i-8094H/i-8094A Motion Control Module
User Manual

(Version 1.3)

Function Library for

WinCon -8000 series PAC controllers

ICP DAS CO., LTD.

泓格科技股份有限公司

Warranty

All products manufactured by ICPDAS Inc. are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICPDAS Inc. assumes no liability for damages consequent to the use of this product. ICPDAS Inc. reserves the right to change this manual at any time without notice. The information furnished by ICPDAS Inc. is believed to be accurate and reliable. However, no responsibility is assumed by ICPDAS Inc. for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Trademark

The names used for identification only maybe registered trademarks of their respective companies.

License

The user can use, modify and backup this software on a single machine. The user may not reproduce, transfer or distribute this software, or any copy, in whole or in part.

Content

1 Preface

1.1 Introduction

- **This manual provides complete and detailed description of i8094H functions for users to develop programs for their control of automatic equipments. Many examples are included in this manual for reference to write efficient application programs.**
- **This manual includes seven chapters. This chapter gives a brief description of this manual. Chapter 2 to 6 is the descriptions of macro functions (MF). Chapter 7contains macro functions (MF) specially for i8094H.**
- **The functions defined in DLL file are described here. This DLL can be used on different developing software platforms, such as eVC++, VB.net, and C#.net, and different OS systems (MiniOS7 / WinCE / Linux).**

1.2 Funciton description

All functions are described by the following parts:

- **Function_name (parameter1, parameter2, …)**

Description: **Explanation of this function.**

Parameters: **Definitions of the parameters and how to use them.**

Return: **The return value of this function.**

Example: **Simple example program in C++.**

Remark: **Comments.**

1.3 Function categories description

RTC (Real Time Command) : Enable i8094H to execute real time command

MP (Macro Program) : Functions to be executed after MP_CREATE

ISR (Interrupt Service Routine) : Functions to be executed in ISR after MP_CREATE

IT (Initial Table) : Functions to be executed in parameters table

Maximum number of Function Line for ISR1 ~ ISR20 and MP1 ~ MP157

ISR(6)	ISR1	ISR2	ISR3	ISR4	ISR5	ISR6			
Total:	8	8	8	8	8	8			
ISR(9)	ISR7	ISR8	ISR9	ISR10	ISR11	ISR12	ISR13	ISR14	ISR15
Total:	16	16	16	16	16	16	16	16	16
ISR(3)	ISR16	ISR17	ISR18						
Total:	32	32	32						
ISR(2)	ISR19	ISR20							
Total:	64	64							
MP(40)	MP1	~	MP40						
Total:	8		8						
MP(50)	MP41	~	MP90						
Total:	16		16						
MP(40)	MP91	~	MP130						
Total:	32		32						
MP(20)	MP131	~	MP150						
Total:	64		64						
MP(5)	MP151	MP152	MP153	MP154	MP155				
Total:	128	128	128	128	128				
MP(2)	MP156	MP157							
Total:	512	512							

In the following Function Table, most functions in sections 2,3,4,5,6 could be used in i8094H_MP_CREATE (please refer to 7.2.1), all values could be replaced by variables(when applied to MP or ISP).

b*varNo*: User-defined variables: bVAR0 ~ bVAR127 (Data type :BYTE)

***varNo*: User-defined variables: VAR0 ~ VAR511 (Data type :longorDWORD)**

Note: In the following sections * indicates functions applied to MP.

In the following sections * indicates functions applied to ISR.

Section	Function	RTC	MP	ISR	IT
Version					
2.2	i8094H_GET_FW_VERSION	◎			
Basic settings					
2.3	i8094H_RESET_CARD	◎			
2.3	i8094H_CLEAR_CARD_BUFFER	◎			
2.4	i8094H_SET_PULSE_MODE	◎	◎		◎
2.5	i8094H_SET_MAX_V	◎	◎		◎
2.6	i8094H_SET_HLMT	◎	◎		◎
2.7	i8094H_LIMITSTOP_MODE	◎	◎		◎
2.8	i8094H_SET_NHOME	◎	◎		◎
2.9	i8094H_SET_HOME_EDGE	◎	◎		◎
2.10	i8094H_SET_SLMT	◎	◎		◎
2.10	i8094H_CLEAR_SLMT	◎	◎		◎
2.11	i8094H_SET_ENCODER	◎	◎		◎
2.12	i8094H_SERVO_ON	◎	◎		◎
2.12	i8094H_SERVO_OFF	◎	◎		◎
2.13	i8094H_SET_ALARM	◎	◎		◎
2.14	i8094H_SET_INPOS	◎	◎		◎
2.15	i8094H_SET_FILTER	◎	◎		◎
2.16	i8094H_VRING_ENABLE	◎	◎		◎
2.16	i8094H_VRING_DISABLE	◎	◎		◎
2.17	i8094H_AVTRI_ENABLE	◎	◎		◎
2.17	i8094H_AVTRI_DISABLE	◎	◎		◎
2.18	i8094H_EXD_MP	◎			
2.18	i8094H_EXD_FP	◎			
2.18	i8094H_EXD_CP	◎			
2.18	i8094H_EXD_DISABLE	◎			
2.19	i8094H_READ_bVAR	◎			
2.19	i8094H_WRITE_bVAR	◎			

2.19	i8094H_READ_VAR	◎			
2.19	i8094H_WRITE_VAR	◎			
2.20	i8094H_READ_MD	◎			
2.20	i8094H_WRITE_MD	◎			
Status reading and setting					
3.1	i8094H_SET_LP	◎	◎	◎	
3.1	i8094H_GET_LP	◎	◎	◎	
3.2	i8094H_SET_EP	◎	◎	◎	
3.2	i8094H_GET_EP	◎	◎	◎	
3.3	i8094H_GET_CV	◎			
Section	Function	RTC	MP	ISR	IT
3.4	i8094H_GET_CA	◎			
3.5	i8094H_GET_DI	◎	◎	◎	
3.5	i8094H_GET_DI_ALL	◎	◎	◎	
3.6	i8094H_GET_ERROR	◎	◎	◎	
3.6	i8094H_GET_ERROR_CODE	◎	◎	◎	
FRnet function					
4.1	i8094H_FRNET_IN	◎	◎	◎	
4.2	i8094H_FRNET_OUT	◎	◎	◎	
Home search					
5.1	i8094H_SET_HV	◎	◎		
5.2	i8094H_HOME _LIMIT	◎	◎		
5.3	i8094H_SET_HOME_MODE	◎	◎		
5.4	i8094H_HOME _START	◎	◎		
Independent Axis Motion Control					
6.1.1	i8094H_NORMAL_SPEED	◎	◎	◎	
6.1.2	i8094H_SET_SV	◎	◎	◎	
6.1.3	i8094H_SET_V	◎	◎	◎	
6.1.4	i8094H_SET_A	◎	◎	◎	
6.1.5	i8094H_SET_D	◎	◎	◎	
6.1.6	i8094H_SET_K	◎	◎	◎	
6.1.7	i8094H_SET_L	◎	◎	◎	
6.1.8	i8094H_SET_AO	◎	◎	◎	
6.1.9	i8094H_FIXED_MOVE	◎	◎	◎	
6.1.9	i8094H_SET_PULSE	◎	◎	◎	
6.1.10	i8094H_CONTINUE_MOVE	◎	◎	◎	
Interpolation Motion					
6.2.1	i8094H_AXIS_ASSIGN	◎	◎	◎	

6.2.2	i8094H_VECTOR_SPEED	◎	◎	◎	
6.2.3	i8094H_SET_VSV	◎	◎	◎	
6.2.4	i8094H_SET_VV	◎	◎	◎	
6.2.5	i8094H_SET_VA	◎	◎	◎	
6.2.6	i8094H_SET_VD	◎	◎	◎	
6.2.7	i8094H_SET_VK	◎	◎	◎	
6.2.8	i8094H_SET_VL	◎	◎	◎	
6.2.9	i8094H_SET_VAO	◎	◎	◎	
6.2.10	i8094H_LINE_2D	◎	◎	◎	
6.2.11	i8094H_LINE_3D	◎	◎	◎	
6.2.12	i8094H_ARC_CW	◎x2	◎x2	◎x2	
Section	Fuction	RTC	MP	ISR	IT
6.2.12	i8094H_ARC_CCW	◎x2	◎x2	◎x2	
6.2.13	i8094H_CIRCLE_CW	◎	◎	◎	
6.2.13	i8094H_CIRCLE_CCW	◎	◎	◎	
Synchronous Actions					
6.3.1	i8094H_SYNC_ACTION	◎x2	◎x2	◎x2	
6.3.2	i8094H_SET_COMPARE	◎	◎	◎	
6.3.3	i8094H_GET_LATCH	◎	◎	◎	
6.3.4	i8094H_SET_PRESET	◎	◎	◎	
6.3.5	i8094H_SET_OUT	◎	◎	◎	
Continuous Interpolation					
6.4.1	i8094H_RECTANGLE	◎x4	◎x4		
6.4.2	i8094H_LINE_2D_INITIAL	◎x2	◎x2		
6.4.2	i8094H_LINE_2D_CONTINUE	◎	◎		
6.4.3	i8094H_LINE_3D_INITIAL	◎x2	◎x2		
6.4.3	i8094H_LINE_3D_CONTINUE	◎	◎		
6.4.4	i8094H_MIX_2D_INITIAL	◎x2	◎x2		
6.4.4	i8094H_MIX_2D_CONTINUE	◎x2	◎x2		
6.4.5	i8094H_CONTINUE_INTP	◎			
6.4.5	i8094H_CONTINUE_INTP_ARRAY	◎			
6.4.6	i8094H_HELIX_3D	◎x3	◎x3		
6.4.7	i8094H_RATIO_INITIAL	◎x2	◎x2		
6.4.7	i8094H_RATIO_2D	◎	◎		
6.4.8	i8094H_LINE_SCAN	◎			
6.4.8	i8094H_LINE_SCAN_START	◎			
6.4.8	i8094H_LINE_SCAN_OFFSET	◎			
Enable/Disable interrupt Function					

◎x2
2-Functions
Line

6.3.6	i8094H_ENABLE_INT	◎	◎		
6.3.6	i8094H_DISABLE_INT	◎	◎		
6.3.6	i8094H_INTFACTOR_ENABLE	◎	◎	◎	
6.3.6	i8094H_INTFACTOR_DISABLE	◎	◎	◎	
6.3.7	i8094H_ENABLE_RINT	◎			
6.3.7	i8094H_DISABLE_RINT	◎			
6.3.7	i8094H_RINT_WAIT	◎			
Other Functions					
6.5.1	i8094H_DRV_HOLD	◎	◎	◎	
6.5.2	i8094H_DRV_START	◎	◎	◎	
6.5.3	i8094H_STOP_WAIT	◎			
Section	Function	RTC	MP	ISR	IT
6.5.4	i8094H_STOP_SLOWLY	◎	◎		
6.5.4	i8094H_STOP_SUDDENLY	◎	◎		
6.5.4	i8094H_VSTOP_SLOWLY	◎	◎		
6.5.4	i8094H_VSTOP_SUDDENLY	◎	◎		
6.5.5	i8094H_CLEAR_STOP	◎	◎		
6.5.6	i8094H_INTP_END	◎	◎		
Additional Functions Supported by i8094H					
7.1	i8094H_LOAD_INITIAL	◎			
7.2.1	i8094H_MP_CREATE	◎			
7.2.1	i8094H_MP_CLOSE		◎		
7.2.2	i8094H_MP_CALL	◎	◎		
7.2.3	i8094H_MP_SET_VAR		◎	◎	
7.2.3	i8094H_MP_SET_RVAR		◎	◎	
7.2.3	i8094H_MP_VAR_CALCULATE		◎	◎	
7.2.4	i8094H_MP_FOR		◎		
7.2.4	i8094H_MP_NEXT		◎		
7.2.5	i8094H_MP_IF		◎	◎	
7.2.5	i8094H_MP_ELSE		◎	◎	
7.2.5	i8094H_MP_IF_END		◎	◎	
7.2.6	i8094H_MP_TIMER		◎		
7.2.7	i8094H_MP_STOP_WAIT		◎		
7.2.8	i8094H_MP_SET_RINT		◎	◎	

2 Basic Settings

2.1 Axes Code Definition

The definitions of axis assignments are as below: X=1, Y=2, Z=4, and U=8. If you assign X and Y axes simultaneously, the code will be 3. In a similar way, AXIS_YZ = 2+4 =0x6; and AXIS_XYZU = 1+2+4+8 = 0xf. You could assign single axis as well as multiple axes at the same time. Available axis codes are listed below.

Table 2-1 Axis assignments and their corresponding codes

Axis	X	Y	Z	U	XY	XZ	XU	YZ
code	0x1	0x2	0x4	0x8	0x3	0x5	0x9	0x6
Variable	AXIS_X	AXIS_Y	AXIS_Z	AXIS_U	AXIS_XY	AXIS_XZ	AXIS_XU	AXIS_YZ
Axis	YU	ZU	XYZ	XYU	XZU	YZU	XYZU	
code	0xa	0xc	0x7	0xb	0xd	0xe	0xf	
Variable	AXIS_YU	AXIS_ZU	AXIS_XYZ	AXIS_XYU	AXIS_XZU	AXIS_YZU	AXIS_XYZU	

Write the setting values into the IT parameter table without making a change of other current settings (please refer to 7.1), the definitions are as follow:

Table(2-1a)

Axis	X	Y	Z	U	XY	XZ	XU	YZ
code	0x11	0x12	0x14	0x18	0x13	0x15	0x19	0x16
Variable	INITIAL_X	INITIAL_Y	INITIAL_Z	INITIAL_U	INITIAL_XY	INITIAL_XZ	INITIAL_XU	INITIAL_YZ
Axis	YU	ZU	XYZ	XYU	XZU	YZU	XYZU	
code	0x1a	0x1c	0x17	0x1b	0x1d	0x1e	0x1f	
Variable	INITIAL_YU	INITIAL_ZU	INITIAL_XYZ	INITIAL_XYU	INITIAL_XZU	INITIAL_YZU	INITIAL_XYZU	

Not apply to Macro Program (MP).

2.2 Modules Registration and getting the LIB version

- **BYTE i8094H_REGISTRATION(BYTE *cardNo*, BYTE *slot*)**

Description:

You are required to register your i8094H before performing any operation. This function enables to register a module by doing the following steps: module registration, assign the slot number the module is installed on, and assing a card number.

Registration must be performed for each I-8094 motion control module before other functions are called. After registration, each module can be identified by its corresponding module number.

Parameters:

	***cardNo*:**	**Module number→ WinCon-8000 : 0~7**
	***slot*:**	**Slot number → WinCon-8000 : 1~7**

Return:

YES	**Normal**
NO	**Abnormal**

Example:

```
//================= for WinCon-8000 ==================
//set each module number the same as the slot number, respectively.
//(slot1 ~ slot7)
BYTE cardNo;
BYTE slot;
int Found = 0;
for (slot = 1; slot < 8; slot++)
{
    cardNo = slot;
    if (i8094H _REGISTRATION(cardNo, slot) == YES)
     {   //slot number begins from 1.
         //if a module is found, the slot number would be registered as the slot number
    of that module.
         i8094H _RESET_CARD(cardNo);
         Found++;
    }
}
if (Found == 0)
{
    //if Wincon cannot find any I-8094H module,
    // please add your code here to take care of the exceptional cases.
    return;
}
//=================== for I-8000 ===================
//set the module number the same as the slot number, respectively.
```

```
//(slot1 ~ slot7)
BYTE cardNo;
BYTE slot;
int Found = 0;
for (slot = 0; slot < 8; slot++)
{
      cardNo = slot + 1;
   //slot number begins from 0, but module number begin from 1.

      if (i8094H_REGISTRATION(cardNo, slot) == YES)
      {
            //if a module is found, then it is registered by giving a number.
            i8094H_RESET_CARD(cardNo);
            Found++;
      }
}
if (Found == 0)
{
      //if Wincon cannot find any I-8094 module,
      // please add your code here to take care of the exceptional cases.
      return;
}
```

- **WORD i8094H_GET_VERSION (void)**

Description:
Read the current versionof i8094H library.

Parameters:
None

Return:
Version code: includes information of the release date (year and the month) (0x0000 ~ 0x9999)

Example:
```
WORD VER_No;
VER_No = i8094H_GET_VERSION();
//Read the version code of i8094Hce.dll
```

Remark:
If the return value is 0x0607
06 : the year is 2006
07: the month is July.

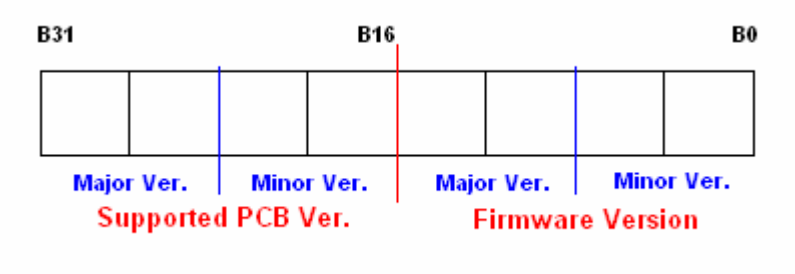
- **DWORD i8094H_GET_FW_VERSION(BYTE *cardNo*)**

Description:
Read current versionof i8094H firmware and compatible PCB version.

Parameters:
***cardNo*: Module number**

Return:
Version code: 0x00000000 ~ 0x99999999

Example:
```
DWORD dwVER_No;
WORD wFirmward_Ver, wPCB_Ver;
dwVER_No = i8094H_GET_FW_VERSION(1);
wPCB_Ver = (WORD)( ( dwVER_No >> 4) & 0xFFFF);
wFirmware_Ver = (WORD)( dwVER_No & 0xFFFF);
//Read Firmware/PCB version codes of the first card.
```



Remark:

The return value of i8094H_GET_FW_VERSION() could be devided into two parts.

For example, if the return value is 0x02210111

It indicates the Firmware Ver. is 1.11

And it supports PCB version 2.21 or above.

2.3 Reset the Module

- **void i8094H_RESET_CARD(BYTE *cardNo*)**

Description:

This function enables to restore the power-on default settings, please refer to Section 7: initial settings after resetting the module.

Parameters:

***cardNo*: Module number**

Return:

None

Example:

```
i8094H_RESET_CARD(1);
//Reset module 1.
```

- **void i8094H_CLEAR_CARD_BUFFER(BYTE *cardNo*)**

Description: Clear all data in i-8094H command buffer.

Parameters:

***cardNo*: Module number**

Return: None

Example: i8094H_ CLEAR_CARD_BUFFER (1);

```
// Clear data buffer in module 1
```

2.4 Pulse Output Mode Setting

- **void i8094H_SET_PULSE_MODE(BYTE *cardNo*, WORD *axis*, BYTE *nMode*)**

Description:

This function sets the pulse output mode to be either CW/CCW or PULSE/DIR for the specific axes and also sets their direction definition.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
	Write the setting values into the parameter table without making a change of current settings(Please refer to Table 2-1a)
***nMode*:**	**Assigned mode (Please refer to Table 2-2)**

Return:

None

Example:

```
i8094H _SET_PULSE_MODE(1, AXIS_XYZ, 2);
//set the pulse mode of X, Y, and Z axes to be mode 2 in module 1
i8094H_SET_PULSE_MODE(1, AXIS_U, 3);
//set the pulse mode of U axis to be mode 3 in module 1
i8094H_SET_PULSE_MODE(1, INITIAL_XYZU, 0);
//set the pulse mode of X Y Z U axes to be mode 0, write into the parameter table (Table 2-2) in module 1
```

Table 2-2 A List of pulse output modes

	mode	Pulse output signals	
		nPP	nPM
CW / CCW	0	CW (rising edge)	CCW (rising edge)
	1	CW (falling edge)	CCW (falling edge)
PULSE / DIR	2	PULSE (rising edge)	DIR (LOW:+dir/ HIGH:-dir)
	3	PULSE (falling edge)	DIR (LOW:+dir/ HIGH:-dir)
	4	PULSE (rising edge)	DIR (HIGH:+dir/ LOW:-dir)
	5	PULSE (falling edge)	DIR (HIGH:+dir/ LOW:-dir)

2.5 Setting the Maximum Speed

- ***void i8094H_SET_MAX_V(BYTE *cardNo*, BYTE *axis*, DWORD *data*)**

Description:

This function sets the maximum rate for the output pulses (speed). A larger value results in a rougher resolution, and vice versa(8000 speed segments). For example, if the maximum speed is set as 8000 PPS, the resolution will be 1 PPS;if the maximum speed is set as 16000 PPS, the resolution will be 2 PPS; if the maximum speed is set as 80000 PPS, the resolution will be 10 PPS, etc. Maximum value 4,000,000 PPS means the resolution of speed will be 500 PPS. This function will change the resolution of speed to reach the desired maximum speed. Since the scale in hardware is changed, other parameters will be updated accordingly too; such as the starting speed, the acceleration, and the jerk. It is recommended to set the maximum speed value to be an integral multiplier of 8000.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
	Write the setting values into the parameter table without making a change of current settings(Please refer to Table 2-1a)
***data*:**	**maximum speed, single axis (8,000~4,000,000 PPS)** **interpolation motion maximum speed, the second axis(8,000~2,828,854 PPS)** **nterpolation motion maximum speed, the third axis (8,000~2,309,468 PPS)**

Return:

None

Example:

```
i8094H_SET_MAX_V(1, AXIS_XY, 200000L);
//The maximum speed for the X and Y axes of module 1 is 200KPPS.
//The resolution of the speed will be 200000/8000 = 25 PPS.
```

2.6 Setting the Active Level of the Hardware Limit Switches

- ***void i8094H_SET_HLMT(BYTE *cardNo*, BYTE *axis*, BYTE *nFLEdge*, BYTE *nRLEdge*)**

Description:

This function sets the active logic level of the hardware limit switch inputs.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
	Write the setting values into the parameter table without making a change of current settings(Please refer to Table 2-1a).
***nFLEdge*:**	**Active level setting for the forward limit switch.**
	0 = low active; 1 = high active
***nRLEdge*:**	**Active level setting for the reverse limit switch.**
	0 = low active; 1 = high active

Return:

None

Example:

```
i8094H_SET_HLMT(1, AXIS_XYZU, 0, 0);
//set all the trigger levels as low-active for all limit switches
//on module 1.
```

2.7 Setting the Motion Stop Mode When Limit Switch being turn on

- ***void** **i8094H_LIMITSTOP_MODE(BYTE *cardNo*, BYTE *axis*, BYTE *nMode*)**

Description:

This function configures the settings of motion stop mode of the axes when the corresponding limit switches being turn on.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
	Write the setting values into the parameter table without making a change of current settings(Please refer to Table 2-1a).
nMode:	**0: stop immediately; 1: decelerating to stop**

Return:

None

Example:

```
i8094H_LIMITSTOP_MODE(1, AXIS_X, 0);
//set X axis to stop immediately if any limit switch on X axis is turned on.
```

2.8 Setting the Trigger Level of the NHOME Sensor

- ***void i8094H_SET_NHOME(BYTE *cardNo*, BYTE *axis*, BYTE *nNHEdge*)**

Description:

This function enables to set up the trigger level of the near home sensor (NHOME).

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)** **Write the setting values into the parameter table without making a change of current settings(Please refer to Table 2-1a).**
***nNHEdge*:**	**Active level setting for the near home sensor.** **0 = low active; 1 = high active**

Return:

None

Example:

```
i8094H_SET_NHOME(1, AXIS_XY, 0);
//set the trigger level of NHOME of X and Y axes on module 1 to be active low.
```

2.9 Setting Trigger Level of the Home sensor

- ***void i8094H_SET_HOME_EDGE(BYTE *cardNo*, BYTE *axis*, BYTE *nHEdge*)**

Description:

This function sets the trigger level of the home sensor (HOME).

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
	Write the setting values into the parameter table without making a change of current settings(Please refer to Table 2-1a).
***nHEdge*:**	**Active level setting for the home sensor.**
	0 = low active; 1 = high active

Return:

None

Example:

```
i8094H_SET_HOME_EDGE(1, AXIS_XYZU, 1);
//set the trigger level as high active for all home sensors on module 1.
```

2.10 Setting and Clearing the Software Limits

- ***void i8094H_SET_SLMT(BYTE *cardNo*, BYTE *axis*, long *dwFL*, long *dwRL*, BYTE *nType*)**

Description:

This function sets the Positive and Negative software limits.

Parameters:

cardNo: Module number

axis: Axis or axes (Please refer to Table 2-1)
Write the setting values into the parameter table without making a change of current settings(Please refer to Table 2-1a).

dwFL: Value of the forward software limit
(-2,000,000,000 ~ +2,000,000,000)

dwRL: Value of the reverse software limit
(-2,000,000,000 ~ +2,000,000,000)

nType: Position counter to be compared:
0 = logical position counter (LP), i.e., the command position
1 = encoder position counter (EP), i.e., the real position

Return:

None

Example:

```
i8094H_SET_SLMT(1, AXIS_XYZU, 20000, -3000, 0);
//set the forward software limit to be 20000 and the reverse
//software limit to be -3000 for all axes on module 1.
```

- ***void i8094H_CLEAR_SLMT(BYTE *cardNo*, BYTE *axis*)**

Description:

This function clears the software limits.

Parameters:

cardNo: Module number

axis: Axis or axes (Please refer to Table 2-1)

Return:

None

Example:

```
i8094H_CLEAR_SLMT(1, AXIS_XYZU);
//clear the software limits for all axes on module 1.
```

2.11 Setting the Encoder Related Parameters

- ***void** **i8094H_SET_ENCODER(BYTE *cardNo*, BYTE *axis*, BYTE *nMode*, BYTE *nDivision*, BYTE *nZEdge*)**

Description:

This function sets the relevant parameters for encoder input.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1) Write the setting values into the parameter table without making a change of current settings(Please refer to Table 2-1a).**
***nMode*:**	**Encoder input type: 0 = A quad B; 1 = up/down**
***nDivision*:**	**Division setting for A quad B input signals: 0 = 1/1 1 = 1/2 2 = 1/4**
***nZEdge*:**	**Sets the trigger level for the Z phase 0 = low active; 1 = high active**

Return:

None

Example:

```
i8094H_SET_ENCODER(1, AXIS_XYZU, 0, 0, 0);
//set the encoder input type as A quad B; the division is module 1;
//and the Z phase is low active.
```

2.12 Setting the Servo Driver (ON/OFF)

- ***void** **i8094H_SERVO_ON(BYTE *cardNo*, BYTE *axis*)**

Description:

This function outputs a DO signal (ENABLE) to enable the motor driver.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1) Write the setting values into the parameter table without making a change of current settings(Please refer to Table 2-1a).**

Return:

None

Example:

```
i8094H_SERVO_ON(1, AXIS_XYZU);
//enables all drivers on module 1.
```

- ***void i8094H_SERVO_OFF(BYTE *cardNo*, BYTE *axis*)**

Description:

This function outputs a DO signal (ENABLE) to disable the motor driver.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
	Write the setting values into the parameter table without making a change of current settings(Please refer to Table 2-1a).

Return:

None

Example:

```
i8094H_SERVO_OFF(1, AXIS_XYZU);
//disables all drivers on module 1.
```

2.13 Setting the SERVO ALARM Function

- ***void** **i8094H_SET_ALARM(BYTE *cardNo*, BYTE *axis*, BYTE *nMode*, BYTE *nAEdge*)**

Description:

This function sets the ALARM input signal related parameters.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1) Write the setting values into the parameter table without making a change of current settings(Please refer to Table 2-1a).**
***nMode*:**	**Mode: 0 = disable ALARM function; 1 = enable ALARM function**
***nAEdge*:**	**Sets the trigger level 0 = low active; 1 = high active**

Return:

None

Example:

```
i8094MF_SET_ALARM(1, AXIS_ZU, 1, 0);
//enable the ALARM for the Z and U axes on module 1 and set them
//as low-active.
```

2.14 Setting the Active Level of the In-Position Signals

- ***void i8094H_SET_INPOS(BYTE *cardNo*, BYTE *axis*, BYTE *nMode*,**
  BYTE *nIEdge*)

Description:

This function sets the INPOS input signal related parameters.

Note: Sometimes, this signal is used to connect the SERVO READY input signal. Users should take care of what signal the daughter board is wired.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
	Write the setting values into the parameter table without making a change of current settings(Please refer to Table 2-1a).
***nMode*:**	**Mode: 0 = disable INPOS input;**
	1 = enable INPOS input
***nIEdge*:**	**Set the trigger level**
	0 = low active; 1 = high active

Return:

None

Example:

```
i8094H_SET_INPOS(1, AXIS_X, 1, 0);
//enable the INPOS function of the X axis on module 1 and set it to be low-active.
```

Note: Please refer to the example shown in Fig. 2.12 for wiring of the general DI input.

2.15 Setting the Time Constant of the Digital Filter

- ***void i8094H_SET_FILTER(BYTE *cardNo*, BYTE *axis*, BYTE *FEn*, BYTE *FLn*)**

Description:

This function selects the axes and sets the time constant for digital filters of the input signals.

Parameters:

***cardNo*: Module number**

***axis*: Axis or axes (Please refer to Table 2-1)**
Write the setting values into the parameter table without making a change of current settings(Please refer to Table 2-1a).

***FEn*: Enabled filters. The sum of the code numbers (0~31) are used to select input signals. Please refer to the following table.**

Code number	Enabling filters
1	EMG, nLMTP, nLMTM, nIN0, nIN1
2	nIN2
4	nINPOS, nALARM
8	nEXPP, nEXPM, EXPLSN
16	nIN3

***FLn*: Sets the filter time constant (0~7) as follows.**

Code	Removable max. noise width	Input signal delay time
0	1.75 μ SEC	2 μ SEC
1	224 μ SEC	256 μ SEC
2	448 μ SEC	512 μ SEC
3	896 μ SEC	1.024mSEC
4	1.792mSEC	2.048mSEC
5	3.584mSEC	4.096mSEC
6	7.168mSEC	8.192mSEC
7	14.336mSEC	16.384mSEC

Return:

None

Example:

```
i8094H_SET_FILTER(1, AXIS_XYZU, 21, 3);
//set the filter time constants of X, Y, Z, and U axes as 1.024mSEC.
//These filters include EMG, nLMTP, nLMTM, nIN0, nIN1, nINPOS, nALARM,
//and nIN3.
//(21 = 1+4+16)     1: EMG + nLMP + nLMPM + nIN0 + nIN1;
//                  4: nINPOS + nALARM;
//                  16: nIN3.
```

Note: The default wiring design is: nIN0 is connected to the NEAR HOME (NHOME) sensors; nIN1 is connected to the HOME sensors; and nIN2 is connected to the index of Encoder input (Z phase).

2.16 Position Counter Variable Ring

- ***void i8094H_VRING_ENABLE(BYTE *cardNo*, BYTE *axis*, DWORD *nVRing*)**

Description:

This function enables the linear counter of the assigned axes as variable ring counters.

Parameters:

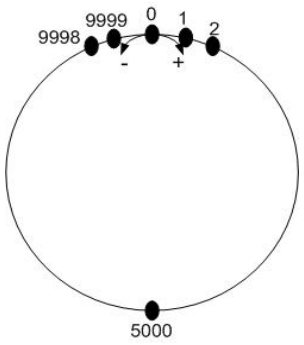
***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)** **Write the setting values into the parameter table without making a change of current settings(Please refer to Table 2-1a).**
***nVRing*:**	**Maximum value of the ring counter** **(0 ~ +2,000,000,000)**

Return:

None

Example:

```
i8094H_ VRING_ENABLE(1, AXIS_X, 9999);
//set the X axis of module 1 to be a ring counter. The encoder
//values will be 0 to 9999.
```



The encoder value ranges from 0 to 9999. When the counter value reaches 9999, one more adding pulse will cause the counter to reset to 0. When the counter value is 0, a lessening pulse will let the counter set to 9999.

Max. ring encoder value = 9999

Note: 1. This function will set the LP and EP simultaneously.
2. If this function is enabled, the software limit function cannot be used.

- ***void i8094H_VRING_DISABLE(BYTE *cardNo*, BYTE *axis*)**

Description:

This function disables the variable ring counter function.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
	Write the setting values into the parameter table without making a change of current settings (Please refer to Table 2-1a).

Return:

None

Example:

```
i8094H_ VRING_DISABLE(1, AXIS_X);
//disable the ring counter function for the X axis
//on module 1.
```

2.17 Triangle prevention of fixed pulse driving

- ***void** **i8094H_AVTRI_ENABLE(BYTE *cardNo*, BYTE *axis*)**

Description:

This function prevents a triangle form in linear acceleration (T-curve) fixed pulse driving even if the number of output pulses is low.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
	Write the setting values into the parameter table without making a change of current settings(Please refer to Table 2-1a).

Return:

None

Example:

```
i8094H_ AVTRI_ENABLE(1, AXIS_X);
//set the X axis of module 1 not to generate a triangle form in its speed profile.
```

- ***void** **i8094H_AVTRI_DISABLE(BYTE *cardNo*, BYTE *axis*)**

Description:

This function disables the function to prevent a triangle form in linear acceleration fixed pulse driving.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
	Write the setting values into the parameter table without making a change of current settings(Please refer to Table 2-1a).

Return:

None

Example:

```
i8094H_ AVTRI_DISABLE(1, AXIS_X);
//enable the X axis of module 1 to generate a triangle form in its
//speed profile if the pulse number for output is too low.
```

2.18 External Pulse Input

Cannot write settings for external input driving into the parameter Table.

2.18.1 Handwheel (Manual Pulsar) Driving

- **void i8094H_EXD_MP(BYTE *cardNo*, BYTE *axis*, DWORD *data*)**

Description:

This function outputs pulses according to the input pulses from a handwheel.

Parameters:

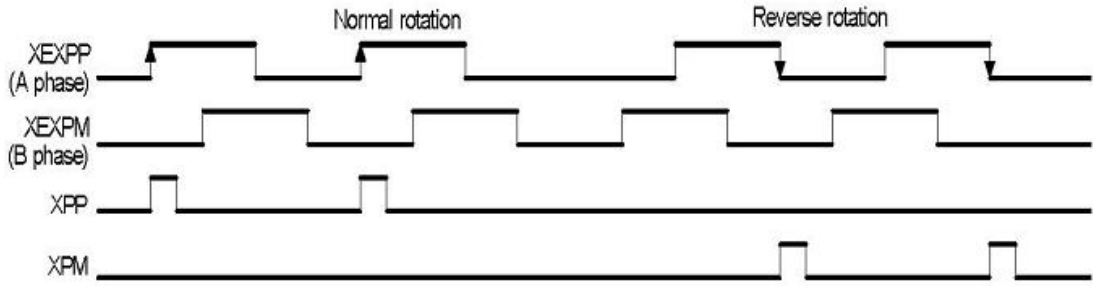
***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1.)**
	The axis can be either X, Y, Z, or U.
***data*:**	**Number of command pulses (Rang: 0 ~ +2,000,000,000).**

Return:

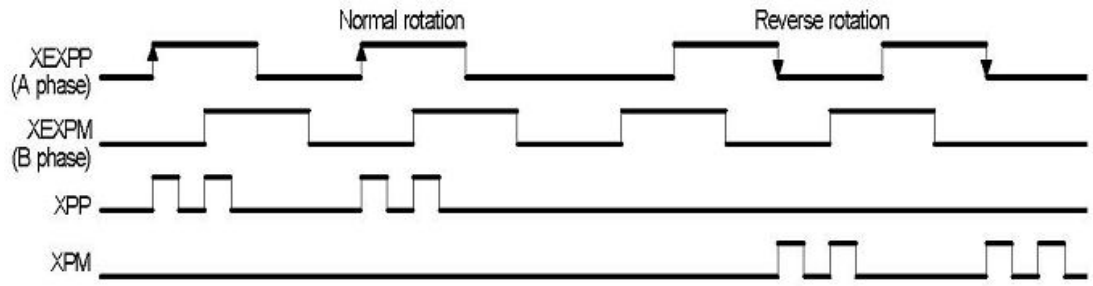
None

Example:

```
i8094H_EXD_MP(1, AXIS_X, 1);
//Each time the handwheel inputs a pulse to the X axis
//on module 1, the controller will output 1 pulse to the motor driver.
```




```
i8094H_EXD_MP(1, AXIS_X, 2);
//Each time the handwheel inputs a pulse to the X axis
//on module 1, the controller will output 2 pulses to the motor driver.
```



2.18.2 Fixed Pulse Driving Mode

- **void i8094H_EXD_FP(BYTE *cardNo*, BYTE *axis*, DWORD *data*)**

Description:

This function outputs fixed pulses according to the trigger input (the falling edge of the nEXP+ signal) from a handwheel.

Parameters:

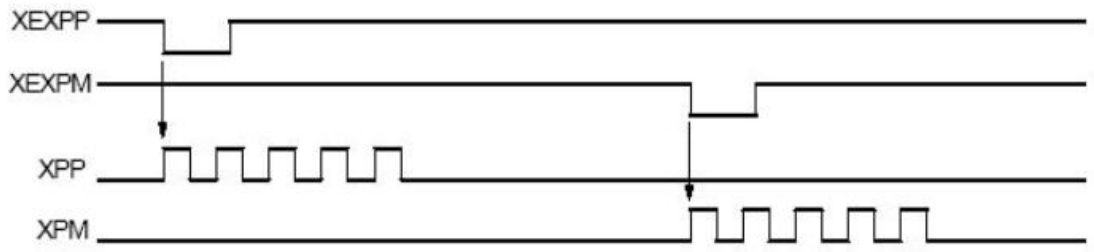
***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1.)**
	The axis can be either X, Y, Z, or U(1 or 2 or 4 or 8).
***data*:**	**Number of fixed pulses(0 ~ +2,000,000,000).**

Return:

None

Example:

```
i8094H_EXD_FP(1, AXIS_X, 5);
//Each time the controller detects a falling edge of an XEXP+
//signal, it will output 5 pulses to the X axis.
```



Example of fixed pulse driving using an external signal

2.18.3 Continuous Pulse Driving Mode

- **void i8094H_EXD_CP(BYTE *cardNo*, BYTE *axis*, DWORD *data*)**

Description:

The controller will continuously output pulses in positive direction if the falling edge of nEXP+ signal from a handwheel is detected. On the contrary, it will continuously output pulses in negative direction if the falling edge of nEXP- signal is detected.

Parameters:

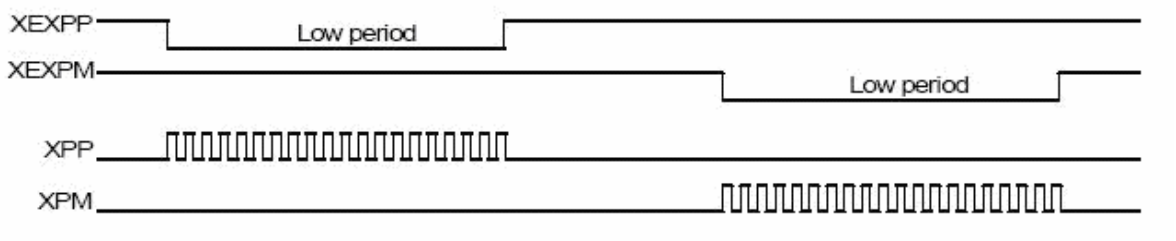
***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1.)**
	The axis can be either X, Y, Z, or U(1 or 2 or 4 or 8).
***data*:**	**Pulse output speed in PPS(0 ~ +2,000,000,000)**

Return:

None

Example:

```
i8094H_EXD_CP(1, AXIS_X, 20);
//Each time the controller detects a falling edge of an XEXP+
//signal, it will continuously drive X axis at the speed of 20 PPS.
```



Continuous driving using an external signal

2.18.4 Disabling the External Signal Input Functions

- **void i8094H_EXD_DISABLE(BYTE *cardNo*, BYTE *axis*)**

Description:

This function turns off the external input driving control functions.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1.)**
	The axis can be either X, Y, Z, or U(1 or 2 or 4 or 8)

Return:

None

Example:

```
i8094H_EXD_DISABLE(1, AXIS_X);
//disable the external input driving control function
//of X axis on module 1
```

2.19 Read/Write User defined Variables (VAR)

- **BYTE i8094H_READ_bVAR(BYTE *cardNo*, BYTE *bvarNo*)**

Description:

This function read variable bVARn, it could be passed to Macro Program(MP), for detail information, please refer to section 7.

Parameters:

***cardNo*:**	**Module number**
b*varNo*:	**custom variable: bVAR0 ~ bVAR127**

Return:

Current value of the variable (0 ~ +255)

Example:

```
BYTE bdata;
bdata = i8094H_READ_bVAR(1, bVAR100);
//read value of VAR100 in module 1
```

- **void i8094H_WRITE_bVAR(BYTE *cardNo*, BYTE *bvarNo*, BYTE *bVar*)**

Description:

This function write variable bVARn, it could be passed to Macro Program(MP), for detail information, please refer to section 7.

Parameters:

***cardNo*:**	**Module number**
b*varNo*:	**custom variable: bVAR0 ~ bVAR127**
***bVar*:**	**variable (0 ~ +255)**

Return:

None

Example:

```
i8094H_WRITE_bVAR(1, bVAR100, 100);
//write bVAR100=100 in module 1
```

- **long i8094H_READ_VAR(BYTE *cardNo*, long *varNo*)**

Description:

This function read variable VARn, it could be passed to Macro Program(MP), for detail information, please refer to section 7.

Parameters:

***cardNo*:**	**Module number**
***varNo*:**	**custom variable: VAR0 ~ VAR511**

Return:

Current value of the variable(-2,000,000,000 ~ +2,000,000,000)

Example:

```
long ldata;
ldata = i8094H_READ_VAR(1, VAR100);
//read value of VAR100 in module 1
```

- **void i8094H_WRITE_VAR(BYTE *cardNo*, long *varNo*, long *lVar*)**

Description:

This function write variable VARn, it could be passed to Macro Program(MP), for detail information, please refer to section 7.

Parameters:

***cardNo*:**	**Module number**
b*varNo*:	**Custom variable: VAR0 ~ VAR511**
***lVar*:**	**Value of variable(-2,000,000,000 ~ +2,000,000,000)**

Return:

None

Example:

```
i8094H_WRITE_VAR(1, VAR100, 10000);
//write VAR100=10,000 in module 1
```

2.20Read/Write Data for Power outage carry-over (MD)

- **void i8094H_READ_MD(BYTE *cardNo*, long *mdNo*, long* *ldata*, float* *fdata*)**

Description:

Read machine data.

Parameters:

***cardNo*:**	**Module number**
***mdNo*:**	**Machine data(long): MD0 ~ MD1023** **Machine data (float): MD1024 ~ MD2047**
&*ldata*:	**Read MD long (-2,147,483,648 ~ +2,147,483,647)**
&*fdata*:	**Read MD float (Integer number plus decimal number giving a total of six places)**

Return:

None

Example:

```
long ldata;
float fdata;
i8094H_WRITE_MD(CardNo, MD100, -99999, 0);
i8094H_READ_MD(CardNo, MD100, &ldata, 0);
//read MD100 long data in module 1
i8094H_WRITE_MD(CardNo, MD1500, 0, -990.999);
i8094H_READ_MD(CardNo, MD1500, 0, &fdata);
//read MD1500 float data in module 1
```

- **void i8094H_WRITE_MD(BYTE *cardNo*, long *mdNo*, long *ldata*, float *fdata*)**

Description:

Write machine data.

Parameters:

***cardNo*:**	**Module number**
***mdNo*:**	**Machine data(long): MD0 ~ MD1023** **Machine data (float): MD1024 ~ MD2047**
***ldata*:**	**Write MD long (-2,147,483,648 ~ +2,147,483,647)**
***fdata*:**	**Write MD float (Integer number plus decimal number giving a total of six places)**

Return:

None

Example:

Same as above

3 Reading and Registers Setting

3.1 Setting and Reading the Command Position

- ****void i8094H_SET_LP(BYTE *cardNo*, BYTE *axis*, long *wdata*)**

Description:

This function sets the command position counter value (logical position counter, LP).

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
***wdata*:**	**Position command (-2,000,000,000 ~ +2,000,000,000)**

Return:

None

Example:

```
i8094H_SET_LP(1, AXIS_XYZU, 0);
//Set the LP as 0 for X, Y, Z, and U axes in module 1
// will clear all LP counters on module 1
```

- ****long i8094H_GET_LP(BYTE *cardNo*, BYTE *axis*)**

Description:

This function reads the command position counter value (logical position counter, LP).

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis (Please refer to Table 2-1) The axis can be either X, Y, Z, or U(1 or 2 or 4 or 8).**

Return:

Current LP value (-2,000,000,000 ~ +2,000,000,000)

Example:

```
long X_LP;
X_LP = i8094H_GET_LP(1, AXIS_X);
//Reads the LP value of the X axis on module 1.
```

3.2 Setting and Reading the Encoder Counter

- ****void i8094H_SET_EP(BYTE *cardNo*, BYTE *axis*, long *wdata*)**

Description:

This function sets the encoder position counter value (real position counter, or EP).

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
***wdata*:**	**Position command (-2,000,000,000 ~ +2,000,000,000)**

Return:

None

Example:

```
i8094H_SET_EP(1, AXIS_XYZU, 0);
//Set the EP as 0 for X, Y, Z, and U axes of module 1.
//This command clears all EP counters on module 1.
```

- ****long i8094H_GET_EP(BYTE *cardNo*, BYTE *axis*)**

Description:

This function reads the encoder position counter value (EP).

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis (Please refer to Table 2-1) The axis can be either X, Y, Z, or U(1 or 2 or 4 or 8).**

Return:

Current EP value (-2,000,000,000 ~ +2,000,000,000)

Example:

```
long X_EP;
X_EP = i8094H_GET_EP(1, AXIS_X);
//reads the encoder position value (EP) of the X axis on module 1.
```

3.3 Reading the Current Velocity

- **DWORD i8094H_GET_CV(BYTE *cardNo*, BYTE *axis*)**

Description:

This function reads the current velocity value.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis (Please refer to Table 2-1)**
	The axis can be either X, Y, Z, or U (1 or 2 or 4 or 8).

Return:

Current speed (in PPS)

Example:

```
DWORD dwdata;
dwdata = i8094H_GET_CV(1, AXIS_X);
//reads the current velocity of the X axis on module 1.
```

3.4 Reading the Current Acceleration

- **DWORD i8094H_GET_CA(BYTE *cardNo*, BYTE *axis*)**

Description:

This function reads the current acceleration value PPS/Sec.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis (Please refer to Table 2-1)**
	The axis can be either X, Y, Z, or U(1 or 2 or 4 or 8).

Return:

Current acceleration (in PPS/Sec)

Example:

```
DWORD dwdata;
dwdata = i8094H_GET_CA(1, AXIS_X);
//reads the current acceleration value of the X axis on module 1.
```

3.5 Reading the DI Status

- ****BYTE i8094H_GET_DI(BYTE *cardNo*, BYTE *axis*, BYTE *nType*)**

Description:

This function reads the digital input (DI) status.

Parameters:

***cardNo*: Module number**

***axis*: Axis (Please refer to Table 2-1)**
The axis can be either X, Y, Z, or U.

***nType*:**

0 → DRIVING	**(Check whether the axis is driving or not.)**
1 → LIMIT+	**(Check whether the limit+ is engaged or not.)**
2 → LIMIT-	**(Check whether the limit- is engaged or not.)**
3 → EMERGENCY	**(Check whether EMG signal is on or not.)**
4 → ALARM	**(Check the ALARM input signal.)**
5 → HOME	**(Check the HOME input signal)**
6 → NHOME	**(Check the Near HOME input signal)**
7 → IN3	**(Check the IN3 input signal)**
8 → INPOS	**(Check the INPOS input signal)**
9 → INDEX	**(Check the encoder Z-phase input signal)**

Return:

YES: on
NO: off

Example:

```
if (i8094H_GET_DI(1, AXIS_X, 1) == YES)
{
    //get the status of limit+ sensor of X axis on module 1
}
```

● **WORD i8094H_GET_DI_ALL(BYTE *cardNo*, BYTE *axis*)

Description:

This function reads the digital input (DI) status.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis (Please refer to Table 2-1)**
	The axis can be either X, Y, Z, or U(1 or 2 or 4 or 8).

Return:

0x001 → DRIVING (Check whether the axis is driving or not.)
0x002 → LIMIT+ (Check whether the limit+ is engaged or not.)
0x004 → LIMIT- (Check whether the limit- is engaged or not.)
0x008 → EMERGENCY (Check whether EMG signal is on or not.)
0x010 → ALARM (Check the ALARM input signal.)
0x020 → HOME (Check the HOME input signal)
0x040 → NEAR HOME (Check the Near HOME input signal)
0x080 → IN3 (Check the IN3 input signal)
0x100 → INPOS (Check the INPOS input signal)
0x200 → Z-Phase (Check the encoder Z-phase input signal)

Example:

```
WORD wStatus;
i8094H_GET_DI_ALL(1, AXIS_X & wStatus);
if ( (wStatus & 0x002) == 0x002 )
{
    //get the status of limit+ sensor of X axis on module 1
}
```

3.6 Reading and Clearing the ERROR Status

- ****BYTE i8094H_GET_ERROR(BYTE *cardNo*)**

Description:

This function checks whether an error occurs or not.

Parameters:

***cardNo*:**	**Module number**

Return:

YES:	**Error(s) occured. Please perform i8094H_GET_ERROR_CODE () to get more information. If *GET_ERROR_CODE =256*, it indicates that the motion stop was due to the "STOP" command, not because of an error happened. Please refer to 6.5.5 and the following example for clearing ERROR.**
NO:	**No error.**

EXAMPLE:

```
If (i8094H_GET_ERROR(1) == YES)
{
    //read module 1 and ERROR is found
      WORD ErrorCode_X = i8094H_GET_ERROR_CODE(1, AXIS_X);
      WORD ErrorCode_Y = i8094H_GET_ERROR_CODE(1, AXIS_Y);
      WORD ErrorCode_Z = i8094H_GET_ERROR_CODE(1, AXIS_Z);
      WORD ErrorCode_U = i8094H_GET_ERROR_CODE(1, AXIS_U);
      if ((ErrorCode_X || ErrorCode_Y || ErrorCode_Z || ErrorCode_U) == 256)
      {
           // It means that motion was stopped due to the stop command was
          issued, not because any error happened. Please take actions to clear
          the malfunction; then clear the STOP status.
           i8094H_CLEAR_STOP(1);
      }

}
```

● **WORD i8094H_GET_ERROR_CODE(BYTE *cardNo*, BYTE *axis*)

Description:

This function reads the ERROR status.

Parameters:

***cardNo*:** **Module number**

***axis*:** **Axis (Please refer to Table 2-1)**
The axis can be either X, Y, Z, or U (1 or 2 or 4 or 8).

Return:

0 → no error
For non-zero return values, please refer to the following table. If there are more than one error, the return value will be the sum of these error code values.

Error Code	Cause of stop	Explanation
1	SOFT LIMIT+	Occurs when the forward software limit is asserted
2	SOFT LIMIT-	Occurs when the reverse software limit is asserted
4	LIMIT+	Occurs when the forward hardware limit is asserted
8	LIMIT-	Occurs when the reverse hardware limit is asserted
16	ALARM	Occurs when the ALARM is asserted
32	EMERGENCY	Occurs when the EMG is asserted
64	Reserved	Reserved
128	HOME	Occurs when both Z phase and HOME are asserted
256	refer to 6.5.4	Occurs when the EMG(software) is asserted

For example, a return code 48 means that ALARM and EMGERENCY occur at the same time.

3.7 Read RTC status

- **BYTE i8094H_CHECK_RTC(BYTE *cardNo*)**

Description:

This function reads current RTC status.

Parameters:

	***cardNo*:**	**Module number**

Return:

YES:	**The buffer is full**
NO:	**The buffer is not full**

EXAMPLE:

```
if (i8094H_CHECK_RTC(1) == YES)
        {
            //buffer is full
            // please add your code here to take care of the exceptional cases.
        }
```

4 FRnet Functions

4.1 Read FRnet DI Signals

- **WORD i8094H_FRNET_IN(BYTE *cardNo*, BYTE *wSA*)

Description:

This function reads the FRnet digital input signals. One group comprises 16 bits data. Therefore, total 128 DI can be defined for one FRnet interface.

Parameters:

***cardNo*:**	**Module number**
***wSA*:**	**Group number, SA8~SA15**

Return:

WORD	**16-bit DI data. (0x0000 ~ 0xffff)**

Example:

```
WORD IN_Data;
IN_Data = i8094H_FRNET_IN(1, 8);
//Read the 16-bit DI which is on module 1 and the group number is 8.
```

4.2 Write data to FRnet DO

- **void i8094H_FRNET_OUT(BYTE *cardNo,* BYTE *wRA,* DWORD *data*)

Description:

This function write data to the FRnet digital output. One group comprises 16 bits data. Therefore, total 128 DO can be defined for one FRnet interface.

Parameters:

***cardNo*:**	**Module number**
***wRA*:**	**Group number, range RA0~RA7**
***data*:**	**0x00000000 ~ 0x0000ffff**

Return:

None

Example:

```
i8094H_FRNET_OUT(1, 0, 0x0000ffff);
// on module 1, set group number RA=0, output data is 0x0000ffff
```

5 Auto Homing Search

I-8094H module provides automatic home search function; after the configuration of proper settings, it would function automatically. The main steps are as bellows:

- Near-home sensor searching(NHOME) under high-speed motion.
- Home sensor searching under low-speed motion.
- Servomotor Z-phase searching under low-speed motion.
- Offset movement to the origin of the working area under high-speed motion.

A few steps could be skipped to adjust settings accordingly to meet customers' actual needs. This operation could be performed automatically, therefore economize on CPU resource and reduce programming efforts.

5.1 Set Up Homing Speed

- ***void i8094H_SET_HV(BYTE *cardNo*, BYTE *axis*, DWORD *data*)**

Description:

This function sets the homing speed.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
***data*:**	**Homing speed (Vmin~Vmax PPS)**

Return:

None

EXAMPLE:

```
i8094H_ SET_HV(1, AXIS_X, 500);
//set the homing speed of the X axis on module 1 to be 500 PPS.
```

5.2 Using an Limit Switch as the HOME sensor

- ***void** **i8094H_HOME _LIMIT(BYTE *cardNo*, BYTE *axis*, BYTE *nType*)**

Description:

This function sets the Limit Switch to be used as the HOME sensor.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis axes (Please refer to Table 2-1)**
***nType*:**	**0: The LIMIT SWITCH is as the HOME sensor;**
	1: Use the LIMIT SWITCH as the HOME sensor

Return:

None

EXAMPLE:

```
i8094H_ HOME_LIMIT(1, AXIS_X, 0);
//Do not use the Limit Switch as the HOME sensor.
```

5.3 Setting the Homing Mode

- ***void** **i8094H_SET_HOME_MODE(BYTE *cardNo*, BYTE *axis*, BYTE *nStep1*,**
  BYTE *nStep2*, BYTE *nStep3*, BYTE *nStep4* , DWORD *data*)

Description:

This function sets the homing method and other related parameters.

Parameters:

***cardNo*:** **Module number**

***axis*:** **Axis or axes (Please refer to Table 2-1)**

***nStep1*:**

0: Step 1 will not be executed
1: Moves in the positive direction
2: Moves in the negative direction

***nStep2*:**

0: Step 2 will not be executed
1: Moves in a positive direction
2: Moves in a negative direction

***nStep3*:**

0: Step 3 will not be executed
1: Moves in the positive direction

2: Moves in the negative direction

***nStep4*:**

0: Step 4 will not be executed
1: Moves in the positive direction
2: Moves in the negative direction

***data*:** **Offset value (0 ~ +2,000,000,000)**

The Four Steps Required for Automatic Homing

Step	Action	Speed	Sensor
1	Searching for the Near Home sensor	V	NHOME (IN0)
2	Searching for the HOME sensor	HV	HOME (IN1)
3	Searching for the encoder Z-phase signal	HV	Z-phase (IN2)
4	Moves to the specified position	V	

Return:

None

Example:

```
//Use the following functions to set the homing mode of the X axis.
i8094H_SET_V(1, 0x1, 20000);
i8094H _SET_HV(1, 0x1, 500);
i8094H _SET_HOME_MODE(1, 0x1, 2, 2, 1, 1, 3500);
i8094H _HOME_START(1, 0x1); //start auto-homing.
i8094H _WAIT_HOME(1, 0x1); //wait until homing is completed.
```

Step	Input Signal	Direction	Speed
1	Near HOME (IN0) is active	-	20000 PPS (V)
2	HOME (IN1) is active	-	500 PPS (HV)
3	Z-phase (IN2) is active	+	500 PPS (HV)
4	No sensor is required. Move 3500 pulses along the X axis.	+	20000 PPS (V)

5.4 Starting the Homing Sequence

- ***void i8094H_HOME _START(BYTE *cardNo*, BYTE *axis*)**

Description:

This function starts the home search of assigned axes.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**

Return:

None

Example:

```
i8094H_ HOME_START(1, AXIS_X);
//start the automatic homing sequence for the X axis on module 1.
```

5.5 Waiting for the Homing Sequence to be Completed

- **Please refer to section 6.5.3or 7.2.7. same as STOP_WAIT function**

6 Basic Motion Control

6.1 Independent Motion Control for each axis

- **Multiple axes could move at the same time.**
- **The motion of each axis can be started independently.**
- **Each axis is moving independently.**
- **Each axis can receive commandeds to change motion, such as changing the number of pulses or the speed.**
- **Each axis can receive commandeds to stop slowly or suddenly to meet the specific requirements.**
- **Independent axis motion can work with interpolation or synchrous action to perform more complicated and versatile motion.**

6.1.1 Setting the Acceleration/Deceleration Mode

- ****void i8094H_NORMAL_SPEED(BYTE *cardNo*, BYTE *axis* , BYTE *nMode*)**

Description:

The function sets the speed mode.

Parameters:

***cardNo*: Module number**

***axis*: Axis (Please refer to Table 2-1)**

***nMode*:**

0 → Symmetric T-curve (Please set SV, V, A, and AO)

1 → Symmetric S-curve (Please set SV, V, K, and AO)

2 → Asymmetric T-curve (Please set SV, V, A, D, and AO)

3 → Asymmetric S-curve (Please set SV, V, K, L, and AO)

Return:

None

Example:

```
BYTE cardNo=1; //select module 1.
i8094H_SET_MAX_V(cardNo, AXIS_XYZU, 20000);
//set the max speed of XYZU axes to be 20K PPS.
//=====================================================
i8094H_NORMAL_SPEED(cardNo, AXIS_XYZU, 0);
//use a symmetric T-curve for all axes on module 1.
i8094H _SET_V(cardNo, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to be 2000 PPS.
i8094H_SET_A(cardNo, AXIS_XYZU,1000);
//set the acceleration of all axes on module 1 to be 1000 PPS/Sec.
i8094H_SET_SV(cardNo, AXIS_XYZU, 2000);
```

```
//set the start speed of all axes on module 1 to be 2000 PPS.
i8094H_SET_AO(cardNo, AXIS_XYZU, 9);
//set the number of remaining offset pulses for all axes to be 9 pulses.
i8094H_FIXED_MOVE(cardNo, AXIS_XYZU, 10000);
//move all axes on module 1 to be 10000 pulses.
//=====================================================
i8094H_NORMAL_SPEED(cardNo, AXIS_XYZU,1);
//use a symmetric S-curve for all axes on module 1.
i8094H_SET_V(cardNo, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to be 2000 PPS.
i8094H_SET_K(cardNo, AXIS_XYZU, 500);
//set the acceleration rate of all axes on module 1 to be 500 PPS/Sec^2.
i8094H_SET_SV(cardNo, AXIS_XYZU, 200);
//set the start speed of all axes on module 1 to be 200 PPS.
i8094H_SET_AO(cardNo, AXIS_XYZU, 9);
//set the number of remaining offset pulses to be 9 pulses for all axes.
i8094H_FIXED_MOVE(cardNo, AXIS_XYZU, -10000);
//move all axes on module 1 to be 10000 pulses in reverse direction.
//=====================================================
i8094H_NORMAL_SPEED(cardNo, AXIS_XYZU,2);
//use an asymmetric T-curve for all axes on module 1.
i8094H_SET_V(cardNo, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to be 2000 PPS.
i8094H_SET_A(cardNo, AXIS_XYZU,1000 );
//set the acceleration of all axes on module 1 to be 1000 PPS/Sec.
i8094H_SET_D(cardNo, AXIS_XYZU, 500);
//set the deceleration of all axes on module 1 to be 500 PPS.
i8094H_SET_SV(cardNo, AXIS_XYZU, 200);
//set the start speed of all axes on module 1 to 200 PPS.
i8094H_SET_AO(cardNo, AXIS_XYZU, 9);
//set the number of remaining offset pulses to be 9 pulses for all axes.
i8094H_FIXED_MOVE(cardNo, axis, 10000);
//move all axes on module 1 to be 10000 pulses.
//=====================================================
i8094H_NORMAL_SPEED(cardNo, AXIS_XYZU,3);
//use an asymmetric S-curve for all axes on module 1.
i8094H_SET_V(cardNo, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to be 2000 PPS.
i8094H_SET_K(cardNo, AXIS_XYZU, 500);
//set the acceleration rate of all axes on module 1 to be 500 PPS/Sec^2.
i8094H_SET_L(cardNo, AXIS_XYZU, 300);
//set the deceleration rate of all axes on module 1 to be 300 PPS/Sec^2.
i8094H_SET_SV(cardNo, AXIS_XYZU, 200);
//set the start speed of all axes on module 1 to be 200 PPS.
i8094H_SET_AO(cardNo, AXIS_XYZU, 9);
//set the number of remaining offset pulses to be 9 pulses for all axes.
i8094H_FIXED_MOVE(cardNo, AXIS_XYZU, 10000);
//move all axes on module 1 to be 10000 pulses.
```

Note: All relevant parameters must be modified accordingly to achieve the desired motion.

6.1.2 Setting the Start Speed

- ****void i8094H_SET_SV(BYTE *cardNo*, BYTE *axis*, DWORD *data*)**

Description:

This function sets the initial speed for the assigned axes.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
***data*:**	**Set up speed(Please refer to 2.5 for maximm speed)PPS**

Return:

None

Example:

```
i8094H_SET_SV(1, AXIS_X, 1000);
//set the starting speed for the X axis on module 1 to 1000 PPS.
```

6.1.3 Setting the Desired Speed

- ****void i8094H_SET_V(BYTE *cardNo*, BYTE *axis*, DWORD *data*)**

Description:

This function sets the desired speed for the assigned axes.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
***data*:**	**Set up speed(Please refer to 2.5 for maximm speed)PPS**

Return:

None

Example:

```
i8094H_SET_V(1, AXIS_X, 120000L);
//set the speed for the X axis on module 1 to 120000 PPS.
```

6.1.4 Setting the Acceleration

- ****void i8094H_SET_A(BYTE *cardNo*, BYTE *axis*, DWORD *data*)**

Description:

This function sets the acceleration value for the assigned axes.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
***data*:**	**The acceleration value (PPS/Sec). This value is related to the maximum speed value defined by i8094H_SET_MAX_V() function. The maximum available acceleration value is MAX_V * 125. The minimum acceleration value is MAX_V ÷ 64, and all other acceleration values are the integral multipliers of this value. The practical value for application depends on the capability of the motor drive and motor.**

Return:

None

Example:

```
i8094H_SET_MAX_V(1, AXIS_X, 20000);
//set the minimum speed value of the X axis as 20,000 ÷ 64 = 312.5 × n ≒
313…625…938…
//set the maximum speed value of the X axis as 20,000 PPS.
//therefore, do not set any acceleration value that is larger than
//20,000*125 PPS/sec.   And 20,000 *125   = 2,500,000.
i8094H_SET_A(1, AXIS_X, 100000L);
//set the acceleration value of the X axis on module 1 to 100K PPS/Sec.
```

6.1.5 Setting the Deceleration

- ****void i8094H_SET_D(BYTE *cardNo*, BYTE *axis*, DWORD *data*)**

Description:

This function sets the deceleration value for the assigned axes.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
***data*:**	**The deceleration value. The units are PPS/Sec. This value is related to the maximum speed value defined by i8094H_SET_MAX_V() function. The maximum available deceleration value is MAX_V * 125. The minimum deceleration value is MAX_V ÷ 64, and all other deceleration values are the integral multipliers of this value. The practical value for application depends on the capability of the motor drive and motor.**

Return:

None

Example:

```
i8094H_SET_MAX_V(1, AXIS_X, 20000);
//set the minimum speed value of the X axis as 20,000 ÷ 64 = 312.5 × n ≒
      313…625…938…
//set the maximum speed value of the X axis as 20,000 PPS.
//therefore, do not set any deceleration value that is larger than
//20,000*125 PPS/sec. And 20,000 *125   = 2,500,000.
i8094H_SET_D(1, AXIS_X, 100000L);
//set the deceleration value of the X axis on module 1 to 100K PPS/Sec.
```

6.1.6 Setting the Acceleration Rate

- ****void i8094H_SET_K(BYTE *cardNo*, BYTE *axis*, DWORD *data*)**

Description:

The function sets the acceleration rate (i.e., Jerk) value for the assigned axes.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
***data*:**	**The acceleration rate (jerk) value(PPS/ Sec^2). This value is related to the maximum speed value defined by i8094H_SET_MAX_V() function. The maximum available acceleration rate value is MAX_V * 781.25. The minimum acceleration value is MAX_V * 0.0119211, and all other acceleration values are the integral multipliers of this value. The practical value for application depends on the capability of the motor drive and motor. Note: since the DWORD can not represent the maximum value; therefore, this value is given by dividing the desired value by 10.**

Return:

None

Example:

```
i8094H_SET_MAX_V(1, AXIS_X, 20000);
//set the minimum speed value of the X axis as 20,000 × 0.0119211 = 238.422
× n ≒238…476…
//set the maximum speed value of the X axis as 20,000 PPS.
//therefore, do not set any jerk value that is larger than
//20,000*781.25 PPS/sec^2. And 20,000 *781.25   = 15,625,000.
i8094H_SET_K(1, AXIS_X, 10000);
//set the acceleration rate value of the X axis on module 1 to
//1,000*10 (= 10,000) PPS/Sec^2.
```

6.1.7 Setting the Deceleration Rate

- ****void i8094H_SET_L(BYTE *cardNo*, BYTE *axis*, DWORD *data*)**

Description:

This function sets the deceleration rate (i.e., Jerk) value for the assigned axes.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
***data*:**	**The deceleration rate value. The units are PPS/Sec^2. This value is related to the maximum speed value defined by i8094H_SET_MAX_V() function. The maximum available deceleration rate value is MAX_V * 781.25. The minimum deceleration value is MAX_V * 0.0119211, and all other deceleration values are the integral multipliers of this value. The practical value for application depends on the capability of the motor drive and motor. Note: since the DWORD can not represent the maximum value; therefore, this value is given by dividing the desired value by 10.**

Return:

None

Example:

```
i8094H_SET_MAX_V(1, AXIS_X, 20000);
//set the maximum speed value of the X axis as 20,000 PPS.
//therefore, do not set any deceleration rate value that is larger
//than 20,000*781.25 PPS/sec^2. And 20,000 *781.25    = 15,625,000.
i8094H_SET_L(1, AXIS_X, 10000);
//set the acceleration rate value of the X axis on module 1 to
//1,000*10 (= 10,000) PPS/Sec^2.
```

6.1.8 Setting the Value of the Remaining Offset Pulses

- ****void i8094H_SET_AO(BYTE *cardNo*, BYTE *axis*, long *data*)**

Description:

This function sets the number of remaining offset pulses for the assigned axes. Please refer to the figure below for a definition of the remaining offset pulse value.

Parameters:

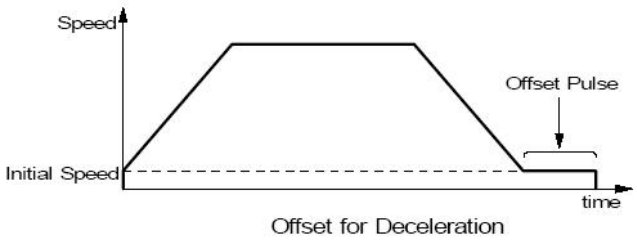
***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
***data*:**	**The number of remaining offset pulses. (-32,768 ~ +32,767)**

Return:

None

Example:

```
i8094H_SET_AO(1, AXIS_X, 200);
//set the number of remaining offset pulses for the X axis on
//module 1 to 200 pulses.
```



Offset for Deceleration

6.1.9 Fixed Pulse Output

- ****void i8094H_FIXED_MOVE(BYTE *cardNo*, BYTE *axis*, long *data*)**

Description:

Command a point-to-point motion for Fixed Position movement.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis (Please refer to Table 2-1.) The axis can be either X, Y, Z, or U.**
***data*:**	**Pulses (-2,000,000,000 ~ +2,000,000,000)**

Return:

YES	**Some errors happen. Use i8094H_GET_ERROR_CODE () to identify the errors.**
NO	**No error.**

Example:

```
BYTE cardNo=1; //select module 1
i8094H_SET_MAX_V(cardNo, AXIS_XYZU, 20000);
//set the max. velocity of all axes on module 1 to be 20K PPS
i8094H_NORMAL_SPEED(cardNo, AXIS_XYZU, 0);
//set the speed profile of all axes on module 1 to be symmetric T-curve
i8094H_SET_V(cardNo, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to be 2000 PPS
i8094H_SET_A(cardNo, AXIS_XYZU,1000);
//set the acceleration value of all axes on module 1 to be 1000 PPS/S
i8094H_SET_SV(cardNo, AXIS_XYZU, 2000);
//set the start velocity of all axes on module 1 to be 2000 PPS
i8094H_SET_AO(cardNo, AXIS_XYZU, 9);
//set the remaining offset pulses to be 9 PPS
i8094H_FIXED_MOVE(cardNo, AXIS_XYZU, 10000);
// move 10000 Pulses for each axis on module 1
```

- ****void i8094H_SET_PULSE(BYTE *cardNo*, BYTE *axis*, DWORD *data*)**

Description:

This command enables to make a change of pulse during outputing fixed pulses on each axis (but not for change of the direction).

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis (Please refer to Table 2-1.)**
	The axis can be either X, Y, Z, or U.
***data*:**	**Pulses (0 ~ +2,000,000,000)**

Return:

None

Example:

```
BYTE cardNo=1; //select module 1
i8094H_SET_MAX_V(cardNo, AXIS_XYZU, 20000);
//set the max velocity of all axes on module 1 to be 20K PPS
i8094H_NORMAL_SPEED(cardNo, AXIS_XYZU, 0);
//set the speed profile of all axes on module 1 to be symmetric T-curve
i8094H_SET_V(cardNo, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to be 2000 PPS
i8094H_SET_A(cardNo, AXIS_XYZU,1000);
//set the acceleration value of all axes on module 1 to be 1000 PPS/S
i8094H_SET_SV(cardNo, AXIS_XYZU, 2000);
//set the start velocity of all axes on module 1 to be 2000 PPS
i8094H_SET_AO(cardNo, AXIS_XYZU, 9);
//set the remaining offset pulses to be 9 PPS
i8094H_FIXED_MOVE(cardNo, AXIS_XYZU, 10000);
// move 10000 Pulses for each axis on module 1
i8094H_SET_PULSE(cardNo, AXIS_XYZU, 9000);
//Set pulse as 9000 Pulse。
```

6.1.10 Continuous Pulse Output

- ****void i8094H_CONTINUE_MOVE(BYTE *cardNo*, BYTE *axis*, long *data*)**

Description:

This function is continuous motion command for several independent axes.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1) The axis can be either X, Y, Z, or U.**
***data*:**	**The specified speed (positive value for CW motion; negative value for CCW motion)**

Return:

YES	**An error has occurred. Use the i8094H_GET_ERROR_CODE() function to identify the errors.**
NO	**No error.**

Example:

```
BYTE cardNo=1; //select module 1
i8094H_SET_MAX_V(cardNo, AXIS_XYZU, 20000);
//set the maximum speed of all axes on module 1 to 20K PPS.
i8094H_NORMAL_SPEED(cardNo, AXIS_XYZU, 0);
//set the speed profile for all axes as a symmetric T-curve.
i8094H_SET_V(cardNo, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to 2000 PPS.
i8094H_SET_A(cardNo, AXIS_XYZU, 1000);
//set the acceleration value of all axes to 1000 PPS/S.
i8094H_SET_SV(cardNo, AXIS_XYZU, 2000);
//set the start velocity of all axes to 2000 PPS
i8094H_CONTINUE_MOVE(cardNo, AXIS_XYZU, 1000);
//move all axes on module 1 at a speed of 1000 PPS.
```

6.2 Interpolation Commands

6.2.1 Assigning the Axes for Interpolation

- ****void i8094H_AXIS_ASSIGN(BYTE *cardNo*, BYTE *axis1*, BYTE *axis2*, BYTE *axis3*)**

Description:

This function assigns the axes to be used for interpolation. Either two or three axes can be assigned by using this function. Interpolation commands will refer to the assigned axes to construct a working coordinate system. The X axis does not necessarily have to be the first axis. However, it is easier to use the X axis as the first axis, the Y axis as the second axis, and the Z axis as the third axis.

Parameters:

***cardNo*:**	**Module number**
***axis1*:**	**The first axis; can be either X, Y, Z, or U(1、2、4、8). Please refer to Table 2-1 for the axis definition.**
***axis2*:**	**The second axis; can be either X, Y, Z, or U(1、2、4、8).**
***axis3*:**	**The third axis; can be either X, Y, Z, or U(1、2、4、8). Without 3rd axis the value is 0.**

Return:

None

EXAMPLE:

```
i8094H_AXIS_ASSIGN(1, AXIS_X, AXIS_Y, 0);
//set the X axis of module 1 as the first axis and the Y axis as the second axis.
```

6.2.2 Setting the Speed and Acc/Dec Mode for Interpolation

- ****void i8094H_VECTOR_SPEED(BYTE *cardNo*, BYTE *nMode*)**

Description:

This function sets vector acceleration or deceleration mode.

Parameters:

***cardNo*: Module number**

***nMode*: 0 → 2-axis linear or circular motion at a constant vector speed (Set VV and VSV; and VV=VSV)**

1 → 2-axis linear motion using a symmetric T-curve velocity profile (set VSV, VV, VA, and VAO)

2 → 2-axis linear motion using a symmetric S-curve velocity profile(set VSV, VV, VK, and VAO)

3 → 2-axis linear motion using an asymmetric T-curve velocity profile (set VSV, VV, VA, VD, and VAO)

4 → 2-axis linear motion using an asymmetric S-curve velocity profile (set VSV, VV, VK, VL, and VAO)

5 → 2-axis circular motion using a symmetric T-curve velocity profile (set VSV, VV, VA, and VAO)

6 → 2-axis circular motion using an asymmetric T-curve velocity profile (set VSV, VV, VA, VD, and VAO)

7 → 3-axis linear motion at a constant vector speed (set VV and VSV; and VV=VSV)

8 → 3-axis linear motion at using a symmetric T-curve velocity profile (set VSV, VV, VA, and VAO)

9 → 3-axis linear motion using a symmetric S-curve velocity profile (set VSV, VV, VK, and VAO)

10 →3-axis linear motion using an asymmetric T-curve velocity profile (set VSV, VV, VA, VD, and VAO)

11 →3-axis linear motion using an asymmetric S-curve velocity profile (set VSV, VV, VK, VL, and VAO)

Return:

None

Example:

```
BYTE cardNo=1; //select module 1.
i8094H_SET_MAX_V(cardNo, AXIS_XYZU, 20000);
//set the maximum speed of all axes to 20K PPS.

//============================================
i8094H_AXIS_ASSIGN(cardNo, AXIS_X, AXIS_Y, 0);
//set the X axis as the first axis and the Y axis as the second axis.
i8094H_VECTOR_SPEED(cardNo, 0);
//set module 1 to perform 2-axis linear or circular motion
//at a constant vector speed.
i8094H_SET_VSV(cardNo, 1000);
//set the starting vector speed to 1000 PPS.
i8094H_SET_VV(cardNo, 1000);
//set the vector speed to 1000 PPS.
i8094H_LINE_2D(1, 12000, 10000);
//execute the 2-axis linear interpolation motion.

//============================================
i8094H_DEC_ENABLE(cardNo);
//enable the deceleration function.
i8094H_AXIS_ASSIGN(cardNo, AXIS_X, AXIS_Y, 0);
//set the X axis as the first axis and the Y axis as the second axis.
i8094H_VECTOR_SPEED(cardNo, 1);
//set module 1 to perform 2-axis linear motion using a symmetric
//S-curve velocity profile.
i8094H_SET_VSV(cardNo, 500);
//set the starting vector speed to 500 PPS.
i8094H_SET_VV(cardNo, 2000);
//set the vector speed to 2000 PPS.
i8094H_SET_VA(cardNo, 1000);
//set the vector acceleration to 1000 PPS/Sec.
i8094H_LINE_2D(cardNo, 20000, 10000);
//execute the 2-axis linear interpolation motion.

//============================================
i8094H_AXIS_ASSIGN(cardNo, AXIS_X, AXIS_Y, 0);
//set the X axis as the first axis and the Y axis as the second axis.
i8094H_VECTOR_SPEED(cardNo, 2);
//2-axis linear motion using a symmetric S-curve velocity profile.
i8094H_SET_VSV(cardNo, 200);
//set the starting vector speed to 200 PPS.
i8094H_SET_VV(cardNo, 2000);
//set the vector speed to 2000 PPS.
i8094H_SET_VK(cardNo, 50);
//set the acceleration rate to 500 PPS/Sec.
i8094H_SET_VAO(cardNo, 20);
//set the value of remaining offset pulses to 20.
i8094H_LINE_2D(cardNo, 10000, 10000);
//execute the 2-axis linear interpolation motion.
```

```
//===============================================
i8094H_DEC_ENABLE(cardNo);
//enable the deceleration function.
i8094H_AXIS_ASSIGN(cardNo, AXIS_X, AXIS_Y, 0);
//set the X axis as the first axis and the Y axis as the second axis.
i8094H_VECTOR_SPEED(cardNo, 3);
//2-axis linear motion using an asymmetric T-curve velocity profile.
i8094H_SET_VSV(cardNo, 100);
//set the start vector speed to 100 PPS.
i8094H_SET_VV(cardNo, 2000);
//set the vector speed to 2000 PPS.
i8094H_SET_VA(cardNo, 1000);
//set the vector acceleration to 1000 PPS/Sec.
i8094H_SET_VD(cardNo, 500);
//set the vector deceleration to 500 PPS/Sec.
i8094H_SET_VAO(cardNo, 20);
//set the value of remaining offset pulses to 20.
i8094H_LINE_2D(cardNo, 10000, 5000);
//execute the 2-axis linear interpolation motion.

//===============================================
long fp1=4000;
long fp2=10000;
int sv=200;
int v=2000;
i8094H_SET_MAX_V(cardNo, AXIS_XYZU, 8000);
i8094H_AXIS_ASSIGN(cardNo, AXIS_X, AXIS_Y, 0);
//set the X axis as the first axis and the Y axis as the second axis.
i8094H_VECTOR_SPEED(cardNo, 4);
//2-axis linear motion using an asymmetric S-curve velocity profile.
i8094H_SET_VSV(cardNo, sv);
//set the starting velocity to sv PPS.
i8094H_SET_VV(cardNo, v);
//set the vector speed to v PPS.
i8094H_SET_VK(cardNo, 50);
//set the acceleration rate to 500 PPS/Sec^2.
i8094H_SET_VL(cardNo, 30);
//set the deceleration rate to 300 PPS/Sec^2.
i8094H_SET_VAO(cardNo, 20);
//set the value of remaining offset pulses to 20.
i8094H_LINE_2D(cardNo, fp1, fp2);
//execute the 2-axis linear motion.

//===============================================
long fp1=11000;
long fp2=9000;
long c1=10000;
long c2=0;
int sv=100;
int v=3000;
int a=5000;
```

```
int d=5000;
i8094H_SET_MAX_V(cardNo, AXIS_XYZU, 8000);
i8094H_AXIS_ASSIGN(cardNo, AXIS_X, AXIS_Y, 0);
//set the X axis as the first axis and the Y axis as the second axis.
i8094H_VECTOR_SPEED(cardNo, 5);
//2-axis circular motion using a symmetric T-curve velocity profile
i8094H_SET_VSV(cardNo, sv);
//set the starting vector speed to sv PPS.
i8094H_SET_VV(cardNo, v);
//set vector speed to v PPS.
i8094H_SET_VA(cardNo, a);
//set the vector acceleration to a PPS/Sec.
i8094H_SET_VAO(cardNo, 0);
//set the value of remaining offset pulses to 0 Pulse.
i8094H_ARC_CW(cardNo, c1,c2, fp1, fp2);
//execute the 2-axis CW circular motion.

//=============================================
long c1=300;
long c2=0;
int sv=100;
int v=3000;
int a=125;
int d=12;
i8094H_SET_MAX_V(cardNo, AXIS_XYZU, 8000);
i8094H_AXIS_ASSIGN(cardNo, AXIS_X, AXIS_Y, 0);
//set the X axis as the first axis and the Y axis as the second axis.
i8094H_VECTOR_SPEED(cardNo, 6);
//2-axis circular motion using an asymmetric T-curve velocity
//profile.
i8094H_SET_VSV(cardNo, sv);
//set the starting vector speed to sv PPS.
i8094H_SET_VV(cardNo, v);
//set vector speed to v PPS.
i8094H_SET_VA(cardNo, a);
//set acceleration to a PPS/Sec.
i8094H_SET_VD(cardNo, d);
//set the deceleration to d PPS/Sec.
i8094H_SET_VAO(cardNo, 0);
//set the value of remaining offset pulses to 0.
i8094H_CIRCLE_CW(cardNo, c1, c2);
//execute the 2-axis CW circular motion.

//=============================================
i8094H_AXIS_ASSIGN(cardNo, AXIS_X, AXIS_Y, AXIS_Z);
//set axis1 as the X axis, axis2 as the Y axis, and axis3 as the Z axis.
i8094H_VECTOR_SPEED(cardNo, 7);
//3-axis linear motion at a constant vector speed (VSV=VV).
i8094H_SET_VSV(cardNo, 1000);
//set the start speed to 1000 PPS.
i8094H_SET_VV(cardNo, 1000);
```

```
//set the constant speed to 1000 PPS.
i8094H_LINE_3D(cardNo, 10000, 10000,10000);
//execute the 3-axis linear motion.

//===============================================
i8094H_AXIS_ASSIGN(cardNo, AXIS_X, AXIS_Y, AXIS_Z);
//set axis1 as the X axis, axis2 as the Y axis, and axis3 as the Z-axis.
i8094H_VECTOR_SPEED(cardNo, 8);
//3-axis linear motion using a symmetric T-curve velocity profile.
i8094H_SET_VSV(cardNo, 100);
//set the starting speed to 1000 PPS.
i8094H_SET_VV(cardNo, 3000);
//set the vector speed to 3000 PPS.
i8094H_SET_VA(cardNo, 500);
//set the vector acceleration to 500 PPS/Sec.
i8094H_SET_VAO(cardNo, 20);
//set the value of remaining offset pulses to 20.
i8094H_LINE_3D(cardNo, 10000, 1000,20000);
//execute the 3-axis linear motion

//===============================================
i8094H_AXIS_ASSIGN(cardNo, AXIS_X, AXIS_Y, AXIS_Z);
//set the axis1 as the X axis, axis2 as the Y axis, and axis3 as the Z axis.
i8094H_VECTOR_SPEED(cardNo, 9);
//3-axis linear motion using a symmetric S-curve velocity profile.
i8094H_SET_VSV(cardNo, 100);
//set the starting speed to 1000 PPS.
i8094H_SET_VV(cardNo, 3000);
//set the vector speed to 3000 PPS.
i8094H_SET_VK(cardNo, 50);
//set the vector acceleration rate to 500 PPS/Sec^2.
i8094H_SET_VAO(cardNo, 20);
//set the value of remaining offset pulses to 20.
i8094H_LINE_3D(cardNo, 10000, 1000,1000);
//execute the 3-axis linear motion.

//===============================================
i8094H_AXIS_ASSIGN(cardNo, AXIS_X, AXIS_Y, AXIS_Z);
//set the axis1 as the X axis, axis2 as the Y axis, and axis3 as the Z axis.
i8094H_VECTOR_SPEED(cardNo, 10);
//set the module 1 to perform 3-axis linear motion
//using an asymmetric T-curve speed profile.
i8094H_SET_VSV(cardNo, 100);
//set the starting speed to 1000 PPS.
i8094H_SET_VV(cardNo, 2000);
//set the vector speed as 3000 PPS.
i8094H_SET_VA(cardNo, 1000);
//set the vector acceleration to 1000 PPS/Sec.
i8094H_SET_VD(cardNo, 500);
//set the vector deceleration to 500 PPS/Sec.
i8094H_SET_VAO(cardNo, 20);
```

```
//set the value of remaining offset pulses to 20.
i8094H_LINE_3D(cardNo, 10000, 1000,1000);
//execute the 3-axis linear motion.

//===========================================
long fp1=4000;
long fp2=10000;
long fp3=20000;
int sv=200;
int v=2000;
i8094H_SET_MAX_V(cardNo, AXIS_XYZU, 8000);

i8094H_AXIS_ASSIGN(cardNo, AXIS_X, AXIS_Y, AXIS_Z);
//set axis1 as the X axis, axis2 as the Y axis, and axis3 as the Z axis.
i8094H_VECTOR_SPEED(cardNo, 11);
//3-axis linear motion using an asymmetric S-curve velocity profile.
i8094H_SET_VSV(cardNo, sv);
//set the starting speed to sv PPS.
i8094H_SET_VV(cardNo, v);
//set the vector speed to v PPS.
i8094H_SET_VK(cardNo, 50);
//set the vector acceleration rate to 500 PPS/Sec^2.
i8094H_SET_VL(cardNo, 30);
//set the vector deceleration rate to 300 PPS/Sec^2.
i8094H_SET_VAO(cardNo, 20);
//set the value of remaining offset pulses to 20.
i8094H_LINE_3D(cardNo, fp1, fp2,fp3);
//execute the 3-axis linear motion.
```

Note: Relevant parameters will be adjusted accordingly before issuing the motion command.

6.2.3 Setting the Vector Starting Speed

- ****void i8094H_SET_VSV(BYTE *cardNo*, DWORD *data*)**

Description:

This function sets the starting speed of the principle axis (axis 1) for the interpolation motion.

Parameters:

***cardNo*:**	**Module number**
***data*:**	**The vector starting speed value (in PPS) (For maximum value please refer to Table 2-5)**

Return:

None

Example:

```
i8094H_SET_VSV(1, 1000);
//set the starting speed of the axis 1 for the interpolation motion
//on module 1 to 1000 PPS.
```

6.2.4 Setting the Vector Speed

- ****void i8094H_SET_VV(BYTE *cardNo*, DWORD *data*)**

Description:

This function sets the vector speed of the interpolation motion. Users do not need to assign any axes on this function. The speed setting will take effect on the current working coordinate system which is defined by the i8094H_AXIS_ASSIGN() function.

Parameters:

***cardNo*:**	**Module number**
***data*:**	**The vector speed value (in PPS) (For maximum value please refer to Table 2-5)**

Return:

None

Example:

```
i8094H_SET_VV(1, 120000L);
//set the vector speed of the interpolation on module 1
//to 120000 PPS.
```

6.2.5 Setting the Vector Acceleration

- ****void i8094H_SET_VA(BYTE *cardNo*, DWORD *data*)**

Description:

This function sets the vector acceleration for interpolation motion. Users do not have to assign any axes on this funciton. This speed setting will take effect on the current working coordinate system which is defined by the i8094H_AXIS_ASSIGN() function.

Parameters:

***cardNo*:**	**Module number**
***data*:**	**The vector acceleration value (in PPS/Sec). The units are PPS/Sec. This value is related to the maximum speed value defined by i8094H_SET_MAX_V() function. The maximum available acceleration value is MAX_V * 125. The minimum acceleration value is MAX_V ÷ 64, and all other acceleration values are the integral multipliers of this value. The practical value for application depends on the capability of the motor drive and motor.**

Return:

None

Example:

```
i8094H_SET_MAX_V(1, AXIS_X, 20000);
//set the maximum speed value of the X axis as 20,000 PPS.
//therefore, do not set any acceleration value that is larger than
//20,000*125 PPS/sec.   And 20,000 *125   = 2,500,000.
i8094H_SET_VA(1, 100000L);
//set the vector acceleration of the interpolation motion
//on module 1 to 100K PPS/Sec.
```

6.2.6 Setting the Vector Deceleration Value

- ****void i8094H_SET_VD(BYTE *cardNo*, DWORD *data*)**

Description:

This function sets the deceleration value for the interpolation motion.

Parameters:

***cardNo*:**	**Module number**
***data*:**	**The vector deceleration value (in PPS/Sec)(Please refer to Table 2.5). This value is related to the maximum speed value defined by i8094H_SET_MAX_V() function. The maximum available deceleration value is MAX_V * 125. The minimum deceleration value is MAX_V ÷ 64, and all other deceleration values are the integral multipliers of this value. The practical value for application depends on the capability of the motor drive and motor.**

Return:

None

Example:

```
i8094H_SET_MAX_V(1, AXIS_X, 20000);
//set the maximum speed value of the X axis as 20,000 PPS.
//therefore, do not set any deceleration value that is larger than
//20,000*125 PPS/sec.   And 20,000 *125   = 2,500,000.
i8094H_SET_VD(1, 100000L);
//set the vector deceleration value of interpolation motion
//on module 1 to 100K PPS/Sec.
```

6.2.7 Setting the Vector Acceleration Rate

- ****void i8094H_SET_VK(BYTE *cardNo*, DWORD *data*)**

Description:

Set the acceleration rate (jerk) value for interpolation motion.

Parameters:

***cardNo*: Module number**

***data*: The acceleration rate (jerk) value (PPS/ Sec^2). (Please refer to Table 2.5). This value is related to the maximum speed value defined by i8094H_SET_MAX_V() function. The maximum available acceleration rate value is MAX_V * 781.25. The minimum acceleration value is MAX_V * 0.0119211, and all other acceleration values are the integral multipliers of this value. The practical value for application depends on the capability of the motor drive and motor. Note: since the DWORD can not represent the maximum value; therefore, this value is given by dividing the desired value by 10.**

Return:

None

Example:

```
i8094H_SET_MAX_V(1, AXIS_X, 20000);
//set the maximum speed value of the X axis as 20,000 PPS.
//therefore, do not set any jerk value that is larger than
//20,000*781.25 PPS/sec^2. And 20,000 *781.25    = 15,625,000.
i8094H_SET_VK(1, 10000);
//set the acceleration rate of the interpolation motion on module
// 1 to 10,000 PPS/ Sec^2.
```

6.2.8 Setting the Vector Deceleration Rate

- ****void i8094H_SET_VL(BYTE *cardNo*, DWORD *data*)**

Description:

This function sets the deceleration rate of the interpolation motion.

Parameters:

***cardNo*: Module number**

***data*: The deceleration rate (Jerk) value(PPS/ Sec^2)(Please refer to Table 2.5). This value is related to the maximum speed value defined by i8094H_SET_MAX_V() function. The maximum available deceleration rate value is MAX_V * 781.25. The minimum deceleration value is MAX_V * 0.0119211, and all other deceleration values are the integral multipliers of this value. The practical value for application depends on the capability of the motor drive and motor. Note: since the DWORD can not represent the maximum value; therefore, this value is given by dividing the desired value by 10.**

Return:

None

Example:

```
i8094H_SET_MAX_V(1, AXIS_X, 20000);
//set the maximum speed value of the X axis as 20,000 PPS.
//therefore, do not set any deceleration rate value that is larger
//than 20,000*781.25 PPS/sec^2. And 20,000 *781.25   = 15,625,000.
i8094H_SET_VL(1, 10000);
//set the deceleration rate of the interpolation on module 1 to 10,000
PPS/Sec^2.
```

6.2.9 Setting the Number of the Remaining Offset Pulses

- ****void i8094H_SET_VAO(BYTE *cardNo*, long *data*)**

Description:

Setting this value will cause the motion control chip to start deceleration earlier. The remaining offset pulses will be completed at low speed to allow the controller to stop immediately when it reaches the offset pulse value. Please refer to the figure below for more information.

Parameters:

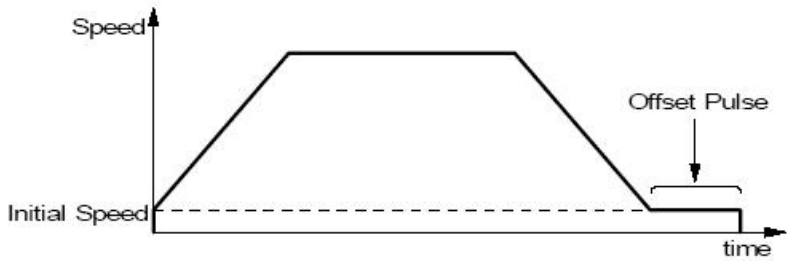
***cardNo*:**	**Module number**
***data*:**	**The number of remaining offset pulses (-32,768 ~ +32,767)**

Return:

None

Example:

```
i8094H_SET_VAO(1, 200);
//set the number of remaining offset pulse value on module 1 to 200.
```



Offset for Deceleration

6.2.10 2-Axis Linear Interpolation Motion

- ****void i8094H_LINE_2D(BYTE *cardNo*, long *fp1*, long *fp2*)**

Description:

This function executes a 2-axis linear interpolation motion.

Parameters:

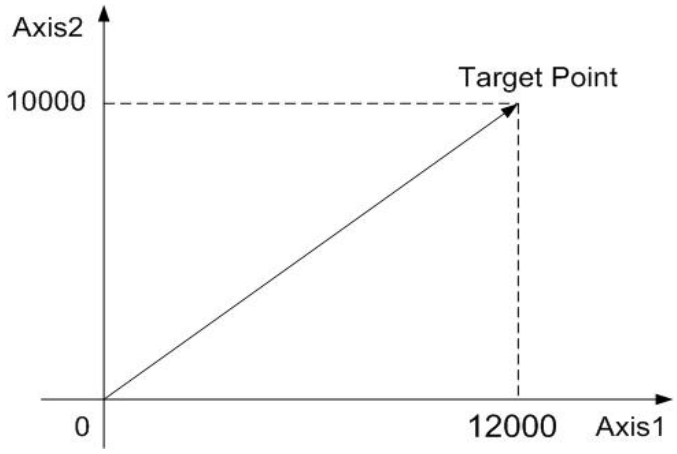
***cardNo*:**	**Module number**
***fp1*:**	**The displacement of the axis 1 in Pulses (-2,000,000,000 ~ +2,000,000,000)**
***fp2*:**	**The displacement of the axis 2 in Pulses (-2,000,000,000 ~ +2,000,000,000)**

Return:

YES	**An error has occurred. Use the i8094H_GET_ERROR_CODE() function to identify the error.**
NO	**No errors.**

Example:

```
i8094H_LINE_2D(1, 12000, 10000);
//execute the 2-axis linear interpolation motion on module 1.
```



2-axis linear interpolation motion

6.2.11 3-axis Linear Interpolation Motion

- ****void i8094H_LINE_3D(BYTE *cardNo*, long *fp1*, long *fp2*, long *fp3*)**

Description:

This function executes a 3-axis linear interpolation motion.

Parameters:

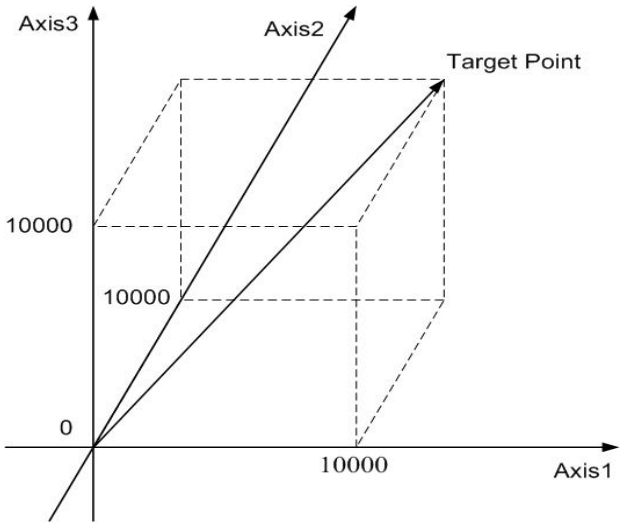
***cardNo*:**	**Module number**
***fp1*:**	**The displacement of the first axis (axis 1) in Pulses (-2,000,000,000 ~ +2,000,000,000)**
***fp2*:**	**The displacement of the second axis (axis 2) in Pulses (-2,000,000,000 ~ +2,000,000,000)**
***fp3*:**	**The displacement of the third axis (axis 3) in Pulses (-2,000,000,000 ~ +2,000,000,000)**

Return:

YES	**An error has occurred. Use the i8094H_GET_ERROR_CODE() function to identify the error.**
NO	**No errors.**

Example:

```
i8094H_LINE_3D(1, 10000, 10000, 10000);
//execute the 3-axis linear interpolation motion on module 1.
```



3-axis linear interpolation motion

6.2.12 2-Axis Circular Interpolation Motion (an Arc)

- ****void i8094H_ARC_CW(BYTE *cardNo*, long *cp1*, long *cp2*, long *fp1*, long *fp2*)**

Description:

This function executes a 2-axis circular interpolation motion in a clockwise (CW) direction.

Parameters:

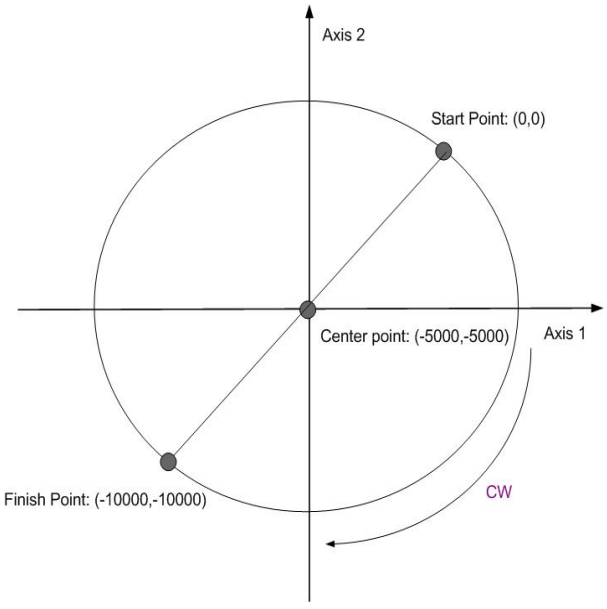
***cardNo*:**	**Module number**
***cp1*:**	**The relative position of the center to the current position of axis 1 in pulses. (-2,000,000,000 ~ +2,000,000,000)**
***cp2*:**	**The relative position of the center to the current position of axis 2 in pulses. (-2,000,000,000 ~ +2,000,000,000)**
***fp1*:**	**The displacement of the axis 1 in pulses. (-2,000,000,000 ~ +2,000,000,000)**
***fp2*:**	**Displacement of the axis 2 in pulses. (-2,000,000,000 ~ +2,000,000,000)**

Return:

YES	**An error has occurred. Use the i8094H_GET_ERROR_CODE () function to identify the error.**
NO	**No errors.**

Example:

```
i8094H_ARC_CW(1, -5000, -5000, -10000, -10000);
//Issues a command to perform a circular motion (an arc)
//in a CW direction. Please refer to the following figure.
```



2-axis circular motion in a CW direction

- ****void i8094H_ARC_CCW(BYTE *cardNo*, long *cp1*, long *cp2*,
  long *fp1*, long *fp2*)**

Description:

This function execute a 2-axis circular interpolation motion in a counter-clockwise (CCW) direction.

Parameters:

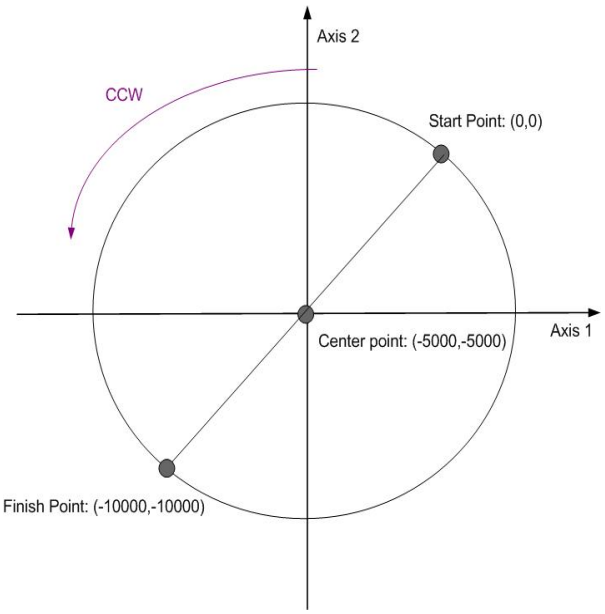
***cardNo*:**	**Module number**
***cp1*:**	**The relative position of the center to the current position of axis 1 in pulses. (-2,000,000,000 ~ +2,000,000,000)**
***cp2*:**	**The relative position of the center to the current position of axis 2 in pulses. (-2,000,000,000 ~ +2,000,000,000)**
***fp1*:**	**The displacement of the axis 1 in pulses. (-2,000,000,000 ~ +2,000,000,000)**
***fp2*:**	**Displacement of the axis 2 in pulses. (-2,000,000,000 ~ +2,000,000,000)**

Return:

YES	**An error has occurred. Use the i8094H_GET_ERROR_CODE() function to identify the errors.**
NO	**No errors.**

Example:

```
i8094H_ARC_CCW(1, -5000, -5000, -10000, -10000);
//Issues a command to perform a circular motion (an arc)
//in a CCW direction. Please refer to the following figure.
```



2-axis circular motion in a CCW direction

6.2.13 2-Axis Circular Interpolation Motion (a Complete Circle)

- ****void i8094H_CIRCLE_CW(BYTE *cardNo*, long *cp1*, long *cp2*)**

Description:

This function executes a 2-axis circular interpolation motion in a clockwise (CW) direction.

Parameters:

***cardNo*:**	**Module number**
***cp1*:**	**The relative position of the center to the current position of axis 1 in pulses. (-2,000,000,000 ~ +2,000,000,000)**
***cp2*:**	**The relative position of the center to the current position of axis 2 in pulses. (-2,000,000,000 ~ +2,000,000,000)**

Return:

YES	**An error has occurred. Use the i8094H_GET_ERROR_CODE() function to identify the errors.**
NO	**No errors.**

Example:

```
i8094H_CIRCLE_CW(1, 0, 10000);
//execute a circular motion (a complete circle) in a CW direction on module 1.
```

- ****void i8094H_CIRCLE_CCW(BYTE *cardNo*, long *cp1*, long *cp2*)**

Description:

This function executes a 2-axis circular interpolation motion in a counter-clockwise (CCW) direction.

Parameters:

***cardNo*:**	**Module number**
***cp1*:**	**The relative position of the center to the current position of axis 1 in pulses. (-2,000,000,000 ~ +2,000,000,000)**
***cp2*:**	**The relative position of the center to the current position of axis 2 in pulses. (-2,000,000,000 ~ +2,000,000,000)**

Return:

YES	**An error has occurred. Use the i8094H_GET_ERROR_CODE () function to identify the error.**
NO	**No errors**

Example:

```
i8094H_CIRCLE_CCW(1, 0, 10000);
//execute a circular motion (a circle) in CCW direction
//on module 1
```

6.3 Synchronous Actions

6.3.1 Setting the Synchronous Action

- ****void i8094H_SYNC_ACTION(**
  BYTE *cardNo*, BYTE *axis1*, BYTE *axis2*, DWORD *nSYNC*,
  BYTE *nDRV*, BYTE *nLATCH*, BYTE *nPRESET*, BYTE *nOUT*,
  BYTE *nINT*, BYTE *isrNoX*, BYTE *isrNoY*, BYTE *isrNoZ*, BYTE *isrNoU*)

Description:

This function sets the activation factors for synchronous action. (It's total-hardware-solution, will not consume any WinCon or I8000 system resources)

Parameters:

***cardNo*:** **Module number**

***axis1*:** **This is the monitored axis. It will be checked by hardware. The axis can be either X, Y, Z, or U (1 or 2 or 4 or 8). (Please refer to Table 2-1.)**

***axis2*:** **This defined the other axes (or axis) that will take action when one of the activation factors occurs. The axes are defined in the following table.**

axis1 / axis2	X	Y	Z	U
0	X	Y	Z	U
1	Y	Z	U	X
2	Z	U	X	Y
3	YZ	ZU	UX	XY
4	U	X	Y	Z
5	YU	ZX	UY	XZ
6	ZU	UX	XY	YZ
7	YZU	ZUX	UXY	XYZ

If axis2 is set as 0, it means the synchronous actions axis and moving axis are the same axis.

***nSYNC*:** **It defines the activation factors. Multiple activation factors can be defined at the same time. Available active factors are listed in the following table.**

Value	Event	Explanation
0x0000		Disable the synchronous action
0x0001	P≥C+	The logical/real position counter value exceeded the COMP+ register value. Use the i8094H_SET_COMPARE() function for selection of a logical/real position(6.3.2).
0x0002	P<C+	The logical/real position counter value became less than the COMP+ register value. Use the i8094H_SET_COMPARE() function for selection of a logical/real position(6.3.2).
0x0004	P<C-	The logical/real position counter value became less than the COMP- register value. Use the i8094H_SET_COMPARE() function for selection of a logical/real position(6.3.2).
0x0008	P≥C-	The logical/real position counter value exceeded the COMP- register value. Use the i8094H_SET_COMPARE() function for selection of a logical/real position(6.3.2).
0x0010	D-STA	Driving started.
0x0020	D-END	Driving terminated.
0x0040	IN3↑	The nIN3 signal rose from the Low to the High level.
0x0080	IN3↓	The nIN3 signal fell from the High to the Low level.

***nDRV*:** **It defines the actions that are related with axial driving. Available actions are listed in the following table. Only one driving action can be chosen.**

Value	Symbol	Explanation
0		Disable driving action.
1	FDRV+	Activates fixed pulse driving in the + direction. It must set the *nPRESET* value to be "OPSET" which indicates that i8094H_SET_PRESET() function will set the offset value for this FDRV. Therefore, the companion function, i8094H_SET_PRESET(), is necessary. However, this command does not take effect if the assigned axes, *axis2*, are moving.
2	FDRV-	Activates fixed pulse driving in the - direction. It must set the *nPRESET* value to be "OPSET" which indicates that i8094H_SET_PRESET() function will set the offset value for this FDRV. Therefore, the companion function, i8094H_SET_PRESET(), is necessary. However, this command does not take effect if the assigned axes, *axis2*, are moving.
3	CDRV+	Activates continuous pulse driving in the + direction. However, this command does not take

		effect if the assigned axes, *axis2*, are moving.
4	CDRV-	Activates continuous pulse driving in the - direction. However, this command does not take effect if the assigned axes, *axis2*, are moving.
5	SSTOP	Stop driving in deceleration.
6	ISTOP	Stop driving immediately.

nLATCH: It defines the actions that is related of latching position. Available actions are listed in the following table. Only one of these actions can be chosen.

Value	Symbol	Explanation
0		Disable position latch function.
1	LPSAV	Saves the current logical position counter value (LP) in the synchronous buffer register (BR). [LP → LATCH]
2	EPSAV	Saves the current real position counter value (EP) in the synchronous buffer register (BR). [EP → LATCH]

After the event is occoured, the i8094H_GET_LATCH() function can be use to get the latched value.

nPRESET: It defines the actions that is related of latching position. Available actions are listed in the following table. Only one of these actions can be chosen.

Value	Symbol	Explanation
0		Disable setting function.
1	LPSET	Indicates that a new value for the logical position (LP) will be set. The new value will be set by i8094H_SET_PRESET() function. [LP ← PRESET]
2	EPSET	Indicates that a new value for the real position (EP) will be set. The new value will be set by i8094H_SET_PRESET() function. [EP ← PRESET]
3	OPSET	Indicates that a new offset value (P) for the fixed pulse driving will be set. The new value will be set by i8094H_SET_PRESET() function. [P ← PRESET]
4	VLSET	Indicates that a new speed value (V) will be set. The new value will be set by i8094H_SET_PRESET() function. [V ← PRESET]

Must be used with i8094H_SET_PRESET together, please refer to section 6.3.4.

***nOUT*: setting trigger output, as the following table**

Value	Symbol	Explanation
0		Disable trigger output
1	OUT	Enable trigger output

Must be used with i8094H_SET_OUT together, please refer to section 6.3.5.

***nINT*: setting interrupt function, as the following table:**

Value	Symbol	Explanation
0		Disable interrupt function
1	INT	Enable interrupt function

1. It will generate a interrupt for the synchronous action of axis2, please write the corresponding number: *isrNoX*、*isrNoY*、*isrNoZ*、*isrNoU*

2. Must be used with i8094H_ENABLE_INT together, please refer to section 6.3.6.

***isrNoX*: ISR1 ~ ISR20 : Specify the interrupt number of X-axis, please refer to the section 7.2.1.**
0: disable

***isrNoY*: ISR1 ~ ISR20 : Specify the interrupt number of Y-axis, please refer to the section 7.2.1.**
0: disable

***isrNoZ*: ISR1 ~ ISR20 : Specify the interrupt number of Z-axis, please refer to the section 7.2.1.**
0: disable

***isrNoU*: ISR1 ~ ISR20 : Specify the interrupt number of U-axis, please refer to the section 7.2.1.**
0: disable

Return: **None**

Example:

```
//Ex1. When the rising edge event of IN3 signal of U-axis occurred,
// the real position (EP) is latched and the driving speed of U-axis is changed,
too.
i8094H_SYNC_ACTION(cardNo, AXIS_U, 0, 0X00000040, 0, 2, 4, 0, 0, 0);
i8094H_SET_MAX_V(cardNo, AXIS_U, 5000);
//Set the maximum speed of U-axis to 5K PPS.
i8094H_NORMAL_SPEED(cardNo, AXIS_U, 0);
//Set the Acc/Dec mode to be symmetric T-curve.
i8094H_SET_V(cardNo, AXIS_U, 2000);
//Set the speed of U-axis to 2000 PPS.
i8094H_SET_A(cardNo, AXIS_U, 100000);
//Set the acceleration of U-axis to 100K PPS/S.
i8094H_SET_SV(cardNo, AXIS_U, 100);
//Set the start speed of U-axis to 100 PPS.
i8094H_FIXED_MOVE(cardNo, AXIS_U, 10000);
//Set the fixed pulse moving command to 10000 Pulses.
i8094H_SET_PRESET(cardNo, AXIS_U, 100);
//Set the new speed of U-axis after even activation to 100 PPS.
while (i8094H_STOP_WAIT(cardNo, AXIS_U) == NO)
{     //If the U-axis of the assigned card is not stop, keep looping
      DoEvents();
      Sleep(1);    //Release the control for a moment
};
 //After the event occurred, following line can get latched position.
long Vsb = i8094H_GET_LATCH(cardNo, AXIS_U);
```

```
//Ex2. When the EP value of U-axis exceeds COMP+ (5,000),
//controller will move the Y-axis by 2,000 PPS.
i8094H_SYNC_ACTION(cardNo, AXIS_U, 2, 0X0001, 1, 0, 3);
i8094H_SET_COMPARE(cardNo, AXIS_U, 0, 1, 5000);
//Set the COMP+ of U-axis 5,000 and te compared source is real position (EP).
i8094H_SET_MAX_V(cardNo, AXIS_YU, 9000);
//Set the maximum speed of axesY and U to 9K PPS.
i8094H_NORMAL_SPEED(cardNo, AXIS_YU, 0);
//Set the Acc/Dec mode to be symmetric T-curve.
i8094H_SET_V(cardNo, AXIS_YU, 3000);
 //Set the speed of axes Y and U to 3,000 PPS.
i8094H_SET_A(cardNo, AXIS_YU, 200000);
//Set the acceleration of axes Y and U to 200K PPS/S.
i8094H_SET_SV(cardNo, AXIS_YU, 200);
//Set the start speed of axes Y and U to 200 PPS.
i8094H_SET_PRESET(cardNo, AXIS_Y, 2000);
//Set the fixed pulse drive of Y-axis to be 2,000 PPS when the activating
//event occurs.
i8094H_FIXED_MOVE(cardNo, AXIS_U, 10000);
//Command the U-axis to move 10,000 Pulses and the synchronous action
//will happen after a while.
```

6.3.2 Setting the COMPARE value

- ****void i8094H_SET_COMPARE(BYTE *cardNo*, BYTE *axis*, BYTE *nSELECT*, BYTE *nTYPE*, long *data*)**

Description:

This function sets the values of COMPARE registers. Howerer, it will disable the functions of software limits.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
***nSELECT*:**	**Select the COMPARE register 0 → C+ 1 → C-**
***nTYPE*:**	**Select the souece for comparison 0 → Position(P) = LP 1 → Position(P) = EP**
***data*:**	**Set the COMPARE value: -2,000,000,000 ~ +2,000,000,000**

Return:

None

Example:

```
i8094H_SET_COMPARE(cardNo, AXIS_U, 0, 1, 5000);
//Set the comparison function for U-Axis.
//Set the compared source to be EP; and the COMP+ value to 5000.
```

6.3.3 Get the LATCH value

- **long i8094H_GET_LATCH(BYTE *cardNo*, BYTE *axis*)

Description:

This function gets the values from the LATCH register.

Parameters:

cardNo:	Module number
axis:	The axis can be either X, Y, Z, or U (1 or 2 or 4 or 8). Please refer to Table 2-1.

Return:

Value of the LATCH register: -2,000,000,000 ~ +2,000,000,000

Example:

```
long data = i8094H_GET_LATCH(1, AXIS_Y);
//Get the latched value which is from Y-axis of card 1.
```

6.3.4 Set the PRESET data for synchronous action

- **void i8094H_SET_PRESET(BYTE *cardNo*, BYTE *axis*, long *data*)

Description:

This function sets the PRESET value for synchronous action(Each synchronous action axis could not be set individually.

Parameters:

cardNo:	Module number
axis:	
data:	LP: -2,000,000,000 ~ +2,000,000,000 EP: -2,000,000,000 ~ +2,000,000,000 P : -2,000,000,000 ~ +2,000,000,000 V : Please refer to section 2.5. If there are more than two synchronous action axes, please set i8094H_SET_MAX_V to be same value.

Return:

None

Example:

Please refer to the examples in section 6.3.1.

6.3.5 Set the Output Data

- ****void i8094H_SET_OUT(BYTE *cardNo*, BYTE *axis*, BYTE *outEdge*,
  BYTE *PulseWidth*)**

Description:

This function sets the output pulse settings.

Parameters:

***cardNo*: Module number**

***axis*: Axis or Axes (Please refer to Table 2-1 for the axis definition.), currently only support Axes X & Y.**

***outEdge*: trigger OUT active logic:**
0 = low active; 1 = high active

***PulseWidth*: trigger OUT output pulse width**

0 =	10	uSec
1 =	20	uSec
2 =	100	uSec
3 =	200	uSec
4 =	1,000	uSec
5 =	2,000	uSec
6 =	10,000	uSec
7 =	20,000	uSec

Return:

None

Example:

Please refer to the examples in section 6.3.1.

6.3.6 The interrupt function of i8094H

- ***void i8094H_ENABLE_INT(BYTE *cardNo*)**

Description: This function enables the interrupt function.

Parameters:

***cardNo*: Module number**

Return:

None

Example:

```
i8094H_ENABLE_INT(1);
//Enable the interrupt function for module 1.
```

- ***void i8094H_DISABLE_INT(BYTE *cardNo*)**

Description: This function disables the interrupt function.

Parameters:

***cardNo*: Module number**

Return:

None

Example:

```
i8094H_DISABLE_INT(1);
//Disable the interrupt function for module 1.
```

- **void i8094H_INTFACTOR_ENABLE(BYTE *cardNo*, BYTE *axis*, BYTE *nINT*, BYTE *isrNo*)

Description: This function sets the condition factor of interrupt.

Paramenters: *cardNo*: Module number

axis: Axis or Axes (Please refer to Table 2-1 for the axis definition.).

nINT: condition factors of interrupt, please refer to the following table

Number	Symbol	Description
0	PULSE	Interupt occurs when pulse is up
1	P>=C-	Interrupt occurs when the value of logical / real position counter is larger than or equal to that of COMP- register.)
2	P<C-	Interrupt occurs when the value of logical / real position counter is smaller than that of COMP- register.
3	P<C+	Interrupt occurs when the value of logical / real position counter is smaller than that of COMP+ register.
4	P>=C+	Interrupt occurs when the value of logical / real position counter is larger than or equal to that of COMP+ register.
5	C-END	Interrupt occurs at the end of the constant speed drive or completion of Acceleration Offset Pulse output.
6	C-STA	Interrupt occurs at the start of the constant speed drive or begin of Acceleration Offset Pulse output.
7	D-END	Interrupt occurs when the driving is finished

isrNo: ISR1 ~ ISR20 : Specify the number of interrupt service,please refer to section (7.2.1)

Return: None

Example:

```
i8094H_SET_COMPARE(0, AXIS_X, 0, 0, 5000);
i8094H_SET_COMPARE(0, AXIS_Y, 0, 0, 10000);
i8094H_INTFACTOR_ENABLE(1, AXIS_X, 4, ISR1);
i8094H_INTFACTOR_ENABLE(1, AXIS_Y, 4, ISR2);

i8094H_SET_LP(0, AXIS_XY, 0);
i8094H_SET_MAX_V(0, AXIS_XY, 1000000);
i8094H_NORMAL_SPEED(0, AXIS_XY, 0);
//Set the Acc/Dec mode to be symmetric T-curve.
i8094H_SET_V(0, AXIS_XY, 5000);
  //Set the speed of axes X and Y to 5,000 PPS.
i8094H_SET_A(0, AXIS_XY, 10000);
//Set the acceleration of axes X and Y to 10K PPS/S.
i8094H_SET_SV(0, AXIS_XY, 200);
 //Set the start speed of axes X and Y to 200 PPS.
i8094H_SET_AO(0, AXIS_XY, 0);
//set the number of remaining offset pulses for the X and Y axis on
i8094H_FIXED_MOVE(0, AXIS_XY, 20000);
```

```
loop:
    if (i8094H_STOP_WAIT(0, AXIS_XY) == NO) goto loop;
```

Note: Cann't be used with software limit at the same time

- ****void i8094H_INTFACTOR_DISABLE(BYTE *cardNo*, BYTE *axis*, BYTE *nINT*)**

Description: This function disables the condition factor of interrupt.

Paramenters: *cardNo*: Module number
axis: Axis or Axes (Please refer to Table 2-1 for the axis definition.).
nINT: condition factors of interrupt, please refer to the following table

Number	Symbol	Description
0	PULSE	Interupt occurs when pulse is up
1	P>=C-	Interrupt occurs when the value of logical / real position counter is larger than or equal to that of COMP- register.)
2	P<C-	Interrupt occurs when the value of logical / real position counter is smaller than that of COMP- register.
3	P>=C+	Interrupt occurs when the value of logical / real position counter is smaller than that of COMP+ register.
4	P<C+	Interrupt occurs when the value of logical / real position counter is larger than or equal to that of COMP+ register.
5	C-END	Interrupt occurs at the end of the constant speed drive or completion of Acceleration Offset Pulse output.
6	C-STA	Interrupt occurs at the start of the constant speed drive or begin of Acceleration Offset Pulse output.
7	D-END	Interrupt occurs when the driving is finished

Return: None

Example:

```
i8094H_INTFACTOR_DISABLE(1, AXIS_XYZU, 1);
i8094H_INTFACTOR_DISABLE(1, AXIS_XYZU, 2);
i8094H_INTFACTOR_DISABLE(1, AXIS_XYZU, 3);
i8094H_INTFACTOR_DISABLE(1, AXIS_XYZU, 4);
// Disable the factors of Module 1
```

6.3.7 The Interrupt of i-8094H for controller system

- **BYTE i8094H_ENABLE_RINT(BYTE *cardNo*)**

Description: this function enables the interrupt function of i-8094H for controller system.

Parameters: *cardNo*: Module number

Return: 0 : Failer
1 : success

Example: i8094H_ENABLE_RINT(1);
//Enable the 1th interrupt function of i-8094H for controller system

- **void i8094H_DISABLE_RINT(BYTE *cardNo*)**

Description: this function disables the interrupt function of i-8094H for controller system.

Parameters: *cardNo*: Module number

Return: 0 : Failer
1 : success

Example: i8094H_DISABLE_RINT(1);
//Disable the 1th interrupt function of i-8094H for controller system

● **BYTE i8094H_RINT_WAIT(BYTE *cardNo*, BYTE *waitNo*, DWORD *TimeOut*)**

Description: Waiting for returning interrupt status number.

Paramemters: ***cardNo*****: Module number**

waitNo**: sets the condition factors, multiple-choice is allowed, as the following table**

Number	Description
0x01	Line Scan finished，(please refer to the section 6.4.8)
0x02	Macro Program finished.(please refer to section 7.2.2)
0x04	User defined RINT finished.(please refer to section 7.2.8)
0x08	being interrupted when execute a continuous interpolation.
0x10	
0x20	Reserved
0x40	Axes Error，(please use with the i8094H_GET_ERROR_CODE in section 3.6，)
0x80	Module Error，(please use when reset module, please refer to section 2.3)

Example: Choose Axes Error and Module Error (0x40 + 0x80 = 0xC0)

TimeOut**: unit: ms. "0"=disable Time Out**

Return:

0 : Time Out

Value(not zero) : receive the number of interrupt status.

Example:

```
BYTE irec = i8094H_ENABLE_RINT(CardNo);

//Please write the following code in Thread Function。
irec = i8094H_RINT_WAIT(CardNo, 0xc8, 0);
switch (irec)
{
    case 0x08:    AfxMessageBox(
    TEXT("check Interrupt event is Finished (waitNo : 0x08)"), MB_OK, -1 );
                  break;
    case 0x40:    AfxMessageBox(
    TEXT("check Interrupt event is Finished (waitNo : 0x40)"), MB_OK, -1 );
                  break;
    case 0x80:    AfxMessageBox(
    TEXT("check Interrupt event is Finished (waitNo : 0x80)"), MB_OK, -1 );
                  break;
    default:
                  break;
}
AfxMessageBox(_T("ScanTime_Thread STOP !!"));
```

6.4 Continuous Interpolation

6.4.1 2-Axis Rectangular Motion

- ***void** **i8094H_RECTANGLE(**
  BYTE *cardNo*, BYTE *axis1*, BYTE *axis2*,
  BYTE *nAcc*, BYTE *Sp*, BYTE *nDir*, long *Lp*, long *Wp*, long *Rp*,
  DWORD *RSV*, DWORD *RV*, DWORD *RA*, DWORD *RD*)

Description:

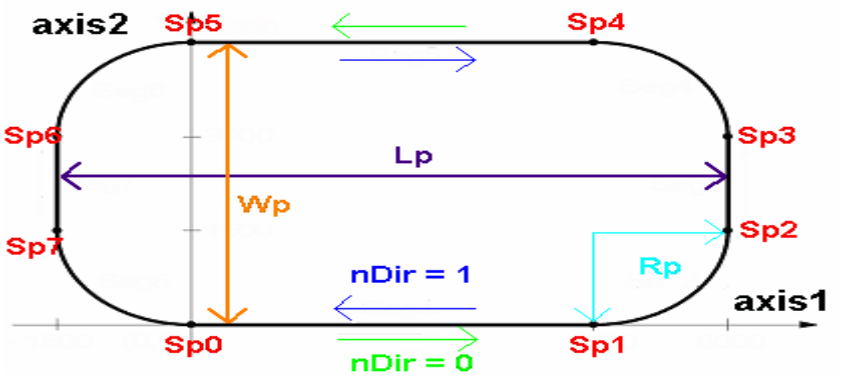
Continuous interpolation will be performed to create a rectangular motion, which is formed by 4 lines and 4 arcs. The length of each side can be changed. The radius of each arc is the same and it can also be changed. The deceleration point will be calculated automatically. This is a command macro command that appears in various motion applications. (It will not consume any WinCon system resources)

Parameters:

***cardNo*:**	**Module number**
***axis1*:**	**The first axis (axis 1). Please refer to Table 2-1. It can be either X, Y, Z, or U(1、2、4、8).**
***axs2*:**	**The second (axis 2). Please refer to Table 2-1. The first axis and It can be either X, Y, Z, or U(1、2、4、8).**
***nAcc*:**	**0 → constant vector speed interpolation mode 1 → symmetric T-curve Acc/Dec interpolation mode**
***Sp*:**	**Start point 0 ~ 7. (Sp0 ~ Sp7 are defined in the following figure)**
***nDir*:**	**Direction of movement 0: CCW; 1: CW**
***Lp*:**	**Length in Pulses (1 ~ 2,000,000,00)**
***Wp*:**	**Width in Pulses (1 ~ 2,000,000,00)**
***Rp*:**	**Radius of each in pulses (1 ~ 2,000,000,00)**
***RSV*:**	**Starting speed (in PPS)**
***RV*:**	**Vector speed (in PPS)**
***RA*:**	**Acceleration (PPS/Sec)**
***RD*:**	**Deceleration of the last segment (in PPS/Sec)**

Return:

YES	**An error has occurred. Use the i8094H_GET_ERROR_CODE() function to identify the error.**
NO	**No errors.**



Example:

```
BYTE cardNo=1; //select module 1.
unsigned short sv=1000;     //starting speed: 1000 PPS.
unsigned short v=10000;     //vector speed: 10000 PPS.
unsigned long a=5000;       //acceleration: 5000 PPS/Sec.
unsigned long d=5000;       //deceleration: 5000 PPS/Sec.
i8094H_SET_MAX_V(1, AXIS_XYZU, 16000);
//set the maximum speed to 16000 PPS.

i8094H_RECTANGLE(1, AXIS_X, AXIS_Y, 1, 0, 0, 20000, 10000, 1000, sv, v, a,
d);
//execute a rectangular motion on the XY plane, will auto-calculate
deceleration point.
```

6.4.2 2-Axis Continuous Linear Interpolation

- ***void i8094H_LINE_2D_INITIAL(BYTE *cardNo*, BYTE *axis1*, BYTE *axis2*,**
  DWORD *VSV* , DWORD *VV* , DWORD *VA*)

Description:

This function sets the necessary parameters for a 2-axis continuous linear interpolation using symmetric T-curve speed profile.

Parameters:

***cardNo*:**	**Module number**
***axis1*:**	**The first axis (axis 1). Please refer to Table 2-1. It can be either X, Y, Z, or U(1、2、4、8).**
***axis2*:**	**The second axis (axis 2). Please refer to Table 2-1. It can be either X, Y, Z, or U(1、2、4、8).**
***VSV*:**	**Starting speed (in PPS)**
***VV*:**	**Vector speed (in PPS)**
***VA*:**	**Vector acceleration (PPS/Sec)**

Return:

None

Example:

```
i8094H_LINE_2D_INITIAL(…);
//This function should be defined before the i8094H_LINE_2D_CONTINUE()
//function is used. Please refer to the example of this function.
```

- ***void i8094H_LINE_2D_CONTINUE(BYTE *cardNo*, BYTE *nType*, long *fp1*, long *fp2*)**

Description:

This function executes a 2-axis continuous linear interpolation
(It will not consume any WinCon system resources.)

Parameters:

***cardNo*:**	**Module number**
***nType*:**	**0: 2-axis linear continuous interpolation** **1: end of 2-axis linear continuous interpolation**
***fp1*:**	**The assigned number of pulses for the axis 1 (in Pulses)** **(-2,000,000,000 ~ +2,000,000,000)**
fp2:	**The assigned number of pulses for the axis 2 (in Pulses)** **(-2,000,000,000 ~ +2,000,000,000)**

Return:

YES	**An error has occurred.** **Use the i8094H_GET_ERROR_CODE () function to identify the error.**
NO	**No errors.**

Example:

```
BYTE cardNo=1;     //select module 1.
unsigned short sv=300;          //starting speed: 300 PPS.
unsigned short v=18000;        //vector speed: 18000 PPS.
unsigned short a=500000; //acceleration: 500000 PPS/Sec.
unsigned short loop1;
i8094H_SET_MAX_V(cardNo, AXIS_XYZU,160000L);
i8094H_LINE_2D_INITIAL(cardNo, AXIS_X, AXIS_Y, sv, v, a);
for (loop1=0; loop1<10000; loop1++)
{
      i8094H_LINE_2D_CONTINUE(cardNo, 0, 100, 100);
      i8094H_LINE_2D_CONTINUE(cardNo, 0, -100, -100);
}
i8094H_LINE_2D_CONTINUE(cardNo, 1, 100, 100);
//
```

6.4.3 3-Axis Continuous Linear Interpolation

- ***void i8094H_LINE_3D_INITIAL(BYTE *cardNo*, BYTE *axis1*, BYTE *axis2*,**
  BYTE *axis3*, DWORD *VSV* , DWORD *VV* , DWORD *VA*)

Description:

This function sets the necessary parameters for a 3-axis continuous linear interpolation using symmetric T-curve speed profile.

Parameters:

***cardNo*:**	**Module number**
***axis1*:**	**The first axis (axis 1). Please refer to Table 2-1. It can be either X, Y, Z, or U (1、2、4、8).**
***axis2*:**	**The second axis (axis 2). Please refer to Table 2-1. It can be either X, Y, Z, or U (1、2、4、8).**
***axis3*:**	**The third axis (axis 3). Please refer to Table 2-1. It can be either X, Y, Z, or U (1、2、4、8).**
***VSV*:**	**Starting speed (in PPS)**
***VV*:**	**Vector speed (in PPS)**
***VA*:**	**Vector acceleration (PPS/Sec)**

Return:

None

Example:

```
i8094H_LINE_3D_INITIAL(...);
//This function should be defined before the i8094H_LINE_3D_CONTINUE()
//function is used. Please refer to the example of this function.
```

- ***void i8094H_LINE_3D_CONTINUE(BYTE *cardNo*, BYTE *nType*, long *fp1*,**
  long *fp2*, long *fp3*)

Description:

This function execute a 3-axis continuous linear interpolation.
(It will not consume any WinCon system resources.)

Parameters:

***cardNo*:**	**Module number**
***nType*:**	**0: 3-axis linear continuous interpolation 1: end of 2-axis linear continuous interpolation**
***fp1*:**	**The assigned number of pulses for axis 1 (-2,000,000,000 ~ +2,000,000,000)**
fp2:	**The assigned number of pulses for axis 2 (-2,000,000,000 ~ +2,000,000,000)**

	fp3:	**The assigned number of pulses for axis 3 (-2,000,000,000 ~ +2,000,000,000)**

Return:

YES	**An error has occurred. Use the i8094MF_GET_ERROR_CODE () function to identify the error.**
NO	**No errors.**

Example:

```
BYTE cardNo=1;   //select module 1.
unsigned short sv=300;          //starting speed: 300 PPS
unsigned short v=18000;         //vector speed: 18000 PPS
unsigned long a=500000;    //acceleration: 500000 PPS/Sec
unsigned short loop1;
i8094H_SET_MAX_V(cardNo, AXIS_XYZU,160000L);
i8094H_LINE_3D_INITIAL(cardNo, AXIS_X, AXIS_Y, sv, v, a);
for (loop1=0; loop1<10000; loop1++)
{
    i8094H_LINE_3D_CONTINUE(cardNo, 0, 100, 100, 100);
    i8094H_LINE_3D_CONTINUE(cardNo, 0, -100, -100, -100);
}
i8094H_LINE_3D_CONTINUE(cardNo, 1, 100, 100, 100);
//execute X, Y, Z 3-axis linear continuous interpolation on module 1
```

6.4.4 Mixed Linear and Circular 2-axis motions in Continuous Interpolation

- ***void** **i8094H_MIX_2D_INITIAL(BYTE *cardNo*, BYTE *axis1*, BYTE *axis2*,**
  BYTE *nAcc*, DWORD *VSV* , DWORD *VV* , DWORD *VA*)

Description:
This function does the initial settings for mixed linear and circular 2-axis motions in continuous interpolation.

Parameters:

***cardNo*:**	**Module number**
***axis1*:**	**The first axis (axis 1). Please refer to Table 2-1. It can be either X, Y, Z, or U(1、2、4、8).**
***axis2*:**	**The second axis (axis 2). Please refer to Table 2-1. It can be either X, Y, Z, or U(1、2、4、8).**
***nAcc*:**	**0 → constant speed (VV) 1 → symmetric T-curve Acc/Dec (VSV、VV、VA)**
***VSV*:**	**Starting speed (in PPS)**
***VV*:**	**Vector speed (in PPS)**
***VA*:**	**Vector acceleration (PPS/Sec)**

Return:
None

Example:

```
i8094H_MIX_2D_INITIAL(...);
//This function should be defined before the i8094H_MIX_2D_CONTINUE()
//function is used. Please refer to the example of this function.
```

- ***void** **i8094H_MIX_2D_CONTINUE(BYTE *cardNo*, BYTE *nAcc*,**
  BYTE *nType*, long *cp1*, long *cp2*, long *fp1*, long *fp2*)

Description:
This function executes mixed linear and circular 2-axis motion in continuous interpolation.
(It will not consume any WinCon system resources.)

Parameters:

***cardNo*:**	**Module number**
***nAcc*:**	**0 → continuous interpolation. 1 → it is the last command of this continuous interpolation. In Acc/Dec mode, it will perform a deceleration stop. In constant speed mode, it will directly stop rather than decelerate.**

***nType*:**

1 →i8094H_LINE_2D(BYTE *cardNo*, long *fp1*, long *fp2*);
2 →i8094H_ARC_CW(BYTE *cardNo*, long *cp1*, long *cp2*, long *fp1*, long *fp2*);
3 →i8094H_ARC_CCW(BYTE *cardNo*, long *cp1*, long *cp2*, long *fp1*, long *fp2*);
4 →i8094H_CIRCLE_CW(BYTE *cardNo*, long *cp1*, long *cp2*);
5 →i8094H_CIRCLE_CCW(BYTE *cardNo*, long *cp1*, long *cp2*);

***cp1*:**	**It assigns the center point data at axis 1. (-2,000,000,000 ~ +2,000,000,000)**
***cp2*:**	**It assigns the center point data at axis 2. (-2,000,000,000 ~ +2,000,000,000)**
***fp1*:**	**It assigned number of pulses for axis 1 (-2,000,000,000 ~ +2,000,000,000)**
***fp2*:**	**It assigned number of pulses for axis 2. (-2,000,000,000 ~ +2,000,000,000)**

Return:

YES	**An error has occurred. Use the i8094H_GET_ERROR_CODE () function to identify the error.**
NO	**No errors.**

Example:

```
BYTE cardNo=1;   //select module 1.
unsigned short sv=300;          //starting speed: 300 PPS
unsigned short v=18000;         //vector speed: 18000 PPS
unsigned long a=500000L;  //acceleration: 500000 PPS/Sec
unsigned short loop1;
i8094H_SET_MAX_V(cardNo, AXIS_XYZU, 160000L);
i8094H_MIX_2D_INITIAL(cardNo, AXIS_X, AXIS_Y, 1, sv, v, a);
for (loop1 = 0; loop1 < 10000; loop1++)
{
    i8094H_MIX_2D_CONTINUE (cardNo, 0, 1, 0, 0, 100, 100);
    i8094H_MIX_2D_CONTINUE (cardNo, 0, 2, 100, 0, 100, 100);
}
i8094H_MIX_2D_CONTINUE (cardNo, 1, 4, 100, 100, 0, 0);
//execute X, Y, 2-axis motions in continuous interpolation on module 1
```

6.4.5 Multi-Segment Continuous Interpolation (Using Array)

- **void i8094H_CONTINUE_INTP(**
  BYTE *cardNo*, BYTE *axis1*, BYTE *axis2*, BYTE *axis3*,
  BYTE *nAcc*, DWORD *VSV*, DWORD *VV*, DWORD *VA*, DWORD *VD*)

Description:

This function executes a multi-segment continuous interpolation(symmetric T-curve). Please write the motion data arrays first. (It will not consume any WinCon system resources.)

Parameters:

***cardNo*:**	**Module number**
***axis1*:**	**The first axis (axis 1). Can be either X, Y, Z, or U axis(1、2、4、8). Please refer to Table 2-1 for the axis definition.**
***axis2*:**	**The second axis (axis 2). Can be either X, Y, Z, or U axis(1、2、4、8).**
***axis3*:**	**The third axis (axis 3). Can be either X, Y, Z, or U axis(1、2、4、8).**
***nAcc*:**	**0 → a constant speed interpolation. Please set VV. 1 → a symmetric T-curve interpolation. Please set VSV, VV, VA, and VD.**
***VSV*:**	**The starting speed (in PPS)**
***VV*:**	**Interpolation vector speed (in PPS)**
***VA*:**	**Acceleration (in PPS/Sec)**
***VD*:**	**Deceleration (in PPS/Sec)**

Return:

YES	**An error has occurred. Use the i8094H_GET_ERROR_CODE () function to identify the error.**
NO	**No errors.**

Example:

```
BYTE cardNo=1; //select module 1.
unsigned short sv=100; //set the starting speed to 100 PPS.
unsigned short v=3000; //set the speed to 3000 PPS.
unsigned long a=2000; //set the acceleration to 2000 PPS/Sec.
unsigned long d=2000; //set the deceleration to 2000 PPS/Sec.
i8094H_SET_MAX_V(cardNo, AXIS_XYZU, 20000);
//set the maximum speed to 20K PPS.
BYTE nType[10]= {     1,     2,     1,     2,     1,7,0,0,0,0};
long cp1[10]=   {     0, 10000,     0,     0,     0,0,0,0,0,0};
long cp2[10]=   {     0,     0,     0,-10000,     0,0,0,0,0,0};
long fp1[10]=   { 10000, 10000,  1000, 10000,-31000,0,0,0,0,0};
long fp2[10]=   { 10000, 10000,     0,-10000,-10000,0,0,0,0,0};
long fp3[10]=   {     0,     0,     0,     0,     0,0,0,0,0,0};
//put data of the required segments in arrays.
i8094H_CONTINUE_INTP_ARRAY(
```

```
                                        cardNo, nType, cp1, cp2, fp1, fp2, fp3);
//=======================================================
i8094H_CONTIUNE_INTP(cardNo, AXIS_X, AXIS_Y, 0, 1, sv, v, a, d);
// Select module 1, execute the multi-segment continuous interpolation.
//The deceleration point will be calculated automatically.
//For this example, the final position of this motion will return to the starting
point.
```

- **void i8094H_CONTINUE_INTP_ARRAY(**
  BYTE *cardNo*, BYTE *nType[ ]*, long *cp1[ ]*, long *cp2[ ]*,
  long *fp1[ ]*, long *fp2[ ]*, long *fp3[ ]*)

Description:

This function executes a multi-segment continuous interpolation.

Parameters:

***cardNo*: Module number**

***nType[ ]*: Maximum segment: 1024 (0 ~ 1023). It contains the interpolation commands defined as follows.**

1 →i8094H_LINE_2D(BYTE *cardNo*, long *fp1*, long *fp2*);
2 →i8094H_ARC_CW(BYTE *cardNo*, long *cp1*, long *cp2*, long *fp1*, long *fp2*);
3 →i8094H_ARC_CCW(BYTE *cardNo*, long *cp1*, long *cp2*, long *fp1*, long *fp2*);
4 →i8094H_CIRCLE_CW(BYTE *cardNo*, long *cp1*, long *cp2*);
5 →i8094H_CIRCLE_CCW(BYTE *cardNo*, long *cp1*, long *cp2*);
6 →i8094H_LINE_3D(BYTE *cardNo*, long *fp1*, long *fp2*, long *fp3*);
7 → It indicates the end of continuous interpolation.

***cp1[ ]*: It contains a list of segment center point data at axis 1. (-2,000,000,000 ~ +2,000,000,000)**

***cp2[ ]*: It contains a list of segment center point data at axis 2. (-2,000,000,000 ~ +2,000,000,000)**

***fp1[ ]*: This array contains a list of segment end point data at axis 1. (-2,000,000,000 ~ +2,000,000,000)**

***fp2[ ]*: This array contains a list of segment end point data at axis 2. (-2,000,000,000 ~ +2,000,000,000)**

***fp3[ ]*: This array contains a list of segment end point data at axis 3. (-2,000,000,000 ~ +2,000,000,000)**

(Note: The 2-axis and 3-axis motion commands can not be mixed together when applying commands. Please fill 0 for the cell values in the array if these cells are not used.)

Return:

None

Example:

Please refer to example for i8094H_CONTINUE_INTP(in previous section).

6.4.6 3-Axis Helical Motion

- ***void i8094H_HELIX_3D(**
  BYTE *cardNo*, BYTE *axis1*, BYTE *axis2*, BYTE *axis3*, BYTE *nDir*,
  DWORD *VV* , long *cp1*, long *cp2*, long *cycle*, long *pitch*)

Description:

This function performs a 3-axis helical motion. However, it is a software macro-function; therefore, it requires CPU resource to run this function. (It will not consume any WinCon system resources.)

Parameters:

***cardNo*:**	**Module number**
***axis1*:**	**The first axis (axis 1). Can be X, Y, Z, or U axis. Please refer to Table 2-1 for the axis definition.**
***Axis2*:**	**The second axis (axis 2). Can be either X, Y, Z, or U axis.**
***Axis3*:**	**The third axis (axis 3). Can be either X, Y, Z, or U axis.**
***nDir*:**	**0 → Move in a CW direction. 1 → Move in a CCW direction.**
***VV*:**	**Vector speed (in PPS)**
***cp1*:**	**The value of center at axis 1 (-2,000,000,000 ~ +2,000,000,000)**
***cp2*:**	**The value of center at axis 2 (-2,000,000,000 ~ +2,000,000,000)**
***cycle*:**	**Number of cycles (-2,000,000,000 ~ +2,000,000,000)**
***pitch*:**	**Pitch per revolution (the advanced distance for each revolution) (-2,000,000,000 ~ +2,000,000,000)**

Return:

YES	**An error has occurred. Use the i8094H_GET_ERROR_CODE () function to identify the error.**
NO	**No errors.**

Example:

```
BYTE cardNo=1; //select module 1.

//================================================
i8094H_SET_MAX_V(cardNo, AXIS_XYZU,160000L);
//set maximum speed for all axes to 16K PPS.
long v=50000;
//set vector speed to 50K PPS.
i8094H_HELIX_3D(cardNo, AXIS_Y, AXIS_Z, AXIS_X, 1, v, 0, 1000, 5, -2000);
//the circular motion is on YZ plane, and the linear motion is
//along the X axis.

//================================================
i8094H_SET_MAX_V(cardNo, AXIS_XYZU, 160000L);
//set the maximum speed for all axes to 160K PPS.
long v=100000L;
//set vector speed to 100K PPS.
i8094H_HELIX_3D(cardNo, AXIS_Y, AXIS_Z, AXIS_U, 1, v, 0, 25000, 50, 3600);
//the circular motion is on YZ plane, and the linear motion is along
//the U axis.
```

6.4.7 2-Axis Ratio Motion

- ***void** **i8094H_RATIO_INITIAL(BYTE *cardNo*, BYTE *axis1*, BYTE *axis2*,**
  DWORD *SV* , DWORD *V* , DWORD *A*, BYTE *CMX*, BYTE *CDV*)

Description:

This function sets the Initial values for ratio motion (motion in ratio) using a symmetric T-curve speed profile.

Parameters:

***cardNo*:**	**Module number**
***axis1*:**	**The first axis (axis 1). Can be either X, Y, Z, or U axis (1,2,4,8). Please refer to Table 2-1 for the axis definition.**
***Axis2*:**	**The second axis (axis 2). Can be either X, Y, Z, or U axis(1,2,4,8).**
***SV*:**	**Set the value for the starting speed (in PPS).**
***V*:**	**Set the value for the vector speed (in PPS).**
***A*:**	**Set the acceleration value (in PPS/Sec).**
***CMX*:**	**Set the molecule value between the two assigned axes(1 ~ +127).**
***CDV*:**	**Set the denominator value between the two assigned axes(1 ~ +127).**

Return:

None

Example:

```
i8094H_RATIO_INITIAL(…);
//Initial setting for i8094H_RATIO_2D(…) function.
//Please refer to the example of i8094H_RATIO_2D() function.
```

- ***void** **i8094H_RATIO_2D(BYTE *cardNo*, BYTE *nType*, long *data*,**
  BYTE *nDir*)

Description:

This function performs a two-axis ratio motion.
(It will not consume any WinCon system resources.)

Parameters:

***cardNo*:**	**Module number**
***nType*:**	**0 → Perform the ratio motion.**
	1 → Declare the end of ratio motion.
***data*:**	**The pulse number of axis1**

(-2,000,000,000 ~ +2,000,000,000)

nDir: Direction of the second axis.
0: CW; 1: CCW

Return:

YES	An error has occurred. Use the i8094H_GET_ERROR_CODE () function to identify the error.
NO	No errors.

Example:

```
BYTE cardNo=1; //select module 1.
unsigned short sv=300; //set starting speed to 300 PPS.
unsigned short v=18000; //set vector speed to 18000 PPS.
unsigned long a=500000; //set acceleration value to 500K PPS/Sec.
unsigned short loop1;
unsigned short loop2;
i8094H_SET_MAX_V(cardNo, 0 AXIS_XYZU,160000L);
//set maximum speed value to 18000 PPS.
i8094H_RATIO_INITIAL(cardNo,AXIS_U, AXIS_X, sv, v, a, 9,25);
//assign U axis as the axis 1 and X axis as the axis 2.
for (loop2 = 0; loop2 < 5; loop2++)
{
    for (loop1 = 0; loop1 < 5; loop1++)
    {
        i8094H_RATIO_2D(cardNo, 0, 3600, 0);
        //perform the ratio motion in the CW direction.
        i8094H_RATIO_2D(cardNo, 0, 3600, 1);
        //perform the ratio motion in the CCW direction.
    }
    i8094H_RATIO_2D(cardNo, 0, 7200, 0);
    i8094H_RATIO_2D(cardNo, 0, 3600, 1);
}
i8094H_RATIO_2D(cardNo, 1, 7200, 0);
 //End the ratio motion.
```

6.4.8 Synchronous Line Scan Motion

- **void i8094H_LINE_SCAN(BYTE *cardNo*, BYTE *axis*, BYTE *Type*,**
  BYTE *outEdge*, BYTE *PulseWidth*, long *Pitch*)

Description:

Equal distance Line Scan trigger out : Max speed < 100KHz (Pulse Width 10 uS).

Unequal distance Line Scan trigger out : Max speed < 18KHz.

Parameters:

***cardNo*:**	**Module number**	
***axis*:**	**Axis number:**	**equal distance X or Y (1,2)**
		unequal distance X(1)
***Type*:**	**0 = equal distance motion**	**(please refer to C+，LP)**
	1 = unequal distance motion	**(please refer to C+，LP)**
	2 = equal distance motion	**(please refer to C+，EP)**
	3 = unequal distance motion	**(please refer to C+，EP)**
***outEdge*:**	**trigger OUT active logic:**	
	0 = low active; 1 = high active	
***PulseWidth*:**	**trigger OUT output pulse width**	
	0 = 10 uSec	
	1 = 20 uSec	
	2 = 100 uSec	
	3 = 200 uSec	
	4 = 1,000 uSec	
	5 = 2,000 uSec	
	6 = 10,000 uSec	
	7 = 20,000 uSec	
***Pitch*:**	**Specify interval pulses (-2,000,000,000 ~ +2,000,000,000)**	

Return:

None

Example:

```
long lp1, lp2, lp3, lp4;
BYTE irec;
i8094H_SET_MAX_V(CardNo, AXIS_XY, 4000000);
//set the maximum speed of X and Y axes to 4M PPS.
i8094H_NORMAL_SPEED(CardNo, AXIS_XY, 0);
//set the driving mode to be symmetric T-curve.
i8094H_SET_V(CardNo, AXIS_XY, 2000000);
//set the maximum speed of X and Y axes to 2M PPS.
i8094H_SET_A(CardNo, AXIS_XY, 20000000);
//set the acceleration of X and Y axes to 2M PPS/S.
i8094H_SET_SV(CardNo, AXIS_XY, 2000000);
```

```
//set the starting speed of X and Y axes to 2M PPS.
i8094H_LINE_SCAN(CardNo, AXIS_X, 0, 0, 0, -39);
i8094H_LINE_SCAN(CardNo, AXIS_Y, 0, 0, 0, -79);
i8094H_LINE_SCAN_START(CardNo, AXIS_XY, 0, -4000000);

irec = i8094H_RINT_WAIT(CardNo, 0x01, 10000);
lp1 = i8094H_GET_LP(CardNo, AXIS_X);
lp2 = i8094H_GET_LP(CardNo, AXIS_Y);
```

- **void i8094H_LINE_SCAN_START(BYTE *cardNo*, BYTE *axis*, BYTE *Type*, long *Position*)**

Description:

Enable Line Scan trigger out motion.

When performing unequal distance motion, please note (there is no restriction for equal distance motion):

a. All ISR will be stopped.

b. Do not execute the following commands:

i8094H_READ_bVAR
i8094H_READ_VAR
i8094H_READ_FLONG
i8094H_GET_LP
i8094H_GET_EP
i8094H_GET_CV
i8094H_GET_CA
i8094H_GET_CVAR
i8094H_GET_DI
i8094H_GET_ERROR
i8094H_GET_ERROR_CODE
i8094H_FRNET_IN
i8094H_HOME_WAIT
i8094H_GET_LATCH
i8094H_STOP_WAIT
i8094H_CLEAR_STOP

Parameters:

cardNo:	Module number
axis:	Axis number: equal distance X or Y (1,2) unequal distance X(1)
Type:	0 = equal distance motion (please refer to C+，LP) 1 = unequal distance motion (please refer to C+，LP, it require to accordingly adjust i8094H_LINE_SCAN_OFFSET) 2 = equal distance motion (please refer to C+，EP, it require to accordingly adjust i8094H_LINE_SCAN_OFFSET) 3 = unequal distance motion (please refer to C+，EP)
Position:	Total pulses, movement distance(-2,000,000,000 ~ +2,000,000,000)

Return:

None

Example:

Please refer to examples in above section.

- **void i8094H_LINE_SCAN_OFFSET(BYTE *cardNo*, DWORD *Total*, char *Offset[ ]*)**

Description:

Set unequal distance interval offset pulses.

Parameters:

cardNo: Module number

Return:

None

Example:

```
long lp1;
BYTE irec;
i8094H_SET_MAX_V(CardNo, AXIS_X, 4000000);
//set the maximum speed of X-axis to be 4M PPS.
i8094H_NORMAL_SPEED(CardNo, AXIS_X, 0);
//set the driving mode of X-axis to be symmetric T-curve.
i8094H_SET_V(CardNo, AXIS_X, 2000000);
//set the maximum speed of X-axis to be 2M PPS.
i8094H_SET_A(CardNo, AXIS_X, 20000000);
//set the acceleration of X-axis to be 2M PPS/S.
i8094H_SET_SV(CardNo, AXIS_X, 2000000);
//set the starting speed of X-axis to 2M PPS.

char offset[40960];
for (lp1 = 0; lp1 < 40960; lp1++)
{offset[lp1] = 0;}
i8094H_LINE_SCAN(CardNo, AXIS_X, 1, 0, 1, 105);
i8094H_LINE_SCAN_OFFSET(CardNo, 40000, offset);
i8094H_LINE_SCAN_START(CardNo, AXIS_X, 1, 4200000);

irec = i8094H_RINT_WAIT(CardNo, 0x01, 10000);
lp1 = i8094H_GET_LP(CardNo, AXIS_X);
```

6.4.9 3-Axis Arc Interpolation

- **long i8094H_ARC_3D(BYTE *cardNo*, BYTE *axis1*, BYTE *axis2*, BYTE *axis3*, long **P1*, long **P2*, DWORD *DR*, DWORD *VV*)**

Description: Triple points to scheme a 3D circle, the start point of the circle:(0, 0, 0), the second point of the circle:(P1[0], P1[1], P1[2]), and end point of the circle:(P2[0], P2[1], P2[2]), using the 3-axes linear interpolation to approximate.(fixed speed)

Parameters:

cardNo: Module number

axis1: The first axis; can be either X, Y, Z, or U(1、2、4、8). Please refer to Table 2-1 for the axis definition.

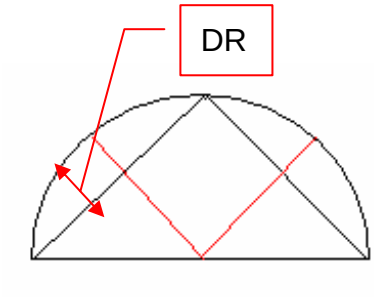
axis2: The second axis; can be either X, Y, Z, or U(1、2、4、8).

axis3: The third axis; can be either X, Y, Z, or U(1、2、4、8).

P1: The relative position of the center to the current circle: (P1[0]_axis1，P1[1]_axis2，P1[2]_axis3)

P2: The relative position of the end to the current circle: (P2[0]_axis1，P2[1]_axis2，P2[2]_axis3)

DR: set the error value for linear approximation (+1 ~ +2,000,000,000 Pulse)



VV: Set the vector speed(PPS)

Return:

< 0 (negative) vector speed exceed the limit, will return the value of the circular maximum vector speed(PPS)

> 0 (postive) the settings are within the range, will return the time interval(ms) for performing circular 3-axis motions

If an error occurrs, execute the i8094H_GET_ERROR_CODE () function to identify the error.

Example:

```
long P1[3], P2[3];
P1[0] = -10000;        P1[1] = 10000;        P1[2] = 5000;
P2[0] = -10000;        P2[1] = -10000;       P2[2] = 10000;
long rec = i8094H_ARC_3D(
                1, AXIS_X, AXIS_Y, AXIS_Z, P1, P2, 5, 1000000);
//DR=5 Pulse、VV=1,000,000 PPS，rec return value: +45 (ms)
long rec = i8094H_ARC_3D(
                1, AXIS_X, AXIS_Y, AXIS_Z, P1, P2, P3, 1, 1000000);
//DR=1 Pulse、VV=1,000,000，rec return value: -724397
//It means the module executing with VV=724,397 PPS
```

6.4.10 Mixed 3-axis motions in Continuous Interpolation

- **void i8094H_MIX_3D_INITIAL(BYTE *cardNo*, BYTE *axis1*, BYTE *axis2*, BYTE *axis3*, DWORD *VSV* , DWORD *VV* , DWORD *VA*)**

Description:

This function performs initial settings for linear and circular 3-axis motions in continuous interpolation and symmetric T-curve Acc/Dec interpolation mode.

Parameters:

***cardNo*:**	**Module number**
***axis1*:**	**The first axis (axis 1). Please refer to Table 2-1. It can be either X, Y, Z, or U(1、2、4、8).**
***axis2*:**	**The second axis (axis 2). Please refer to Table 2-1. It can be either X, Y, Z, or U(1、2、4、8).**
***axis3*:**	**The second axis (axis 2). Please refer to Table 2-1. It can be either X, Y, Z, or U(1、2、4、8).**
***VSV*:**	**Starting speed (in PPS)**
***VV*:**	**Vector speed (in PPS)**
***VA*:**	**Vector acceleration (PPS/Sec)**

Return:

None

Example:

This function should be used along with function: long i8094H_MIX_3D_CONTINUE(BYTE *cardNo*, BYTE *nAcc*, BYTE *nType*, long **P1*, long **P2*, DWORD *DR*)

- **long i8094H_MIX_3D_CONTINUE(BYTE *cardNo*, BYTE *nAcc*, BYTE *nType*, long **P1*, long **P2*, DWORD *DR*)**

Description:

This function executes linear and circular 3-axis motions in continuous interpolation.

Parameters:

***cardNo*:**	**Module number**
The first axis and	**It can be either X, Y, Z, or U(1、2、4、8).**

nAcc:	**0 → continuous interpolation** **1 → slowdown to stop interpolation motion (if it is moving in constant speed, it does not need to slowdown for a stop)**
nType:	**0 →i8094H_LINE_3D(*cardNo*, **P1*)** **1 →i8094H_ARC_3D(*cardNo*, **P1*, **P2*, *DR*)**
P1:	**3-axis linear relative position:** **(P1[0]_axis1，P1[1]_axis2，P1[2]_axis3)** **Circular midpoint relative position:** **(P1[0]_axis1，P1[1]_axis2，P1[2]_axis3)**
P2:	**Circular endpoint relative position:** **(P2[0]_axis1，P2[1]_axis2，P2[2]_axis3)**
DR:	**linear approximation of curve errors (+1 ~ +2,000,000,000 Pulse)**

Return:

< 0 (negative) vector speed exceed the limit, will return the value of the circular maximum vector speed(PPS)
> 0 (postive) the settings are within the range, will return the time interval(ms) for performing circular 3-axis motions
If an error occurrs, execute the i8094H_GET_ERROR_CODE () function to identify the error.

Example:

```
i8094H_MIX_3D_INITIAL(
CardNo, AXIS_X, AXIS_Y, AXIS_Z, 100000 , 100000, 100000);
long P1[3], P2[3];
long rec1, rec2, rec3, rec4;
long lp1, lp2, lp3, lp4;
DWORD t0, t1, t;
P1[0] = 10000;      P1[1] = 10000;      P1[2] = 10000;
rec1 = i8094H_MIX_3D_CONTINUE(CardNo, 0, 0, P1, P2, 1);

// max RTC speed = 5KHz
P1[0] = -10000;          P1[1] = 10000;      P1[2] = 5000;
P2[0] = -10000;          P2[1] = -10000;     P2[2] = 10000;
rec2 = i8094H_MIX_3D_CONTINUE(CardNo, 0, 1, P1, P2, 1);

// max RTC speed = 5KHz
P1[0] = -10000;          P1[1] = 10000;      P1[2] = 5000;
P2[0] = -10000;          P2[1] = -10000;     P2[2] = 10000;
rec3 = i8094H_MIX_3D_CONTINUE(CardNo, 0, 1, P1, P2, 1);

P1[0] = -10000;          P1[1] = -10000;          P1[2] = -10000;
```

```
rec4 = i8094H_MIX_3D_CONTINUE(CardNo, 1, 0, P1, P2, 1);



loop:
//Sleep(100);
if (i8094H_STOP_WAIT(CardNo, AXIS_XYZU) == NO) goto loop;

lp1 = i8094H_GET_LP(CardNo, AXIS_X);
lp2 = i8094H_GET_LP(CardNo, AXIS_Y);
lp3 = i8094H_GET_LP(CardNo, AXIS_Z);
lp4 = i8094H_GET_LP(CardNo, AXIS_U);
```

6.5 Other Functions

6.5.1 Holding the Driving Command

- ****void i8094H_DRV_HOLD(BYTE *cardNo*, BYTE *axis*)**

Description:

This command is usually used when users desire to start multi-axis driving simultaneously. When this command is issued, users may write other driving commands to the control card. All the driving commands will be held after i8094H_DRV_HOLD() is issued, and these commands will be started once the i8094H_DRV_START() is issued. However, if in driving, this command will not cause the driving to be stopped. But the next command will be held.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or Axes (Please refer to Table 2-1 for the axis definition.)**

Return:

None

Example:

Please refer to the example in section 6.5.2.

6.5.2 Release the Holding Status, and Start the Driving

- ****void i8094H_DRV_START(BYTE *cardNo*, BYTE *axis*)**

Description:

This command releases the holding status, and start the driving of the assigned axes immediately.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or Axes (Please refer to Table 2-1 for the axis definition.)**

Return:

None

Example:

```
BYTE cardNo=1; //select card 1.
i8094H_DRV_HOLD(cardNo, AXIS_XYU); //hold the driving command to XYU
```

```
i8094H_SET_MAX_V(cardNo, AXIS_U, 10000);
//set the maximum speed of U-axis to be 10K PPS.
i8094H_NORMAL_SPEED(cardNo, AXIS_U, 0);
//set the driving mode to be symmetric T-curve.
i8094H_SET_V(cardNo, AXIS_U, 2000);
//set the speed of U-axis to 2,000 PPS.
i8094H_SET_A(cardNo, AXIS_U, 1000);
//set the acceleration of U-axis to 1,000 PPS/S.
i8094H_SET_SV(cardNo, AXIS_U, 2000);
//set the starting speed to 2,000 PPS.
i8094H_SET_AO(cardNo, AXIS_U, 9);
// set the AO to 9 Pulses.
i8094H_SET_MAX_V(cardNo, AXIS_XY, 20000);
//set the maximum speed of X and Y axes to 20K PPS.
i8094H_AXIS_ASSIGN(cardNo, AXIS_X, AXIS_Y, 0);
//set the X-axis as the axis 1 and Y-axis as the axis 2 for a 2-axis interpolation.
i8094H_VECTOR_SPEED(cardNo, 0);
//set constant speed motion.   Therefore, VSV=VV. Only VV is required.
i8094H_SET_VV(cardNo, 5000);
//set the vector speed for card 1 to 5,000 PPS.
i8094H_FIXED_MOVE(cardNo, AXIS_U, 5000);
//command U-axis to move 5,000 Pulse.   This command is be held.
i8094H_LINE_2D(cardNo, 12000, 10000);
//command a linear interpolation motion on the XY planes. It is held, too.
i8094MF_DRV_START(cardNo, AXIS_XYU);
//release the holding status. X,Y , and U axes will start to move
simultaneously.
```

6.5.3 Waiting until the Motion Is Completed

- **BYTE i8094H_STOP_WAIT(BYTE *cardNo*, BYTE *axis*)**

Description:

This function can be used to assign commands to be performed while waiting for all motion to be completed (stopped).

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**

Return:

YES	**Motion is complete**
NO	**Motion is not complete**

EXAMPLE:

```
BYTE cardNo=1; //select module 1
i8094H_SET_MAX_V(cardNo, AXIS_XYZU, 20000);
//set the maximum speed of all axes on module 1 to 20K PPS.
i8094H_NORMAL_SPEED(cardNo, AXIS_XYZU, 0);
//set the speed profile of all axes on module 1 to be symmetric T-curve
i8094H_SET_V(cardNo, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to 2000 PPS.
i8094H_SET_A(cardNo, AXIS_XYZU,1000);
//set the acceleration value of all axes on module 1 to 1000 PPS/S.
i8094H_SET_SV(cardNo, AXIS_XYZU, 2000);
//set the start velocity of all axes on module 1 to 2000 PPS.
i8094H_SET_AO(cardNo, AXIS_XYZU, 9);
//set the value of remaining offset pulses to 9 pulses.
i8094H_FIXED_MOVE(cardNo, AXIS_XYZU, 10000);
// move all axes on module 1 for 10000 pulses.

if (i8094H_STOP_WAIT(cardNo, AXIS_X) == NO)
{
    //perform some actions here if the X axis has not finished its
    //motion.
}
```

6.5.4 Stopping the Axes

- ***void i8094H_STOP_SLOWLY(BYTE *cardNo*, BYTE *axis*)**

Description:

This function commands to decelerate and finally stops the assigned axes slowly.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**

Return:

None

Example:

```
i8094H_STOP_SLOWLY(1, AXIS_XY);
//decelerate and stop the X and Y axes
```

- ***void i8094H_STOP_SUDDENLY(BYTE *cardNo*, BYTE *axis*)**

Description:

This function commands to immediately stop the assigned axes.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**

Return:

None

Example:

```
i8094H_STOP_SUDDENLY(1, AXIS_ZU);
//immediately stop the Z and U axes.
```

- **void i8094H_VSTOP_SLOWLY(BYTE *cardNo*)**

Description:
This function commands to stop interpolation motion of the assigned module in a decelerating way.

Parameters:
***cardNo*: Module number**

Return:
None

Example:
```
i8094H_VSTOP_SLOWLY(1);
//stop the interpolation of card 1 in a decelerating way.
```

- **void i8094H_VSTOP_SUDDENLY(BYTE *cardNo*)**

Description:
This function commands to stop interpolation motion of the assigned module immediately.

Parameters:
***cardNo*: Module number**

Return:
None

Example:
```
i8094H_VSTOP_SUDDENLY(1);
// stop the interpolation of card 1 immediately.
```

6.5.5 Clear the Stop Status

- ***void** **i8094H_CLEAR_STOP(BYTE *cardNo*)**

Description:

After using anyone of the stop functions mentioned in section 6.5.4, please solve the malfunction, and then issue this function to clear the stop status.

Paramters:

***cardNo*:** **Module number**

Return:

None

Example:

```
i8094H_VSTOP_SUDDENLY(1);
//command the card 1 to stop motion immediately.
i8094H_CLEAR_STOP(1);
//clear the error status of card 1.
```

6.5.6 End of Interpolation

- ***void** **i8094H_INTP_END(BYTE *cardNo*, BYTE *type*)**

Description:

- **If the current motion status is running a interpolation motion and you would like to issue a single axis motion or change the coordinate definition, you should call this function before the new command is issued.**
- **You can redefine the MAX_V for each axis. In this way, you do not have to execute i8094H_INTP_END() function.**

Parameters:

***cardNo*:** **Module number**
***type*:** **0 → 2-axis interpolation**
1 → 3-axis interpolation

Return:

None

Example:

```
i8094H_INTP_END(1, 0); //declear the end of a 2-axis interpolation on card 1.
```

7 Additional Functions Supported by i8094H

- **Default Settings for i8094H when system boots:**

  i8094H_SET_PULSE_MODE(*cardNo*, AXIS_XYZU, 0);
  i8094H_SET_MAX_V(*cardNo*, AXIS_XYZU, 2000000L);
  i8094H_SET_HLMT(*cardNo*, AXIS_XYZU, 0, 0);
  i8094H_LIMITSTOP_MODE(*cardNo*, AXIS_XYZU, 0);
  i8094H_SET_NHOME(*cardNo*, AXIS_XYZU, 0);
  i8094H_SET_HOME_EDGE(*cardNo*, AXIS_XYZU, 0);
  i8094H_CLEAR_SLMT(*cardNo*, AXIS_XYZU);
  i8094H_SET_ENCODER(*cardNo*, AXIS_XYZU, 0, 0, 0);
  i8094H_SERVO_OFF(*cardNo*, AXIS_XYZU);
  i8094H_SET_ALARM(*cardNo*, AXIS_XYZU, 0, 0);
  i8094H_SET_INPOS(*cardNo*, AXIS_XYZU, 0, 0);
  i8094H_SET_FILTER(*cardNo*, AXIS_XYZU, 0, 0);
  i8094H_VRING_DISABLE(*cardNo*, AXIS_XYZU);
  i8094H_AVTRI_DISABLE(*cardNo*, AXIS_XYZU);
  i8094H_EXD_DISABLE(*cardNo*, AXIS_XYZU);

- **Functions could be used in downloading parameter table (please refer to 7.1):**
  i8094H_SET_PULSE_MODE
  i8094H_SET_MAX_V
  i8094H_SET_HLMT
  i8094H_LIMITSTOP_MODE
  i8094H_SET_NHOME
  i8094H_SET_HOME_EDGE
  i8094H_SET_SLMT (i8094H_CLEAR_SLMT)
  i8094H_SET_ENCODER
  i8094H_SERVO_ON (i8094H_SERVO_OFF)
  i8094H_SET_ALARM
  i8094H_SET_INPOS
  i8094H_SET_FILTER
  i8094H_ VRING_ENABLE (i8094H_ VRING_DISABLE)
  i8094H_ AVTRI_ENABLE (i8094H_ AVTRI_DISABLE)

- **Set the return value of the functions(please refer to 7.2.3) to variables:**

  i8094H_GET_LP

  i8094H_GET_EP

  i8094H_GET_DI

  i8094H_GET_DI_ALL

  i8094H_GET_ERROR

  i8094H_GET_ERROR_CODE

  i8094H_FRNET_DI

  i8094H_HOME_WAIT

  i8094H_GET_LATCH

  i8094H_STOP_WAIT

7.1 Initial Parameter Table

There two ways to download the parameter table:

- **Please use i8094H_Tool to download parameter table into i8094H. For detail information, please refer to manual "i8094H_Getting_Started_1.1tc.pdf".**

- **void i8094H_LOAD_INITIAL(BYTE *cardNo*)**

Description:

Another way is to write the settings into i8094H parameter table (please refer to table 2-1a) by functions; and then load the initial setting values into i8094H by i8094H_LOAD_INITIAL.

Parameters:

***cardNo*: Module number**

Return:

None

Example:

```
i8094H_SET_PULSE_MODE(1, INITIAL_XYZU, 0);
        //set the pulse mode of X, Y, Z, and U axes as 0, write into the parameter table
        in module 1.

        i8094H_SET_MAX_V(1, INITIAL_XY, 200000L);
        //The maximum speed for the X and Y axes of module 1 is 200KPPS. Write into
        //the parameter table.

        i8094H_SET_HLMT(1, INITIAL_XYZU, 0, 0);
        //set all the trigger levels as low-active for all limit switches
        //on module 1. Write into the parameter table.

        i8094H_LIMITSTOP_MODE(1, INITIAL_X, 0);
        //set X axis to stop immediately if any limit switch on X axis is triggered on
        //module 1. Write into the parameter table.


        i8094H_SET_NHOME(1, INITIAL_XY, 0);
        //set the trigger level of NHOME of X and Y axes on module 1 to be active low.
        //Write into the parameter table.

        i8094H_ LOAD_INITIAL(1);
        // load the initial setting values of the parameter table into i8094H
```

7.2 Create Macro Program(MP)

7.2.1 Create Marco Program Codes

- **void i8094H_MP_CREATE(BYTE *cardNo*, BYTE *mpNo*)**
- **void i8094H_MP_CREATE(BYTE *cardNo*, BYTE *isrNo*)**

Description:

Write in Macro Program codes into i8094H, and execute the program byi8094H_MP_CALL.

1. **All Macro Programs that were write inti i8094H could power outage carry-over.**
2. **It could not be applied to Macro Program.**

Parameters:

***cardNo*:**	**Module number**
***mpNo*:**	**Macro Program number (MP1 ~ MP157)**
***isrNo*:**	**Interrupt Service Program number (ISR1 ~ ISR20)**

Return:

None

Example:

```
i8094H_MP_CREATE(1, MP21); //Write Macro Program into i8094H.
//=======================================================
//The following functions will not be executed, but will be written into i8094H
// for further execution.
i8094H_SET_MAX_V(1, AXIS_XYZU, 20000);
//The maximum speedof the axis is 20K PPS.
i8094H_NORMAL_SPEED(1, AXIS_XYZU, 0);
//Set symmetric T-curve for XYZU axes on module.
i8094H_SET_V(1, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to be 2000 PPS.
i8094H_SET_A(1, AXIS_XYZU, 1000);
//set the acceleration of XYZU axes to be 1000 PPS/Sec
i8094H_SET_SV(1, AXIS_XYZU, 2000);
//set the start speed of XYZU axes to be 2000 PPS.
i8094H_SET_AO(1, AXIS_XYZU, 9);
//set the number of remaining offset pulses for XYZU axes to be 9 pulses.
i8094H_FIXED_MOVE(1, AXIS_XYZU, 10000);
//move XYZU axes to be 10000 pulses.
i8094H_MP_CLOSE(1);
// module 1 finish, write Macro Program into i8094H
//=======================================================
```

- ***void i8094H_MP_CLOSE(BYTE *cardNo*)**

Description:

Finish writing Macro Program into i8094H, this function will take effect only when working with i8094H_MP_CREATE.

Parameters:

***cardNo*: Module number**

Return:

None

Example:

Please refer to examples in above section.

7.2.2 Execute Macro Program(MP)

- ***void i8094H_MP_CALL(BYTE *cardNo*, BYTE *mpNo*)**

Description:

Execute a specific Macro Program on i8094H. For MP, the maximum layers could be up to 7. For ISR, only one layer.

Parameters:

***cardNo*:**	**Module number**
***mpNo*:**	**Macro Program number (MP1 ~ MP157)**

Return:

None

Example:

```
i8094H_MP_CREATE(1, MP21); //Write #21 Macro Program into i8094H.
//=======================================================
//The following functions will not be executed, but will be written into i8094H
// for further execution.
i8094H_SET_MAX_V(1, AXIS_XYZU, 20000);
//The maximum speedof the axis is 20K PPS.
i8094H_NORMAL_SPEED(1, AXIS_XYZU, 0);
//Set symmetric T-curve for XYZU axes on module.
i8094H_SET_V(1, AXIS_XYZU, 2000);
//set the speed of all axes on module 1 to be 2000 PPS.
i8094H_SET_A(1, AXIS_XYZU, 1000);
//set the acceleration of XYZU axes to be 1000 PPS/Sec
i8094H_SET_SV(1, AXIS_XYZU, 2000);
//set the start speed of XYZU axes to be 2000 PPS.
i8094H_SET_AO(1, AXIS_XYZU, 9);
//set the number of remaining offset pulses for XYZU axes to be 9 pulses.
i8094H_FIXED_MOVE(1, AXIS_XYZU, 10000);
//move XYZU axes to be 10000 pulses.
i8094H_MP_CLOSE(1);
// module 1 finish, write Macro Program into i8094H
//=======================================================
```

7.2.3 User Defined Variables

- ****void i8094H_MP_SET_VAR(BYTE *cardNo*, long *varNo*, long *varNo1*)**
- ****void i8094H_MP_SET_VAR(BYTE *cardNo*, long *varNo*, long *lVar1*)**

Description:

Set variable VARn = VARn
Set variable VARn = Value of variable

Parameters:

***cardNo*:**	**Module number**
***varNo*:**	**Variables to be used in Macro Program: VAR0 ~ VAR511**
***varNo1*:**	**Variables: VAR0 ~ VAR511**
***lVar1*:**	**Value of variable (-2,000,000,000 ~ +2,000,000,000)**

Return:

None

Example:

Please refer to examples in i8094H_MP_VAR_CALCULATE section.

- ****void i8094H_MP_SET_RVAR(BYTE *cardNo*, long *varNo*)**

Description:

Set variable VARn=return value of the function (the value will be transfered automatically to (long))

Parameters:

***cardNo*:**	**Module number**
***varNo*:**	**Variables to be used in Macro Program: VAR0 ~ VAR511**

Return:

None

Example:

Please refer to examples in i8094H_MP_VAR_CALCULATE section.

- ****void i8094H_MP_VAR_CALCULATE(**
  BYTE *cardNo*, long *varNo*, BYTE *Operator*, long *varNo1*, long *varNo2*)
- ****void i8094H_MP_VAR_CALCULATE(**
  BYTE *cardNo*, long *varNo*, BYTE *Operator*, long *lVar1*, long *varNo2*)

Description:

Variable calculation, *varNo* (+-*/) *varNo1* = *varNo2*
Variable calculation, *varNo* (+-*/) *value of valuable* = *varNo2*

Parameters:

***cardNo*:**	**Module number**
***varNo*:**	**Variables to be used in Macro Program: VAR0 ~ VAR511**
***Operator*:**	**'＋' addition**
	'－' subtraction
	'＊' multiplication
	'／' division(rounding down to the nearest integer)
***varNo1*:**	**Variables: VAR0 ~ VAR511**
***varNo1*:**	**Instance variable for calculation: VAR0 ~ VAR511**
***varNo2*:**	**Variable for storing calculation result: VAR0 ~ VAR511**
***lVar1*:**	**Value of variable(-2,000,000,000 ~ +2,000,000,000)**

Return:

None

Example:

```
i8094H_MP_CREATE(1, MP100); // Write #100 Macro Program into i8094H.
//=======================================================
//The following functions will not be executed, but will be written into i8094H
// for further execution.
i8094H_MP_SET_VAR(1, VAR1, 100); //VAR1 = 100。
i8094H_MP_SET_VAR(1, VAR2, 200); //VAR2 = 200。
i8094H_MP_SET_VAR(1, VAR10, 10); //VAR10 = 10。
//-------------------------------------------------------------------------------------------------
//VAR6 = i8094H_FRNET_RA(1, 8)
i8094H_FRNET_RA(1, 8);
i8094H_MP_SET_RVAR(1, VAR6); //VAR6 = current input of RA8 on module 1
//-------------------------------------------------------------------------------------------------
//VAR5 = i8094H_GET_LP(1, AXIS_X)
        i8094H_GET_LP(1, AXIS_X);
//Reads the LP value of the X axis on module 1.
i8094H_MP_SET_RVAR(1, VAR5);
//VAR5 = LP value of the X axis on module 1.
//-------------------------------------------------------------------------------------------------
```

```
i8094H_MP_VAR_CALCULATE(1, VAR1, '+', VAR2, VAR3);
//VAR1 + VAR2 = VAR3。
i8094H_MP_VAR_CALCULATE(1, VAR3, '-', VAR1, VAR3);
//VAR3 – VAR1 = VAR3。
i8094H_MP_VAR_CALCULATE(1, VAR3, '*', VAR10, VAR2);
//VAR3 x VAR10 = VAR3。
i8094H_MP_VAR_CALCULATE(1, VAR3, '/', VAR2, VAR1);
//VAR3 / VAR2 = VAR1。
//Final result VAR1 = 10、VAR2 = 200、VAR3 = 2,000、VAR10 = 10。
i8094H_MP_CLOSE(1);
// module 1 finish, write #100 Macro Program into i8094H
//=======================================================
```

7.2.4 Command Loop (FOR~NEXT)

- ***void i8094H_MP_FOR(BYTE *cardNo*, long *varNo*)**
- ***void i8094H_MP_FOR(BYTE *cardNo*, long *lVar*)**

Description:

This function performs command loop, it could work on as many as 7 levels.

Parameters:

***cardNo*:**	**Module number**
***varNo*:**	**Variables to be used in Macro Program: VAR0 ~ VAR511**
***lVar*:**	**Value of variable(-2,000,000,000 ~ +2,000,000,000)**

Return:

None

Example:

```
i8094H_MP_CREATE(1, MP100); // Write #100 Macro Program into i8094H.
//=======================================================
// The following functions will not be executed, but will be written into i8094H
// for further execution.
i8094H_SET_MAX_V(1, AXIS_X, 20000);
//The maximum speed of the axis X is 20K PPS.
i8094H_NORMAL_SPEED(1, AXIS_X, 0);
//Set symmetric T-curve for axis X
i8094H_SET_V(1, AXIS_X, 2000);
//set the speed of axis X to be 2000 PPS.
i8094H_SET_A(1, AXIS_X, 1000);
//set the acceleration of axis X to be 1000 PPS/Sec
i8094H_SET_SV(1, AXIS_X, 2000);
//set the start speed of axis X to be 2000 PPS.
i8094H_SET_AO(1, AXIS_X, 0);
//set the number of remaining offset pulses for axis X to be 0 PPS.
//-------------------------------------------------------------------------------------------------
i8094H_MP_SET_VAR(1, VAR1, 100); //VAR1 = 100。
i8094H_MP_FOR(1, VAR1);
// enable axis X to move back and forward 1,000 Pulse, loop for 100 times.
//or input the instant operating value i8094H_MP_FOR(1, 100)
i8094H_FIXED_MOVE(1, AXIS_X, 1000);
//move axis X to be 10000 pulses.
i8094H_FIXED_MOVE(1, AXIS_X, -1000); /
//move axis X to be -10000 pulses.
i8094H_MP_NEXT(1);
i8094H_MP_CLOSE(1);
// module 1 finish, write Macro Program into i8094H
//=======================================================
i8094H_MP_CALL(1, MP100);
// call module 1 i8094H, execute #100 Macro Program.
```

- ***void i8094H_MP_NEXT(BYTE *cardNo*)**

Description:

This function has to work together with i8094H_MP_FOR.

Parameters:

***cardNo*: Module number**

Return:

None

Please refer to examples in above section.

7.2.5 Condition Command (IF~ELSE)

- **void i8094H_MP_IF(BYTE *cardNo*, long *varNo*, WORD *Operator*, long*varNo1*)
- **void i8094H_MP_IF(BYTE *cardNo*, long *varNo*, WORD *Operator*, long *lVar1*)

Description:

Conditions to be described under condition command: For MP, it could work on as many as 7 levels. For ISR, it could work on 1 level at most.

Parameters:

***cardNo*:**	**Module number**
***varNo*:**	**Variable to be calculated: VAR0 ~ VAR511**
***Operator*:**	**'<' is less than** **'>' is greater than** **'<=' is less than or equal to** **'>=' is greater than or equal to** **'==' is equal to** **'!=' is not equal to** **(*Operator* : the single quotes(' ') represent a single or complex operators.)**
***varNo1*:**	**Condition variable: VAR0 ~ VAR511**
***lVar1*:**	**Value of variable (-2,000,000,000 ~ +2,000,000,000)**

Return:

None

Example:

```
i8094H_MP_CREATE(1, MP100); // Write #100 Macro Program into i8094H.
//=======================================================
// The following functions will not be executed, but will be written into i8094H
// for further execution.
i8094H_SET_MAX_V(1, AXIS_X, 20000);
//The maximum speed of the axis X is 20K PPS.
i8094H_NORMAL_SPEED(1, AXIS_X, 0);
//Set symmetric T-curve for axis X
i8094H_SET_V(1, AXIS_X, 2000);
//set the speed of axis X to be 2000 PPS.
i8094H_SET_A(1, AXIS_X, 1000);
//set the acceleration of axis X to be 1000 PPS/Sec
i8094H_SET_SV(1, AXIS_X, 2000);
//set the start speed of axis X to be 2000 PPS.
i8094H_SET_AO(1, AXIS_X, 0);
//set the number of remaining offset pulses for axis X to be 0 PPS.
//-------------------------------------------------------------------------------------------------
i8094H_MP_SET_VAR(1, VAR1, 100); //VAR1 = 100。
i8094H_MP_SET_VAR(1, VAR2, 200); //VAR2 = 200。
```

```
i8094H_MP_IF(1, VAR1, '<', VAR2);
i8094H_FIXED_MOVE(1, AXIS_X, 1000);
//move axis X to be 10000 pulses.
i8094H_MP_ELSE(1);
i8094H_FIXED_MOVE(1, AXIS_X, -1000);
//move axis X to be -10000 pulses.
i8094H_MP_IF_END(1);
i8094H_MP_CLOSE(1);
// module 1 finish, write #100 Macro Program into i8094H
//=======================================================
i8094H_MP_CALL(1, MP100);
// call module 1 i8094H, execute #100 Macro Program.
```

- ****void i8094H_MP_ELSE(BYTE *cardNo*)**

Description:

This function has to work together with i8094H_MP_IF.

Parameters:

***cardNo*: Module number**

Return:

None

Example:

Please refer to examples in above section.

- ****void i8094H_MP_IF_END(BYTE *cardNo*)**

Description:

Close i8094H_MP_IF function.

Parameters:

***cardNo*: Module number**

Return:

None

Example:

Please refer to examples in above section.

7.2.6 TIMER

- ***void** **i8094H_MP_TIMER(BYTE *cardNo*, long *varNo*)**
- ***void** **i8094H_MP_TIMER(BYTE *cardNo*, long *ms*)**

Description:

Delay the execution of the function.

Parameters:

***cardNo*:**	**Module number**
***varNo*:**	**Variables to be used in Macro Program: VAR0 ~ VAR511**
***ms*:**	**Value of variable(0 ~ +2,000,000,000) ms**

Return:

None

Example:

```
i8094H_MP_TIMER(1, 200);
//delay the execution of the function for 200ms.
```

7.2.7 Wait Motion Stop (For MP only)

- ***void** **i8094H_MP_STOP_WAIT(BYTE *cardNo*, BYTE *axis*)**

Description:

The MP function could be executed only after the axes completely stop.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**

Return:

None

Example:

None

7.2.8 User-defined RINT

- ****void i8094H_MP_SET_RINT(BYTE *cardNo*)**

Description:

This function configurate user-defined RINT settings. When this function is executed in MP, the module will send out 0x04 interrupt to the controller.

Parameters:

***cardNo*: Module number**

Return:

None

Example:

None