

---

ADT-8940  
4-Axis Movement Control Card

# User Manual



**Adtech (Shenzhen) CNC Technology Co., Ltd**

Add: F5, Block 36, Majialong Industrial park, Nanshan District, ShenZhen

Postal Code: 518052 Tel: 0755-26722719 (20 lines) Fax: 0755-26722718

## **Copyrights**

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

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

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

## **Trademark**

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

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

***All copyrights reserved by Adtech (Shenzhen) CNC Technology Co., Ltd***

Version Upgrading Instruction

Version Number	Modification Date	Instruction
1.0	2009/4/8	The First Version

Remarks: the meanings of the three numbers in the version number are as follows:



Bank Main Version Number/ Bank Secondary Version Number/ Reservation

**Notes: the above version table only refers to the version updating of the modification of the instruction**

## Version Upgrading Record

Version	Revised in	Descriptions
V110	2006/03	The initial version
V220	2007/03	Optimized hardware performance, supplement Stop and Delay functions; Cancelled Range Setting function, and all the statements in this User Manual involved with range setting functions or calculation of speed or acceleration through range only apply for V110; actually there are no more relationship between range and calculation of speed or acceleration; Added application library function

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



Hardware version number



Major version number



Minor version number

# Contents

<b>Chapter 1 General information</b> .....	<b>6</b>
☞ INTRODUCTION .....	6
☞ MAIN FUNCTIONS.....	6
☞ APPLICATION SCOPE.....	7
<b>Chapter 2 Hardware installation</b> .....	<b>8</b>
☞ PARTS.....	8
☞ INSTALLATION .....	8
<b>Chapter 3 Electrical connection</b> .....	<b>9</b>
☞ P1 LINE .....	10
☞ P3 LINE .....	12
☞ P2 LINE .....	14
☞ CONNECTION FOR PULSE/ DIRECTION INPUT SIGNAL .....	15
☞ CONNECTION FOR ENCODER INPUT SIGNAL .....	16
☞ CONNECTION FOR DIGITAL INPUT.....	17
☞ CONNECTION FOR DIGITAL OUTPUT .....	19
<b>Chapter 4 Software installation</b> .....	<b>21</b>
☞ DRIVE INSTALLATION UNDER WIN2000.....	21
☞ DRIVE INSTALLATION UNDER WINXP .....	24
<b>Chapter 5 Functions</b> .....	<b>27</b>
☞ PULSE OUTPUT METHOD .....	27
☞ HARDWARE RESTRICTION SIGNAL .....	27
☞ STRAIGHT LINE INSERTING-COMPENSATOR.....	27
<b>Chapter 6 List of ADT8940 basic library functions</b> .....	<b>29</b>
<b>Chapter 7 Details of ADT8940 basic library functions</b> .....	<b>31</b>
☞ CATEGORY OF BASIC PARAMETER SETTING.....	31
☞ CATEGORY OF DRIVE STATUS CHECK .....	33
☞ CATEGORY OF MOVEMENT PARAMETER SETTING .....	34
☞ CATEGORY OF MOVEMENT PARAMETER CHECK .....	37
☞ CATEGORY OF DRIVE.....	38
☞ CATEGORY OF SWITCH AMOUNT INPUT/ OUTPUT.....	40
<b>Chapter 8 Guide to movement control function library</b> .....	<b>42</b>
<b>Chapter 9 Briefing on movement control development</b> .....	<b>45</b>
☞ CARD INITIALIZATION .....	45

☞ SPEED SETTING .....	46
<b>Chapter 10 Programming samples in movement control development .....</b>	<b>47</b>
☞ VB PROGRAMMING SAMPLES .....	47
☞ VC PROGRAMMING SAMPLES .....	60
<b>Chapter 11 Normal failures and solutions .....</b>	<b>76</b>
MOTOR SERVICE FAILURE .....	76
☞ ABNORMAL SWITCH AMOUNT INPUT .....	78
<b>Appendix A Typical wiring for motor driver .....</b>	<b>81</b>
<b>Appendix B Introduction on application function library .....</b>	<b>83</b>

## Chapter 1 General information

### ☛ INTRODUCTION

ADT8940 Card is a kind of high-performance 4-axis servo/ stepping control card based on PCI bus and supporting Plug & Play, while one system can support up to 16 control cards and control up to 64 lines of servo/ stepping motors.

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

It supports 2-4 axis of straight line inserting-compensator, with the maximum inserting-compensator speed of 1MHz.

Speed may be controlled in fixed speed or trapezoidal acceleration/ deceleration.

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

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

### ☛ MAIN FUNCTIONS

- 32-digit PCI bus, enabling Plug & Play
- All the input and output are under photoelectric coupler isolation, with strong resistance to disturbance.
- 4-axis servo/ stepping motor control, with every axis able to move independently without mutual effects
- Frequency tolerance for pulse output is less than 0.1%
- **The maximum pulse output frequency is 2MHz**
- Pulse output may be single (pulse+ direction) or double(pulse+ pulse)
- **All the 4 axes have position feedback input in 32-digit counting, giving the maximum counting range of -2,147,483,648~+2,147,483,647**
- Trapezoidal acceleration/ deceleration
- 2-4 axis straight line inserting compensator
- **Maximum inserting compensator speed: 1MHz**

---

## ADT8940 4-axis servo/ stepping movement control card

---

- Real-time reading of logic, actual and driving speeds during movement
- 40-line digital input (each axis of position feedback may be used as 2 input points, altogether 8)
- Two limit input for each axis may be set as Nil and work as general input
- Up to 16 control cards supported within one system
- DOS/WINDOWS95/98/NT/2000/XP supported

### ☛ APPLICATION SCOPE

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

## Chapter 2 Hardware installation

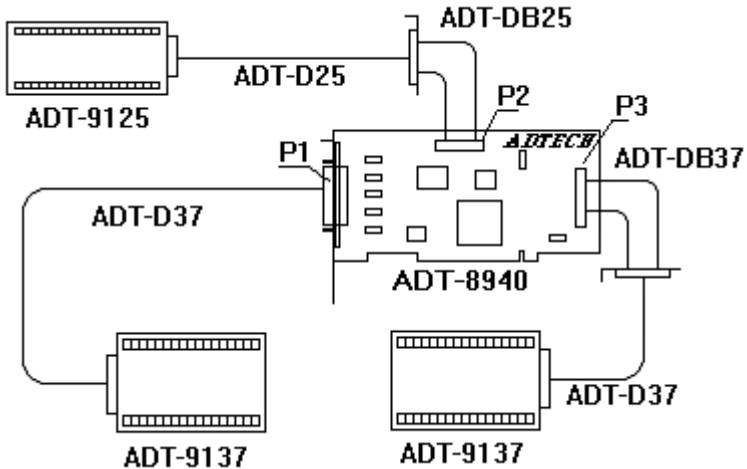
### ☛ PARTS

1. ADT-8940 User Manual (this manual)
2. ADT-8940 4-axis PCI bus high-performance movement control card
3. ADT-8940 user CD
4. ADT-9137 37-pin signal connecting plate, 2 pcs
5. ADT-D37 37-pin blocking cable, 2 pcs
6. ADT-9125 25-pin switch signal connecting plate, 1 pc
7. ADT-D25 25-pin blocking cable, 1 pc
8. ADT-DB25 25-pin flat cable, 1 pc
9. ADT-DB37 37-pin flat cable, 1 pc

### ☛ INSTALLATION

1. Switch off the computer power supply (for ATX supply case, switch off the overall power)
2. Open the back cover of the computer case
3. Select an available PCI slot and insert ADT-8940
4. Ensure the golden finger of ADT-8940 has been fully inserted the slot and then secure the bolt
5. Decide whether to install P2 and P3 interface cables, depending on users' scenarios

### Chapter 3 Electrical connection

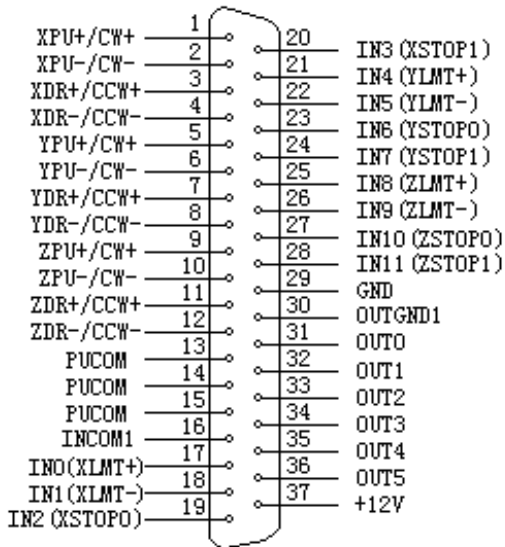


There are three input/ output interfaces inside an Adt8940 card, whereby P1 and P3 are for 37-pin socket and P2 is for 25-pin.

**P1** is the signal cable for pulse output, switch amount input and switch amount output along X, Y and Z axis (OUT0-OUT5); **P3** is the signal cable for encoder input and switch amount input along X, Y, Z and W axis; and **P2** is the signal cable for pulse output, switch amount input and switch amount output along W axis (OUT6-OUT15).

Signals are defined as follows:

**P1 LINE**



Line number	Signal	Introduction
1	XPU+/CW+	X pulse signal +
2	XPU-/CW-	X pulse signal -
3	XDR+/CCW+	X direction signal +
4	XDR-/CCW-	X direction signal -
5	YPU+/CW+	Y pulse signal +
6	YPU-/CW-	Y pulse signal -
7	YDR+/CCW+	Y direction signal +
8	YDR-/CCW-	Y direction signal -
9	ZPU+/CW+	Z pulse signal +
10	ZPU-/CW-	Z pulse signal -
11	ZDR+/CCW+	Z direction signal +
12	ZDR-/CCW-	Z direction signal -
13	PUCOM	Used for single-port input, not available for external power supply
14	PUCOM	Used for single-port input, not available for external power supply
15	PUCOM	Used for single-port input, not available for external power supply

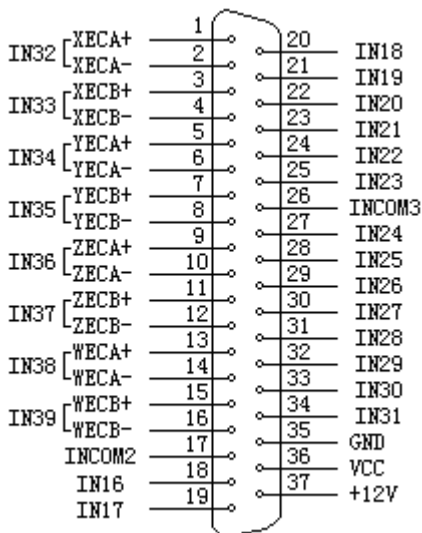
---

**ADT8940 4-axis servo/ stepping movement control card**

---

16	INCOM1	Public port for photoelectric coupling input (for following signals)
17	IN0(XLMT+)	Limit signal for X+, able to work as general input signal 0
18	IN1(XLMT-)	Limit signal for X-, able to work as general input signal 1
19	IN2(XSTOP0)	General input signal 2
20	IN3(XSTOP1)	General input signal 3
21	IN4(YLMT+)	Limit signal for Y+, able to work as general input signal 4
22	IN5(YLMT-)	Limit signal for Y-, able to work as general input signal 5
23	IN6(YSTOP0)	General input signal 6
24	IN7(YSTOP1)	General input signal 7
25	IN8(ZLMT+)	Limit signal for Z+, able to work as general input signal 8
26	IN9(ZLMT-)	Limit signal for Z-, able to work as general input signal 9
27	IN10(ZSTOP0)	General input signal 10
28	IN11(ZSTOP1)	General input signal 11
29	GND	Internal power supply earthing
30	OUTGND1	General negative port for OUT0-5 switch amount output points
31	OUT0	Output points for switch amount
32	OUT1	
33	OUT2	
34	OUT3	
35	OUT4	
36	OUT5	
37	+12V	Positive port of internal +12V power supply, not available for external power supply

**P3 line**



Line number	Signal	Introduction
1	XECA+	X-axis encoder A-phase input+
2	XECA-	X-axis encoder A-phase input -, able to work as general input signal 32
3	XECB+	X-axis encoder B-phase input +
4	XECB-	X-axis encoder B-phase input -, able to work as general input signal 33
5	YECA+	Y-axis encoder A-phase input+
6	YECA-	Y-axis encoder A-phase input -, able to work as general input signal 34
7	YECB+	Y-axis encoder B-phase input +
8	YECB-	Y-axis encoder B-phase input -, able to work as general input signal 35
9	ZECA+	Z-axis encoder A-phase input+
10	ZECA-	Z-axis encoder A-phase input -, able to work as general input signal 36
11	ZECB+	Z-axis encoder B-phase input +
12	ZECB-	Z-axis encoder B-phase input -, able to work as general input signal 37

## ADT8940 4-axis servo/ stepping movement control card

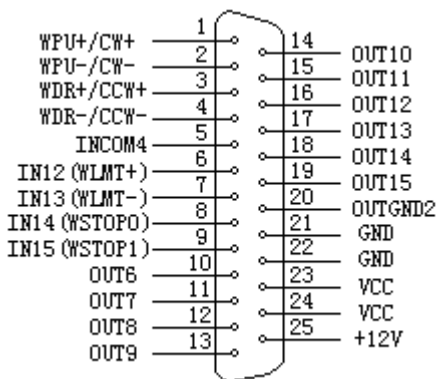
13	WECA+	W-axis encoder A-phase input+
14	WECA-	W-axis encoder A-phase input -, able to work as general input signal 38
15	WECB+	W-axis encoder B-phase input +
16	WECB-	W-axis encoder B-phase input -, able to work as general input signal 39
17	INCOM2	Public port for photoelectric coupling input (for following signals)
18	IN16	General input signal 16
19	IN17	General input signal 17
20	IN18	General input signal 18
21	IN19	General input signal 19
22	IN20	General input signal 20
23	IN21	General input signal 21
24	IN22	General input signal 22
25	IN23	General input signal 23
26	INCOM3	Public port for photoelectric coupling input (for following signals)
27	IN24	General input signal 24
28	IN25	General input signal 25
29	IN26	General input signal 26
30	IN27	General input signal 27
31	IN28	General input signal 28
32	IN29	General input signal 29
33	IN30	General input signal 30
34	IN31	Hardware termination signal, able to work as general input signal 31
35	GND	Internal power supply earthing
36	VCC	Positive port of internal +5V power supply, not available for external power supply
37	+12V	Positive port of internal +12V power supply, not available for external power supply

**Remark:** In case an encoder is used for general input signals, XECA+, XECB+, YECA+, YECB+, ZECA+, ZECB+, WECA+, and WECB+ will be respectively used as public ports of corresponding input signals.

**Voltage at the public ports can only be +5V;** in case of using an external+12V power supply, users must serially connect a 1K resistance. Please refer to the following digital input connection part for wiring method.

## ADT8940 4-axis servo/ stepping movement control card

### ☞ P2 line



Line number	Signal	Introduction
1	WPU+/CW+	W pulse signal +
2	WPU-/CW-	W pulse signal -
3	WDR+/CCW+	W direction signal +
4	WDR-/CCW-	W direction signal -
5	INCOM4	Public port for photoelectric coupling input (for following signals)
6	IN12(WLMT+)	Limit signal for W+, able to work as general input signal 12
7	IN13(WLMT-)	Limit signal for W-, able to work as general input signal 13
8	IN14(WSTOP0)	General input signal 14
9	IN15(WSTOP1)	General input signal 15
10	OUT6	Output points for switch amount
11	OUT7	
12	OUT8	
13	OUT9	
14	OUT10	
15	OUT11	
16	OUT12	
17	OUT13	
18	OUT14	

## ADT8940 4-axis servo/ stepping movement control card

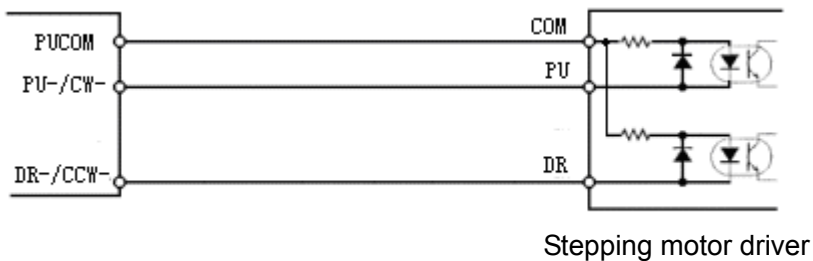
19	OUT15	
20	OUTGND2	General negative earthing for OUT6-15 switch amount output points
21	GND	Internal power supply earthing
22	GND	Internal power supply earthing
23	VCC	Positive port of internal +5V power supply
24	VCC	Positive port of internal +5V power supply
25	+12V	Positive port of internal +12V power supply

### ☛ CONNECTION FOR PULSE/ DIRECTION INPUT SIGNAL

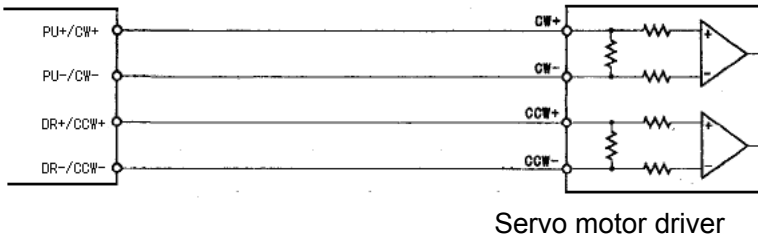
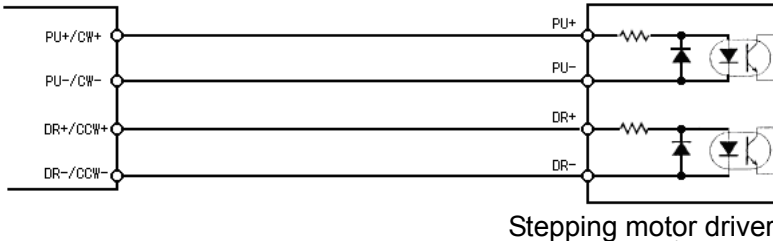
Pulse output is in differential output.

May be conveniently connected with a stepping/ servo driver

The following picture shows connection between pulse and direction anode.

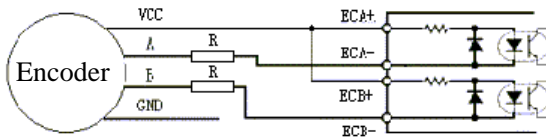


The following picture shows independent connection between pulse and direction signals; this method is recommended as it is differential connection with strong resistance to disturbance.

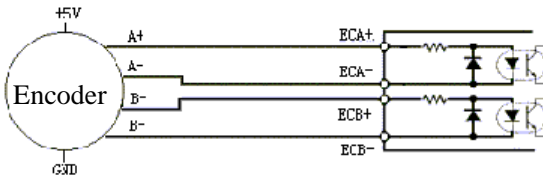


Remark: Refer to Appendix A for wiring maps of stepping motor drivers, normal servo motor driver and terminal panel.

**☞ CONNECTION FOR ENCODER INPUT SIGNAL**

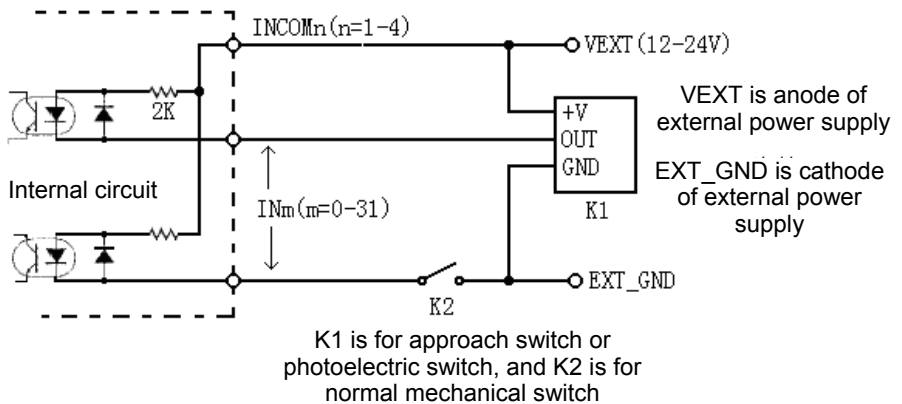


Wiring map for an open-collector output-type encoder. For +5V power supply, R is not required; for +12V power supply, R= 1K $\Omega$ , and for +24V power supply, R= 2K $\Omega$



Wiring map for a differential-driver output-type encoder

☛ CONNECTION FOR DIGITAL INPUT



Remark:

(1) Public terminal for IN0-IN11: INCOM1

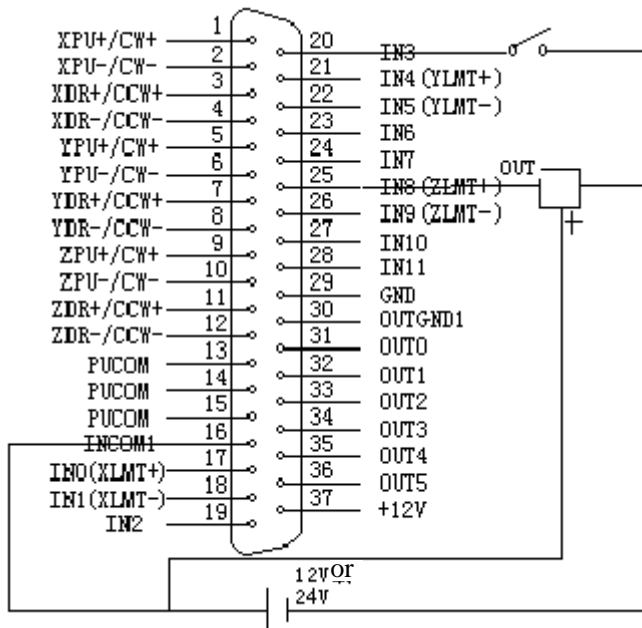
Public terminal for IN12-IN15: INCOM4

Public terminal for IN16-IN23: INCOM2

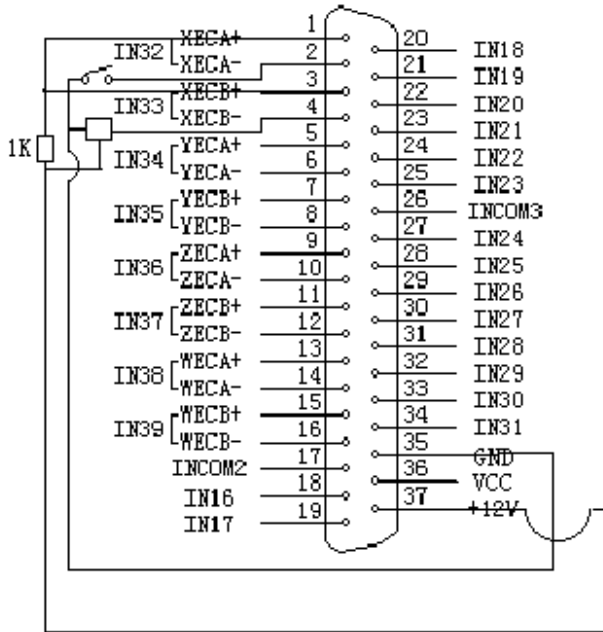
Public terminal for IN24-IN31: INCOM3

(2) To make input signals effective, users shall make sure: firstly, the photoelectric coupling public ports for corresponding input signals (INCOM1, INCOM2, INCOM3 or INCOM4) have been connected with anodes of 12V/ 24V power supply; secondly, one port of the normal switch or earthing cable of the approach switch has been connected with the cathode (earthing cable); and lastly, the other port of the normal switch or the control of the approach switch has been connected with the input port corresponding by the terminal panel.

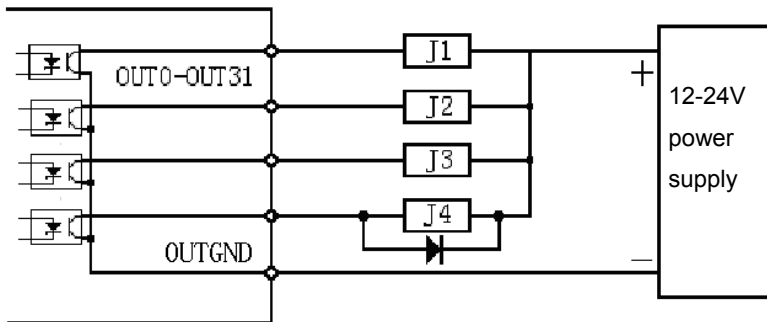
(3) The following is the actual wiring map of power supply from normal switch and approach switch to photoelectric coupling public ports, through external power supply.



- (4) The following picture is the power supply from internal 5V supply to photoelectric coupling anode of the encoder; in case of voltage higher than 5V, a corresponding resistance must be serially connected, for example, a 1K resistance shall be serial connected for 12V voltage, as shown in the picture. Wiring for using external power supply is same as that for normal input points, please just refer to the above picture.



☛ CONNECTION FOR DIGITAL OUTPUT



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

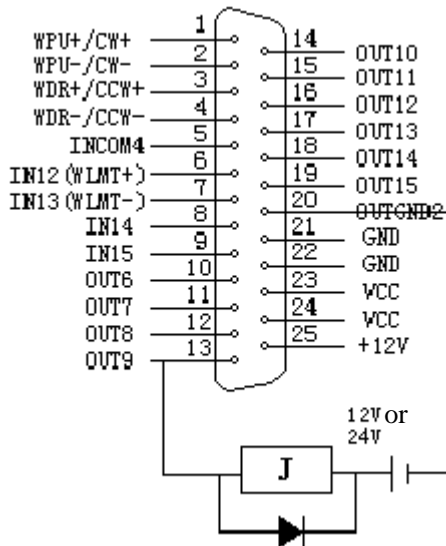
Remark:

(1) Public terminal for OUT0-OUT5: OUTGND1

Public terminal for OUT6-OUT15: OUTGND2

(2) To make output signals effective, users must make sure of connection between the output public port OUTGND1 or OUTGND2 and cathode of external power supply (earthing cable) if using external power supply, or connection between internal power supply earthing (GND) and the ground if using internal power supply. Relay coils must have one side connected with the power supply anode and the other side connected with the corresponding output port of the terminal panel.

(3) The following picture is the actual wiring map for power supply by external power supply.



## Chapter 4 Software installation

ADT8940 card must be used with drive installed under Win95/ Win98/ NT/ Win2000/ WinXP, but in case of DOS, no drive is required to be installed.

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

Drive for the control card is located in the DevelopmentPackage/ Drive/ ControlCardDrive folder within the CD, and the drive file is named as ADT8940.INF.

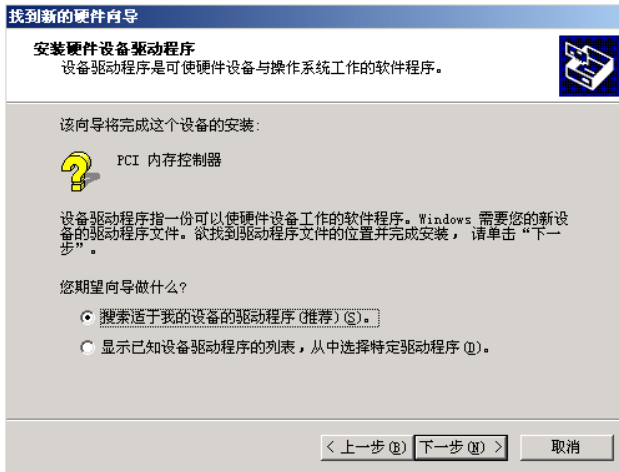
### ☛ DRIVE INSTALLATION UNDER WIN2000

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

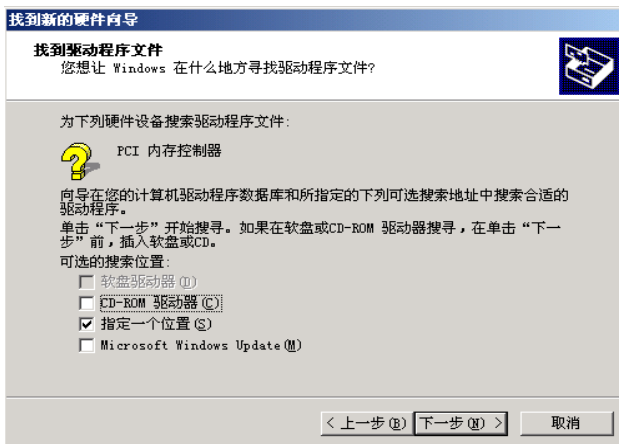
After inserting the ADT8940 card into the PCI slot of a computer, a user shall log in as administrator to the computer; upon display of the initial interface, the computer shall notify “Found new hardware” as follows:



Just click “Next” to display the following picture:

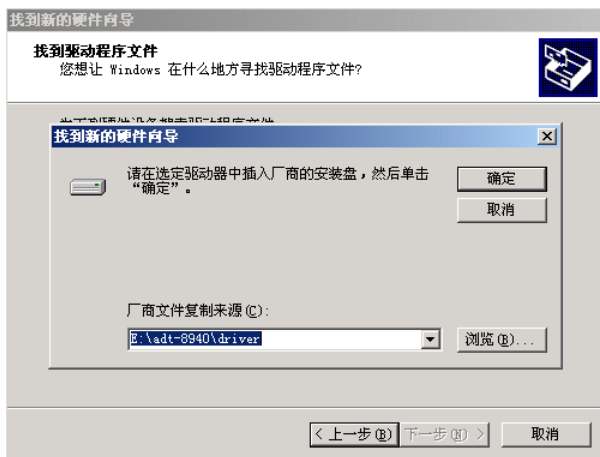


Click again “Next” to display the following picture:



Then select “Specify a location” and Click again “Next” to display the following picture:

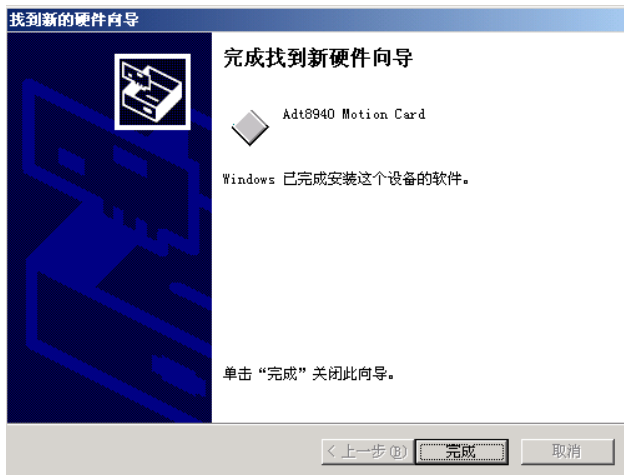
Click “Browse” button to select DevelopmentPackage/ Drive/ ControlCardDrive and find the ADT8940.INF file, then click “OK” to display the following interface:



Click “Next” to display the following picture:

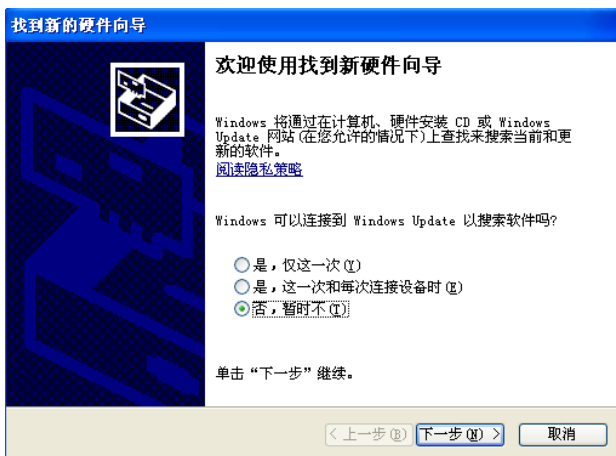


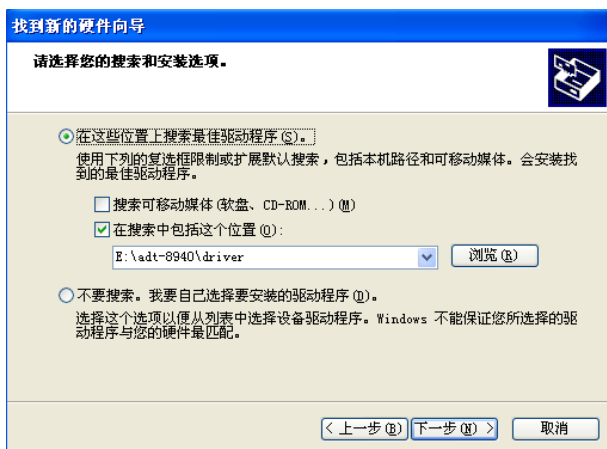
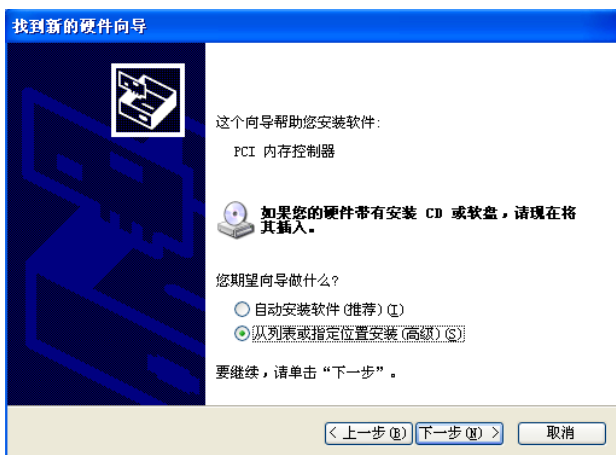
Finally click “Done” to complete installation.



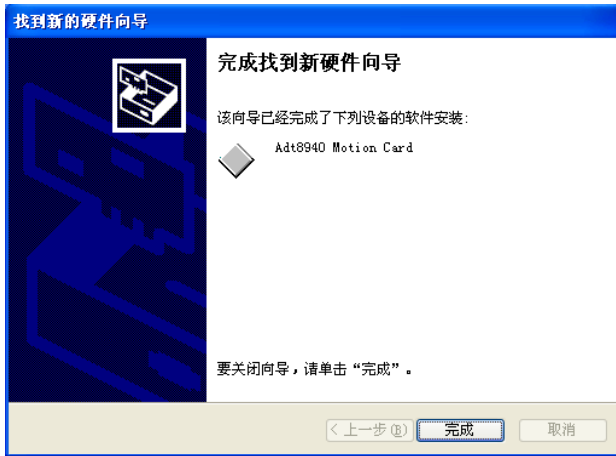
#### DRIVE INSTALLATION UNDER WINXP

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





Click “Browse” button to select DevelopmentPackage/ Drive/ ControlCardDrive and find the ADT8940.INF file, then click “Next” to display the following interface:



Then click "Done" to complete installation.

## Chapter 5 Functions

### ☛ Pulse output method

Pulse output may be realized through either independent 2-pulse or 1-pulse. In case of independent 2-pulse, the positive direction drive has PU/CW outputting drive pulses, and the negative direction drive has DR/CCW outputting drive pulses. In case of 1-pulse, PU/CW outputs drive pulse and DR/CCW outputs direction signals.

Pulse/ direction are both of positive logic setting

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

### ☛ Hardware restriction signal

Hardware restriction signals LMT+ and LMT- are respectively to limit the input signals outputted by drive pulse along positive and negative directions, able to be set as “Apply”, “Don’t Apply” with high/ low levels.—Actually “Apply” or “Don’t Apply” can be set for positive limit and negative limit individually; in case “Don’t Apply” is selected, they may work as ordinary input points.

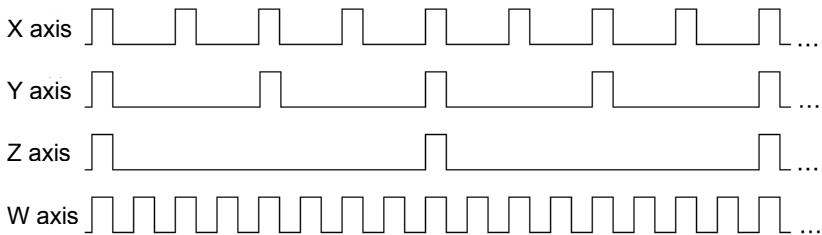
Hardware restriction signals STOP0 and STOP1 are input signals that may realize hardware termination for all axis drive and may be set as “Apply”, “Don’t Apply” as well as the termination method for high/ low levels. In case “Don’t Apply” is selected, they may work as ordinary input points. Besides, they, when working as drive for inserting compensator, are effective for the minimum inserting compensator axis only.

### ☛ Straight line inserting-compensator

This card may work for 2-4 axis straight line inserting compensator and support any 2 axis or 3 axis straight line inserting compensator, under the modified method of point-by-point comparison, which can ensure uniform pulse along the long axis, giving the precision within one pulse.

Firstly, take the axis outputting the maximum pulses among the axes joining inserting compensation as the long axis, and proportionally distribute for the rest axes. Speed control applies only to speed of the long axis, for example: (1-X axis, 2-Y axis, 3-Z axis, and 4-W axis)

Take four axes for straight line inserting compensation, while Axis 1 outputs 1000 pulses, Axis 2 outputs 500, Axis 3 outputs 250 and Axis outputs 2000.



From the above figure, W axis (Axis 4) shall be the long axis, and the rest axes proportionally share the pulses.

Setting of inserting compensation speed takes the minimum speed of an axis among the joined axes as the benchmark, for example, if Axis 2 and Axis 3 join straight line inserting compensation, the inserting compensation speed will be determined by speed of Axis 2. Moreover, speed of inserting compensation is only half of the single axis. Example in more details:

Axis 2 and Axis 3 work for two-axis straight line inserting compensation, with Axis 2 outputs 10000 pulses along positive direction and Axis 3 outputs 5000 pulses along negative direction, which means Axis 2 is the long axis.

```
set_range(0,2,500); // Card with multiple is 1 and hardware version of 1 must be set
set_startv(0,2,1000);
set_speed(0,2,1000);
inp_move2(0,2,3,10000,-5000);
```

After execution of the above program, Axis 2 will send 10000 pulses in the frequency of  $1000/2=500\text{Hz}$ , while frequency of Axis 3 shall be  $500*5000/10000=250\text{Hz}$ .

If speed of Axis 2 realizes trapezoidal acceleration/ deceleration, inserting compensation will also follow such trapezoidal acceleration/ deceleration.

## Chapter 6 List of ADT8940 basic library functions

### List of V110 library functions

Function type	Function name	Function description	Page
Basic parameters	adt8940_initial	Initialize card	30
	get_lib_version	Get version	30
	set_pulse_mode	Set pulse mode	30
	set_limit_mode	Set limit mode	31
	set_stop0_mode	Set stop mode	31
	set_stop1_mode	Set stop mode	32
Check for drive status	get_status	Get status of single-axis drive	32
	get_inp_status	Get status of inserting compensation drive	32
Movement parameter setting	set_range	Set range	33
	set_acc	Set acceleration	34
	set_startv	Set starting speed	34
	set_speed	Set drive speed	35
	set_command_pos	Set logic counter	35
	set_actual_pos	Set actual counter	35
Check for	get_command_pos	Get logic position	36

## ADT8940 4-axis servo/ stepping movement control card

movement parameters	get_actual_pos	Get actual position	36
	get_speed	Get drive speed	36
Drive category	pmove	Single-axis quantitative drive	37
	dec_stop	Deceleration stop	37
	sudden_stop	Sudden stop	37
	I N P _ M O V E 2	2-axis inserting compensation	38
	inp_move3	3-axis inserting compensation	38
	inp_move4	4-axis inserting compensation	38
Switch amount category	read_bit	Read single input point	39
	write_bit	Output single input point	39

### Supplementary functions in V220

Status category	get_hardware_ver	Hardware version	39
	get_delay_status	Delay status	40
Setting category	set_delay_time	Set delay	40
	set_suddenstop_mode	Set hardware stop	40

### Revised functions in V220

set_range	This function must be used for V110 cards; for V220 cards, this function, although remained, has been senseless but to ensure new cards are compatible with old ones.
get_lib_version	Added card number parameter to ensure concurrent use of multiple cards with convenient operations

## Chapter 7 Details of ADT8940 basic library functions

### ☛ CATEGORY OF BASIC PARAMETER SETTING

#### 1.1 Initialize card

```
int adt8940_initial(void);
```

- (1) Returning value >0 means amount of installed adt8940 cards; in case the returning value is 3, the available card numbers shall be 0, 1, and 2;
- (2) Returning value =0 means no installation of adt8940 card;
- (3) Returning value <0 means no installation of port drive program if the value is -1 or PCI bus failure is the value is -2.

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

#### 1.2 Get current library version

```
int get_lib_version(); (VER:110)
```

```
int get_lib_version(int cardno); (VER:220)
```

Here returning values are combination of hardware and library version number.

Returning value 110 means that the hardware version is 1 and the library version is 1.0, returning value 220 means that the hardware version is 2 and the library version is 2.0, and so on.

#### 1.3 Set outputted pulse mode

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

cardno     Card number

axis        Axis number (1-4)

value      0: Pulse method     1: Pulse + direction method

Pulse/ direction are both of positive logic setting

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

logic     0: Positive logic pulse                      1: Negative logic pulse

Positive logic pulse:                      Negative logic pulse:

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

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

Returning value     0: Correct                                      1: Wrong

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

#### 1.4 Set mode of nLMT signal input along positive/ negative direction

**int set\_limit\_mode(int cardno, int axis, int v1,int v2,int logic);**

cardno     Card number

axis        Axis number (1-4)

v1         0: Apply negative limit                      1: Don't apply negative limit

v2         0: Apply low level                              1: Apply high level

logic      0: Apply positive limit                      1: Don't apply positive limit

Returning value     0: Correct                                      1: Wrong

**Default mode: Apply positive and negative limits with low level**

### 1.5 Set mode of stop0 input signal

**int set\_stop0\_mode(int cardno, int axis, int v,int logic);**

cardno Card number

axis Axis number (1-4)

v 0: Don't apply 1: Apply

logic 0: Apply low level 1: Apply high level

Returning value 0: Correct 1: Wrong

**Default mode: Don't apply**

### 1.6 Set mode of stop1 input signal

**int set\_stop1\_mode(int cardno, int axis, int v,int logic);**

Cardno Card number

Axis Axis number (1-4)

v 0: Don't apply 1: Apply

logic 0: Apply to low level 1: Apply to high level

Returning value 0: Correct 1: Wrong

**Default mode: Don't apply**

## ☛ CATEGORY OF DRIVE STATUS CHECK

### 2.1 Get status of single-axis drive

**int get\_status(int cardno,int axis,int \*value)**

cardno Card number

axis Axis number (1-4)

value Indicator of drive status

0: Drive completed

Non-0: Drive in process

Returning value 0: Correct 1: Wrong

## 2.2 Get status of inserting compensation drive

**int get\_inp\_status(int cardno,int \*value)**

cardno Card number

value Indicator of inserting compensation drive:

0: Inserting compensation completed      1: Inserting compensation in process

Returning value      0: Correct                      1: Wrong

## ☞ CATEGORY OF MOVEMENT PARAMETER SETTING

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

## 3.1 Set range

**int set\_range(int cardno,int axis,long value);**

cardno Card number

axis Axis number

value Range (500-2)

Returning value      0: Correct                      1: Wrong

**Remark: set\_range function must be used within cards with hardware version of 1, and has been stopped within cards with hardware version of 2. Users don't have to set this function in cards, however, setting, if any, will not make any effects. Users must avoid to build relationship with range when setting speed and acceleration or getting speed. To keep compatibility between the old and new versions, this function is still listed under this function list, but setting for relevant speed and acceleration in the application library functions have been synchronically updated, and users need only take the new dynamic library to replace the old one.**

🔴\* **Attention: Range is the multiple parameter determining speed, acceleration/ deceleration and rate of change of acceleration/ deceleration; assuming a range value of R, multiple rate shall be calculated as 500 / R.**

As the setting range for parameters such as drive speed, starting speed and acceleration/ deceleration is 1~8000, in case of requiring to set values over 8000, users must increase the multiple rate; such increase will make higher drive speed,

however, the speed resolution turns tough. Please set the minimum value in the used speed range, for example, if requiring a speed of 40KPPS, users better set the multiple rate within the speed range 1~8000 to be 5, i.e., set R to be 100.

For R within 500-2, the corresponding multiple rate shall be 1-250

Users are expected not to change the range of R during drive, otherwise the speed will be confused.

For example:

```
set_range(0,1,50);
```

This statement means to set the multiple rate of the first card along X axis to be  $500/50 = 10$

```
set_range(0,3,2);
```

This statement means to set the multiple rate of the first card along Z axis to be  $500/2 = 250$

### 3.2 Set acceleration

cardno     Card number

axis        Axis number

Range (1-8000 for hardware version 1 and also hardware version 2)

Returning value     0: Correct                     1: Wrong

Acceleration is the speed change parameter for straight line acceleration/

deceleration drive; its setting value is A, subject with the following calculation:

Acceleration (PPS/SEC) =  $A * 125 * \text{Multiple Rate}$  (For hardware version 1)

i.e., Acceleration (PPS/SEC) =  $A * 125 * (500/R)$

Acceleration (PPS/SEC) =  $A * 125$  (For hardware version 2)

Range for the acceleration setting value A is 1~8,000.

For example:

```
set_range(0,1,5);
```

```
set_acc(0,1,100);
```

Then the acceleration will be:

$100 * 125 * (500/5) = 1250000 \text{ PPS/SEC}$

### 3.3 Set starting speed

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

cardno     Card number





### 4.3 Get drive speed

**int get\_speed(int cardno,int axis,long \*speed)**

cardno Card number

axis Axis number

speed Indicator of current drive speed

Returning value 0: Correct 1: Wrong

Its data unit is same as that for drive setting value V.

This function can get the axis drive speed at any time.

## ☛ CATEGORY OF DRIVE

### 5.1 Single-axis quantitative drive

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

cardno Card number

axis Axis number

pulse Outputted pulses

>0: move along positive direction

<0: move along negative direction

Range (-268435455~+268435455)

Returning value 0: Correct 1: Wrong

**Remark: Users must correctly set the parameters required by speed curve before making drive commands.**

### 5.2 Deceleration stop

**int dec\_stop(int cardno,int axis)**

cardno Card number

axis Axis number

Returning value 0: Correct 1: Wrong

During drive pulse output, this command will make deceleration stop. Users may also use this command to stop when the drive speed is lower than the starting speed.



Range (-8388608~+8388607)

Returning value 0: Correct

1: Wrong

## 5.6 4-axis inserting compensation

**int inp\_move4(int cardno,long pulse1,long pulse2,long pulse3,long pulse4)**

cardno Card number

pulse1,pulse2,pulse3,pulse4

Relative distance of movement along X-Y-Z-W axis

Range (-8388608~+8388607)

Returning value 0: Correct

1: Wrong

## ☛ CATEGORY OF SWITCH AMOUNT INPUT/ OUTPUT

### 6.1 Read single input point

**int read\_bit(int cardno,int number)**

cardno Card number

number Input point (0-39)

Returning value

0: low level

1: high level

-1: error

### 6.2 Output single input point

**int write\_bit(int cardno,int number,int value)**

cardno Card number

number Output point (0-15)

value 0: low 1: high

Returning value

0: correct

1: wrong

A number corresponding to the output number

## ☞ Supplementary library functions

### 1. Hardware version

**int** `get_hardware_ver(int cardno)`

cardno    Card number

Return value    1: Version 1 hardware

                  2: Version 2 hardware

### 2. Hardware stop

**int** `set_suddenstop_mode(int cardno,int v,int logical)`

cardno    Card number

v            0: Apply; 1: Don't apply

logical    0: low level; 1: high level

Returning value    0: Correct                    1: Wrong

**Remark:** Hardware stop signals are assigned to use the 34 pin at the P3 terminal panel (IN31)

### 3. Delay time

**int** `set_delay_time(int cardno,long time)`

cardno    Card number

time      Delay time

Returning value    0: Correct                    1: Wrong

**Remark:** The time unit is 1/8us, with the maximum *integer* value as its maximum value

### 4. Delay status

**int** `get_delay_status(int cardno)`

cardno    Card number

Returning value    0: delay stop

                  1: delay in process

## Chapter 8 Guide to movement control function library

### 1. Introduction on ADT8940 function library

ADT8940 function library is actually the interface for users to operate the movement control card; users can control the movement control card to execute corresponding functions simply by calling interface functions.

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

### 2. Calling dynamic link library under Windows

The dynamic link library Adt8940.dll under Windows is programmed in VC, applicable for general programming tools under Windows, including VB, VC, C++Builder, VB.NET, VC.NET, Delphi and group software LabVIEW.

#### 2.1 Calling under VC

- (1) Create a new item;
- (2) Copy the adt8940.lib and adt8940.h files from DevelopmentPackage/VC in the CD to the routing of the newly created item;
- (3) Under File View of the Work Area of the new item, right click mouse to select "Add Files to Project" and then in the pop-up file dialogue select the file type to be "Library Files(.lib)", then search out "adt8940.lib" and select it, finally click "OK" to finish loading of the static library;
- (4) Add #include "adt8940.h" in the declaim part of the source file, header or overall header "StdAfx.h".

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

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

#### 2.2 Calling under VB

- (1) Create a new item;
- (2) Copy the adt8940.h file from DevelopmentPackage/VB in the CD to the routing of the newly created item;

- (3) Select the menu command Engineering/Add module and subsequently Save Current in the dialogue to search out the adt8940.bas module file, finally click the Open button.

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

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

## **2.3 Calling under C++Builder**

- (1) Create a new item;
- (2) Copy the adt8940.lib and adt8940.h files from DevelopmentPackage/C++Builder in the CD to the routing of the newly created item;
- (3) Select the menu command "Project\Add to Project", and in the pop-up dialogue select the file type to be "Library files(\*.lib)", then search out the "adt8940.lib" file and click Open button;
- (4) Add #include "adt8940.h" in the declaim part of the program file.

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

## **2.4 Calling under LabView 8**

- (1) Create a new VI;
- (2) Copy the adt8940.lib and adt8940.dll files from DevelopmentPackage/LabVIEW in the CD to the routing of the newly created item;
- (3) Right click mouse in the blank area of the program interface to display the Function Palette, select "Select a VI.." and subsequently in the pop-up window select the adt8940.llb file, finally select the required library function in the "Select the VI to Open" window and drag into the program interface.

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

## **3. Calling library functions under DOS**

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

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

- (1) Under the development environment of Borland C, select the "Project\Open Project" command to create a new item;

- (2) Copy the ADT8940H.LIB or ADT8940L.LIB file and ADT8940.H file from DevelopmentPackage/ C (or C++) in the CD to the routing of the newly created item;
  - (3) Select the “Project\Add Item” command and further in the dialogue select “ADT8940H.LIB” or “ADT8940L.LIB”, finally click the Add button;
  - (4) Add #include “adt8940.h statement in the user program file.
- After the above four steps, users can call functions in the dynamic link library.

#### **4. Returning values of library functions and their meanings**

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

Except “int adt8940\_initial(void)” and “int read\_bit(int cardno, int number)” with special returning values, other functions have only “0” and “1” as the returning values, where “0” means successful calling and “1” means failed calling.

The following list introduces meanings of function returning values.

Function name	Returning value	Meaning
Adt8940_initial	-1	No installation of port drive
	-2	PCI slot failure
	0	No installation of control card
	>0	Amount of control card
Read_bit	0	Low level
	1	High level
	-1	Card number or input point out of limit
Other functions	0	Correct
	1	Wrong

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

## Chapter 9 Briefing on movement control development

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

### ☛ CARD INITIALIZATION

At the beginning users shall call the `adt8940_initial()` function and ensure the `adt8940` card has been correctly installed, then set pulse output mode and limit switch mode. The above parameters shall be set for individual machine, and normally only one setting is required during program initialization, instead of any later setting.

Besides, range setting function `set_range` shall normally be set following the maximum pulse frequency to be used and no more changed in the future. Each axis may vary on such range setting, for example, X axis has the maximum frequency of 100K, as the maximum value for `set_speed` is 8000, the multiple rate shall be  $100000/8000=12.5$ , and subsequently range R shall be  $500/12.5=40$ , i.e.,

```
set_range (cardno,1,40);
```

For 100K frequency to be outputted:

```
set_speed (cardno,1,8000)
```

For 10K frequency to be outputted:

```
set_speed(cardno,1,800)
```

The minimum output frequency shall be  $1*12.5=12.5\text{Hz}$

Normally users shall based on the maximum frequency possibly to be used to determine the R value and no more change the value during use unless the minimum frequency can no more meet requirements.

**Remark: Library function `adt8940_initial` is the door to ADT8940 card, thus calling other functions are of sense only after successful card initialization with calling to this function.**

## ☞ SPEED SETTING

### 2.1 Uniform speed movement

Parameter setting is so simple that users just set the drive speed equal to the starting speed; other parameters need no setting.

Relevant functions:

set\_startv

set\_speed

☛\* **Remark: values used by functions will give the actual speed only after multiplying by the multiple rate.**

### 2.2 Inserting compensation speed

Adt8940 card can take any 2 axes, any 3 axes or all the 4 axes for straight line inserting compensation.

For speed of inserting compensation, speed parameter for the earliest axis will apply as speed of the long axis, for example,

inp\_move2 (0,3,1,100,200) is to apply the speed parameters of the first axis, i.e., X axis, independent of parameter sequence.

inp\_move3 (0,3,4,2,100,200,500) is to apply the speed parameters of the second axis, i.e., Y axis, independent of parameter sequence.

**Remark: speed multiple rate during inserting compensation is half of that during single-axis movement, which means under the same parameters speed of inserting compensation is only half of that of single-axis movement.**

## Chapter 10 Programming samples in movement control development

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

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

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

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

Now let's brief application of ADT8940 card function library in VB and VC; users using other programming languages may take reference.

### ☛ VB PROGRAMMING SAMPLES

#### 1.1 Preparation

- (1) Create a new item and save as "test.vbp";
- (2) Add the adt8940.bas module in the item following the above-introduced method;

#### 1.2 Movement control module

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

- (2) At first, within the movement control module self-define initialization functions of the movement control card and initialize library functions to be sealed into initialization functions;
- (3) Further self-define relevant movement control functions such as speed setting function, single-axis movement function, and inserting compensation movement function;
- (4) Source code of ctrcard.bas is:

```
***** Movement control module *****
To simply, conveniently and quickly develop popular, expandable and
easy-for-maintenance application systems, we have classified sealed all the
library functions based on the function library of the control card. The following
example is subject with one movement control card.
*****

Public Result As Integer
Public Const MULTIPLE = 10 'Multiple rate
Const MAXAXIS = 4 'Maximum number of axis
Public g_CardVer=0 'Hardware version

*****Initialization function*****
This function contain those library functions frequently used in control card
initialization, which is the foundation to call other functions and must be firstly
called in this example program.
'Returning value <=0 means initialization failure and >0 means initialization
success
*****

Public Function Init_Card() As Integer
    Result = adt8940_initial 'Card initialization functions
    If Result <= 0 Then
        Init_Card = Result
        Exit Function
    End If
    g_CardVer=get_hardware_ver(0)
    For i = 1 To MAXAXIS
```

```
set_range 0, i, 500 / MULTIPLE "Set range
set_limit_mode 0, i, 0, 0, 0 'Set limit mode
set_command_pos 0, i, 0 'Set logic counter
set_actual_pos 0, i, 0 'Set actual counter
If g_CardVer=1 Then
    set_startv 0, i, 1000/ MULTIPLE 'Set starting speed
    set_speed 0, i, 1000/ MULTIPLE 'Set drive speed
ElseIf g_CardVer=2 Then
    set_startv 0, i, 1000 ' Set starting speed
    set_speed 0, i, 1000 ' Set drive speed
End If
Next i
Init_Card = Result
End Function

*****Speed setting module*****
'Based on parameter value, judge it's in uniform speed or acceleration/
deceleration
'Returning value =0 means success, and Returning value =1 means error
*****

Public Function Setup_Speed(ByVal axis As Integer, ByVal StartV As Long, ByVal Speed
As Long, ByVal Add As Long) As Integer
If (StartV - Speed >= 0) Then
    If g_CardVer=1 Then
        Result = set_startv(0, axis, StartV/ MULTIPLE)
        set_speed 0, axis, StartV/ MULTIPLE
    ElseIf g_CardVer=2 Then
        Result = set_startv(0, axis, StartV)
        set_speed 0, axis, StartV
    End If
Else
    If g_CardVer=1 Then
```

```
Result = set_startv(0, axis, StartV/ MULTIPLE)
set_speed 0, axis, Speed/ MULTIPLE
set_acc 0, axis, Add /125/MULTIPLE
ElseIf g_CardVer=2 Then
    Result = set_startv(0, axis, StartV)
    set_speed 0, axis, Speed
    set_acc 0, axis, Add /125
End If
End If
Setup_Speed = Result
End Function

*****Single-axis drive function*****
'This function is used to drive movement of a single axis
' Returning value =0 means success, and Returning value =1 means error
*****

Public Function Axis_Pmove(ByVal axis As Integer, ByVal value As Long) As Integer
    Result = pmove(0, axis, value)
    Axis_Pmove = Result
End Function

*****Any 2-axis inserting compensation function*****
' This function is used to drive any 2 axis to carry on inserting compensation
movement
' Returning value =0 means success, and Returning value =1 means error
*****

Public Function Interp_Move2(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal
value1 As Long, ByVal value2 As Long) As Integer
    Result = inp_move2(0, axis1, axis2, value1, value2)
    Interp_Move2 = Result
End Function
```

```
*****Any 3-axis inserting compensation function *****
' This function is used to drive any 3 axis to carry on inserting compensation
  movement
' Returning value =0 means success, and Returning value =1 means error
*****
Public Function Interp_Move3(ByVal axis1 As Integer, ByVal axis2 As Integer, ByVal
  axis3 As Integer, ByVal value1 As Long, ByVal value2 As Long, ByVal value3 As
  Long) As Integer
  Result = inp_move3(0, axis1, axis2, axis3, value1, value2, value3)
  Interp_Move3 = Result
End Function

*****4-axis inserting compensation function *****
' This function is used to drive the 4 axis to carry on inserting compensation
  movement
' Returning value =0 means success, and Returning value =1 means error
*****
Public Function Interp_Move4(ByVal value1 As Long, ByVal value2 As Long, ByVal
  value3 As Long, ByVal value4 As Long) As Integer

  Result = inp_move4(0, value1, value2, value3, value4)
  Interp_Move4 = Result
End Function

*****Get movement information*****
'This function is used to feedback the current logic position, actual position and
drive speed of the selected axis
' Returning value =0 means success, and Returning value =1 means error
*****
Public Function Get_CurrentInf(ByVal axis As Integer, value() As Long) As Integer
```

```
Result = get_command_pos(0, axis, value(0))
get_actual_pos 0, axis, value(1)
get_speed 0, axis, value(2)
If g_CardVer=1 Then
    value(2)=value(2)* MULTIPLE
End If
Get_CurrentInf = Result
End Function
```

\*\*\*\*\*Stop axis drive function\*\*\*\*\*

'This function provides either sudden stop mode or deceleration stop mode

'Returning value =0 means success, and Returning value =1 means error

\*\*\*\*\*

```
Public Function StopRun(ByVal axis As Integer, ByVal mode As Integer)
```

```
    If mode = 0 Then
```

```
        Result = sudden_stop(0, axis)           '立即停止
```

```
    Else
```

```
        Result = dec_stop(0, axis)             '減速停止
```

```
    End If
```

```
    StopRun = Result
```

```
End Function
```

## 1.3 Function realization module

### 1.31 Interface design



Introduction:

- (1) Speed setting part—used to set starting speed, drive speed and acceleration of every axis; position setting—used to set drive pulse for every axis; drive information—used to real-time display logic position, actual position and operation speed of every axis;
- (2) Drive object—users determine axis joining simultaneous movement or inserting compensation by selecting drive objects;
- (3) Simultaneous movement—Used to send single-axis drive commands to all the axis of the selected drive object; inserting compensation –Used to send inserting compensation command to all the axis of the selected drive object; stop—stop all the pulse outputs of all axis.

All the above data take pulse as the unit.

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

```
Private Sub Form_Load()
    If Init_Card <= 0 Then
        MsgBox MsgBox "Control card initialization failure!"
    End
    Else
        MsgBox MsgBox "Control card is ready for use!"
    End
End Sub
```

```
End If
For i = 0 To 3
    StartV(i).Text = 100
    Speed(i).Text = 200
    Add(i).Text = 100
Next i
For i = 0 To 3
    Pos(i).Text = 10000
Next i
End Sub
```

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

```
Private Sub CmdPmove_Click()
    If X.value = 1 And Y.value = 1 And Z.value = 1 And W.value = 1 Then
        For i = 1 To 4      'XYZW Simultaneous movement among the 4 axis
            Setup_Speed i, StartV(i - 1).Text, Speed(i - 1).Text, Add(i - 1).Text
            Axis_Pmove i, Pos(i - 1).Text
        Next i
    ElseIf X.value = 1 And Y.value = 1 And Z.value = 1 Then
        For i = 1 To 3      'XYZ' Simultaneous movement among the 3axis
            Setup_Speed i, StartV(i - 1).Text, Speed(i - 1).Text, Add(i - 1).Text
            Axis_Pmove i, Pos(i - 1).Text
        Next i
    ElseIf X.value = 1 And Y.value = 1 And W.value = 1 Then
        For i = 1 To 4      'XYW Simultaneous movement among the 3axis
            If i <> 3 Then
                Setup_Speed i, StartV(i - 1).Text, Speed(i - 1).Text, Add(i - 1).Text
```

```
        Axis_Pmove i, Pos(i - 1).Text
    End If
Next i
ElseIf X.value = 1 And Z.value = 1 And W.value = 1 Then
    For i = 1 To 4      'XZW Simultaneous movement among the 3axis
        If i <> 2 Then
            Setup_Speed i, StartV(i - 1).Text, Speed(i - 1).Text, Add(i - 1).Text
            Axis_Pmove i, Pos(i - 1).Text
        End If
    Next i
ElseIf Y.value = 1 And Z.value = 1 And W.value = 1 Then
    For i = 1 To 4      'YZW Simultaneous movement among the 3axis
        If i <> 1 Then
            Setup_Speed i, StartV(i - 1).Text, Speed(i - 1).Text, Add(i - 1).Text
            Axis_Pmove i, Pos(i - 1).Text
        End If
    Next i
ElseIf X.value = 1 And Y.value = 1 Then
    For i = 1 To 2      'XY Simultaneous movement among the 2axis
        Setup_Speed i, StartV(i - 1).Text, Speed(i - 1).Text, Add(i - 1).Text
        Axis_Pmove i, Pos(i - 1).Text
    Next i
ElseIf X.value = 1 And Z.value = 1 Then      ' XZ Simultaneous movement among the
2axis
    Setup_Speed 1, StartV(0).Text, Speed(0).Text, Add(0).Text
    Axis_Pmove 1, Pos(0).Text
    Setup_Speed 3, StartV(2).Text, Speed(2).Text, Add(2).Text
    Axis_Pmove 3, Pos(2).Text
ElseIf X.value = 1 And W.value = 1 Then ' XW Simultaneous movement among the
2axis
```

Setup\_Speed 1, StartV(0).Text, Speed(0).Text, Add(0).Text  
Axis\_Pmove 1, Pos(0).Text

Setup\_Speed 4, StartV(3).Text, Speed(3).Text, Add(3).Text  
Axis\_Pmove 4, Pos(3).Text

ElseIf Y.value = 1 And Z.value = 1 Then ' YZ Simultaneous movement among the  
2axis

Setup\_Speed 2, StartV(1).Text, Speed(1).Text, Add(1).Text  
Axis\_Pmove 2, Pos(1).Text  
Setup\_Speed 3, StartV(2).Text, Speed(2).Text, Add(2).Text  
Axis\_Pmove 3, Pos(2).Text

ElseIf Y.value = 1 And W.value = 1 Then ' YW Simultaneous movement among  
the2axis

Setup\_Speed 2, StartV(1).Text, Speed(1).Text, Add(1).Text  
Axis\_Pmove 2, Pos(1).Text  
Setup\_Speed 4, StartV(3).Text, Speed(3).Text, Add(3).Text  
Axis\_Pmove 4, Pos(3).Text

ElseIf Z.value = 1 And W.value = 1 Then ' ZW Simultaneous movement among  
the 2axis

Setup\_Speed 3, StartV(2).Text, Speed(2).Text, Add(2).Text  
Axis\_Pmove 3, Pos(2).Text  
Setup\_Speed 4, StartV(3).Text, Speed(3).Text, Add(3).Text  
Axis\_Pmove 4, Pos(3).Text

ElseIf X.value = 1 Then ' X轴驱动  
Setup\_Speed 1, StartV(0).Text, Speed(0).Text, Add(0).Text  
Axis\_Pmove 1, Pos(0).Text

ElseIf Y.value = 1 Then ' Y轴驱动  
Setup\_Speed 2, StartV(1).Text, Speed(1).Text, Add(1).Text  
Axis\_Pmove 2, Pos(1).Text

ElseIf Z.value = 1 Then ' Z轴驱动

```
Setup_Speed 3, StartV(2).Text, Speed(2).Text, Add(2).Text
Axis_Pmove 3, Pos(2).Text
ElseIf W.value = 1 Then                                ' W轴驱动
Setup_Speed 4, StartV(3).Text, Speed(3).Text, Add(3).Text
Axis_Pmove 4, Pos(3).Text
End If
```

End Sub

1.3.4 Inserting compensation codes are inside the click event of CmdInp button, whereby various selected objects send corresponding drive commands. The four check boxes (to select objects) are respectively named as X, Y, Z and W, subject with the following code:

```
Private Sub CmdInp_Click()
If X.value = 1 And Y.value = 1 And Z.value = 1 And W.value = 1 Then
' Inserting compensation of the 4 axis
Setup_Speed 1, StartV(0).Text, Speed(0).Text, Add(0).Text
Interp_Move4 Pos(0).Text, Pos(1).Text, Pos(2).Text, Pos(3).Text
Elseif X.value = 1 And Y.value = 1 And Z.value = 1 Then 'XYZ Inserting
compensation of the 3 axis
Setup_Speed 1, StartV(0).Text, Speed(0).Text, Add(0).Text
Interp_Move3 1, 2, 3, Pos(0).Text, Pos(1).Text, Pos(2).Text
Elseif X.value = 1 And Y.value = 1 And W.value = 1 Then 'XYW Inserting
compensation of the 3 axis
Setup_Speed 1, StartV(0).Text, Speed(0).Text, Add(0).Text
Interp_Move3 1, 2, 4, Pos(0).Text, Pos(1).Text, Pos(3).Text
Elseif X.value = 1 And Z.value = 1 And W.value = 1 Then 'XZW Inserting
compensation of the 3 axis
Setup_Speed 1, StartV(0).Text, Speed(0).Text, Add(0).Text
Interp_Move3 1, 3, 4, Pos(0).Text, Pos(2).Text, Pos(3).Text
Elseif Y.value = 1 And Z.value = 1 And W.value = 1 Then 'XZW Inserting
compensation of the 3 axis
```

---

## ADT8940 4-axis servo/ stepping movement control card

---

```
Setup_Speed 2, StartV(1).Text, Speed(1).Text, Add(1).Text
Interp_Move3 2, 3, 4, Pos(1).Text, Pos(2).Text, Pos(3).Text
Elseif X.value = 1 And Y.value = 1 Then                                'XY Inserting
compensation of the 2 axis
    Setup_Speed 1, StartV(0).Text, Speed(0).Text, Add(0).Text
    Interp_Move2 1, 2, Pos(0).Text, Pos(1).Text
Elseif X.value = 1 And Z.value = 1 Then                                'XZ Inserting compensation of the 2 axis
    Setup_Speed 1, StartV(0).Text, Speed(0).Text, Add(0).Text
    Interp_Move2 1, 3, Pos(0).Text, Pos(2).Text

Elseif X.value = 1 And W.value = 1 Then                                'XW Inserting compensation of the 2 axis
    Setup_Speed 1, StartV(0).Text, Speed(0).Text, Add(0).Text
    Interp_Move2 1, 4, Pos(0).Text, Pos(3).Text
Elseif Y.value = 1 And Z.value = 1 Then                                'YZ Inserting compensation of the 2 axis
    Setup_Speed 2, StartV(1).Text, Speed(1).Text, Add(1).Text
    Interp_Move2 2, 3, Pos(1).Text, Pos(2).Text
Elseif Y.value = 1 And W.value = 1 Then                                'YW Inserting compensation of the 2 axis
    Setup_Speed 2, StartV(1).Text, Speed(1).Text, Add(1).Text
    Interp_Move2 2, 4, Pos(1).Text, Pos(3).Text
Elseif Z.value = 1 And W.value = 1 Then                                'ZW Inserting compensation of the 2 axis
    Setup_Speed 3, StartV(2).Text, Speed(2).Text, Add(2).Text
    Interp_Move2 3, 4, Pos(2).Text, Pos(3).Text
End If
End Sub
```

## 1.4 Monitoring module

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

```
Private Sub Timer1_Timer()  
    Dim value(3) As Long  
    Dim status(4) As Long  
    'Get axis drive information  
    For i = 1 To 4  
        Get_CurrentInf i, value  
        For m = 0 To 2  
            Inf((i - 1) * 3 + m).Caption = value(m)  
        Next m  
    Next i  
    ' Get axis drive status  
    For i = 1 To 4  
        get_status 0, i, status(i - 1)  
    Next i  
    If status(0) = 0 And status(1) = 0 And status(2) = 0 And status(3) = 0 Then  
        CmdPmove.Enabled = True  
        CmdInp.Enabled = True  
    Else  
        CmdPmove.Enabled = False  
        CmdInp.Enabled = False  
    End If  
End Sub
```

## **1.5 Stop module**

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

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

## **☛ VC PROGRAMMING SAMPLES**

### **2.1 Preparation**

- (1) Create a new item and save as "VCExample.dsw";
- (2) Load the static library adt8940.lib into the item following the above-introduced method;

### **2.2 Movement control module**

- (1) Add a new category in the item and save the header as "CtrlCard.h" and source file as "CtrlCard.cpp";
- (2) At first, within the movement control module self-define initialization functions of the movement control card and initialize library functions to be sealed into initialization functions;
- (3) Further self-define relevant movement control functions such as speed setting function, single-axis movement function, and inserting compensation movement function;
- (4) Source codes of the header CtrlCard.h are as follows:

```
#ifndef __ADT8940__CARD__  
#define __ADT8940__CARD__  
/***** Movement control module *****/
```

To simply, conveniently and quickly develop popular, expandable and easy-for-maintenance application systems, we have classified sealed all the library

functions based on the function library of the control card. The following example is subject with one movement control card.

```
*****/
#define MULTIPLE 10 //Multiple rate
#define MAXAXIS 4 //Maximum axis number
class CCtrlCard
{
public:
    int Get_Status(int axis, int &value, int mode);
    int StopRun(int axis,int mode);
    int Get_CurrentInf(int axis, long &LogPos, long &ActPos, long &Speed);
    int Interp_Move4(long value1,long value2,long value3,long value4);
    int Interp_Move3(int axis1,int axis2,int axis3,long value1,long value2,long value3);
    int Interp_Move2(int axis1,int axis2,long value1,long value2);
    int Axis_Pmove(int axis, long value);
    int Setup_Speed(int axis,long startv,long speed,long add);
    int Init_Board();
    CCtrlCard();
    int Result;
};
#endif
```

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

```
# include "stdafx.h"
# include "adt8940.h"
# include "CtrlCard.h"
# include "VCExample.h"
extern int g_CardVer;
CCtrlCard::CCtrlCard()
{
}
```

/\*\*\*\*\*\* Initialization function \*\*\*\*\*/  
'This function contain those library functions frequently used in control card initialization, which is the foundation to call other functions and must be firstly called in this example program.

'Returning value <=0 means initialization failure and >0 means initialization success

\*\*\*\*\*/

```
int CCtrlCard::Init_Board()
{
    Result = adt8940_initial() ;           // Card initialization functions
    if (Result <= 0) return Result;
    for (int i = 1; i<=MAXAXIS; i++){
        set_range (0, i, 500 / MULTIPLE);   // Set range
        set_limit_mode (0, i, 0, 0, 0);     // Set limit mode
        set_command_pos (0, i, 0);         //Set logic counter
        set_actual_pos (0, i, 0);          //Set actual counter
        set_startv (0, i, 1000/ MULTIPLE);  //Set starting speed
        set_speed (0, i, 1000/ MULTIPLE);   //Set drive speed
    }
    return 1;
}
```

/\*\*\*\*\*\*Speed setting module\*\*\*\*\*/

Based on parameter value, judge it's in uniform speed or acceleration/ deceleration  
Returning value =0 means success, and Returning value =1 means error

\*\*\*\*\*/

```
int CCtrlCard::Setup_Speed(int axis, long startv, long speed, long add)
{
    if (startv - speed >= 0) {             //Uniform speed drive
        if (g_CardVer==1){                // Hardware version 1
            Result = set_startv(0, axis, startv/ MULTIPLE);
```

```
        set_speed (0, axis, startv/ MULTIPLE);
    }
    else if (g_CardVer==2){        // Hardware version 2
        Result = set_startv(0, axis, startv);
        set_speed (0, axis, startv);
    }
}
else{                            // Acceleration/ deceleration drive
    if (g_CardVer==1){            // Hardware version 1
        Result = set_startv(0, axis, startv/ MULTIPLE);
        set_speed (0, axis, speed/ MULTIPLE);
        set_acc (0, axis, add/125/ MULTIPLE);
    }
    else if (g_CardVer==2){
        Result = set_startv(0, axis, startv);
        set_speed (0, axis, speed);
        set_acc (0, axis, add/125);
    }
}
return Result;
}
```

/\*\*\*\*\*\* Single-axis drive function\*\*\*\*\*\*/

This function is used to drive movement of a single axis

Returning value =0 means success, and Returning value =1 means error

\*\*\*\*\*/

```
int CCtrlCard::Axis_Pmove(int axis, long value)
```

```
{
    Result = pmove(0, axis, value);
    return Result;
}
```

}

/\*\*\*\*\*\* Any 2-axis inserting compensation function\*\*\*\*\*

This function is used to drive any 2 axis to carry on inserting compensation movement

Returning value =0 means success, and Returning value =1 means error

\*\*\*\*\*/

int CCtrlCard::Interp\_Move2(int axis1, int axis2, long value1, long value2)

{

Result = inp\_move2(0, axis1, axis2, value1, value2);

return Result;

}

/\*\*\*\*\*\* Any 3-axis inserting compensation function\*\*\*\*\*

This function is used to drive any 3 axis to carry on inserting compensation movement

Returning value =0 means success, and Returning value =1 means error

\*\*\*\*\*/

int CCtrlCard::Interp\_Move3(int axis1, int axis2, int axis3, long value1, long value2, long value3)

{

Result = inp\_move3(0, axis1, axis2, axis3, value1, value2, value3);

return Result;

}

/\*\*\*\*\*\* 4-axis inserting compensation function\*\*\*\*\*

This function is used to drive the 4 axis to carry on inserting compensation movement

Returning value =0 means success, and Returning value =1 means error

\*\*\*\*\*/

int CCtrlCard::Interp\_Move4(long value1, long value2, long value3, long value4)

```
{
    Result = inp_move4(0, value1, value2, value3, value4);
    return Result;
}
```

\*\*\*\*\* Get movement information\*\*\*\*\*

This function is used to feedback the current logic position, actual position and drive speed of the selected axis

Returning value =0 means success, and Returning value =1 means error

\*\*\*\*\*/

```
int CCtrlCard::Get_CurrentInf(int axis, long &LogPos, long &ActPos, long &Speed)
```

```
{
    Result = get_command_pos(0, axis, &LogPos);
    get_actual_pos (0, axis, &ActPos);
    get_speed (0, axis, &Speed);
    if (g_CardVer==1){
        Speed=Speed* MULTIPLE;
    }
    return Result;
}
```

\*\*\*\*\* Stop axis drive function\*\*\*\*\*

This function provides either sudden stop mode or deceleration stop mode

Returning value =0 means success, and Returning value =1 means error

\*\*\*\*\*/

```
int CCtrlCard::StopRun(int axis, int mode)
```

```
{
    if (mode == 0)
        Result = sudden_stop(0, axis);           //Sudden stop
    else
        Result = dec_stop(0, axis);             //Deceleration stop
}
```

```

return Result;
}

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

```

## 2.3 Function realization module

### 2.3.1 Interface design



Remark:

- (1) Speed setting part—used to set starting speed, drive speed and acceleration of every axis; position setting—used to set drive pulse for every axis; drive information—used to real-time display logic position, actual position and operation speed of every axis;
- (2) Drive object—users determine axis joining simultaneous movement or inserting compensation by selecting drive objects;
- (3) Simultaneous movement—Used to send single-axis drive commands to all the axis of the selected drive object; inserting compensation –Used to send inserting compensation command to all the axis of the selected drive object; stop—stop all the pulse outputs of all axis.

All the above data take pulse as the unit.

Int g\_CardVer=1 defines global variables, and g\_CardVer indicates hardware version.

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

```
if (g_CtrlCard.Init_Board() <= 0){  
    MessageBox ("Movement control card initialization failure!");  
    return false;  
}  
Else  
    MessageBox ("Movement control card is ready for use!");
```

```
g_CardVer=get_hardware_ver(0);
```

```
UINT
```

```
nID1[]={IDC_EDIT_STARTX,IDC_EDIT_STARTY,IDC_EDIT_STARTZ,IDC_EDIT_STARTW};
```

```
UINT
```

```
nID2[]={IDC_EDIT_XSPEED,IDC_EDIT_YSPEED,IDC_EDIT_ZSPEED,IDC_EDIT_WSPEED};
```

```
UINT
```

```
nID3[]={IDC_EDIT_ADDX,IDC_EDIT_ADDY,IDC_EDIT_ADDZ,IDC_EDIT_ADDW};
```

```
UINT
```

```
nID4[]={IDC_EDIT_POSX,IDC_EDIT_POSY,IDC_EDIT_POSZ,IDC_EDIT_POSW};
```

```
CEdit *pEdit;
    for (int i = 0 ; i<4; i++){           //Data initialization
        pEdit=(CEdit*)GetDlgItem(nID1[i]); //Starting speed: 100
        pEdit->SetWindowText("100");
        pEdit=(CEdit*)GetDlgItem(nID2[i]); //Drive speed: 200
        pEdit->SetWindowText("200");
        pEdit=(CEdit*)GetDlgItem(nID3[i]); //Acceleration: 100
        pEdit->SetWindowText("100");
        pEdit=(CEdit*)GetDlgItem(nID4[i]); //Target position: 10000
        pEdit->SetWindowText("10000");
    }
        SetTimer(1001,100,NULL);        //Activate timer
```

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

```
void CVCExampleDlg::OnButtonPmove()
{
    UpdateData(TRUE);
    long startv[]={m_nStartX,m_nStartY,m_nStartZ,m_nStartW};
    long Speed[]={m_nXSpeed,m_nYSpeed,m_nZSpeed,m_nWSpeed};
    long Add[]={m_nAddX,m_nAddY,m_nAddZ,m_nAddW};
    long Pos[]={m_nPosX,m_nPosY,m_nPosZ,m_nPosW};
    if (m_bX && m_bY && m_bZ && m_bW){           //Simultaneous movement
among the 4 axis
        for (int i = 1; i<5; i++){
            g_CtrlCard.Setup_Speed(i, startv[i-1], Speed[i-1], Add[i-1]);
            g_CtrlCard.Axis_Pmove(i, Pos[i-1]);
        }
    }
}
```

```
else if(m_bX && m_bY && m_bZ){ //XYZ Simultaneous movement among the
3 axis
    for (int i = 1; i<4; i++){
        g_CtrlCard.Setup_Speed(i, startv[i-1], Speed[i-1], Add[i-1]);
        g_CtrlCard.Axis_Pmove(i, Pos[i-1]);
    }
}
else if(m_bX && m_bY && m_bW){ //XYW Simultaneous
movement among the 3 axis
    for (int i = 1; i<5; i++){
        if (i != 3){
            g_CtrlCard.Setup_Speed(i, startv[i-1], Speed[i-1], Add[i-1]);
        }
    }
}
else if(m_bX && m_bZ && m_bW){ //XZW Simultaneous movement among the
3 axis
    for (int i = 1; i<5; i++){
        if (i !=2){
            g_CtrlCard.Setup_Speed(i, startv[i-1], Speed[i-1], Add[i-1]);
            g_CtrlCard.Axis_Pmove(i, Pos[i-1]);
        }
    }
}
else if (m_bY && m_bZ && m_bW){ //YZW Simultaneous movement among
the 3 axis
    for (int i = 1; i<5; i++){
        if (i !=1){
            g_CtrlCard.Setup_Speed(i, startv[i-1], Speed[i-1], Add[i-1]);
        }
    }
}
```

```
    }
}
else if( m_bX && m_bY){    //XY Simultaneous movement among the 2 axis
    for( int i = 1 ; i<3; i++){
        g_CtrlCard.Setup_Speed(i,startv[i-1],Speed[i-1],Add[i-1]);
        g_CtrlCard.Axis_Pmove(i,Pos[i-1]);
    }
}
else if( m_bX && m_bZ){    //XZ Simultaneous movement among the 2 axis
    g_CtrlCard.Setup_Speed(1,startv[0],Speed[0],Add[0]);
    g_CtrlCard.Axis_Pmove(1,Pos[0]);
    g_CtrlCard.Setup_Speed(3,startv[2],Speed[2],Add[2]);
    g_CtrlCard.Axis_Pmove(3,Pos[2]);
}
else if( m_bX && m_bW){    //XW Simultaneous movement among the 2 axis
    g_CtrlCard.Setup_Speed(1,startv[0],Speed[0],Add[0]);
    g_CtrlCard.Axis_Pmove(1,Pos[0]);
    g_CtrlCard.Setup_Speed(4,startv[3],Speed[3],Add[3]);
    g_CtrlCard.Axis_Pmove(4,Pos[3]);
}
else if( m_bY && m_bZ){    //YZ Simultaneous movement among the 2 axis
    g_CtrlCard.Setup_Speed(2,startv[1],Speed[1],Add[1]);
    g_CtrlCard.Axis_Pmove(2,Pos[1]);
    g_CtrlCard.Setup_Speed(3,startv[2],Speed[2],Add[2]);
    g_CtrlCard.Axis_Pmove(3,Pos[2]);
}
else if( m_bY && m_bW){    //YW Simultaneous movement among the 2 axis
    g_CtrlCard.Setup_Speed(2,startv[1],Speed[1],Add[1]);
    g_CtrlCard.Axis_Pmove(2,Pos[1]);
    g_CtrlCard.Setup_Speed(4,startv[3],Speed[3],Add[3]);
```

```
        g_CtrlCard.Axis_Pmove(4,Pos[3]);
    }
    else if( m_bZ && m_bW){ //ZW Simultaneous movement among the 2 axis
        g_CtrlCard.Setup_Speed(3,startv[2],Speed[2],Add[2]);
        g_CtrlCard.Axis_Pmove(3,Pos[2]);
        g_CtrlCard.Setup_Speed(4,startv[3],Speed[3],Add[3]);
        g_CtrlCard.Axis_Pmove(4,Pos[3]);
    }
    else if( m_bX){ //X 轴驱动
        g_CtrlCard.Setup_Speed(1,startv[0],Speed[0],Add[0]);
        g_CtrlCard.Axis_Pmove(1,Pos[0]);
    }
    else if( m_bY){ //Y 轴驱动
        g_CtrlCard.Setup_Speed(2,startv[1],Speed[1],Add[1]);
        g_CtrlCard.Axis_Pmove(2,Pos[1]);
    }
    else if( m_bZ){ //Z 轴驱动
        g_CtrlCard.Setup_Speed(3,startv[2],Speed[2],Add[2]);
        g_CtrlCard.Axis_Pmove(3,Pos[2]);
    }
    else if( m_bW){ //W 轴驱动
        g_CtrlCard.Setup_Speed(4,startv[3],Speed[3],Add[3]);
        g_CtrlCard.Axis_Pmove(4,Pos[3]);
    }
}
```

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

```
void CVCExampleDlg::OnButtonInpmove()
{
    UpdateData(TRUE);
    long startv[]={m_nStartX,m_nStartY,m_nStartZ,m_nStartW};
    long Speed[]={m_nXSpeed,m_nYSpeed,m_nZSpeed,m_nWSpeed};
    long Add[]={m_nAddX,m_nAddY,m_nAddZ,m_nAddW};
    long Pos[]={m_nPosX,m_nPosY,m_nPosZ,m_nPosW};
    if(m_bX && m_bY && m_bZ && m_bW){           //Inserting compensation of
the 4 axis
        g_CtrlCard.Setup_Speed(1,startv[0],Speed[0],Add[0]);
        g_CtrlCard.Interp_Move4(Pos[0],Pos[1],Pos[2],Pos[3]);
    }
    else if(m_bX && m_bY && m_bZ){           //XYZ Inserting compensation
of the 3 axis
        g_CtrlCard.Setup_Speed(1,startv[0],Speed[0],Add[0]);
        g_ctrlCard.Interp_Move3(1,2,3,Pos[0],Pos[1],Pos[2]);
    }
    else if(m_bX && m_bY && m_bW){           //XYW Inserting compensation
of the 3 axis
        g_CtrlCard.Setup_Speed(1,startv[0],Speed[0],Add[0]);
        g_CtrlCard.Interp_Move3(1,2,4,Pos[0],Pos[1],Pos[3]);
    }
    else if(m_bX && m_bZ && m_bW){           //XZW Inserting compensation of
the 3 axis
        g_CtrlCard.Setup_Speed(1,startv[0],Speed[0],Add[0]);
        g_CtrlCard.Interp_Move3(1,3,4,Pos[0],Pos[2],Pos[3]);
    }
    else if(m_bY && m_bZ && m_bW){           //YZW Inserting compensation of
the 3 axis
        g_CtrlCard.Setup_Speed(2,startv[1],Speed[1],Add[1]);
```

```
        g_CtrlCard.Interp_Move3(2,3,4,Pos[1],Pos[2],Pos[3]);
    }
else if(m_bX && m_bY){ //XY Inserting compensation of the
2 axis
    g_CtrlCard.Setup_Speed(1,startv[0],Speed[0],Add[0]);
    g_CtrlCard.Interp_Move2(1,2,Pos[0],Pos[1]);
}
else if(m_bX && m_bZ){ //XZ Inserting compensation of the 2 axis
    g_CtrlCard.Setup_Speed(1,startv[0],Speed[0],Add[0]);
    g_CtrlCard.Interp_Move2(1,3,Pos[0],Pos[2]);
}
else if(m_bX && m_bW){ //XW Inserting compensation of the 2 axis
    g_CtrlCard.Setup_Speed(1,startv[0],Speed[0],Add[0]);
    g_CtrlCard.Interp_Move2(1,4,Pos[0],Pos[3]);
}
else if(m_bY && m_bZ){ //YZ Inserting compensation of the 2 axis
    g_CtrlCard.Setup_Speed(2,startv[1],Speed[1],Add[1]);
    g_CtrlCard.Interp_Move2(2,3,Pos[1],Pos[2]);
}
else if(m_bY && m_bW){ //YW Inserting compensation of the 2 axis
    g_CtrlCard.Setup_Speed(2,startv[1],Speed[1],Add[1]);
    g_CtrlCard.Interp_Move2(2,4,Pos[1],Pos[3]);
}
else if(m_bZ && m_bW){ //ZW Inserting compensation of the 2 axis
    g_CtrlCard.Setup_Speed(3,startv[2],Speed[2],Add[2]);
    g_CtrlCard.Interp_Move2(3,4,Pos[2],Pos[3]);
}
}
```

## **2.4 Monitoring module**

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

```
void CVCEXampleDlg::OnTimer(UINT nIDEvent)
{
    long log,act,spd;
    UINT
        nID1[]={IDC_STATIC_LOGX,IDC_STATIC_LOGY,IDC_STATIC_LOGZ,ID
        C_STATIC_LOGW};
    UINT
        nID2[]={IDC_STATIC_ACTX,IDC_STATIC_ACTY,IDC_STATIC_ACTZ,ID
        C_STATIC_ACTW};
    UINT
        nID3[]={IDC_STATIC_SPEEDX,IDC_STATIC_SPEEDY,IDC_STATIC_SPE
        EDZ,IDC_STATIC_SPEEDW};
    CStatic *lbl;
    CString str;
    int status[4];
    for (int i=1; i<5; i++){
        g_CtrlCard.Get_CurrentInf(i,log,act,spd); //获取当前信息
        lbl=(CStatic*)GetDlgItem(nID1[i-1]); //逻辑位置显示
        str.Format("%ld",log);
        lbl->SetWindowText(str);
        lbl=(CStatic*)GetDlgItem(nID2[i-1]); //实际位置显示
        str.Format("%ld",act);
        lbl->SetWindowText(str);
        lbl=(CStatic*)GetDlgItem(nID3[i-1]); //运行速度显示
        str.Format("%ld",spd);
        lbl->SetWindowText(str);
    }
}
```

```
g_CtrlCard.Get_Status(i,status[i-1],0);    //获取驱动状态
}

CButton *btn;
if (status[0]==0 && status[1]==0 && status[2]==0 && status[3]==0)
{
    btn=(CButton*)GetDlgItem(IDC_BUTTON_PMOVE);
    btn->EnableWindow(TRUE);
    btn=(CButton*)GetDlgItem(IDC_BUTTON_INPMOVE);
    btn->EnableWindow(TRUE);
}
else
{
    btn=(CButton*)GetDlgItem(IDC_BUTTON_PMOVE);
    btn->EnableWindow(FALSE);
    btn=(CButton*)GetDlgItem(IDC_BUTTON_INPMOVE);
    btn->EnableWindow(FALSE);
}
CDialog::OnTimer(nIDEvent);
}
```

## 2.5 Stop module

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

```
void CVCExampleDlg::OnButtonStoprun()
{
    for (int i = 1; i<5; i++){
```

```
        g_CtrlCard.StopRun(i,0);  
    }  
}
```

## Chapter 11 Normal failures and solutions

### ☞ Movement control card detection failure

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

- (1) Check whether drive program for the control card has been installed step by step following installation guide and whether there is the dynamic library file for the control card under the system menu (System32 or System);
- (2) Check touch between the movement control card and the slot; users may test it by re-inserting or changing the slot, alternatively, use a rubber to clean dirt on the golden finger of the control card and re-insert;
- (3) Under the system equipment manager, check whether there is conflict between the movement control card and other hardware. In case of use of PCI card, users may remove other cards or boards first, such as sound card and network card; in case of PC104 card, users may adjust the dialing switch and reset the base address, while the base address used during card initialization must be same as the actual base address;
- (4) Check whether there are any problems with the operating system; users may test it through re-installing other versions of operating systems;
- (5) If failing to find the control card after the above steps, users may change the control card for further detection so as to discover whether there is damage with the control card.

### MOTOR SERVICE FAILURE

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

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

## ADT8940 4-axis servo/ stepping movement control card

---

- Check cable between the control card and the terminal panel;
  - Check whether the pulse and direction signal wire of the motor driver has been correctly connected to the terminal panel;
  - Check connection of the external power supply for the servo driver;
  - Check whether there is alarming status in the servo/ stepping motor driver; in case of any alarm there, follow codes corresponding to alarms to check the reason.
  - Check connection to the servo SON and whether there is excitation status in the servo motor ;
  - In case of servo motor, check control method of the driver; control card of our company support the Position Control Method.
  - Damage to the motor/ driver
- (2) Stepping motor makes abnormal noise during service and motor makes obvious out-steps.
- Calculate motor speed and make sure the stepping motor is under 10-15 rounds per second instead of faster speed;
  - Check internal obstruction in the mechanical part or resistance to the machine;
  - Change to large-moment motors if the current motor is not sufficient;
  - Check current and voltage of the driver; current shall be set as 1.2 of the nominated current and supply voltage shall be within the nominated range;
  - Check the starting speed of the controller; normal starting speed shall be 0.5-1 and the acceleration/ deceleration time shall be over 0.1 second.
- (3) Servo/ stepping motor makes obvious vibration or noises during processing
- Reduce the position ring gain and speed ring gain of the driver while allowed by the positioning precision, if the cause is such ring gains are too big;
  - Adjust machine structure if the cause is poor machine rigidity;
  - Change to large-moment motors if the current motor is not sufficient;
  - Avoid the co-vibration area of the motor or increase partitions so as not to have the speed of stepping motor within the co-vibration area of the motor.

- (4) Motor positions inaccurately
- Check whether the mechanic screw pitch and pulses per round comply with the parameters set in the actual application system, i.e., pulse equivalent;
  - Enlarge position ring gain and speed ring gain in case of servo motor;
  - Check screw gap of the machine in the way of measuring the backward gap of a screw through a micrometer and adjust the screw if there is any gap;
  - In case of inaccurate positioning out of regular time or position, check external disturbance signals;
  - Check whether it is due to non-powerful motor that there is shaking or out-step.
- (5) Motor makes no direction
- Check DR+ DR- cable for connection error or loose connection;
  - Make sure the pulse mode applied in the control card comply with the actual driver mode; this control card support either “pulse + direction” or “pulse + pulse” mode.
  - Check broken cable or loose connection along the motor cable, in case of stepping motor.

#### **🔧 ABNORMAL SWITCH AMOUNT INPUT**

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

- (1) No signal input
- Check whether the wiring is correct according to the above-introduced wiring maps for normal switch and approach switch and ensure the public port for photoelectric coupling of input signals have been connected with anode of internal or external power supply (+12V or 24V);
  - Check switch model and wiring method; the input switch for I/O points of our company is of NPN model.
  - Check whether there is damage with the photoelectric coupler. In case of

normal wiring, input status will not change no matter the input point is broken or closed; users may use multi-use meter to check whether the photoelectric coupler has been broken, and if yes, replace with a new one;

- Check the 12V or 24V power supply to the switch;
- Check whether there is damage to the switch.

### (2) Non-continuous signals

- Check whether there is disturbance by detecting signal status in the I/O test interface; in case of disturbance, increase with Model 104 multiple layer capacitor or apply blocking cables;
- If the machine makes obvious shaking or unusual work stop during normal service, check whether there is disturbance to the limit switch signals or the limit switch work reliably;
- Check connection of external cables.

### (3) Inaccurate reset

- Too high speed decreases reset speed ;
- Check disturbance source if the problem is there is external disturbance to signals;
- Wrong resetting direction;
- Improper installation position of the reset switch or loose switch

### (4) Limit out of use

- Check whether the limit switch still works under the I/O test;
- Too high speed during manual or automatic processing;
- Check disturbance source if the problem is there is external disturbance to signals;
- Wrong manual direction;
- Improper installation position of the reset switch or loose switch

## **Abnormal output of switch amount**

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

(1) Abnormal output

- Check whether the wiring is correct following the above-introduced wiring for output points and ensure the output public port (earthing line) has been connected with the earthing line of the to-be-used power supply;
- Check whether there is any damage to the output components;
- Check whether there is damage with the photoelectric coupler. Users may use multi-use meter to check whether the photoelectric coupler has been broken, and if yes, replace with a new one;
- Safety issue. Continuous diode (model: IN4007 or IN4001) must be serial connected in case of output with sensitive loading.

(2) Judgment method for improper output

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

## **Abnormal encoder performance**

Abnormal encoder performance may be checked in the following method:

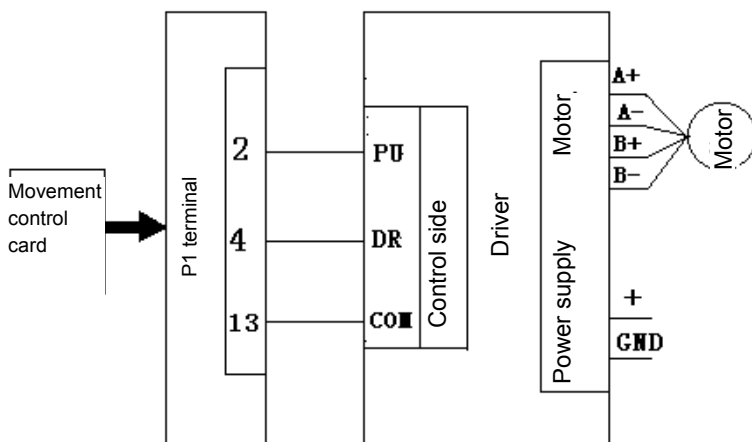
- (1) Check encoder cables and make sure they comply with the above-introduced differential or collecting electrode wiring method;
- (2) Check encoder voltage. The movement control card normally accepts +5V signals. In case a +12V or +24V encoder is selected, users must serial connect a 1K (+12V) resistor between the Phase A /B of the encoder and Phase A /B of the terminal panel;
- (3) Inaccurate encoder counting. External cables to the encoder must be

blocking double-twisted cables, and shall be tied free from those cables with strong disturbance such as strong electricity, specifically, they shall be separated for over 30~50MM.

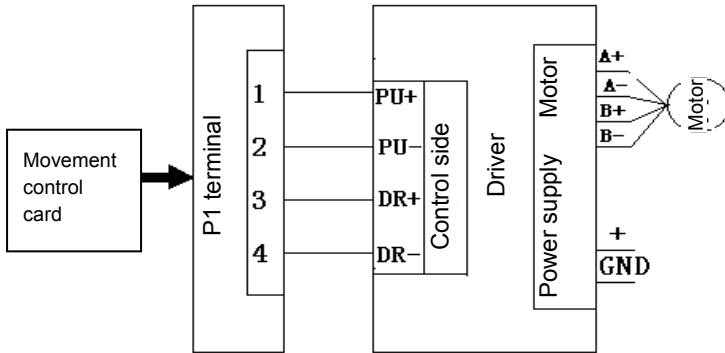
## Appendix A Typical wiring for motor driver

All the following wiring takes X axis as example.

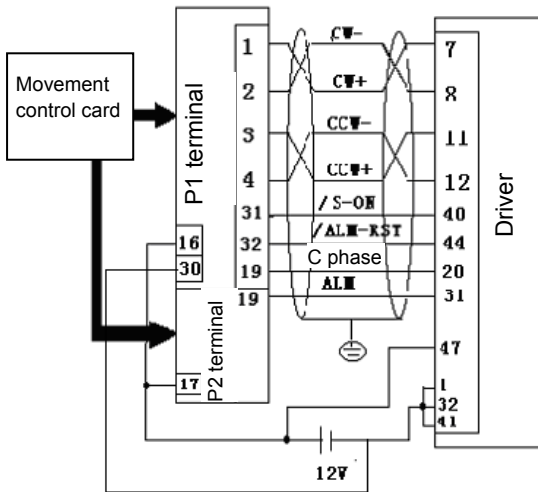
### ☛ Stepping motor driver common anode wiring



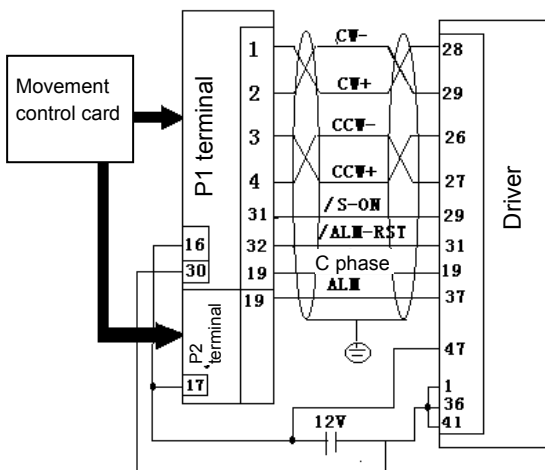
### ☛ Stepping motor driver differential wiring



Yaskawa servo driver wiring



Panasonic A4 servo driver wiring



## Appendix B Introduction on application function library

To ensure users conveniently and quickly program high-quality application system through the movement control card, our company, based on years of frontline development experiences, has sealed basic library functions and generated applicable function library.

An applicable function library contains two parts, i.e., management tools and library functions.

### ➤ Management tools for applicable library functions



Management tools are used to set system parameters and movement parameters of all the axes, which is also the foundation to use applicable library functions. Users can click the Save button to save the setting they have made for each axis. The tools also include the one for adjusting applicable library functions, and users may verify reasonability of their setting through such management tools.

➤ **List of application library functions**

Function category	Function name	Function description	Page
Initialization		Card initialization	82
Back to origin		Sing-axis back to origin	83
		2-axis back to origin	83
		3-axis back to origin	83
		4-axis back to origin	83
Multi-axis		Sing-axis movement	84

**ADT8940 4-axis servo/ stepping movement control card**

<b>simultaneous movement</b>		2-axis simultaneous movement	84
		3-axis simultaneous movement	84
		4-axis simultaneous movement	84
<b>Inserting compensation</b>		2-axis straight line inserting compensation	85
		3-axis straight line inserting compensation	85
		4-axis straight line inserting compensation	85
		2-axis arc inserting compensation	85
<b>Continuous movement</b>		Single-axis continuous movement	85
		2-axis continuous movement	86
		3-axis continuous movement	86
		4-axis continuous movement	86
<b>Manual movement</b>		Manual movement	86
<b>Free movement</b>		Single-axis free movement	86
		2-axis free movement	87
		3-axis free movement	87
		4-axis free movement	87
<b>Stop drive</b>		Stop single-axis drive	87
		Stop all-axis drive	88
<b>Get information</b>		Get logic position	88
		Get actual position	88
		Get drive speed	88
<b>Switch amount</b>		Get status of individual point	88
		Output individual point	88

➤ **Details on applicable library functions**

1. Adt8940\_init\_board—Card initialization  
 Prototype: int adt8940\_init\_board (void);

Function: read parameter files, initialize card, set range/ pulse mode/ stop mode and etc

Returning value: -3—No parameter files (opening management tools can automatically create them); -2—PCI slot error; -1—No port drive installed (users need only install the relevant port drive); 0—No card inserted; 1—Successful initialization.

2. Home1—Sing-axis back to origin

Prototype: int home1(int axis,FUNCTION pfn);

Function: Designated axis to go back to 0 direction and 0 speed to search origin signals following the origin signal type specified in management tools; in case the signal type is just origin, stop0 signals are used, while stop1 signals are for the encoder using Z axis.

Parameters: axis— axis number; **pfn— indicator function, used to transfer the control right so as to ensure to respond to external information during drive, such as to get position and speed and respond to stop.**

Returning value: -2—Fail to detect origin signals; -1—Movement involved with prohibited axis; 0—External stop; 1—Success.

3. Home2—2 axes back to origin

Prototype: int home2(int axis1,int axis2,FUNCTION pfn);

Function: Similar to home1

Parameters: axis1, axis2— axis numbers; pfn— indicator function

Returning value: Same as that for home1

4. Home3—3 axes back to origin

Prototype: int home3(int axis1,int axis2,int axis3,FUNCTION pfn);

Function: Similar to home1

Parameters: axis1, axis2, axis3— axis numbers; pfn— indicator function

Returning value: Same as that for home1

5. Home4—4-axis back to origin

Prototype: int home4 (FUNCTION pfn);

Function: Four axes to individually follow the specified origin type to go back to 0

direction and 0 speed to search for origin signals

Parameters: pfn—indicator function

Returning value: Same as that for home1

6. Work\_move1—Single-axis movement

Prototype: int work\_move1 (int axis,float d,int oppmode,int speedmode, FUNCTION pfn);

Function: Designated axis to move to targeted position in the specified mode

Parameters: axis— axis number; d— movement distance; pfn—indicator function

**Oppmode—0: Relative movement (referring to the current point); 1: Absolute movement (referring to origin)**

**Speedmode—0: Processing speed; 1: Movement speed**

Returning value: -3—Limit signal stop; -2—Soft limit stop; -1—Movement involved with prohibited axis; 0— External stop; 1—Success

7. Work\_move2—2-axis simultaneous movement

Prototype: int work\_move2 (int axis1,float d1,int axis2,float d2,int oppmode,int speedmode,FUNCTION pfn);

Function: Designated axis to move to targeted position following the specified mode

Parameters: axis1, axis2—axis numbers; d1, d2—movement distances; pfn—indicator function; oppmode—movement mode; speedmode—speed mode

Returning value: Same as that for work\_move1

8. Work\_move3—3-axis simultaneous movement

Prototype: int work\_move3 (int axis1,float d1,int axis2,float d2,int axis3,float d3,int oppmode,int speedmode,FUNCTION pfn);

Function: Designated axis to move to targeted position following the specified mode

Parameters: axis1, axis2, axis3— axis numbers; d1, d2, d3— movement distances; pfn— indicator function; oppmode— movement mode; speedmode— speed mode

Returning value: Same as that for work\_move1

9. **Work\_move4—4-axis simultaneous movement**

Prototype: int work\_move4 (float d1,,float d2,float d3,float d4,int oppmode,int speedmode,FUNCTION pfn);

Function: All the four axes to move to targeted position following the specified mode

Parameters: d1, d2, d3, d4— movement distances of respectively X, Y, Z, W axis; pfn— indicator function; oppmode— movement mode; speedmode— speed mode

Returning value: Same as that for work\_move1

10. **Work\_inp2—2-axis straight line inserting compensation**

Prototype: int work\_inp2 (int axis1,float d1,int axis2,float d2,int oppmode,int speedmode,FUNCTION pfn);

Function: Designated axis1 and axis2 move to targeted position in the specified straight line inserting compensation mode

Parameters: Same as that for 2-axis simultaneous movement

Returning value: Same as that for work\_move1

11. **Work\_inp3—3-axis straight line inserting compensation**

Prototype: int work\_inp3 (int axis1,float d1,int axis2,float d2,int axis3,float d3,int oppmode,int speedmode,FUNCTION pfn);

Function: Designated axis1, axis 2 and axis3 move to targeted position in the specified straight line inserting compensation mode

Parameters: Same as that for 3-axis simultaneous movement

Returning value: Same as that for work\_move1

12. **Work\_inp4—4-axis straight line inserting compensation**

Prototype: int work\_inp4(float d1,float d2,float d3,float d4,int oppmode,int speedmode,FUNCTION pfn);

Function: All the four axes move to targeted position in the specified straight line inserting compensation mode

Parameters: Same as that for 4-axis simultaneous movement

Returning value: Same as that for work\_move1

13. **Work\_arc—2-axis arc inserting compensation**  
Prototype: `int work_arc (int axis1,int axis2,float cood[ ],int speedmode,FUNCTION pfn)`  
Function: Any 2 axis to make arc inserting compensation movement following the specified speed mode  
Parameters: axis1, axis2—axis number; cood[ ]—coordinates of 3 points at an arc, including 6 factors  
Returning value: -4—The 3 points fail to form an arc; other returning values are same as that for work\_move1
14. **Continue\_move1—Single-axis continuous movement**  
Prototype: `int continue_move1(int axis,int dir, int speedmode,FUNCTION pfn);`  
Function: Single axis continuously moves until limit signal or external stop  
Parameters: axis— axis number; dir— direction (0: positive direction; 1: negative direction); pfn—indicator function  
Returning value: Same as that for work\_move1
15. **Continue\_move2—2-axis continuous movement**  
Prototype: `int continue_move2 (int axis1,int dir1,int axis2,int dir2, int speedmode,FUNCTION pfn);`  
Function: 2 axis continuously move until limit signal or external stop  
Parameters: axis1, axis2— axis numbers; dir1, dir2— directions (0: positive direction; 1: negative direction); pfn—indicator function  
Returning value: Same as that for work\_move1
16. **Continue\_move3—3-axis continuous movement**  
Prototype: `int continue_move3 (int axis1,int dir1,int axis2,int dir2,int axis3,int dir3, int speedmode,FUNCTION pfn);`  
Function: 3 axis continuously move until limit signal or external stop  
Parameters: axis1, axis2, axis3— axis numbers; dir1, dir2, dir3— directions (0: positive direction; 1: negative direction); pfn—indicator function  
Returning value: Same as that for work\_move1
17. **Continue\_move4—4-axis continuous movement**  
Prototype: `int continue_move4(int dir1,int dir2,int dir3,int dir4, int speedmode,FUNCTION pfn);`  
Function: 4 axis continuously move until limit signal or external stop

Parameters: dir1, dir2, dir3, dir4— directions (0: positive direction; 1: negative direction); pfn—indicator function

Returning value: Same as that for work\_move1

**18. Hand\_move—Manual function**

Prototype: int hand\_move (int axis,int dir,FUNCTION pfn);

Function: Designated axis continuously moves following the manual speed and specified direction until limit signal or external stop

Parameters: axis — axis number; dir— directions (0: positive direction; 1: negative direction); pfn—indicator function

Returning value: -3—Limit signal stop; -1—Movement involved with prohibited axis; 0—External stop

**19. Free\_move1—Single-axis free movement**

Prototype: int free\_move1 (int axis,float d,float startv,float speed,float addtime,int oppmode,FUNCTION pfn)

Function: Designated axis moves to targeted position following the specified speed and direction

Parameters: axis— axis number; d— targeted position; startv— starting speed; speed— drive speed; addtime— acceleration time; oppmode—movement mode (0: Relative movement; 1: Absolute movement); pfn—indicator function

Returning value: Same as that for work\_move1

**20. Free\_move2—2-axis free movement**

Prototype: int free\_move2(int axis1,float d1,float startv1,float speed1,float addtime1,int axis2,float d2,float startv2,float speed2,float addtime2,int oppmode,FUNCTION pfn);

Function: Designated axis moves to targeted position following the specified speed and movement mode

Parameters: Similar to that for free\_move1

Returning value: Same as that for work\_move1

**21. Free\_move3—3-axis free movement**

Prototype: int free\_move3 (int axis1,float d1,float startv1,float speed1,float addtime1,int axis2,float d2,float startv2,float speed2,float addtime2,int axis3,float startv3,float speed3,float addtime3,int oppmode,FUNCTION pfn);

Function: Designated axis moves to targeted position following the specified speed and movement mode

Parameters: Similar to that for free\_move1

Returning value: Same as that for work\_move1

**22. Free\_move4—4-axis free movement**

Prototype: int free\_move4 (float d1,float startv1,float speed1,float addtime1,float d2,float startv2,float speed2,float addtime2,float startv3,float speed3,float addtime3,float startv4,float speed4,float addtime4,int oppmode,FUNCTION pfn);

Function: All the four axis moves to targeted position following the specified speed and movement mode

Parameters: Similar to that for free\_move1

Returning value: Same as that for work\_move1

**23. One\_stop—Stop single-axis drive**

Prototype: int one\_stop (int axis,int mode);

Function: Stop drive of designated axis in designate mode

Parameters: axis— axis number; mode—stop mode (0: sudden stop; 1: decelerating stop)

**24. All\_stop—Stop all-axis drive**

Prototype: int all\_stop(int mode);

Function: Stop drive of all the axis in designate mode

Parameters: mode— stop mode (0: sudden stop; 1: decelerating stop)

**25. Get\_logical\_pos—Get logic position**

Prototype: float get\_logical\_pos (int axis,int mode);

Function: Get logic position of designated axis in designated mode

Parameters: axis— axis number; mode— position mode (0: distance mode; 1: pulse mode);

Returning value: logic position

**26. Get\_fact\_pos—Get actual position**

Prototype: float get\_fact\_pos(int axis,int mode);

Function: Get actual position of designated axis in designated mode

Parameters: axis— axis number; mode— position mode (0: distance mode; 1: pulse mode);

Returning value: actual position

**27. Get\_move\_speed—Get movement speed**

Prototype: float get\_move\_speed (int axis,int mode);

Function: Get movement speed of designated axis in designated mode

Parameters: axis— axis number; mode— position mode (0: distance mode; 1: pulse mode);

Returning value: movement speed

**28. Get\_input—Get status of individual point**

Prototype: int get\_input (int number);

Function: Get status of individual input point

Parameters: number—Input point

Returning value: 0—low level; 1—high level; -1—error

**29. Set\_output—Output individual point**

Prototype: int set\_output (int number,int value);

Function: Set status of designated output points

Parameters: number— Output number; value—0: low; 1: high

Returning value: 0—Correct; 1: Wrong