

## I-8090 3-axis encoder card

I-8090 is a 3-axis encoder counter board on I-8000 platform. I-8090 encoder card has internal digital filter, 16 bits counter and high counting rate 1Mpps. The application of I-8090 board is position/distance measurement, velocity measurement, feedback for motor control, handwheel input and so on.

A system including I-8000 (main system), I-8091 (2-axis stepping/servo control card), I-8090 (3-axis encoder card) can be implemented as a standalone motion controller system for low cost automatical machine.

### Features

- I-8000 series.
- 3-axis, 16 bits encoder counter.
- 32 bits encoder counter by software.
- Maximum counting rate : 1M pulse/sec.
- Differential input A+, A-, B+, B-, C+, C-.
- Quadrant counting mode, CW/CCW counting mode, Pulse/Dir counting mode.
- 2500V optical isolation

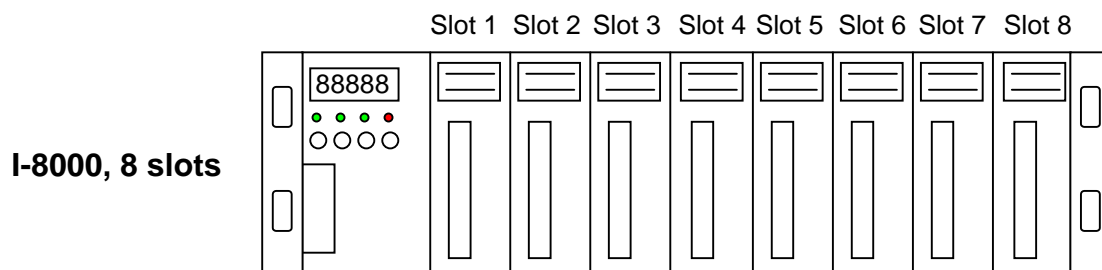
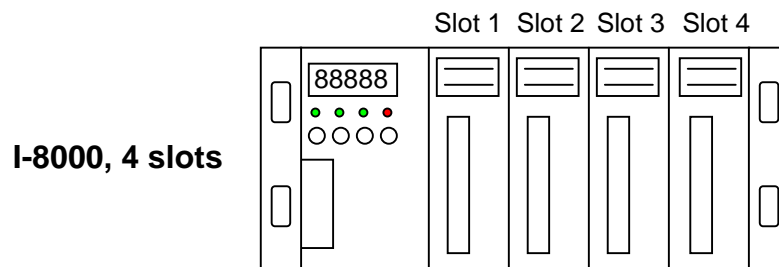
# 1. Hardware

## 1.1 I-8000 hardware address

The hardware address of I-8000 main system is fixed as following table.

There are 4 slots I-8000 and 8 slots I-8000.

	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7	Slot 8
I-8000, 4 slot Address	0x080	0x0A0	0x0C0	0x0E0	---	---	---	---
I-8000, 8 slot Address	0x080	0x0A0	0x0C0	0x0E0	0x140	0x160	0x180	0x1A0



Fig(1) I-8000 hardware address

## 1.2 Registers of I-8090 board

The **I-8090** card's registers table as following.

Register	Add.	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ID	0x00	R	0x0D							
XDATA	0x01	R	X-axis encoder value							
YDATA	0x02	R	Y-axis encoder value							
ZDATA	0x03	R	Z-axis encoder value							
INDEX	0x04	R						ZI	YI	XI
XCTRL	0x00	W			S1	S0		/RST	/INH	/SEL
YCTRL	0x01	W			S1	S0		/RST	/INH	/SEL
ZCTRL	0x02	W			S1	S0		/RST	/INH	/SEL

Register	Add.	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ID	0x00	R	0x0D							

The ID register is read only and its value is fixed as 0x0D. User can check this register to identify I-8090 card or not.

Register	Add.	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
XDATA	0x01	R	X-axis encoder value							

XDATA: the X-axis encoder counter value can be read out from this register. The low byte value of 16 bits encoder counter can be read out when set /SEL=0 (XCTRL register), the high byte can be read out when set /SEL=1 (XCTRL register).

Register	Add.	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
YDATA	0x02	R	Y-axis encoder value							

YDATA : the Y-axis encoder counter value can be read out from this register. The low byte value of 16 bits encoder counter can be read out when set /SEL=0 (YCTRL register), the high byte can be read out when set /SEL=1 (YCTRL register).

Register	Add.	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ZDATA	0x03	R	Z-axis encoder value							

ZDATA : the Z-axis encoder counter value can be read out from this register. The low byte value of 16 bits encoder counter can be read out when set

/SEL=0 (ZCTRL register), the high byte can be read out when set /SEL=1 (ZCTRL register).

Register	Add.	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INDEX	0x04	R						ZI	YI	XI

The index input C+/C- can read out from this register. These bits are active high.

XI : indicate the index of X-axis (C+/C- input).

YI : indicate the index of Y-axis (C+/C- input).

ZI : indicate the index of Z-axis (C+/C- input).

Register	Add.	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
XCTRL	0x00	W			S1	S0		/RST	/INH	/SEL
YCTRL	0x01	W			S1	S0		/RST	/INH	/SEL
ZCTRL	0x02	W			S1	S0		/RST	/INH	/SEL

The XCTRL, YCTRL and ZCTRL register are control registers for X-axis, Y-axis, Z-axis respectively.

**/RST** : reset counter to zero

**/INH** : inhibit the counter data latch. This bit must be set 0 before read out the counter value to inhibit the counter data latch to DATA registers.

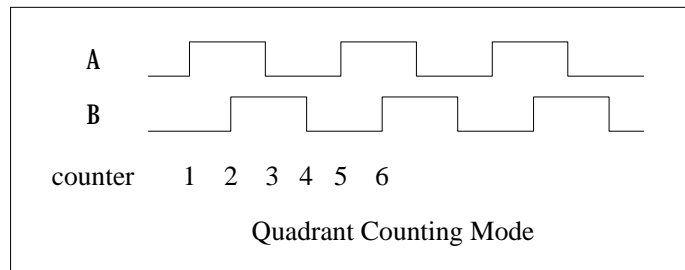
**/SEL** : to select low byte or high byte for reading the counter value.

0 : low byte

1 : high byte

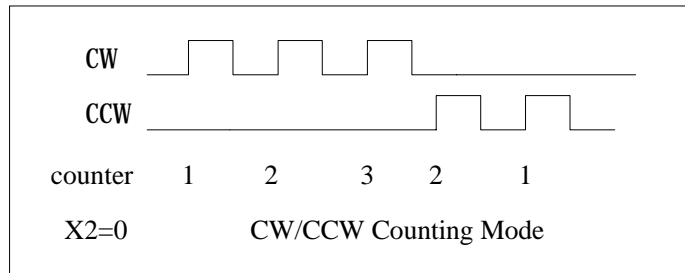
**S1, S0** : to select counting mode

**00** : quadrant counting mode



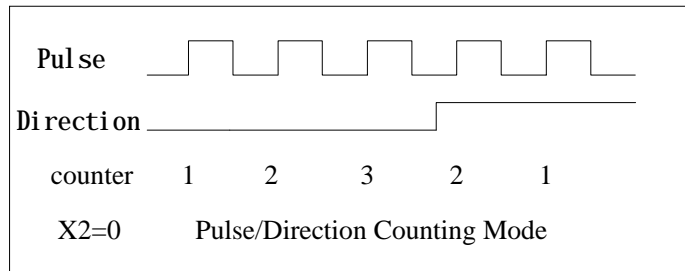
Fig(2) Quadrant counting mode

**01** : CW/CCW counting mode



Fig(3) CW/CCW counting mode

**10** : Pulse/Direction counting mode



Fig(4) Pulse/Direction counting mode

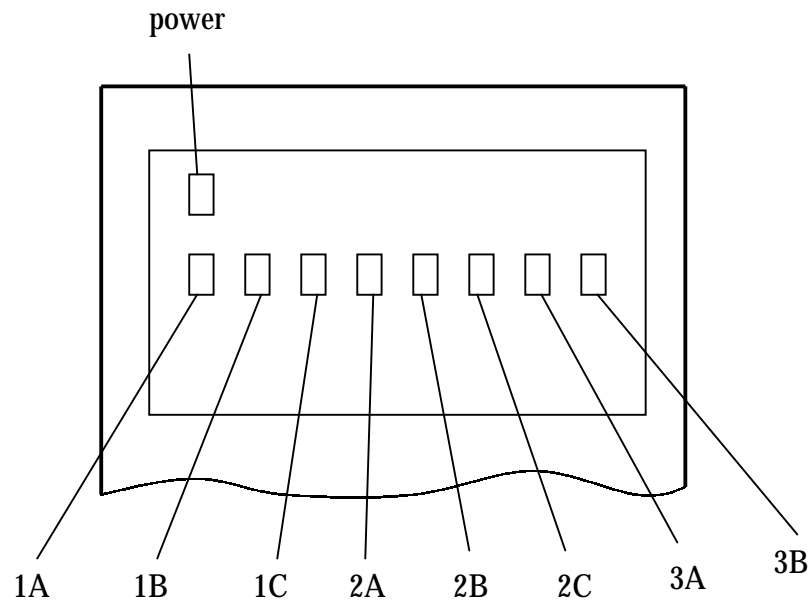
Example: assign counting mode

```
x_mode=y_mode=z_mode=0x00;
card[cardNo].ctrl1 = 0x07 | x_mode;
card[cardNo].ctrl2 = 0x07 | y_mode;
card[cardNo].ctrl3 = 0x07 | z_mode;
outportb(card[cardNo].base + WR1, card[cardNo].ctrl1);
outportb(card[cardNo].base + WR2, card[cardNo].ctrl2);
outportb(card[cardNo].base + WR3, card[cardNo].ctrl3);
```

Example: read X-axis encoder value

```
card[cardNo].ctrl1 &= 0xFC; //1111 1100 low byte
outportb(card[cardNo].base + WR1, card[cardNo].ctrl1);
value = inportb(card[cardNo].base + RD1);
card[cardNo].ctrl1 |= 0x01; //0000 0001 high byte
outportb(card[cardNo].base + WR1, card[cardNo].ctrl1);
value += inportb(card[cardNo].base + RD1)*256;
card[cardNo].ctrl1 |= 0x03; //0000 0011
outportb(card[cardNo].base + WR1, card[cardNo].ctrl1);
```

## 1.3 LED Indicator



Fig(5) I-8090 LED indicator

Where

1A, 1B, 1C indicate X-axis's 1A+/1A-, 1B+/1B-, 1C+/1C- signal input.

2A, 2B, 2C indicate Y-axis's 2A+/2A-, 2B+/2B-, 2C+/2C- signal input.

3A, 3B, indicate Z-axis's 3A+/3A-, 3B+/3B- signal input.

## 1.4 Connection

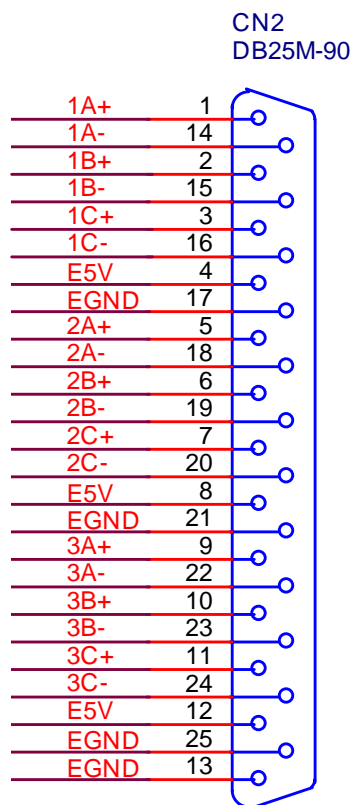


Fig (6) Pin out of CN2 connector

Table of CN2 connector

Pin name	Pin number	description
1A+	1	A+ input of X-axis encoder
1A-	14	A- input of X-axis encoder
1B+	2	B+ input of X-axis encoder
1B-	15	B- input of X-axis encoder
1C+	3	C+ input of X-axis encoder
1C-	16	C- input of X-axis encoder
E5V	4	Isolated 5V supply, max. 50mA (sum of pin 4,8,12)
EGND	17	Signal ground
2A+	5	A+ input of Y-axis encoder
2A-	18	A- input of Y-axis encoder
2B+	6	B+ input of Y-axis encoder
2B-	19	B- input of Y-axis encoder
2C+	7	C+ input of Y-axis encoder
2C-	20	C- input of Y-axis encoder

E5V	8	Isolated 5V supply, max. 50mA (sum of pin 4,8,12)
EGND	21	Signal ground
3A+	9	A+ input of Z-axis encoder
3A-	22	A- input of Z-axis encoder
3B+	10	B+ input of Z-axis encoder
3B-	23	B- input of Z-axis encoder
3C+	11	C+ input of Z-axis encoder
3C-	24	C- input of Z-axis encoder
E5V	12	Isolated 5V supply, max. 50mA (sum of pin 4,8,12)
EGND	25	Signal ground
EGND	13	Signal ground

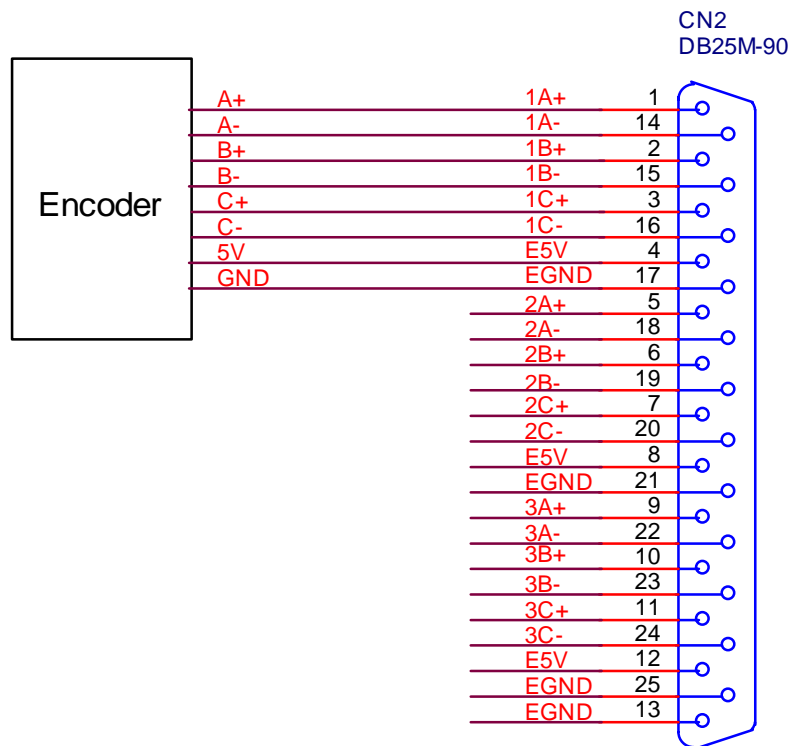


Fig (7) Connection between encoder and I-8090 card



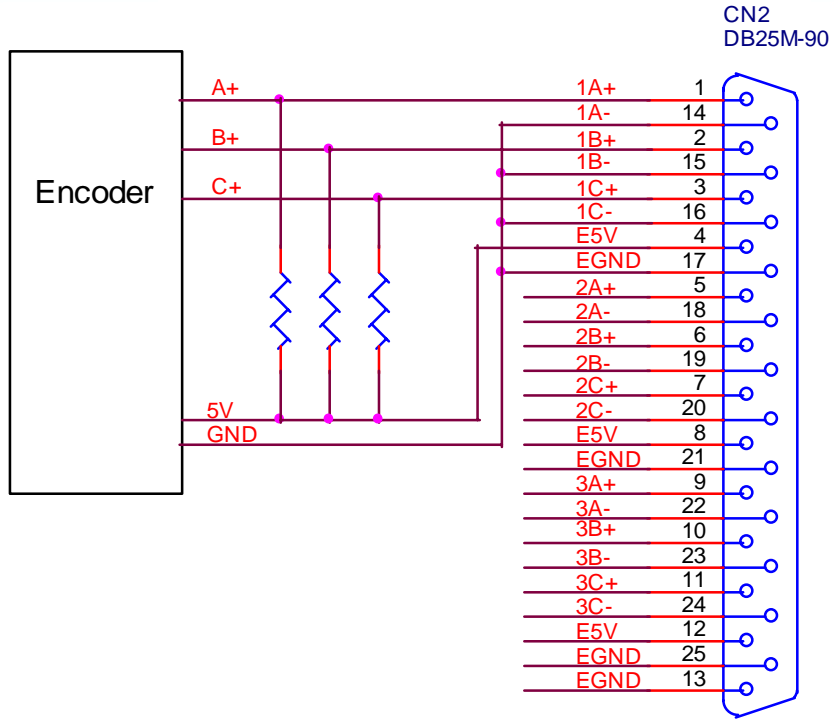


Fig (8) Connection between open collector type encoder and I-8090 card

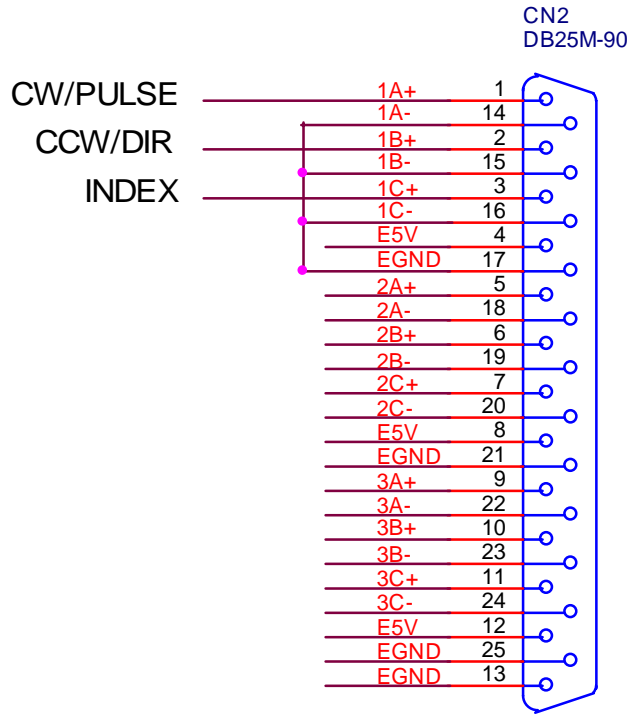


Fig (9) The connection for CW/CCW or Pulse/Direction counting mode

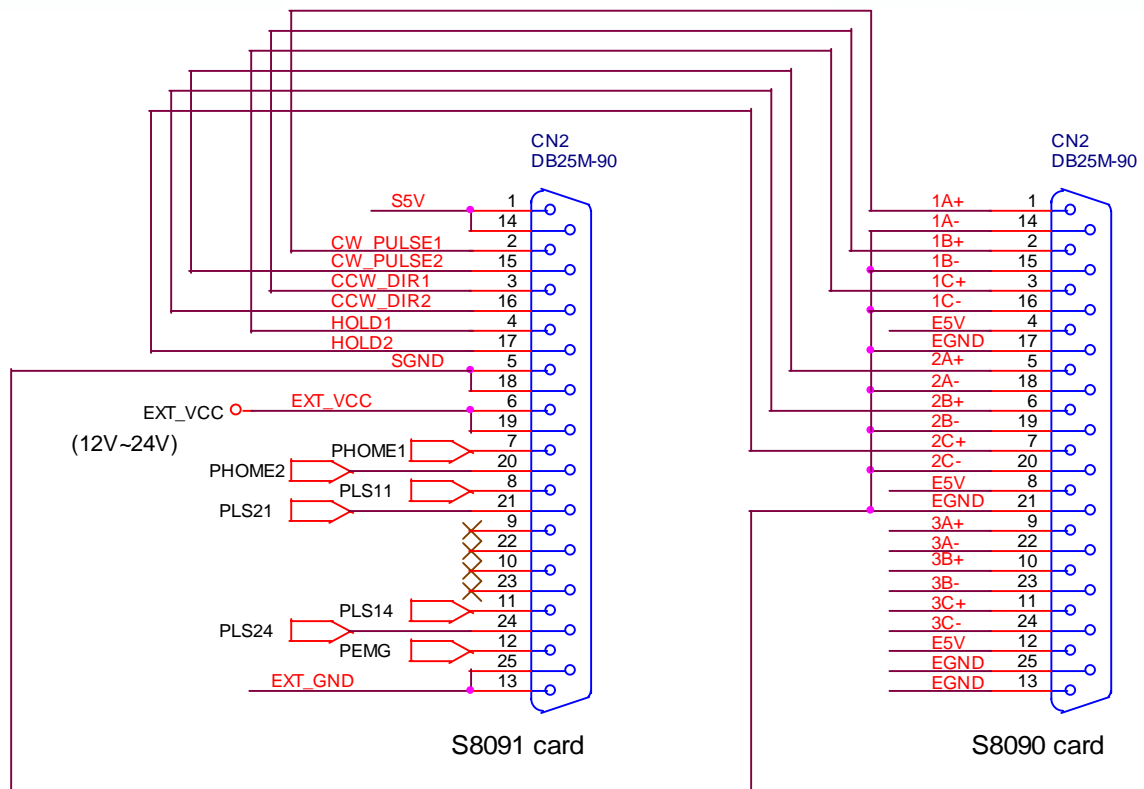


Fig (10) The connection between I-8090 and I-8091 for function testing or pulse feedback by I-8090 encoder card.

## 2. Software

---

User's applications could be compiled under DOS Turbo/Borland C/C++ environment. It should be include i8090.h and i8090.LIB to compile the target execution file. The execution files can be downloaded under I-8000 main system (execute 7188x.exe), and then run the target execution file as under PC system. About the I-8000's resource or environment, please refer to the manual of I-8000 system or its software programming guide.

The following section will introduce the I-8090's functions and examples.

### 2.1 constants and functions

#### Constants

```
#define YES    1
#define NO     0
#define ON     1
#define OFF    0

#define X_axis    1
#define Y_axis    2
#define Z_axis    3

#define ENC_QUADRANT    0x00
#define ENC_CW_CCW     0x10
#define ENC_PULSE_DIR  0x20
```

#### Functions

##### **(1) unsigned char i8090\_REGISTRATION(unsigned char cardNo, unsigned int address)**

In order to distinguish more than one I-8090 card in I-8000 platform, the I-8090 cards should be registrated before using it. This command will assign a card number="cardNo" to I-8090 card address="address" . If there is not I-8090 at the given address, this command will return "NO".

cardNo: 0~19, assign the address as which card.

address: hardware address which defined at chapter 1.1

Return: “YES” : registration successful

“NO” : registration failure.

Example: This example will assign I-8090 card address=0x080 as CARD1 (1). Then initial the I-8090 card and reset X,Y,Z axis encoder counter value to 0.

```
#define CARD1 1
```

```
...
```

```
i8090_REGISTRATION(CARD1, 0x080);
```

```
i8090_INIT_CARD(CARD1, ENC_QUADRANT, ENC_QUADRANT,  
                ENC_QUADRANT);
```

```
i8090_RESET_ENCODER(CARD1, X_axis);
```

```
i8090_RESET_ENCODER(CARD1, Y_axis);
```

```
i8090_RESET_ENCODER(CARD1, Z_axis);
```

**(2) void i8090\_INIT\_CARD(unsigned char cardNo,  
                           unsigned char x\_mode,  
                           unsigned char y\_mode,  
                           unsigned char z\_mode)**

This command will reset all three axis's counter value of “cardNo” card, and assign its counting mode. The counting mode (S1,S0) has been explained in registers XCTRL, YCTRL, ZCTRL.

cardNo: 0~19, select which card.

x\_mode, y\_mode, z\_mode: select the counting mode.

0x00 : quadrant counting mode

0x10 : CW/CCW counting mode

0x20 : Pulse/Direction counting mode

Example:

```
#define ENC_QUADRANT 0x00
```

```
#define ENC_CW_CCW 0x10
```

```
#define ENC_PULSE_DIR 0x20
```

i8090\_INIT\_CARD(CARD1, ENC\_QUADRANT, ENC\_QUADRANT,  
 ENC\_QUADRANT);

**(3) unsigned int i8090\_GET\_ENCODER(unsigned char cardNo,  
 unsigned char axis)**

This command will return the counter value of the selected “axis” and “cardNo”.

cardNo: 0~19, select which card.

axis : select which axis.

1 : X-axis

2 : Y-axis

3 : Z-axis

return : a 16 bits unsigned integer value.

**(4) void i8090\_RESET\_ENCODER(unsigned char cardNo, unsigned  
 char axis)**

This command will reset the counter value of the selected “axis” and “cardNo”.

cardNo: 0~19, select which card.

axis : select which axis.

1 : X-axis

2 : Y-axis

3 : Z-axis

**(5) unsigned char i8090\_GET\_INDEX(unsigned char cardNo)**

It will return the “INDEX” register’s value of the selected “cardNo” card.

cardNo: 0~19, select which card.

Register	Add.	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INDEX	0x04	R						ZI	YI	XI

The index input C+/C- can read out from this register. These bits are active high.

XI : indicate the index of X-axis.

YI : indicate the index of Y-axis.

ZI : indicate the index of Z-axis.

### 32 bits encoder counts command sets

(6) void **i8090\_ENCODER32\_ISR(unsigned char cardNo)**

(7) void **i8090\_RESET\_ENCODER32(unsigned char cardNo, unsigned char axis)**

(8) long **i8090\_GET\_ENCODER32(unsigned char cardNo, unsigned char axis)**

cardNo: 0~19, select which card.

axis : select which axis.

1 : X-axis

2 : Y-axis

3 : Z-axis

The above three commands provided a software method to get 32 bits encoder counts.

The **i8090\_ENCODER32\_ISR(unsigned char cardNo)** command calculates the difference pulse between present and last time, and then add this difference into a "*long type*" variable. According to this idea, so, the **i8090\_ENCODER32\_ISR()** command should be executed periodically in 2~10ms by timer interrupt or manually call it.

The **i8090\_RESET\_ENCODER32((unsigned char cardNo, unsigned char axis)** command can reset the "*long type*" variable to zero.

The long **i8090\_GET\_ENCODER32(unsigned char cardNo, unsigned char axis)** command can return the value of the "*long type*" variable.