2-axis Motion Control Module User Manual

(I-8092F)

(Version 2.4)

Macro Function Library in C++ for

WinCon and I-8000 series PAC controllers

ICP DAS CO., LTD.

INDEX

1 PREFACE 6

1.1 Introduction 6
1.2 Macro functions 6
1.3 Funciton description 7

2 BASIC SETTINGS 8

2.1 Code numbers for axes 8
2.2 Registration of Modules and getting the LIB version 8
2.3 Resetting Module 11
2.4 Pulse Output Mode Setting 11
2.5 Setting the Maximum Speed 12
2.6 Setting the Active Level of the Hardware Limit Switches 13
2.7 Setting the Motion Stop Method When Limit Switch Is Sensed 14
2.8 Setting the Trigger Level of the NHOME Sensor 14
2.9 Setting Trigger Level of the Home sensor 15
2.10 Setting and Clearing the Software Limits 15
2.11 Setting the Encoder Related Parameters 16
2.12 Setting the Servo Driver (ON/OFF) 17
2.13 Setting the SERVO ALARM Function 18
2.14 Setting the Active Level of the In-Position Signals 19
2.15 Setting the Time Constant of the Digital Filter 19
2.16 Position Counter Variable Ring 21
2.17 Triangle prevention of fixed pulse driving 22
2.18 External Pulse Input 23
2.18.1 Handwheel (Manual Pulsar) Driving 23
2.18.2 Fixed Pulse Driving Mode 24
2.18.3 Continuous Pulse Driving Mode 25
2.18.4 Disabling the External Signal Input Functions 26
2.19 Configure hardware with pre-defined configuration file 27

3 READING AND SETTING THE REGISTERS 28

3.1 Setting and Reading the Command Position (LP) 28
3.2 Setting and Reading the Encoder Counter 29
3.3 Reading the Current Velocity 30
3.4 Reading the Current Acceleration 30
3.5 Reading the DI Status 31
3.6 Reading and Clearing the ERROR Status 33
3.7 Setting the general Dinigtal output 34

4 FRNET FUNCTIONS (FOR I8092F ONLY) 35
4.1 Read FRnet DI Signals 35
4.2 Write data to FRnet DO 36
5 AUTO HOMING 37
5.1 Setting the Homing Speed 38
5.2 Using an Limit Switch as the HOME sensor 39
5.3 Setting the Homing Mode 39
5.4 Starting the Homing Sequence 41
5.5 Waiting for the Homing sequence to be Completed 41
6 GENERAL MOTION CONTROL 42
6.1 Independent Axis Motion Control 42
6.1.1 Setting the Acceleration/Deceleration Mode 42
6.1.2 Setting the Start Speed 44
6.1.3 Setting the Desired Speed 45
6.1.4 Setting the Acceleration 45
6.1.5 Setting the Deceleration 46
6.1.6 Setting the Acceleration Rate 47
6.1.7 Setting the Value of the Remaining Offset Pulses 48
6.1.8 Fixed Pulse Output 49
6.1.9 Continuous Pulse Output 50
6.2 Interpolation Commands 51
6.2.1 Setting the Speed and Acc/Dec Mode for Interpolation 51
6.2.2 Setting the Vector Starting Speed 55
6.2.3 Setting the Vector Speed 55
6.2.4 Setting the Vector Acceleration 56
6.2.5 Setting the Vector Deceleration Value 57
6.2.6 Setting the Vector Acceleration Rate 58
6.2.7 Setting the Number of the Remaining Offset Pulses 59
6.2.8 2-Axis Linear Interpolation Motion 60
6.2.9 2-Axis Circular Interpolation Motion (an Arc) 61
6.2.10 2-Axis Circular Interpolation Motion 63
6.3 Continuous Interpolation 65
6.3.1 2-Axis Rectangular Motion 65
6.3.2 2-Axis Continuous Linear Interpolation 66
6.3.3 Multi-Segment Continuous Interpolation (Using Array) 68
6.3.4 2-Axis Ratio Motion 69
6.3.5 Mixed Linear and Circular 2-axis motions in Continuous Interpolation 71
6.4 Set the Interrupt Factors 74
6.4.1 Set the Interrupt Factors 74
6.4.2 Interrupt Disabled 76
6.4.3 Read the Interrupt Occurrence 77
6.5 Other functions 78

6.5.1 Holding the Driving Command 78
6.5.2 Release the Holding Status, and Start the Driving..79
6.5.3 Waiting until the Motion Is Completed..80
6.5.4 Stopping the Axes...81
6.5.5 Clear the Stop Status..85
6.5.6 End of Interpolation 85
6.5.7 Setting the COMPARE value 86

1 Preface

1.1 Introduction

- **This manual provides complete and detailed description of i8092F 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 six chapters. This chapter gives a brief description of this manual. Chapter 2 to 6 is the explanations of macro functions (MF).**

- **The functions defined in DLL file are explained 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).**

1.2 Macro functions

- **Macro functions provide a set of much easy-to-use functions that simplify the programming for users. Some necessary settings, such as speed range and deceleration point, are calculated inside those functions to ease the loading of users on having to understand the motion chip. Some useful costumed functions are provided for users to develop applications efficiently.**

1.3 Funciton description

All functions are listed in following form:

- **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.**

2 Basic Settings

2.1 Code numbers for axes

The axis assignments follow the definitions listed below: X=1, Y=2. If X and Y axes are assigned simultaneously, then the code number is 3. An assignment for either single axis or multiple axes at the same time is possible. Available axis code numbers are listed below. The type of the axis argument used in the functions is defined as WORD.

Table 2-1 Axis assignments and their corresponding codes

Axis	X	Y	XY
Code	0x1	0x2	0x3
Name	AXIS_X	AXIS_Y	AXIS_XY

2.2 Registration of Modules and getting the LIB version

- **BYTE i8092MF_REGISTRATION(BYTE *cardNo*, BYTE *slot*)**

Description:

This function registers a module that is installed in slot number, *slot*, by assigning a card number. Registration must be performed for each I-8092F motion control module before other functions are called. After registration, each module can be identified by its corresponding module number.

Parameters:

cardNo:	Module number

slot:	Slot number
	for I-8000: 0~7
	for 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 (i8092MF_REGISTRATION(cardNo, slot) == YES)
          {   //slot number begins from 1.
                //if a module is found, then it is registered as its slot number.
                i8092MF_RESET_CARD(cardNo);
                Found++;
        }
}
if (Found == 0)
{
        //if Wincon cannot find any I-8092F module,
        //please add codes to handle the exception here.
        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 (i8092MF_REGISTRATION(cardNo, slot) == YES)
        {
                //if a module is found, then it is registered by giving a number.
                i8092MF_RESET_CARD(cardNo);
                Found++;
        }
}
if (Found == 0)
{
        //if Wincon cannot find any I-8092F module,
        //please add codes to handle the exception here.
       return;
}
```

- **WORD i8092MF_GET_VERSION(void)**

Description:

Read the version of current i8092 library.

Parameters:

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

Return:

Version code:
including information of the year and the month: 0x0000 ~ 0x9999

Example:

```
WORD VER_No;
VER_No = i8092MF_GET_VERSION();
//Read the version code of i8092.dll
```

Remark:

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

2.3 Resetting Module

● **void i8092MF_RESET_CARD(BYTE *cardNo*)**

Description:

This function resets module using a software command.

Parameters:

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

Return:

None

Example:

```
i8092MF_RESET_CARD(1);
//Reset module 1.
```

2.4 Pulse Output Mode Setting

● **void i8092MF_SET_PULSE_MODE(BYTE *cardNo*, WORD *axis*, BYTE *nMode*)**

Description:

This function sets the pulse output mode as either CW/CCW or PULSE/DIR for the assigned axes and their direction definition.

Parameters:

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

Return:

None

Example:

```
i8092_SET_PULSE_MODE(1, AXIS_XYZ, 2);
//set the pulse mode of X, Y, and Z axes as mode 2
i8092_SET_PULSE_MODE(1, AXIS_U, 3);
//set the pulse mode of U axis as mode 3
```

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 i8092MF_SET_MAX_V(BYTE *cardNo*, WORD *axis*, DWORD *data*)**

Description:

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

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
***data*:**	**The assigned maximum speed of each axis when the controller performs an interpolation motion. However, setting the value of axis 1 is enough. For axis 1, the maximum value is 4,000,000 PPS.**

Return:

None

Example:

```
i8092MF_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 i8092MF_SET_HLMT(BYTE *cardNo*, WORD *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)
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:

```
i8092MF_SET_HLMT(1, AXIS_XY, 0, 0);
//set all the trigger levels as low-active for all limit switches
//on module 1.
```

2.7 Setting the Motion Stop Method When Limit Switch Is Sensed

- **void i8092MF_LIMITSTOP_MODE (BYTE *cardNo*, WORD *axis*, BYTE *nMode*)**

Description:

This function sets the motion stop mode of the axes when the corresponding limit switches are detected.

Parameters:

cardNo:	Module number
axis:	Axis or axes (Please refer to Table 2-1)
nMode:	0: stop immediately; 1: decelerating to stop

Return:

None

Example:

```
i8092MF_LIMITSTOP_MODE(1, AXIS_X, 0);
//set X axis to stop immediately if any limit switch on X axis is triggered.
```

2.8 Setting the Trigger Level of the NHOME Sensor

- **void i8092MF_SET_NHOME(BYTE *cardNo*, WORD *axis*, BYTE *nNHEdge*)**

Description:

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

Parameters:

cardNo:	Module number
axis:	Axis or axes (Please refer to Table 2-1)
nNHEdge:	Active level setting for the near home sensor. 0 = low active; 1 = high active

Return:

None

Example:

```
i8092MF_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 i8092MF_SET_HOME_EDGE(BYTE *cardNo*, WORD *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)**
***nHEdge*:**	**Active level setting for the home sensor. 0 = low active; 1 = high active**

Return:

None

Example:

```
i8092MF_SET_HOME_EDGE(1, AXIS_XY, 1);
//set the trigger level as high active for all home sensors on module 1.
```

2.10 Setting and Clearing the Software Limits

- **void i8092MF_SET_SLMT(BYTE *cardNo*, WORD *axis*, long *dwFL*, long *dwRL*, BYTE *nType*)**

Description:

This function sets the software limits.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
***dwFL*:**	**Value of the forward software limit (-2,147,483,648 ~ +2,147,483,647)**
***dwRL*:**	**Value of the reverse software limit (-2,147,483,648 ~ +2,147,483,647)**
***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:

```
i8092MF_SET_SLMT(1, AXIS_XY, 20000, -3000, 0);
//set the forward software limit as 20000 and the reverse
//software limit as -3000 for all axes on module 1.
```

- **void i8092MF_CLEAR_SLMT(BYTE *cardNo*, WORD *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:

```
i8092MF_CLEAR_SLMT(1, AXIS_XY);
//clear the software limits for all axes on module 1.
```

2.11 Setting the Encoder Related Parameters

- **void i8092MF_SET_ENCODER(BYTE *cardNo*, WORD *axis*, BYTE *nMode*, BYTE *nDivision*, BYTE *nZEdge*)**

Description:

This function sets the encoder input related parameters.

Parameters:

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

```
i8092MF_SET_ENCODER(1, AXIS_XY, 0, 0, 0);
//set the encoder input type as A quad B; the division is 1;
//and the Z phase is low active.
```

- **void i8092MF_SET_EN_DIR(BYTE *cardNo*, WORD *axis*, BYTE *nDir*)**

Description:

This function sets the encoder input direction.

Parameters:

cardNo:	Module number
axis:	Axis or axes (Please refer to Table 2-1)
nDir:	Encoder input direction: 0=positive dir; 1= negative dir

Return:

None

Example:

```
i8092MF_SET_EN_DIR(1, AXIS_XY, 0);
//set the encoder input direction to positive direction;
```

2.12 Setting the Servo Driver (ON/OFF)

- **void i8092_SERVO_ON(BYTE *cardNo*, WORD *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)

Return:

None

Example:

```
i8092_SERVO_ON(1, AXIS_XY);
//enables all drivers on module 1.
```

- **void i8092_SERVO_OFF(BYTE *cardNo*, WORD *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)

Return:

None

Example:

```
i8092_SERVO_OFF(1, AXIS_XY);
//disables all drivers on module 1.
```

2.13 Setting the SERVO ALARM Function

- **void i8092MF_SET_ALARM(BYTE *cardNo*, WORD *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)
nMode:	Mode: 0 = disable ALARM function; 1 = enable ALARM function
nAEdge:	Sets the trigger level 0 = low active; 1 = high active

Return:

None

Example:

```
i8092MF_SET_ALARM(1, AXIS_XY, 1, 0);
//enable the ALARM for X and Y axes on module 1 and set them
//as low-active.
```

2.14 Setting the Active Level of the In-Position Signals

- **void i8092MF_SET_INPOS(BYTE *cardNo*, WORD *axis*, BYTE *nMode*, BYTE *nlEdge*)**

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)**
***nMode*:**	**Mode: 0 = disable INPOS input; 1 = enable INPOS input**
***nlEdge*:**	**Set the trigger level 0 = low active; 1 = high active**

Return:

None

Example:

```
i8092MF_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 i8092MF_SET_FILTER(BYTE *cardNo*, WORD *axis*, WORD *FEn*, WORD *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)**
***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 μSEC	2 μSEC
1	224 μSEC	256 μSEC
2	448 μSEC	512 μSEC
3	896 μSEC	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:

```
i8092MF_SET_FILTER(1, AXIS_XY, 21, 3);
//set the filter time constants of X and Y 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 i8092MF_VRING_ENABLE(BYTE *cardNo*, WORD *axis*, DWORD *nVRing*)

Description:

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

Parameters:

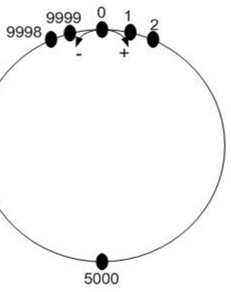
cardNo:	Module number
axis:	Axis or axes (Please refer to Table 2-1)
nVRing:	Maximum value of the ring counter (-2,147,483,648 ~ +2,147,483,647)

Return:

None

Example:

```
i8092MF_ 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 is 0 to 9999. When the counter value reach 9999, an 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 i8092MF_VRING_DISABLE(BYTE *cardNo*, WORD *axis*)

Description:

This function disables the variable ring counter function.

Parameters:

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

Return:

None

Example:

```
i8092MF_ 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 i8092MF_AVTRI_ENABLE (BYTE *cardNo*, WORD *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)

Return:

None

Example:

```
i8092MF_ AVTRI_ENABLE(1, AXIS_X);
//set the X axis of module 1 not to generate a triangle form in its speed profile.
```

● **void i8092MF_AVTRI_DISABLE (BYTE *cardNo*, WORD *axis*)**

Description:

This function disable the function that prevents a triangle form in linear acceleration fixed pulse driving.

Parameters:

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

Return:

None

Example:

```
i8092MF_ 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

2.18.1 Handwheel (Manual Pulsar) Driving

- **void i8092MF_EXD_MP(BYTE *cardNo*, WORD *axis*, long *data*)**

Description:

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

Parameters:

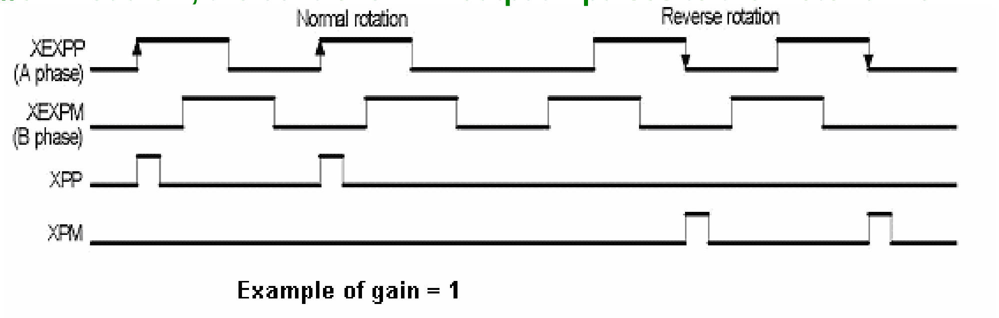
***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1.)**
	The axis can be either X and Y
***data*:**	**Gain (a multiplier)**

Return:

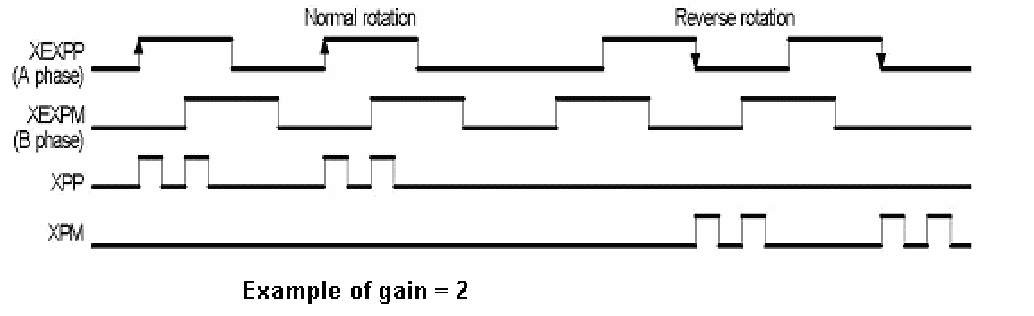
None

Example:

```
i8092MF_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 pulses to the motor driver.
```



```
i8092MF_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 i8092MF_EXD_FP(BYTE *cardNo*, WORD *axis*, long *data*)**

Description:

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

Parameters:

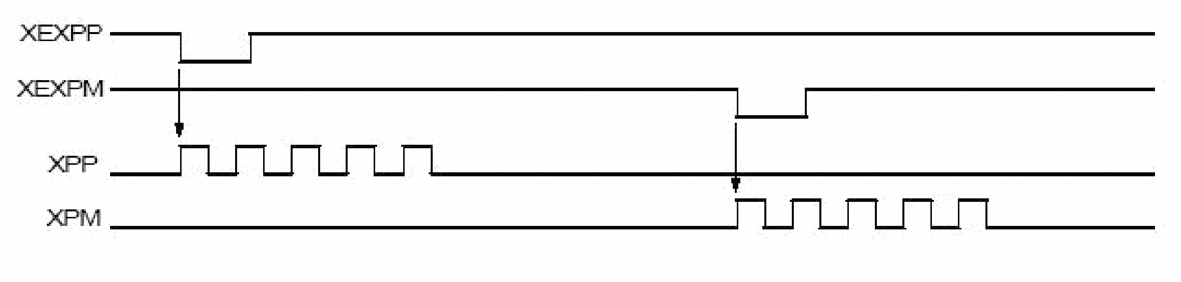
***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1.)**
	The axis can be either X and Y
***data*:**	**Number of fixed pulses.**

Return:

None

Example:

```
i8092MF_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 i8092MF_EXD_CP(BYTE *cardNo*, WORD *axis*, long *data*)**

Description:

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

Parameters:

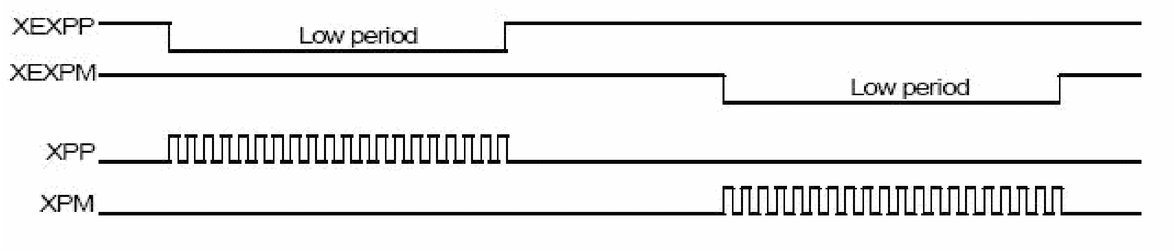
***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1.)**
	The axis can be either X and Y
***data*:**	**Pulse output speed in PPS**

Return:

None

Example:

```
i8092MF_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 i8092MF_EXD_DISABLE(BYTE *cardNo*, WORD *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 and Y

Return:

None

Example:

```
i8092MF_EXD_DISABLE(1, AXIS_X);
//disable the external input driving control function
//of X axis on module 1
```

2.19 Configure hardware with pre-defined configuration file

- **short i8092MF_ LOAD_CONFIG (BYTE *cardNo*)**

Description:

This function loads the pre-defined configuration file and set the target I8092 module automatically. The configuration file is generated by the PACEzGo.

Parameters:

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

Return:

0: successfully

-1: cannot open the pre-defined configuration file.

Example:

```
i8092MF_ LOAD_CONFIG (1);
//load the configuration file and configure the module 1.
```

3 Reading and Setting the Registers

3.1 Setting and Reading the Command Position (LP)

- **void i8092MF_SET_LP(BYTE *cardNo*, WORD *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 (-2,147,483,648 ~ +2,147,483,647)

Return:

None

Example:

```
i8092MF_SET_LP(1, AXIS_XY, 0);
//Set the LP for the X, Y, Z, and U axes of module 1 as 0,
//which means that all LP counters on module 1 will be cleared.
```

- **long i8092MF_GET_LP(BYTE *cardNo*, WORD *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 and Y

Return:

The current LP value (-2,147,483,648 ~ +2,147,483,647)

Example:

```
long X_LP;
X_LP = i8092MF_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 i8092MF_SET_EP(BYTE *cardNo*, WORD *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 (-2,147,483,648 ~ +2,147,483,647)

Return:

None

Example:

```
i8092MF_SET_EP(1, AXIS_XY, 0);
//Set the EP for the X, Y, Z, and U axes of module 1 as 0.
//This command clears all EP counters on module 1.
```

- **long i8092MF_GET_EP(BYTE *cardNo*, WORD *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 and Y

Return:

Current EP value (-2,147,483,648 ~ +2,147,483,647)

Example:

```
long X_EP;
X_EP = i8092MF_GET_EP(1, AXIS_X);
//reads the encoder value (EP) of the X axis on module 1.
```

3.3 Reading the Current Velocity

- **DWORD i8092MF_GET_CV(BYTE *cardNo*, WORD *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 and Y

Return:

Current speed (in PPS)

Example:

```
DWORD dwdata;
dwdata = i8092MF_GET_CV(1, AXIS_X);
//reads the current velocity of the X axis on module 1.
```

3.4 Reading the Current Acceleration

- **DWORD i8092MF_GET_CA(BYTE *cardNo*, WORD *axis*)**

Description:

This function reads the current acceleration value.

Parameters:

cardNo:	Module number
axis:	Axis (Please refer to Table 2-1) The axis can be either X and Y

Return:

Current acceleration (in PPS/Sec)

Example:

```
DWORD dwdata;
dwdata = i8092MF_GET_CA(1, AXIS_X);
//reads the current acceleration value of the X axis on module 1.
```

3.5 Reading the DI Status

● **BYTE i8092MF_GET_DI(BYTE *cardNo*, WORD *axis*, WORD *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 and Y

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 (i8092MF_GET_DI(1, AXIS_X, 1) == YES)
{
    //get the status of limit+ sensor of X axis on module 1
}
```

● **WORD i8092MF_GET_DI_ALL(BYTE *cardNo*, WORD *axis*)**

Description:

This function reads All digital inputs (DI) status.

Parameters:

cardNo:	Module number
axis:	Axis (Please refer to Table 2-1) The axis can be either X and Y

Return: a 16 bits value (0=Low,1=High)

bit 0	NHOME signal
bit 1	HOME signal
bit 2	Z-PHASE signal
bit 3	EMG signal(Only AXIS_X)
bit 4	EXP+ signal
bit 5	EXP- signal
bit 6	READY(INPOS) signal
bit 7	ALARM signal
bit 8	N/A
bit 9	N/A
bit 10	N/A
bit 11	IN3 signal
bit 12	N/A
bit 13	N/A
bit 14	LMT+ signal
bit 15	LMT- signal

Example:

```
WORD DI_Flag=i8092MF_GET_DI_ALL(1, AXIS_X) ;
    // get all status of module 1。
```

3.6 Reading and Clearing the ERROR Status

- **BYTE i8092MF_GET_ERROR(BYTE *cardNo*)**

Description:

This function checks whether an error occurs or not.

Parameters:

cardNo:	Module number

Return:

YES:	Some errors happened. Please use i8092MF_GET_ERROR_CODE () to get more information. If *GET_ERROR_CODE =256*, it means that the motion stop was due to the "STOP" command, not because an error happened. Please refer to 6.5.5 and following example to clear ERROR.
NO:	No error.

EXAMPLE:

```
If (i8092MF_GET_ERROR(1) == YES)
{
   //read module 1 and ERROR is found
    WORD ErrorCode_X = i8092MF_GET_ERROR_CODE(1, AXIS_X);
    WORD ErrorCode_Y = i8092MF_GET_ERROR_CODE(1, AXIS_Y);
    if ((ErrorCode_X || ErrorCode_Y) == 256)
    {
         //It means that motion was stopped due to the stop command was
        issued, not because any error happened. Please take some actions to
        clear the malfunction; then clear the STOP status.
         i8092MF_CLEAR_STOP(1);
    }

}
```

- **WORD i8092MF_GET_ERROR_CODE(BYTE *cardNo*, WORD *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 and Y

Return:

0 → no error
For non-zero return values, please refer to the following table. If there are not only one errors, the return value becomes the sum of these error code values.
For example, a return code 48 means that ALARM and EMGERENCY occurs at the same time.

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

Example:

```
if (i8092MF_GET_ ERROR_CODE(1, AXIS_X) & 10 )
{
    //Check if either the software limit or hardware limit (2+8)
    //in the reverse direction is asserted.
}
```

3.7 Setting the general Dinigtal output

- **void i8092MF_SET_OUT0(BYTE cardNo, WORD axis, WORD nLevel)**

Description:

This Function sets the Digital Output status.

Paramenter:

cardNo:	Module number
axis:	Axis (Please refer to Table 2-1) The axis can be either X and Y
nLevel:	DO output: 0=OFF,1=ON

Return: no

Example:

```
i8092MF_SET_OUT0 (1, AXIS_XY, 1);
//set the DO of X and Y to ON。
```

4 FRnet Functions (for i8092F only)

4.1 Read FRnet DI Signals

- **WORD i8092MF_FRNET_IN(BYTE *cardNo*, WORD *wRA*)**

Description:

This function reads the FRnet digital input signals. RA means the *Receiving Address* which can be one of the legal group number of FRnet. One group comprises 16 bits data. Therefore, total 128 DI can be defined for one FRnet interface.

Parameters:

***cardNo*:**	**Module number**
***wRA*:**	**Group number, range 8~15**
	Note: 0~7 are used for digital outputs

Return:

WORD	**16-bit DI data.**

Example:

```
WORD IN_Data;
IN_Data = i8092MF_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 i8092MF_FRNET_OUT(BYTE *cardNo*, WORD *wSA*, WORD *data*)**

Description:

This function write data to the FRnet digital output. SA means the *Sending Address* which can be one of the legal group number of FRnet. One group comprises 16 bits data. Therefore, total 128 DO can be defined for one FRnet interface.

Parameters:

***cardNo*:**	**Module number**
***wSA*:**	**Group number, range 0~7**
	Note: 8~15 are used by digital inputs
***data*:**	**16-bit data**

Return:

None

Example:

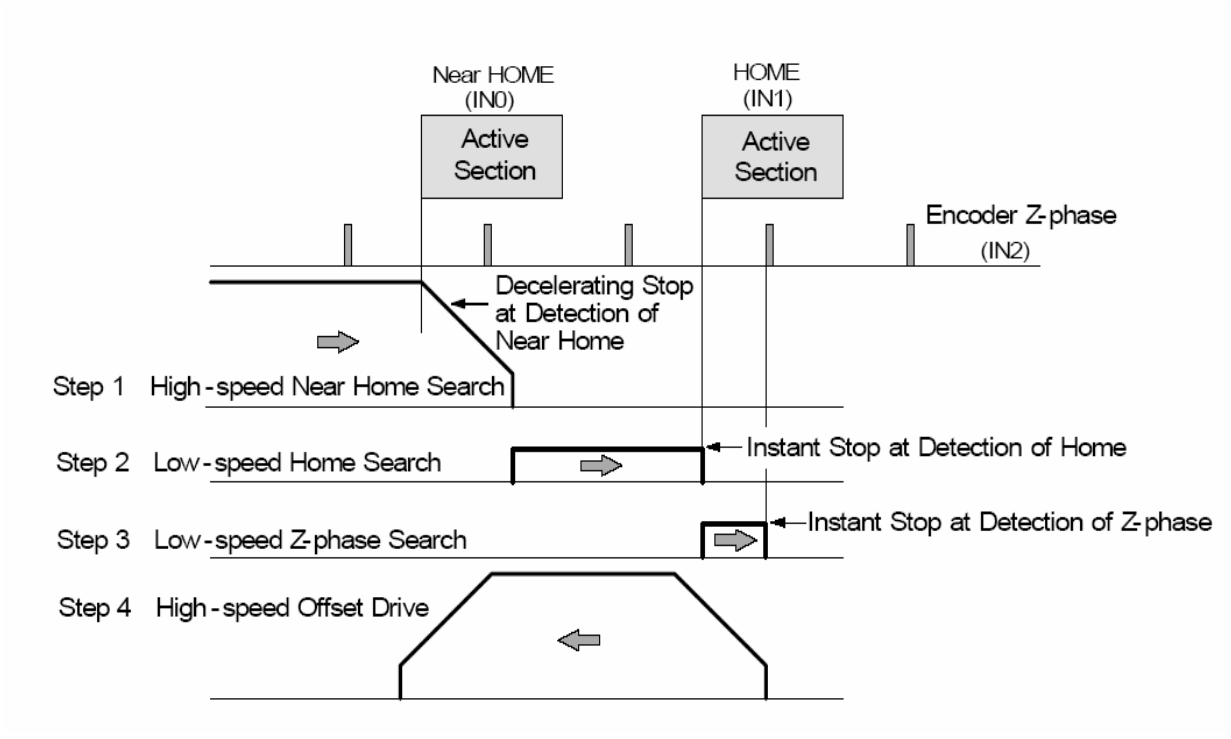
```
i8092MF_FRNET_OUT(1, 0,0xffff);
//Write 0xffff to the 16-bit DO which is on module 1 and the group number is 0.
```

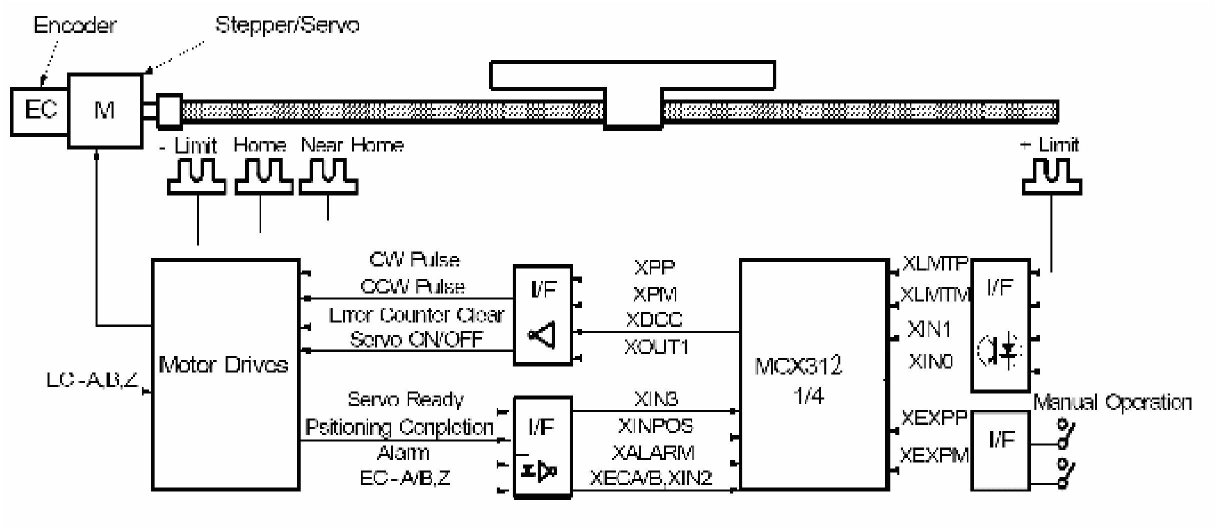
5 Auto Homing

The I-8092F module provides an automatic homing function. After setting the appropriate parameters, the assigned axes are able to perform automatic homing. Settings are required to be made in four steps for performing the automatic HOME search:

- **Search for the near home sensor (NHOME) at a normal speed (V).**
- **Search for the HOME sensor at low speed (HV).**
- **Search for the Encoder Z-phase (index) at low speed (HV).**
- **Move a specified number of offset pulses to the predefined origin point at normal speed (V).**

Some steps can be omitted. A detailed description of the related functions is provided in the following sections. Fully automated homing can reduce both programming time and CPU processing time.


Near HOME (IN0)
Active Section
HOME (IN1)
Active Section
Encoder Z-phase (IN2)
Decelerating Stop at Detection of Near Home
Step 1 High-speed Near Home Search
Step 2 Low-speed Home Search
Instant Stop at Detection of Home
Step 3 Low-speed Z-phase Search
Instant Stop at Detection of Z-phase
Step 4 High-speed Offset Drive



5.1 Setting the Homing Speed

- **void i8092MF_SET_HV(BYTE *cardNo*, WORD *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 (in PPS)**

Return:

None

EXAMPLE:

```
i8092MF_ SET_HV(1, AXIS_X, 500);
//set the homing speed of the X axis on module 1 to 500 PPS.
```

5.2 Using an Limit Switch as the HOME sensor

- **void i8092MF_HOME _LIMIT(BYTE *cardNo*, WORD *axis*, WORD *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: Does not use the LIMIT SWITCH as the HOME sensor; 1: Use the LIMIT SWITCH as the HOME sensor

Return:

None

EXAMPLE:

```
i8092MF_ HOME_LIMIT(1, AXIS_X, 0);
//Do not use the Limit Switch as the HOME sensor.
```

5.3 Setting the Homing Mode

- **void i8092MF_SET_HOME_MODE(BYTE *cardNo*, WORD *axis*, WORD *nStep1*, WORD *nStep2*, WORD *nStep3*, WORD *nStep4* , long *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 is not executed
	1: Moves in a positive direction
	2: Moves in a negative direction
nStep2:	
	0: Step 2 is not executed
	1: Moves in a positive direction
	2: Moves in a negative direction
nStep3:	
	0: Step 3 is not executed
	1: Moves in a positive direction

2: Moves in a negative direction

nStep4:

0: Step 4 is not executed
1: Moves in a positive direction
2: Moves in a negative direction

data: Offset value (0 ~ 2,147,483,647)

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.

i8092MF_SET_V(1, 0x1, 20000);
i8092MF_SET_HV(1, 0x1, 500);
i8092MF_SET_HOME_MODE(1, 0x1, 2, 2, 1, 1, 3500);
i8092MF_HOME_START(1, 0x1); //start auto-homing.
i8092MF_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 i8092MF_HOME _START(BYTE *cardNo*, WORD *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:

```
i8092MF_ 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

- **BYTE i8092MF _HOME_WAIT(BYTE *cardNo*, WORD *axis*)**

Description:

This function assigns commands to be performed while waiting for the automatic home search of all assigned axes to be completed.

Parameters:

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

Return:

YES	**The Homing sequence has been completed.**
NO	**The Homing sequence is not complete.**

Example:

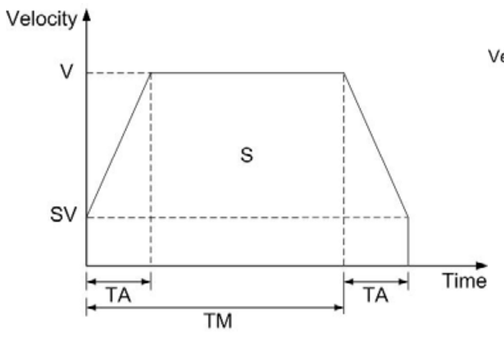
```
if (i8092MF_HOME_WAIT(1, AXIS_X) == NO)
{
    //perform some actions here if the X axis on module 1 has not completed
    //its homing sequence.
}
```

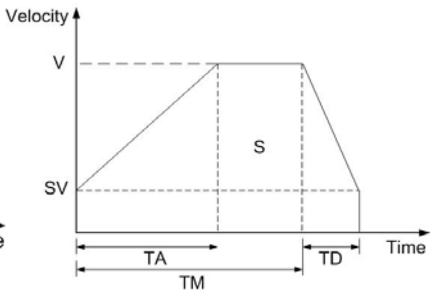
6 General Motion Control

6.1 Independent Axis Motion Control

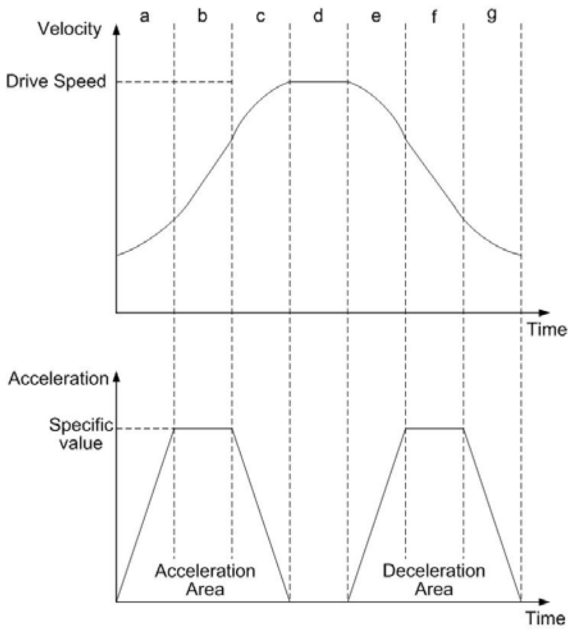
- **The motion of each axis can be started independently.**
- **Two axes are moving at the same time.**
- **Each axis is moving independently.**
- **Each axis can be commanded to change motion, such as changing the number of pulses or the speed.**
- **Each axis can be commanded to stop slowly or suddenly to meet the individual requirements.**

6.1.1 Setting the Acceleration/Deceleration Mode





Symmetric T-curve **Asymmetric T-curve**



Symmetric S-curve

- **void i8092MF_NORMAL_SPEED(BYTE *cardNo*, WORD *axis* , WORD *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)

Return:

None

Example:

```
BYTE cardNo=1; //select module 1.
i8092MF_SET_MAX_V(cardNo, AXIS_XY, 20000);
//set the max. speed of XY axes to 20K PPS.

//=========================================================
i8092MF_NORMAL_SPEED(cardNo, AXIS_XY, 0);
//use a symmetric T-curve for all axes on module 1.
i8092MF_SET_V(cardNo, AXIS_XY, 2000);
//set the speed of all axes on module 1 to 2000 PPS.
i8092MF_SET_A(cardNo, AXIS_XY,1000);
//set the acceleration of all axes on module 1 to 1000 PPS/Sec.
i8092MF_SET_SV(cardNo, AXIS_XY, 2000);
//set the start speed of all axes on module 1 to 2000 PPS.
i8092MF_SET_AO(cardNo, AXIS_XY, 9);
//set the number of remaining offset pulses for all axes to 9 pulses.
i8092MF_FIXED_MOVE(cardNo, AXIS_XY, 10000);
//move all axes on module 1 for 10000 pulses.

//=========================================================
i8092MF_NORMAL_SPEED(cardNo, AXIS_XY,1);
//use a symmetric S-curve for all axes on module 1.
i8092MF_SET_V(cardNo, AXIS_XY, 2000);
//set the speed of all axes on module 1 to 2000 PPS.
i8092MF_SET_K(cardNo, AXIS_XY, 50);
//set the acceleration rate of all axes on module 1 to 500 PPS/Sec^2.
i8092MF_SET_SV(cardNo, AXIS_XY, 200);
//set the start speed of all axes on module 1 to 200 PPS.
i8092MF_SET_AO(cardNo, AXIS_XY, 9);
//set the number of remaining offset pulses to 9 pulses for all axes.
i8092MF_FIXED_MOVE(cardNo, AXIS_XY, -10000);
```

```
//move all axes on module 1 for 10000 pulses in reverse direction.
//=====================================================
i8092MF_NORMAL_SPEED(cardNo, AXIS_XY,2);
//use an asymmetric T-curve for all axes on module 1.
i8092MF_SET_V(cardNo, AXIS_XY, 2000);
//set the speed of all axes on module 1 to 2000 PPS.
i8092MF_SET_A(cardNo, AXIS_XY,1000 );
//set the acceleration of all axes on module 1 to 1000 PPS/Sec.
i8092MF_SET_D(cardNo, AXIS_XY, 500);
//set the deceleration of all axes on module 1 to 500 PPS.
i8092MF_SET_SV(cardNo, AXIS_XY, 200);
//set the start speed of all axes on module 1 to 200 PPS.
i8092MF_SET_AO(cardNo, AXIS_XY, 9);
//set the number of remaining offset pulses to 9 pulses for all axes.
i8092MF_FIXED_MOVE(cardNo, axis, 10000);
//move all axes on module 1 for 10000 pulses.
```

Note: Relevant parameters must be set to achieve the desired motion.

6.1.2 Setting the Start Speed

- **void i8092MF_SET_SV(BYTE *cardNo*, WORD *axis*, DWORD *data*)**

Description:

This function sets the start speed for the assigned axes.

Parameters:

cardNo:	Module number
axis:	Axis or axes (Please refer to Table 2-1)
data:	The range is the same as for speed, and must not be zero or larger than the maximum speed. The maximum value is 4,000,000 PPS. For interpolation, set the speed value for axis1 is enough.

Return:

None

Example:

```
i8092MF_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 i8092MF_SET_V(BYTE *cardNo*, WORD *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:	The range is the same as for speed, and must not be zero or larger than the maximum speed. The maximum value is 4,000,000 PPS. For interpolation, set the speed value for axis1 is enough.

Return:

None

Example:

```
i8092MF_SET_V(1, AXIS_X, 120000);
//set the speed for the X axis on module 1 to 120000 PPS.
```

6.1.4 Setting the Acceleration

- **void i8092MF_SET_A(BYTE *cardNo*, WORD *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. The units are PPS/Sec. This value is related to the maximum speed value defined by i8092MF_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:

```
i8092MF_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.
i8092MF_SET_A(1, AXIS_X, 100000);
//set the acceleration value of the X axis on module 1 to 100K PPS/Sec.
```

6.1.5 Setting the Deceleration

- **void i8092MF_SET_D(BYTE *cardNo*, WORD *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 i8092MF_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:

```
i8092MF_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.
i8092MF_SET_D(1, AXIS_X, 100000);
//set the deceleration value of the X axis on module 1 to 100K PPS/Sec.
```

6.1.6 Setting the Acceleration Rate

- **void i8092MF_SET_K(BYTE *cardNo*, WORD *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. The units are $\mathrm{PPS/Sec}^2$. This value is related to the maximum speed value defined by i8092MF_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:

```
i8092MF_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.
i8092MF_SET_K(1, AXIS_X, 1000);
//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 Value of the Remaining Offset Pulses

- **void i8092MF_SET_AO(BYTE *cardNo*, WORD *axis*, short int *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:

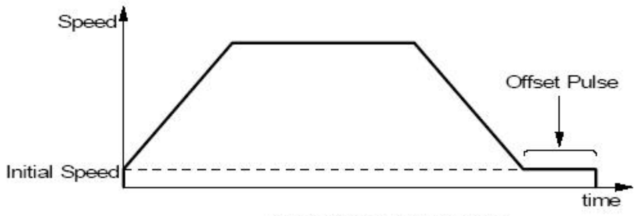
***cardNo*:**	**Module number**
***axis*:**	**Axis or axes (Please refer to Table 2-1)**
***data*:**	**The number of remaining offset pulses. (-32,768 ~ +32,767)**

Return:

None

Example:

```
i8092MF_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

6.1.8 Fixed Pulse Output

- **BYTE i8092MF_FIXED_MOVE(BYTE *cardNo*, WORD *axis*, long *data*)**

Description:

Command a point-to-point motion for several independent axes.

Parameters:

***cardNo*:**	**Module number**
***axis*:**	**Axis (Please refer to Table 2-1.)** **The axis can be either X and Y**
***data*:**	**Pulses (-268,435,455 ~ + 268,435,455)**

Return:

YES	**Some errors happen. Use i8092MF_GET_ERROR_CODE () to identify the errors.**
NO	**No error.**

Example:

```
BYTE cardNo=1; //select module 1
i8092MF_SET_MAX_V(cardNo, AXIS_XY, 20000);
//set the max. velocity of all axes on module 1 to be 20K PPS
i8092MF_NORMAL_SPEED(cardNo, AXIS_XY, 0);
//set the speed profile of all axes on module 1 to be symmetric T-curve
i8092MF_SET_V(cardNo, AXIS_XY, 2000);
//set the speed of all axes on module 1 to be 2000 PPS
i8092MF_SET_A(cardNo, AXIS_XY,1000);
//set the acceleration value of all axes on module 1 to be 1000 PPS/S
i8092MF_SET_SV(cardNo, AXIS_XY, 2000);
//set the start velocity of all axes on module 1 to be 2000 PPS
i8092MF_SET_AO(cardNo, AXIS_XY, 9);
//set the remaining offset pulses to be 9 PPS
i8092MF_FIXED_MOVE(cardNo, AXIS_XY, 10000);
// move 10000 Pulses for each axis on module 1
```

6.1.9 Continuous Pulse Output

- **BYTE i8092MF_CONTINUE_MOVE(BYTE *cardNo*, WORD *axis*, long *data*)**

Description:

This function issues a 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 and Y**
***data*:**	**The specified speed (positive value for CW motion; negative value for CCW motion)**

Return:

YES	**An error has occurred.** **Use the i8092MF_GET_ERROR_CODE() function to identify the errors.**
NO	**No error.**

Example:

```
BYTE cardNo=1; //select module 1
i8092MF_SET_MAX_V(cardNo, AXIS_XY, 20000);
//set the maximum speed of all axes on module 1 to 20K PPS.
i8092MF_NORMAL_SPEED(cardNo, AXIS_XY, 0);
//set the speed profile for all axes as a symmetric T-curve.
i8092MF_SET_V(cardNo, AXIS_XY, 2000);
//set the speed of all axes on module 1 to 2000 PPS.
i8092MF_SET_A(cardNo, AXIS_XY, 1000);
//set the acceleration value of all axes to 1000 PPS/S.
i8092MF_SET_SV(cardNo, AXIS_XY, 2000);
//set the start velocity of all axes to 2000 PPS
i8092MF_CONTINUE_MOVE(cardNo, AXIS_XY, 1000);
//move all axes on module 1 at a speed of 1000 PPS.
```

6.2 Interpolation Commands

I-8092/F is a motion module of 2-axes, so first axis of interpolation is fixed X-axis and second axis of interpolation is Y-axis.

6.2.1 Setting the Speed and Acc/Dec Mode for Interpolation

- **void i8092MF_VECTOR_SPEED(BYTE *cardNo*, WORD *nMode*)**

Description:

This function assigns the mode of vector speed of interpolation. Each interpolation mode will refer to construct a working coordinate system. The X-axis necessarily have to be the first axis. Different modes need different settings. Please refer to the mode definitions.

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)

Return:

None

Example:

```
BYTE cardNo=1; //select module 1.
i8092MF_SET_MAX_V(cardNo, AXIS_XY, 20000);
//set the maximum speed of all axes to 20K PPS.

//=============================================
i8092MF_VECTOR_SPEED(cardNo, 0);
//set module 1 to perform 2-axis linear or circular motion
//at a constant vector speed.
i8092MF_SET_VSV(cardNo, 1000);
```

```
//set the starting vector speed to 1000 PPS.
i8092MF_SET_VV(cardNo, 1000);
//set the vector speed to 1000 PPS.
i8092MF_LINE_2D(1, 12000, 10000);
//execute the 2-axis linear interpolation motion.

//===============================================
i8092MF_VECTOR_SPEED(cardNo, 1);
//set module 1 to perform 2-axis linear motion using a symmetric
//S-curve velocity profile.
i8092MF_SET_VSV(cardNo, 500);
//set the starting vector speed to 500 PPS.
i8092MF_SET_VV(cardNo, 2000);
//set the vector speed to 2000 PPS.
i8092MF_SET_VA(cardNo, 1000);
//set the vector acceleration to 1000 PPS/Sec.
i8092MF_LINE_2D(cardNo, 20000, 10000);
//execute the 2-axis linear interpolation motion.

//===============================================
i8092MF_VECTOR_SPEED(cardNo, 2);
//2-axis linear motion using a symmetric S-curve velocity profile.
i8092MF_SET_VSV(cardNo, 200);
//set the starting vector speed to 200 PPS.
i8092MF_SET_VV(cardNo, 2000);
//set the vector speed to 2000 PPS.
i8092MF_SET_VK(cardNo, 50);
//set the acceleration rate to 500 PPS/Sec.
i8092MF_SET_VAO(cardNo, 20);
//set the value of remaining offset pulses to 20.
i8092MF_LINE_2D(cardNo, 10000, 10000);
//execute the 2-axis linear interpolation motion.

//===============================================
i8092MF_VECTOR_SPEED(cardNo, 3);
//2-axis linear motion using an asymmetric T-curve velocity profile.
i8092MF_SET_VSV(cardNo, 100);
//set the start vector speed to 100 PPS.
i8092MF_SET_VV(cardNo, 2000);
//set the vector speed to 2000 PPS.
i8092MF_SET_VA(cardNo, 1000);
//set the vector acceleration to 1000 PPS/Sec.
i8092MF_SET_VD(cardNo, 500);
//set the vector deceleration to 500 PPS/Sec.
i8092MF_SET_VAO(cardNo, 20);
//set the value of remaining offset pulses to 20.
i8092MF_LINE_2D(cardNo, 10000, 5000);
```

```
//execute the 2-axis linear interpolation motion.

//================================================
long fp1=4000;
long fp2=10000;
int sv=200;
int v=2000;
i8092MF_SET_MAX_V(cardNo, AXIS_XY, 8000);
i8092MF_VECTOR_SPEED(cardNo, 4);
//2-axis linear motion using an asymmetric S-curve velocity profile.
i8092MF_SET_VSV(cardNo, sv);
//set the starting velocity to sv PPS.
i8092MF_SET_VV(cardNo, v);
//set the vector speed to v PPS.
i8092MF_SET_VK(cardNo, 50);
//set the acceleration rate to 500 PPS/Sec^2.
i8092MF_SET_VL(cardNo, 30);
//set the deceleration rate to 300 PPS/Sec^2.
i8092MF_SET_VAO(cardNo, 20);
//set the value of remaining offset pulses to 20.
i8092MF_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;
i8092MF_SET_MAX_V(cardNo, AXIS_XY, 8000);
i8092MF_VECTOR_SPEED(cardNo, 5);
//2-axis circular motion using a symmetric T-curve velocity profile
i8092MF_SET_VSV(cardNo, sv);
//set the starting vector speed to sv PPS.
i8092MF_SET_VV(cardNo, v);
//set vector speed to v PPS.
i8092MF_SET_VA(cardNo, a);
//set the vector acceleration to a PPS/Sec.
i8092MF_SET_VAO(cardNo, 0);
//set the value of remaining offset pulses to 0 Pulse.
i8092MF_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;
i8092MF_SET_MAX_V(cardNo, AXIS_XY, 8000);
i8092MF_VECTOR_SPEED(cardNo, 6);
//2-axis circular motion using an asymmetric T-curve velocity
//profile.
i8092MF_SET_VSV(cardNo, sv);
//set the starting vector speed to sv PPS.
i8092MF_SET_VV(cardNo, v);
//set vector speed to v PPS.
i8092MF_SET_VA(cardNo, a);
//set acceleration to a PPS/Sec.
i8092MF_SET_VD(cardNo, d);
//set the deceleration to d PPS/Sec.
i8092MF_SET_VAO(cardNo, 0);
//set the value of remaining offset pulses to 0.
i8092MF_CIRCLE_CW(cardNo, c1, c2);
//execute the 2-axis CW circular motion.
```

Note: **Relevant parameters should be set before issuing the motion command.**

6.2.2 Setting the Vector Starting Speed

- **void i8092MF_SET_VSV(BYTE *cardNo*, DWORD *data*)**

Description:

This function sets the starting speed of the principle X-axis for the interpolation motion.

Parameters:

***cardNo*:**	**Module number**
***data*:**	**The vector starting speed value (in PPS)**

Return:

None

Example:

```
i8092MF_SET_VSV(1, 1000);
//set the starting speed of the axis 1 for the interpolation motion
//on module 1 to 1000 PPS.
```

6.2.3 Setting the Vector Speed

- **void i8092MF_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.

Parameters:

***cardNo*:**	**Module number**
***data*:**	**The vector speed value (in PPS)**

Return:

None

Example:

```
i8092MF_SET_VV(1, 120000);
//set the vector speed of the interpolation on module 1
//to 120000 PPS.
```

6.2.4 Setting the Vector Acceleration

- **void i8092MF_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.

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 i8092MF_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:

```
i8092MF_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.
i8092MF_SET_VA(1, 100000);
//set the vector acceleration of the interpolation motion
//on module 1 to 100K PPS/Sec.
```

6.2.5 Setting the Vector Deceleration Value

- **void i8092MF_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). This value is related to the maximum speed value defined by i8092MF_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:

```
i8092MF_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.
i8092MF_SET_VD(1, 100000);
//set the vector deceleration value of interpolation motion
//on module 1 to 100K PPS/Sec.
```

6.2.6 Setting the Vector Acceleration Rate

- **void i8092MF_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. The units are $\mathrm{PPS/Sec}^2$. This value is related to the maximum speed value defined by i8092MF_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:

```
i8092MF_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.
i8092MF_SET_VK(1, 10000);
//set the acceleration rate of the interpolation motion on module
// 1 to 10,000 PPS/ Sec^2.
```

6.2.7 Setting the Number of the Remaining Offset Pulses

- **void i8092MF_SET_VAO(BYTE *cardNo*, short int *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 the offset pulse value has been reached. Please refer to the figure below for more information.

Parameters:

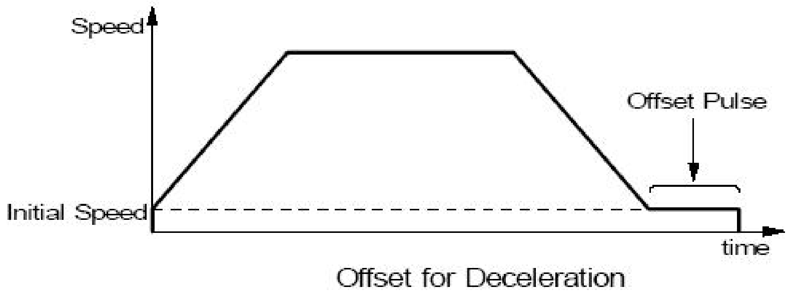
***cardNo*:**	**Module number**
***data*:**	**The number of remaining offset pulses (-32,768 ~ +32,767)**

Return:

None

Example:

```
i8092MF_SET_VAO(1, 200);
//set the number of remaining offset pulse value on module 1 to 200.
```



6.2.8 2-Axis Linear Interpolation Motion

- **BYTE i8092MF_LINE_2D(BYTE *cardNo*, long *fp1*, long *fp2*)**

Description:

This function executes a 2-axis linear interpolation motion.

Parameters:

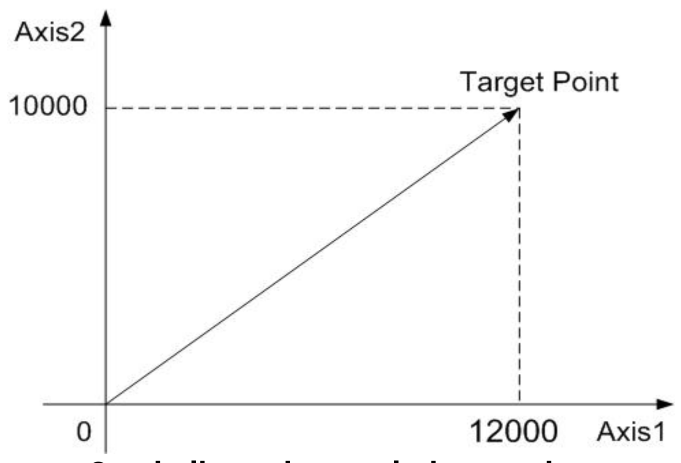
***cardNo*:**	**Module number**
***fp1*:**	**The displacement of the X-axis in Pulses (-8,388,607 ~ +8,388,607)**
***fp2*:**	**The displacement of the Y-axis in Pulses (-8,388,607 ~ +8,388,607)**

Return:

YES	**An error has occurred. Use the i8092MF_GET_ERROR_CODE() function to identify the error.**
NO	**No errors.**

Example:

```
i8092MF_LINE_2D(1, 12000, 10000);
//execute the 2-axis linear interpolation motion on module 1.
```



2-axis linear interpolation motion

6.2.9 2-Axis Circular Interpolation Motion (an Arc)

- **BYTE i8092MF_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:

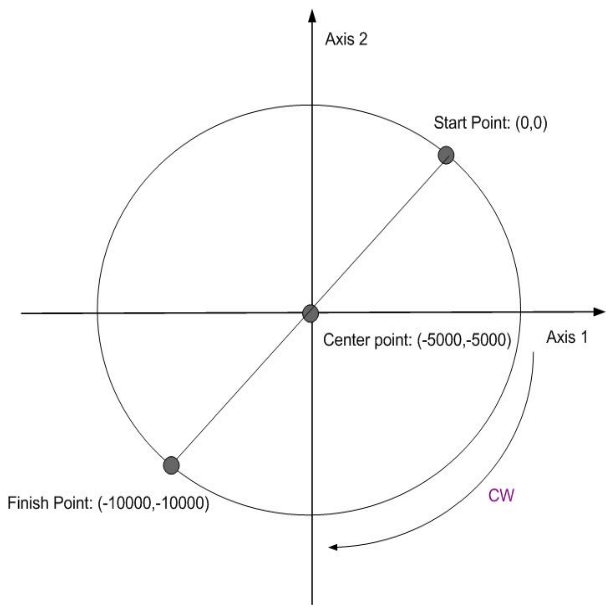
***cardNo*:**	**Module number**
***cp1*:**	**The relative position of the center to the current position of X-axis in pulses. (-8,388,607 ~ +8,388,607)**
***cp2*:**	**The relative position of the center to the current position of *Y*-axis in pulses. (-8,388,607 ~ +8,388,607)**
***fp1*:**	**The displacement of the X-axis in pulses. (-8,388,607 ~ +8,388,607)**
***fp2*:**	**Displacement of the Y-axis in pulses. (-8,388,607 ~ +8,388,607)**

Return:

YES	**An error has occurred. Use the i8092MF_GET_ERROR_CODE () function to identify the error.**
NO	**No errors.**

Example:

```
i8092MF_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

- **BYTE i8092MF_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:

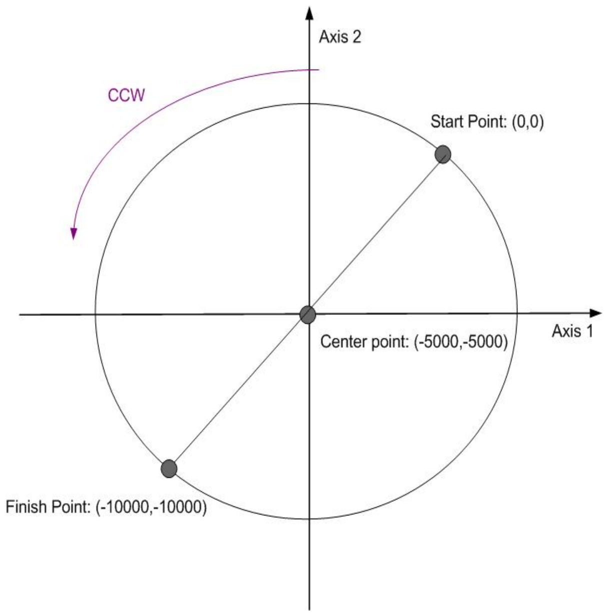
***cardNo*:**	**Module number**
***cp1*:**	**The relative position of the center to the current position of X-axis in pulses. (-8,388,607 ~ +8,388,607)**
***cp2*:**	**The relative position of the center to the current position of Y-axis in pulses. (-8,388,607 ~ +8,388,607)**
***fp1*:**	**The displacement of the X-axis in pulses. (-8,388,607 ~ +8,388,607)**
***fp2*:**	**Displacement of the Y-axis in pulses. (-8,388,607 ~ +8,388,607)**

Return:

YES	**An error has occurred. Use the i8092MF_GET_ERROR_CODE() function to identify the errors.**
NO	**No errors.**

Example:

```
i8092MF_ARC_CCW(1, -5000, -5000, -10000, -10000);
//Issues a command to perform a circular motion (an arc)
//in a CCW direction. Refer to the following figure.
```



2-axis circular motion in a CCW direction

6.2.10 2-Axis Circular Interpolation Motion

- **BYTE i8092MF_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 *X*-axis in pulses. (-8,388,607 ~ +8,388,607)
cp2:	The relative position of the center to the current position of Y-axis in pulses. (-8,388,607 ~ +8,388,607)

Return:

YES	An error has occurred. Use the i8092MF_GET_ERROR_CODE() function to identify the errors.
NO	No errors.

Example:

```
i8092MF_CIRCLE_CW(1, 0, 10000);
//execute a circular motion (a complete circle) in a CW direction on module 1.
```

- **BYTE i8092MF_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 X-axis in pulses. (-8,388,607 ~ +8,388,607)
cp2:	The relative position of the center to the current position of Y-axis in pulses. (-8,388,607 ~ +8,388,607)

Return:

YES	An error has occurred. Use the i8092MF_GET_ERROR_CODE () function to identify

	the error.
NO	**No errors**

Example:

```
i8092MF_CIRCLE_CCW(1, 0, 10000);
//execute a circular motion (a circle) in CCW direction
//on module 1
```

6.3 Continuous Interpolation

If it is broken and stopped, please solve it refer in section 6.5.5 !

6.3.1 2-Axis Rectangular Motion

- **BYTE i8092MF_RECTANGLE(**
  BYTE *cardNo*, WORD *nAcc*, WORD *Sp*, WORD *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. However, it is a software macro-function; therefore, it requires CPU resource to run this function.

Parameters:

***cardNo*:**	**Module number**
***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 ~ 8,388,607)**
***Wp*:**	**Width in Pulses (1 ~ 8,388,607)**
***Rp*:**	**Radius of each in pulses (1 ~ 8,388,607)**
***RSV*:**	**Starting speed (in PPS)**
***RV*:**	**Vector speed (in PPS)**
***RA*:**	**Acceleration (PPS/Sec)**
***RD*:**	**Deceleration of the last segment (in PPS/Sec)**

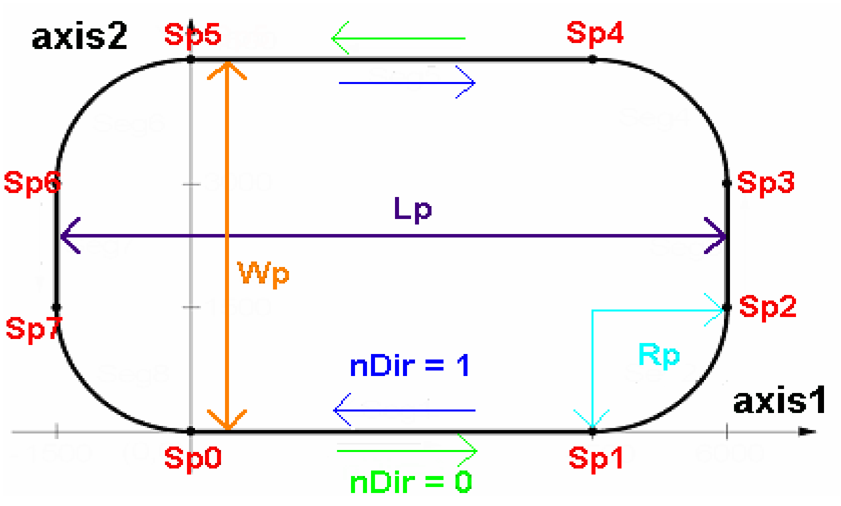
Return:

YES	**An error has occurred.** **Use the i8092MF_GET_ERROR_CODE() function to identify the error.**
NO	**No errors.**

Example:

```
BYTE cardNo=1; //select module 1.
int sv=1000;    //starting speed: 1000 PPS.
int v=10000;     //vector speed: 10000 PPS.
int a=5000;      //acceleration: 5000 PPS/Sec.
int d=5000;      //deceleration: 5000 PPS/Sec.
i8092MF_SET_MAX_V(cardNo, AXIS_XY, 16000);
//set the maximum speed to 16000 PPS.

i8092MF_RECTANGLE(cardNo, AXIS_X, AXIS_Y, 1, 0, 0, 20000, 10000, 1000, sv, v,
a, d);
//execute a rectangular motion on the XY plane
```



6.3.2 2-Axis Continuous Linear Interpolation

- **BYTE i8092MF_LINE_2D_INITIAL(BYTE *cardNo*, 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
VSV:	Starting speed (in PPS)
VV:	Vector speed (in PPS)
VA:	Vector acceleration (PPS/Sec)

Return:

None

Example:

```
i8092MF_LINE_2D_INITIAL(…);
//This function should be defined before the i8092MF_LINE_2D_CONTINUE()
//function is used. Please refer to the example of this function.
```

- **BYTE i8092MF_LINE_2D_CONTINUE(BYTE *cardNo*, WORD *nType*, long *fp1*, long *fp2)***

Description:

This function executes a 2-axis continuous linear interpolation. However, it is a software macro-function; therefore, it requires CPU resource to run this function.

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) (-8,388,607 ~ +8,388,607)
fp2:	The assigned number of pulses for the axis 2 (in Pulses) (-8,388,607 ~ +8,388,607)

Return:

YES	An error has occurred. Use the i8092MF_GET_ERROR_CODE () function to identify the error.
NO	No errors.

Example:

```
BYTE cardNo=1;     //select module 1.
int sv=300;           //starting speed: 300 PPS.
int v=18000;        //vector speed: 18000 PPS.
long a=500000; //acceleration: 500000 PPS/Sec.
int loop1;
i8092MF_SET_MAX_V(cardNo, AXIS_XY,160000);
i8092MF_LINE_2D_INITIAL(cardNo, AXIS_X, AXIS_Y, sv, v, a);
for (loop1=0; loop1<10000; loop1++)
{
    i8092MF_LINE_2D_CONTINUE(cardNo, 0, 100, 100);
    i8092MF_LINE_2D_CONTINUE(cardNo, 0, -100, -100);
}
i8092MF_LINE_2D_CONTINUE(cardNo, 1, 100, 100);
```

6.3.3 Multi-Segment Continuous Interpolation (Using Array)

- BYTE i8092MF_CONTINUE_INTP(
  BYTE *cardNo*, WORD *nAcc*, DWORD *VSV*, DWORD *VV*, DWORD *VA*,
  DWORD *VD*, BYTE *nType[ ]*, long *cp1[ ]*, long *cp2[ ]*, long *fp1[ ]*, long
  fp2[ ])

Description:

This function executes a multi-segment continuous interpolation. Those segments are stored in arrays declared in the arguments . The speed profile can be either a constant speed or a symmetric T-curve. The deceleration point will be calculated automatically. However, it is a software macro-function; therefore, it requires CPU resource to run this function.

Parameters:

cardNo:	Module number
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)

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

1 →i8092MF_LINE_2D(BYTE *cardNo*, long *fp1*, long *fp2*);
2 →i8092MF_ARC_CW(BYTE *cardNo*, long *cp1*, long *cp2*, long *fp1*, long *fp2*);
3 →i8092MF_ARC_CCW(BYTE *cardNo*, long *cp1*, long *cp2*, long *fp1*, long *fp2*);
4 →i8092MF_CIRCLE_CW(BYTE *cardNo*, long *cp1*, long *cp2*);
5 →i8092MF_CIRCLE_CCW(BYTE *cardNo*, long *cp1*, long *cp2*);
7 → It indicates the end of continuous interpolation.

cp1[ ]:	It contains a list of segment center point data at axis 1. (-8,388,607 ~ +8,388,607)
cp2[ ]:	It contains a list of segment center point data at axis 2. (-8,388,607 ~ +8,388,607)
fp1[ ]:	This array contains a list of segment end point data at axis 1. (-8,388,607 ~ +8,388,607)
fp2[ ]:	This array contains a list of segment end point data at axis 2. (-8,388,607 ~ +8,388,607)

Return:

YES	An error has occurred. Use the i8092MF_GET_ERROR_CODE () function to identify the error.
NO	No errors.

Example:

```
BYTE cardNo=1; //select module 1.
int sv=100; //set the starting speed to 100 PPS.
int v=3000; //set the speed to 3000 PPS.
int a=2000; //set the acceleration to 2000 PPS/Sec.
int d=2000; //set the deceleration to 2000 PPS/Sec.
i8092MF_SET_MAX_V(cardNo, AXIS_XY, 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};
//put data of the required segments in arrays.

i8092MF_CONTIUNE_INTP(
cardNo, AXIS_X, AXIS_Y, 0, 1, sv, v, a, d, nType, cp1, cp2, fp1, fp2);
//execute the 2-axis continuous interpolation.
//The deceleration point will be calculated automatically.
//For this example, the final position of this motion will return to the starting
point.
```

6.3.4 2-Axis Ratio Motion

- **BYTE i8092MF_RATIO_INITIAL(BYTE *cardNo*, DWORD *SV* , DWORD *V* , DWORD *A*, float *ratio*)**

Description:

This function sets the Initial values for ratio motion (motion in ratio) using a symmetric T-curve speed profile. However, it is a software macro-function; therefore, it requires CPU resource to run this function.

Parameters:

cardNo:	Module number
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).
ratio:	Set the ratio value between the two assigned axes.

Return:

None

Example:

```
i8092MF_RATIO_INITIAL(…);
//Initial setting for i8092MF_RATIO_2D(…) function.
//Please refer to the example of i8092MF_RATIO_2D() function.
```

- **BYTE i8092MF_RATIO_2D(BYTE *cardNo*, WORD *nType*, long *data*, WORD *nDir*)**

Description:

This function performs a two-axis ratio motion.

Parameters:

cardNo:	Module number
nType:	0 → Perform the ratio motion. 1 → Declare the end of ratio motion.
data:	The pulse number of X-axis (-8,388,607 ~ +8,388,607)
nDir:	Direction of the Y-axis. 0: CW; 1: CCW

Return:

YES	An error has occurred. Use the i8092MF_GET_ERROR_CODE () function to identify the error.
NO	No errors.

Example:

```
BYTE cardNo=1; //select module 1.
int sv=300; //set starting speed to 300 PPS.
int v=18000; //set vector speed to 18000 PPS.
long a=500000; //set acceleration value to 500K PPS/Sec.
int loop1, loop2;
i8092MF_SET_MAX_V(cardNo, 0Xf,160000);
//set maximum speed value to 18000 PPS.
i8092MF_RATIO_INITIAL(cardNo, sv, v, a, 0.36);
//The ratio is 0.36.
for (loop2 = 0; loop2 < 5; loop2++)
{
    for (loop1 = 0; loop1 < 5; loop1++)
    {
        i8092MF_RATIO_2D(cardNo, 0, 3600, 0);
```

```
                //perform the ratio motion in the CW direction.
                i8092MF_RATIO_2D(cardNo, 0, 3600, 1);
                //perform the ratio motion in the CCW direction.
            }
            i8092MF_RATIO_2D(cardNo, 0, 7200, 0);
            i8092MF_RATIO_2D(cardNo, 0, 3600, 1);
        }
        i8092MF_RATIO_2D(cardNo, 1, 7200, 0);
        //End the ratio motion.
```

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

- **void i8092MF_MIX_2D_INITIAL(BYTE *cardNo*, WORD *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
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:

```
i8092MF_MIX_2D_INITIAL(…);
//This function should be defined before the i8092MF_MIX_2D_CONTINUE()
//function is used. Please refer to the example of this function.
```

- **BYTE i8092MF_MIX_2D_CONTINUE(BYTE *cardNo*, WORD *nAcc*, WORD *nType*, long *cp1*, long *cp2*, long *fp1*, long *fp2*)**

Description:

This function executes mixed linear and circular 2-axis motion in continuous interpolation. However, it is a software macro-function; therefore, it requires CPU resource to run this function.

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 →i8092MF_LINE_2D(BYTE *cardNo*, long *fp1*, long *fp2*);
2 →i8092MF_ARC_CW(BYTE *cardNo*, long *cp1*, long *cp2*, long *fp1*, long *fp2*);
3 →i8092MF_ARC_CCW(BYTE *cardNo*, long *cp1*, long *cp2*, long *fp1*, long *fp2*);
4 →i8092MF_CIRCLE_CW(BYTE *cardNo*, long *cp1*, long *cp2*);
5 →i8092MF_CIRCLE_CCW(BYTE *cardNo*, long *cp1*, long *cp2*);

cp1:	It assigns the center point data at X-axis. (-8,388,607 ~ +8,388,607)
cp2:	It assigns the center point data at Y-axis. (-8,388,607 ~ +8,388,607)
fp1:	It assigns the end point data at X-axis. (-8,388,607 ~ +8,388,607)
fp2:	It assigns the end point data at Y-axis. (-8,388,607 ~ +8,388,607)

Return:

YES	An error has occurred. Use the i8092MF_GET_ERROR_CODE () function to identify the error.
NO	No errors.

Example:

```
BYTE cardNo=1;    //select module 1.
int sv=300;       //starting speed: 300 PPS
int v=18000;      //vector speed: 18000 PPS
long a=500000;    //acceleration: 500000 PPS/Sec

unsigned short loop1;
i8092MF_SET_MAX_V(cardNo, AXIS_XY, 160000);
```

```
i8092MF_MIX_2D_INITIAL(cardNo, 1, sv, v, a);
for (loop1 = 0; loop1 < 10000; loop1++)
{
      i8092MF_MIX_2D_CONTINUE (cardNo, 0, 1, 0, 0, 100, 100);
      i8092MF_MIX_2D_CONTINUE (cardNo, 0, 2, 100, 0, 100, 100);
}
i8092MF_MIX_2D_CONTINUE (cardNo, 1, 4, 100, 100, 0, 0);
```

6.4 Set the Interrupt Factors

6.4.1 Set the Interrupt Factors

- **void i8092MF_INTFACTOR_ENABLE(BYTE *cardNo*, WORD *axis*, *WORD* *nINT*)**

Description:

This function sets the interrupt factors

Parameters:

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

nINT Interrupt factors

Value	Symbol	Statement
1	P>=C-	Interrupt occurs when the value of logical / real position counter is larger than or equal to that of COMP- register. The COMP- must be pre-configired with i8092MF_SET_COMPARE( ) (please refer to Section 6.5.7)
2	P<C-	Interrupt occurs when the value of logical / real position counter is smaller than that of COMP- register. The COMP- must be pre-configired with i8092MF_SET_COMPARE( ) (please refer to Section 6.5.7)
3	P>=C+	Interrupt occurs when the value of logical / real position counter is smaller than that of COMP+ register. The COMP+ must be pre-configired with i8092MF_SET_COMPARE( ) (please refer to Section 6.5.7)
4	P<C+	Interrupt occurs when the value of logical / real position counter is larger than or equal to that of COMP+ register. The COMP+ must be pre-configired with i8092MF_SET_COMPARE( ) (please refer to Section 6.5.7)
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:

```
HANDLE hINT; //Interrupt event handle
HANDLE i8092_hThread; //IST handle
DWORD WINAPI i8092_ThreadFunction(LPVOID lParam); //IST function
BYTE CardNo=1;
BYTE Slot1=1;

//MFC button event: Create the thread and set the interrupt factor
void CI8092QCDlg::OnTestint()
{
DWORD dwThreadID = 0;
HWND hWnd = NULL;
//Create thread: i8092_ThreadFunction
i8092_hThread = CreateThread(NULL, 0, i8092_ThreadFunction, hWnd, 0,
&dwThreadID);
BYTE axis=AXIS_XY;
i8092MF_SET_MAX_V(CardNo, axis, 20000);
i8092MF_NORMAL_SPEED(CardNo, axis, 0);
i8092MF_SET_V(CardNo, axis, 20000);
i8092MF_SET_A(CardNo, axis, 100000);
i8092MF_SET_SV(CardNo, axis, 20000);
i8092MF_SET_AO(CardNo, axis, 0);
//Initialize the interrupt
hINTP=Slot_Register_Interrupt(Slot1);
//Set the interrupt factor: D-END
i8092MF_INTFACTOR_ENABLE(CardNo, AXIS_X, 7);
// 4-Axis fixed pulse drive
i8092MF_FIXED_MOVE(CardNo, AXIS_XY, 10000);

    while (i8092MF_STOP_WAIT(CardNo, 0xf) == NO)
    {   //Wait for motion done
        DoEvents();
        Sleep(1);
    }
}

//IST function
DWORD WINAPI i8092_ThreadFunction(LPVOID lParam)
{
DWORD dwEvent;
WORD RR3_X;
if(hINTP != NULL)
{
//Wait the event object
dwEvent = WaitForSingleObject(hINTP, INFINITE);
switch(dwEvent)
{
```

```
          case WAIT_OBJECT_0:
               //Get the interrupt event object successfully
               //While the driving stop, clear the position counter
               i8092MF_SET_LP(CardNo, AXIS_X, 0)
               // …
               //Other user codes in the IST
               // …
               //End of the interrupt
               Slot_Interrupt_Done(Slot1);
               //Get the interrupt status
               RR3_X = i8092_GET_RR3(CardNo, AXIS_X);
               //Disable the interrupt factor
               i8092MF_INTFACTOR_DISABLE(CardNo, AXIS_X);
               //Close the interrupt
               Slot_Interrupt_Close(Slot1);
               break;
          case WAIT_TIMEOUT:
               break;
          case WAIT_FAILED:
               break;
          }
          }

     return 1;
}
```

Note:

Please refer the three functions: Slot_Register_Interrupt(BYTE Slot)， Slot_Interrupt_Done(BYTE Slot)， Slot_Interrupt_Close(BYTE Slot) in the WinConSDK.

6.4.2 Interrupt Disabled

- **void i8092MF_INTFACTOR_DISABLE(BYTE *cardNo*, WORD *axis*)**

Description:

This function disables the interrupt factors

Parameters:

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

Return:

None

Example:

Please refer to 6.4.1

6.4.3 Read the Interrupt Occurrence

- **WORD i8092MF_GET_RR3(BYTE *cardNo*, WORD *axis*)**

Description:

Read the RR3 register that reflects the occurrence of Interrupt.

Parameters:

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

Return:

The content of RR3 register.

RR3 Value		說明
0x002	P>=C-	Once the value o flogic / real position counter is larger than that of COMP- register
0x004	P<C-	Once the value o flogic / real position counter is smaller than that of COMP- register
0x008	P<C+	Once the value o flogic / real position counter is smaller than that of COMP+ register
0x010	P>=C+	Once the value o flogic / real position counter is larger than that of COMP+ register
0x020	C-END	Interrupt occurs at the end of the constant speed drive or completion of Acceleration Offset Pulse output.
0x040	C-STA	Interrupt occurs at the start of the constant speed drive or begin of Acceleration Offset Pulse output.
0x080	D-END	Interrupt occurs when the driving is finished

Example:

```
i8092MF_GET_RR3 (cardNo, AXIS_X);
  //read the Interrupt status of AXIS_X
```

6.5 Other functions

6.5.1 Holding the Driving Command

- **void i8092MF_DRV_HOLD(BYTE *cardNo*, WORD *axis*)**

Description:

This command is usually used when users desire to starti 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 i8092MF_DRV_HOLD() is issued, and these commands will be started once the i8092MF_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 i8092MF_DRV_START(BYTE *cardNo*, WORD *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.
i8092MF_DRV_HOLD(cardNo, AXIS_XY); //hold the driving command to XY
i8092MF_SET_MAX_V(cardNo, AXIS_XY, 10000);
//set the maximum speed of X-axis and Y-axis to be 10K PPS.
i8092MF_NORMAL_SPEED(cardNo, AXIS_XY, 0);
//set the driving mode to be symmetric T-curve.
i8092MF_SET_V(cardNo, AXIS_X, 2000);
//set the speed of X-axis to 2,000 PPS.
i8092MF_SET_A(cardNo, AXIS_X, 1000);
//set the acceleration of X-axis to 1,000 PPS/S.
i8092MF_SET_SV(cardNo, AXIS_X, 2000);
//set the starting speed to 2,000 PPS.
i8092MF_SET_V(cardNo, AXIS_Y, 2000);
//set the speed of Y-axis to 2,000 PPS.
i8092MF_SET_A(cardNo, AXIS_Y, 1000);
//set the acceleration of Y-axis to 1,000 PPS/S.
i8092MF_SET_SV(cardNo, AXIS_Y, 2000);
//set the starting speed to 2,000 PPS.
i8092MF_FIXED_MOVE(cardNo, AXIS_X, 5000);
//command X-axis to move 5,000 Pulse.   This command is be held.
i8092MF_FIXED_MOVE(cardNo, AXIS_Y, 15000);
//command Y-axis to move 15,000 Pulse.   This command is be held.
i8092MF_DRV_START(cardNo, AXIS_XY);
//release the holding status. X and Y axes will start to move simultaneously.
```

6.5.3 Waiting until the Motion Is Completed

- **BYTE i8092MF_STOP_WAIT(BYTE *cardNo*, WORD *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
i8092MF_SET_MAX_V(cardNo, AXIS_XY, 20000);
//set the maximum speed of all axes on module 1 to 20K PPS.
i8092MF_NORMAL_SPEED(cardNo, AXIS_XY, 0);
//set the speed profile of all axes on module 1 to be symmetric T-curve
i8092MF_SET_V(cardNo, AXIS_XY, 2000);
//set the speed of all axes on module 1 to 2000 PPS.
i8092MF_SET_A(cardNo, AXIS_XY,1000);
//set the acceleration value of all axes on module 1 to 1000 PPS/S.
i8092MF_SET_SV(cardNo, AXIS_XY, 2000);
//set the start velocity of all axes on module 1 to 2000 PPS.
i8092MF_SET_AO(cardNo, AXIS_XY, 9);
//set the value of remaining offset pulses to 9 pulses.
i8092MF_FIXED_MOVE(cardNo, AXIS_XY, 10000);
// move all axes on module 1 for 10000 pulses.

if (i8092MF_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 i8092MF_STOP_SLOWLY(BYTE *cardNo*, WORD *axis*)**

Description:

This function decelerates and finally stops the assigned axes slowly.

Parameters:

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

Return:

None

Example:

```
i8092MF_STOP_SLOWLY(1, AXIS_XY);
//decelerate and stop the X and Y axes
```

- **void i8092MF_STOP_SUDDENLY(BYTE *cardNo*, WORD *axis*)**

Description:

This function immediately stops the assigned axes.

Parameters:

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

Return:

None

Example:

```
i8092MF_STOP_SUDDENLY(1, AXIS_XY);
//immediately stop the X and Y axes.
```

- **void i8092MF_VSTOP_SLOWLY(BYTE *cardNo*)**

Description:

This function stops interpolation motion of the assigned module in a decelerating way.

Parameters:

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

Return:

None

Example:

```
i8092MF_VSTOP_SLOWLY(1);
//stop the interpolation of card 1 in a decelerating way.
```

- **void i8092MF_VSTOP_SUDDENLY(BYTE *cardNo*)**

Description:

This function stops interpolation motion of the assigned module immediately.

Parameters:

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

Return:

None

Example:

```
i8092MF_VSTOP_SUDDENLY(1);
// stop the interpolation of card 1 immediately.
```

- **void i8092MF_SSTOP_SLOWLY(BYTE *cardNo*, WORD *axis*)**

Description:

Except for State-Control, This function provides the similar feature with i8092MF_STOP_ SLOWLY (). Stop pulse output simply (no ERROR_CODE 256 returned).

Parameters:

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

Return:

None

Example:

```
i8092MF_SSTOP_SLOWLY(1, AXIS_XY);
//decelerate and stop the X and Y axes
```

- **void i8092MF_SSTOP_SUDDENLY(BYTE *cardNo*, WORD *axis*)**

Description:

Except for State-Control, This function provides the similar feature with i8092MF_ VSTOP_ SUDDENLY (). Stop pulse output simply(no ERROR_CODE 256 returned).

Parameters:

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

Return:

None

Example:

```
i8092MF_SSTOP_SUDDENLY(1, AXIS_XY);
//immediately stop the X and Y axes.
```

- **void i8092MF_SVSTOP_SLOWLY(BYTE *cardNo*)**

Description:

Except for State-Control, This function provides the similar feature with i8092MF_ VSTOP_ SLOWLY (). Stop pulse output simply(no ERROR_CODE 256 returned).

Parameters:

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

Return:

None

Example:

```
i8092MF_SVSTOP_SLOWLY(1);
//stop the interpolation of card 1 in a decelerating way.
```

- **void i8092MF_SVSTOP_SUDDENLY(BYTE *cardNo*)**

Description:

Except for State-Control, This function provides the similar feature with i8092MF_ VSTOP_ SUDDENLY (). Stop pulse output simply(no ERROR_CODE 256 returned).

Parameters:

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

Return:

None

Example:

```
i8092MF_SVSTOP_SUDDENLY(1);
// stop the interpolation of card 1 immediately.
```

6.5.5 Clear the Stop Status

- **void i8092MF_CLEAR_STOP(BYTE *cardNo*)**

Description:

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

Paramters:

***cardNo*:**	Module number

Return:

None

Example:

```
i8092MF_CLEAR_STOP(1);
//clear the error status of card 1.
```

6.5.6 End of Interpolation

- **void i8092MF_INTP_END(BYTE *cardNo*, WORD *type*)**

Description:

1. 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.
2. You can redefine the MAX_V for each axis. In this way, you do not have to execute i8092MF_INTP_END() function.

Parameters:

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

Return:

None

Example:

```
i8092MF_INTP_END(1, 0); //declear the end of a 2-axis interpolation on card 1.
```

6.5.7 Setting the COMPARE value

- **void i8092MF_SET_COMPARE(BYTE *cardNo*, WORD *axis*, WORD *nSELECT*, WORD *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 → COMP+ 1 → COPM-
nTYPE:	Select the souece for comparison 0 → Position(P) = LP 1 → Position(P) = EP
data:	Set the COMPARE value: -2,147,483,648 ~ +2,147,483,647

Return:

None

Example:

```
i8092MF_SET_COMPARE(cardNo, AXIS_X, 0, 1, 5000);
//Set the comparison function for X-Axis.
//Set the compared source to be EP; and the COMP+ value to 5000.
```