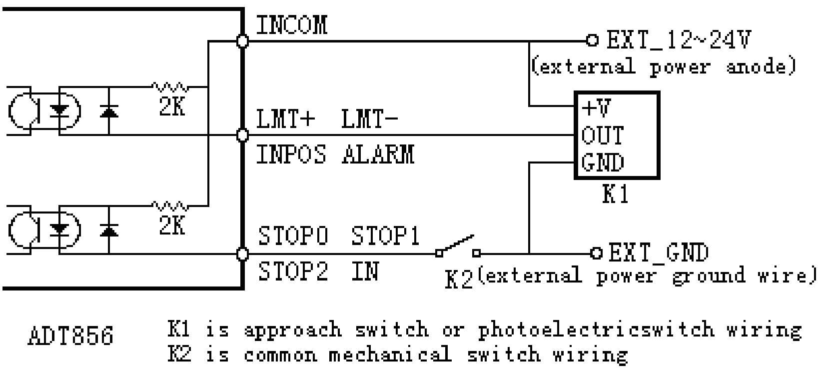
Wiring map for an open-collect output-type encoder. For +5V power supply, R is not required; for +12V power supply, R= 1KΩ; and for +24V power supply, R= 2KΩ



Wiring map for a differential-driver output-type encoder

差分驱动 (LINE DRIVER)输出型编码器接线图

☞ CONNECTION FOR DIGITAL INPUT

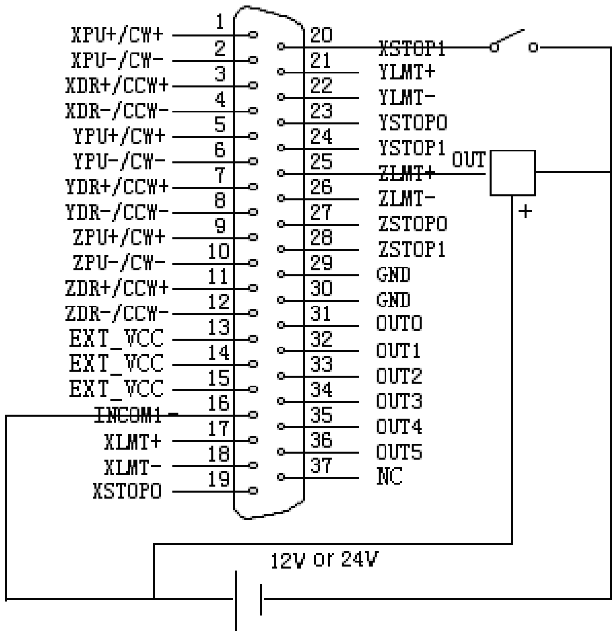


Remark:

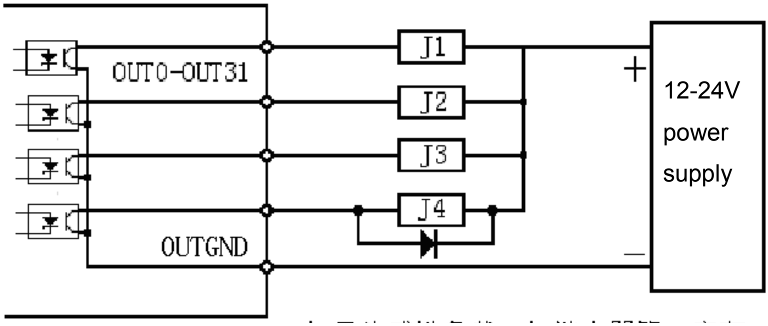
(1) To make the input signals valid, connect the "common photoelectric coupling port" of corresponding input signals (INCOM1, INCOM2, INCOM3, INCOM4, INCOMA, INCOMB) to the positive port of 12V or 24V power supply; connect one end of common switch or ground

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 FOR DIGITALOUTPUT



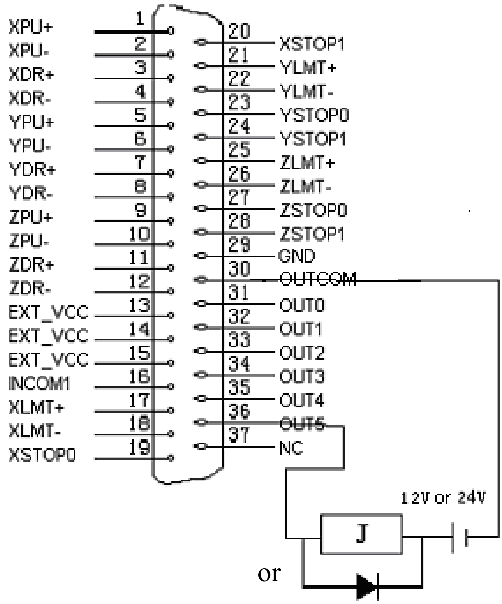
For inductive loading such as relay, add continuous dioxide at the two ends of the loading, as shown in J4

Remark:

(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 4 Software installation

ADT856 card must be used with drive installed under Win95/ Win98/ NT/ Win2000/ WinXP, but in case of DOS, no drive is required to be installed.

The following part takes Win2000 and WinXP for example, and users may refer to other operating systems.

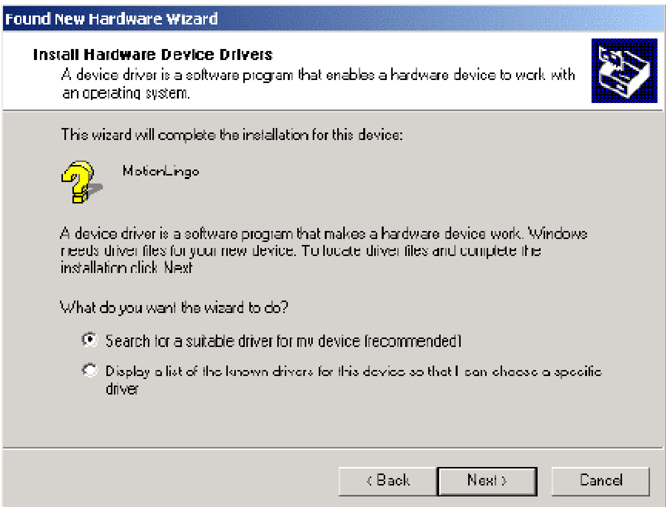
Drive for the control card is located in the Drive/ ControlCardDrive folder within the CD, and the drive file is named as ADT856.INF.

☞ DRIVE INSTALLATION IN WIN2000

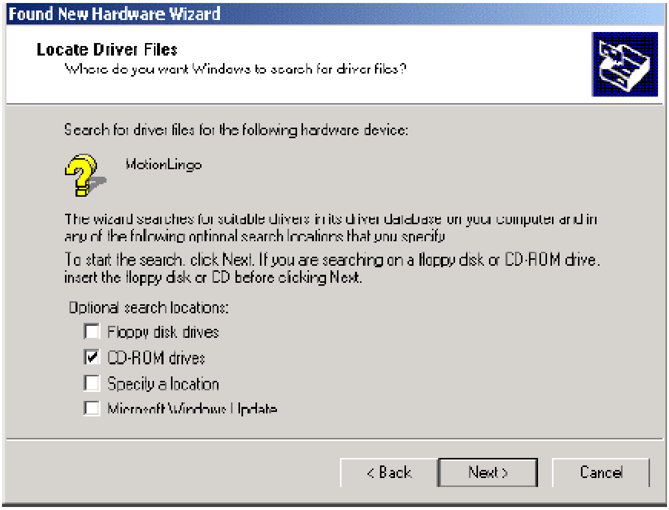
The following part takes Win2000 Professional Version as example to indicate installation of the drive; other versions of Win2000 are similar.

After attaching the ADT856 card to the PCI slot of a computer, a user shall log in as administrator to the computer; upon display of the initial interface, the computer shall notify “Found new hardware” as follows:

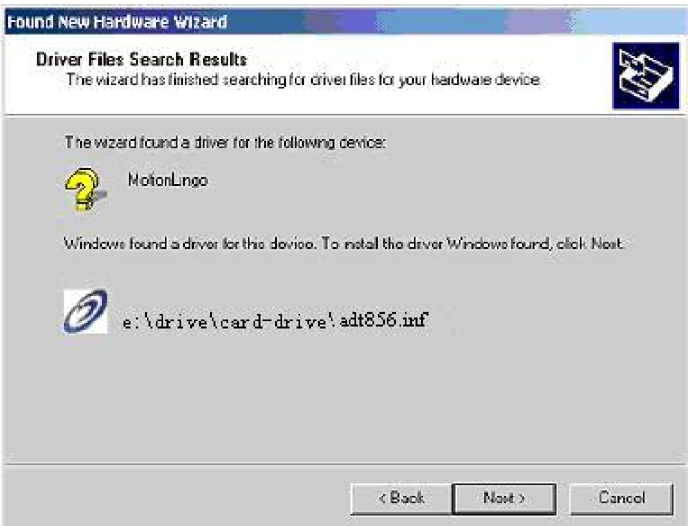
Just click “Next” to display the following picture:



Click again “Next” to display the following picture:



Then select “Specify a location” and Click again “Next” and Click “Browse” button to select DevelopmentPackage/ Drive/ CardDrive and find the ADT856.INF file, then click “OK” to display the following interface:



Click “Next” to display the following picture:

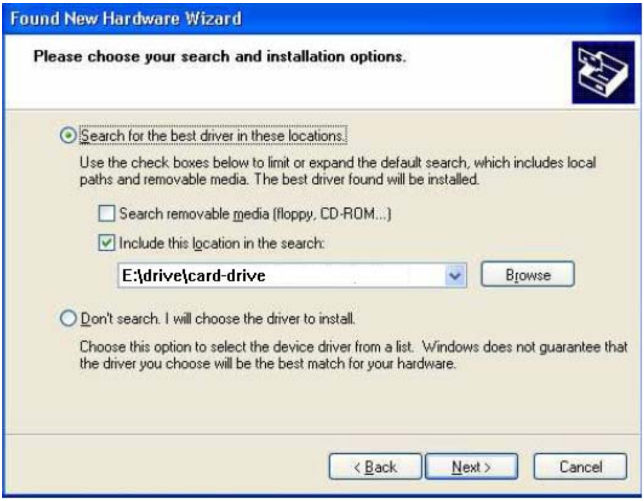
Finally click “Finish” to complete installation.

☞ DRIVE INSTALLATION UNDER WINXP

Installation under WinXP is similar to that under Win2000, specifically:


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.
< Back
Next >
Cancel


Found New Hardware Wizard
Welcome to the Found New Hardware Wizard
This wizard helps you install software for:
NetMos Unusable Parallel Port
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.
< Back
Next >
Cancel



Click "Browse" button to select Drive/ CardDrive and find the ADT856.INF file, then click "Next" to display the following interface:

Then click "Finish" to complete installation.

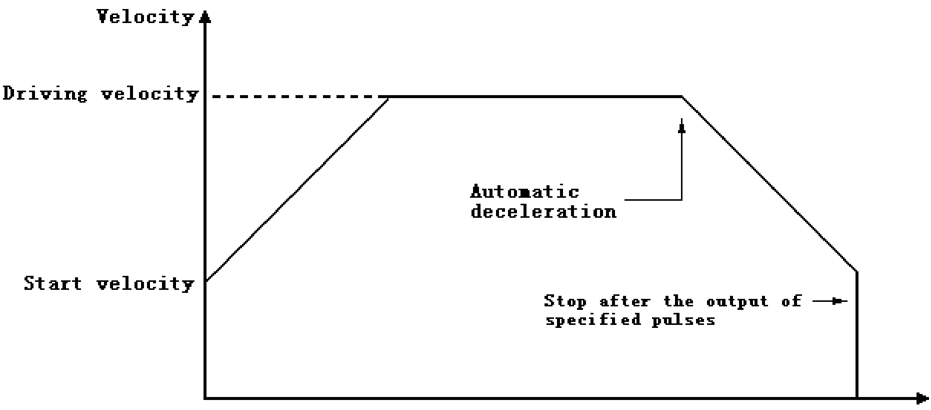
Chapter 5 Functions

☞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:

- Range R
- Acceleration/deceleration A/D
- Start velocity SV
- Driving velocity V
- Output pulse P



Quantitative driving

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.

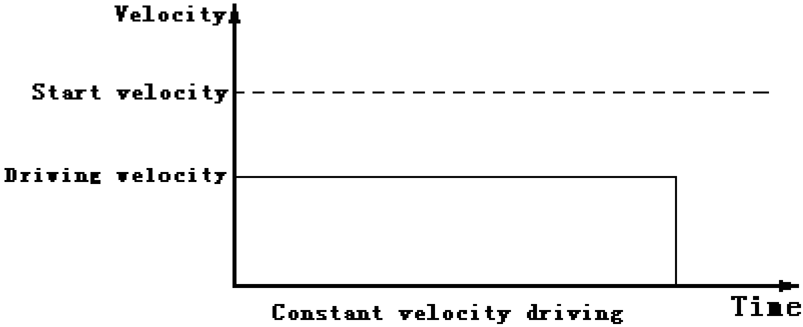
☞Velocity curve

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:

1. Range R
2. Start velocity SV
3. Driving velocity V



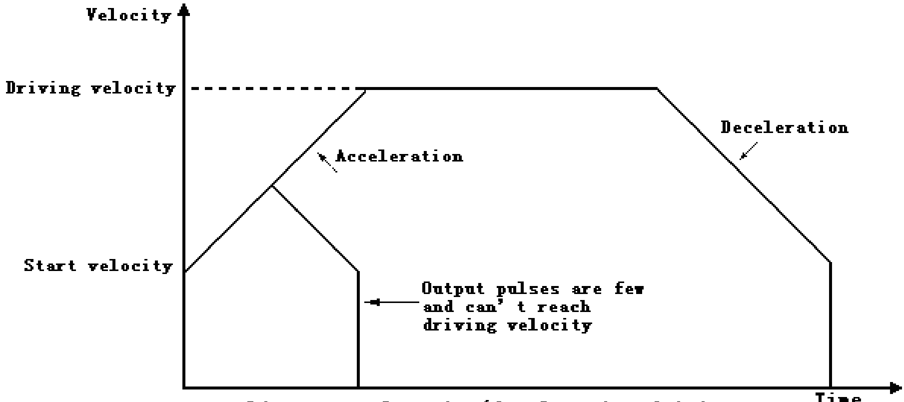
3.2 Linear acceleration/deceleration driving

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

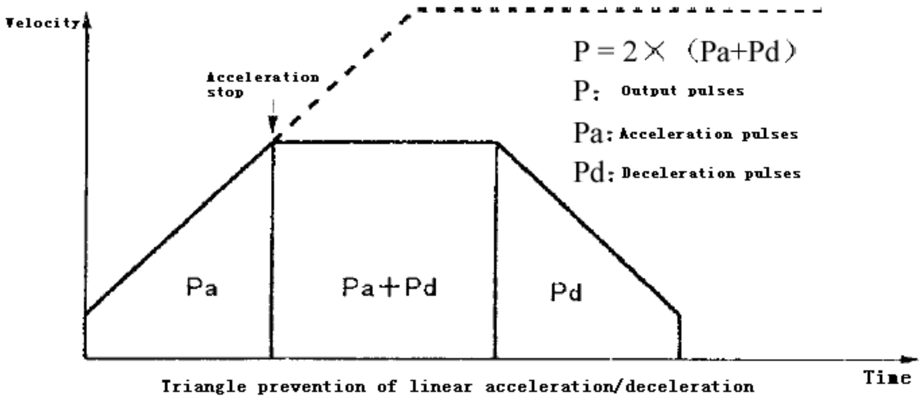


Linear acceleration/deceleration driving

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.

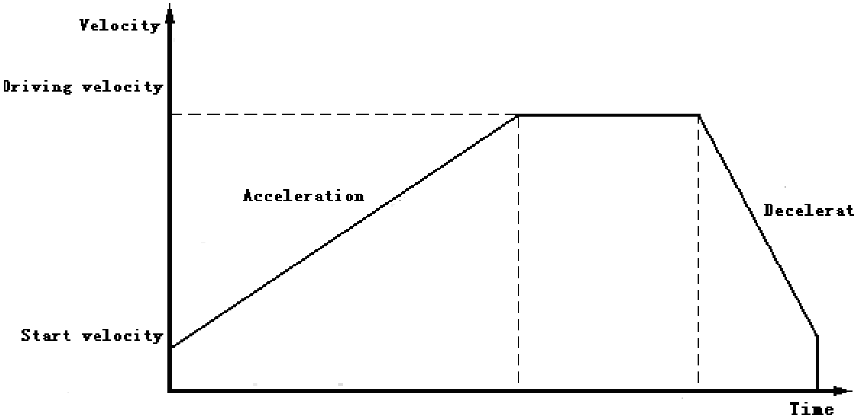


Fig. 1 Asymmetrical linear acceleration/deceleration driving (acceleration < deceleration)

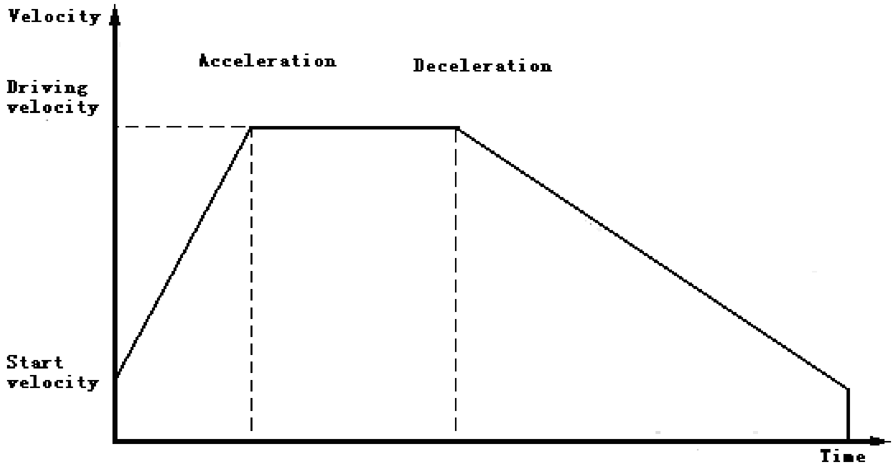


Fig. 2 Asymmetrical linear acceleration/deceleration driving (acceleration > deceleration)

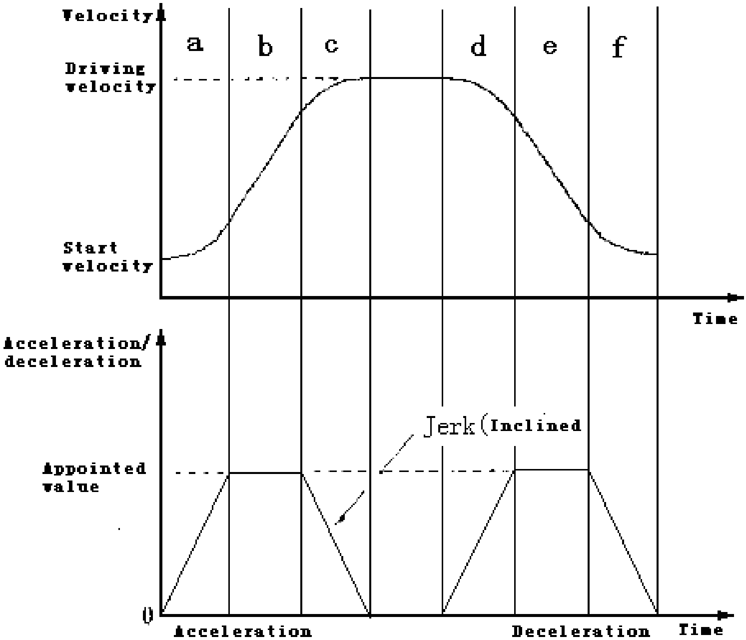
Configure the following parameters like common linear acceleration/deceleration driving:

1. Range R
2. Acceleration A
3. Deceleration D

4. Start velocity SV
5. 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:

6. Range R
7. Change rate of acceleration/deceleration K
8. Acceleration A
9. Deceleration D (if necessary)

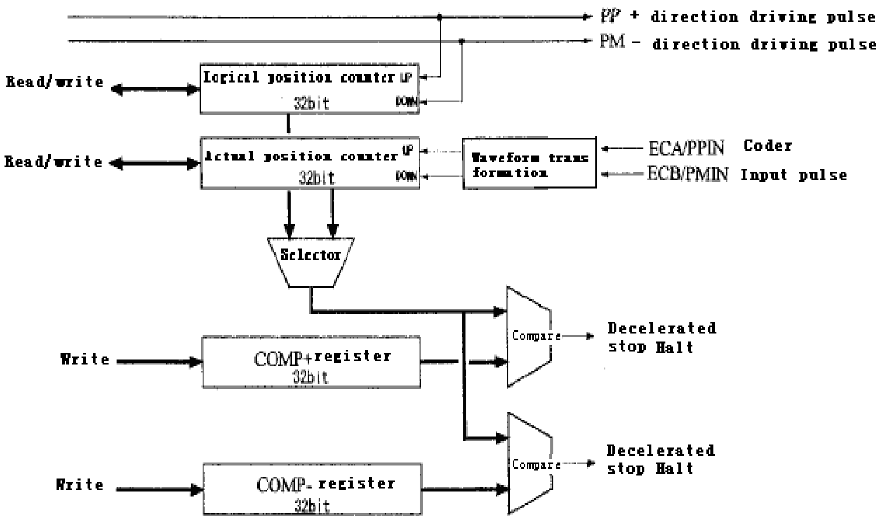
1. Start velocity SV
2. Driving velocity V

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

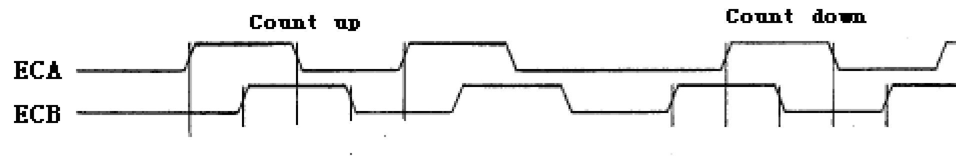
4.1 Logical position counter and actual position counter

Logical position counter is used to count the positive/negative output pulses of ADT856 card. It counts up 1 after the output of one positive pulse and counts down 1 after the output of one negative pulse.

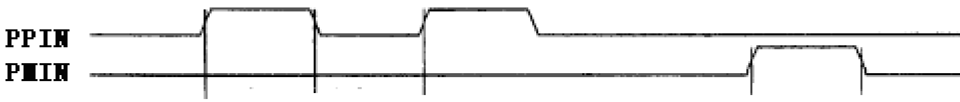
Actual position counter counts the input pulses from external coder. You can select the type of input pulse A/B phase signal or independent 2-pulse up/down counting signals. The counting direction can be customized.

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-.

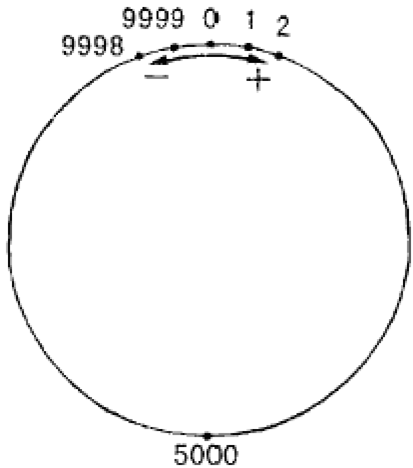
COMP+ register and COMP- register can be written at any time.

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

一、 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.

二、 Software limit is invalid if variable circle function is activated.

☞Interpolation

ADT856 card can perform linear interpolation of random 2-6 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

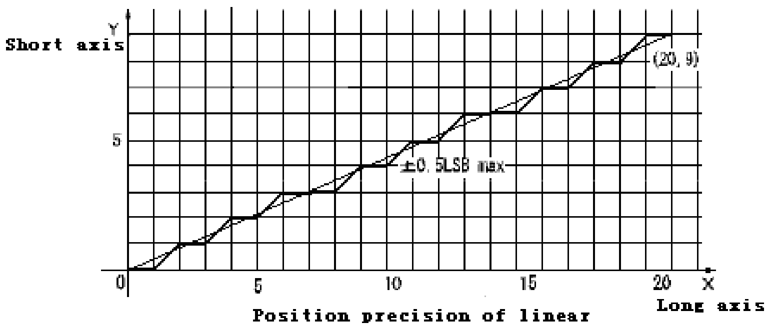
The interpolation will stop if the hardware limit or software limit in any direction (+/-) is activated in the arc interpolation process. Therefore, you must be careful when perform arc interpolation and can't leave limit area.

■ In-position signal of servo motor

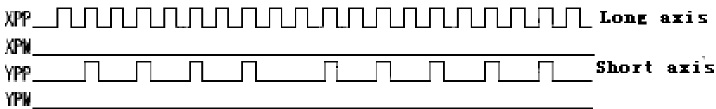
The interpolation driving will stop once the INPOS signal of each axis is valid. The INPOS signals of all axes are in effective electricity level when the interpolation driving is finished.

5.1 2-6 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

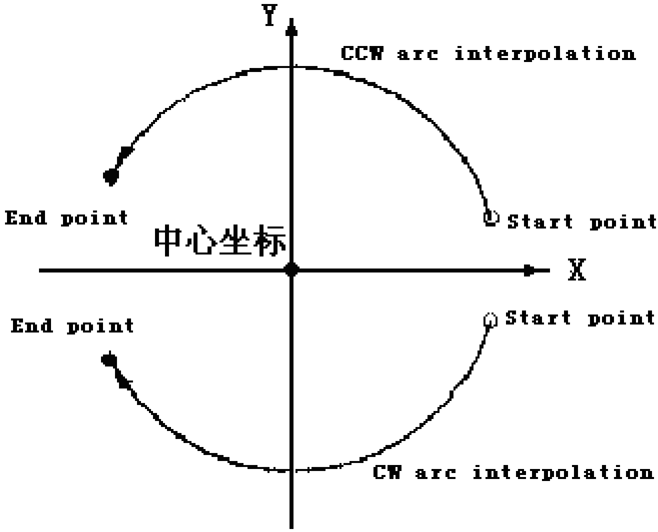


Example of end point (X: 20, Y: 9) driving pulse output

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.

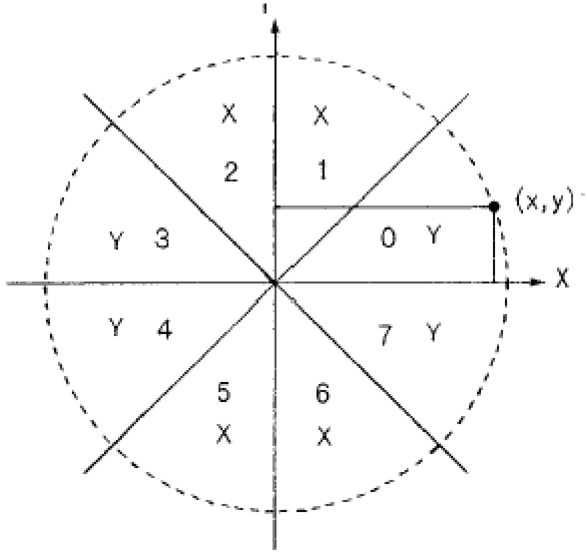
5.2 Arc interpolation

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



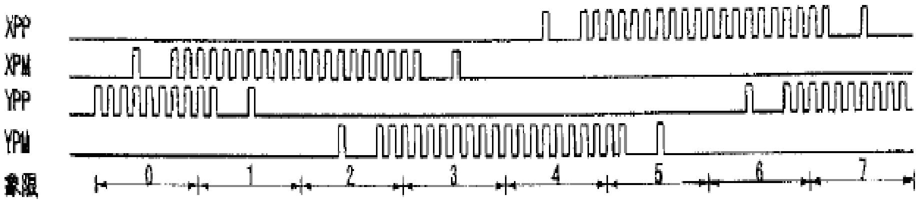
CW arc interpolation draws arc from current coordinates to end-point coordinates in clockwise direction around the center coordinates. CCW arc interpolation draws arc around the center coordinates in counterclockwise direction. It will draw a complete circle if the end point is (0, 0).

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.

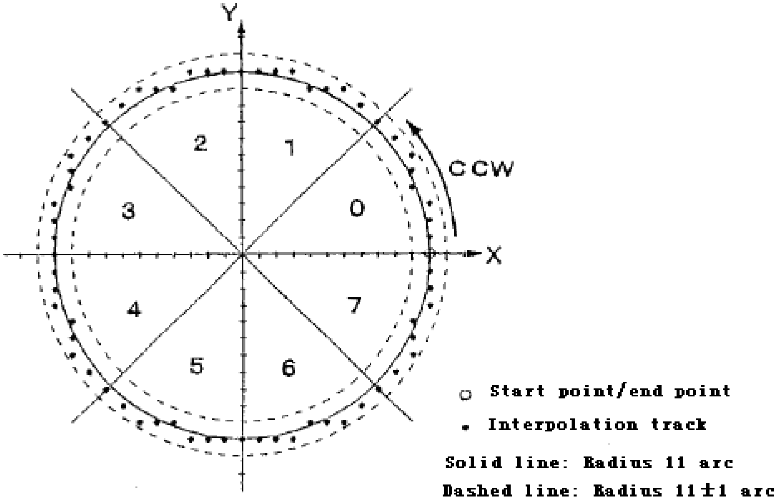


0-7 quadrants and short axis of arc interpolation arithmetic

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.

ADT856 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 ADT856 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.

Pulse/ direction are both of positive logic setting

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

Chapter 6 List of ADT856 basic library functions

List of V110 library functions

Function type	Function name	Function description	Page
Basic parameters	adt856_initial	Initialize card	47
	adt856_end	Release card	47

	set_stop0_mode	Set stop mode	47
	set_stop1_mode	Set stop mode	48
	set_stop2_mode	Set stop mode	48
	set_actualcount_mode	actual position counte	48
	set_pulse_mode	Set pulse mode	49
	set_limit_mode	Set limit mode	50
	set_ softlimit _mode1	Soft limit mode	50
	set_softlimit_mode2	Soft limit mode	51
	set_softlimit_mode3	Soft limit mode	51
	set_inpos_mode	Servo in-position	51
	set_alarm_mode	Servo alarm	52
	set_ad_mode	Add mode	52
	set_dec1_mode	Dec mode	52
	set_dec2_mode	Dec mode	52
	set_circle_mode	Circle mode	53
	set_input_filter	input filtering	53
	set_filter_time	Filtering time	53
Check for drive status	get_status	Get status of single-axis drive	54

	get_stopdata	Get stop data of error	54
	get_inp_status	Get status of interpolation	55
	get_inp_status2	Get status of continuous interpolation	55
Movement parameter setting	set_rang	Set rang	56
	Set_acac	Set acceleration changing rate	57
	set_acc	Set acceleration	57
	Set_dec	Set deceleration	58
	set_startv	Set starting speed	58
	set_speed	Set drive speed	59
	set_command_pos	Set logical position counter	59
	set_actual_pos	Set real position counter	60
	set_comp1	Set register	60
	set_comp2	Set register	60
	set_dec_pos	Set deceleration position	60
Check for motion parameters	get_command_pos	Get logical position	61
	get_actual_pos	Get real position	61
	get_speed	Get drive speed	61
	get_ad	Get acceleration	62

Drive category	pmove	Single-axis quantitative drive	62
	continue_move	Continuous driving	62
	dec_stop	Deceleration stop	63
	sudden_stop	Sudden stop	63
	Inp_move2	2-axis linear interpolation	63
	inp_cw_arc	Clockwise arc interpolation	64
	inp_ccw_arc	Counterclockwise arc interpolation	64
	inp_move3	3-axis linear interpolation	64
	inp_move4	4-axis linear interpolation	65
	inp_move6	6-axis linear interpolation	65
	inp_dec_enable	Enable deceleration of interpolation	66
	inp_dec_disable	Disable deceleration of interpolation	66
	inp_clear	clean the error of interpolation	66
Switch amount category	read_bit	Read single input point	66
	write_bit	Output single output point	66

Chapter 7 Details of ADT856 basic library functions

☞ CATEGORY OF BASIC PARAMETER SETTING

1.1 Initialize card

int adt856_initial(void);

Function

Initialize motion card

(1) Return >0 means amount of installed ADT856 cards; in case the Return is 3, the available card numbers shall be 0, 1, and 2;

(2) Return =0 means no installation of ADT856 card;

(3) Return <0 means no installation of service if the value is -1 or PCI bus failure is the value is -2.

Remark: Initialization functions are preliminary conditions to call other functions, thus must be called firstly so as to verify available cards and initialize some parameters.

1.2 Release ADT-856 card

int adt856_end(void);

Function

Release the resources of motion card

Return value 0:correct 1:wrong

1.3 Set the mode of stop0 signal

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

Function

Release the resources of motion card

Parameter

cardno	card number	
axis	axis number(1-6)	
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

1.4 Set the mode of stop1 signal

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

Function

Set COMP + register as software limit

Parameter

cardno	card number	
axis	axis number(1-6)	
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

1.5 Set the mode of stop2 signal

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

Function

Set COMP + register as software limit

Parameter

cardno	card number	
axis	axis number(1-6)	
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.

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

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

Parameter

cardno card number

axis axis number(1-6)

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

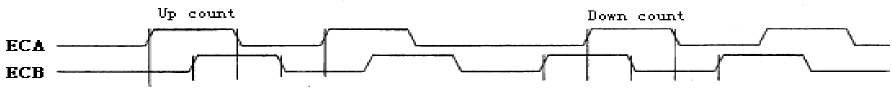
freq Frequency multiplication of A/B pulse input， Invalid for Up/Down pulse input

0: 4× 1: 2× 2: 1×

Return value 0:correct 1:wrong

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

■ 2-phase pulse input



■ Up/Down (PPIN/PMIN) pulse input

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

1.7 Set output pulse mode

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

Function

Set pulse mode

Parameter

cardno Card number

axis Axis number (1-6)

value 0： Pulse + Pulse method 1： Pulse + direction method

Pulse/ direction are both of positive logic setting

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

logic 0: Positive logic pulse 1: Negative logic pulse

Positive logic pulse  Negative logic pulse 

dir-logic 0: Positive logic direction input signal 1: Negative logic direction input signal

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

Return 0: Correct 1: Wrong

Default mode: Pulse + direction, with positive logic pulse and positive logic direction input signal

1.8 Set mode of nLMT signal input along positive/ negative direction

int set_limit_mode(int cardno, int axis, int v1,int v2,int logic);

Function

Set mode of nLMT signal input along positive/ negative direction

Parameter

cardno Card number

axis Axis number (1-6)

v1 0: positive limit is effective 1: positive limit is ineffective

v2 0: negative limit is effective 1: negative limit is ineffective

logic 0: low level is effective 1: high level is effective

Return value 0: Correct 1: Wrong

Default mode: positive and negative limits with low level are effective

- **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(1-6)

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

1.10 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(1-6)	
value	0:invalid	1:valid

Return value 0:correct 1:wrong

Default modes invalid

Notice

The same as above

1.11 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(1-6)	
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.

1.12 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(1-6)	
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.

1.13 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(1-6)	
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

1.14 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(1-6)	
value	0:linear A/D	1:S-curve A/D

Return value 0:correct 1:wrong

Default modes linear A/D

1.15 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(1-6)
value	deceleration mode (0: symmetric deceleration, 1: asymmetric deceleration)

Return value 0:correct 1:wrong

Default modes symmetric acceleration/deceleration

1.16 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(1-6)

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

1.17 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(1-6)

value 0:invalid 1:valid

Return value 0:correct 1:wrong

Default modes invalid

1.18 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

1.19 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

☞ CATEGORY OF DRIVE STATUS CHECK

2.1 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(1-6)

value Index of driving status; the meanings are:

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.

2.2 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(1-6)
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

2.3 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.

2.4 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.

☞ CATEGORY OF MOVEMENT PARAMETER SETTING

Remark: The following parameters are not determined after initialization thus must be set before use.

3.1 Range setting

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

Function

set range

Parameter

cardno	card number
axis	axis number(1-6)
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.

3.2 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(1-6)

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

3.3 Acceleration setting

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

Function

Set the value of acceleration

Parameter

cardno card number

axis axis number(1-6)

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

3.4 Deceleration setting

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

Function

Set the value of deceleration

Parameter

cardno card number
axis axis number(1-6)
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

3.5 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(1-6)
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.

3.6 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(1-6)
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

3.7 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(1-6)
value range value(-2147483648~+2147483647)

Return value 0:correct 1:wrong

Notice

You can access the logical position counter in real time

3.8 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(1-6)
value	range value(-2147483648~+2147483647)

Return value 0:correct 1:wrong

Notice

You can access the actual position counter in real time

3.9 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(1-6)
value	range value(-2147483648~+2147483647)

Return value 0:correct 1:wrong

Notice

You can access the COMP+ register in real time

3.10 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(1-6)
value	range value(-2147483648~+2147483647)

Return value 0:correct 1:wrong

Notice

You can access the COMP- register in real time

3.11 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(1-6)

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

☞ CATEGORY OF MOTION PARAMETER CHECK

The following functions can be called at any time

4.1 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(1-6)

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

4.2 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(1-6)

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.

4.3 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(1-6)
speed Index of current driving speed

Return value 0:correct 1:wrong

Notice

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

4.4 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(1-6)
ad Index of current acceleration

Return value 0:correct 1:wrong

☞ CATEGORY OF DRIVE

5.1 Quantitative driving

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

Function

Single axis quantitative driving

Parameter

cardno card number
axis axis number(1-6)
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.

5.2 Continuous driving

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

Function

Single axis continuous driving

Parameter

cardno card number
axis axis number(1-6)
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.

5.3 Driving decelerated stop

int dec_stop(int cardno,int axis);

Function

Stop current driving process in deceleration

Parameter

cardno card number

axis axis number(1-6)

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.

5.4 Driving Sudden stop

int dec_stop(int cardno,int axis);

Function

Stop current driving process in sudden

Parameter

cardno card number

axis axis number(1-6)

Return value 0:correct 1:wrong

Notice

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

5.5 Two axes linear interpolation

int inp_move2(int cardno,int no,long pulse1,long pulse2);

Function

Two axes linear interpolation

Parameter

cardno card number

no: 1:X-Y 2:Z-A

pulse1 Moving distance of axis1

pulse2 Moving distance of axis2

Return value 0:correct 1:wrong

Notice

The X-Y interpolation takes the speed of X-axis as base,Y-axis do not need to set.

The Z-W interpolation takes the speed of Z-axis as base,A-axis do not need to set.

5.6 CW arc interpolation

int inp_cw_arc(int cardno,int no,long x,long y,long i,long j);

Function

Two axes CW arc interpolation

Parameter

cardno card number

no 1:X-Y 2:Z-W

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 X-Y interpolation takes the speed of X-axis as base,Y-axis do not need to set.

The Z-W interpolation takes the speed of Z-axis as base,A-axis do not need to set.

5.7 CCW arc interpolation

int inp_ccw_arc(int cardno,int no,long x,long y,long i,long j);

Function

Two axes CCW arc interpolation

Parameter

cardno card number

no 1:X-Y 2:Z-W

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 X-Y interpolation takes the speed of X-axis as base,Y-axis do not need to set.

The Z-W interpolation takes the speed of Z-axis as base,A-axis do not need to set

5.8 Three axes linear interpolation

int inp_move3(int cardno,long pulse1,long pulse2,long pulse3);

Function

Three axes linear interpolation

Parameter

cardno card number

pulse1 Moving distance of axis1
pulse2 Moving distance of axis2
pulse3 Moving distance of axis3

Return value 0:correct 1:wrong

Notice

The interpolation speed takes the speed of the axis X as the standard (axis1).the magnification and running speed of Z-axis must be the same as X-axis.The start speed of Z-axis must be the same as running speed of X-axis (not start speed),Y-axis do not need to set.

5.9 Four axes linear interpolation

int inp_move4(int cardno,long pulse1,long pulse2,long pulse3,long pulse4,long pulse5,long pulse6);

Function

four axes linear interpolation

Parameter

cardno card number
pulse1, pulse2, pulse3, pulse 4 , pulse5, pulse6 Moving distance of axis X, Y, Z, A,B,C

Return value 0:correct 1:wrong

Notice

The interpolation speed takes the speed of the axis X as the standard (axis1).the magnification and running speed of Z-axis must be the same as X-axis.The start speed of Z-Axis must be the same as running speed of X-axis (not start speed),Y-axis、A-axis ,C-axis do not need to set.

5.10 Six axes linear interpolation

int inp_move6(int cardno,long pulse1,long pulse2,long pulse3,long pulse4);

Function

four axes linear interpolation

Parameter

cardno card number
pulse1, pulse2, pulse3, pulse4 Moving distance of axis X, Y, Z, A

Return value 0:correct 1:wrong

Notice

The interpolation speed takes the speed of the axis X as the standard (axis1).the magnification and running speed of Z-axis must be the same as X-axis.The start speed of Z-Axis must be the same as running speed of X-axis (not start speed),Y-axis、A-axis do not need to set.

5.11 Enable interpolation deceleration

int inp_dec_enable(int cardno);

Function

Enable deceleration of interpolation

Parameter

cardno card number

Return value 0:correct 1:wrong

5.12 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.

5.13 Clean error of interpolation deceleration

int inp_clean(int cardno);

Function

Clean the error of deceleration of interpolation

Parameter

cardno card number

Return value 0:correct 1:wrong

☞ CATEGORY OF SWITCH AMOUNT INPUT/ OUTPUT

6.1 Read single input point

int read_bit(int cardno,int number)

Function

Get the status of single input bit

Parameter

cardno Card number

number Input point (0-47)

Return 0: low level 1: high level -1: error

6.2 Output single output point

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

Function

Corresponding port performs output operation.

Parameter

cardno	Card number
number	Output point (0-31)
value	0: low 1: high

Return

0: correct

1: wrong

A number corresponding to the output number

Chapter 8 Guide to motion control function library

➢ Introduction on ADT856 function library

ADT856 function library is actually the interface for users to operate the movement control card; users can control the movement control card to execute corresponding functions simply by calling interface functions.

The movement control card provides movement function library under DOS and dynamic link library under Windows; the following part will introduce the library calling method under DOS and Windows.

➢ Calling dynamic link library under Windows

The dynamic link library ADT856.dll under Windows is programmed in VC, applicable for general programming tools under Windows, including VB, VC, C++Builder, VB.NET, VC.NET, Delphi and group software LabVIEW.

a) Calling under VC

2. Create a new project;
3. Copy the ADT856.lib and ADT856.h files from DevelopmentPackage/VC in the CD to the routing of the newly created item;
4. Under File View of the Work Area of the new item, right click mouse to select “Add Files to Project” and then in the pop-up file dialogue select the file type to be “Library Files(.lib)”, then search out “ADT856.lib” and select it, finally click “OK” to finish loading of

the static library;

5. Add #include “ADT856.h“ in the declaim part of the source file, header or overall header “StdAfx.h”.

After the above four steps, users can call functions in the dynamic link library.

Remark: The calling method under VC.NET is similar.

b) Calling under VB

1. Create a new project;
2. Copy the ADT856.h file from DevelopmentPackage/VB in the CD to the routing of the newly created item;
3. Select the menu command Engineering/Add module and subsequently Save Current in the dialogue to search out the ADT856.bas module file, finally click the Open button.

After the above three steps, users can call functions in the dynamic link library.

Remark: The calling method under VB.NET is similar.

c) Calling under C++Builder

(1) Create a new project;

(2) Copy the ADT856.lib and ADT856.h files from DevelopmentPackage/ C++Builder in the CD to the routing of the newly created item;

(3) Select the menu command “Project\Add to Project”, and in the pop-up dialogue select the file type to be “Library files(*.lib)”, then search out the “ADT856.lib” file and click Open button;

(4) Add #include “ADT856.h“ in the declaim part of the program file.

After the above four steps, users can call functions in the dynamic link library.

d) Calling under LabView 8

(1) Create a new VI;

(2) Copy the ADT856.lib and ADT856.dll files from DevelopmentPackage/ LabVIEW in the CD to the routing of the newly created item;

(3) Right click mouse in the blank area of the program interface to display the Function Palette, select “Select a VI..” and subsequently in the pop-up window select the ADT856.llb file, finally select the required library function in the “Select the VI to Open” window and drag into the program interface.

After the above three steps, users can call functions in the dynamic link library.

Calling library functions under DOS

Function libraries under DOS are edited in Borland C3.1 and saved in the DevelopmentPackage/C++ (or C) folder. Library functions may be categorized into large and huge modes, applicable for standard C and Borland C3.1or above versions.

The method of calling function library with Borland C is as follows:

2 Under the development environment of Borland C, select the "Project\Open Project" command to create a new project;

3 Copy the ADT856H.LIB or ADT856L.LIB file and ADT856.H file from DevelopmentPackage/ C (or C++) in the CD to the path of the newly created project;

4 Select the "Project\Add Item" command and further in the dialogue select "ADT856H.LIB" or "ADT856L.LIB", finally click the Add button;

5 Add #include "ADT856.h statement in the user program file.

After the above four steps, users can call functions in the dynamic link library.

Returns of library functions and their meanings

To ensure users will correctly know execution of library functions, each library function in the function library after completion of execution will return to execution results of the library functions. Users, based on such Returns, can conveniently judge whether function calling has succeeded.

Except "int ADT856_initial(void)" and "int read_bit(int cardno, int number)" with special Returns, other functions have only "0" and "1" as the Returns, where "0" means successful calling and "1" means failed calling.

The following list introduces meanings of function Returns.

Function name	Return	Meaning
ADT856_initial	-1	No installation of service
	-2	PCI slot failure
	0	No installation of control card
	>0	Amount of control card
Read_bit	0	Low level

	1	High level
	-1	Card number or input point out of limit
Other functions	0	Correct
	1	Wrong

Remark: Return 1 means calling error, and the normal cause is wrong cardno (Card Number) or axis (Axis Number) passed during the process of calling library functions. Card number have their values starting as 0, 1, and 2, thus in case there is only one card, the card number must be 0; similarly values of axis number can only be 1, 2, 3 and 4, other values are all wrong.

Chapter 9 Briefing on motion control development

This card will encounter some problems during programming, but most problems are due to failure in understanding the methods of this control card. The following part will give explanation on some unusual and easy-to-misunderstand scenarios.

☞ CARD INITIALIZATION

Invoke the adt856_initial() function first, and make sure that the ADT856 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.

Remark: Library function ADT856_initial is the door to ADT856 card, thus

calling other functions are of sense only after successful card initialization with calling to 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 10 Programming samples in motion control development

All movement control functions return immedietly; once a drive command is made, the movement process will be controlled by the movement control card until completion; then the host computer software of users can real-time monitor the whole movement process or force to stop the process.

Remark: Axis during motion are not allowed to send new drive commands to motion axis, otherwise the previous drive will be given up so as to execute the new drive.

Although programming languages vary in types, they can still be concluded as Three Structures and One Spirit. Three Structures refer to sequential structure, cycling structure and branch structure emphasized by all the programming languages, and One Spirit refer to calculation and module division involved in order to complete design assignments, which is also the key and hard point in whole programming design.

To ensure a program is popular, standard, expandable and easy for maintenance, all the later samples will be divided into the following modules in terms of project design: movement control module (to further seal library functions provided by the control card), function realization module (to cooperate code phase of specific techniques), monitoring module and stop processing module.

Now let's brief application of ADT856 card function library in VB and VC; users using other programming languages may take reference.

☞ VB PROGRAMMING SAMPLES

1.1 PREPARATION

(1) Create a new item and save as "test.vbp";

(2) Add the ADT856.bas module in the item following the above-introduced method;

1. Movement control module

(1) Add a new module in the project and save as "ctrlcard.bas";

(2) At first, within the motion control module self-define initialization functions of the motion control card and initialize library functions to be sealed into initialization functions;

(3) Further self-define relevant motion control functions such as speed setting function, single-axis motion function, and iinterpolation function;

(4) Source code of ctrcard.bas is:

```
'/********************** Motion control module *******************
'      For developing an application system of great generality,
'    extensibility and convenientmaintenance easily and swiftly,
'    we envelop all the library functions by category basing on
'    the card function library
'*************************************************************/
Public Result As Integer          'return
Const MAXAXIS = 6                    'axis number
'*****************initial motion-card***********************
'      this function is boot of using motion-card
'      Return<=0 fail to initial motion-card，
'      Return>0    Succeed in initial motion-card
'*****************************************************
Public Function Init_Card() As Integer
```

```
        Result = adt856_initial
        If Result <= 0 Then
            Init_Card = Result
            Exit Function
        End If
        For i = 1 To MAXAXIS
            set_range 0, i, CLng(8000000 / 5)       'set range,set ratio as 5
            set_command_pos 0, i, 0                    'set logic pos as 0
            set_actual_pos 0, i, 0                  'set real pos as 0
            set_startv 0, i, 1000                   'set start-speed
            set_speed 0, i, 1000                      'set motion-speed
            set_acc 0, i, 625                        'set acceleration
        Next i
    Init_Card = Result
    End Function
    '************************** release of ADT8948 source occupied**********************
    Public Function End_Board() As Integer
          Result = adt856_end
          End_Board = Result
    End Function
    '
    '***********************set stop0 mode*********************
    'Set mode of stop0 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_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 mode********************
    '    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 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 real position counter**********************
'cardno         Card number
'axis           Axis number（1-6）
'value          Input of pulse pattern
'0：            A/B pulse input   1：up/down（PPIN/PMIN）pulse input
'dir            Counter direction
'0：            A is over B or PPIN impulse input is up counted.
'               B is over A or PMIN impulse is down counted
'1：            B is over A or PPIN impulse input is up counted
'               A is over B or PMIN impulse is down counted.
'freq           During double frequency of A/B input up/down impulse input is
non-effective
'0：4-time frequency        1：2-time frequency           2：No-time frequency
'Returning value: 0: Correct      1: False
'Initialized status: During A/B phase impulse input direction is of 0 and 4-time
frequency.
'*********************************************************************
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 pulse output mode************************
'      set the mode of pulse output
'       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
'      and positive logic direction input 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+counter as soft limit****************
'cardno          card number
'axis            axis number（1-6）
'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-counter as soft limit*****************
'
'cardno        card number
'axis          axis number（1-6）
'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+/-counter****************
'cardno        card number
'axis          axis number（1-6）
'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_LimitMode3(ByVal axis As Integer, ByVal value As Integer) As Integer
      Result = set_softlimit_mode3(0, axis, value)
```

```
    Setsoft_LimitMode3 = Result
End Function

'*******************set nINPOS(in-position input sigal)*****************
'cardno        card number
'axis          axis number（1-6）
'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 :   noneffective, 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

'*******************set nALARM(alarm input sigal)*****************
'cardno        card number
'axis          axis number（1-6）
'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 :   noneffective, 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 speed*********************
'    according as para,judge whether is constant-speed
'    set range to set ratio
'    set start-speed ,motion-speed and acceleration
'para:       axis:     axis number
'startv:               start -speed
```

```
'speed:                 motion -speed
'add:                   acceleration
'dec:                   decelerate
'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
            set_range 0, axis, 8000000 / ratio
            Result = set_startv(0, axis, startv / ratio)
            set_speed 0, axis, startv / ratio
      Else
            Select Case Mode
            Case 0
                  set_dec1_mode 0, axis, 0
                  set_dec2_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_ad_mode 0, axis, 0
             Case 1
                   set_dec1_mode 0, axis, 1
                   set_dec2_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
                  set_ad_mode 0, axis, 0
              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
                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
                Setup_Speed = Result
            End Select
        End If
End Function

'***************************single-axis motion***************************
'       drive one axis motion
'       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***************************
'       drive one axis continuous motion
'       para：  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

'/*********************2-axis interpolation*********************
'   for XY or ZW 2-axis linear interpolation
'       no ->     1: X-Y          2:Z-W
'           Return=0 correct，Return=1 wrong
'*********************************************************/
```

```
Public Function Interp_Move2(ByVal no As Integer, ByVal pulse1 As Long, ByVal pulse2 As Long) As Integer
      Result = inp_move2(0, no, pulse1, pulse2)
      Interp_Move2 = Result
End Function

'/********************3-axis interpolation*********************
'   for XYZ 3-axis linear interpolation
'
'      Return=0 correct，Return=1 wrong
'*********************************************************/
Public Function Interp_Move3(ByVal pulse1 As Long, ByVal pulse2 As Long, ByVal pulse3 As Long) As Integer

      Result = inp_move3(0, pulse1, pulse2, pulse3)

      Interp_Move3 = Result

End Function

'/********************4-axis interpolation*********************
'   for XYZW 4-axis linear interpolation
'      Return=0 correct，Return=1 wrong
'*********************************************************/
Public Function Interp_Move4(ByVal pulse1 As Long, ByVal pulse2 As Long, ByVal pulse3 As Long, ByVal pulse4 As Long) As Integer
      Result = inp_move4(0, pulse1, pulse2, pulse3, pulse4)
      Interp_Move4 = Result
End Function

'/********************6-axis interpolation*********************
'   for XYZWUV 6-axis linear interpolation
'      Return=0 correct，Return=1 wrong
'*********************************************************/
Public Function Interp_Move6(ByVal pulse1 As Long, ByVal pulse2 As Long, ByVal pulse3 As Long, ByVal pulse4 As Long, ByVal pulse5 As Long, ByVal pulse6 As Long) As Integer
      Result = inp_move6(0, pulse1, pulse2, pulse3, pulse4, pulse5, pulse6)
      Interp_Move6 = Result
End Function
```

```
'********************Clockwise CW circular interpolation*******************
'axis1,axis2      1：X     2:Y     3：Z    4:W
' x,y            Terminal position of Arc SR (corresponding to starting point)
' i,j            Central point position of SR circle arc (corresponding to starting point)
'return          0：Correct        1：False
'*****************************************************************

Public Function Interp_Arc(ByVal no 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, no, x, y, i, j)
End Function

'*********************Clock against CCW circular interpolation*********************
'axis1,axis2      1：X      2：Y      3：Z     4：W
'x,y                 Terminal position of Arc SR (corresponding to starting point)
'i,j                 Central point position of SR circle arc (Corresponding to starting point)
'return          0：Correct        1：False
'*****************************************************************
Public Function Interp_CcwArc(ByVal no 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, no, x, y, i, j)
End Function

'********************position counter variable ring***************

'axis        axis number（1-6）
'Value       0: ineffective           1: effective
'return      0：Correct        1：False
'Default mode :   noneffective

'*****************************************************************
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 signal wave filtering function********************
'     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(1-6)
'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

'/****************stop motion*******************************
'   stop motion in the way of sudden or decelerate
'      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                        'sudden stop
            Result = sudden_stop(0, axis)
      Else                                    'dec stop
            Result = dec_stop(0, axis)
      End If
End Function
```

```
'/************************set position counter******************************
'   set logic-pos or   real-pos
'      para：axis-axis number,pos-the set value
'      mode 0—set logic pos,non 0—set real pos
'         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

'/************************set COMP+ register*****************************
'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

'/****************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_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 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, axis, value)
      End If
End Function

'/***************************read input******************************
'   read status of input
'      para：number-input port(0 ~ 47)
'         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
```

```
'/***************************output******************************
'   set status 0f output
'      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
'/*****************************deceleration enable*************************
'   no            1：X-Y or X-Y-Z or X-Y-Z-W interpolation      2：Z-W interpolation
'      the function allowable deceleration during driving
'         retutrn 0: correct 1: wrong
'******************************************************************/
Public Function AllowDec(ByVal no As Integer) As Integer
      Result = inp_dec_enable(0, no)
      AllowDec = Result
End Function

'/*************************deceleration disable****************************
'   no            1：X-Y or X-Y-Z or X-Y-Z-W interpolation      2：Z-W interpolation
'      the function for deceleration disable during driving
'         retutrn 0: correct 1: wrong
'*******************************************************************/
Public Function ForbidDec(ByVal no As Integer) As Integer
      Result = inp_dec_disable(0, no)
      ForbidDec = Result
End Function

'/***************getting information about mistaken stop on all axes*****************
'This function is   for getting information about the axis stop
'   value:       error message   0：no error   1：error
'      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****************
```

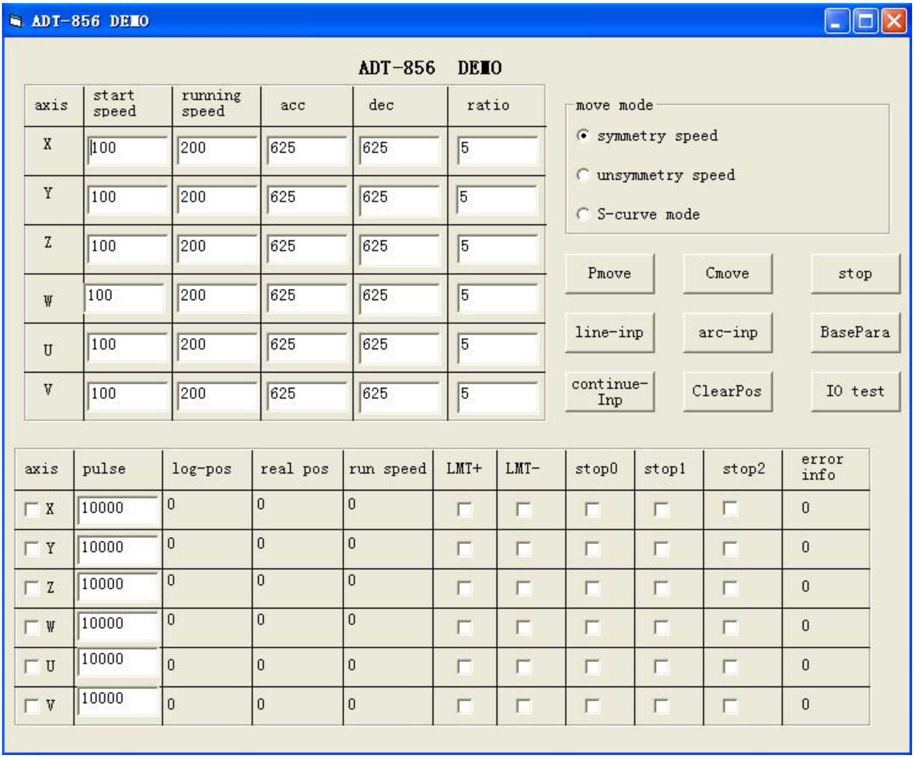
```
'   value：    inter-infor   0：write disable 1：write able
'      retutrn 0: correct 1: wrong
'**********************************************************************/
Public Function Get_AllowInpStatus(ByVal no As Integer, value As Integer) As Integer
      Result = get_inp_status2(0, no, value)
      Get_AllowInpStatus = Result
End Function

'/***********************set deceleration type**********************
'deceleration for symmetry/asymmetry/automatic/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 deceleration point****************************
'set deceleration point during 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
```

2. Function realization module

a) Interface design



Introduction:

1 Speed setting part—used to set starting speed, motion speed and acceleration of every axis; position setting—used to set drive pulse for every axis; drive information—used to real-time display logical position, real position and operation speed of every axis;

2 Motion object—users determine axis joining simultaneous motion or interpolation by selecting drive objects;

3 Simultaneous movement—Used to send single-axis drive commands to all the axis of the selected drive object; interpolation –Used to send interpolation command to all the axis of the selected drive object; stop—stop all the pulse outputs of all axis. All the above data take pulse as the unit.

1.3.2 Initialization codes are inside the window loading event, with the following contents:

```
Private Sub Init_Board()
    Dim count As Integer
    count = Init_Card
    If count < 1 Then MsgBox "Initial adt856 failed"
End Sub
```

1.3.3 Simultaneous movement codes are inside the click event of axisPmove button, whereby various selected objects send corresponding drive commands. The four check boxes (to select objects) are respectively named as X, Y, Z and A, subject with the following code:

```
Private Sub AxisPmove_Click()
    For i = 1 To 6
            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_bW.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
    If m_bU.value = vbChecked Then
        Setup_Speed 5, m_nStartV(4).Text, m_nSpeed(4).Text, m_nAdd(4).Text, m_nDec(4).Text, m_nRatio(4).Text, nAddMode
        Axis_Pmove 5, m_nPulse(4).Text
    End If
```

```
    If m_bV.value = vbChecked Then
        Setup_Speed 6, m_nStartV(5).Text, m_nSpeed(5).Text, m_nAdd(5).Text, m_nDec(5).Text, m_nRatio(5).Text, nAddMode
        Axis_Pmove 6, m_nPulse(5).Text
    End If

End Sub
```

1.3.4 Interpolation codes are inside the click event of InterpMove button, whereby various selected objects send corresponding drive commands.

```
'****************************linear interpolation button    *****************************
Private Sub InterpMove_Click()
    For i = 1 To 6
        g_Ratio(i - 1) = CLng(m_nRatio(i - 1).Text)
    Next i
'************************************linear interpolation***********************************
'**********************************interp-move with six axes (XYZWUV)****************************************
   If m_bX.value = vbChecked And m_bY.value = vbChecked And m_bZ.value = vbChecked And m_bW.value = vbChecked And m_bU.value = vbChecked And m_bV.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
        Setup_Speed 3, m_nSpeed(0).Text, m_nSpeed(0).Text, m_nAdd(2).Text, m_nDec(2).Text, m_nRatio(0).Text, nAddMode
        Setup_Speed 5, m_nSpeed(0).Text, m_nSpeed(0).Text, m_nAdd(4).Text, m_nDec(4).Text, m_nRatio(0).Text, nAddMode
        Interp_Move6 m_nPulse(0).Text, m_nPulse(1).Text, m_nPulse(2).Text, m_nPulse(3).Text, m_nPulse(4).Text, m_nPulse(5).Text
'*********************************interp-move with four axes*********************************
    ElseIf m_bX.value = vbChecked And m_bY.value = vbChecked And m_bZ.value = vbChecked And m_bW.value = vbChecked And m_bU.value <> vbChecked And m_bV.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
        Setup_Speed 3, m_nSpeed(0).Text, m_nSpeed(0).Text, m_nAdd(2).Text, m_nDec(2).Text, m_nRatio(0).Text, nAddMode
        Interp_Move4 m_nPulse(0).Text, m_nPulse(1).Text, m_nPulse(2).Text, m_nPulse(3).Text
'*********************************interp-move with three axes*********************************
```

```
'********************************XYZ********************************
    ElseIf m_bX.value = vbChecked And m_bY.value = vbChecked And m_bZ.value = vbChecked And m_bW.value <> vbChecked And m_bU.value <> vbChecked And m_bV.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
        Setup_Speed 3, m_nSpeed(0).Text, m_nSpeed(0).Text, m_nAdd(2).Text, m_nDec(2).Text, m_nRatio(0).Text, nAddMode
        Interp_Move3 m_nPulse(0).Text, m_nPulse(1).Text, m_nPulse(2).Text
'********************************interp-move with two axes********************************
'********************************XY********************************
    ElseIf m_bX.value = vbChecked And m_bY.value = vbChecked And m_bZ.value <> vbChecked And m_bW.value <> vbChecked And m_bU.value <> vbChecked And m_bV.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, m_nPulse(0).Text, m_nPulse(1).Text
'********************************ZW********************************
    ElseIf m_bZ.value = vbChecked And m_bW.value = vbChecked And m_bX.value <> vbChecked And m_bY.value <> vbChecked And m_bU.value <> vbChecked And m_bV.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 2, 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_bW.value <> vbChecked And m_bU.value <> vbChecked And m_bV.value <> vbChecked Then
        MsgBox "Please choose axis for inp-move", , "Notice"
    Else
       MsgBox "Wrong axis !", , "Notice"
    End If
End Sub
```

1.4 Monitoring module

The monitoring module is used to real-time get motion information of all the axes and display motion information, at the same time of controlling them in motion process without any new motion commands. This module is completed by the timer event, with the following codes:

```
Private Sub Timer1_Timer()
    Dim nLogPos As Long
```

```
        Dim nActPos As Long
        Dim nSpeed As Long
        Dim nstatus(6) As Integer
        Dim InpStatus As Integer
        Dim nStopData(6) As Integer
        Get_MoveStatus 1, InpStatus, 1
        For i = 1 To 6
            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)
'**************LMT+ -、stop0、stop1、stop2*****************
'(XLMT+ : 0,YLMT+ :8,ZLMT+ :16,WLMT+ :24)
            If Read_Input((i - 1) * 8) = 0 Then
               m_bPLimit(i - 1).value = 1
            Else
               m_bPLimit(i - 1).value = 0
            End If
'(XLMT- : 1,YLMT- :9,ZLMT- :17,WLMT- :25)
            If Read_Input((i - 1) * 8 + 1) = 0 Then
               m_bNLimit(i - 1).value = 1
            Else
               m_bNLimit(i - 1).value = 0
            End If
  'stop0(XSTOP0 : 2,YSTOP0 :10,ZSTOP0 :18,WSTOP0 :26)
            If Read_Input((i - 1) * 8 + 2) = 0 Then
                m_bStop0(i - 1).value = 1
            Else
                m_bStop0(i - 1).value = 0
            End If
'stop1(XSTOP1 : 3,YSTOP1 :11,ZSTOP1 :19,WSTOP1 :27)
            If Read_Input((i - 1) * 8 + 3) = 0 Then
                m_bStop1(i - 1).value = 1
            Else
                m_bStop1(i - 1).value = 0
            End If
 'stop2(XSTOP2 : 4,YSTOP2 :12,ZSTOP2 :20,WSTOP2 :28)
```

```
            If Read_Input((i - 1) * 8 + 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 And
    nstatus(4) = 0 And nstatus(5) = 0) And 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

This module is mainly used to control unexpected events during drive process and will immediately stop drive of all the axes. Codes of this stop module are within the click event of CmdStop button, with the following codes:

```
 Private Sub CmdStop_Click()
    For i = 1 To 6
        StopRun i, 0
    Next i
    inp_clear 0
```

End Sub

☞ VC PROGRAMMING SAMPLES

2.1 Preparation

(1) Create a new item and save as "VCExample.dsw";

(2) Load the static library ADT856.lib into the item following the above-introduced method;

2.2 Movement control module

(1) Add a new category in the item and save the header as "CtrlCard.h" and source file as "CtrlCard.cpp";

(2) At first, within the movement control module self-define initialization functions of the movement control card and initialize library functions to be sealed into initialization functions;

(3) Further self-define relevant movement control functions such as speed setting function, single-axis motion function, and interpolation function;

(4) Source codes of the header CtrlCard.h are as follows:

```
# ifndef __ADT856__CARD__
# define __ADT856__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
#define    MAXAXIS     6

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 no);
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 no, 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 no, long value1, long value2);
int Interp_Move3(long value1, long value2, long value3);
int Interp_Move4(long value1, long value2, long value3, long value4);
int Interp_Move6(long value1, long value2, long value3, long value4, long value5, long value6);
int Setup_Range(int axis, long value);
int Interp_Arc(int no, long x, long y, long i,long j);
int Interp_CcwArc(int no, long x, long y, long i,long j);
int End_Board();
int ForbidDec(int no);
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);
CCtrlCard();
int Result;
int no;
```

```
};
;# endif
```

(5) Source codes of the source file CtrlCard.cpp are as follows:

```
# include "stdafx.h"
# include "ADT856.h"
# include "CtrlCard.h"
# include "VCExample.h"
extern int g_CardVer;
CCtrlCard::CCtrlCard()
{
}
/****************** Initialization function ***********************************************
'This function contain those library functions frequently used in control card initialization, which is the foundation to call other functions and must be firstly called in this example program.
'Return <=0 means initialization failure and >0 means initialization success
*****************************************************************************************/
int CCtrlCard::Init_Board()
{
    Result = adt856_initial() ;              //intiial motion-card
    if (Result <= 0) return Result;
    for (int i = 1; i<=MAXAXIS; i++)
    {
        set_range (0, i, 8000000 / 5);       //set range,set ratio as 5
        set_command_pos (0, i, 0);           //set logic pos as 0
        set_actual_pos (0, i, 0);            //set real pos as 0
        set_startv (0, i, 100);              //set start-speed
        set_speed (0, i, 100);               //set motion-speed
        set_acc (0, i, 625);                 //set acceleration
    }
    return 1;
}
/********************set speed**********************
according as para,judge whether is constant-speed
set range to set ratio
```

set start-speed ,motion-speed and acceleration
para: axis: axis number
startv: start-speed
speed: motion-speed
add: acceleration
dec: decelerate
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//Trapezoidal acceleration/ deceleration
    {
            if (mode == 0)//choose Strait line acceleration/deceleration type
            {
                    set_dec1_mode(0,axis,0);//Set symmetry type
                    set_dec2_mode(0,axis,0);//Set automatic deceleration
                    set_ad_mode(0,axis,0);//Set as Strait acceleration/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);
            }
            else if(mode==1)//choose Strait line acceleration/deceleration type
            {
                    set_dec1_mode(0,axis,1);//asymmetry
                    set_dec2_mode(0,axis,0);//Set automatic deceleration
                    set_ad_mode(0,axis,0);//Set as Strait acceleration/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);
            }
            else if(mode==2)
            {//choose S-curve acceleration/deceleration type
                float time;//time
                float addvar;//changing rate of add
                long k;
                 time = (float)(speed-startv)/(add/2);
                addvar=add/(time/2);;//changing rate of add
                k=(long)(62500000/addvar)*ratio;
                 set_dec2_mode(0,axis,0);//automatic deceleration
                 set_ad_mode(0,axis,1);//as Strait acceleration/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_acac (0, axis,k );
            }
        }
        return Result;
    }
/*****************************************************

OnButtonPmove() :axis move
*****************************************************/
void CDEMODlg::OnButtonPmove()
{
        UpdateData(TRUE);
long Startv[]={m_nStartvX,m_nStartvY,m_nStartvZ,m_nStartvW,m_nStartvU,m_nStartvV};
//start-speed
long Speed[]={m_nSpeedX,m_nSpeedY,m_nSpeedZ,m_nSpeedW,m_nSpeedU,m_nSpeedV};
//run-speed
long Dec[]=       {m_nDecX,m_nDecY,m_nDecZ,m_nDecW,m_nDecU,m_nDecV };
//dec
long Add[]    ={m_nAddX,m_nAddY,m_nAddZ,m_nAddW,m_nAddU,m_nAddV};
//acc
long Ratio[]={m_nRatioX,m_nRatioY,m_nRatioZ,m_nRatioW,m_nRatioU,m_nRatioV};
//ratio
        //*************X axis move******************//
        if(m_bX )
```

```
        {
            //************set speed*****************//
            g_CtrlCard.Setup_Speed(1, m_nStartvX, m_nSpeedX, m_nAddX, m_nDecX,
m_nRatioX, m_nAddMode);
            g_CtrlCard.Axis_Pmove(1, m_nPulseX);
        }
        //*************Y axis move******************//
        if(m_bY )
        {
            //************set speed*****************//
            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 )
        {
            //************set speed*****************//
            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_bW )
        {
            //************set speed*****************//
            g_CtrlCard.Setup_Speed(4, m_nStartvW, m_nSpeedW, m_nAddW, m_nDecW,
m_nRatioW, m_nAddMode);

            g_CtrlCard.Axis_Pmove(4, m_nPulseW);
        }
        if(m_bU )
        {
            //************set speed*****************//
            g_CtrlCard.Setup_Speed(5, m_nStartvU, m_nSpeedU, m_nAddU, m_nDecU,
m_nRatioU, m_nAddMode);
            g_CtrlCard.Axis_Pmove(5, m_nPulseU);
        }
        if(m_bV )
        {
            //************set speed*****************//
            g_CtrlCard.Setup_Speed(6, m_nStartvV, m_nSpeedV, m_nAddV, m_nDecV,
```

```
m_nRatioV, m_nAddMode);
            g_CtrlCard.Axis_Pmove(6, m_nPulseV);
      }
}
/*****************************************************
   OnButtonInpmove():linear interpolation button
*****************************************************/
void CDEMODlg::OnButtonInpmove()
{
      UpdateData();
long Startv[]={m_nStartvX,m_nStartvY,m_nStartvZ,m_nStartvW,m_nStartvU,m_nStartvV};
long Speed[]={m_nSpeedX,m_nSpeedY,m_nSpeedZ,m_nSpeedW,m_nSpeedU,m_nSpeedV};
      long Add[]   ={m_nAddX,m_nAddY,m_nAddZ,m_nAddW,m_nAddU,m_nAddV};
      long Dec[]   ={m_nDecX,m_nDecY,m_nDecZ,m_nDecW,m_nDecU,m_nDecV};
      long Pulse[]={m_nPulseX,m_nPulseY,m_nPulseZ,m_nPulseW,m_nPulseU,m_nPulseV};
      long Ratio[]={m_nRatioX,m_nRatioY,m_nRatioZ,m_nRatioW,m_nRatioU,m_nRatioV};
      //************interp-move with two axes************//

      if(m_bX && m_bY && !m_bZ && !m_bW && !m_bU && !m_bV)
//XY
      {
            g_CtrlCard.Setup_Speed(1, Startv[0], Speed[0], Add[0], Dec[0], Ratio[0],
m_nAddMode);
            g_CtrlCard.Interp_Move2(1,   Pulse[0], Pulse[1]);
      }
      else if(!m_bX && !m_bY && m_bZ && m_bW && !m_bU && !m_bV)
//ZW
      {
            g_CtrlCard.Setup_Speed(3, Startv[2], Speed[2], Add[2], Dec[2],   Ratio[2],
m_nAddMode);
            g_CtrlCard.Interp_Move2(2, Pulse[2], Pulse[3]);
      }
      //************interp-move with three axes***************//
      else if(m_bX && m_bY && m_bZ && !m_bW && !m_bU && !m_bV)
//XYZ
      {
            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( Pulse[0], Pulse[1], Pulse[2]);
        }
        //*************interp-move with four axes**************//
        else if(m_bX && m_bY && m_bZ && m_bW && !m_bU && !m_bV)
//XYZW
        {
            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_Move4(Pulse[0], Pulse[1], Pulse[2], Pulse[3]);
//**************interp-move with six axes***************//
        else if(m_bX && m_bY && m_bZ && m_bW && m_bU && m_bV)
//XYZWUV
        {
            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.Setup_Speed(5, Speed[0], Speed[0], Add[4], Dec[4] , Ratio[0],
m_nAddMode);
            g_CtrlCard.Interp_Move6(Pulse[0], Pulse[1], Pulse[2], Pulse[3],Pulse[4], Pulse[5]);
        }
        else if(!m_bX && !m_bY && !m_bZ && !m_bW && !m_bU && !m_bV)
        {
            MessageBox("Please choose axis for inp-move!","Notice");
        }
        else
        {
            MessageBox(" wrong axis!","Notice");
        }
}
        /*************** Get information of movement*****************************
        This function is used to feedback the current logic position, real position and motion
        speed of the selected axis
        Return =0 means success, and Return =1 means error
        ********************************************************************************/
        int CCtrlCard::Get_CurrentInf(int axis, long &LogPos, long &ActPos, long &Speed)
```

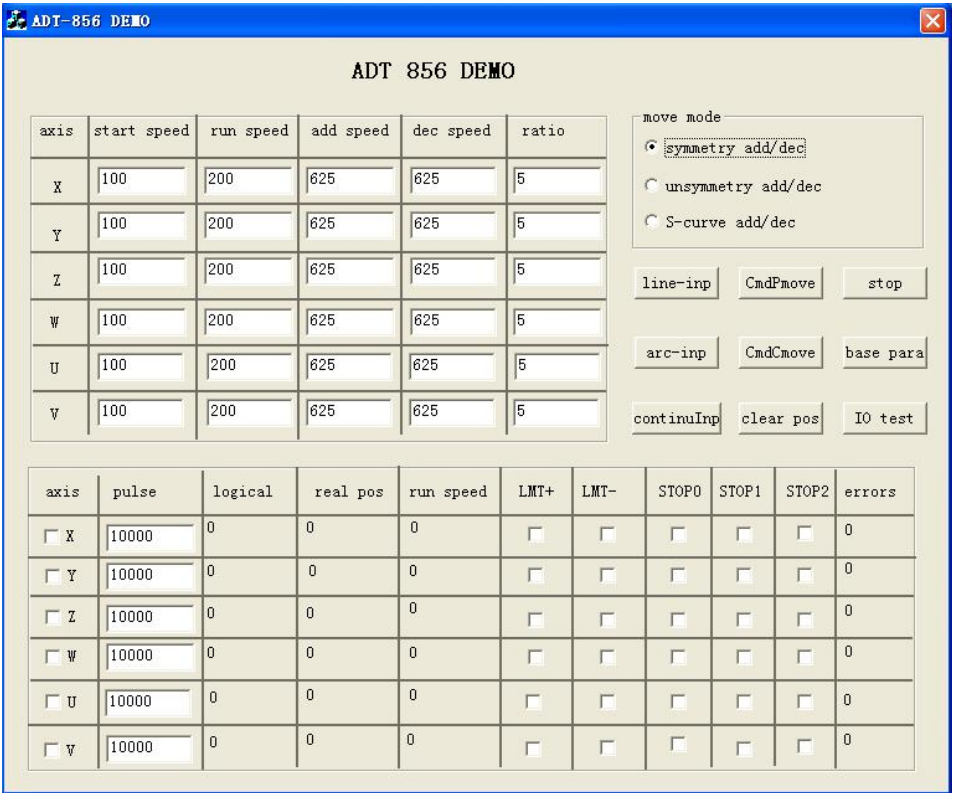
```
{
        Result = get_command_pos(0, axis, &LogPos);
        get_actual_pos(0, axis, &ActPos);
        get_speed(0, axis, &Speed);
                return Result;
}

/**************** Stop motion function**************************************************
This function provides either sudden stop mode or deceleration stop mode
Return =0 means success, and Return =1 means error
*************************************************************************************/
int CCtrlCard::StopRun(int axis, int mode)
{
        if (mode == 0)
              Result = sudden_stop(0, axis);              //Sudden stop
       else
              Result = dec_stop(0, axis);                  //Deceleration stop
       return Result;
}

/****************Get motion status****************************************************
This function is used to get single-axis drive status or interpolation drive status
Return =0 means success, and Return =1 means error
*************************************************************************************/
int CCtrlCard::Get_Status(int axis, int &value, int mode)
{
        if (mode==0)                //Get single-axis motion status
                Result=get_status(0,axis,&value);
        Else                          //Get motion status of interpolation
                Result=get_inp_status(0,&value);
        return Result;
}
```

2.3 Function realization module

2.3.1 Interface design



Remark:

(1) Speed setting part—used to set starting speed, drive speed and acceleration of every axis; position setting—used to set drive pulse for every axis; motion information—used to real-time display logical position, real position and motion speed of every axis;

(2) Drive object—users determine axis joining simultaneous movement or interpolation by selecting drive objects;

(3) Simultaneous movement—Used to send single-axis drive commands to all the axis of the selected drive object; interpolation –Used to send interpolation command to all the axis of the selected drive object; stop—stop all the pulse outputs of all axis.

All the above data take pulse as the unit.

2.3.2 Initialization codes for the movement control card are inside window initialization, while users shall supplement the following codes:

```
int i=g_CtrlCard.Init_Board();
//************initial 856 motion-card*************
if (g_CtrlCard.Init_Board() <= 0)
    MessageBox (“Fail to initial motion-card!");
else
    MessageBox ("Succeed in initial motion-card!");
    //******* set start-speed 100        *********
    m_nStartvX = 100;
    m_nStartvY = 100;
    m_nStartvZ = 100;
    m_nStartvW = 100;
    m_nStartvU = 100;
    m_nStartvV = 100;
    //*********set run-speed 200********
    m_nSpeedX   = 200;
    m_nSpeedY   = 200;
    m_nSpeedZ   = 200;
    m_nSpeedW   = 200;
    m_nSpeedU   = 200;
    m_nSpeedV   = 200;
    //*********set add 125**********
    m_nAddX      = 625;
    m_nAddY      = 625;
    m_nAddZ      = 625;
    m_nAddW      = 625;
    m_nAddU      = 625;
    m_nAddV      = 625;
    //*********set ratio 5**************
    m_nRatioX   = 5;
    m_nRatioY   = 5;
    m_nRatioZ   = 5;
    m_nRatioW   = 5;
    m_nRatioU   = 5;
    m_nRatioV   = 5;
    //********set pulse 10000******
    m_nPulseX   = 10000;
    m_nPulseY   = 10000;
    m_nPulseZ   = 10000;
```

```
        m_nPulseW   = 10000;
        m_nPulseU   = 10000;
        m_nPulseV   = 10000;
        /************set dec 25****************/
        m_nDecW=625;
        m_nDecX=625;
        m_nDecY=625;
        m_nDecZ=625;
        m_nDecU=625;
            m_nDecV=625;
        //***********set time*************
        SetTimer(MAINTIMER,100,NULL);
```

2.3.3 Simultaneous movement codes are inside the click message of Simultaneous movement button and will send various drive commands for various selected targets; the codes are as follows:

```
void CDEMODlg::OnButtonPmove()
{
    UpdateData(TRUE);
long Startv[]={m_nStartvX,m_nStartvY,m_nStartvZ,m_nStartvW,m_nStartvU,m_nStartvV};
//start-speed
long Speed[]={m_nSpeedX,m_nSpeedY,m_nSpeedZ,m_nSpeedW,m_nSpeedU,m_nSpeedV};
//run-speed
long Dec[]=     {m_nDecX,m_nDecY,m_nDecZ,m_nDecW,m_nDecU,m_nDecV };
//dec
long Add[]   ={m_nAddX,m_nAddY,m_nAddZ,m_nAddW,m_nAddU,m_nAddV};
//acc
long Ratio[]={m_nRatioX,m_nRatioY,m_nRatioZ,m_nRatioW,m_nRatioU,m_nRatioV};
//ratio
    //**************X axis move*******************//
    if(m_bX )
    {
        //*************set speed******************//
        g_CtrlCard.Setup_Speed(1, m_nStartvX, m_nSpeedX, m_nAddX, m_nDecX,
m_nRatioX, m_nAddMode);
        g_CtrlCard.Axis_Pmove(1, m_nPulseX);
    }
    //**************Y axis move*******************//
    if(m_bY )
```

```
        {
            //************set speed****************//
            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 )
        {
            //************set speed****************//
            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_bW )
        {
            //************set speed****************//
            g_CtrlCard.Setup_Speed(4, m_nStartvW, m_nSpeedW, m_nAddW, m_nDecW,
m_nRatioW, m_nAddMode);

            g_CtrlCard.Axis_Pmove(4, m_nPulseW);
        }
        if(m_bU )
        {
            //************set speed****************//
            g_CtrlCard.Setup_Speed(5, m_nStartvU, m_nSpeedU, m_nAddU, m_nDecU,
m_nRatioU, m_nAddMode);
            g_CtrlCard.Axis_Pmove(5, m_nPulseU);
        }
        if(m_bV )
        {
            //************set speed****************//
            g_CtrlCard.Setup_Speed(6, m_nStartvV, m_nSpeedV, m_nAddV, m_nDecV,
m_nRatioV, m_nAddMode);
            g_CtrlCard.Axis_Pmove(6, m_nPulseV);
        }
}
```

2.3.4 Interpolation codes are inside the click message of the inp_move button and will send various drive commands for various selected targets; the codes are as follows:

```
void CDEMODlg::OnButtonInpmove ()
    {
UpdateData();
long Startv[]={m_nStartvX,m_nStartvY,m_nStartvZ,m_nStartvW,m_nStartvU,m_nStartvV};
long Speed[]={m_nSpeedX,m_nSpeedY,m_nSpeedZ,m_nSpeedW,m_nSpeedU,m_nSpeedV};
    long Add[]   ={m_nAddX,m_nAddY,m_nAddZ,m_nAddW,m_nAddU,m_nAddV};
    long Dec[]   ={m_nDecX,m_nDecY,m_nDecZ,m_nDecW,m_nDecU,m_nDecV};
    long Pulse[]={m_nPulseX,m_nPulseY,m_nPulseZ,m_nPulseW,m_nPulseU,m_nPulseV};
    long Ratio[]={m_nRatioX,m_nRatioY,m_nRatioZ,m_nRatioW,m_nRatioU,m_nRatioV};
    //*************interp-move with two axes*************//

    if(m_bX && m_bY && !m_bZ && !m_bW && !m_bU && !m_bV)
//XY
    {
        g_CtrlCard.Setup_Speed(1, Startv[0], Speed[0], Add[0], Dec[0], Ratio[0],
m_nAddMode);
        g_CtrlCard.Interp_Move2(1,   Pulse[0], Pulse[1]);
    }
    else if(!m_bX && !m_bY && m_bZ && m_bW && !m_bU && !m_bV)
//ZW
    {
        g_CtrlCard.Setup_Speed(3, Startv[2], Speed[2], Add[2], Dec[2],   Ratio[2],
m_nAddMode);
        g_CtrlCard.Interp_Move2(2, Pulse[2], Pulse[3]);
    }
    //*************interp-move with three axes****************//
    else if(m_bX && m_bY && m_bZ && !m_bW && !m_bU && !m_bV)
//XYZ
    {
        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( Pulse[0], Pulse[1], Pulse[2]);
    }
    //**************interp-move with four axes***************//
    else if(m_bX && m_bY && m_bZ && m_bW && !m_bU && !m_bV)
//XYZW
    {
```

```
            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_Move4(Pulse[0], Pulse[1], Pulse[2], Pulse[3]);
//***************interp-move with six axes****************//
      else if(m_bX && m_bY && m_bZ && m_bW && m_bU && m_bV)
//XYZWUV
      {
            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.Setup_Speed(5, Speed[0], Speed[0], Add[4], Dec[4] , Ratio[0],
m_nAddMode);
            g_CtrlCard.Interp_Move6(Pulse[0], Pulse[1], Pulse[2], Pulse[3],Pulse[4], Pulse[5]);
      }
      else if(!m_bX && !m_bY && !m_bZ && !m_bW && !m_bU && !m_bV)
      {
            MessageBox("Please choose axis for inp-move!","Notice");
      }
      else
      {
            MessageBox(" wrong axis!","Notice");
      }
    }
```

2.4 Monitoring module

The monitoring module is used to real-time get drive information of all the axes and display movement information, at the same time of controlling them in drive process without any new drive commands. This module is completed through timer messages, with the following codes:

```
 void CDEMODlg::OnTimer(UINT nIDEvent)
{
      long log,act,spd;
      int Stopdata[6];
      UINT
```

```
nID1[]={IDC_POS_LOGX,IDC_POS_LOGY,IDC_POS_LOGZ,IDC_POS_LOGW,IDC_POS_
LOGU,IDC_POS_LOGV};
UINT nID2[]={IDC_POS_ACTX,IDC_POS_ACTY,IDC_POS_ACTZ,IDC_POS_ACTW,
IDC_POS_ACTU,IDC_POS_ACTV};
UINT nID3[]={IDC_RUNSPEED_X,IDC_RUNSPEED_Y,IDC_RUNSPEED_Z,
IDC_RUNSPEED_W,IDC_RUNSPEED_U,IDC_RUNSPEED_V};
UINT nID4[]={m_nRatioX,m_nRatioY,m_nRatioZ,m_nRatioW,m_nRatioU,m_nRatioV};
UINT nID5[]={IDC_STOPDATA_X, IDC_STOPDATA_Y, IDC_STOPDATA_Z,
IDC_STOPDATA_W,IDC_STOPDATA_U, IDC_STOPDATA_V};
        CStatic *lbl;
        CString str,stopinf;
        int status[6];
        for (int i=1; i<MAXAXIS+1; i++)
        {
                g_CtrlCard.Get_CurrentInf(i,log,act,spd);    //Get logic-pos ,actual-pos and run-speed
                //******display logic-pos*********//
                lbl=(CStatic*)GetDlgItem(nID1[i-1]);
                str.Format("%ld",log);
                lbl->SetWindowText(str);
                //******display actual-pos*********//
                lbl=(CStatic*)GetDlgItem(nID2[i-1]);
                str.Format("%ld",act);
                lbl->SetWindowText(str);
                //******display run-speed*********//
                lbl=(CStatic*)GetDlgItem(nID3[i-1]);
                str.Format("%ld",spd*nID4[i-1]);
                lbl->SetWindowText(str);
                //******Get status*********//
                g_CtrlCard.Get_Status(i,status[i-1],0);
                //******Get error Information*********//
                g_CtrlCard.Get_ErrorInf(i, Stopdata[i-1]);
            stopinf.Format("%d",Stopdata[i-1]);
        //      stopinf.Format("%d",status[0]);
                lbl=(CStatic*)GetDlgItem(nID5[i-1]);
                lbl->SetWindowText(stopinf);
        }
        //TRACE("%d\n",status[0]);
//*********************信号检测****************************
//              XLMT+ -0                        XLMT-  —1
//              XSTOP0 -2                       XSTOP1  —3
```

```
//            XSTOP2  －4
//            YLMT+ -8               YLMT-   －9
//            YSTOP0 -10             YSTOP1  －11
//            YSTOP2  －12
//            ZLMT+ -16              ZLMT-  －17
//            ZSTOP0 -18             ZSTOP1  －19
//            ZSTOP1  －20
//            WLMT+ -24              WLMT-  －25
//            WSTOP0 -26             WSTOP1  －27
//            WSTOP2  －28
//            LMT+ -32               ULMT-  －33
//            USTOP0 -34             USTOP1  －35
//            USTOP1  －36
//            VLMT+ -40              VLMT-  －41
//            VSTOP0 -42             VSTOP1  －43
//            VSTOP2  －44
//*************************************************************
    UINT nIDIN[]={    IDC_LIMIT_X,IDC_LIMIT_X2,            //XLMT+/XLMT-
                        IDC_STOP0_X,IDC_STOP1_X,
                        IDC_STOP2_X3,
                      IDC_LIMIT_Y,IDC_LIMIT_Y2,            //YLMT+/YLMT-
                        IDC_STOP0_Y,IDC_STOP1_Y2,
                    IDC_STOP2_Y3,
                       IDC_LIMIT_Z,IDC_LIMIT_Z2,           //ZLMT+/ZLMT-
                        IDC_STOP0_Z,IDC_STOP1_Z,
                    IDC_STOP2_Z2,
                        IDC_LIMIT_W,IDC_LIMIT_W2,
//WLMT+/WLMT-
                        IDC_STOP0_W,IDC_STOP1_W ,
                        IDC_STOP2_W2,
                     IDC_LIMIT_U,IDC_LIMIT_U2,             //ULMT+/WLMT-
                        IDC_STOP0_U,IDC_STOP1_U,
                    IDC_STOP2_U2,
                        IDC_LIMIT_V,IDC_LIMIT_V2,
//VLMT+/WLMT-
                        IDC_STOP0_V,IDC_STOP1_V ,
                        IDC_STOP2_V2
    };
    int
io[]={0,1,2,3,4,8,9,10,11,12,16,17,18,19,20,24,25,26,27,28,32,33,34,35,36,40,41,42,43,44};
```

```
    CButton *btn;
    int value;
    for (i=0; i<30; i++)
    {
        value=g_CtrlCard.Read_Input(io[i]);                    //read input signal
        btn=(CButton*)GetDlgItem(nIDIN[i]);
        btn->SetCheck(value==0?1:0);
    }
        CDialog::OnTimer(nIDEvent);
}
```

2.5 Stop module

This module is mainly used to control unexpected events during drive process and will immediately stop drive of all the axes. Codes of this stop module are within the click messages of Stop button, with the following codes:

```
void CDEMODlg::OnButtonStoprun()
{
    for (int i = 1; i<= MAXAXIS; i++){
        g_CtrlCard.StopRun(i,0);
     }
}
```

Chapter 11 Normal failures and solutions

(1) Movement control card detection failure

During use of control card, if encountering failure to detect the control card, users may follow the following items to check:

1 Check whether drive program for the control card has been installed step by step following installation guide and whether there is the dynamic library file for the control card under the system menu (System32 or System);
2 Check touch between the movement control card and the slot; users may test it by re-inserting or changing the slot, alternatively, use a rubber to clean dirt on the golden finger of the control card and re-insert;
3 Under the system equipment manager, check whether there is conflict between the movement control card and other hardware. In case of use of PCI card, users may

remove other cards or boards first, such as sound card and network card; in case of PC104 card, users may adjust the dialing switch and reset the base address, while the base address used during card initialization must be same as the actual base address;

4 Check whether there are any problems with the operating system; users may test it through re-installing other versions of operating systems;

5 If failing to find the control card after the above steps, users may change the control card for further detection so as to discover whether there is damage with the control card.

MOTOR SERVICE FAILURE

In case the motor breakdowns while the movement control card works normally, users may follow the following points for troubleshooting.

(1) Motor makes no reaction when the movement control card outputs pulses

a) Check cable between the control card and the terminal panel;

b) Check whether the pulse and direction signal wire of the motor driver has been correctly connected to the terminal panel;

c) Check connection of the external power supply for the servo driver;

d) Check whether there is alarming status in the servo/ stepping motor driver; in case of any alarm there, follow codes corresponding to alarms to check the reason.

e) Check connection to the servo SON and whether there is excitation status in the servo motor ;

f) In case of servo motor, check control method of the driver; control card of our company support the Position Control Method.

g) Damage to the motor/ driver

(2) Stepping motor makes abnormal noise during service and motor makes obvious out-steps.

a) Calculate motor speed and make sure the stepping motor is under 10-15 rounds per second instead of faster speed;

b) Check internal obstruction in the mechanical part or resistance to the machine;

c) Change to large-moment motors if the current motor is not sufficient;

d) Check current and voltage of the driver; current shall be set as 1.2 of the nominated current and supply voltage shall be within the nominated range;

e) Check the starting speed of the controller; normal starting speed shall be 0.5-1 and the acceleration/ deceleration time shall be over 0.1 second.

(3) Servo/ stepping motor makes obvious vibration or noises during processing

a) Reduce the position ring gain and speed ring gain of the driver while allowed by the positioning precision, if the cause is such ring gains are too big;

b) Adjust machine structure if the cause is poor machine rigidity;

c) Change to large-moment motors if the current motor is not sufficient;

d) Avoid the co-vibration area of the motor or increase partitions so as not to have the speed of stepping motor within the co-vibration area of the motor.

(4) Motor positions inaccurately

a) Check whether the mechanic screw pitch and pulses per round comply with the parameters set in the actual application system, i.e., pulse equivalent;

b) Enlarge position ring gain and speed ring gain in case of servo motor;

c) Check screw gap of the machine in the way of measuring the backward gap of a screw through a micrometer and adjust the screw if there is any gap;

d) In case of inaccurate positioning out of regular time or position, check external disturbance signals;

e) Check whether it is due to non-powerful motor that there is shaking or out-step.

(5) Motor makes no direction

a) Check DR+ DR- cable for connection error or loose connection;

b) Make sure the pulse mode applied in the control card comply with the actual driver mode; this control card support either “pulse + direction” or “pulse + pulse” mode.

c) Check broken cable or loose connection along the motor cable, in case of stepping motor.

☞ ABNORMAL SWITCH AMOUNT INPUT

In case some input signals give unusual detection results during system adjusting and running, users may check in accordance with the following methods:

(1) No signal input

- Check whether the wiring is correct according to the above-introduced wiring maps for normal switch and approach switch and ensure the public port for photoelectric coupling of input signals have been connected with anode of internal or external power supply (+12V or 24V);
- Check switch model and wiring method; the input switch for I/O points of our company is of NPN model.
- Check whether there is damage with the photoelectric coupler. In case of normal wiring, input status will not change no matter the input point is broken or closed; users may use multi-use meter to check whether the photoelectric coupler has been broken, and if yes, replace with a new one;
- Check the 12V or 24V power supply to the switch;
- Check whether there is damage to the switch.

(2) Non-continuous signals

i. Check whether there is disturbance by detecting signal status in the I/O test interface; in case of disturbance, increase with Model 104 multiple layer capacitor or apply blocking cables;

ii. If the machine makes obvious shaking or unusual work stop during normal service, check whether there is disturbance to the limit switch signals or the limit switch work reliably;

iii. Check connection of external cables.

(3) Inaccurate reset

iv. Too high speed decreases reset speed ;

v. Check disturbance source if the problem is there is external disturbance to signals;

vi. Wrong resetting direction;

vii. Improper installation position of the reset switch or loose switch

(4) Limit out of use

viii. Check whether the limit switch still works under the I/O test;

ix. Too high speed during manual or automatic processing;

x. Check disturbance source if the problem is there is external disturbance to signals;

xi. Wrong manual direction;

xii. Improper installation position of the reset switch or loose switch

☞ Abnormal output of switch amount

Abnormal output of switch amount may be checked in the following method:

(1) Abnormal output

xiii. Check whether the wiring is correct following the above-introduced wiring for output points and ensure the output public port (earthing line) has been connected with the earthing line of the to-be-used power supply;

xiv. Check whether there is any damage to the output components;

xv. Check whether there is damage with the photoelectric coupler. Users may use multi-use meter to check whether the photoelectric coupler has been broken, and if yes, replace with a new one;

xvi. Safety issue. Continuous dioxide (model: IN4007 or IN4001) must be serial connected in case of output with sensitive loading.

(2) Judgment method for improper output

Break the external cable at the output point and connect at the output point a pull-up resistor of around 10K to the power supply, while earthing line of the output must be connected with GND of the power supply; then users use the red pen of a multi-use meter to touch the 12V anode, and black pen to touch the signal output port, at the same time of using hand to touch the button on the test interface to see whether there is voltage output; in case of any voltage output, check the external circuit, otherwise check connection to the public port of boards/ cards and internal photoelectric couplers.

☞ Abnormal encoder performance

Abnormal encoder performance may be checked in the following method:

(1) Check encoder cables and make sure they comply with the above-introduced differential or collecting electrode wiring method;

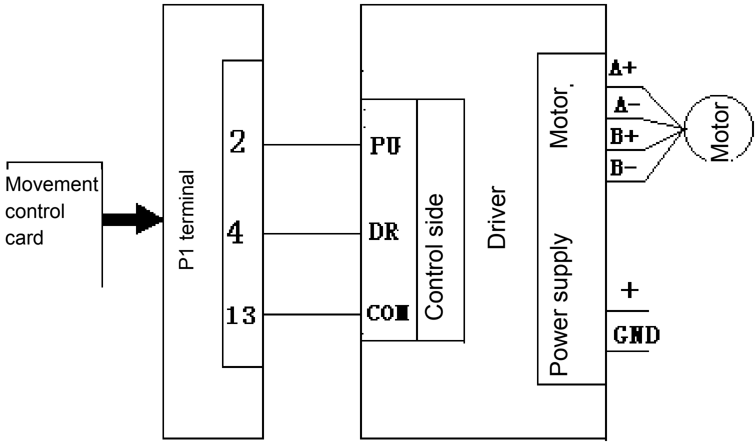
(2) Check encoder voltage. The movement control card normally accepts +5V signals. In case a +12V or +24V encoder is selected, users must serial connect a 1K (+12V) resistor between the Phase A /B of the encoder and Phase A /B of the terminal panel;

(3) Inaccurate encoder counting. External cables to the encoder must be blocking double-twisted cables, and shall be tied free from those cables with strong disturbance such as strong electricity, specifically, they shall be separated for over 30～50MM.

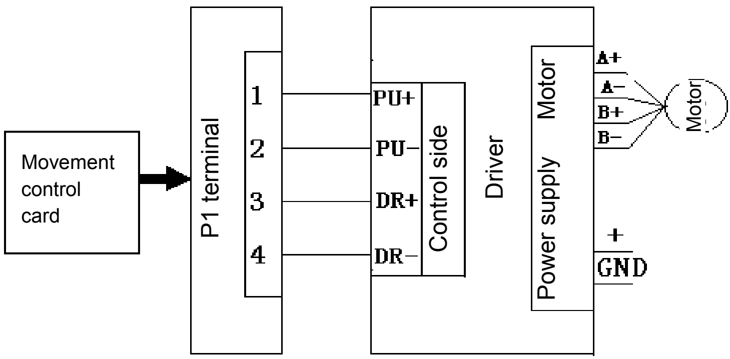
Appendix A Typical wiring for motor driver

All the following wiring takes X axis as example.

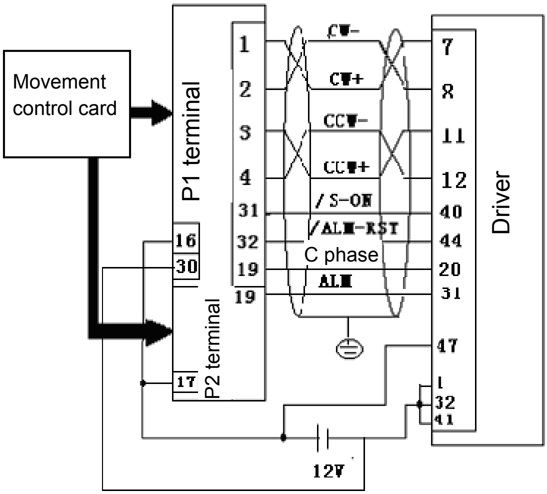
☞ Stepping motor driver common anode wiring



☞ **Stepping motor driver differential wiring**



☞ **Yaskawa servo driver wiring**



☞ **Panasonic A4 servo driver wiring**