4-axis Motion Control Card
ADT-8949

Basic Information

This Manual is written by Adtech (Shenzhen) Technology Co., Ltd. This Manual is mainly written by: Ai Xiaoyun.

This Manual was first released on May 12, 2013, with version No. 14.08.29.16 and item number BZ001C057A.

Copyright Notice

Precautions

◆ Transport and storage:

☞ Do not stack product package more than six layers;

☞ Do not climb, stand on or place heavy stuff on the product package;

☞ Do not pull the cable still connecting with machine to move product.

☞ Forbid impact and scratch on the panel and display;

☞ Prevent the product package from humidity, sun exposure, and rain.

◆ Open-box inspection:

☞ Open the package to confirm the product to be purchased by you.

☞ Check damages situation after transportation;

☞ Confirm the integrity of parts comparing with the parts list or damages situation;

☞ Contact our company promptly for discrepant models, shortage accessories, or transport damages.

◆ Wiring

☞ Ensure the persons involved into wiring and inspecting are specialized staff;

☞ Guarantee the product is grounded with less than 4Ω grounding resistance. Do not use neutral line (N) to substitute earth wire.

☞ Ensure grounding to be correct and solid, in order to avoid product failures or unexpected consequences;

☞ Connect the surge absorption diodes to the product in the required direction, otherwise, the product will be damaged;

☞ Ensure the power switch is OFF before inserting or removing plug, or disassembling chassis.

◆ Overhauling

☞ Ensure the power is OFF before overhauling or components replacement;

☞ Make sure to check failures after short circuit or overloading, and then restart the machine after troubleshooting

☞ Do not allow to frequently connect and disconnect the power, and at least one minute interval between power-on and power-off.

◆ **Miscellaneous**

☞ Do not open housing without permission;

☞ Keep power OFF if not in use for a long time;

☞ Pay close attention to keep dust and ferrous powder away from control;

☞ Fix freewheel diode on relay coil in parallel if non-solid state relay is used as output relay. Check whether power supply meets the requirement to ensure not burning the control.

☞ Install cooling fan if processing field is in high temperature, due to close relationship between service life of the control and environmental temperature. Keep proper operative temperature range for the control: 0℃ ~ 60℃.

☞ Avoid using the product in the overheating, humid, dusty, or corrosive environments;

☞ Add rubber rails as cushion on the place with strong vibration.

◆ **Maintenance**

Please implement routine inspection and regular check upon the following items, under the general usage conditions (i.e. environmental condition: daily average 30℃, load rate: 80%, and operating rate: 12 hours/ day)

Table 1 Equipment Inspection Requirements

Routine Inspection	Routine	● Confirm environmental temperature, humidity, dust, or foreign objects. ● Confirm abnormal vibration and noise; ● Check whether vents are blocked by yarn etc.
Regular Check	One year	● Check whether solid components are loose ● Confirm whether terminal block is

Contents

Contents..4
Overview...8
Chapter 1 Product Introduction..8
1.1 DESCRIPTION:..8
1.2 HARDWARE SPECIFICATIONS: 8
1.3 CONTROL FUNCTIONS:..9
1.4 SOFTWARE SUPPORT: 11
1.5 APPLICATIONS: 11
1.6 ADT-8949 SERIES FEATURES LIST:..11
Hardware...14
Chapter 1 Hardware Installation 14
1.1 PACKING LIST: 14
1.2 INSTALLATION SIZE:...15
1.3 INSTALLATION STEPS: 15
Chapter 2 Electrical Connection 16
2.1 WIRING DIAGRAM: 16
2.2 25-PIN PORT I/O SIGNAL DEFINITION: 16
2.3 SIGNAL DEFINITION OF J2 INTERFACE..................................19
2.4 SIGNAL DEFINITION OF J3 INTERFACE..................................22
2.5 J4 SINGAL DEFINITION, RESERVED, NO SPECIFIC FUNCTION 24
2.6 J5 SINGAL DEFINITION 24
2.7 CONNECTING PULSE/DIRECTION OUTPUT SIGNAL:............................25
2.8 CONNECTING ENCODER INPUT SIGNAL: 26
2.9 CONNECTING DIGITAL INPUT: 26
2.10 CONNECTING DIGITAL OUTPUT: 30
Chapter 3 Software Installation 32
3.1 INSTALLING DRIVERS ON WIN98: 32
3.2 INSTALLING DRIVERS ON WINXP: 35
3.3 INSTALLING DRIVERS ON WIN7 37
Chapter 4 Electrical Specifications 42
4.1 SWITCH INPUT: 42
4.2 COUNT INPUT:...42
4.3 PULSE OUTPUT:...43

4.4 SWITCHING OUTPUT: 43
4.5 POWER OUTPUT: 43
Chapter 5 Common Servo Wiring Diagrams..43
5.1 PANASONIC A5 SERVO WIRING DIAGRAM: 43
Chapter 6 Working Environment 45
6.1 OPERATING TEMPERATURE: 45
6.2 STORAGE TEMPERATURE: 45
6.3 OPERATING HUMIDITY: 46
6.4 STORAGE HUMIDITY: 46
Software Programming..46
Chapter 1 Function Description..46
1.1 Pulse output mode: 46
1.2 HARDWARE LIMIT SIGNAL: 46
1.3 QUANTITATIVE DRIVING: 47
1.4 CONTINUOUS DRIVING: 48
1.5 SPEED CURVE: 49
1.6 POSITION LATCH: 52
1.7 EXTERNAL SIGNAL DRIVING: 53
1.8 LARGE CACHE SMALL SEGMENT: 53
1.9 SPEED ADAPTIVE MODEL: 53
1.10 SPHERICAL ARC INTERPOLATION: 54
1.11 VARIOUS ACCELERATION AND DECELERATION MODES: 55
1.12 NURBS INTERPOLATION: 55
1.13 SIMULTANEOUS CONTROL OF MULTIPLE PROCESSES: 55
1.14 GANTRY DUAL-DRIVE, CHANGING DRIVE SPEED AND TARGET POSITION IN MOTION: 56
1.15 SPECIFY THE TIME FOR FOUR-AXIS LINEAR INTERPOLATION: 56
1.16 PULSE GENERATOR: 56
1.17 GET ARC LENGTH, SET 15 FILTER LEVELS FOR INPUT POINT: 57
Chapter 2 Motion Control Library Function Guide 57
2.1 Introduction of ADT-8949 function library..57
2.2 CALLING DLL ON WINDOWS 57
2.3 CALLING IN VC 58
2.4 CALLING IN VB 58
2.5 CALLING IN C++BUILDER 58
2.6 RETURN VALUE AND MEANING OF LIBRARY FUNCTION 59
Chapter 3 Key Points for Motion Control Development 60

3.1 Card initialization..60
3.2 SPEED SETTING..61
Chapter 4 System Security Mechanism..62
4.1 Monitoring error message: 62
4.2 LIMIT:..62
Chapter 5 High Speed Capture of External Signal Homing.......63
5.1 Homing motion: 63
Chapter 6 Linkage Control..65
6.1 SINGLE AXIS QUANTITATIVE UNIFORM MOTION:................................66
6.2 SINGLE AXIS QUANTITATIVE SYMMETRY TRAPEZOIDAL ACCELERATION / DECELERATION MOTION: 68
6.3 SINGLE AXIS QUANTITATIVE ASYMMETRY TRAPEZOIDAL ACCELERATION / DECELERATION MOTION: 70
6.4 SINGLE AXIS QUANTITATIVE S-CURVE ACCELERATION / DECELERATION MOTION: 73
6.5 SINGLE AXIS QUANTITATIVE EXPONENTIAL ACCELERATION / DECELERATION MOTION: 75
6.6 SINGLE AXIS QUANTITATIVE TRIGONOMETRIC ACCELERATION AND DECELERATION MODE MOTION:..77
6.7 MULTI-AXIS MOTION 79
Chapter 7 Interpolation Motion Control......................................83
7.1 TWO-AXIS LINEAR INTERPOLATION (CONSTANT SPEED) 83
7.2 TWO-AXIS LINEAR INTERPOLATION (ACCELERATION / DECELERATION) 86
7.3 2D ARC INTERPOLATION (ACCELERATION/DECELERATION).............89
7.4 3D ARC INTERPOLATION (ACCELERATION/DECELERATION).............92
Chapter 8 Track Motion Control 94
8.1 CACHE INTERPOLATION 95
Chapter 9 Universal Digital I/O 99
9.1 INPUT PORT DEFINITION: SEE HARDWARE CHAPTER 2 ELECTRICAL CONNECTION FOR SPECIFIC DEFINITION 99
9.2 OUTPUT PORT DEFINITION: SEE HARDWARE CHAPTER 2 ELECTRICAL CONNECTION FOR SPECIFIC DEFINITION 99
9.3 OUTPUT PORT:..99
9.4 INPUT PORT:..100
Chapter 10 Auxiliary Control...101

11.1 POSITION LOCKING:......101
Chapter 11 List of ADT8949 Basic Library Functions 103
Chapter 12 Detailed Description of ADT8949 Basic Library Functions 112
12.1 BASIC PARAMETER SETTING......112
12.2 RESET MOTION CARD......120
12.3 DRIVING STATUS CHECKING......121
12.4 MOTION PARAMETER SETTING 122
12.5 PARAMETER CHECKING......132
12.6 DRIVING 136
12.7 SWITCH QUANTITY 145
12.8 FIFO OPERATION OUTPUT 147
12.9 HANDWHEEL FUNCTION 150
12.10 DA OUTPUT......151
13.11 HOMING MODULE 151
Chapter 13 Troubleshooting 154
13.1 MOTION CONTROL CARD DETECTIONS FAILS......154
13.2 MOTOR RUNNING ABNORMAL 155
13.3 SWITCH INPUT ABNORMAL 157
13.3 SWITCH OUTPUT ABNORMAL 158
13.4 ENCODER ABNORMAL......159

Overview

Chapter 1 Product Introduction

1.1 Description:

ADT-8949 multi-axis motion control card is one of the high-performance four-axis motion control cards of Adtech based on PCI bus. One system supports up to 10 control cards and controls 40 channels of servo / stepper motor. It features T-shaped, S-shaped and E-shaped acceleration and deceleration, point and track motion planning, electronic gear, electronic cam synchronized motion planning, linear interpolation, arc interpolation planning, splines, follow and other functions. It is especially suitable for occasions that have high-speed and high-precision position control requirement, and is widely used in testing, semiconductor packaging, mechanical arm, dispensing, packaging, engraving, and PCB processing.

1.2 Hardware specifications:

- 32-bit PCI bus, plug and play.
- Controllable axes: Four-axis pulse control.
- Maximum pulse output frequency 5MHz.
- Four-axis encoder feedback, frequency up to 4MHz, A/B phase difference pulse input and upper/lower pulse input are available, 32-bit count, 4 ratios.
- Support hardware serial number, allow third-party encryption.
- DSP + FPGA chip dedicated motion technology, provide high-speed and high-performance track smoothing and speed optimization.
- Pulse output types: Pulse + direction (PUL+DIR) or double pulse (CW + CCW).
- 36 channels digital inputs, all optically isolated.
- 32 channels open collector output.
- 2 channels DA output.

1.3 Control functions:

- Hardware cache: large capacity multi-axis hardware interpolation cache, store up to 10,000 interpolation instructions, continuous small line segments and large cache, used for multi-segment continuous track applications, such as engraving machine or cutting machine, so that the discrete data of CAM can be restored to the processing model.
- Speed preview: small line segment pretreatment, speed adaptive model, ensure high speed under high precision, automatic speed optimization, used for milling machine, tooling and other applications requiring high precision control, making the machine run smoothly from speed planning, and track error can be controlled.
- Multi-axis linkage: each axis moves or stops independently in accordance with the set speed and target position, and the driving speed can be changed in real time.
- Any 2-4 axis linear interpolation, 1-4 axis hardware cache interpolation, 2-3 axis hardware arc interpolation.
- 3D arc interpolation (spherical interpolation): achieve arc in any spatial plane and spherical arc. 3D hardware-level arc interpolation, support for hardware buffer interpolation, only occupies four data segments, and arc approximation accuracy is determined by interpolation speed dynamically to avoid discrete error of small segment approximation and contradictions of difficult speed tradeoff.
- Various acceleration and deceleration modes: T-shaped, S-shaped, E-exponential and C-cosine, support asymmetric acceleration and deceleration, smooth running, quiet motor
- NURBS interpolation: Compared to the traditional linear interpolation and arc interpolation, achieve curve and surface track;

NURBS interpolation can express various curve and surface tracks more accurately to improve curve and surface control accuracy.

- Synchronous dual drive: Two-axis pulse strict synchronization for dual-drive control in gantry structure.
- Position locking: In locking mode, when the locking signal is triggered, the hardware locks the logic location and actual position of the rising edge and falling edge, and can be used for quick homing, position measurement and other applications.
- Hardware cache IO event: In hardware cache function, not only the motion instructions but also IO output action and pulse generator can be cached. After the set axis of pulse generator reaches the specified location, the specified output port flips the level, and the flip times and frequency can be set. IO output cache can easily achieve precise position comparison output.
- Signal filtering: 15 filter levels can be set for input point to increase the anti-jamming capability of input signal
- Simultaneous control of multiple processes: You can open two programs to control one card. (A single monitoring program run simultaneously with the execution program; the execution program doesn’t need to switch a lot of time for display, which makes the display more real-time.
- Various homing modes: Before precise motion, it is necessary to set the home of mechanical coordinates, usually perform mechanical homing; the action is to reset. The system provides a variety of homing modes to facilitate efficient homing.
- Hardware upgrade: Upgrade dynamic library to complete hardware upgrade, which facilitate hardware system upgrade and client features customization.

1.4 Software support:

Operating system: DOS, WINDOWS95/98/NT/2000/XP, WINCE, WIN7

Programming environment: C/BC++/VC/VB/C#/C++Builder/ Delphi/ LabVIEW/ EVC

Application examples of open-DOS and Windows

1.5 Applications:

- Machine vision, automatic test equipment, AOI;
- Biological, medical automatic sampling equipment;
- Cutting equipment: diamond cutter, sponge cutting machine;
- Dispensing industry;
- Semiconductor packaging industry: Bonder;
- Advertising industry: CNC bending machine;
- Packaging and printing equipment: printing machine, pad printer;
- Engraving equipment;
- Industrial robot equipment;
- PCB processing, SMT and other industries;

1.6 ADT-8949 series features list:

√ Yes, - No, ***** optional

No.	Function Name	ADT-8949A1 (Universal)	ADT-8949B1 (Track)	ADT-8949G1 (Enhanced)
1	Pulse output mode: Pulse + direction, pulse + pulse	√	√	√
2	4 channels incremental encoder input	√	√	√
3	2 channels DA (-10V~+10V)	√	√	√
4	36 channels digital signal input	√	√	√
5	32 channels digital signal output	√	√	√
6	Home signal input	√	√	√
7	Limit signal input	√	√	√
8	Hardware emergency stop input	√	√	√
9	Drive alarm signal input	√	√	√
10	Drive enable signal output	√	√	√
11	Driving reset signal output	√	√	√

12	Multi-axis linkage (point motion)	√	√	√
13	2-4 axis linear interpolation	√	√	√
14	2-axis hardware arc interpolation	√	√	√
15	3-axis hardware arc interpolation	-	√	√
16	1-4 axis hardware cache interpolation	√	√	√
17	Hardware cache IO event	√	√	√
18	Speed preview	-	√	√
19	Various acceleration and deceleration modes (T, S, E, C)	√	√	√
20	Asymmetric acceleration and deceleration	-	√	√
21	Online changing of drive speed	√	√	√

22	Position locking	√	√	√
23	Various homing mode	√	√	√
24	Signal filtering	√	√	√
25	Multiple processes simultaneously control	√	√	√
26	NURBS interpolation	-	-	√
27	Synchronous dual-drive (semi-closed)	-	-	√
28	Hardware upgrade	-	-	√

Hardware

Chapter 1 Hardware Installation

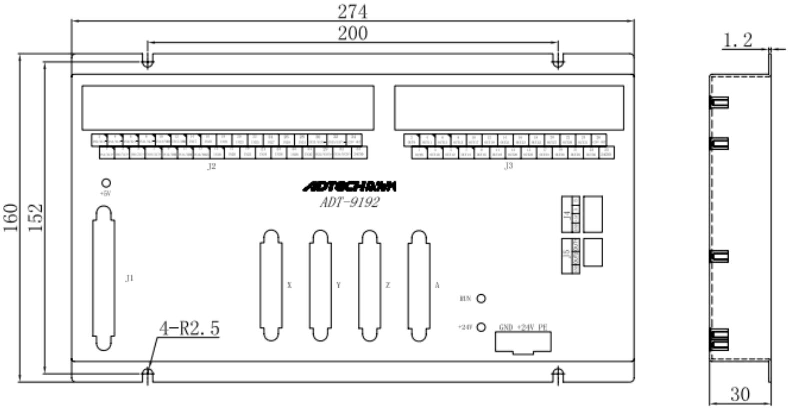
1.1 Packing list:

No.	Name	Description	Qty.
1	User's Manual	Instructions	1
2	User CD	SDK, examples and other information	1

		package	
3	ADT-8949	Four-axis motion card	1
4	ADT-9192	Terminal box	1
5	ADT-D62GG	Data cable	1

1.2 Installation size:

The appearance and installation dimensions of ADT-9192 are as follows:



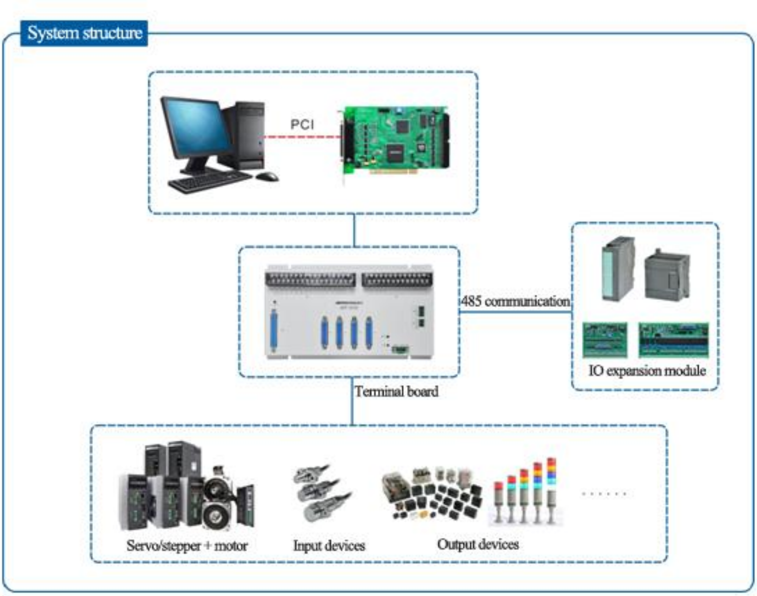
1.3 Installation steps:

1. Turn off the PC (Note: Turn off the master switch for ATX power supply)
2. Open the rear cover of PC case
3. Select an unoccupied PCI slot, and insert ADT-8949.
4. Ensure that the gold finger of ADT-8949 is completely inserted into the PCI slot, and tighten the screws.
5. Connect one end of the D62GG cable to J4 interface of the control card and connect the other end to J1 of ADT-9192

board.

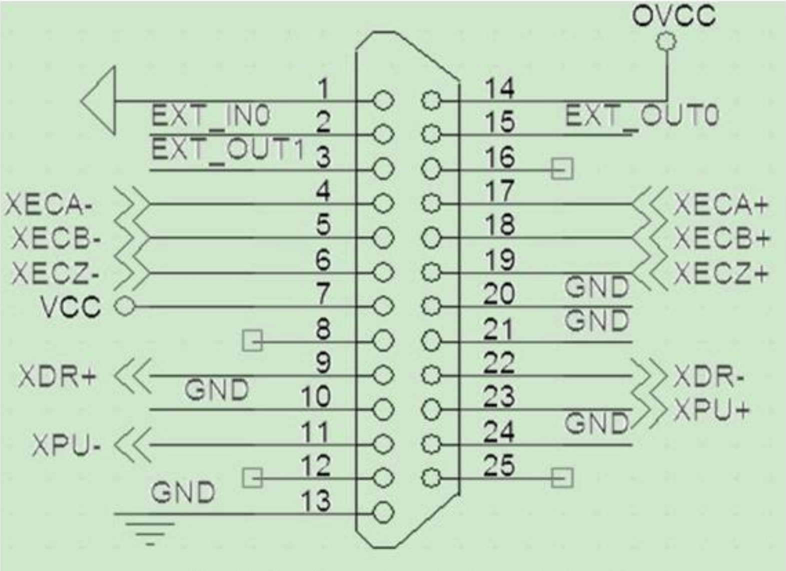
Chapter 2 Electrical Connection

2.1 Wiring diagram:



2.2 25-pin port I/O signal definition:

Four DB terminals correspond to four axes (XYZA). Below is the example of X axis. Other axes are similar.



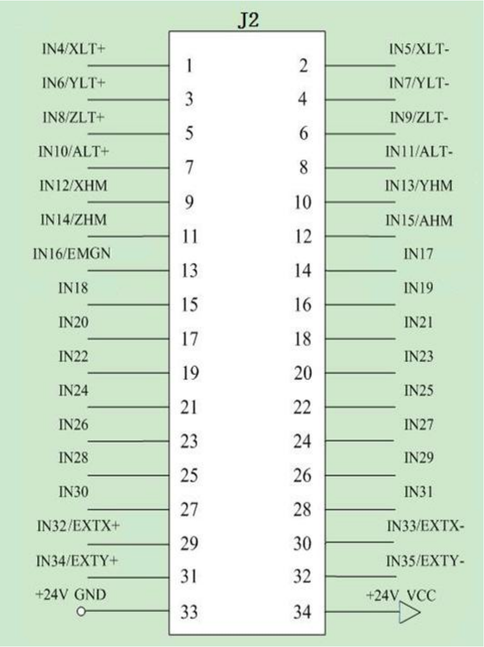
X-axis direction, pulse, signal encoder interface

No.	Signal name	Definition
1	◁	External 24V power grounding
2	**EXT_IN0**	External input, low level active, **does not support two-wire sensor connection.**
3	**EXT_OUT1**	Output control signal, opto-isolated output, low voltage level active
4	**XECA-**	X-axis encoder A phase input negative, can be used as an general input point, corresponding to sample program input point **IN36**, Y-axis, Z-axis and A-axis correspond to **IN37, IN38, and IN39** respectively. **When used as general input point, refer to 3.9 digital input connection - encoder signal as general input for wiring.**

5	**XECB-**	X-axis encoder B phase input negative, can be used as an general input point, corresponding to sample program input point **IN40**, Y-axis, Z-axis and A-axis correspond to **IN41, IN42, and IN43** respectively. **When used as general input point, refer to 3.9 digital input connection - encoder signal as general input for wiring.**
6	**XECZ-**	X-axis encoder Z phase input negative, can be used as an general input point, X-axis corresponds to **XSTOP1 (IN44)**, Y-axis corresponds to **YSTOP1 (IN45)**, Z-axis corresponds to **ZSTOP1 (IN46)** and A-axis corresponds to **ASTOP1 (IN47)**. **When used as general input point, refer to 3.9 digital input connection - encoder signal as general input for wiring.**
7	**VCC**	+5V power output (can’t be connected to external power supply)
8	**NC**	
9	**XDR+**	X axis direction positive signal
10	**GND**	5V power grounding
11	**XPU-**	X axis pulse negative signal
12	**NC**	
13	**GND**	5V power grounding
14	**OVCC**	+24V power output (can’t be connected to external 24V+)
15	**EXT_OUT0**	Output control signal, opto-isolated output, low voltage level active
16	**NC**	
17	**XECA+**	X axis encoder phase A input positive

18	**XECB+**	X axis encoder phase B input positive
19	**XECZ+**	X axis encoder phase Z input positive
20	**GND**	5V power grounding
21	**GND**	5V power grounding
22	**XDR-**	X axis direction negative signal
23	**XPU+**	X axis pulse positive signal
24	**GND**	5V power grounding
25	**NC**	

2.3 Signal definition of J2 interface



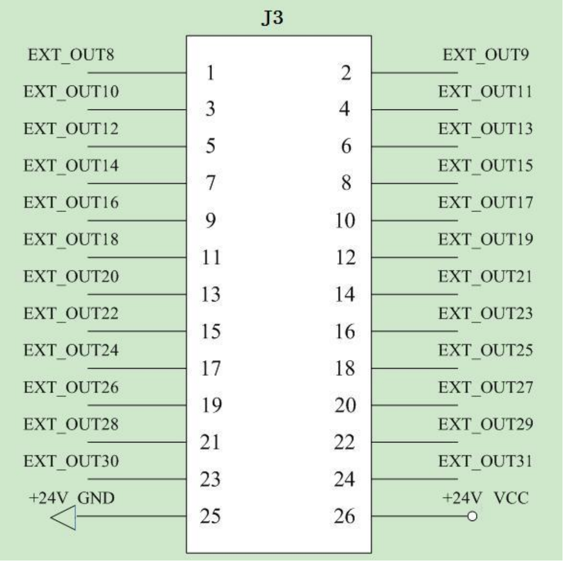
34-bit double dislocation terminal wiring is defined as below

1	**IN4/XLT+**	X positive limit signal, can be used as general input

2	**IN5/XLT-**	X negative limit signal, can be used as general input
3	**IN6/YLT+**	Y positive limit signal, can be used as general input
4	**IN7/YLT-**	Y negative limit signal, can be used as general input
5	**IN8/ZLT+**	Z positive limit signal, can be used as general input
6	**IN9/ZLT-**	Z negative limit signal, can be used as general input
7	**IN10/ALT+**	A positive limit signal, can be used as general input
8	**IN11/ALT-**	A negative limit signal, can be used as general input
9	**IN12/XHM**	X home signal (STOP0), can be used as general input
10	**IN13/YHM**	Y home signal (STOP0), can be used as general input
11	**IN14/ZHM**	Z home signal (STOP0), can be used as general input
12	**IN15/AHM**	A home signal (STOP0), can be used as general input
13	**IN16/EMGN**	Emergency stop signal, can be used as general input
14	**IN17**	General input
15	**IN18**	General input
16	**IN19**	General input
17	**IN20**	General input
18	**IN21**	General input
19	**IN22**	General input

20	**IN23**	General input
21	**IN24**	General input
22	**IN25**	General input
23	**IN26**	General input
24	**IN27**	General input
25	**IN28**	General input
26	**IN29**	General input
27	**IN30**	General input
28	**IN31**	General input
29	**IN32/EXTX+**	X manual forward rotation signal, can be used as general input
30	**IN33/EXTX-**	X manual reverse rotation signal, can be used as general input
31	**IN34/EXTY+**	Y manual forward rotation signal, can be used as general input
32	**IN35/EXTY-**	Y manual reverse rotation signal, can be used as general input
33	**EXT_+24V GND**	24V power grounding
34	**EXT_+24V VCC**	+24V power output (can’t be connected to external 24V+)

2.4 Signal definition of J3 interface



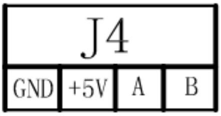
26-bit double dislocation terminal wiring is defined as below

Wire No.	Name	Function
1	**EXT_OUT8**	Output control signal, opto-isolated output, low voltage level active
2	**EXT_OUT9**	Output control signal, opto-isolated output, low voltage level active
3	**EXT_OUT10**	Output control signal, opto-isolated output, low voltage level active
4	**EXT_OUT11**	Output control signal, opto-isolated output, low voltage level active
5	**EXT_OUT12**	Output control signal, opto-isolated output, low voltage level active
6	**EXT_OUT13**	Output control signal, opto-isolated output, low

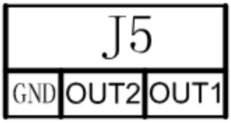
		voltage level active
7	**EXT_OUT14**	Output control signal, opto-isolated output, low voltage level active
8	**EXT_OUT15**	Output control signal, opto-isolated output, low voltage level active
9	**EXT_OUT16**	Output control signal, opto-isolated output, low voltage level active
10	**EXT_OUT17**	Output control signal, opto-isolated output, low voltage level active
11	**EXT_OUT18**	Output control signal, opto-isolated output, low voltage level active
12	**EXT_OUT19**	Output control signal, opto-isolated output, low voltage level active
13	**EXT_OUT20**	Output control signal, opto-isolated output, low voltage level active
14	**EXT_OUT21**	Output control signal, opto-isolated output, low voltage level active
15	**EXT_OUT22**	Output control signal, opto-isolated output, low voltage level active
16	**EXT_OUT23**	Output control signal, opto-isolated output, low voltage level active
17	**EXT_OUT24**	Output control signal, opto-isolated output, low voltage level active
18	**EXT_OUT25**	Output control signal, opto-isolated output, low voltage level active
19	**EXT_OUT26**	Output control signal, opto-isolated output, low voltage level active
20	**EXT_OUT27**	Output control signal, opto-isolated output, low voltage level active
21	**EXT_OUT28**	Output control signal, opto-isolated output, low

		voltage level active
22	**EXT_OUT29**	Output control signal, opto-isolated output, low voltage level active
23	**EXT_OUT30**	Output control signal, opto-isolated output, low voltage level active
24	**EXT_OUT31**	Output control signal, opto-isolated output, low voltage level active
25	**EXT_+24V GND**	24V power grounding
26	**EXT_+24V VCC**	+24V power output (can’t be connected to external 24V+)

2.5 J4 signal definition, reserved, no specific function



2.6 J5 signal definition



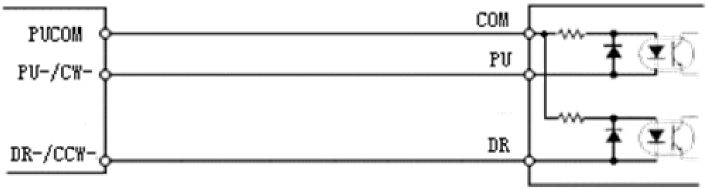
Wire No.	Name	Function
1	**OUT1**	DA1 output, 0~10V output
2	**OUT2**	DA2 output, 0~10V output
3	**GND**	Reference grounding

2.7 Connecting pulse/direction output signal:

Pulse output is differential output

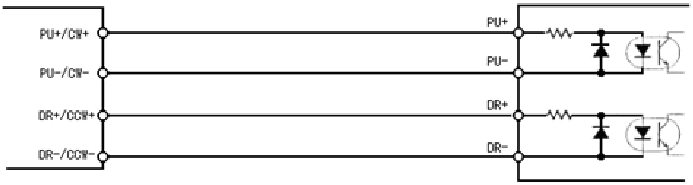
Can be easily connected to the stepper / servo drives

Below is the connection that pulse anode and direction anode have been connected.

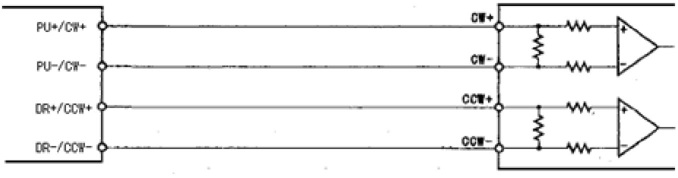


Step motor drive

Below is independent connection of pulse and direction signal. Differential connection is recommended due to strong anti-interference.



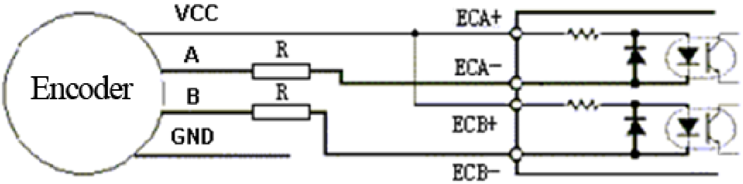
Step motor drive



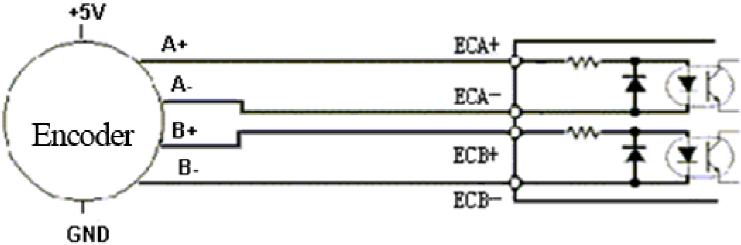
Servo motor drive

Note: Refer to Appendix A for the wiring diagram of step motor drive, common servo motor drive and terminal board.

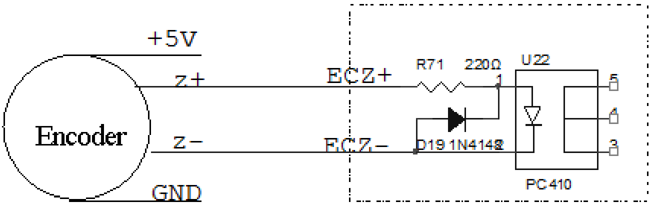
2.8 Connecting encoder input signal:



Open Collect Output Encoder Wiring Diagram
For +5V power, R can be omitted; for +12V power, R=1kΩ; for +24V power, R=2kΩ



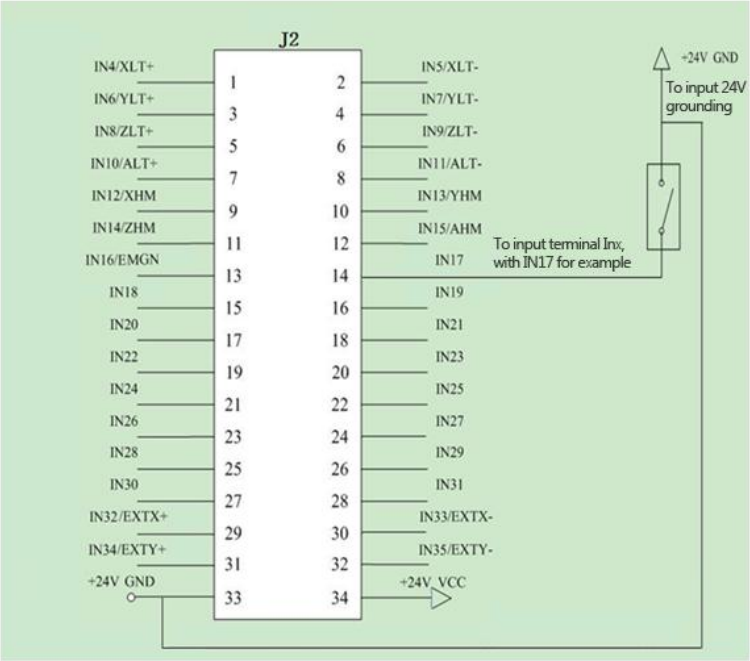
Line Driver Output Encoder Wiring Diagram



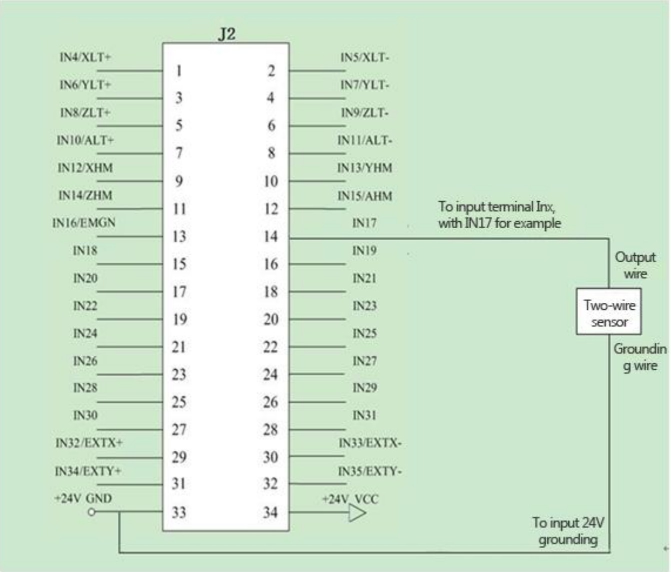
Encoder Z phase connection

2.9 Connecting digital input:

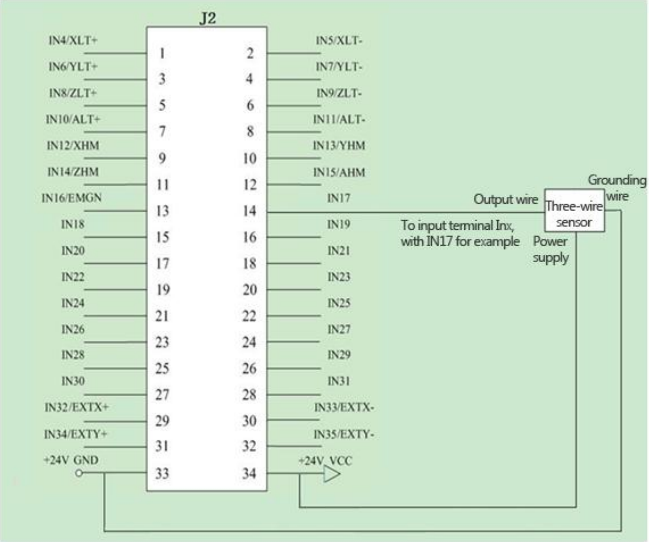
The following figure shows the mechanical switch connection with IN17 for example (regardless of switch polarity):



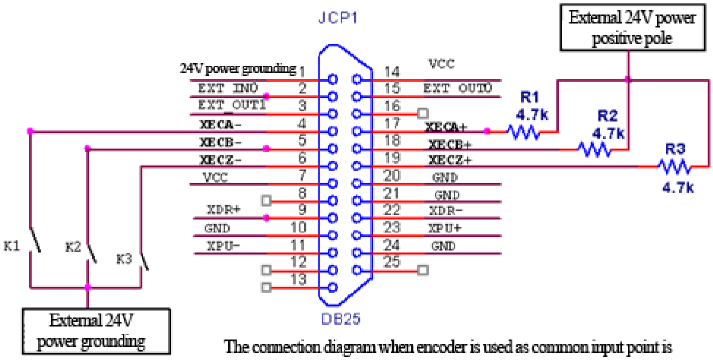
The following figure shows the two-wire sensor connection (note the polarity of two-wire sensor; blue is ground wire, and brown is output wire). **Note: The input points IN0~IN3 of four-axis 25-pin DB connector do not support two-wire sensor connection.**



Below is three-wire sensor connection (pay attention to the polarity of three-wire sensor; refer to corresponding manual):



The connection diagram of encoder AB phase signal as normal input point (K1, K2, K3 in mechanical switch connection):



The connection diagram when encoder is used as common input point is described with X-axis as the example

Note: After ADT-9192 is connected to the power, the power of wiring terminal J2 is synchronized to 24V, XECA+, XECB+ and XECC+ are 4.7K resistance respectively, and then connect to the positive pole of the 24V power supply of J2 terminal, XECA-, XECB- and XECC- correspond the corresponding input point. Other axes are similar.

The relationship between encoder as common input point and sample program input point

XECA- corresponds to IN36, XECB- corresponds to IN40, and XECZ- corresponds to IN44

YECA- corresponds to IN37, XECB- corresponds to IN41, and XECZ- corresponds to IN45

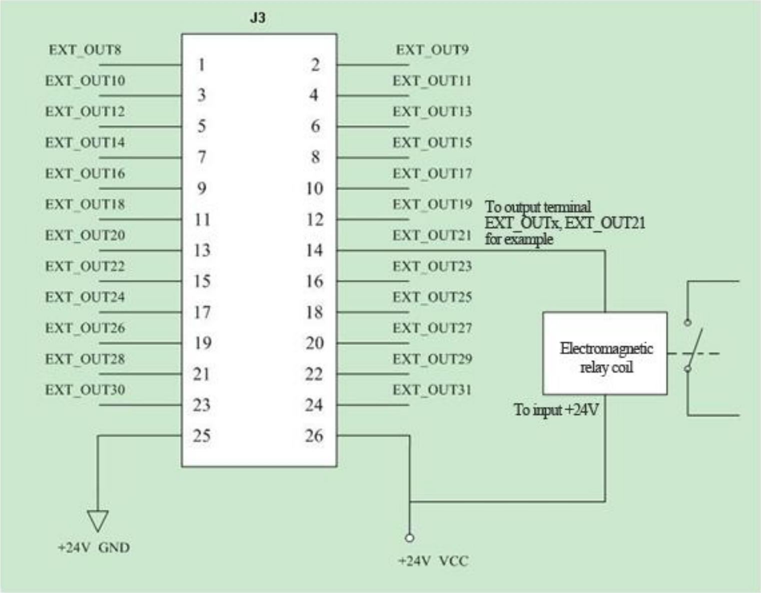
ZECA- corresponds to IN38, XECB- corresponds to IN42, and XECZ- corresponds to IN46

AECA- corresponds to IN39, XECB- corresponds to IN43, and XECZ- corresponds to IN47

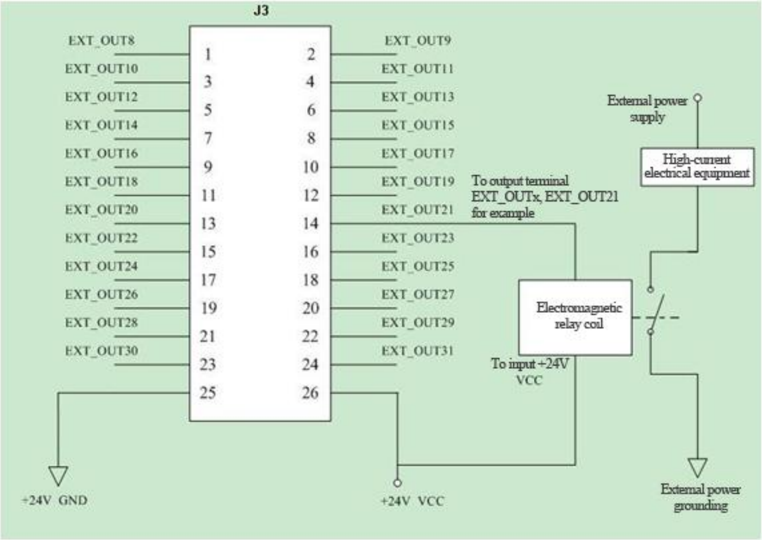
2.10 Connecting digital output:

All common output points of the board are open drain output, and the drive current at each point is within 1A. Before use, consider if the drive current of the output point is adequate; if not, expand with external relay and connect freewheeling protection diode.

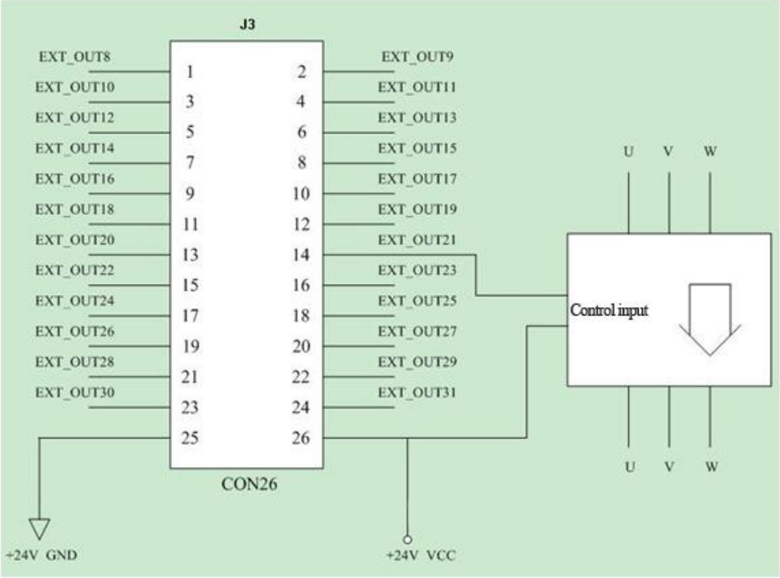
Below is the connection of output point driven general electromagnetic relay (regardless of relay coil polarity unless otherwise specified):



The connection of electromagnetic relay expansion flow output point (only normally open relay can be used for expansion flow)



Solid state relay connection (pay attention to the polarity of solid state relay control terminal):



The figure shows the connection of three-phase solid state relay. Two-phase and single-phase solid state relays have the similar connection.

Chapter 3 Software Installation

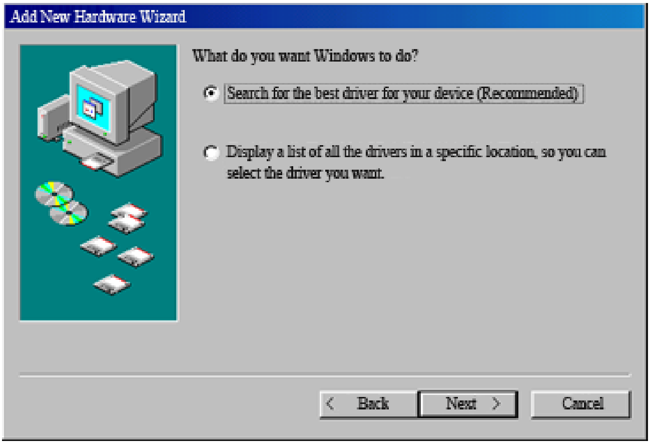
To use ADT-8949 card properly on Win95 / Win98 / NT / Win2000 / WinXP, you need to install the drivers. No drivers are needed in DOS. Below is the example on Win2000 and WinXP. Refer to this section for other systems.

The drivers of the control card are located on the CD "Development kits \ Drivers \ Control Card Drivers", and the file name is 8949.INF.

3.1 Installing drivers on Win98:

Below is an example of driver installation on Win98 Professional Chinese version. Other versions are similar to Win98.

After installing the ADT-8949 card into a PCI slot on the computer, log in as administrator, and the computer should find the new hardware and pop up the following screen:



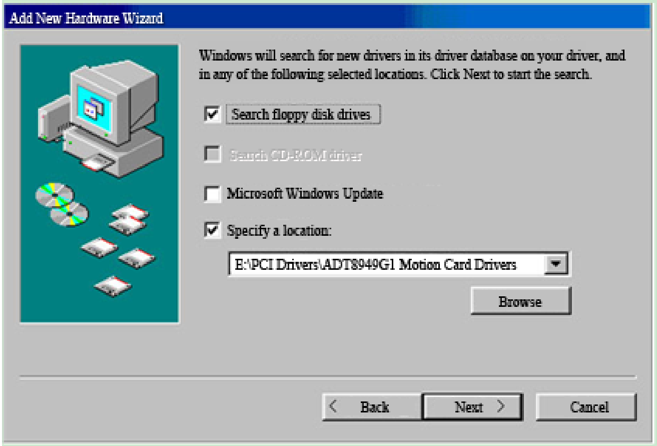
Add New Hardware Wizard

What do you want Windows to do?

● Search for the best driver for your device (Recommended)

○ Display a list of all the drivers in a specific location, so you can select the driver you want.

Click “Next” to pop up the following screen



Add New Hardware Wizard

Windows will search for new drivers in its driver database on your driver, and in any of the following selected locations. Click Next to start the search.

☑ Search floppy disk drives

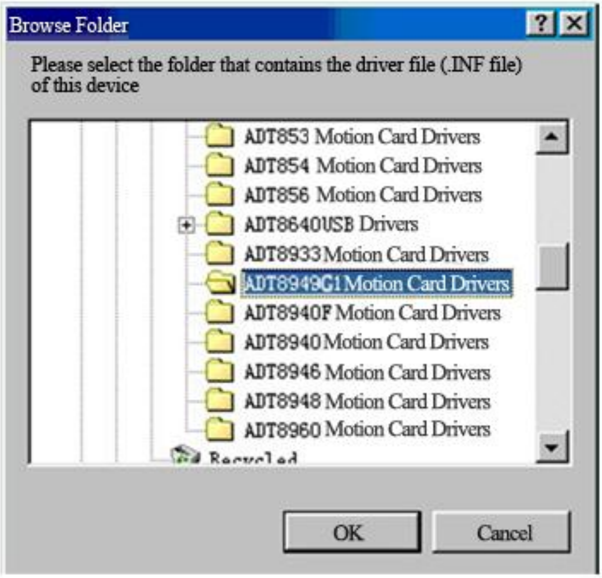
□ Search CD-ROM driver

□ Microsoft Windows Update

☑ Specify a location:

E:\PCI Drivers\ADT8949G1 Motion Card Drivers

Click the "Browse" button, select CD "Development Kits \ Drivers \ Motion Card Driver" and find the file 8949.inf, and click "OK" to pop up the following interface



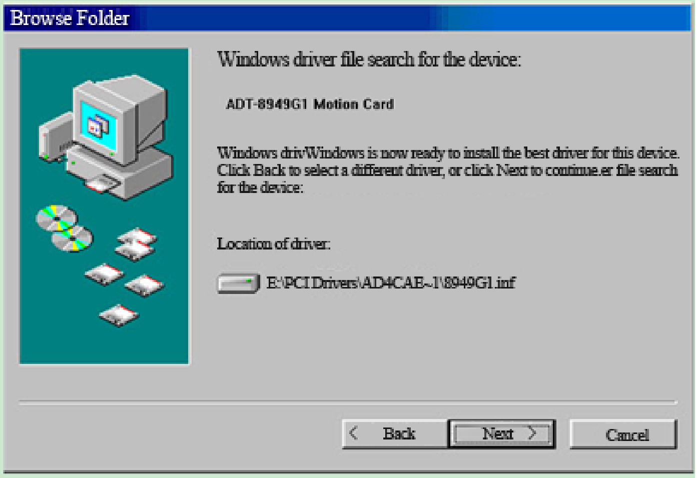
Browse Folder

Please select the folder that contains the driver file (.INF file) of this device

- ADT853 Motion Card Drivers
- ADT854 Motion Card Drivers
- ADT856 Motion Card Drivers
- ADT8640USB Drivers
- ADT8933Motion Card Drivers
- ADT8949G1 Motion Card Drivers
- ADT8940F Motion Card Drivers
- ADT8940Motion Card Drivers
- ADT8946 Motion Card Drivers
- ADT8948 Motion Card Drivers
- ADT8960 Motion Card Drivers
- Recycled

Click “OK” to pop up the following screen:

Click "Next" to pop up the following screen



Browse Folder

Windows driver file search for the device:

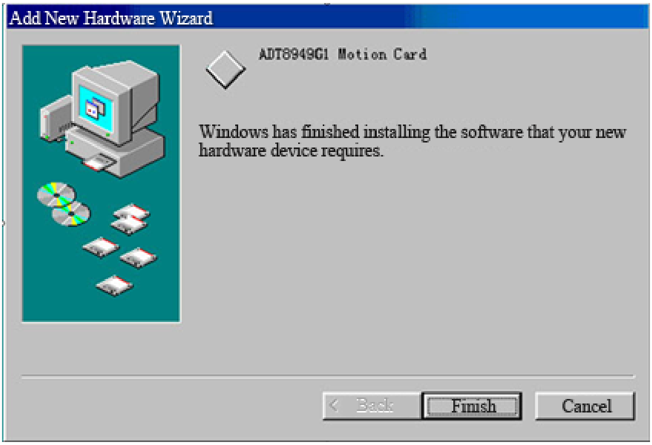
ADT-8949G1 Motion Card

Windows drivWindows is now ready to install the best driver for this device. Click Back to select a different driver, or click Next to continue.er file search for the device:

Location of driver:

E:\PCI Drivers\AD4CAB~1\8949G1.inf

Click "Next" to pop up the following screen



Add New Hardware Wizard

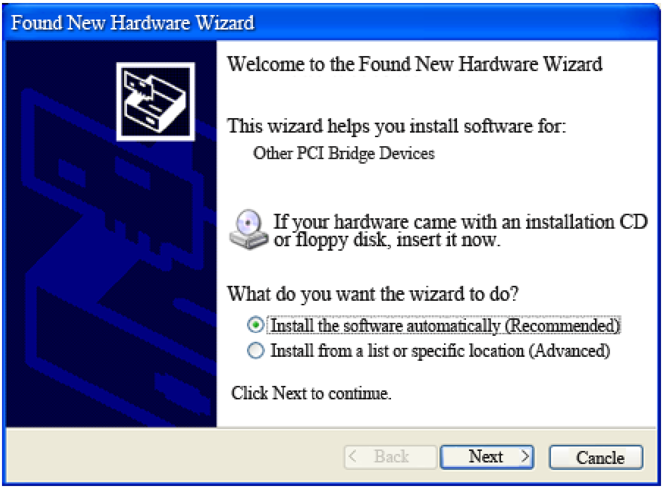
ADT8949G1 Motion Card

Windows has finished installing the software that your new hardware device requires.

Click "Finish" to finish the installation of ADT-8949 card

3.2 Installing drivers on WinXP:

Below is the example of installing drivers on WinXP. Other systems are similar.



Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

This wizard helps you install software for:

Other PCI Bridge Devices

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

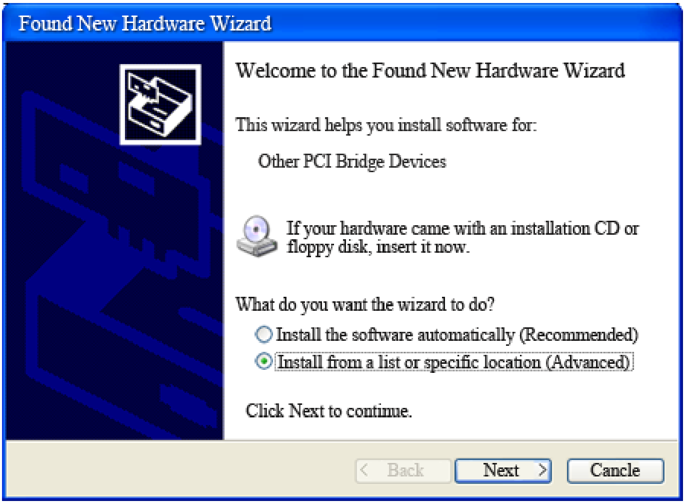
What do you want the wizard to do?

● Install the software automatically (Recommended)

○ Install from a list or specific location (Advanced)

Click Next to continue.

Select in above figure to pop up the following screen



Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

This wizard helps you install software for:

Other PCI Bridge Devices

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

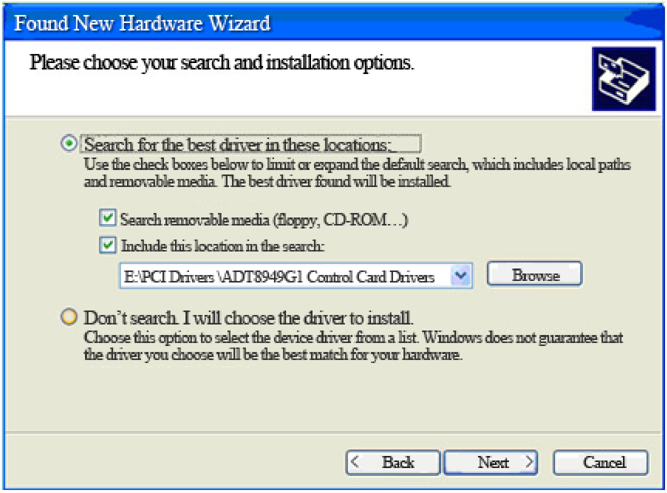
What do you want the wizard to do?

○ Install the software automatically (Recommended)

● Install from a list or specific location (Advanced)

Click Next to continue.

Select in above figure and click Next to pop up the following screen



Found New Hardware Wizard

Please choose your search and installation options.

● Search for the best driver in these locations.

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

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

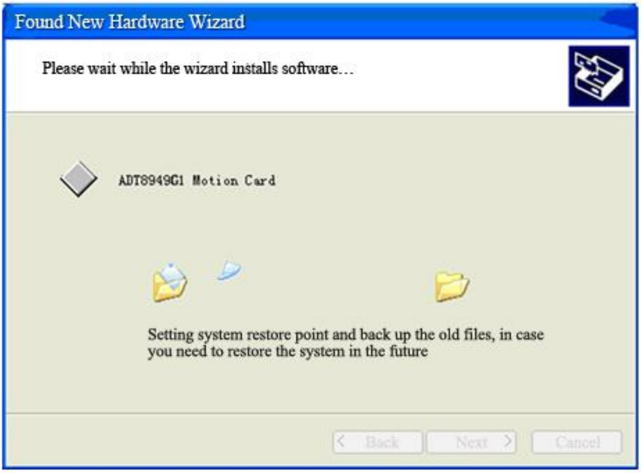
☑ Include this location in the search:

E:\PCI Drivers \ADT8949G1 Control Card Drivers

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

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

Click “Browse”, select CD “Development Kits\Drivers\Motion Card Drivers” to locate the file 8949.INF, and click Next to pop up the following screen

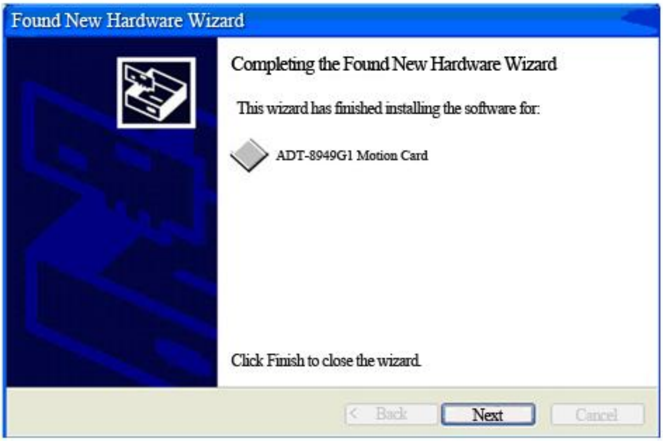


Found New Hardware Wizard

Please wait while the wizard installs software...

ADT8949G1 Motion Card

Setting system restore point and back up the old files, in case you need to restore the system in the future



Found New Hardware Wizard

Completing the Found New Hardware Wizard

This wizard has finished installing the software for:

ADT-8949G1 Motion Card

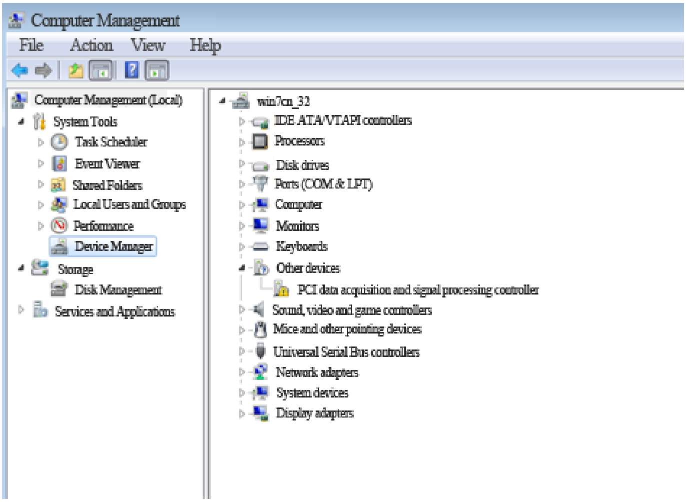
Click Finish to close the wizard.

Click "Finish" to finish the installation of ADT-8949 card

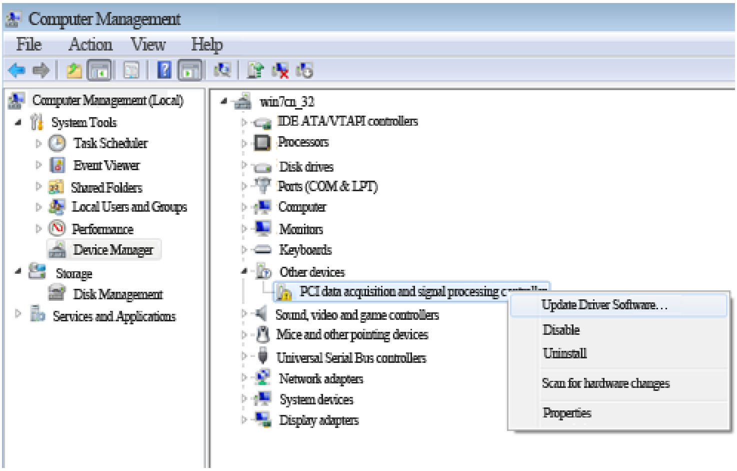
3.3 Installing drivers on Win7

Install the drivers on Win7 system as follows:

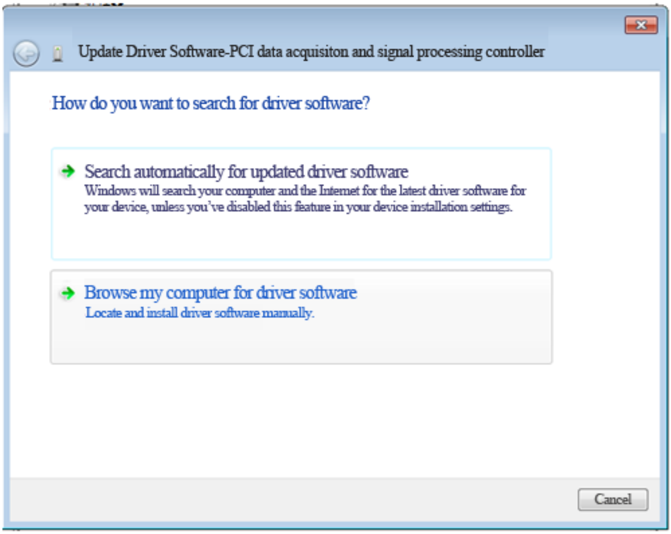
1. Insert the control card into PCI slot, right click "My Computer" and select "Properties" to enter Device Manager, as shown below:



Expand "Other devices", select "PCI data acquisition and signal processing controller", and right click, as shown below:



2. In the popup dialog box, click "Update Driver Software" to show the following dialog box:

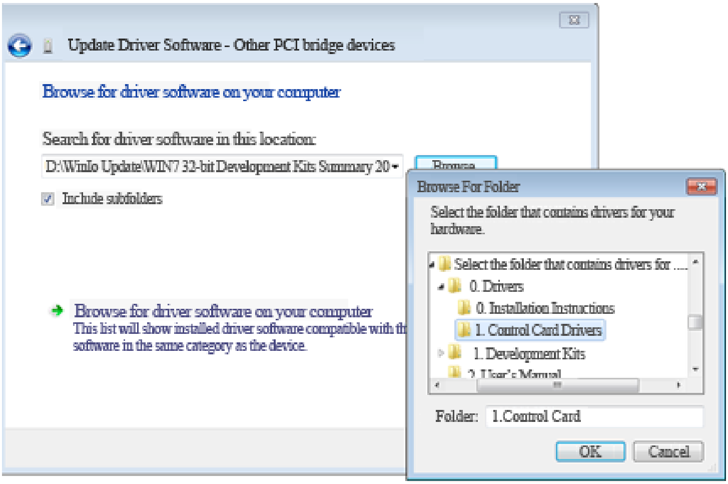

Update Driver Software-PCI data acquisiton and signal processing controller

How do you want to search for driver software?

Search automatically for updated driver software
Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.

Browse my computer for driver software
Locate and install driver software manually.

Select the option "Browse my computer for driver software", and then click the "Browse" button to specify the path for the driver, as shown below:



Update Driver Software - Other PCI bridge devices

Browse for driver software on your computer

Search for driver software in this location:

D:\WinIo Update\WIN7 32-bit Development Kits Summary 20

☑ Include subfolders

Browse for driver software on your computer
This list will show installed driver software compatible with th
software in the same category as the device.

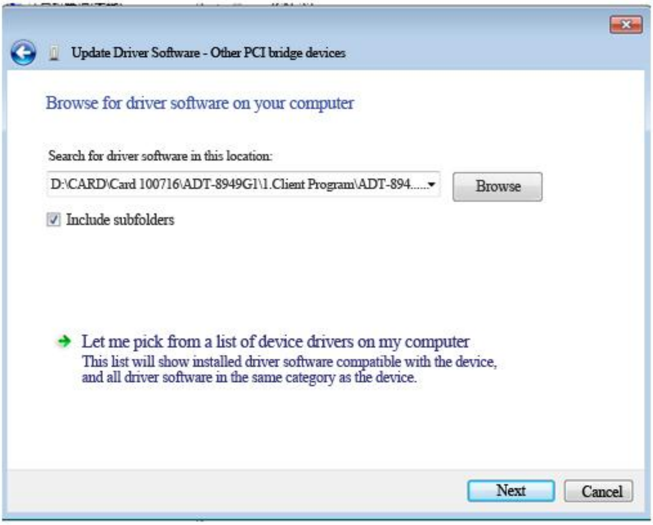
Browse For Folder

Select the folder that contains drivers for your hardware.

- Select the folder that contains drivers for ....
  - 0. Drivers
    - 0. Installation Instructions
    - 1. Control Card Drivers
  - 1. Development Kits
  - 2. User's Manual

Folder: 1.Control Card

3. Click "OK", and the following dialog box appears:



Update Driver Software - Other PCI bridge devices

Browse for driver software on your computer

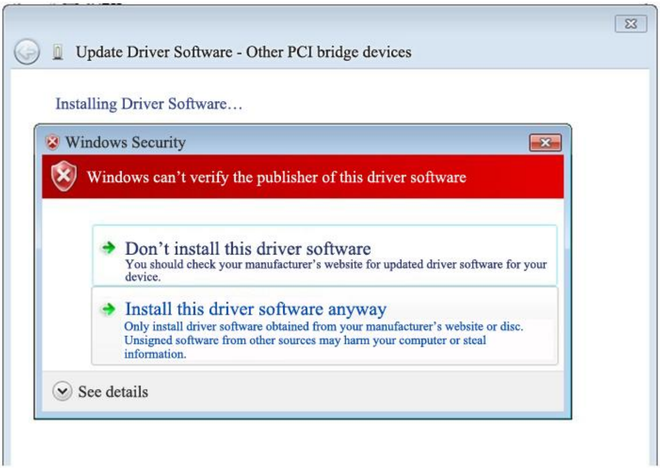
Search for driver software in this location:

D:\CARD\Card 100716\ADT-8949G1\1.Client Program\ADT-894.....

☑ Include subfolders

Let me pick from a list of device drivers on my computer
This list will show installed driver software compatible with the device,
and all driver software in the same category as the device.

4. Click "Next" to install the drivers, and the following interface appears:



Update Driver Software - Other PCI bridge devices

Installing Driver Software…

Windows Security

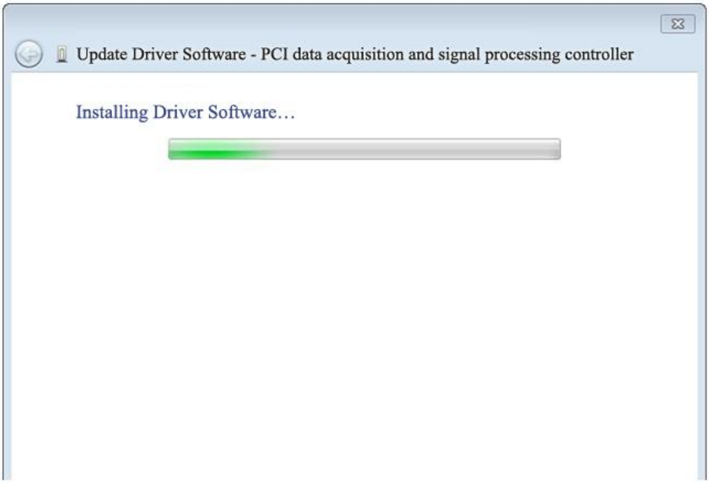
Windows can't verify the publisher of this driver software

Don't install this driver software
You should check your manufacturer's website for updated driver software for your device.

Install this driver software anyway
Only install driver software obtained from your manufacturer's website or disc. Unsigned software from other sources may harm your computer or steal information.

See details

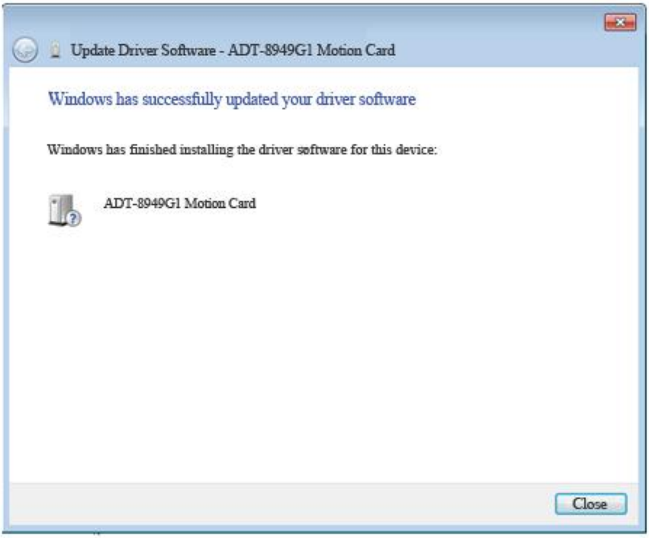
5. Select "Install this driver software", and the following interface appears



Update Driver Software - PCI data acquisition and signal processing controller

Installing Driver Software…

Wait to complete, and the following dialog box appears



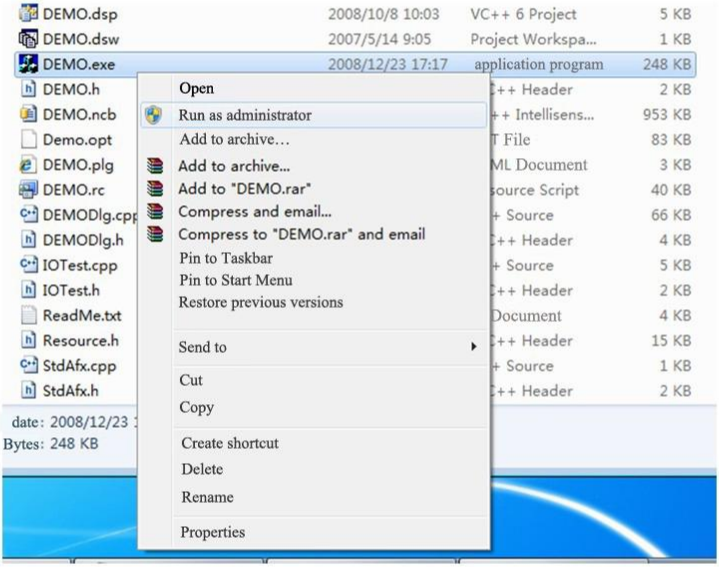
Update Driver Software - ADT-8949G1 Motion Card

Windows has successfully updated your driver software

Windows has finished installing the driver software for this device:

ADT-8949G1 Motion Card

The ADT-8949 card is installed. **Note: WIN7 system requires administrator privileges to load the PCI drivers. If the control card application is run for the first time, double-clicking will lead to the control card initialization failed. Therefore, when the drivers are installed, right click the control card application (e.g. VC demonstration program "DEMO.EXE") and select "Run as Administrator" (see below). You can double-click the application to run it normally later.**



DEMO.dsp	2008/10/8 10:03	VC++ 6 Project	5 KB
DEMO.dsw	2007/5/14 9:05	Project Workspa...	1 KB
DEMO.exe	2008/12/23 17:17	application program	248 KB
DEMO.h		++ Header	2 KB
DEMO.ncb		++ Intellisens...	953 KB
Demo.opt		T File	83 KB
DEMO.plg		ML Document	3 KB
DEMO.rc		ource Script	40 KB
DEMODlg.cp		+ Source	66 KB
DEMODlg.h		++ Header	4 KB
IOTest.cpp		+ Source	5 KB
IOTest.h		++ Header	2 KB
ReadMe.txt		Document	4 KB
Resource.h		++ Header	15 KB
StdAfx.cpp		+ Source	1 KB
StdAfx.h		++ Header	2 KB

date: 2008/12/23 1

Bytes: 248 KB

Chapter 4 Electrical Specifications

4.1 Switch input:

Channel: 36 channels, all optically isolated.

Input voltage: 12-24V

High voltage level > 4.5V

Low voltage level <1.0V

Isolation voltage: 2500V DC

4.2 Count input:

Channel: 4 channels AB phase encoder input, all optically isolated.

Maximum count frequency: 4MHz

Input voltage: 5-24V

High voltage level >4.5V

Low voltage level <1.0V

Isolation voltage: 2500V DC

4.3 Pulse output:

Channel: 4 pulses, four directions, all optically isolated.

Maximum pulse frequency: 5MHz

Output type: 5V differential output

Output: Pulse + direction or pulse + pulse

4.4 Switching output:

Output channels: 32 channels, all optically isolated.

Output type: NPN open collector 5-24VDC, maximum current of single output of common output port: 1A; maximum current of single output of DB terminal: 50mA.

4.5 Power output:

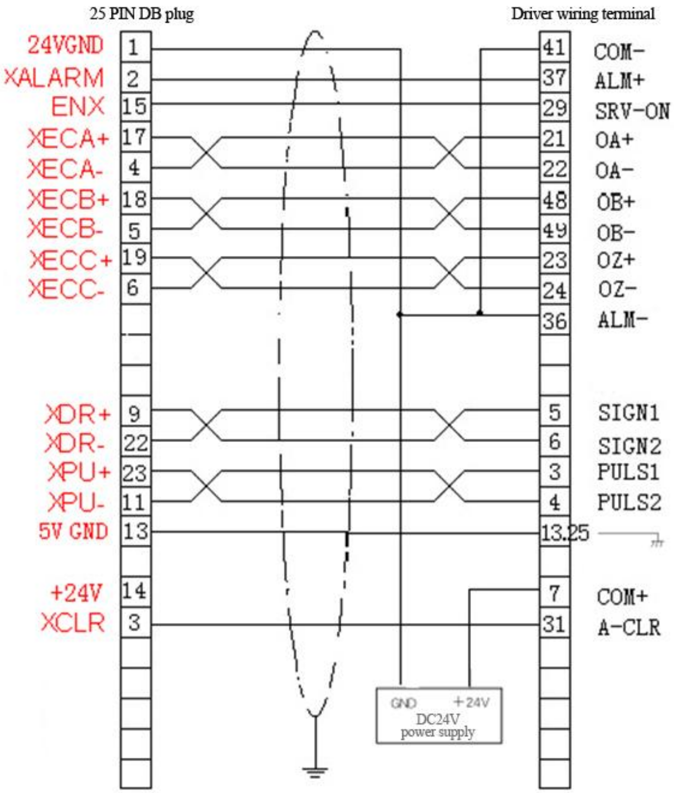
Output voltage: + 5V.

Output type: DC source, maximum current 500mA.

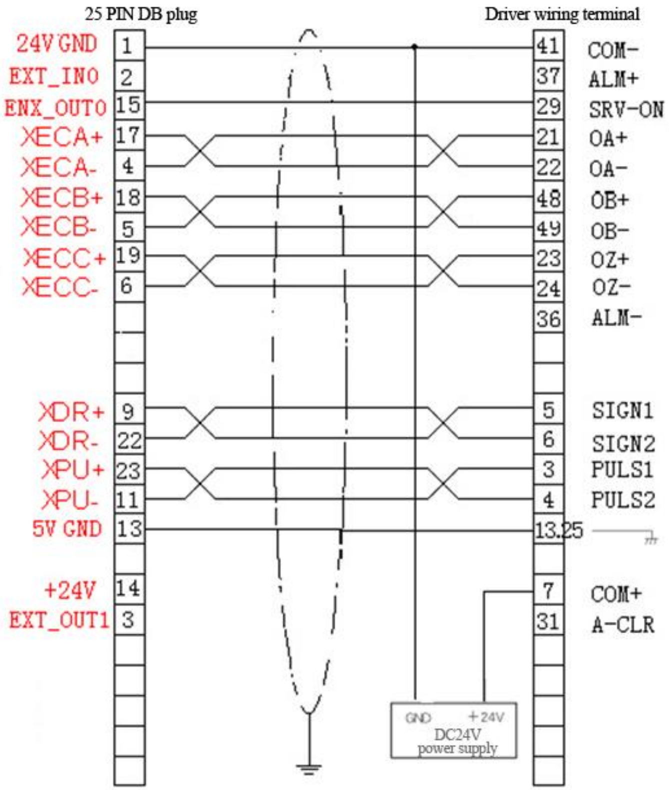
Chapter 5 Common Servo Wiring Diagrams

5.1 Panasonic A5 servo wiring diagram:

Below is the wiring of motion card connected to Panasonic A5 servo drive: use external power supply, external enable, including alarm signal.



Below is the wiring of motion card connected to Panasonic A5 servo drive: use external power supply, external enable, alarm information not specified.



Chapter 6 Working Environment

6.1 Operating temperature:

Operating temperature: 0℃~60℃

6.2 Storage temperature:

Storage temperature: -20℃~80℃

6.3 Operating humidity:

Operating humidity: 20%~95%

6.4 Storage humidity:

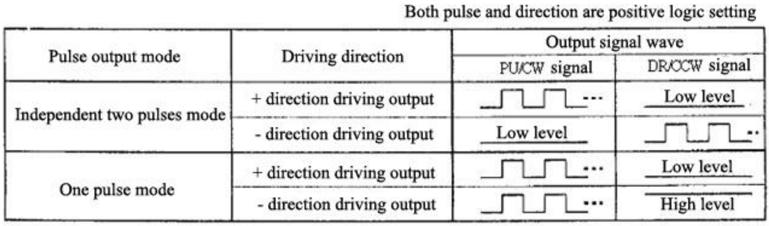
Storage humidity: 0%~95%

Software Programming

Chapter 1 Function Description

1.1 Pulse output mode:

Driving output pulse has the following two pulse output modes below. When independent two pulses mode is used in positive driving, PU/CW outputs driving pulse; in negative driving, DR/CCW outputs driving pulse; when one pulse mode is used, PU/CW outputs driving pulse and DR/CCW outputs direction signal.



Both pulse and direction are positive logic setting

Pulse output mode	Driving direction	Output signal wave	
		PU/CW signal	DR/CCW signal
Independent two pulses mode	+ direction driving output	⎍⎍···	Low level
	- direction driving output	Low level	⎍⎍··
One pulse mode	+ direction driving output	⎍⎍···	Low level
	- direction driving output	⎍⎍···	High level

1.2 Hardware limit signal:

Hardware limit signal (LMT+, LMT-) is the input signal that limits positive and negative driving pulse. It can be set to valid, invalid, high level or low level, and the positive and negative limits can be set be valid / invalid independently. When set to invalid, it can be used as normal input point.

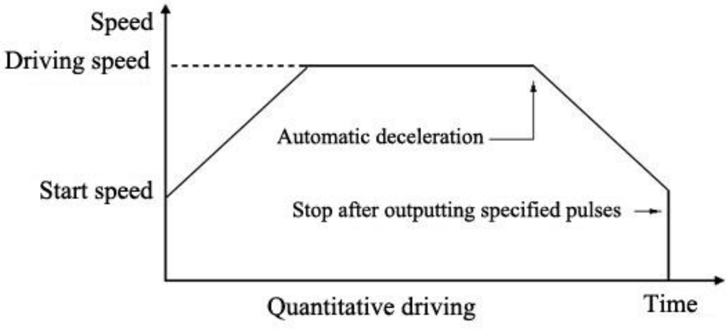
Hardware limit signal (STOP0, STOP1) can achieve input signal to stop driving of each axis. It can be set to valid, invalid, high level or low level. When set to invalid, it can be used as normal input point. In addition, STOP0 and STOP1 signals are only valid for the minimum interpolation axis in interpolation driving.

1.3 Quantitative driving:

Quantitative driving is to output specified quantity of pulse in constant speed or acceleration/deceleration. Use this function to move to specified position or perform specific action. Quantitative driving of acceleration/deceleration is shown below. When the remaining of output pulse is less than acceleration accumulated pulses, it starts accelerating, and the driving also stops after outputting specified pulses.

The following parameters should be set for the quantitative driving for acceleration/ deceleration:

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



Acceleration/deceleration quantitative driving usually starts automatic acceleration from the calculated deceleration point shown in the picture above; besides, manual deceleration is also possible. In the following cases, it is not possible to or can't calculate the automatic deceleration point, and thus manual calculation is required:

- Change speed frequently during linear acceleration/ deceleration quantitative driving
- Run arc interpolation and continuous interpolation in acceleration/deceleration

It is required to change to manual deceleration mode and set the deceleration point.

1.4 Continuous driving:

During continuous driving, output driving pulse continuously until high level stop command or external stop signal is valid. Use this function for home search, scanning operation and motor rotation control.

Two stop commands are available, one is deceleration stop and the other is immediate stop. Each axis has two external signals (STOPO, and STOP1) used for deceleration/ immediate stop. Each signal can set valid/invalid level. STOPO and STOP1 signals are deceleration stop in acceleration/deceleration driving and immediate stop in constant speed driving.

Home search action of continuous driving

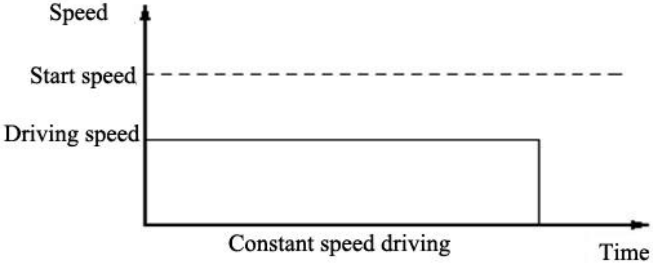
Set home approach signal, home signal and encoder Z phase signal to STOPO and STOP1. Set the valid/invalid and logical level of every signal in every axis. During high speed search, use acceleration/deceleration continuous driving, and decelerate to stop when the set valid signal is in activated level. During low speed search, use constant speed continuous driving, and immediately stop when the set valid signal is in activated level. For acceleration/deceleration continuous driving, all the parameters except output pulses must be same as quantitative driving.

1.5 Speed curve:

1.5.1 Constant speed driving

Constant speed driving will output driving pulse in constant speed. If the driving speed is lower than the start speed, there will be constant speed driving only instead of acceleration/deceleration driving. When use home search, encoder Z phase and similar signals, acceleration/deceleration driving isn't required if stop immediately after signal is searched; instead, the system runs low speed constant driving. The following parameters should be set for constant speed driving:

- Start speed SV
- Driving speed V



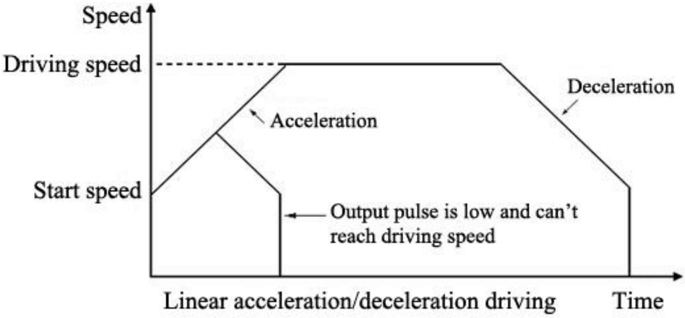
1.5.2 T-shaped linear acceleration/deceleration driving

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

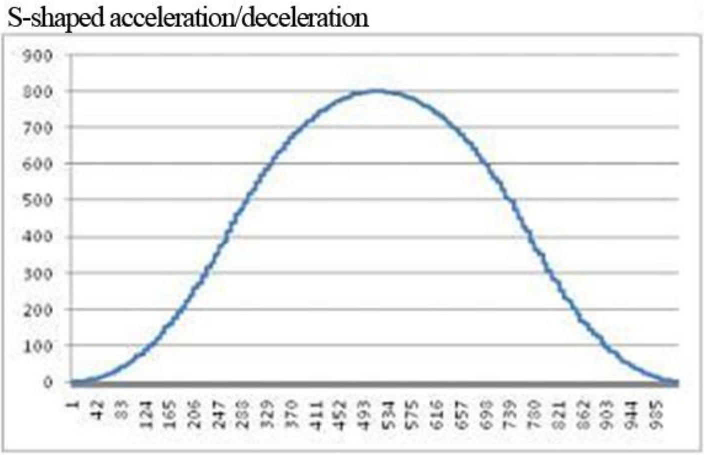
During quantitative driving, the acceleration counter records the accumulated pulses. When the remaining output pulses are less than acceleration pulse, it starts decelerating (automatic deceleration), and decelerates to start speed linearly in specified deceleration.

The following parameters should be set for linear acceleration/ deceleration driving:

- Acceleration A acceleration and deceleration
- Start speed SV
- Driving speed V

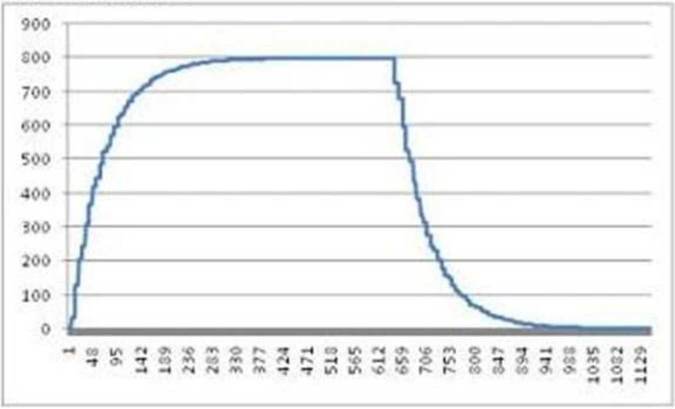

Speed
Driving speed
Acceleration
Deceleration
Start speed
Output pulse is low and can’t reach driving speed
Linear acceleration/deceleration driving
Time

1.5.3 S-shaped curve acceleration/deceleration driving



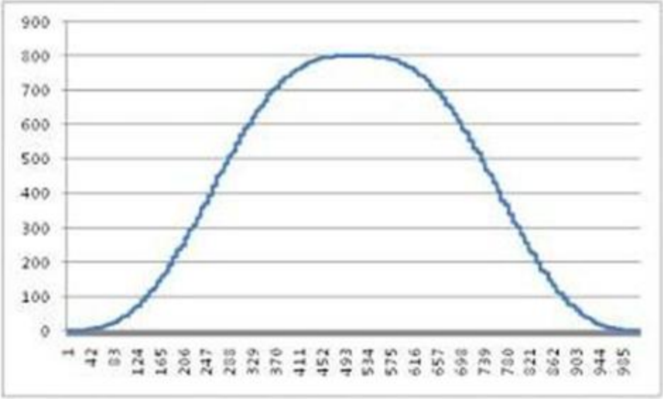
1.5.4 Exponential acceleration / deceleration driving

Exponential acceleration / deceleration



1.5.5 Trigonometric acceleration / deceleration driving

Trigonometric acceleration / deceleration



1.6 Position latch:

Use IN signal of each axis to achieve hardware position latch function. Use one latch signal to lock current position of all axes, and the locked position can be logic location or actual position.

Position latch function has important applications in the measurement system.

1.7 External signal driving:

External signal driving is the movement controlled by external signal (hand wheel or switch), and is mainly used for manual debugging, particularly convenient in teaching system.

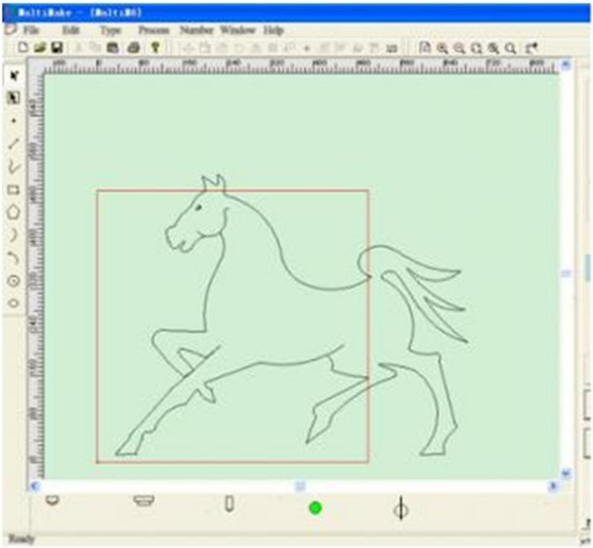
1.8 Large cache small segment:

Large cache small segment: large capacity multi-axis cache interpolation, store 10K interpolation instructions; small segments and large cache are used for engraving or cutting applications to make discrete CAM data can be restored to processing model .

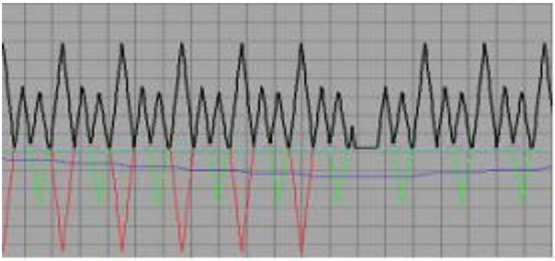
1.9 Speed adaptive model:

Speed adaptive model: to ensure precision under high speed, automatic speed optimization; used for milling machine, tooling and other applications requiring high precision control, makes the motor work in reasonable error range from speed planning.

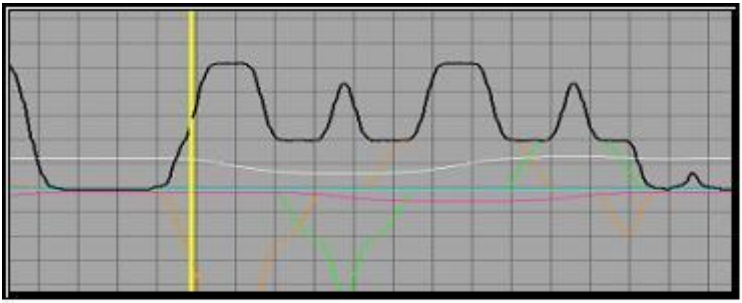
For example, to run the following track, monitor the speed curve:



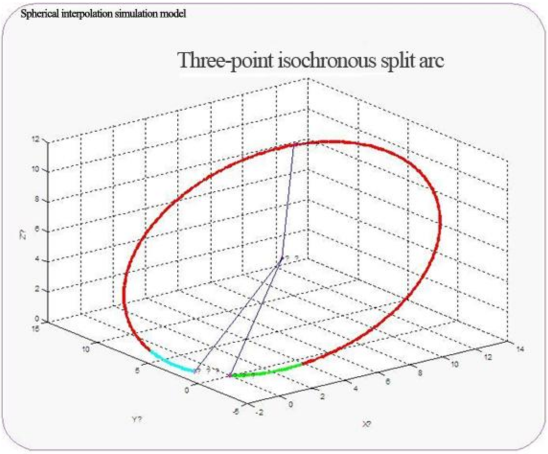
General speed curve:



Speed adaptive model curve:



1.10 Spherical arc interpolation:



3D arc interpolation (spherical interpolation): achieve arc in any spatial plane and spherical arc, suitable for teaching operation of simplified complex graphics. Comprehensive 3D arc interpolation, hardware-level support, support cache interpolation, only occupies four data segments, arc approximation accuracy is determined by interpolation speed dynamically to avoid discrete error of small segment approximation and contradictions of difficult speed trade-off. - Helical interpolation and plane arc interpolation can be easily achieved based on spherical interpolation technology.

1.11 Various acceleration and deceleration modes:

Various acceleration and deceleration modes: T, S, E, C-type acceleration and deceleration, and support asymmetric acceleration and deceleration, smooth running, quiet motor.

1.12 NURBS interpolation:

General cards provide only linear and circular interpolation. For non-linear and arc curve, linear and arc piecewise fitting method is used for interpolation. This method may cause large data size, poor accuracy, uneven feed rate, complicated programming and other issues in processing complex curves, which will inevitably lead to a greater impact on the processing quality and costs. Spline is a method that enables direct interpolation of complicated free curves and surfaces.

1.13 Simultaneous control of multiple processes:

Simultaneous control of multiple processes: You can open two programs to control one card. (A single monitoring program run simultaneously with the execution program; the execution program doesn’t need to switch a lot of time for display, which makes the display more real-time.)

1.14 Gantry dual-drive, changing drive speed and target position in motion:

Gantry dual-drive, changing drive speed and target position in motion in real time

1.15 Specify the time for four-axis linear interpolation:

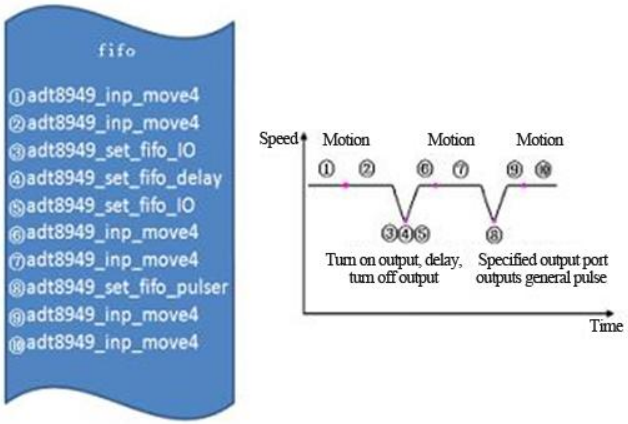
Specify the time for four-axis linear interpolation, facilitate speed planning customization

1.16 Pulse generator:

Insert up to 15-channel pulse generator function, specify the number of turns of the output level and holding time of high/low level, the accuracy is milliseconds, and can be used in dispensing, cutting, production line and other industries.

For example, in dispensing industry, the capacity of glue gun is fixed and it is required to feed the glue gun from time to time during motion. For conventional practice, the user calculates the glue amount and feed the glue gun by operating the output point on host computer. This approach is inaccurate, the operation is not flexible and result in untimely feeding easily.

The pulse generator function only needs to specify the glue position of the glue gun track and number of operations of glue feeding piston to achieve the glue feeding action easily. When the machine tool moves to the position specified by the user, the motion card will automatically activate the pulse generator to drive the piston, which can ensure easier operation and more accurate control.



1.17 Get arc length, set 15 filter levels for input point:

Get the arc length, and set 15 filter levels for input point.

Chapter 2 Motion Control Library Function Guide

2.1 Introduction of ADT-8949 function library

ADT-8949 function library is the interface for users operating motion control card. The users can control the motion card to complete corresponding function by transferring interface functions.

The motion control card provides the motion function library in DOS and DLL in Windows. The transferring methods of function library in DOS and Windows are introduced respectively below.

2.2 Calling DLL on Windows

The DLL “adt8949.dll” in Windows is written with VC. It is in “Development Kit \Drivers\DLL” in the CD, and is suitable for the programming language tools in Window: VB, VC, C++Builder, VB.NET, VC.NET, Delphi and configuration software LabVIEW, etc.

2.3 Calling in VC

(1) Create a new project;

(2) Copy the files “adt8949.lib” and “adt8949.h” from “Development Kits \VC” in the CD to the path of the new project;

(3) In the “File View” of the “Work Area” of the new project, right click the mouse and select “Add Files to Project”, select “Library Files(.lib)” in the Add Files dialog box, search and select “adt8949.lib” and click “OK” to load the static library;

(4) Add #include “adt8949.h” to the statement of source file or header file or global header filer “StdAfx.h”;

After above four steps, the user can call the functions in the DLL.

Note: The calling in VC.NET is similar as VC.

2.4 Calling in VB

(1) Create a new project;

(2) Copy the file “adt8949lib.bas” from “Development Kits \VB” in the CD to the path of the new project;

(3) Select “Project\Add module” menu, and select the “Save” tab in the dialog box, search the “adt8949lib.bas” module file, and click the Open button;

After above three steps, the user can call the functions in the DLL.

Note: The calling in VB.NET is similar as VB.

2.5 Calling in C++Builder

(1) Create a new project;

(2) Copy the files “adt8949.lib” and “adt8949.h” from “Development Kits \ C++Builder” in the CD to the path of the new project;

(3) Select the “Project\Add to Project” menu, select “Library Files(.lib)” in the dialog box, search and select “adt8949.lib”

and click the "OK" button;

(4) Add #include "adt8949.h" to the statement of the program file;

After above four steps, the user can call the functions in the DLL.

2.6 Return value and meaning of library function

To ensure that the user can control the execution when using library function, every function in the library will return the result after execution. The user can check whether the function transfer is successfully according to the return value.

Except "int adt8949_initial(void)" and "int read_bit(int cardno, int number)" in the function library, other functions only return "0" and "1", where "0" indicates successful transfer and "1" indicates failed.

The meanings of the return values are described in the table below.

Function name	Return value	Meaning
adt8949_initial	-1	Service is not installed
	-2	PCI bridge fault
	-3	DSP program download error
	-4	Hardware exception or DLL version does not match
	-5	Failed to create mutex
	-6	Failed to open mutex
	-7	Other causes
	0	Control card is not installed
	>0	Quantity of control cards
adt8949_read_bit	0	Low level
	1	High level

	-1	Card number or input point overrun error
All other functions	0	Correct
	Non-0	Wrong

Note: Return 1 error is usually caused by false cardno (card number) or axis (axis number) during transferring library function. The value of card number must be 0, 1, 2 in sequence, and the Card number must be 0 if there is only one card; axis number must be 1, 2, 3, 4, and other values are false.

Chapter 3 Key Points for Motion Control Development

There will be some problems in the programming of the card. In fact, most of the problems are caused by misunderstanding the principal of the control card. Below is the description of some familiar instances that are easily understood.

3.1 Card initialization

3.1.1 Description

At the beginning of the program, call function adt8949_initial() first, check whether ADT8949 card is installed properly, and then set the pulse output mode and limit switch work mode. Above parameters should be set according to specific machine, and set only once when the program is initialized.

Note: library function “adt8949_initial” is the “door” to ADT8949 card. Calling other functions has meaning only after transferring this function and initializing the motion control card successfully.

3.1.2 Electronic gear ratio setting

If the motor drive moves 1 mm for 10,000 pulses, the electronic gear ratio is set to 10000, that is, the gear parameter of adt8949_set_gear is set to 10000. Before executing linkage and interpolation motion instructions, the axis number parameter of adt8949_set_gear starts from 1, 2, 3, 4.

3.2 Speed setting

3.2.1 Constant speed motion

The parameter setting is simple. It is only required to set the driving speed to be lower than or equal to the start speed, and other parameters do not need setting.

Related functions:

set_startv

set_speed

3.2.2 Trapezoidal acceleration/deceleration

This is a most commonly used mode. It requires setting start speed, driving speed and acceleration, and uses automatic deceleration.

Related functions:

set_startv

set_speed

set_acc

3.2.4 STOP0, STOP1 signal

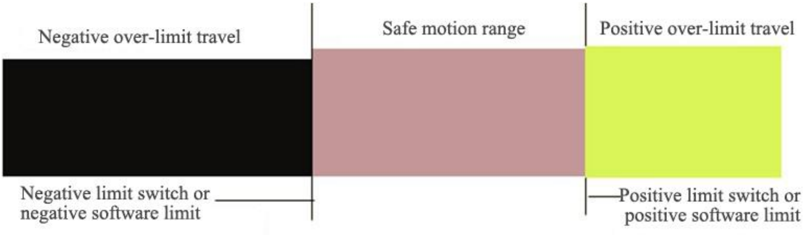
First, STOP0 and STOP1 (encoder Z phase signal) are the signals that every axis has, and thus there are 8 STOP signals totally. These signals are mainly used for home operation. Home mode can use one or several signals accordingly. However, please note that this signal is decelerating stop. For high speed home operation, please add a deceleration switch before the home switch, that is to say, use two STOP signals, one home switch and one deceleration switch. It is also possible to use one signal, which decelerates stop after meeting STOP signal, moves reversely at constant speed and stops after meeting again.

Chapter 4 System Security Mechanism

4.1 Monitoring error message:

Using get_stopdata () function to get the error message can get the stop information of the axis, including the stop caused by hardware limit, stop caused by origin signal, normal stop, and other stops.

4.2 Limit:


Negative over-limit travel
Safe motion range
Positive over-limit travel
Negative limit switch or negative software limit
Positive limit switch or positive software limit

The motion control card can use limit switch or software limit to control motion range of axes. If negative limit switch or negative software limit is triggered, it only moves towards positive direction; if positive limit switch or positive software limit is triggered, it only moves towards negative direction; note that the software limit is available only after successful homing.

- Use of software limit

The positive and negative limit of software limit is a concept of absolute position rather than an incremental value: the positive limit travel and negative limit travel are relative to the origin of the axis. Therefore, in a specific project, be sure to enable the software limit only when each axis has been reset (successful homing), and the mechanical origin at this time is the origin of the axis. The use of software limit needs to call set_soft_limit and enable_soft_limit. See Chapter 14 for detailed explanation of functions.

Chapter 5 High Speed Capture of External Signal Homing

5.1 Homing motion:

5.1.1 List of required functions:

Set homing mode	SetHomeMode_Ex
Set homing speed	SetHomeSpeed_Ex
Start homing	HomeProcess_Ex
Check status	GetHomeStatus_Ex
Note	See Important Note

Below is an example of single axis homing; multi-axis is similar.

Use STOP0 as home signal

(1) Homing is divided into three steps:

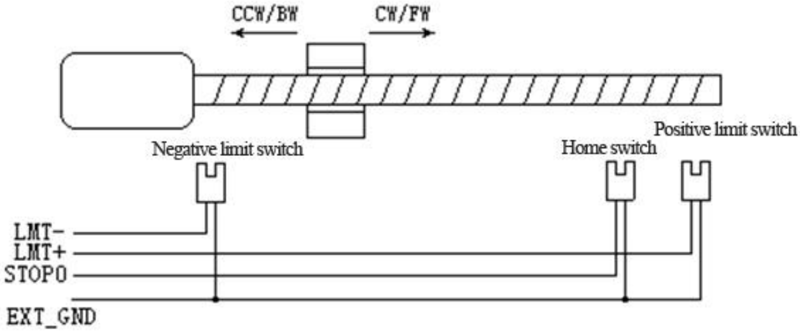
Step 1: approach stop0 quickly (logical0 home setting), and find stop0;

Step 2: record the locking position, and move to the recorded position.

Step 3: approach stop1 slowly (logical1 encoder Z-phase).

(2) You can choose whether to perform in the third step through

logical1.



5.1.2 Routine

```
void main()
{
        //X-axis homing for example
        const int CARDNO = 0;
        const int X_AXIS = 1;
        int res = -1;

        res = SetHomeMode_Ex(CARDNO, X_AXIS, 0,
                0, 0, -1,
                10, 0,
                0,1);
        res = SetHomeSpeed_Ex(CARDNO, X_AXIS, 2, 10,10,
0);

        res = HomeProcess_Ex(CARDNO,X_AXIS,16,1000);
        while(true)
        {
```

```
                        DoEvent();
                        Sleep(1);
                        res = GetHomeStatus_Ex(CARDNO,X_AXIS);
                        if(res == 0)//Homing successfully
                        {
                            break;
                        }

                }
        }
```

5.1.3 Important Note

Note the parameters setting for homing. See detailed explanation of functions for the specific meaning of the parameters. The start speed of homing (STOP0) can’t be greater than the homing speed. If it is not necessary to search STOP1 signal, set this parameter to -1. When necessary, follow the function instructions to set. Abovementioned is the example of X-axis. For multi-axis homing, just add function calling. Please note that the example doesn’t search Z phase signal, i.e. STOP1 signal. When necessary, set and call it.

Chapter 6 Linkage Control

Linkage control includes single axis point motion and multi-axis point motion. The specific implementation in the project depends on the requirement.

All the routines in this chapter are independent of each other. In the process of implementing the project, card initialization and gear only need to be set once in the process of system initialization. For the setting of electronic gear ratio, see Chapter 3 Section 3.1.2 Electronic gear ratio settings.

6.1 Single axis quantitative uniform motion:

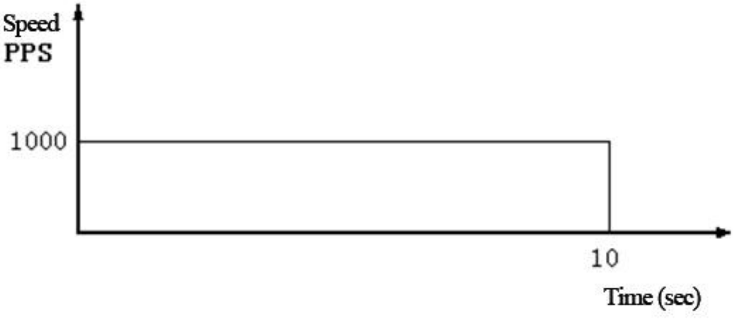
6.1.1 List of required functions

Set start speed	adt8949_set_startv
Set driving speed	adt8949_set_speed
Driving instruction	adt8949_pmove
Read driving status	adt8949_get_status
Note	See Important Note

6.1.2 Routine

Purpose:

Let X-axis stepper motor moves 10,000 steps at 1000 pps speed:



The procedure is as follows:

```
#include “adt8949.h”
void main()
{
        int cardno;
        cardno= adt8949_initial();
        if(cardno<=0) return;                  //ADT8949 card isn’t
installed
```

```
                                   //Below is the X-axis operation
of the first card; multi-axis operation is allowed
                                   //If there is more than one card,
i.e. cardno > 1
                                   //You can modify the card
number to operate other cards
    adt8949_set_pulse_mode(0,1,1,0,0);  //Set X-axis to pulse
+ direction
    adt8949_set_gear(0, 1,1000);//1000 pulses run 1mm
    adt8949_set_startv(0,1,1);//1000/1000=1
    adt8949_set_speed(0,1,1); //If the start speed is greater than
or equal to the //driving speed, it is constant speed motion,
1000/1000=1
    adt8949_pmove(0,1,10);      //Start driving 10000/1000=10
    int s;
    while(1)
    {
        adt8949_get_status(0,1,&s);       //Read driving status
        if(s==0)break;                    //Driving ends and exits
        ……                                //Perform reading
keyboard, displaying position and other functions
    }
    return ;
    }
```

6.1.3 Important Note

This routine only involves control card initialization function. Before setting the pulse mode and electronic gear ratio mode functions, it only needs to be set in program initialization and doesn’t need to set again.

6.2 Single axis quantitative symmetry trapezoidal acceleration / deceleration motion:

6.2.1 List of required functions:

Set to trapezoidal acceleration / deceleration	adt8949_set_admode
Set start speed	adt8949_set_startv
Set driving speed	adt8949_set_speed
Set acceleration	adt8949_set_acc
Driving instruction	adt8949_pmove
Read driving status	adt8949_get_status
Note	See Important Note

6.2.2 Example:

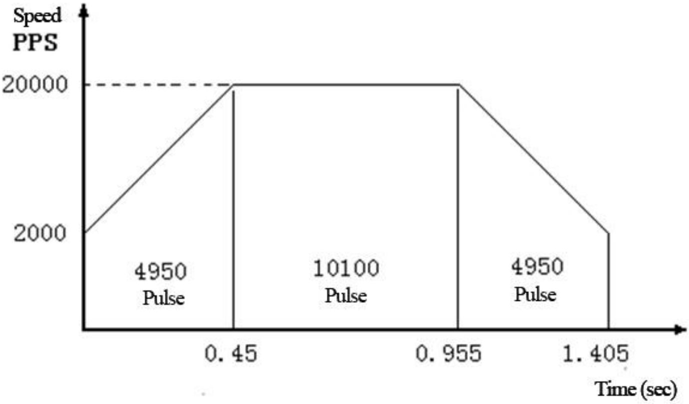
Purpose:

Let X-axis moves 20,000 steps at the following speed:

Start speed: 2000 pss

Driving speed: 20000 pss

Acceleration / deceleration: 40000 pss



Acceleration time should be (20000-2000)/40000=0.45 sec

Acceleration pulse should be 0.45*(20000+2000)/2=4950 pcs

Deceleration is same as acceleration.

```
#include “adt8949.h”
void main()
{
        int cardno;
        cardno=adt8949_initial();
        if(cardno<=0) return;                  //ADT8949 card isn’t
installed
                                               //Below is the X-axis operation
of the first card
                                               //If there is more than one card,
i.e. cardno > 1
                                               //You can modify the card
number to operate other cards
        adt8949_set_pulse_mode(0,1,1,0,0);         //Set X-axis to
pulse + direction
        adt8949_set_gear(0, 1,1000);//1000 pulses run 1mm
        adt8949_set_ad_mode(0,1,0);            //Set to trapezoidal
acceleration / deceleration
        adt8949_set_startv(0,1,2);    //Start speed 2000/1000=2
        adt8949_set_speed(0,1,20);       //Driving speed
                                                   20000/1000=20
        adt8949_set_acc(0,1,40); //Acceleration / deceleration 40000
/1000=40
        adt8949_pmove(0,1,20);     //Start driving 20000/1000=20
```

```
          int s;
          while(1)
          {
                adt8949_get_status(0,1,&s);       //Read driving status
                if(s==0)break;                    //Driving ends and exits
                ……                    //Perform reading keyboard,
      displaying position and other functions
          }
          return ;
      }
```

6.2.2 Important Note:

This routine only involves control card initialization function. Before setting the pulse mode and electronic gear ratio mode functions, it only needs to be set in program initialization and doesn’t need to set again.

6.3 Single axis quantitative asymmetry trapezoidal acceleration / deceleration motion:

6.3.1 List of required functions:

Set to trapezoidal acceleration / deceleration	adt8949_set_admode
Set start speed	adt8949_set_startv
Set driving speed	adt8949_set_speed
Set acceleration	adt8949_set_acc
Set deceleration	adt8949_set_dec
Driving instruction	adt8949_pmove

Read driving status	adt8949_get_status
Note	See Important Note

6.3.2 Example:

Purpose:

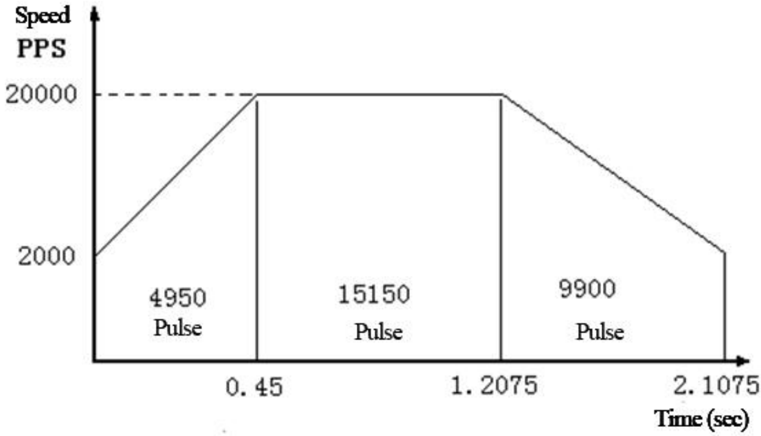
Let X-axis moves 30,000 steps at the following speed:

Start speed: 2000 pss

Driving speed: 20000 pss

Acceleration: 40000 pss

Deceleration: 20000 pss



Acceleration time should be (20000-2000)/40000=0.45 sec

Acceleration pulse should be 0.45*(20000+2000)/2=4950 pcs

Deceleration time should be (20000-2000)/20000=0.9 sec

Deceleration pulse should be 0.9*(20000+2000)/2=9900 pcs

```
    #include “adt8949.h”
    void main()
    {
        int cardno;
        cardno=adt8949_initial();
        if(cardno<=0) return;   //ADT8949 card isn’t installed
                                //Below is the X-axis operation of
    the first card
                                //If there is more than one card, i.e.
    cardno > 1
                                //You can modify the card number
    to operate other cards
        adt8949_set_pulse_mode(0,1,1,0,0);//Set X-axis to pulse +
    direction
        adt8949_set_gear(0, 1,1000);//1000 pulses run 1mm
        adt8949_set_admode(0,1,1);          //Set to trapezoidal
acceleration / deceleration
        adt8949_set_startv(0,1,2); //Start speed 2000/1000=2
        adt8949_set_speed(0,1,20);      //Driving speed
                                         20000/1000=20
        adt8949_set_acc(0,1,40);    //Acceleration 40000/1000=40
        adt8949_set_dec(0,1,20);     //Deceleration 20000/1000=20
        adt8949_pmove(0,1,20); //Start driving 20000/1000=20
        int s;
        while(1)
        {
            adt8949_get_status(0,1,&s);      //Read driving status
            if(s==0)break;                  //Driving ends and exits
            ……                              //Perform reading
    keyboard, displaying position and other functions
```

```
            }
            return ;
        }
```

6.3.3 Important Note:

This routine only involves control card initialization function. Before setting the pulse mode and electronic gear ratio mode functions, it only needs to be set in program initialization and doesn't need to set again.

6.4 Single axis quantitative S-curve acceleration / deceleration motion:

6.4.1 List of required functions:

Set to S-curve acceleration / deceleration	adt8949_set_admode
Set start speed	adt8949_set_startv
Set driving speed	adt8949_set_speed
Set acceleration	adt8949_set_acc
Set jerk	adt8949_set_jcc
Driving instruction	adt8949_pmove
Read driving status	adt8949_get_status
Note	See Important Note

Example 1: Full S-curve

Purpose:

Let X-axis moves 20,000 steps of S-curve acceleration motion at the following speed

Start speed: 1000 pss
Driving speed: 40000 pss
Acceleration time: 0.4 sec

```
#include “adt8949.h”
void main()
{
          int cardno;
          cardno=adt8949_initial();
          if(cardno<=0) return;                    //ADT8949 card isn’t
installed
                                                  //Below is the X-axis operation
of the first card
                                                 //If there is more than one card,
i.e. cardno > 1
                                                  //You can modify the card
number to operate other cards
          adt8949_set_pulse_mode(0,1,1,0,0);   //Set X-axis to pulse
+ direction
          adt8949_set_gear(0, 1,1000);//1000 pulses run 1mm
          adt8949_set_ad_mode(0,1,0);     //Set to S-curve
acceleration / deceleration
          adt8949_set_startv(0,1,1);//Start speed    1000/1000=1
          adt8949_set_speed(0,1,40); //Driving speed
                                                        40000/1000=40
          adt8949_set_acc(0,1,200);//Acceleration / deceleration
200000/1000=200
          adt8949_set_jcc(0,1,5);  //Set jerk or use the default value
          adt8949_pmove(0,1,20);      //Start driving 20000/1000=20
          int s;
```

```
        while(1)
        {
              adt8949_get_status(0,1,&s);        //Read driving status
              if(s==0)break;                    //Driving ends and exits
              ……                                //Perform reading
    keyboard, displaying position and other functions
        }
        return ;
    }
```

6.5 Single axis quantitative exponential acceleration / deceleration motion:

6.5.1 List of required functions:

Set to exponential acceleration / deceleration	adt8949_set_admode
Set start speed	adt8949_set_startv
Set driving speed	adt8949_set_speed
Set acceleration	adt8949_set_acc
Driving instruction	adt8949_pmove
Read driving status	adt8949_get_status
Note	See Important Note

Example 1: Full S-curve

Purpose:

Let X-axis moves 20,000 steps of exponential acceleration motion at the following speed

Start speed: 1000 pss

Driving speed: 40000 pss

Acceleration time: 0.4sec

```
          #include "adt8949.h"
          void main()
          {
              int cardno;
              cardno=adt8949_initial();
              if(cardno<=0) return;          //ADT8949 card isn't
          installed
                                             //Below is the X-axis operation
          of the first card
                                            //If there is more than one card,
          i.e. cardno > 1
                                             //You can modify the card
          number to operate other cards
              adt8949_set_pulse_mode(0,1,1,0,0);  //Set X-axis to pulse
          + direction
              adt8949_set_gear(0, 1,1000);//1000 pulses run 1mm
              adt8949_set_ad_mode(0,1,2);    //Set to exponential
      acceleration / deceleration
              adt8949_set_startv(0,1,1);//Start speed   1000/1000=1
              adt8949_set_speed(0,1,40); //Driving speed
                                                  40000/1000=40
              adt8949_set_acc(0,1,200);//Acceleration / deceleration
      200000/1000=200
              adt8949_pmove(0,1,20);     //Start driving 20000/1000=20
              int s;
              while(1)
              {
                  adt8949_get_status(0,1,&s);      //Read driving status
```

```
                if(s==0)break;                  //Driving ends and exits
                ……                              //Perform reading
    keyboard, displaying position and other functions
        }
        return ;
}
```

6.6 Single axis quantitative trigonometric acceleration and deceleration mode motion:

6.6.1 List of required functions:

Set to exponential acceleration / deceleration	adt8949_set_admode
Set start speed	adt8949_set_startv
Set driving speed	adt8949_set_speed
Set acceleration	adt8949_set_acc
Driving instruction	adt8949_pmove
Read driving status	adt8949_get_status
Note	See Important Note

Example 1: Full S-curve

Purpose:

Let X-axis moves 20,000 steps of exponential acceleration motion at the following speed; if the electronic gear ratio is 1000, i.e. 20mm

Start speed: 1000 pss

Driving speed: 40000 pss

Acceleration time: 0.4sec

```
    #include “adt8949.h”
    void main()
    {
        int cardno;
        cardno=adt8949_initial();
        if(cardno<=0) return;          //ADT8949 card isn’t
    installed
                                      //Below is the X-axis operation
    of the first card
                                      //If there is more than one card,
    i.e. cardno > 1
                                      //You can modify the card
    number to operate other cards
        adt8949_set_pulse_mode(0,1,1,0,0);   //Set X-axis to pulse
    + direction
         adt8949_set_gear(0, 1,1000);//1000 pulses run 1mm
        adt8949_set_ad_mode(0,1,3);     // Set to exponential
acceleration / deceleration
        adt8949_set_startv(0,1,1);//Start speed    1000/1000=1
        adt8949_set_speed(0,1,40); //Driving speed
                                           40000/1000=40
        adt8949_set_acc(0,1,200);//Acceleration / deceleration
200000/1000=200
        adt8949_pmove(0,1,20);     //Start driving 20000/1000=20
        int s;
        while(1)
        {
            adt8949_get_status(0,1,&s);     //Read driving status
            Sleep(1);
            if(s==0)break;               //Driving ends and exits
            ……                         //Perform reading
    keyboard, displaying position and other functions
        }
        return ;
}
```

6.7 Multi-axis motion

6.7.1 List of required functions:

Set to trapezoidal acceleration /deceleration	adt8949_set_admode
Set start speed	adt8949_set_startv
Set driving speed	adt8949_set_speed
Set acceleration	adt8949_set_acc
Set jerk	adt8949_set_jcc
Driving instruction	adt8949_pmove
Continuous motion instructions	adt8949_continue_move
Read driving status	adt8949_get_status
Note	See Important Note

Although the above is single axis, you can set the data of additional axes at the same time in practice. They won't affect each other. In X-axis driving, set the parameters of Y-axis properly and then drive the Y-axis. It will not have any effect on the motion of X-axis, so that four axes can be operated independently.

Below is a simple example. X-axis moves 1000 steps at a constant speed (1000pps), Y-axis moves 300,000 steps in linear acceleration/deceleration (start speed: 10,000pps, drive speed: 200,000pps, acceleration time: 0.2 sec), Z-axis performs complete S-curve accelerated continuous motion (start speed: 1000 pps, drive speed: 4000 pss, acceleration time: 1.2 sec), and W-axis performs continuous motion at a constant speed (300,000 pps); press the “S” key to stop.

6.7.2 Example:

The procedure is as follows:

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include “adt8949.h”
void main()
{
    int cardno;
    cardno=ADT-8949_initial();
    if(cardno<=0) return;                //ADT-8949 card isn’t
installed
                                        //Below is the X-axis operation
of the first card
                                        //If there is more than one card,
i.e. cardno > 1
                                        //You can modify the card
number to operate other cards
    adt8949_set_pulse_mode(0,1,1,0,0);        //Set X-axis to
pulse + direction
    adt8949_set_pulse_mode(0,2,1,0,0);        //Set Y-axis to
    pulse + direction
            adt8949_set_pulse_mode(0,3,1,0,0);          //Set Z-axis to
    pulse + direction
            adt8949_set_pulse_mode(0,4,1,0,0);          //Set W-axis to
    pulse + direction

            adt8949_set_gear(0, 1,1000);//1000 pulses run 1mm
            adt8949_set_gear(0, 2,1000);//1000 pulses run 1mm
            adt8949_set_gear(0, 3,1000);//1000 pulses run 1mm
            adt8949_set_gear(0, 4,1000);//1000 pulses run 1mm
    //Y-axis acceleration /deceleration setting
            adt8949_set_admode(0,2,1);                // Set to trapezoidal
acceleration /deceleration

            //Z-axis acceleration /deceleration setting
            adt8949_set_admode(0,3,0);        //Set to S-curve
acceleration /deceleration

            //X-axis
            adt8949_set_startv(0,1,1);          //Start speed 1000/1000=1
            adt8949_set_speed(0,1,1);       //Driving speed
                                                  1000/1000=1

            //Y-axis
            adt8949_set_startv(0,2,10);         //Start speed
10000/1000=10
            adt8949_set_speed(0,2,200);      //Driving speed
                                                  200000/1000=200
            adt8949_set_acc(0,2,950);             //Acceleration
                                                  950000/1000=950
```

```
        //Z-axis
        adt8949_set_startv(0,3,1000);//1000/1000=1
        adt8949_set_speed(0,3,4000);//4000/1000=4
        adt8949_set_acc(0,3,53);    //6667/125=53.3
        adt8949_set_jcc(0,3,10);

        //W-axis
        adt8949_set_startv(0,4,7500);    //300000/1000=300
        adt8949_set_speed(0,4,300);//300000/1000=300
        adt8949_pmove(0,1,1000);  //Start driving 1000/1000=1
        adt8949_pmove(0,2,300);//300000/1000=300
        adt8949_continue_move(0,3,0);
        adt8949_continue_move(0,4,0);

        int s1,s2,s3,s4;
        while(1)
        {
        adt8949_get_status(0,1,&s1);          //Read X driving
status
                adt8949_get_status(0,2,&s2);    //Read Y driving
        status
                adt8949_get_status(0,3,&s3);    //Read Z driving
        status
                adt8949_get_status(0,4,&s4);    //Read W driving
        status

                if(s1==0 && s2==0 && s3==0 && s4==0)break;
                                                  //Driving ends and exits
                if(kbhit())
                key=getch();
            else
                key=-1;
            if(key==’s’)
            {
                adt8949_dec_stop(0,3);
                adt8949_dec_stop(0,4);
            }
        }
        return ;
    }
```

6.7.3 Important Note:

This routine only involves control card initialization function. Before setting the pulse mode and electronic gear ratio mode functions, it only needs to be set in program initialization and doesn’t need to set again.

Chapter 7 Interpolation Motion Control

All the routines in this chapter are independent of each other. In the process of implementing the project, card initialization and gear only need to be set once in the process of system initialization. For the setting of electronic gear ratio, see Chapter 3 Section 3.1.2 Electronic gear ratio settings.

7.1 Two-axis linear interpolation (constant speed)

7.1.1 List of required functions:

Set start speed	adt8949_set_startv
Set driving speed	adt8949_set_speed
Driving instruction	adt8949_inp_move4
Read interpolation status	adt8949_get_inp_status
Number of segments of pretreatment cache	adt8949_set_precount
Note	See Important Note

Interpolation speed bases on resultant velocity. Below is a simple example of constant speed linear interpolation. The constant speed driving of arc interpolation is basically same as multi-axis linear interpolation.

7.1.2 Example:

The procedure is as follows:

```
#include “adt8949.h”
void main()
{
        int cardno;
        cardno=adt8949_initial();
        if(cardno<=0) return;                    //ADT8949 card isn’t
installed
                                                        //Below is the X-axis operation
of the first card
                                                       //If there is more than one card,
i.e. cardno > 1
                                                        //You can modify the card
number to operate other cards
```

```
        adt8949_set_pulse_mode(0,1,1,0,0);          //Set X-axis to
pulse + direction
        adt8949_set_pulse_mode(0,2,1,0,0);          //Set Y-axis to
pulse + direction
        adt8949_set_pulse_mode(0,3,1,0,0);          //Set Z-axis to
pulse + direction
        adt8949_set_pulse_mode(0,4,1,0,0);          //Set W-axis to
pulse + direction
        adt8949_set_gear(0, 1,1000);//1000 pulses run 1mm
        adt8949_set_precount(0,0);//Number of segments of
pretreatment cache
        adt8949_set_startv(0,63,1);         //X start speed
1000/1000
        adt8949_set_speed(0,63,1);            //X driving speed
                                              1000/1000

        adt8949_inp_move4(0, 0,10,-20,0,0); //X-Y start
interpolating
                                         //X moving forward
10000/1000
                                         //Y moving backward
20000/1000
        int s1;
        while(1)
        {
                adt8949_get_inp_status(0,&s1);  //Read interpolation
status
                if(s1==0)break;     //Interpolation ends and exits
                ……                               //Perform reading
```

```
    keyboard, displaying position and other functions
    }
        return ;
    }
```

7.1.3 Important Note:

This routine only involves control card initialization function. Before setting the pulse mode and electronic gear ratio mode functions, it only needs to be set in program initialization and doesn’t need to set again.

7.2 Two-axis linear interpolation (acceleration / deceleration)

7.2.1 List of required functions:

Set acceleration / deceleration mode	adt8949_set_admode
Set start speed	adt8949_set_startv
Set driving speed	adt8949_set_speed
Set acceleration	adt8949_set_acc
Set deceleration	adt8949_set_dec
Interpolation instruction	adt8949_inp_move4
Read interpolation status	adt8949_get_inp_status
Number of segments of pretreatment cache	adt8949_set_precount
Note	See Important Note

For the acceleration / deceleration driving of two-axis linear interpolation, set the interpolation axis to trapezoidal acceleration/ deceleration, S-curve acceleration/ deceleration, exponential acceleration/ deceleration, and trigonometric acceleration/ deceleration.

7.2.2 Example:

The procedure is as follows:

```
#include “adt8949.h”
void main()
{
        int cardno;
        cardno=adt8949_initial();
        if(cardno<=0) return;                    //ADT8949 card isn’t
installed
                                                  //
                                                 //If there is more than one card,
i.e. cardno > 1
                                                  //You can modify the card
number to operate other cards

        adt8949_set_pulse_mode(0,1,1,0,0);   //Set X-axis to pulse
+ direction
        adt8949_set_pulse_mode(0,2,1,0,0);          //Set Y-axis to
pulse + direction
        adt8949_set_pulse_mode(0,3,1,0,0);          //Set Z-axis to
pulse + direction
        adt8949_set_pulse_mode(0,4,1,0,0);          //Set W-axis to
pulse + direction
        adt8949_set_precount(0,0);//Number of segments of
pretreatment cache
        adt8949_set_gear(0, 1,1000);//1000 pulses run 1mm
        adt8949_set_gear(0, 2,1000);//1000 pulses run 1mm

        adt8949_set_admode(0,63,1);
        adt8949_set_startv(0,63,1);//X start speed 1000/1000=1
        adt8949_set_speed(0,63,8); //X driving speed
                                                      8000/1000=8
        adt8949_set_acc(0,63,1);//1000/1000=1
        adt8949_set_dec(0,63,1);//1000/1000=1

        adt8949_inp_move4(0,0,10,-20,0,0);  //X-Y start
interpolating
                                                  //X moving forward
10000/1000=10
                                                  //Y moving backward
20000/1000=20
        int s1;
        while(1)
        {
            adt8949_get_inp_status(0,&s1);  //Read interpolation
status
            if(s1==0)break;    // Interpolation ends and exits
            ……                              //Perform reading
keyboard, displaying position and other functions
        }
        return ;
}
```

7.2.3 Important Note:

This routine only involves control card initialization function. Before setting the pulse mode function, it only needs to be set in program initialization and doesn't need to set again.

Linear interpolation is suitable for trapezoidal, S-shaped, exponential, and trigonometric acceleration/deceleration.

7.3 2D arc interpolation (acceleration/deceleration)

7.3.1 List of required functions:

Set acceleration / deceleration mode	adt8949_set_admode
Set start speed	adt8949_set_startv
Set driving speed	adt8949_set_speed
Set acceleration	adt8949_set_acc
Interpolation instruction	adt8949_inp_arc2
Read interpolation status	adt8949_get_inp_status
Number of segments of pretreatment cache	adt8949_set_precount
Note	See Important Note

Two-axis arc interpolation is generally driven by constant speed or trapezoidal and trigonometric acceleration/deceleration, but can't be driven by S-curve or exponential acceleration/deceleration.

Constant speed drive is relatively simple. You just need to set the start speed of the first axis same to the driving speed.

Acceleration/deceleration driving requires setting acceleration/deceleration mode. The example below illustrates the driving method by driving a full circle of 10 mm radius.

7.3.2 Example:

The procedure is as follows:

```
#include “adt8949.h”
void main()
{
        int cardno;
        cardno=adt8949_initial();
        if(cardno<=0) return;                    //ADT8949 card isn’t
installed
                                                //Below is the X-axis
operation of the first card
                                                //If there is more than one
card, i.e. cardno > 1
                                                //You can modify the card
number to operate other cards

        adt8949_set_pulse_mode(0,1,1,0,0);        //Set X-axis to
pulse + direction
        adt8949_set_pulse_mode(0,2,1,0,0);        //Set Y-axis to
pulse + direction
        adt8949_set_pulse_mode(0,3,1,0,0);        //Set Z-axis to
pulse + direction
        adt8949_set_pulse_mode(0,4,1,0,0);        //Set W-axis to
pulse + direction
        adt8949_set_precount(0,0);//Number of segments of
pretreatment cache
        adt8949_set_ad_mode(0,63,1);              //Linear
acceleration / deceleration
        adt8949_set_startv(0,63,1);                   //X start
```

```
    speed    500/5
        adt8949_set_speed(0,63,4);                         //X driving
                                                  speed    40000/1000
        adt8949_set_acc(0,63,4);                     //X acceleration
Float m_fPulseEnd[4]={0,0,10,10};
Float m_fPulseCenter[4]={10,10,10,10};
        adt8949_inp_arc2(0,0,3,m_fPulseEnd,m_fPulseCenter,1);
    //X-Y clockwise arc interpolation
                                                  //End point 0, 0, i.e. draw
    a full circle
                                                  // Center position 10, 10
        int s1;
        while(1)
        {
            adt8949_get_inp_status(0,&s1);  //Read X-Y
    interpolation status
            if(s1==0 )break;          //Interpolation ends and exits
            ……                        //Perform reading keyboard,
    displaying position and other functions
        }
        return ;
    }
```

7.3.3 Important Note:

This routine only involves control card initialization function. Before setting the pulse mode function, it only needs to be set in program initialization and doesn’t need to set again.

Arc interpolation is only suitable for trapezoidal and trigonometric acceleration/deceleration.

7.4 3D arc interpolation (acceleration/deceleration)

7.4.1 List of required functions:

Set acceleration / deceleration mode	adt8949_set_admode
Set start speed	adt8949_set_startv
Set driving speed	adt8949_set_speed
Set acceleration	adt8949_set_acc
Interpolation instruction	adt8949_inp_arc3
Read interpolation status	adt8949_get_inp_status
Number of segments of pretreatment cache	adt8949_set_precount
Note	See Important Note

Three-axis arc interpolation is generally driven by constant speed or trapezoidal and trigonometric acceleration/deceleration, but can’t be driven by S-curve or exponential acceleration/deceleration. Constant speed drive is relatively simple. You just need to set the start speed of the first axis same to the driving speed. Acceleration/deceleration driving requires setting acceleration/deceleration mode. The example below illustrates the driving method by driving a full circle of 10 mm radius.

7.4.2 Example:

The procedure is as follows:

```
#include “adt8949.h”
void main()
{
        int cardno;
        cardno=adt8949_initial();
        if(cardno<=0) return;                    //ADT854 card isn't
installed
                                                 //Below is the X-axis
operation of the first card
                                                 //If there is more than one
card, i.e. cardno > 1
                                                 //You can modify the card
number to operate other cards

        adt8949_set_pulse_mode(0,1,1,0,0);       //Set X-axis to
pulse + direction
        adt8949_set_pulse_mode(0,2,1,0,0);       //Set Y-axis to
pulse + direction
        adt8949_set_pulse_mode(0,3,1,0,0);       //Set Z-axis to
pulse + direction
        adt8949_set_pulse_mode(0,4,1,0,0);       //Set W-axis to
pulse + direction
        adt8949_set_precount(0,0);//Number of segments of
pretreatment cache
        adt8949_set_gear(0, 1,1000);//1000 pulses run 1mm
        adt8949_set_gear(0, 2,1000);//1000 pulses run 1mm

        adt8949_set_ad_mode(0,63,1);             //Linear
acceleration / deceleration
        adt8949_set_startv(0,63,1);                   //X start
speed   500/5
        adt8949_set_speed(0,63,4);            //X driving speed
        adt8949_set_acc(0,63,4);                 //X acceleration
```

```
Float m_fPulseEnd[4]={0,0,10,10};
Float m_fPulseCenter[4]={10,10,10,10};
        adt8949_inp_arc3(0,0,7,m_fPulseCenter,m_fPulseEnd,1);
    //X-Y clockwise arc interpolation
                                        // End point 0, 0, 10, i.e.
    draw a full circle
                                    //Center position 10, 10, 10
        int s1;
        while(1)
        {
            adt8949_get_inp_status(0,&s1);  //Read X-Y
    interpolation status
            if(s1==0 )break;          //Interpolation ends and exits
            ……                        //Perform reading keyboard,
    displaying position and other functions
        }
        return ;
    }
```

7.4.3 Important Note:

This routine only involves control card initialization function. Before setting the pulse mode function, it only needs to be set in program initialization and doesn’t need to set again.

Arc interpolation is only suitable for trapezoidal and trigonometric acceleration/deceleration.

Chapter 8 Track Motion Control

All the routines in this chapter are independent of each other. In the process of implementing the project, card initialization and gear only need to be set once in the process of system initialization. For the setting of electronic gear ratio, see Chapter 3 Section 3.1.2 Electronic gear ratio settings.

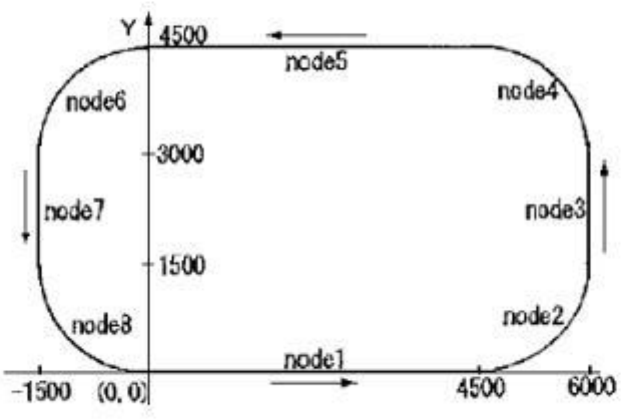
8.1 Cache interpolation

8.1.1 List of required functions:

Set to acceleration/deceleration mode	adt8949_set_admode
Set start speed	adt8949_set_startv
Set driving speed	adt8949_set_speed
Set acceleration	adt8949_set_acc
Set deceleration	adt8949_set_dec
Linear interpolation instruction	adt8949_inp_move4
Arc interpolation instruction	adt8949_inp_arc2
Note	See Important Note

Constant speed interpolation cache is similar to single interpolation. Just set the start speed same as the driving speed and set the number of preprocessing cache segments to a predetermined value.

Cache interpolation of acceleration/deceleration only requires setting the drive speed greater than the start speed.



Example of Cache Interpolation

In the above case, the start speed is set to 1000 pps, the driving speed is 2000 pps, the acceleration is 2000 pps/sec, the last segment in the figure is arc interpolation and the radius is 1500

8.1.2 Example:

The procedure is as follows:

```
#include "adt8949.h"
void main()
{
    int cardno;
    int s1,s2;
    //Arc interpolation data
    Float m_fPulseEndOne[4]={1500/1000,1500/1000,0,0};
    float m_fPulseCenterOne[4]={0,1500/1000,0,0};

    Float m_fPulseEndTwo[4]={-1500/1000,1500/1000,0,0};
    float m_fPulseCenterTwo[4]={-1500/1000,0,0,0};
```

```
        Float m_fPulseEndThree[4]={-1500/1000,-1500/1000,0,0};
        float m_fPulseCenterThree[4]={0,-1500/1000,0,0};

        Float m_fPulseEndThree[4]={1500/1000,-1500/1000,0,0};
        float m_fPulseCenterThree[4]={1500/1000,0,0,0};

        cardno=adt8949_initial();
        if(cardno<=0) return;                //ADT8949 card isn't
installed
                                             //Below is the X-axis operation
of the first card
                                             //If there is more than one card,
i.e. cardno > 1
                                             //You can modify the card
number to operate other cards

        adt8949_set_pulse_mode(0,1,1,0,0);        //Set X-axis to
pulse + direction
        adt8949_set_pulse_mode(0,2,1,0,0);        // Set Y-axis to
pulse + direction
        adt8949_set_precount(0,50);//Number of segments of
pretreatment cache
        adt8949_set_gear(0, 1,1000);//1000 pulses run 1mm
        adt8949_set_gear(0, 2,1000);//1000 pulses run 1mm

        adt8949_set_admode(0,63,1);               //Linear
acceleration / deceleration
        adt8949_set_startv(0,63,5);           //Start speed
        adt8949_set_speed(0,63,20);          //Driving speed
```

```
	adt8949_set_acc(0,63,20);		//Acceleration
	adt8949_set_dec(0,63,10);		//

	//Segment 1
	adt8949_inp_move4(0,0,4500/1000,4500/1000,0,0);
	//Segment 2
	adt8949_inp_arc2(0,0,3,m_fPulseEndOne,m_fPulseCenterOne,0);
	//Segment 3
	adt8949_inp_move4(0,0,0,1500/1000,0,0);
	//Segment 4
	adt8949_inp_arc2(0,0,3,m_fPulseEndTwo,m_fPulseCenterTwo,0);

	//Segment 5
	adt8949_inp_move4(0,0,-4500/1000,0,0,0);
	//Segment 6
	adt8949_inp_arc2(0,0,3,m_fPulseEndThree,m_fPulseCenterThree,0);
	//Segment 7
	adt8949_inp_move4(0,0,0,-1500/1000,0,0);
	//Segment 8
	adt8949_inp_arc2(0,0,3,m_fPulseEndThree,m_fPulseCenterThree,0);
	return ;
}
```

8.1.3 Important Note:

This routine only involves control card initialization function.

Before setting the pulse mode and electronic gear ratio mode functions, it only needs to be set in program initialization and doesn’t need to set again.
The number of pretreatment segment should be set. Only interpolation cache over 10000 segments needs to be queried with adt8949_get_fifo_len.

Chapter 9 Universal Digital I/O

The motion control card provides users with a universal digital input/output port. The host can operate the input/output port through instructions.

9.1 Input port definition: See Hardware CHAPTER 2 Electrical Connection for specific definition

9.2 Output port definition: See Hardware CHAPTER 2 Electrical Connection for specific definition

9.3 Output port:

9.3.1 List of required functions:

Output function	write_bit
Note	See Important Note

9.3.2 Example:

The procedure is as follows: 1# port for example, turn on the output

```
#include “adt8949.h”
void main()
{
      int cardno;
```

```
          cardno=adt8949_initial();
          if(cardno<=0) return;                //ADT8948 card isn't
    installed
          write_bit(0,1,1);
          return ;
          }
```

9.3.3 Important Note:

Output port defines port number as needed.

9.4 Input port:

9.4.1 List of required functions:

Read the function of input point	read_bit
Note	See Important Note

9.4.2 Example:

The procedure is as follows: 1# port for example, read the voltage level of 1# port, low level active

```
    #include "adt8949.h"
    void main()
    {
          int cardno;
          cardno=adt8949_initial();
          if(cardno<=0) return;                //ADT8948 card isn't
    installed

            int value = -1;
          value   = read_bit(0,1);
            if(value == 0)
            {
                 //Perform other corresponding operations
            }
        return ;
    }
```

9.4.3 Important Note:

This function is called according to the return value of the port.

Chapter 10 Auxiliary Control

All the routines in this chapter are independent of each other. In the process of implementing the project, card initialization and gear only need to be set once in the process of system initialization. For the setting of electronic gear ratio, see Chapter 3 Section 3.1.2 Electronic gear ratio settings.

11.1 Position locking:

11.1.1 List of required functions:

Locking mode	set_lock_position//
Check if position lock is executed	get_lock_status
Get locking position	get_lock_position
Note	See Important Note

11.1.2 Example:

Purpose:

Describe binding locking signal with X-axis for example

```
#include “adt8949.h”
void main()
```

```
{
        int cardno;
       int status = -1;
       long pos = -1;
        cardno=adt8949_initial();
        if(cardno<=0) return;                //ADT8948 card isn't
installed
                                              //Below is the X-axis operation
of the first card
                                             //If there is more than one card,
i.e. cardno > 1
                                              //You can modify the card
number to operate other cards
        set_lock_position (0,1,0,0);
          get_lock_status(0,    1, &status);
          if(status == 1)
        {
            get_lock_position(0, 1, &pos);
        }
        return ;
}
```

11.1.3 Important Note:

One locking signal is bound to one axis.

This routine only involves control card initialization function. Before setting the pulse mode and electronic gear ratio mode functions, it only needs to be set in program initialization and doesn't need to set again.

Chapter 11 List of ADT8949 Basic Library Functions

List of V100 library functions

Function Category	Function Name	Description	**Page**
Basic parameters	adt8949_initial	Initialize the card	**12.1.1**
	adt8949_close_card	Close source of motion card	**12.1.2**
	adt8949_get_lib_version	Get current library version	**12.1.3**
	adt8949_get_firmware_ver	Get current firmware version	**12.1.4**
	adt8949_set_pulse_mode	Set the working mode of output pulse	**12.1.5**
	adt8949_set_limit_mode	Limit mode	**12.1.6**
	adt8949_set_stop0_mode	Stop mode	**12.1.7**
	adt8949_set_stop1_mode	Stop mode	**12.1.8**
	adt8949_set_input_mode	Set input signal mode (including positive/negative limit, home)	**12.1.9**
	adt8949_set_gear	Set the electronic gear	**12.1.10**

		ratio of each axis (default: 1000)	
	adt8949_set_input_filter	Set filter level of input signal	**12.1.11**
	adt8949_set_actual_count_mode	Set the working mode of actual counter (encoder input)	**12.1.12**
	adt8949_set_emergency_stop_mode	Set the mode of positive/negative limit input nLMT signal and stop signal mode	**12.1.13**
Reset	adt8949_reset_card	Reset the motion card	**12.2.1**
Drive status checking	adt8949_get_status	Get axis drive state	**12.3.1**
	adt8949_get_status_all	Get the drive state of all axes	**12.3.2**
	adt8949_get_inp_status	Get interpolation drive status	**12.3.3**
Motion paramete	adt8949_set_precount	Set the number of	**12.4.1**

r setting		interpolation cache segments (default: 0)	
	adt8949_set_jcc	Set S-shaped jerk	**12.4.2**
	adt8949_set_acc	Set axis acceleration (set_acc)	**12.4.3**
	adt8949_set_dec	Set axis deceleration (set_dec)	**12.4.3**
	adt8949_set_admode	Set axis acceleration/ deceleration mode	**12.4.4**
	adt8949_set_speed	Set axis running speed	**12.4.5**
	adt8949_set_speed_constraint	Set speed constraint for motion path connection	**12.4.6**
	adt8949_set_acc_constraint	Set acceleration constraint for motion path connection	**12.4.7**
	adt8949_set_arc_speed_clamp	Set arc speed clamp	**12.4.8**
	adt8949_set_startv	Set start speed	**12.4.5**

		of axis	
	adt8949_set_endv	Set end speed of axis	**12.4.5**
	adt8949_set_command_pos	Set axis pulse logic position	**12.4.9**
	adt8949_set_actual_pos	Set axis pulse actual position	**12.4.10**
	adt8949_set_syncpos	Synchronize axis cache position and actual position	**12.4.11**
	adt8949_set_precount	Set the number of pretreatment cache segments	**12.4.12**
	adt8949_set_follow_axis	Set follow axis	**12.4.13**
	adt8949_set_rate1	Set total speed rates	**12.4.14**
	adt8949_set_rate2	Set speed rate of single axis	**12.4.15**
	adt8949_set_input_filter	Set filter level of input signal	**12.4.16**
Parameter checking	adt8949_get_command_pos	Get the logical location of each axis	**12.5.1**
	adt8949_get_actual_pos	Get the actual position of	**12.5.2**

		each axis	
	adt8949_get_speed	Get current driving speed of each axis	**12.5.3**
	adt8949_get_fifo_len	Query remaining segments in 10000 segments interpolation cache area	**12.5.4**
	adt8949_get_arc2_length	Get the length of two-axis arc	**12.5.5**
	adt8949_get_arc3_length	Get the length of three-axis arc	**12.5.6**
	adt8949_get_fifo_len	Query remaining segments in 10000 segments interpolation cache area	**12.5.7**
	adt8949_get_syserr	Get the latest error number of the system	**12.5.8**
	adt8949_get_stopdata	Get the stop data of each axis	**12.5.9**

Driving	adt8949_pmove	Single axis quantitative motion	**12.6.1**
	adt8949_abs_pmove	Absolute coordinates quantitative driving	**12.6.2**
	adt8949_continue_move	Single axis continuous motion	**12.6.3**
	adt8949_inp_arc2	Two-axis arc interpolation	**12.6.4**
	adt8949_inp_arc3	Three-axis arc interpolation	**12.6.5**
	adt8949_inp_abs_move4	Set four-axis interpolation instruction (absolute position)	**12.6.6**
	adt8949_inp_move4	Set four-axis interpolation instruction (relative position)	**12.6.7**
	adt8949_time_move4	Four-axis relative coordinates linear interpolation (specify	**12.6.8**

		motion time)	
	adt8949_time_abs_move4	Four-axis absolute coordinates linear interpolation (specify motion time)	**12.6.9**
	adt8949_inp_NURBS	NURBS interpolation	**12.6.10**
	adt8949_dec_stop	Driving deceleration stop	**12.6.11**
	adt8949_sudden_stop	Driving immediately stop	**12.6.12**
Switch quantity	adt8949_read_bit	Get input IO status (read_bit)	**12.7.1**
	adt8949_write_bit	Set output IO status (write_bit)	**12.7.1**
	adt8949_get_out	Get output point status	**12.7.1**
	adt8949_get_gpio	Read IO status by group	**12.7.2**
	adt8949_set_gpio	Operate output by group	**12.7.3**
	adt8949_set_multi_io	Set voltage level for	**12.7.4**

		multiple output points	
FIFO operation output	adt8949_set_fifo_event	Set FIFO event	**12.8.1**
	adt8949_set_fifo_io	Single point IO output in interpolation	**12.8.2**
	adt8949_set_fifo_multi_io	Set voltage level for multiple output points in interpolation	**12.8.3**
	adt8949_set_fifo_delay	Specific position delay motion in interpolation	**12.8.4**
	adt8949_set_fifo_pulser	Insert pulse generator in interpolation	**12.8.5**
Position locking	adt8949_set_lock_position	Set position locking function	**12.9.1**
	adt8949_get_lock_status	Get position locking status	**12.9.2**
	adt8949_get_lock_position	Get the position of position locking	**12.9.3**
	adt8949_clr_lock_status	Clear locking	**12.9.4**

		status	
	adt8949_set_EXlock_position	Set extended position locking function	**12.9.5**
	adt8949_get_EXlock_status	Get extended position locking status	**12.9.6**
	adt8949_get_EXlock_position	Get the position of extended position locking	**12.9.7**
	adt8949_clr_EXlock_status	Clear extended locking status	**12.9.8**
Handwheel function	adt8949_set_hand_wheel_mode	Set handwheel function mode	**12.10.1**
DA output	adt8949_set_daout	Set DA output voltage	**12.10.1**
Homing module	adt8949_SetHomeMode_Ex	Set home parameter	**13.11.1**
	adt8949_SetHomeSpeed_Ex	Home speed parameter	**13.11.2**
	adt8949_HomeProcess_Ex	Start homing	**13.11.3**
	adt8949_GetHomeStatus_Ex	Get home status	**13.11.4**

Chapter 12 Detailed Description of ADT8949 Basic Library Functions

12.1 Basic parameter setting

12.1.1 Initialize the card

Function	int _stdcall adt8949_initial(void)
Function description:	Initialize the card
Parameter settings:	None
Return value:	◆ If return value >0, it indicates the quantity of ADT8949 cards. If it is 3, the available card numbers will be 0, 1 and 2 respectively; ◆ If return value =0, no ADT8949 card is installed;
	◆ If return value <0, -1 indicates port drivers are not installed; ◆ -2 indicates PCI bridge failure. ◆ -3 indicates DSP program download error ◆ -4 indicates hardware exception or DLL version does not match ◆ -5 indicates failed to create mutex ◆ -6 indicates failed to open mutex ◆ -7 indicates other causes

12.1.2 Close source of motion card:

Function	int adt8949_close_card(void)
Function description:	Close source of motion card
Parameter settings	No
Return value:	0: Successful; -1: Failed
Remark	After calling this function, the control card will automatically reset, clear all motion instructions, and clear all output points; Using VB6.0 programming, if this function isn't called, the encoder may exit unexpectedly

12.1.3 Get current library version:

Function	int adt8949_get_lib_version(int cardno)
Function description	Get current library version
Parameter settings	cardno : Card number
Return value:	◆ Version number of the library
Remark	◆ ADT-8949 has driver library version number and firmware version number. The same firmware version may have different driver library version, so pay attention to the information of both version numbers for problem tracing.

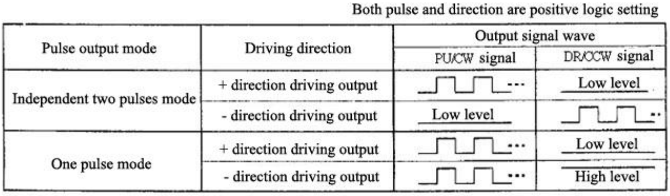
12.1.4 Get current firmware version:

Function	int adt8949_get_firmware_ver(int cardno)
Function description	Get current firmware version
Parameter settings	cardno : Card number
Return value:	◆ Return value contains the information of firmware version and card ID. ◆ The first five digits indicate card DIP ID. ◆ The last 27 digits indicate the firmware version
Remark	◆ ADT-8949 has driver library version number and firmware version number. The same firmware version may have different driver library version, so pay attention to the information of both version numbers for problem tracing.

12.1.5 Set the work mode of output pulse

Function	int adt8949_set_pulse_mode(int cardno, int axis, int value,int logic,int dir_logic);
Function description	Set the work mode of output pulse Default mode: pulse + direction, positive logic pulse, positive logic direction signal
Parameter settings	cardno Card number axis Axis number (1-4) value 0: pulse + pulse 1: pulse + direction logic 0: positive logic pulse, 1: negative logic pulse dir-logic 0: positive logic direction signal, 1: negative logic direction signal
Return	◆ 0: Correct

<table>
<tr><td>value:</td><td>◆ Non-0: Wrong</td></tr>
<tr><td>Remark</td><td>
Fig. 1:



Both pulse and direction are positive logic setting

<table>
<tr><th>Pulse output mode</th><th>Driving direction</th><th colspan="2">Output signal wave</th></tr>
<tr><th></th><th></th><th>PU/CW signal</th><th>DR/CCW signal</th></tr>
<tr><td>Independent two pulses mode</td><td>+ direction driving output</td><td>⊓⊓···</td><td>Low level</td></tr>
<tr><td></td><td>- direction driving output</td><td>Low level</td><td>⊓⊓··</td></tr>
<tr><td>One pulse mode</td><td>+ direction driving output</td><td>⊓⊓···</td><td>Low level</td></tr>
<tr><td></td><td>- direction driving output</td><td>⊓⊓···</td><td>High level</td></tr>
</table>

Fig. 2:

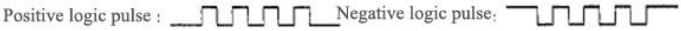
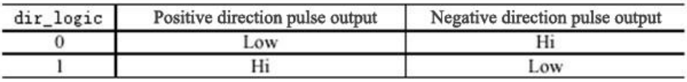


Fig. 3:



<table>
<tr><th>dir_logic</th><th>Positive direction pulse output</th><th>Negative direction pulse output</th></tr>
<tr><td>0</td><td>Low</td><td>Hi</td></tr>
<tr><td>1</td><td>Hi</td><td>Low</td></tr>
</table>
</td></tr>
</table>

12.1.6 Set positive/negative limit signal mode:

Function	int adt8949_set_input_mode(int cardno,int axis, int v1,int v2,int logic);
Function description	Set positive/negative limit mode
Parameter settings	cardno Card number axis Axis number (1-4) v1 0: Positive limit active, 1: Positive limit inactive v2 0: Negative limit active, 1: Negative limit inactive logic 0: Low level active, 1: High level active
Return	0: Correct Non-0: Wrong

value:	
Remark	The default mode is: positive limit active, negative limit active, low level active

12.1.7 Set STOP0 (mechanical home) signal mode:

Function	int adt8949_set_stop0_mode(int cardno,int axis, int v,int logic,int admode);
Function description	Set positive/negative limit mode
Parameter settings	cardno Card number axis Axis number (1-4) v 0: Inactive 1: Active logic 0: Low level active, 1: High level active admode 0: Deceleration stop, 1: Immediate stop
Return value:	0: Correct Non-0: Wrong
Remark	The default mode is: STOP0 inactive

12.1.8 Set STOP1 (servo home) signal mode:

Function	int adt8949`_set_stop1_mode(int cardno,int axis, int v,int logic,int admode);
Function description	Set positive/negative limit mode
Parameter settings	cardno Card number axis Axis number (1-4) v 0: Inactive 1: Active logic 0: Low level active, 1: High level active admode 0: Deceleration stop, 1: Immediate

	stop
Return value:	0: Correct Non-0: Wrong
Remark	The default mode is: STOP1 inactive

12.1.9 Set input signal mode (including positive/negative limit, home):

Function	int adt8949_set_input_mode(int cardno,int axis,int mode,int port,int logic,int admode);
Function description	Set input signal mode (including positive/negative limit, home)
Parameter settings	cardno Card number axis Axis number (1-4) mode 0: Positive limit, 1: Negative limit 2: Home (or encoder Z phase signal) port Input port number quickly (0-18, 36-47), 255or -1: Set corresponding mode to inactive; in addition, the corresponding ports of encoder signals (phase A, B & Z) of the four driving axes: IN36~IN39: A1...A4 IN40~IN43: B1...B4 IN44~IN47: Z1...Z4 logic Active voltage level: 0: Low level active, 1: High level active admode Whether use deceleration when limit is active, 0: deceleration stop; 1: immediate stop
Return value:	0: Correct Non-0: Wrong
Remark	To facilitate the setting of machine limit, home, and servo phase-Z signal, the following functions have been

	packaged: adt8949_set_limit_mode(…), adt8949_set_stop0_mode(…), adt8949_set_stop1_mode(…) The default mode is: Positive limit low level active, negative limit low level active, home signal (or encoder Z-phase signal) inactive; Positive and negative limit temporarily support the immediate stop mode only; The home or encoder Z-phase signal mode used for homing can't be enabled at the same time. You can set homing first and then reset encoder Z-phase.

12.1.10 Set electronic gear ratio of axis:

Function	int adt8949_set_gear(int cardno, int axis,float gear)
Function description	Set the electronic gear ratio of each axis, i.e. the pulses corresponding to 1mm Default: 1000, i.e. 1000 pulses moves 1mm
Parameter settings	cardno Card number axis Axis number (1-4) gear Gear ratio Default: 1000
Return value:	0: Correct Non-0: Wrong
Remark	

12.1.11 Set the filter level of input signal:

Function	int adt8949_set_input_filter(int cardno,int gp,int grade);

Function description	Set the filter level of input signal The default mode is no filter for all input signals.
Parameter settings	cardno Card number gp Input signal types 0: limit, home, common IO 1: encoder signal (A, B, Z) grade Range: 0-15. 0 indicates no filter; if it is set to n, the filter time is 2 $^{(n-1)}$ us
Return value:	0: Correct Non-0: Wrong
Remark	

12.1.12 Set the working mode of actual position counter (encoder input):

Function	int adt8949_set_actual_count_mode(int cardno, int axis, int value,int dir_logic);
Function description	Set the working mode of actual position counter (encoder input) Default mode: A/B phase pulse input, positive direction logic
Parameter settings	cardno Card number axis Axis number (1-6), X-axis number of handwheel is 5, and Y-axis number is 6 value 0: A/B pulse input 1: up/down (PPIN/PMIN) pulse input dir_logic count direction, 0: input signal direction positive logic, 1: input signal direction negative logic
Return value:	0: Correct Non-0: Wrong
Remark	When it is set to A/B phase pulse input, the pulse count is the data by 4 times frequency multiplication

12.1.13 Set the mode of positive/negative limit input nLMT signal and stop signal mode:

Function	int adt8949_set_emergency_stop_mode(int cardno,int port,int logic);
Function description	Set the mode of positive/negative limit input nLMT signal and stop signal mode
Parameter settings	cardno Card number port input port number quickly (0-18, 36-47), 255 or -1: set corresponding mode to inactive logic active voltage level 0: Low level active, 1: High level active
Return value:	0: Correct Non-0: Wrong
Remark	Remark: Emergency stop is inactive by default. Once external emergency stop is active, you need to call adt8949_reset_card(…) to reset the motion card, or else the calling of any motion instruction will be inactive even if the emergency stop active level has been canceled.

12.2 Reset motion card

12.2.1 Reset motion card

Function	int adt8949_reset_card(int cardno);
Function description	Reset motion card
Parameter settings	cardno Card number
Return	0: Correct -1: Wrong

value:	
Remark	After calling this function, the control will clear all cache events and motion instruction data; if synchronized axis has been set, the synchronization relationship of axes will also be cleared, but limit, acceleration, gear ratio and other motion parameters that have been set won't be cleared, and do not need to re-set; When the machine has external emergency stop, positive/negative limit or abnormal stop, or before cache interpolation of large amount of data, it is recommended to call the function to reset the control card.

12.3 Driving status checking

12.3.1 Get the driving status of each axis

Function	int adt8949_get_status(int cardno,int axis,int *v)
Function description	Get the driving status of each axis
Parameter settings	cardno Card number axis Axis number (1-4) v Pointer of driving status 0: Driving ends Non-0: Driving
Return value	0: Correct 1: Wrong
Remark	This function is used to query API. For consecutive query, insert a Sleep(1) sentence between two queries

12.3.2 Get driving status of all axes:

Function	int adt8949_get_status_all(int cardno,int *v)
Function description	Get driving status of all axes
Parameter settings	cardno Card number v Pointer of driving status 0: Driving ends Non-0: Driving
Return value	0: Correct 1: Wrong
Remark	This function is used to query API. For consecutive query, insert a Sleep(1) sentence between two queries

12.3.3 Get driving status of interpolation:

Function	int adt8949_get_inp_status(int cardno,int *v)
Function description	Get driving status of interpolation
Parameter settings	cardno Card number v Pointer of driving status 0: Driving ends Non-0: Driving
Return value	0: Correct 1: Wrong
Remark	

12.4 Motion parameter setting

💣 Note: The following parameters are uncertain after initialized and should be set before using

12.4.1 Set cache segments of interpolation:

Function	int adt8949_set_precount(int cardno,unsigned short prec)
Function	Set cache segments of interpolation; default: 0

description	
Parameter settings	cardno Card number prec Preview segments
Return value	0: Correct 1: Wrong
Remark	

12.4.2 Set S-shaped jerk:

Function	int adt8949_set_jcc(int cardno, int axis,unsigned short jcc);
Function description	Set S-shaped jerk; default: 0
Parameter settings	cardno Card number axis Axis number (1-4, number of interpolation axis: INPA_AXISREG) Jcc Jerk: 0~10
Return value	0: Correct Non-0: Wrong
Remark	◆ The smaller the Jcc value, the more obvious the S acceleration/deceleration effect

12.4.3 Set axis acceleration (adt8949_set_acc) and deceleration (adt8949_set_dec):

Function	int adt8949_set_acc(int cardno, int axis,float add) int adt8949_set_dec(int cardno, int axis,float add)
Function description	Set axis acceleration (set_acc) and deceleration (set_dec) Unit: mm/sec^2 Default: acceleration = deceleration =500mm/sec^2
Parameter	cardno Card number

settings	axis 1~4 single axis INPA_AXISREG interpolation axis add Acceleration/deceleration value (100~100000)
Return value	0: Correct 1: Wrong
Remark	◆ ADT-8949 supports asymmetric acceleration/deceleration; when setting acceleration, deceleration will be equal to the acceleration by default; therefore, to set the deceleration, place set_dec after set_acc, or else the deceleration value will be overwritten by acceleration. ◆ INPA_AXISREG is the axis number of group A interpolator, interpolation acceleration/deceleration, which calculates the synthesized position, is also set independently; therefore, the acceleration/deceleration of interpolation axis is less than or equal to interpolation acceleration/deceleration. #define INPA_AXISREG 0x3f

12.4.4 Set axis acceleration/deceleration mode:

Function	int adt8949_set_admode(int cardno, int axis,unsigned short mode)
Function description	Set axis acceleration/deceleration mode
Parameter settings	cardno Card number axis Axis number (1-4, number of interpolation axis: INPA_AXISREG) mode Range (0-3) 0 S-shaped acceleration/deceleration mode

	1 trapezoidal acceleration/deceleration mode 2 exponential acceleration/deceleration mode 3 trigonometric acceleration/deceleration mode
Return value	0: Correct 1: Wrong
Remark	Remark: The default option is trapezoidal acceleration/ deceleration mode Point motion and single linear interpolation can use any mode, Single arc interpolation uses mode 1 and 3. Spline interpolation uses mode 1. To use non-trapezoidal acceleration/deceleration mode in interpolation, ensure that the segment of pretreatment cache is zero.

12.4.5 Set axis running speed, start speed and end speed in sequence:

Function	int adt8949_set_speed(int cardno, int axis,float speed);
	int adt8949_set_startv(int cardno, int axis,float speed);
	int adt8949_set_endv(int cardno, int axis,float speed);

Function description	Set axis running speed, start speed and end speed in sequence
Parameter settings	cardno Card number axis 1~4 single axis INPA_AXISREG interpolation axis speed Speed, unit: mm/sec,0.001~100000
Return value	0: Correct Non-0: Wrong
Remark	◆ If endv is set before startv, default endv= startv

12.4.6 Set the maximum axis speed at the connection of two line segments in cache interpolation:

Function	int adt8949_set_speed_constraint (int cardno, int axis,float speed);
Function description	Set speed constraint at motion path connection
Parameter settings	cardno Card number axis 1~4 single axis INPA_AXISREG interpolation axis speed Speed, unit: mm/sec,0.001~100000
Return value	0: Correct Non-0: Wrong
Remark	◆ This function doesn't need to be called generally. ◆ In special process, it is used to set the maximum axis speed at the connection of two line segments in cache interpolation; it is active in pretreatment, and limits the maximum speed of each axis at the connection of line segments.

12.4.7 Set the acceleration constraint of axis in cache interpolation; active in pretreatment, limit the maximum speed change of each axis at the connection of line segments:

Function	int adt8949_set_acc_constraint(int cardno, int axis,float add);
Function description	Set acceleration constraint at motion path connection
Parameter settings	cardno Card number axis Axis number (1-4)

	add Range (100-100000), default: 500mm/sec^2
Return value	0: Correct Non-0: Wrong
Remark	◆ In cache interpolation, it is used to set the acceleration constraint of axis; it is active in pretreatment, and limits the maximum speed change of each axis at the connection of line segments. ◆ In the debugging process, constraint value can be gradually increased as long as the interpolation accuracy won't be affected, which will reduce machine shake in interpolation.

12.4.8 Set arc speed clamp:

Function	int adt8949_set_arc_speed_clamp(int cardno, float radius,float speed);
Function description	When using cache interpolation, set axis acceleration constraint active in pretreatment, and limit the maximum speed changes of each axis at the connection of line segments
Parameter settings	cardno Card number radius Radius coefficient speed Speed coefficient, range (0.01-100000 mm/sec) Default Radius coefficient = 10mm, speed coefficient = 100mm/sec
Return value	0: Correct Non-0: Wrong
Remark	◆ The function will be active in flat arc or spatial arc interpolation; if the function is called and the actual radius is smaller than the radius coefficient, the arc

	speed is less limited; ◆ If the actual radius is larger than the radius coefficient, the arc speed limit is greater. If the actual radius is equal to the radius coefficient, the maximum speed of arc is equal to the speed coefficient.

12.4.9 Set logic position of axis pulse:

Function	int adt8949_set_command_pos(int cardno, int axis,long pos)
Function description	Set logic position of axis pulse
Parameter settings	cardno Card number axis 1~4 single axis pos Logic position Set range (-2147483648~+2147483647)
Return value	0: Correct Non-0: Wrong
Remark	◆

12.4.10 Set actual position of axis pulse:

Function	int adt8949_set_actual_pos(int cardno, int axis,long pos);
Function description	Set actual position
Parameter settings	cardno Card number axis Axis number (1-6), X-axis number of handwheel is 5, and Y-axis number is 6 pos Range (-2147483648~+2147483647)

Return value	0: Correct -1: Wrong
Remark	◆

12.4.11 Axis cache position and logic position synchronization:

Function	int adt8949_set_syncpos(int cardno, int axisbit)
Function description	Axis cache position and logic position synchronization
Parameter settings	cardno Card number axisbit Axis mapping bit
Return value	0: Successful; non-0: Failed
Remark	Set after zeroing and tool setting and before one group of interpolation motion to improve accuracy. Do not set when the machine is in motion.

12.4.12 Set number of pretreatment cache segments:

Function	int adt8949_set_precount(int cardno,unsigned short prec);
Function description	Set number of pretreatment cache segments
Parameter settings	cardno Card number prec Number of cache segments
Return value	0: Successful; non-0: Failed
Remark	

12.4.13 Set follow axis:

Function	intadt8949_set_follow_axis(int cardno,int slaveaxis,int masteraxis);
Function description	Set follow axis
Parameter settings	cardno Card number slaveaxis Slave axis (following master axis), Axis number (1-4) masteraxis Master axis (followed by slave axis), Axis number (0-4), 0: cancel follow
Return value	0: Successful; non-0: Failed
Remark	

12.4.14 Set total speed rate:

Function	int adt8949_set_rate1(int cardno,float rate);
Function description	Set total speed rate
Parameter settings	cardno Card number rate Rate (0~2.0)
Return value	0: Successful; -1: Failed
Remark	Refresh immediately; if the rate of change is too large, it will cause speed step; the ideal method is to set timing gradually to produce a deceleration effect; when the rate is set to 0, the effect is equivalent to suspend. ◆ ADT-8949 does not support online speed change. The value is to change the speed dynamically in the concept of rate, and acceleration/deceleration also will be changed accordingly. The acceleration/deceleration mode will not be affected, that is, when S-shaped acceleration/deceleration is

	selected, the acceleration curve is still S shape even if the speed rate is set to the highest. ◆ All axes are affected rate1 in all modes; when rate1 is set to 0, the effect is equivalent to suspend. ◆ Rate1 refreshes immediately; so if the rate of change is too large, it will cause speed step; the ideal method is to set timing gradually to produce a deceleration effect. The impact of rate1 on actual speed also depends on rate2,; the actual axis speed = rate1 * rate2 [axis] * speed;

12.4.15 Set speed rate of single axis:

Function	int adt8949_set_rate2(int cardno,int axis ,float rate);
Function description	Set speed rate of each axis
Parameter settings	cardno Card number axis Axis number :Ax,Ay,Az,Aa,INPA_AXISREG, INPB_AXISREG rate Rate (0~2.0)
Return value	0: Successful; -1: Failed
Remark	Remark: After calling this function, the speed rate of the axis will refresh immediately, so if the rate of change is too large, it will cause speed step; the ideal method is to set timing gradually to produce a deceleration effect. ◆ It has same effect as rate1, but the scope is only limited to the specified axis number. ◆ Rate2 also refreshes immediately, so pay attention to speed step; set update gradually according to the

	note of rate1. ◆ The impact of rate2 on the actual speed of the axis is rate1 * rate2 [axis] * speed Through multiple proportions of rate1 and rate2, the speed can be increased to four times of the set speed, and the minimum is 0.

12.5 Parameter checking

The following functions can be called at any time

12.5.1 Get the logic position of each axis:

Function	int adt8949_get_command_pos(int cardno,int axis,long *pos)
Function description	Get the pulse count value sent by the card
Parameter settings	cardno Card number axis 1~4 single axis pos Pulse count
Return value	0: Correct Non-0: Wrong
Remark	This function can get the logic position of the axis at any time; in case that the motor isn't out of step, pos value is the current position of the axis.

12.5.2 Get the value of motor AB phase encoder feedback counter:

Function	int adt8949_get_actual_pos(int cardno,int axis,long *pos)
Function description	Get the actual position of each axis
Parameter settings	cardno Card number axis Axis number (1-6), X-axis number of handwheel is 5, and Y-axis number is 6 pos Pointer of actual position
Return value	0: Correct -1: Wrong
Remark	

12.5.3 Get current running speed of the axis (instantaneous speed):

Function	int adt8949_get_speed(int cardno,int axis,float *speed);
Function description	Get current running speed of the axis (instantaneous speed)
Parameter settings	cardno Card number axis 1~4 single axis INPA_AXISREG interpolation axis speed Instantaneous speed, unit: mm/sec
Return value	0: Correct Non-0: Wrong
Remark	◆ If the number of feed axis is INPA_AXISREG, the feedback is resultant speed;

12.5.4 Query number of left segments in 10000 segments of interpolation cache zone:

Function	int adt8949_get_fifo_len(int cardno,int *len);
Function description	Query margin in interpolation cache zone
Parameter settings	cardno Card number len Cache length variable to be gotten
Return value	0: Correct Non-0: Wrong
Remark	The interpolation cache zone has 10000 segments in total. Interpolation data and cache events are stored in different regions of the control card. An arc occupies four segments of cache space, and a full circle occupies eight segments of cache space. It is recommended to issue motion data when the cache margin is greater than 8 segments. This function is used to query API. For consecutive query, insert a Sleep(1) sentence between two queries.

12.5.5 Get arc length of two axes:

Function	int adt8949_get_arc2_length(unsigned char arcmap,float pos[4],float Center[4],int dir,float *length);
Function description	Get arc length of two axes
Parameter settings	axismap Axis selection mapping flag, mark any two axes in the space for plane arc interpolation pos Target point coordinates of the arc (relative to the current point) Center Center coordinates of the arc (relative to the current point) dir Arc direction (1: clockwise; 0: counterclockwise)

	length Arc length of two axes
Return value	0: Correct Non-0: Wrong
Remark	

12.5.6 Get arc length of three axes:

Function	int adt8949_get_arc3_length(unsigned char arcmap,float pos2[4],float pos3[4],float *length);
Function description	Get arc length of three axes
Parameter settings	arcmap Axis selection mapping flag; up to three positions can be marked as 1, that is, support arc interpolation of up to three axes pos2 Second point coordinates of the arc (relative to the current point) pos3 Third point coordinates of the arc (relative to the current point) length Arc length of three axes
Return value	0: Correct Non-0: Wrong
Remark	

12.5.7 Get the latest error number of the system:

Function	int adt8949_get_syserr(int cardno,int *ErrNum);
Function description	Get the latest error number of the system
Parameter settings	cardno Card number ErrNum Pointer of the system error number
Return value	Remark: Get system error number regularly, and view the running of the motion card
Remark	◆

12.5.8 Get stop data of each axis:

Function	int adt8949_get_stopdata(int cardno,int axis,int *value);
Function description	Function: Get stop data of each axis
Parameter settings	cardno Card number axis Axis number (1-4) value Pointer of stop data (0: no error stop; non-0: has limit, home or encoder Z phase signal triggers stop): bit0==1: positive limit triggers stop bit1==1: negative limit triggers stop bit2==1: home signal triggers stop bit3==1: encoder Z phase signal triggers stop bit4==1: external emergency stop signal triggers stop
Return value	0: Correct -1: Wrong
Remark	◆ Remark: Stop data may appear in combination, e.g. if both bit0 and bit1 are 1, positive limit and negative limit are triggered, resulting in axis stop ◆ For consecutive query, insert a Sleep(1) sentence between two queries, or else it will affect the efficiency of the control card.

12.6 Driving

12.6.1 Single axis quantitative motion:

Function	int adt8949_pmove(int cardno,int axis,float pos)
Function	Single axis quantitative relative motion (PTP)

description	Set the relative position of axis motion, unit: mm
Parameter settings	cardno Card number axis Axis number (1~4) pos >0: positive motion <0: negative motion Range (+/- 9999999.999mm)
Return value	0: Correct Non-0: Wrong state
Remark	◆ Before writing drive command, be sure to set the acceleration/deceleration parameters required by the speed curve properly

12.6.2 Absolute coordinates quantitative driving:

Function	int adt8949_abs_pmove(int cardno,int axis,float pos)
Function description	Absolute coordinates quantitative driving, unit: mm
Parameter settings	cardno Card number axis Axis number (1-4) pos Unit: mm, (+/- 9999999.999) >0: positive driving, <0: negative driving
Return value	0: Correct Non-0: Wrong state
Remark	

12.6.3 Single axis continuous driving:

Function	int adt8949_continue_move(int cardno,int axis,int dir);
Function description	Absolute coordinates quantitative driving, unit: mm
Parameter	cardno Card number

settings	axis Axis number (1-4) pos Unit: mm, (+/- 9999999.999) >0: positive driving, <0: negative driving
Return value	0: Correct Non-0: Wrong state
Remark	Note: Before writing drive command, be sure to set the speed parameters properly

12.6.4 Two-axis arc interpolation:

Function	int adt8949_inp_arc2(int cardno,unsigned short index,unsigned char arcmap,float pos[4],float Center[4],int dir);
Function description	Two-axis arc interpolation, unit: mm
Parameter settings	cardno Card number index Data index, used to identify the data of the motion, generally set to 0 to axismap Axis selection mapping flag, mark any two axes in the space for plane arc interpolation pos Target point coordinates of the arc (relative to the current point) Center Center coordinates of the arc (relative to the current point) dir Arc direction (1: clockwise; 0: counterclockwise)
Return value	0: Correct Non-0: Wrong state
Remark	Axismap: axis mapping. AZYX correspond to binary. 0011 indicates that X, Y axis participate in two-axis arc interpolation, 0110 indicates that Y, Z axis participate in two-axis arc interpolation

12.6.5 Three-axis arc interpolation instruction:

Function	int adt8949_inp_arc3(int cardno,int arcmap,float pos2[4],float pos3[4])
Function description	Three-axis arc interpolation instruction
Parameter settings	cardno Card number arcmap Arc participating in axes mapping mark pos2[4] Coordinates of second point (relative to current point) pos3[4] Coordinates of third point (relative to current point)
Return value	0: Correct Non-0: Wrong
Remark	◆ arcmap indicates arc axis, bit0 indicates 1# axis, bit1 indicates 2# axis, and so on. Up to three-axis flags appear; ◆ pos2, pos3 are coordinates of second and third point.

12.6.6 Four-axis interpolation instruction (absolute position):

Function	int adt8949_inp_abs_move4(int cardno,int axismap,float pos1,float pos2,float pos3,float pos4)
Function description	Set four-axis interpolation instruction (absolute position)
Parameter settings	cardno Card number axismap Axis mapping bit, bit0 indicates 1# axis, bit1 indicates 2# axis pos1,pos2 pos3,pos4 Absolute position of the motor
Return value	0: Correct Non-0: Wrong

Remark	◆ The interpolation instruction will be cached. Since the control card has 5000 segments of cache, the instruction can be returned soon; if you want to determine whether the motion ends, combine the interpolation status to determine. ◆ Axismap is used to identify the control axis; if the corresponding bit is marked as 0, it won't be processed regardless of pos value, and the card will automatically take the current position. ◆ If an axis shift is set to 0, the axis will not be occupied by interpolator, that is, the position can also be driven by PTP. ◆ Similarly, if the set axis number has been occupied by PTP, the interpolation command will fail, all cache instructions including previously fed instructions will be invalid and should be re-computed and fed into the interpolator. ◆ If the return value isn't zero, search the content of error through error status table; if the cache is full, it will also return an error.

12.6.7 Four-axis interpolation instruction (relative position):

Function	int adt8949_inp_move4(int cardno,float pos1,float pos2,float pos3,float pos4)
Function description	Set four-axis interpolation instruction (relative position)
Parameter settings	cardno Card number pos1,pos2,pos3,pos4 shift relative to current point
Return value	0: Correct Non-0: Wrong
Remark	◆ The interpolation instruction will be cached. Since

	the control card has 5000 segments of cache, the instruction can be returned soon; if you want to determine whether the motion ends, combine the interpolation status to determine. ◆ If an axis shift is set to 0, the axis will not be occupied by interpolator, that is, the position can also be driven by PTP. ◆ Similarly, if the set axis number has been occupied by PTP, the interpolation command will fail, all cache instructions including previously fed instructions will be invalid and should be re-computed and fed into the interpolator. ◆ If the return value isn't zero, search the content of error through error status table; if the cache is full, it will also return an error.

12.6.8 Four-axis relative coordinates linear interpolation (specifying motion time)

Function	int adt8949_time_move4(int cardno,unsigned short index,float pos1,float pos2,float pos3,float pos4,float time);
Function description	Set four-axis interpolation instruction (relative position)
Parameter settings	cardno Card number index Data index, used to identify the data of the motion, generally set to 0 to pulse1,pulse2,pulse3,pulse4 indicate the relative distance of axis XYZW time Time required by motion of the straight line, unit: ms

Return value	0: Correct Non-0: Wrong
Remark	Note: The straight line moves at constant speed within the specified time. Before calling this function, ensure that the number of preprocessing segments is zero, or else the call will fail; If synchronized axis is set, pos parameter of slave axis should be set to 0, or else the call will fail.

12.6.9 Four-axis absolute coordinates linear interpolation (specifying motion time)

Function	int adt8949_time_abs_move4(int cardno,unsigned short index,unsigned char axismap,float pos1,float pos2,float pos3,float pos4,float time);
Function description	Four-axis absolute coordinates linear interpolation (specifying motion time)
Parameter settings	cardno Card number index Data index, used to identify the data of the motion, generally set to 0 to axismap Axis mapping bit, bit0 indicates 1# axis, bit1 indicates 2# axis; if the axis is not marked, the target position will not take effect. pulse1,pulse2,pulse3,pulse4 indicate the coordinates that axis XYZW move to time Time required by motion of the straight line (non-cumulative time), unit: ms
Return value	0: Correct Non-0: Wrong
Remark	Note: The straight line moves at constant speed within the specified time. Before calling this function, ensure that the number of preprocessing segments is zero, or

	else the call will fail. If synchronized axis is set, pos parameter of slave axis should be set to 0, and axismap value doesn’t need to consider the slave axis position.

12.6.10 NURBS interpolation

Function	int adt8949_inp_NURBS(int cardno,unsigned short index,unsigned char axismap,float conp[][4],float weight[],float node[],int nodenum);
Function description	NURBS interpolation
Parameter settings	cardno Card number index Data index, used to identify the data of the motion, generally set to 0 to Axismap Axis selection mapping flag, only three positions can be marked to 1, that is, support NURBS interpolation up to three axes conp Spline control point (relative to the current point, the coordinates of first control point should be 0, or else it will return error) Weight Weight corresponding to control point of spline. The weight of each control point of AutoCAD is -1 by default node Node value of spline nodenum Number of spline nodes
Return value	0: Correct Non-0: Wrong
Remark	Remark: For NURBS interpolation, it is recommended to set a bigger number of pre-processing cache segments and use trapezoidal acceleration/deceleration mode; calling the function to set arc speed clamp will affect planning of spline interpolation speed;

	Using spline interpolation will occupy a larger number of cache segments of control card. To continue to call interpolation instruction, check the remaining cache capacity in advance.

12.6.11 Set deceleration stop for manual intervention of axis

Function	int adt8949_dec_stop(int cardno,int axis)
Function description	Set deceleration stop for manual intervention of axis Deceleration is the effective value in the last setting Manual intervention deceleration is allowed in linear mode only
Parameter settings	cardno Card number axis Axis number (1~4)
Return value	0: Correct Non-0: Wrong state
Remark	◆ If the set axis participates in the interpolation, other corresponding axes in the interpolation will be decelerated to stop.

12.6.12 Set immediate stop for manual intervention of axis

Function	int adt8949_sudden_stop(int cardno,int axis)
Function description	Set immediate stop for manual intervention of axis
Parameter settings	cardno Card number axis Axis number (1~4)
Return value	0: Correct Non-0: Wrong state
Remark	◆ If the set axis participates in the interpolation, other corresponding axes in the interpolation will be decelerated to stop.

12.7 Switch quantity

12.7.1 Read IO status and read/write IO

Function	int adt8949_get_out(int cardno, int number) int adt8949_read_bit(int cardno,int number) int adt8949_write_bit(int cardno,int number,int value)
Function description	Get output IO status (gadt8949_et_out) Get input IO status (adt8949_read_bit) Set output IO status (adt8949_write_bit)
Parameter settings	cardno Card number number Corresponding IO number, counting from 0
Return value	-1 Get error 0 OFF state 1 ON state
Remark	

12.7.2 Read IO status by groups

Function	int adt8949_get_gpio(int cardno,int gp,short int *map);
Function description	Read IO status by groups
Parameter settings	cardno Card number gp Group number (the number of OUT0~OUT15 is 0, the number of IN0~IN15 is 0x20, the number of IN16~IN31 is 0x21) map Status of this group of IO (determine the status of certain IO through bit value, e.g. OUT0 corresponds to bit0, OUT15 corresponds to bit15)
Return value	0: Low voltage level Non-0: Wrong
Remark	

12.7.3 Operate output by group

Function	int adt8949_set_gpio(int cardno,int gp,short int map);
Function description	Operate output by group
Parameter settings	Card number gp Group number (the number of OUT0~OUT15 is 0; only output point settings of group are available) iomap Specify the output point to be operated by bit (bit0~bit15); if the bit value is 1, operate corresponding output point; if the bit value is 0, output point isn’t affected levelmap Voltage level setting of this group of IO (determine the status of certain output point through bit value, e.g. OUT0 corresponds to bit0, OUT15 corresponds to bit15); only the output points with iomap bit value 1 are affected
Return value	0: Correct -1: Wrong
Remark	

12.7.4 Set voltage level for multiple output points at the same time

Function	int adt8949_set_multi_io(int cardno,int gp,short int iomap,short int levelmap);
Function description	Set voltage level for multiple output points at the same time
Parameter settings	cardno Card number gp Group number (the number of

	OUT0~OUT15 is 0; only output point settings of group are available) iomap Specify the output point to be operated by bit (bit0~bit15); if the bit value is 1, operate corresponding output point; if the bit value is 0, output point isn’t affected levelmap Voltage level setting of this group of IO (determine the status of certain output point through bit value, e.g. OUT0 corresponds to bit0, OUT15 corresponds to bit15); only the output points with iomap bit value 1 are affected
Return value	0: Low voltage level Non-0: Wrong
Remark	

12.8 FIFO operation output

12.8.1 Single point IO output in interpolation:

Function	int adt8949_set_fifo_io(int cardno,int number,int value,float speed);
Function description	Single point IO output in interpolation
Parameter settings	cardno Card number number Output point (0-14) value 0: Low level 1: High level speed -1: no speed constraint before motion by default, other: range (0.0-100000.0 mm/sec)
Return value	0: Low voltage level Non-0: Wrong
Remark	Speed setting is same as the motion track speed, and motion track won’t decelerate. If the speed setting is

	smaller, the motion track will decelerate to the set speed in advance before corresponding IO operation, and then accelerate to motion track speed, and form V-shaped process.

12.8.2 Set voltage level for multiple output points at the same time in interpolation:

Function	int adt8949_set_fifo_multi_io(int cardno,int gp,short int iomap,short int levelmap,float speed);
Function description	Set voltage level for multiple output points at the same time in interpolation
Parameter settings	ccardno Card number gp Group number (the number of OUT0~OUT14 is 0; only output point settings of group are available) iomap Specify the output point to be operated by bit (bit0~bit15); if the bit value is 1, operate corresponding output point; if the bit value is 0, output point isn't affected levelmap Voltage level setting of this group of IO (determine the status of certain output point through bit value, e.g. OUT0 corresponds to bit0, OUT15 corresponds to bit15); only the output points with iomap bit value 1 are affected speed -1: no speed constraint before motion by default, other: range (0.0-100000.0 mm/sec)
Return value	0: Low voltage level Non-0: Wrong
Remark	Speed setting is same as the motion track speed, and motion track won't decelerate. If the speed setting is

	smaller, the motion track will decelerate to the set speed in advance before corresponding IO operation, and then accelerate to motion track speed, and form V-shaped process.

12.8.3 Specific position delay motion in interpolation:

Function	int adt8949_set_fifo_delay(int cardno,int millisecond);
Function description	Specific position delay motion in interpolation
Parameter settings	cardno Card number millisecond delay time, unit: ms
Return value	0: Correct Non-0: Wrong
Remark	

12.8.4 Insert pulse generator in interpolation:

Function	int adt8949_set_fifo_pulser(int cardno,int port,int NormalLevel,int NormalTime,int UnNormalTime,int ReverseNum,float speed);
Function description	Insert pulse generator in interpolation
Parameter settings	cardno Card number port Output point port of pulse generator (0-14) NormalLevel Normal state voltage level 0: Low, 1: High NormalTime Holding time of normal level, unit: ms UnNormalTime Holding time of non-normal

	level, unit: ms ReverseNum Reverse times of output level speed -1: no speed constraint before motion by default, other: range (0.0-100000.0 mm/sec)
Return value	0: Correct Non-0: Wrong
Remark	Speed setting is same as the motion track speed, and motion track won't decelerate. If the speed setting is smaller, the motion track will decelerate to the set speed in advance before corresponding IO operation, and then accelerate to motion track speed, and form V-shaped process.

12.9 Handwheel function

12.9.1 Set handwheel mode

Function	int adt8949_set_hand_wheel_mode(int cardno,int axis,int fun_mode);
Function description	Set handwheel mode
Parameter settings	cardno Card number axis Axis number (5-6), X-axis number of handwheel is 5, and Y-axis number is 6 fun_mode Function mode, 0: General input port mode; 1: Handwheel encoder signal input mode
Return value	0: Correct -1: Wrong
Remark	Remark: After using handwheel function, set to slow input point mode in time, or else it will affect the efficiency of the wiring board. Slow input point mode is default. After setting to handwheel encoder signal input mode, the default mode is PPIN/PMIN pulse input and input signal direction

	positive logic; to modify encoder input signal type or direction logic, please call the function adt8949_set_actual_count_mode(…).

12.10 DA output

12.10.1 Set DA output voltage:

Function	int adt8949_set_daout(int cardno,int port,int value);
Function description	Set DA output voltage
Parameter settings	cardno Card number port Set DA output port (1-2) value Set DA output size (0-10), unit: V
Return value	0: Correct Non-0: Wrong
Remark	DA output voltage is accurate to two decimal places

13.11 Homing module

13.11.1 Set home signal mode:

Function	adt8949_SetHomeMode_Ex(int m_nCardNum,int m_nAxisNum,int m_nHomeDir, int m_nStop0Active,int m_nLimitActive,int m_nStop1Active,long m_nBackRange,long m_nEncoderZRange,long m_nOffset);
Function description	Set home signal, step parameter
Parameter settings	m_nCardNum Card number m_nAxisNum Axis number m_nHomeDir Home direction, 0: negative direction, 1: positive direction

	m_nStop0Active stop0 active level setting; 0: Low level stop; 1: High level stop m_nLimitActive limit signal active level setting; 0: Low level stop; 1: High level stop m_nStop1Active stop1 active level setting; 0: Low level stop; 1: High level stop m_fBackRange Reverse distance >1, shouldn't exceed the distance between positive limit and stop0 m_fEncoderZRange Encoder Z phase range >1 m_fOffset Home offset; ==0: No, >0: positive direction, <0: negative direction
Return value	0: Correct -1~-8: Wrong type -1: Parameter error -2: Parameter error -3: Parameter error -4: Parameter error -5: Parameter error -6: Parameter error -7: Parameter error

13.11.2 Homing speed parameter setting:

Function	adt8949_ SetHomeSpeed _Ex(int m_nCardNum,int m_nAxisNum,long m_nStartSpeed,long m_nSearchSpeed,long m_nHomeSpeed,long m_nAcc,long m_nZPhaseSpeed);
Function description	Set home signal, step parameter
Parameter	m_nCardNum Card number

settings	m_nAxisNum Axis number m_nStartSpeed Home (STOP0) search start speed m_nSearchSpeed Home search speed m_nHomeSpeed Slow homing speed m_nAcc Acceleration in homing process m_nZPhaseSpeed Encoder Z phase (STOP1) search speed
Return value	0: Correct -1~-8: Wrong type -x: x parameter error

13.11.2 Start homing:

Function	adt8949_ HomeProcess _Ex(int m_nCardNum,int m_nAxisNum,float m_fGear);
Function description	Set home signal, step parameter
Parameter settings	m_nCardNum Card number m_nAxisNum Axis number m_fGear Electronic gear ratio:(1-10000)
Return value	0: Correct -1~-3: Parameter error -x: x parameter error

13.11.4 Get home status:

Function	adt8949_ GetHomeStatus _Ex(int m_nCardNum,int m_nAxisNum);
Function description	Get home status, step parameter
Parameter	m_nCardNum Card number

settings	m_nAxisNum Axis number m_fGear Electronic gear ratio: (1-10000)
Return value	0: Home successfully; -1: Parameter 1 error; -2: Parameter 2 error; -3: Homing isn't started; (1-10) steps 1: Approach home fast and search STOP0 2: Check if STOP0 is found 3: Exit home reversely 4: Check if reverse exit is complete 5: Approach home slowly and search STOP0 6: Check if STOP0 searching completes 7: Approach Z phase slowly, and search STOP1. If STOP1 is set to -1, skip step 7 & 8. 8: Check if STOP1 searching completes 9: Home offset 10: Check home offset -100x: The x homing step is abnormal, e.g.: -1001: the first homing step is abnormal -1020: Homing is terminated

Chapter 13 Troubleshooting

13.1 Motion control card detections fails

If the control card can be detected in the process of using the control card, please follow the method below to eliminate the problem.

(1) Be sure to follow the installation instructions of the control card to install the drivers step by step, and make sure that the system directory (system32 or System) has the control card dll file;

(2) Check if the motion control card and the socket contact properly. You can re-insert or replace the slot to test, or clean the gold finger of the control card with an eraser before test;

(3) In System Device Manager, check if the motion control card conflicts with other hardware. When using PCI cards, remove other boards and cards, such as: sound card and network card; PC104 card allows adjusting DIP switch to reset the base address; the base address used for card initialization in the program must be same as the actual base address;

(4) Check if the operating system has any problem by re-installing other operating systems;

(5) If the motion control card still can't be detected after examination with the above steps, please replace the motion control card, and further test if the motion control card has been damaged;

13.2 Motor running abnormal

If the motion control card is normal, but the motor is abnormal, refer to the following troubleshooting method.

(1) The motor doesn't run when the motion control card sends pulses

- ➢ Check if the control card and the terminal board are connected properly;
- ➢ Check if the pulse and direction signal cables of the motor drive are properly connected to the terminal board;
- ➢ Check if the external power of the servo drive has been connected properly;
- ➢ Check if the servo and stepper motor drive have alarm; if yes, check the reason according to the alarm code;
- ➢ Check if servo SON is connected properly, and if the servo motor has excitation status;
- ➢ For servo motor, please check the drive control; the control card supports "position control mode";
- ➢ Motor or drive is damaged

(2) The stepper motor screams in operation, motor is significantly out of step.

- Controller speed is too high; please calculate the motor speed, 10~15 rotations per second is normal for stepper motor;
- Mechanical part is stuck or too much resistance;
- Motor selection is inappropriate; please replace a motor of high torque;
- Check the drive current and voltage, set the current to 1.2 times the rated current of the motor, and set the supply voltage within the rated range of the drive;
- Check the initial speed of the controller, which is generally 0.5~1; deceleration time is 0.1 second or more;

(3) Servo or stepper motor has significant vibration or noise during processing

- Position loop gain and speed loop gain of servo drive are too large, reduce the position loop gain and speed loop gain of the servo drive in permitted positioning accuracy;
- Machine rigidity is poor; please adjust the structure of the machine;
- Stepper motor selection is inappropriate; please replace a motor of high torque;
- The speed of stepper motor is in the resonance region of the motor, please avoid the resonance region or increase the subdivision;

(4) Motor positioning inaccurate

- Check if the mechanical screw pitch and number of pulses per revolution of the motor and are consistent with the parameters set by the practical application system, i.e. pulse equivalent;
- For servo motor, increase the position loop gain and speed loop gain;
- Check the screw gap of the machine, test the backlash of lead screw with a dial gauge, and adjust the screw if there is gap;
- If the time and position of inaccurate positioning are indefinite, check the external interference signal;
- Motor selection is inappropriate, and shake or out of step occurs in motion;

(5) The motor has no direction

- Check if the DR+ and DR- wiring have error, and are connected securely;
- Check if the pulse mode of the control card is consistent with the actual drive mode; the control card supports “pulse + direction” and “pulse + pulse” mode
- For stepper motor, check if the motor lines have breakage or poor contact;

13.3 Switch input abnormal

In the process of system commissioning and operation, if some input signals are abnormal, please use the method described below to check.

(1) No signal input

- Check if the wiring is correct according to the wiring diagram of common switch and proximity switch, and ensure that the "optocoupler common terminal" of input signal has been connected to the positive terminal of internal or external power supply (+ 12V or 24V);
- The I\O point input switch of the company is NPN type; if not available, please check the switch model and wiring;
- Check if the optical coupler has been damaged. If the line is normal, the input state does not change when the input point is opened and closed; please use a multimeter to test if the optical coupler has been broken down, and solve the breakdown problem by replacing the optical coupler;
- Check if 12V or 24V switching power supply is normal;
- The switch is damaged;

(2) Signal is intermittent

- Check if there is interference, and test the signal status in I\O test screen; if interference exists, add model 104 monolithic capacitor or use shielded lines;
- The machine has significant shake or unusual stop during normal operation; please check if the limit switch signal has interference or if the limit switch has reliable performance;
- Check if external wiring is in proper contact;

(3) Inaccurate homing

- Speed too fast; reduce homing speed;
- External signal interference; check the sources of interference;
- Homing direction error;
- Homing switch is installed in improper position or switch is loose;

(4) Invalid limit

- In I\O test, check if the limit switch is valid;
- Manual or automatic processing speed is too fast;
- External signal interference; check the sources of interference;
- Manual direction error;
- Limit switch is installed in improper position or switch is loose;

13.3 Switch output abnormal

Switch output is abnormal; please troubleshoot in the method described below.

(1) Output abnormal

- Check if the lines are correct according to the wiring diagram of the output points described earlier, and ensure that the common output terminal (ground wire) is connected to the ground wire of the power supply;
- Check if the output device has been damaged;
- Check if the optical coupler has been damaged; please use a multimeter to test if the optical coupler has been broken down, and solve the breakdown problem by replacing the optical coupler;
- Security essentials; when the output uses inductive load, connect a freewheeling diode (IN4007 model or IN4001) in parallel;

(2) Method to determine output problem

Disconnect the external wiring of output point, connect a 10K pull-up resistor from the output point to the power supply terminal; connect the output ground wire to GND terminal of the power, put the red pen of the multimeter on the 12V positive pole, put the black pen on the output terminal, and tap the button on the test screen to check if there is voltage output; if yes, check external circuit; if not, check if the common terminal of the card is connected properly, and if the optical coupler has problem;

13.4 Encoder abnormal

If the encoder is abnormal while operating, please check in the method below.

(1) Check encoder wiring. Make sure that the wiring of encoder complies with differential or collector open circuit mode previously introduced;

(2) Check encoder voltage. The motion control card accepts +5V signal. If +12V or +24V encoder is used, please connect 1K (+12V) resistor in serial between A/B phase of the encoder and A/B phase of terminal board;

(3) Encoder counting is inaccurate. The external wire of the encoder must be shielded twisted pair, and the encoder wire should be at least 30~50MM away from the strong electric wires that have strong interference;