

MPC-3004

4-axis Positioning Card

Software Manual (V1.2)

健昇科技股份有限公司

JS AUTOMATION CORP.

台北縣汐止市中興路 100 號 6 樓

6F, No.100, Chungshin Rd.

Shitsu, Taipei, Taiwan, R.O.C.

TEL : 886-2-2647-6936

FAX : 886-2-2647-6940

<http://www.automation.com.tw>

E-mail : control.cards@automation.com.tw

Correction record

Version	Record

Contents

1.	How to install the software of MPC3004	5
1.1	Install the PCI driver.....	5
1.2	How to install the utilities.....	5
2.	About the MPC-3004 software	6
2.1	What you need to get started.....	6
2.2	Software programming choices	6
3.	MPC3004 Language support	7
3.1	Building applications with the MPC3004 software library	7
3.2	MPC3004 Windows libraries.....	7
4.	Software overview	8
4.1	Initialization section.....	8
4.2	System parameter section	8
4.3	Motion / Homing section	9
4.4	Input / Output section	10
4.5	Soft limit section.....	11
4.6	Security section.....	11
4.7	Miscellaneous section	11
4.8	Linear interpolation section	12
5.	Flow chart of implement an application	13
5.1	MPC3004 Flow chart of implementation	13
5.2	MPC3004 Flow chart of speed mode application.....	16
6.	Function reference	17
6.1	Error codes and address	17
6.2	Variable data types	18
6.3	Programming language considerations	19
6.4	MPC3004 Functions	21
	Initialization Function.....	21
	mpc3004_initial	21
	mpc3004_close	21
	mpc3004_info	22
	System Parameter Function	23
	mpc3004_load_steps_lines_rev	23
	mpc3004_read_steps_lines_rev	24
	mpc3004_load_backlash.....	24
	mpc3004_read_backlash.....	25
	mpc3004_enable_zero	25
	mpc3004_set_dir_pol.....	26
	mpc3004_read_dir_pol	26
	Motion / Homing Section	27

mpc3004_read_axis_status	27
mpc3004_load_moving_speed	28
mpc3004_read_moving_speed	29
mpc3004_load_home_speed.....	30
mpc3004_read_home_speed.....	31
mpc3004_load_target_pos	31
mpc3004_read_target_pos	32
mpc3004_load_pos	32
mpc3004_reset_pos	33
mpc3004_read_pos	33
mpc3004_set_act_mode.....	34
mpc3004_read_act_mode	38
mpc3004_multi_start	38
mpc3004_start_motion	39
mpc3004_stop_motion	39
mpc3004_set_free_speed_range.....	40
mpc3004_free_speed_change.....	40
Input / Output Section.....	41
mpc3004_set_g_input_pol.....	41
mpc3004_read_g_input_pol	41
mpc3004_read_g_input	42
mpc3004_read_g_input_point	42
mpc3004_set_g_output_pol.....	43
mpc3004_read_g_output_pol	43
mpc3004_read_g_output	44
mpc3004_set_g_output.....	44
mpc3004_read_g_output_point	45
mpc3004_set_g_output_point.....	45
mpc3004_set_limit_pol	46
mpc3004_read_limit_pol.....	46
mpc3004_enable_limit.....	47
mpc3004_read_enable_limit.....	47
mpc3004_read_limit_stat	48
Soft limit section.....	49
mpc3004_read_block_status.....	49
mpc3004_read_block_axis_status	50
mpc3004_read_block_alarm_dir	50
mpc3004_load_soft_limit	51
mpc3004_read_soft_limit	51
Security Section	52
mpc3004_read_secret_status	52

mpc3004_open_secret_code.....	52
mpc3004_set_secret_code.....	53
mpc3004_change_secret_code.....	53
mpc3004_disable_secret_code.....	54
Miscellaneous section.....	55
mpc3004_read_counter.....	55
Linear interpolation section.....	56
mpc3004_move_P2P.....	56
mpc3004_move_line2.....	57
mpc3004_move_line3.....	58
mpc3004_move_line4.....	59
6.5 Dll list.....	60
7. MPC-3004 Error codes summary.....	62
7.1 MPC3004 Error codes table.....	62

1. **How to install the software of MPC3004**

1.1 Install the PCI driver

The PCI card is a plug and play card, once you add a new card on the window system will detect while it is booting. Please follow the following steps to install your new card.

In Windows 98/2000 system you should: (take win98 as example)

1. Make sure the power is off
2. Plug in the interface card
3. Power on
4. A hardware install wizard will appear and tell you it finds a new PCI card
5. Tell the wizard the directory of the driver files (\MPC3004\win98\driver), then it will automatically setup the driver
6. After installation, power off
7. Power on , it's ready to use

1.2 How to install the utilities

Execute the file \install\setup3004.exe will automatically install all the utilities and setup a new directory for your new interface card.

For more detail of step by step installation guide, please refer the file “installation.pdf “ on the CD come with the product or register as a member of our user’s club at:

<http://automation.com.tw/>

to download the complementary documents.

2. About the MPC-3004 software

MPC3004 software includes a set of dynamic link library (DLL) and system driver that you can utilize to control the interface card's ports and points separately.

Your MPC3004 software package includes setup driver, tutorial example and test program that help you how to setup and run appropriately, as well as an executable file which you can use to test each of the MPC3004 functions within Windows' operation system environment.

2.1 What you need to get started

To set up and use your MPC3004 software, you need the following:

- MPC3004 software
- MPC3004 hardware
 - Main board
 - Wiring board (Option)

2.2 Software programming choices

You have several options to choose from when you are programming MPC3004 software. You can use Borland C/C++, Microsoft Visual C/C++, Microsoft Visual Basic, or any other Windows-based compiler that can call into Windows dynamic link libraries (DLLs) for use with the MPC3004 software.

3. MPC3004 Language support

The MPC3004 software library is a DLL used with Windows 98/2000/XP. You can use these DLL with any Windows integrating development environment that can call Windows DLLs.

3.1 Building applications with the MPC3004 software library

The MPC3004 function reference topic contains general information about building MPC3004 applications, describes the nature of the MPC3004 files used in building MPC3004 applications, and explains the basics of making applications using the following tools:

Applications tools

- ◆ **Borland C/C++**
- ◆ **Microsoft Visual C/C++**
- ◆ **Microsoft Visual Basic**

If you are not using one of the tools listed, consult your development tool reference manual for details on creating applications that call DLLs.

3.2 MPC3004 Windows libraries

The MPC3004 for Windows function library is a DLL called **MPC3004.dll**. Since a DLL is used, MPC3004 functions are not linked into the executable files of applications. Only the information about the MPC3004 functions in the MPC3004 import libraries is stored in the executable files. Import libraries contain information about their DLL-exported functions. They indicate the presence and location of the DLL routines. Depending on the development tools you are using, you can make your compiler and linker aware of the DLL functions through import libraries or through function declarations.

Refer to **Table 1** to determine to which files you need to link and which to include in your development to use the MPC3004 functions in MPC3004.dll.

Header Files and Import Libraries for Different Development Environments		
Development Environment	Header File	Import Library
Microsoft C/C++	MPC3004.h	MPC3004VC.lib
Borland C/C++	MPC3004.h	MPC3004BC.lib
Microsoft Visual Basic	MPC3004.bas	

Table 1

4. Software overview

These topics describe the features and functionality of the MPC3004 boards and briefly describes the MPC3004 functions.

4.1 Initialization section

mpc3004_initial (void)	Software initializaton
mpc3004_close (void)	Software close
mpc3004_info (u8 CardID, u16 *VenID, u16 *address)	Get card information

4.2 System parameter section

mpc3004_load_steps_lines_rev (u8 CardID,u8 Axis,u16 NN,u16 MM)	Define the pulses per motor revolution and the moving distance per motor revolution
mpc3004_read_steps_lines_rev (u8 CardID,u8 Axis,u16 *NN,u16 *MM)	The pulses per motor revolution and the pitch per motor revolution readback
mpc3004_load_backlash (u8 CardID,u8 Axis,u16 backlash)	Define the pulses of backlash compensation
mpc3004_read_backlash (u8 CardID,u8 Axis,u16 *backlash)	The pulses of backlash compensation readback
mpc3004_enable_zero (u8 CardID,u8 Axis,u8 enable)	Enable / disable zero phase input as reciprocal homing homed condition.
mpc3004_set_dir_pol (u8 CardID,u8 Axis,u8 DIR)	Set axis coordinate direction
mpc3004_read_dir_pol (u8 CardID,u8 Axis,u8 *DIR)	Axis coordinate direction readback

4.3 Motion / Homing section

<u>mpc3004_read_axis_status</u> (u8 CardID,u8 Axis,u8 *ACT_rdy,u8 *POS_ok,u8 *DEC_ok)	Axis status readback
<u>mpc3004_load_moving_speed</u> (u8 CardID,u8 Axis,u32 VL,u32 VH,u32 ACC)	Set moving speed
<u>mpc3004_read_moving_speed</u> (u8 CardID,u8 Axis,u32 *VL,u32 *VH,u32 *ACC)	Moving speed readback
<u>mpc3004_load_home_speed</u> (u8 CardID,u8 Axis,u32 VL,u32 VH,u32 ACC,u16 VS)	Set homing speed
<u>mpc3004_read_home_speed</u> (u8 CardID,u8 Axis,u32 *VL,u32 *VH,u32 *ACC,u16 *VS)	Homing speed readback
<u>mpc3004_load_target_pos</u> (u8 CardID,u8 Axis,i32 pos)	Load value to target position
<u>mpc3004_read_target_pos</u> (u8 CardID,u8 Axis,i32 *pos)	Read target position
<u>mpc3004_load_pos</u> (u8 CardID,u8 Axis,i32 pos)	Load value to current position
<u>mpc3004_reset_pos</u> (u8 CardID,u8 Axis)	Reset current position
<u>mpc3004_read_pos</u> (u8 CardID,u8 Axis,i32 *pos)	Read current position
<u>mpc3004_set_act_mode</u> (u8 CardID,u8 Axis,u8 mode)	Set active mode
<u>mpc3004_read_act_mode</u> (u8 CardID,u8 Axis,u8 *mode)	Active mode readback
<u>mpc3004_multi_start</u> (u8 CardID,u8 ACT)	Start multiple axes simultaneously
<u>mpc3004_start_motion</u> (u8 CardID,u8 Axis)	Start single axis simultaneously
<u>mpc3004_stop_motion</u> (u8 CardID,u8 Axis,u8 Mode)	Stop single axis simultaneously
<u>mpc3004_set_free_speed_range</u> (u8 CardID,u8 Axis,u32 max_speed)	Set the free speed range of designated axis.
<u>mpc3004_free_speed_change</u> (u8 CardID,u8 Axis,i32 Speed)	Set/change the free speed of designated axis.

4.4 Input / Output section

<u>mpc3004_set_g_input_pol</u> (u8 CardID,u8 data)	Set general input polarity.
<u>mpc3004_read_g_input_pol</u> (u8 CardID,u8 *data)	General input polarity readback.
<u>mpc3004_read_g_input</u> (u8 CardID,u8 *data)	General input port data readback
<u>mpc3004_read_g_input_point</u> (u8 CardID,u8 Bit,u8 *stat)	Read general input point status readback
<u>mpc3004_set_g_output_pol</u> (u8 CardID,u8 data)	Set general output polarity
<u>mpc3004_read_g_output_pol</u> (u8 CardID,u8 *data)	General output polarity readback
<u>mpc3004_read_g_output</u> (u8 CardID,u8 *data)	General output port data readback
<u>mpc3004_set_g_output</u> (u8 CardID,u8 data)	Write general output command
<u>mpc3004_read_g_output_point</u> (u8 CardID,u8 Bit,u8 *stat)	General output point status readback
<u>mpc3004_set_g_output_point</u> (u8 CardID,u8 Bit,u8 stat)	Write general output point command
<u>mpc3004_set_limit_pol</u> (u8 CardID,u8 Axis,u8 LSP,u8 LSN,u8 HOME)	Set special purpose input point polarity
<u>mpc3004_read_limit_pol</u> (u8 CardID,u8 Axis,u8 *LSP,u8 *LSN,u8 *HOME)	Special purpose input point polarity readback
<u>mpc3004_enable_limit</u> (u8 CardID,u8 Axis,u8 LSP,u8 LSN,u8 HOME)	Enable / disable special purpose input point
<u>mpc3004_read_enable_limit</u> (u8 CardID,u8 Axis,u8 *LSP,u8 *LSN,u8 *HOME)	Enable / disable special purpose input point readback
<u>mpc3004_read_limit_stat</u> (u8 CardID,u8 Axis,u8 *LSP,u8 *LSN,u8 *HOME)	Special purpose input point status readback

4.5 Soft limit section

<u>mpc3004_read_block_status</u> (u8 CardID,u8 *b0,u8 *b1,u8 *b2,u8 *b3,u8 *b4,u8 *b5,u8 *b6,u8 *b7)	Read soft limit block were normal or alarm
<u>mpc3004_read_block_axis_status</u> (u8 CardID,u8 *X,u8 *Y,u8 *Z,u8 *A)	Read axes function enable/disable after software limit check
<u>mpc3004_read_block_alarm_dir</u> (u8 CardID,u8 *X,u8 *Y,u8 *Z,u8 *A)	Read moving direction during block alarm generated
<u>mpc3004_load_soft_limit</u> (u8 CardID,u8 Block,u8 Axis,i32 Ndata,i32 Pdata,u8 Inhibit,u8 enable)	Set soft limit parameter.
<u>mpc3004_read_soft_limit</u> (u8 CardID,u8 Block,u8 Axis,i32 *Ndata,i32 *Pdata,u8 *Inhibit,u8 *enable)	Soft limit parameter readback

4.6 Security section

<u>mpc3004_read_secret_status</u> (u8 CardID,u8 *open,u8 *enable)	Read security status
<u>mpc3004_open_secret_code</u> (u8 CardID,u16 O0,u16 O1,u16 O2,u16 O3,u16 O4)	Unlock the security.
<u>mpc3004_set_secret_code</u> (u8 CardID,u16 N0,u16 N1,u16 N2,u16 N3,u16 N4)	Enable the security.
<u>mpc3004_change_secret_code</u> (u8 CardID,u16 O0,u16 O1,u16 O2,u16 O3,u16 O4,u16 N0,u16 N1,u16 N2,u16 N3,u16 N4)	Change password
<u>mpc3004_disable_secret_code</u> (u8 CardID,u16 O0,u16 O1,u16 O2,u16 O3,u16 O4)	Disable the security.

4.7 Miscellaneous section

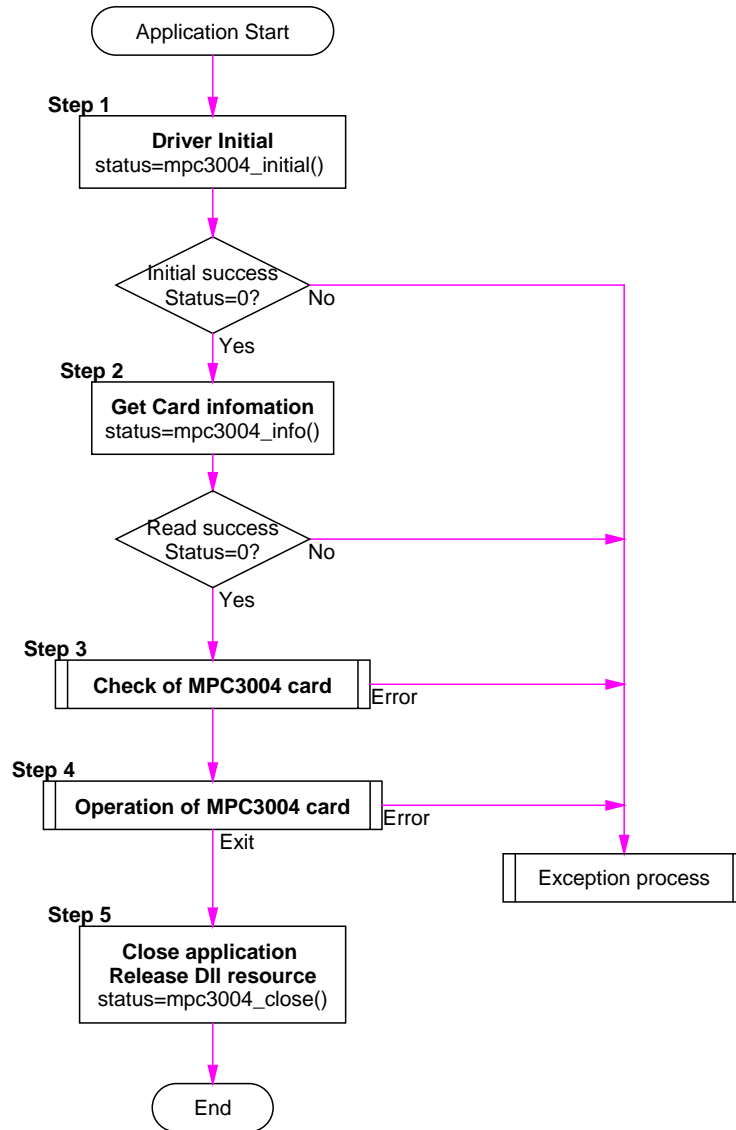
<u>mpc3004_read_counter</u> (u8 CardID,u16 *count)	Read MPC3004 card timer
--	-------------------------

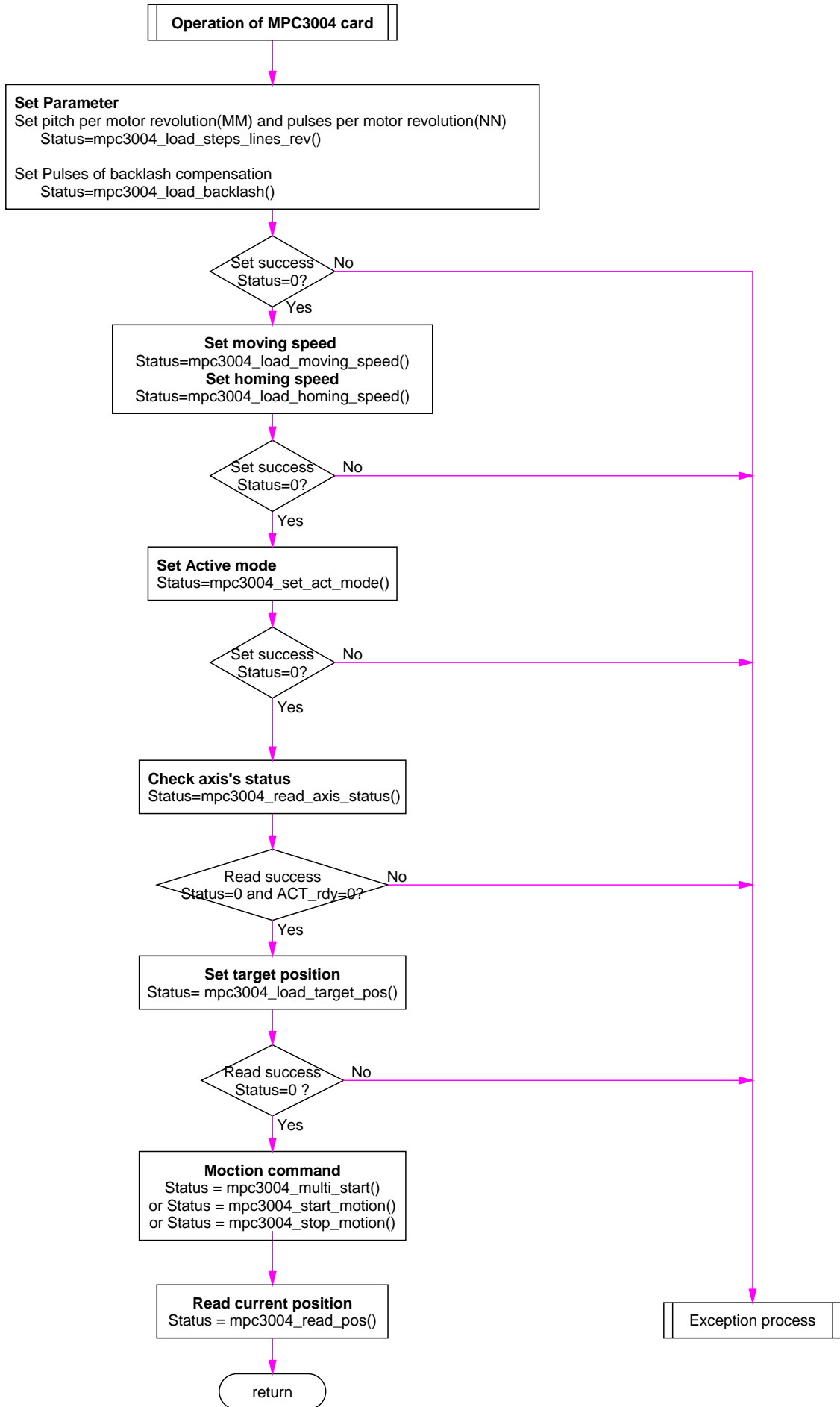
4.8 Linear interpolation section

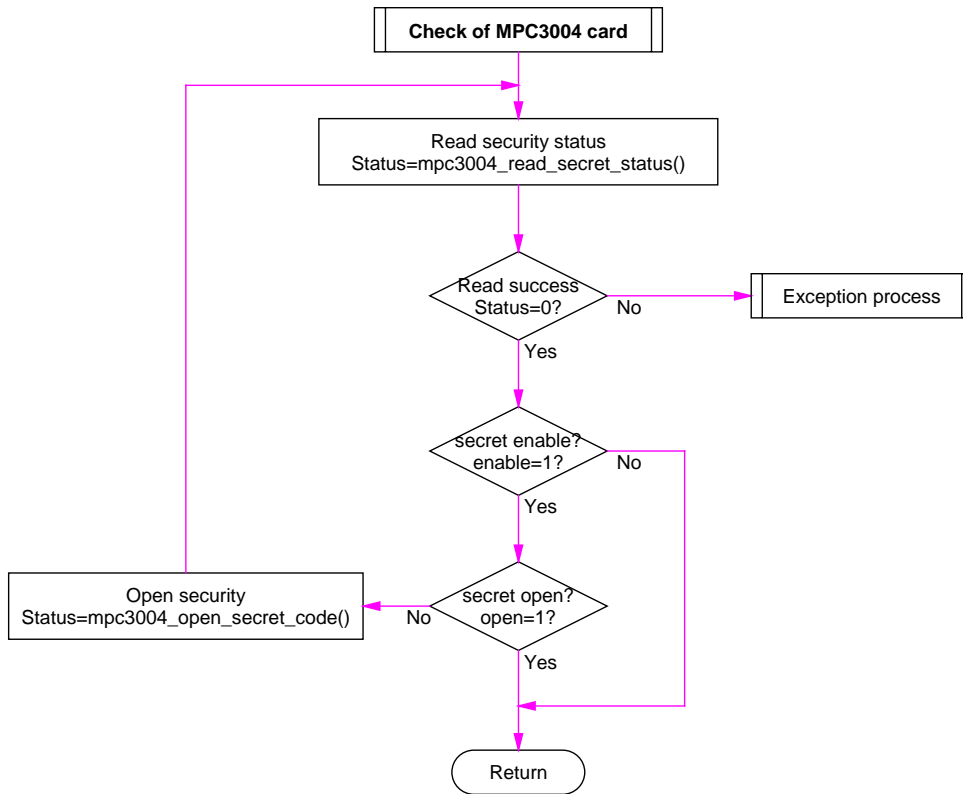
mpc3004_move_P2P (u8 CardID,u8 Axis,i32 POS,u8 Pos_mode,u32 Vl,u32 Vh,u32 Vacc)	Command to move at positioning mode.
mpc3004_move_line2 (u8 CardID, u8 SetAxes, i32 POS0, i32 POS1, u8 Pos_mode,u32 Vl,u32 Vh,u32 Vacc)	Command to move at linear interpolation mode.
mpc3004_move_line3 (u8 CardID, u8 SetAxes, i32 POS0, i32 POS1, i32 POS2,u8 Pos_mode,u32 Vl,u32 Vh,u32 Vacc)	Command to move at 3-axes linear interpolation mode.
mpc3004_move_line4 (u8 CardID, i32 POS0, i32 POS1, i32 POS2, i32 POS3, u8 Pos_mode,u32 Vl,u32 Vh,u32 Vacc)	Command to move at 4-axes linear interpolation mode.

5. Flow chart of implement an application

5.1 MPC3004 Flow chart of implementation





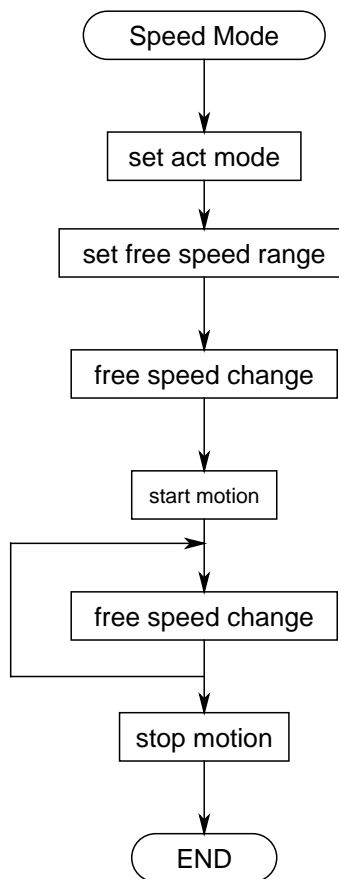


5.2 MPC3004 Flow chart of speed mode application

Speed mode is a special function of positioning, you may take it as the target position at infinite, but in reality, the digital system has finite word length, you can't move to infinite length. If you forgot the position constrain, you may use speed mode freely.

The speed mode features of MPC3004:

1. speed re-programmable on the fly
2. pre-settable speed range for better speed accuracy



6. **Function reference**

These topics contain detailed descriptions of each MPC3004 function. The functions are arranged alphabetically by function name. Refer to MPC3004 Function Reference for additional information.

6.1 Error codes and address

Every MPC3004 function is consist of the following format:

Status = function_name (parameter 1, parameter 2, ... parameter n)

Each function returns a value in the **Status** global variable that indicates the success or failure of the function. A returned **Status** equal to zero that indicates the function executed successfully. A non-zero status indicates failure that the function did not execute successfully because of an error, or executed with an error.

Note : Status is a 32-bit unsigned integer.

The first parameter to almost every MPC3004 function is the parameter **CardID** which is located the driver of MPC3004 board you want to use those given operation. The **CardID** is assigned by DIP SW. You can utilize multiple devices with different card ID within one application; to do so, simply pass the appropriate **CardID** to each function.

Note: CardID is set by dip switch (0x0-0xF)

6.2 Variable data types

Every function description has a parameter table that lists the data types for each parameter. The following sections describe the notation used in those parameter tables and throughout the manual for variable data types.

Primary Type Names					
Name	Description	Range	C/C++	Visual BASIC	Pascal (Borland Delphi)
u8	8-bit ASCII character	0 to 255	char	Not supported by BASIC. For functions that require character arrays, use string types instead.	Byte
i16	16-bit signed integer	-32,768 to 32,767	short	Integer (for example: deviceNum%)	SmallInt
u16	16-bit unsigned integer	0 to 65,535	unsigned short for 32-bit compilers	Not supported by BASIC. For functions that require unsigned integers, use the signed integer type instead. See the i16 description.	Word
i32	32-bit signed integer	-2,147,483,648 to 2,147,483,647	long	Long (for example: count&)	LongInt
u32	32-bit unsigned integer	0 to 4,294,967,295	unsigned long	Not supported by BASIC. For functions that require unsigned long integers, use the signed long integer type instead. See the i32 description.	Cardinal (in 32-bit operating systems). Refer to the i32 description.
f32	32-bit single-precision floating-point value	-3.402823E+38 to 3.402823E+38	float	Single (for example: num!)	Single
f64	64-bit double-precision floating-point value	-1.797683134862315E+308 to 1.797683134862315E+308	double	Double (for example: voltage Number)	Double

Table 2

6.3 Programming language considerations

Apart from the data type differences, there are a few language-dependent considerations you need to be aware of when you use the MPC3004 API. Read the following sections that apply to your programming language.

Note: Be sure to include the declaration functions of MPC3004 prototypes by including the appropriate MPC3004 header file in your source code. Refer to Building Applications with the MPC3004 Software Library for the header file appropriate to your compiler.

6.3.1 C/C++

For C or C++ programmers, parameters listed as Input/Output parameters or Output parameters are pass-by-reference parameters, which means a pointer points to the destination variable should be passed into the function. For example, the Read current position has the following format:

```
Status = mpc3004_read_pos (u8 CardID, u8 axis, i32 *pos);
```

where **CardID** and **axis** are input parameters, and **pos** is an output parameter.

Consider the following example:

```
u8 CardID, axis ;
```

```
i32 pos,
```

```
u32 Status;
```

```
Status = mpc3004_read_pos (u8 CardID , u8 axis , i32 &pos);
```

6.3.2 Visual basic

The file **mpc3004.bas** contains definitions for constants required for obtaining MPC Card information and declared functions and variable as global variables. You should use these constants symbols in the **mpc3004.bas**, do not use the numerical values.

In Visual Basic, you can add the entire **mpc3004.bas** file into your project. Then you can use any of the constants defined in this file and call these constants in any module of your program. To add the **mpc3004.bas** file for your project in Visual Basic 4.0, go to the **File** menu and select the **Add File... option**. Select **mpc3004.bas**, which is browsed in the **mpc3004 \ api** directory. Then, select **Open** to add the file to the project.

To add the **mpc3004.bas** file to your project in Visual Basic 5.0 and 6.0, go to the **Project** menu and select **Add Module**. Click on the Existing tab page. Select **mpc3004.bas**, which is in the **mpc3004 \ api** directory. Then, select **Open** to add the file to the project.

6.3.3 Borland C++ builder

To use Borland C++ builder as development tool, you should generate a **.lib** file from the **.dll** file by `implib.exe`.

```
implib mpc3004bc.lib mpc3004.dll
```

Then add the **mpc3004bc.lib** to your project and add **#include "mpc3004.h"** to main program.

Now you may use the dll functions in your program. For example, the Read Input function has the following format:

```
Status = mpc3004_read_pos (u8 CardID, u8 axis, i32 *pos);
```

where **CardID** and **axis** are input parameters, and **pos** is an output parameter.

Consider the following example:

```
u8 CardID, axis ;
```

```
i32 pos;
```

```
u32 Status;
```

```
Status = mpc3004_read_pos (u8 CardID , u8 axis , i32 &pos);
```

Initialization Function

- **mpc3004_initial**

Description

This function is used to initialize mpc3004 card. Every mpc3004 card windows application has to be initialized by this function before calling other functions.

The mpc3004_close () function is accompanied with mpc3004_initial () function to make mpc3004 card windows application program completely ended and memory fully be released.

Syntax

```
u32 status = mpc3004_initial (void)
```

Input parameters

Null

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004_close**

Description

This function is used to release windows application resource that created by mpc3004 card. Every mpc3004 card windows application has to call this function before ending the windows application program.

The mpc3004_close () function is corresponded with mpc3004_initial () function to make mpc3004 card windows application program completely ended and memory fully be released.

Syntax

```
u32 status = mpc3004_close (void)
```

Input parameters

Null

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004 info**

Description

This function is used to get mpc3004card information from device driver. It is not every necessary but may be useful for users to call this function.

Syntax

u32 status = mpc3004_info (u8 CardID,u16 *VenID , u16 *address)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Echo

VenID: return sub system ID(0x3004).

address: the card base address assigned by window system.

Status: the function working status. Reference appendix A for status value corresponds to related error code.

System Parameter Function

● mpc3004_load_steps_lines_rev

Description

This function is used to definition of pulses per motor revolution and the moving distance per motor revolution.

$$Position_{(mm,\mu m)} = Position_{(pulse)} \times \frac{MM_{(mm/rev,\mu m/rev)}}{NN_{(pulse/rev)}}$$

Syntax

u32 status = mpc3004_load_steps_lines_rev (u8 CardID,u8 Axis , u16 NN, u16 MM)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

NN: Pulses per motor revolution (= 0 ~ 65535 default value=1).

MM: Moving distance per motor revolution (= 0 ~ 65535 default value=1). The moving distance value may be based on any unit

Note:

To get a better dynamic range of programming the MM and NN value is recommended to take MM as denominator and NN as numerator and reduce their common factor. The final result fill the MM and NN value.

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

● mpc3004_read_steps_lines_rev

Description

This function is used to get axis pulses per motor revolution and the moving distance per motor revolution.

$$Position_{(mm,\mu m)} = Position_{(pulse)} \times \frac{MM_{(mm/rev,\mu m/rev)}}{NN_{(pulse/rev)}}$$

Syntax

u32 status = mpc3004_read_steps_lines_rev (u8 CardID,u8 Axis , u16 *NN, u16 *MM)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

Echo

NN: Pulses per motor revolution (= 0 ~ 65535).

MM: Moving distance per motor revolution (= 0 ~ 65535). The moving distance value may be based on any unit

Status: the function working status. Reference appendix A for status value corresponds to related error code.

● mpc3004_load_backlash

Description

This function is used to definition pulses of axis backlash compensation.

Setting this value will compensate the backlash during direction change.

Syntax

u32 status = mpc3004_load_backlash (u8 CardID,u8 Axis , u16 backlash)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

backlash: mpc3004 card pulses per motor revolution (= 0 ~ 65535 default value=0).

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

● **mpc3004_read_backlash**

Description

This function is used to get pulses of axis backlash compensation.

Syntax

u32 status = mpc3004_read_backlash (u8 CardID,u8 Axis , u16 *backlash)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

Echo

backlash: mpc3004 card pulses per motor revolution (= 0 ~ 65535).

Status: the function working status. Reference appendix A for status value corresponds to related error code.

● **mpc3004_enable_zero**

Description

This function is used to definition of axis zero phase enabled during reciprocal homing mode.

Note:

To use the zero phase input as homing reference, the input signal must connect as the following terminal assignment and clear counter output connect to servo driver is recommended. To match the output range of encoder/linear scale JP6 JP7 of wiring board also short pin 1-2 for 24V, short pin 2-3 for 5V. (Ref 5.4 Wiring board setting)

Axis	Designated I/p terminal	Designated O/p terminal
X	I0 (as X ZERO I/p)	O0 (as X CLR CNTR O/p)
Y	I1 (as Y ZERO I/p)	O1 (as Y CLR CNTR O/p)
Z	I4 (as Z ZERO I/p)	O4 (as Z CLR CNTR O/p)
A	I5 (as A ZERO I/p)	O5 (as A CLR CNTR O/p)

Syntax

u32 status = mpc3004_enable_zero (u8 CardID,u8 Axis , u8 enable)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

enable: 0 → Disable zero input.

1 → Enable zero input.

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004_set_dir_pol**

Description

This function is used to definition of axis coordinate direction.

Syntax

u32 status = mpc3004_set_dir_pol (u8 CardID,u8 Axis , u8 DIR)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

DIR: Direction settings

0 → Non inverting.(Default)

1 → Inverting.

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004_read_dir_pol**

Description

This function is used to get axis coordinate direction.

Syntax

u32 status = mpc3004_read_dir_pol (u8 CardID,u8 Axis , u8 *DIR)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

Echo

DIR: Direction settings.

0 → Non inverting.

1 → Inverting.

Status: the function working status. Reference appendix A for status value corresponds to related error code.

Motion / Homing Section

● mpc3004_read_axis_status

Description

This function is used to get axis status.

Syntax

```
u32 status = mpc3004_read_axis_status (u8 CardID,u8 Axis , u8 *ACT_rdy, u8 *POS_ok,  
u8 *DEC_ok)
```

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

Echo

ACT_rdy: Echo of execution status.

0 → Ready to accept new command request.(Not Busy)

1 → Command request in execution. (Busy)

2 → Hardware error.

POS_ok: End of positioning sets 1,else 0.(It doesn't include homing mode)

DEC_ok: End of decelerate to stop sets 1, else 0

Status: the function working status. Reference appendix A for status value corresponds to related error code.

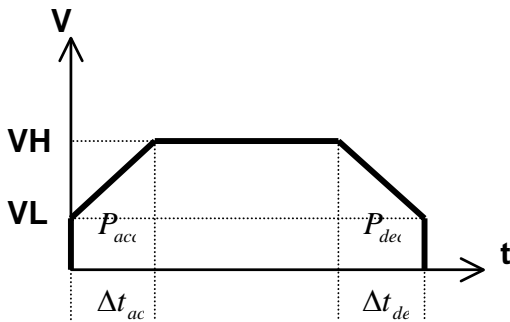
● **mpc3004 load moving speed**

Description

This function is used to set axis initial (Low) speed and final (High) speed and acceleration rate of positioning.

Note :

When you make a position command while final speed > 4M pps or start speed > final speed or acceleration rate > 4M pps/s, a 1K pps speed will override the VH and VL ,a 1pps/sec will override the ACC.



- VL : Initial (Low) speed of positioning(pps)
- VH : Final (High) speed of positioning(pps)
- ACC : Acceleration rate of positioning. (pps/sec)
- Δt_{acc} : Acceleration time of positioning.(sec)
- Δt_{dec} : Deceleration time of positioning.(sec)
- P_{acc} : Acceleration pulse number of positioning.(pulse)
- P_{dec} : Deceleration pulse number of positioning. (pulse)

$$\Delta t_{acc} = \Delta t_{dec} = \frac{VH - VL}{ACC}$$

$$P_{acc} = P_{dec} = \frac{VH^2 - VL^2}{2 \times ACC}$$

Syntax

u32 status = mpc3004_load_moving_speed (u8 CardID,u8 Axis , u32 VL ,u32 VH, u32 ACC)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

VL: Initial (Low) speed of positioning.(0pps ~4M pps ,default value=2000)

VH: Final (High) speed of positioning.(0pps ~4M pps ,default value= 20000)

ACC: Acceleration rate of positioning.(1pps/sec ~4M pps/sec , default value=10000)

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004 read moving speed**

Description

This function is used to get axis initial speed and final (High) speed and acceleration rate of positioning.

Syntax

u32 status = mpc3004_load_moving_speed (u8 CardID,u8 Axis , u32 *VL ,u32 *VH, u32 *ACC)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

Echo

VL: Initial (Low) speed of positioning.

VH: Final (High) speed of positioning.

ACC: Acceleration rate of positioning.

Status: the function working status. Reference appendix A for status value corresponds to related error code.

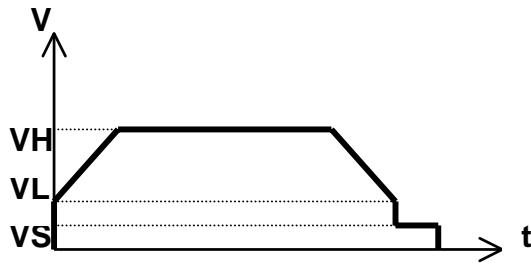
● mpc3004 load home speed

Description

This function is used to set axis initial (Low) speed and final (High) speed and acceleration rate and Creep (Slow) speed.

Note :

When you make a position command while final speed > 4M pps or start speed > final speed or acceleration rate > 4M pps/s, a 1K pps speed will override the VH and VL ,a 1pps/sec will override the ACC.



VL: Initial (Low) speed of homing (pps)

VH: Final (High) speed of homing (pps)

ACC : Acceleration rate of homing (pps/sec).

VS: Creep speed (pps/sec) for entrance and re-entrance of home(ORG) L/S region during homing.

Syntax

```
u32 status = mpc3004_load_homing_speed (u8 CardID,u8 Axis , u32 VL ,u32 VH, u32 ACC,
                                         u16 VS)
```

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

VL: Initial (Low) speed of positioning.(0pps ~4M pps ,default value=2000)

VH: Final (High) speed of positioning.(0pps ~4M pps ,default value= 20000)

ACC: Acceleration rate of positioning.(1pps/sec ~4M pps/sec , default value=10000)

VS: Creep (Slow) speed of positioning.(0pps ~65535 pps ,default value= 1000)

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004_read_home_speed**

Description

This function is used to get axis initial (Low) speed and final (High) speed and acceleration rate and creep (Slow) speed.

Syntax

```
u32 status = mpc3004_read_homing_speed (u8 CardID,u8 Axis , u32 *VL ,u32 *VH,  
u32 *ACC,u16 *VS)
```

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

VL: Initial (Low) speed of positioning.(0pps ~4M pps ,default value=2000)

VH: Final (High) speed of positioning.(0pps ~4M pps ,default value= 20000)

ACC: Acceleration rate of positioning.(1pps/sec ~4M pps/sec , default value=10000)

VS: Creep (Slow) speed of positioning.(0pps ~65535 pps ,default value= 1000)

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004_load_target_pos**

Description

This function is used to load axis desired target position value. In relative mode it is length to go, in absolute mode it is position to reach.

ABS mode: Acceptable range is -8388608(FF800000H) ~ 8388607(007FFFFFFH),

REL mode: Acceptable range is 16777215(00FFFFFFH)

Note:

Bit length of register is 32,but only 24 bit is meaningful. Negative value should use 2's complement.

Syntax

```
u32 status = mpc3004_load_target_pos(u8 CardID,u8 Axis , i32 pos)
```

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

pos: Target position.(ABS mode:-8388608~8388607, REL mode:0~16777215, default value= 10000)

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004_read_target_pos**

Description

This function is used to get axis desired target position value.

Note:

Bit length of register is 32, but only 24 bit is meaningful. Negative value should use 2's complement.

Syntax

```
u32 status = mpc3004_read_target_pos(u8 CardID, u8 Axis, i32 *pos)
```

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

Echo

pos: Target position.

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004_load_pos**

Description

This function is used to load axis current position value.

Syntax

```
u32 status = mpc3004_load_pos(u8 CardID, u8 Axis, i32 pos)
```

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

pos: Current position. (-2147483648 ~ 2147483647)

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004_reset_pos**

Description

This function is used to clear axis current position value.

Syntax

```
u32 status = mpc3004_reset_pos(u8 CardID,u8 Axis)
```

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004_read_pos**

Description

This function is used to get axis current position value.

Syntax

```
u32 status = mpc3004_read_pos(u8 CardID,u8 Axis,i32 *pos)
```

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

Echo

pos: Current position.

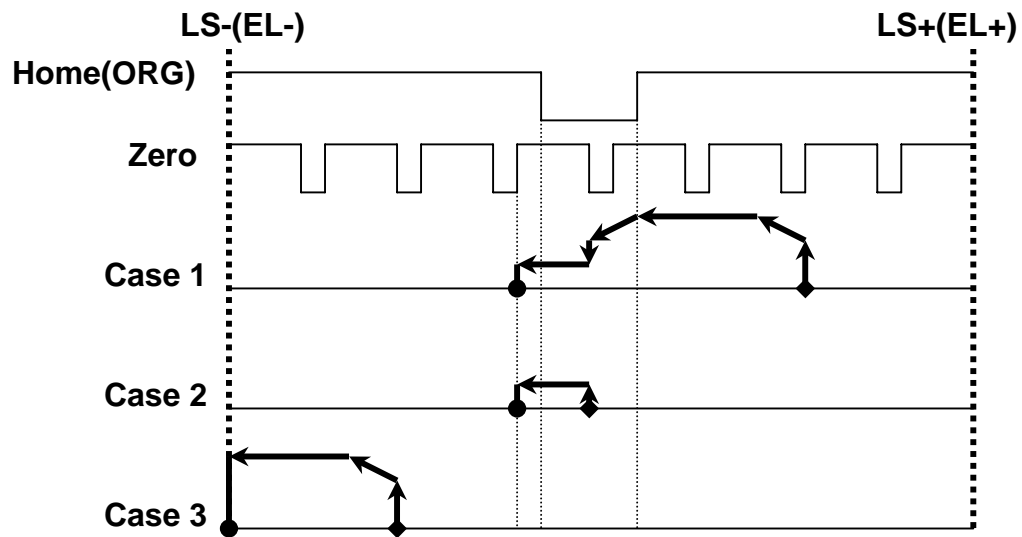
Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004 set act mode**

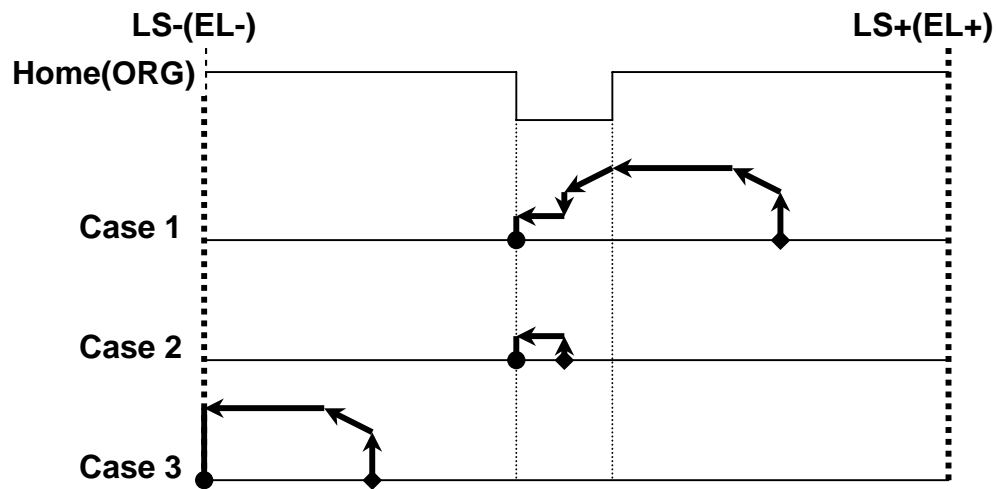
Description

This function is used to set axis active mode .

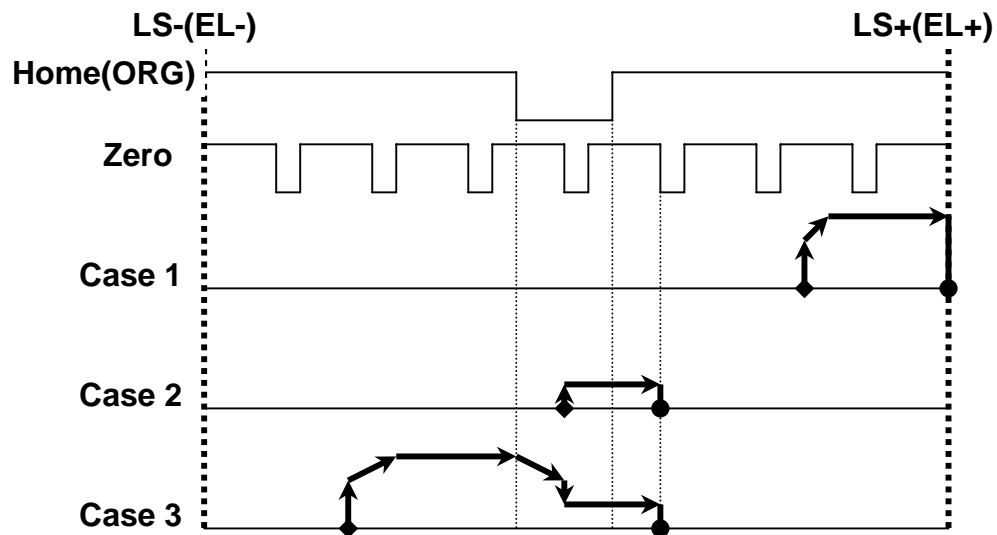
Mode 1 → NEG_HOME_MODE: Homing in negative direction



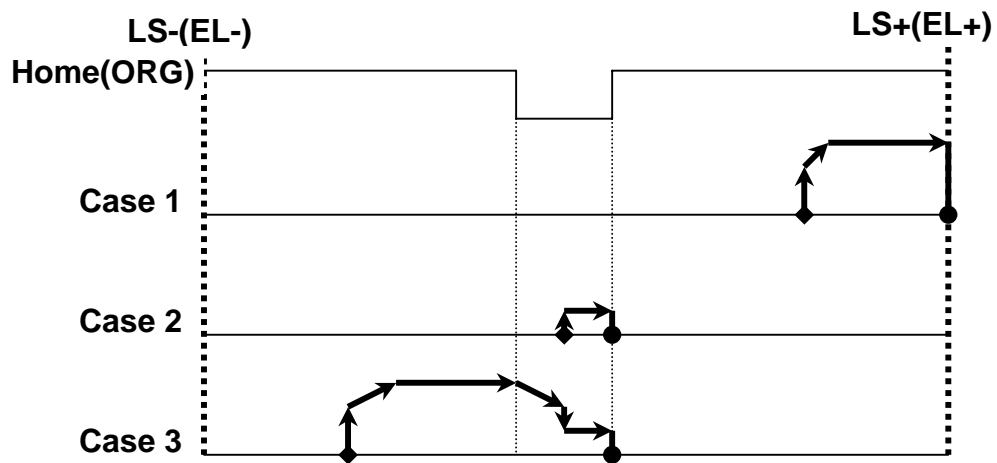
With encoder zero phase



Mode 2 → POS_HOME_MODE: Homing in positive direction



With encoder zero phase

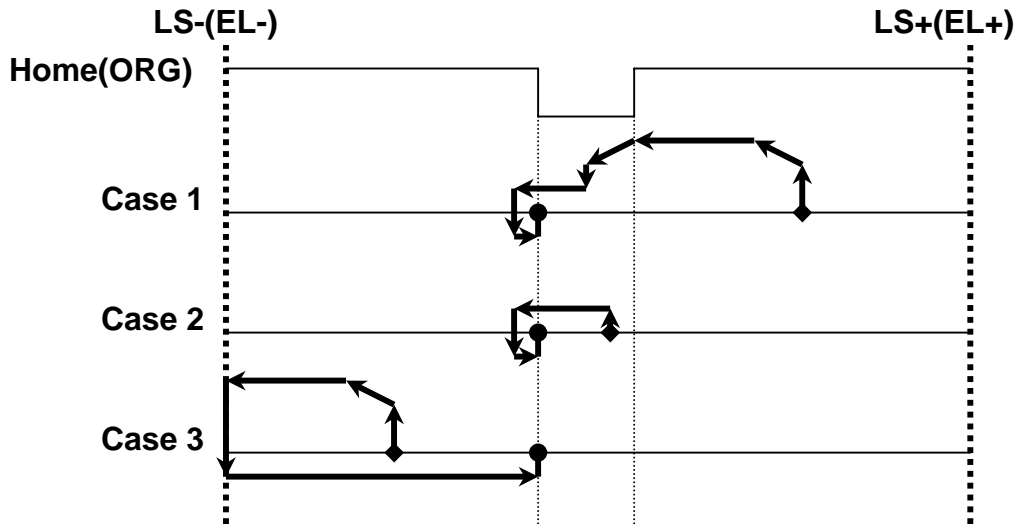
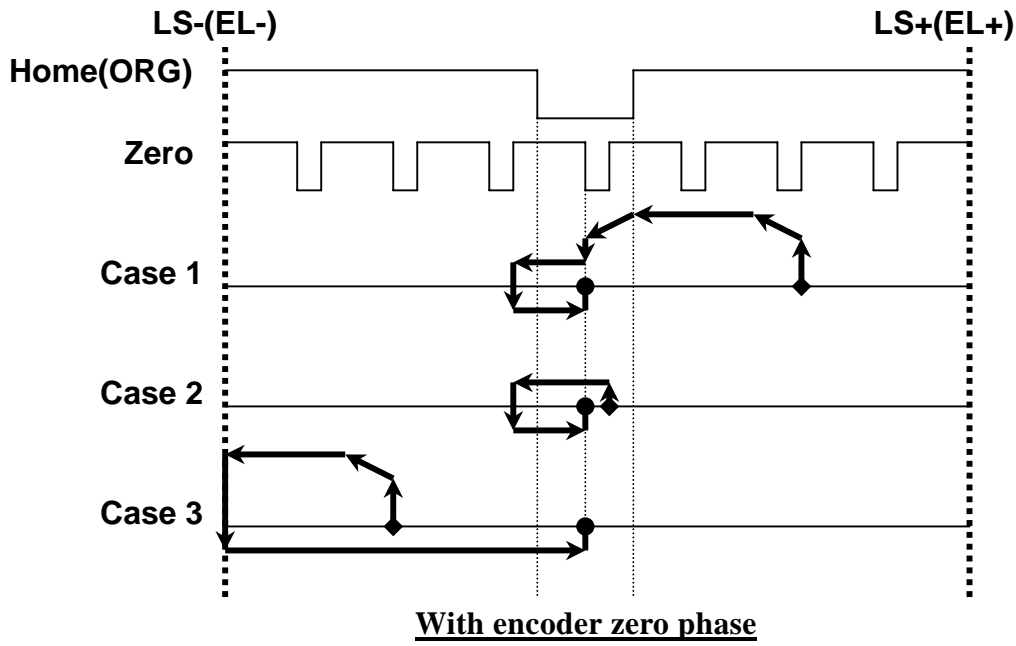


Mode 3 → ABS_MOVE_MODE: Absolute positioning

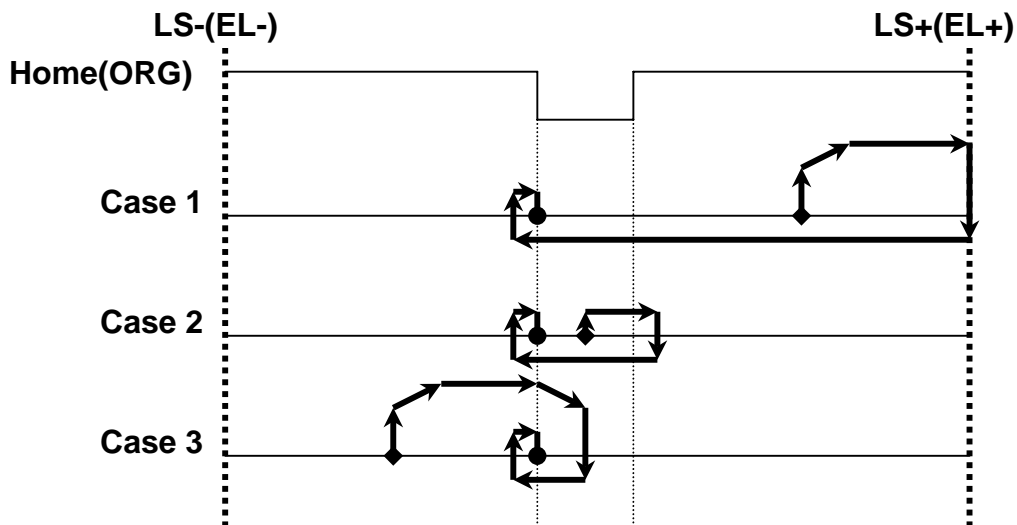
Mode 4 → REL_POS_MOVE_MODE: Positive relative positioning

Mode 5 → REL_NEG_MOVE_MODE: Negative relative positioning

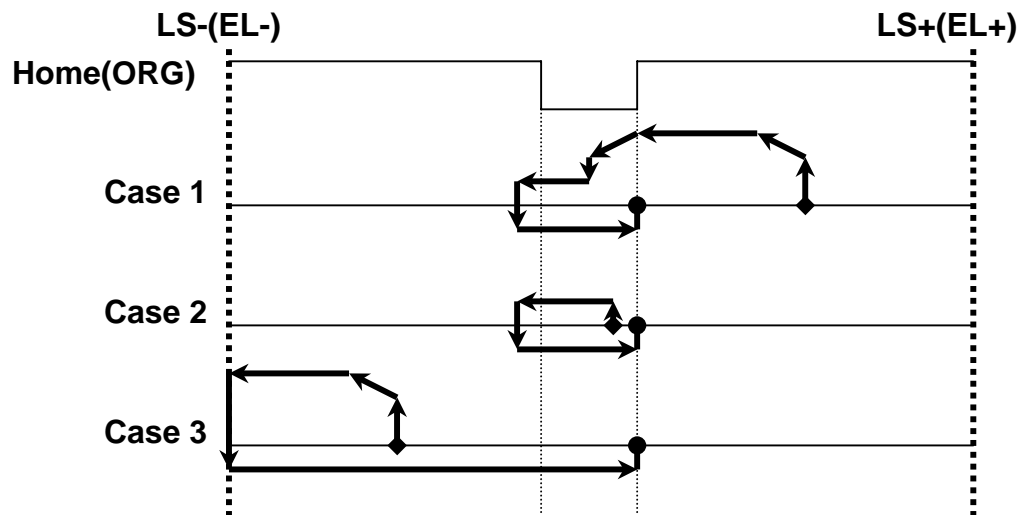
Mode 6 → RET_SNEN_HOME_MODE: Reciprocal homing



Mode 7 → RET_SPEN_HOME_MODE: Reciprocal homing in positive direction



Mode 8 → RET_SNEP_HOME_MODE: Reciprocal homing in negative direction



Syntax

u32 status = mpc3004_set_act_mode(u8 CardID,u8 Axis,u8 mode)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

mode : Active mode (1~8)

1 → NEG_HOME_MODE: Homing in negative direction

2 → POS_HOME_MODE: Homing in positive direction

3 → ABS_MOVE_MODE: Absolute positioning

4 → REL_POS_MOVE_MODE: Positive relative positioning

5 → REL_NEG_MOVE_MODE: Negative relative positioning

6 → RET_SNEP_HOME_MODE: Reciprocal homing

7 → RET_SNEP_HOME_MODE: Reciprocal homing in positive direction

8 → RET_SNEP_HOME_MODE: Reciprocal homing in negative direction

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

● **mpc3004 read act mode**

Description

This function is used to get axis active mode .

Syntax

u32 status = mpc3004_read_act_mode(u8 CardID,u8 Axis,u8 *mode)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

Echo

mode : Active mode (1~8)

1→ NEG_HOME_MODE: Homing in negative direction

2→ POS_HOME_MODE: Homing in positive direction

3→ ABS_MOVE_MODE: Absolute positioning

4→ REL_POS_MOVE_MODE: Positive relative positioning

5→ REL_NEG_MOVE_MODE: Negative relative positioning

6→ RET_SNEN_HOME_MODE: Reciprocal homing

7→ RET_SPEN_HOME_MODE: Reciprocal homing in positive direction

8→ RET_SNEP_HOME_MODE: Reciprocal homing in negative direction

Status: the function working status. Reference appendix A for status value corresponds to related error code.

● **mpc3004 multi start**

Description

This function allows you to start multiple axes simultaneously.

Syntax

u32 status = mpc3004_multi_statr(u8 CardID,u8 ACT)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

ACT: Bitmap of axes to start. (0~15)

ACT has the following values and format:

1→Start Axis 0→ Do Not Start Axis

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ACT	N/A	N/A	N/A	N/A	Axis 3 A	Axis 2 Z	Axis 1 Y	Axis 0 X

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004_start_motion**

Description

Starts motion for the axis specified.

Syntax

u32 status = mpc3004_start_motion(u8 CardID,u8 Axis)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004_stop_motion**

Description

Sets the stop modes and stop motion on a specific axis.

Syntax

u32 status = mpc3004_stop_motion(u8 CardID,u8 Axis,u8 Mode)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

Mode: stop mode (0 or 2)

0→ EMG_STOP: Emergent Stop (Halt Stop)

2→ DEC_STOP : Decelerate to Stop

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004 set free speed range**

Purpose : Set the free speed range of designated axis.

Format : `u32 status = mpc3004_set_free_speed_range(u8 CardID, u8 Axis,u32 max_speed)`

Input:

Name	Type	Description
CardID	u8	assigned by DIP (rotary) SW
Axis	u8	0:X 1:Y 2:Z 3:A
max_speed	u32	the max speed range of speed $31,000 \leq \text{max_speed} \leq 4M$

Note:

To set the maximum speed range is to specify the usable range of the speed parameter, an adequate speed range can give you better accuracy, since the digital hardware is finite word length.

- **mpc3004 free speed change**

Purpose : Set/change the free speed of designated axis.

Format : `u32 status = mpc3004_free_speed_change(u8 CardID, u8 Axis,i32 Speed)`

Input:

Name	Type	Description
CardID	u8	assigned by DIP (rotary) SW
Axis	u8	0:X 1:Y 2:Z 3:A
Speed	i32	speed, must in the range of max_speed. The sign of speed defines the direction of rotation.

Input / Output Section

● mpc3004_set_g_input_pol

Description

This function is used to set general input port polarity.

Syntax

```
u32 status = mpc3004_set_g_input_pol(u8 CardID,u8 data)
```

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

data: Bitmap of polarity (0~255)

data has the following values and format:

0 → Non inverting. (Default) 1 → Inverting.

data	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	IN 7	IN 6	IN 5	IN 4	IN 3	IN 2	IN 1	IN 0

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

● mpc3004_read_g_input_pol

Description

This function is used to get general input port polarity.

Syntax

```
u32 status = mpc3004_read_g_input_pol(u8 CardID,u8 *data)
```

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Echo

data: Bitmap of polarity. (0~255)

data has the following values and format:

0 → Non inverting. 1 → Inverting.

data	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	IN 7	IN 6	IN 5	IN 4	IN 3	IN 2	IN 1	IN 0

Status: the function working status. Reference appendix A for status value corresponds to related error code.

● **mpc3004_read_g_input**

Description

This function is used to read general input port status.

Calling mpc3004_set_g_input_pol () can change the related polarity.

Syntax

u32 status = mpc3004_read_g_input (u8 CardID,u8 *data)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Echo

data: general input port status. (0~255)

data has the following values and format:

0 → inactive. 1 → active.

data	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	IN 7	IN 6	IN 5	IN 4	IN 3	IN 2	IN 1	IN 0

Status: the function working status. Reference appendix A for status value corresponds to related error code.

● **mpc3004_read_g_input_point**

Description

This function is used to read general input point status.

Calling mpc3004_set_g_input_pol () can change the related polarity.

Syntax

u32 status = mpc3004_read_g_input_point (u8 CardID,u8 Bit , u8 *state)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Bit: Designated which general input point to be read. (0~7)

data	7	6	5	4	3	2	1	0
	IN 7	IN 6	IN 5	IN 4	IN 3	IN 2	IN 1	IN 0

Echo

Stat: 0 → inactive 1 → active

Status: the function working status. Reference appendix A for status value corresponds to related error code.

● **mpc3004 set g output pol**

Description

This function is used to set general output port polarity.

Syntax

u32 status = mpc3004_set_g_output_pol(u8 CardID,u8 data)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

data: Bitmap of polarity (0~255)

data has the following values and format:

0 → Non inverting. (Default) 1 → Inverting.

data	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	OUT 7	OUT 6	OUT 5	OUT 4	OUT 3	OUT 2	OUT 1	OUT 0

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

● **mpc3004 read g output pol**

Description

This function is used to get general output port polarity.

Syntax

u32 status = mpc3004_read_g_output_pol(u8 CardID,u8 *data)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Echo

data: Bitmap of polarity. (0~255)

data has the following values and format:

0 → Non inverting. 1 → Inverting.

data	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	OUT 7	OUT 6	OUT 5	OUT 4	OUT 3	OUT 2	OUT 1	OUT 0

Status: the function working status. Reference appendix A for status value corresponds to related error code.

● mpc3004_read_g_output

Description

This function is used to read general output port status.

Calling `mpc3004_set_g_output_pol ()` can change the related polarity.

Syntax

```
u32 status = mpc3004_read_g_output (u8 CardID,u8 *data)
```

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Echo

data: output port status. (0~255)

data has the following values and format:

0 → inactive. 1 → active.

data	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	OUT 7	OUT 6	OUT 5	OUT 4	OUT 3	OUT 2	OUT 1	OUT 0

Status: the function working status. Reference appendix A for status value corresponds to related error code.

● mpc3004_set_g_output

Description

This function is used to set general output port status.

Calling `mpc3004_set_g_output_pol ()` can change the related polarity.

Syntax

```
u32 status = mpc3004_set_g_output (u8 CardID,u8 *data)
```

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Echo

data: output port status. (0~255)

data has the following values and format:

0 → inactive. 1 → active.

data	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	OUT 7	OUT 6	OUT 5	OUT 4	OUT 3	OUT 2	OUT 1	OUT 0

Status: the function working status. Reference appendix A for status value corresponds to related error code.

● **mpc3004 read g output point**

Description

This function is used to read general output point status.

Calling mpc3004_set_g_output_pol () can change the related polarity.

Syntax

u32 status = mpc3004_read_g_output_point (u8 CardID,u8 Bit , u8 *State)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Bit: Designated which general output point to be read. (0~7)

	7	6	5	4	3	2	1	0
data	OUT 7	OUT 6	OUT 5	OUT 4	OUT 3	OUT 2	OUT 1	OUT 0

Echo

Stat: 0 → inactive..... 1 → active

Status: the function working status. Reference appendix A for status value corresponds to related error code.

● **mpc3004 set g output point**

Description

This function is used to set general output point status.

Calling mpc3004_set_g_output_pol () can change the related polarity.

Syntax

u32 status = mpc3004_set_g_output_point (u8 CardID,u8 Bit , u8 State)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Bit: Designated which general output point to be set. (0~7)

	7	6	5	4	3	2	1	0
data	OUT 7	OUT 6	OUT 5	OUT 4	OUT 3	OUT 2	OUT 1	OUT 0

Stat: 0 → inactive..... 1 → active

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004 set limit pol**

Description

Sets the polarity of the limit switches (forward, reverse, and home(ORG)).

Syntax

u32 status = mpc3004_set_limit_pol(u8 CardID,u8 Axis,u8 LSP,u8 LSN,u8 HOME)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

LSP: Forward (LS+(EL+)) polarity. (1 → Inverting (Default); 0 → Non inverting)

LSN: Reverse (LS-(EL-)) polarity. (1 → Inverting (Default); 0 → Non inverting)

HOME(ORG): Home(ORG) polarity. (1 → Inverting (Default); 0 → Non inverting)

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004 read limit pol**

Description

Read the polarity of the limit switches (forward, reverse, and home(ORG)).

Syntax

u32 status = mpc3004_read_limit_pol(u8 CardID,u8 Axis,u8 *LSP,u8 *LSN,u8 *HOME)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

Echo

LSP: Forward (LS+(EL+)) polarity. (1 → Inverting; 0 → Non inverting)

LSN: Reverse (LS-(EL-)) polarity. (1 → Inverting; 0 → Non inverting)

HOME(ORG): Home(ORG) polarity. (1 → Inverting; 0 → Non inverting)

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004 enable limit**

Description

Enable/disable the limit switches (forward, reverse, and home(ORG)).

Note :

Disable the Home(ORG) L/S check do not effect the homing function.

Enable means the card will alarm and stop the pulse during positioning.

Syntax

u32 status = mpc3004_enable_limit(u8 CardID,u8 Axis,u8 LSP,u8 LSN,u8 HOME)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

LSP: Forward limit switch (LS+(EL+)). (1 → Enable (Default); 0 → Disable)

LSN: Reverse limit switch (LS-(EL-)). (1 → Enable (Default); 0 → Disable)

HOME(ORG): Home(ORG) limit switch. (1 → Enable; 0 → Disable (Default))

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004 read enable limit**

Description

Read enable/disable of the limit switches (forward, reverse, and home(ORG)).

Note :

Disable the Home(ORG) L/S check do not effect the homing function.

Enable means the card will alarm and stop the pulse during positioning.

Syntax

u32 status = mpc3004_read_enable_limit(u8 CardID,u8 Axis,u8 *LSP,u8 *LSN,u8 *HOME)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

Echo

LSP: Forward limit switch (LS+(EL+)). (1 → Enable; 0 → Disable)

LSN: Reverse limit switch (LS-(EL-)). (1 → Enable; 0 → Disable)

HOME(ORG): Home(ORG) limit switch. (1 → Enable; 0 → Disable)

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004_read_limit_stat**

Description

Read status of the limit switches (forward, reverse, and home(ORG)).

Syntax

u32 status = mpc3004_read_enable_limit(u8 CardID,u8 Axis,u8 *LSP,u8 *LSN,u8 *HOME)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Axis: designated which axis to be set. (=0, 1, 2, 3)

Echo

LSP: Forward limit switch (LS+(EL+)). (0 → inactive; 1 → active)

LSN: Reverse limit switch (LS-(EL-)). (0 → inactive; 1 → active)

HOME(ORG): Home(ORG) limit switch. (0 → inactive; 1 → active)

Status: the function working status. Reference appendix A for status value corresponds to related error code.

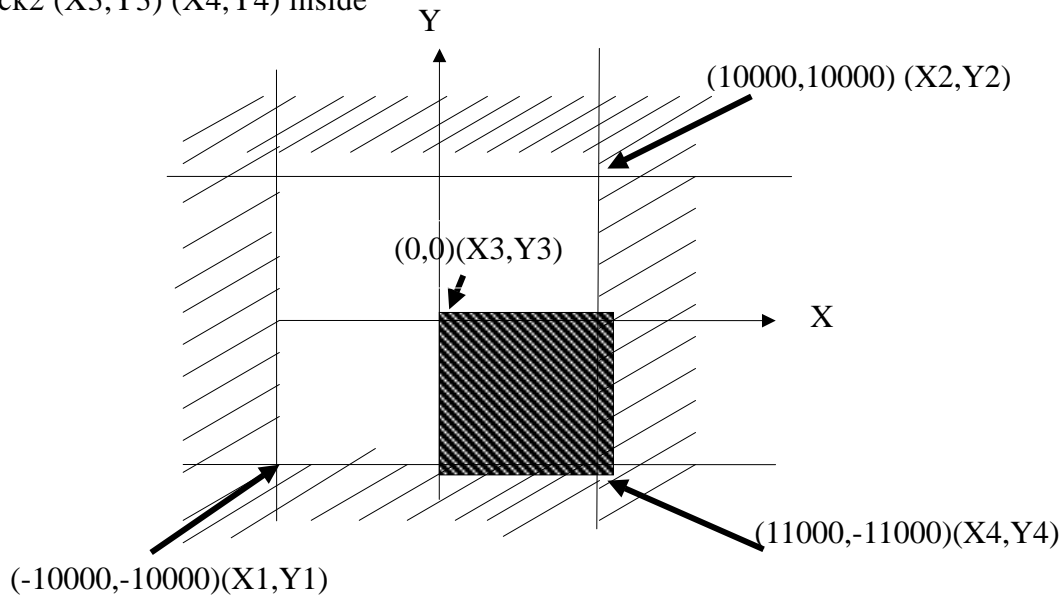
Soft limit section

MPC3004 have 8 user configurable zone (block0~7) to define the inside or outside of permissible working area. Once any axis enters the forbidden zone MPC card will stop all the pulse out .

For example we want to set the shaded area as forbidden zone, the corner coordinate is P1(X1,Y1), P2(X2,Y2),P3(X3,Y3),P4(X4,Y4).

We set block0 (X1, X2) outside, block1 (Y1,Y2) outside

Block2 (X3,Y3) (X4,Y4) inside



```
mpc3004_load_soft_limit(CardID, Block0, Axis_X, -10000, 10000, 1,1);  
mpc3004_load_soft_limit(CardID, Block1, Axis_Y, -10000, 10000, 1,1);  
mpc3004_load_soft_limit(CardID, Block2, Axis_X, 0, -11000, 0,1);  
mpc3004_load_soft_limit(CardID, Block2, Axis_Y, -11000, 0, 0,1);
```

● mpc3004_read_block_status

Description

Read soft limit block were normal or alarm.

Syntax

```
u32 status = mpc3004_read_block_status(u8 CardID,u8 *b0,u8 *b1,u8 *b2,u8 *b3,u8 *b4,  
u8 *b5,u8 *b6,u8 *b7)
```

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Echo

b0~7: block status. (0 → Normal; 1 → Alarm)

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004_read_block_axis_status**

Description

This function is used to read axes function enable/disable after software limit check.

Syntax

u32 status = mpc3004_read_block_axis_status(u8 CardID,u8 *X,u8 *Y,u8 *Z,u8 *A)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Echo

X~A: Axis status. (0 → Normal; 1 → Alarm)

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004_read_block_alarm_dir**

Description

This function is used to read moving direction during block alarm generated.

Syntax

u32 status = mpc3004_read_block_alarm_dir(u8 CardID,u8 *X,u8 *Y,u8 *Z,u8 *A)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Echo

X~A: Axis status. (0 → negative direction; 1 → positive direction)

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004 load soft limit**

Description

This function is used to set soft limit parameter.

Syntax

```
u32 status = mpc3004_load_soft_limit(u8 CardID,u8 Block,u8 Axis,i32 Ndata,i32 Pdata,  
                                     u8 Inhibit,u8 enable)
```

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Block: designated which block to be set. (=0~7)

Axis: designated which axis to be set. (=0, 1, 2, 3)

Ndata: Soft limit position 1. (= -2147483648 ~ +2147483647)

Pdata: Soft limit position 2. (= -2147483648 ~ +2147483647)

Inhibit: Inhibit setting (1 → Outside inhibit; 0 → Inside inhibit)

Enable: Enable/disable axis on this block (1 → Enable; 0 → Disable)

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004 read soft limit**

Description

This function is used to read soft limit parameter.

Syntax

```
u32 status = mpc3004_read_soft_limit(u8 CardID,u8 Block , u8 Axis , i32 *Ndata , i32 *Pdata,  
                                     u8 *Inhibit , u8 *enable)
```

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Block: designated which block to be set. (=0~7)

Axis: designated which axis to be set. (=0, 1, 2, 3)

Echo

Ndata: Soft limit position 1.

Pdata: Soft limit position 2.

Inhibit: Inhibit setting (1 → Outside inhibit; 0 → Inside inhibit)

Enable: Enable/disable axis on this block (1 → Enable; 0 → Disable)

Status: the function working status. Reference appendix A for status value corresponds to related error code.

Security Section

Once you have enable the security function , you must unlock every time the card is reset to function correctly.

Note:

If you give the card more than 10 times error unlock , SEC_CMD will set to 3 to signal no more unlock command is acceptable. MPC will not function any more. Please send back to our factory to unlock.

● mpc3004_read_secret_status

Description

This function is used to read security status. If unlock successful (open return value=1) then MPC may function correctly, else moving function will was extraordinary.

Syntax

```
u32 status = mpc3004_read_secret_status(u8 CardID,u8 *open,u8 *enable)
```

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Echo

open: Lock status. (0 → Locked ; 1 → Unlock successful / security disable)

enable: Security status. (0 → Disable ;1 → Enable)

Status: the function working status. Reference appendix A for status value corresponds to related error code.

● mpc3004_open_secret_code

Description

This function is used to unlock.

Syntax

```
u32 status = mpc3004_open_secret_code(u8 CardID,u16 O0,u16 O1,u16 O2,u16 O3,u16 O4)
```

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

O0: Old password 0. (0 ~ 65535)

O1: Old password 1. (0 ~ 65535)

O2: Old password 2. (0 ~ 65535)

O3: Old password 3. (0 ~ 65535)

O4: Old password 4. (0 ~ 65535)

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004 set secret code**

Description

This function is used to enable security function and set security password.

Syntax

u32 status = mpc3004_set_secret_code(u8 CardID,u16 N0,u16 N1,u16 N2,u16 N3,u16 N4)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

N0: New password 0. (0 ~ 65535)

N1: New password 1. (0 ~ 65535)

N2: New password 2. (0 ~ 65535)

N3: New password 3. (0 ~ 65535)

N4: New password 4. (0 ~ 65535)

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004 change secret code**

Description

This function is used to change security password.

Syntax

u32 status = mpc3004_change_secret_code(u8 CardID,u16 O0,u16 O1,u16 O2,u16 O3,u16 O4,
u16 N0,u16 N1,u16 N2,u16 N3,u16 N4)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

O0: Old password 0. (0 ~ 65535)

O1: Old password 1. (0 ~ 65535)

O2: Old password 2. (0 ~ 65535)

O3: Old password 3. (0 ~ 65535)

O4: Old password 4. (0 ~ 65535)

N0: New password 0. (0 ~ 65535)

N1: New password 1. (0 ~ 65535)

N2: New password 2. (0 ~ 65535)

N3: New password 3. (0 ~ 65535)

N4: New password 4. (0 ~ 65535)

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

- **mpc3004 disable secret code**

Description

This function is used to disable security function.

Syntax

u32 status = mpc3004_disable_secret_code(u8 CardID,u16 O0,u16 O1,u16 O2,u16 O3,u16 O4)

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

O0: Old password 0. (0 ~ 65535)

O1: Old password 1. (0 ~ 65535)

O2: Old password 2. (0 ~ 65535)

O3: Old password 3. (0 ~ 65535)

O4: Old password 4. (0 ~ 65535)

Echo

Status: the function working status. Reference appendix A for status value corresponds to related error code.

Miscellaneous section

- **mpc3004_read_counter**

Description

This function is used to read MPC circular timer (0~60000mS).

Syntax

```
u32 status = mpc3004_read_counter(u8 CardID,u16 *count)
```

Input parameters

CardID: assigned card number by dipswitch setting on board. (= 0 ~ 15)

Echo

count: circular timer (0~60000mS)

Status: the function working status. Reference appendix A for status value corresponds to related error code.

Linear interpolation section

The following functions are added to implement the **Linear interpolation** for upgrade of MPC3004 motion control card. Each function has the sub-functions as follows:

1. Check the availability of motion axis and returns error code.
2. Calculate the vector speed on each motion axis.
3. Set the motion mode (absolute or relative) of each motion axis.
4. Set the motion profile.
5. Set the destination point of motion.
6. Command to move. If success returns 0, else error code returns.

● **mpc3004_move_P2P**

Purpose: Command to move at positioning mode. If the call returns success, the motion is under the control of the card.

Format : **u32 Status = mpc3004_move_P2P(u8 CardID,u8 Axis,i32 POS,u8 Pos_mode, u32 Vl,u32 Vh,u32 Vacc)**

Input:

Name	Type	Description
CardID	u8	assigned by DIP (rotary) SW
Axis	u8	0:X axis 1:Y axis 2:Zaxis 3:Aaxis
POS	i32	Destination position in absolute positioning, distance to go in relative positioning.
Pos_mode	u8	0: absolute positioning mode. 1: relative positioning mode. POS is positive for increment movement, negative for decrement movement
Vl	u32	Initial (low) speed (in pps).
Vh	u32	Final (high) speed (in pps)
Vacc	u32	Acceleration rate (in pps/sec)

● **mpc3004 move_line2**

Purpose: Command to move at linear interpolation mode. If the call returns success, the motion is under the control of the card.

Format : **u32 Status = mpc3004_move_line2(u8 CardID, u8 SetAxes, i32 POS0, i32 POS1, u8 Pos_mode, u32 Vl, u32 Vh, u32 Vacc)**

Input:

Name	Type	Description
CardID	u8	assigned by DIP (rotary) SW
SetAxis	u8	0:X,Y axes 1:X,Z axes 2:X,A axes 3:Y,Z axes 4:Y,A axes 5:Z,A axes
POS0 POS1	i32	Destination position in absolute positioning, distance to go in relative positioning.
Pos_mode	u8	0: absolute positioning mode. 1: relative positioning mode. POSn is positive for increment movement, negative for decrement movement
Vl	u32	Initial (low) speed (in pps).
Vh	u32	Final (high) speed (in pps)
Vacc	u32	Acceleration rate (in pps/sec)

Note: Relation of active axes designation and POS0,POS1 in linear motion

Designation code	Active axes	POS0	POS1
0	X,Y	X	Y
1	X,Z	X	Z
2	X,A	X	A
3	Y,Z	Y	Z
4	Y,A	Y	A
5	Z,A	Z	A

● **mpc3004 move_line3**

Purpose: Command to move at 3-axes linear interpolation mode. If the call returns success, the motion is under the control of the card.

Format : **u32 Status = mpc3004_move_line3(u8 CardID, u8 SetAxes, i32 POS0, i32 POS1, i32 POS2,u8 Pos_mode,u32 Vl,u32 Vh,u32 Vacc)**

Input:

Name	Type	Description
CardID	u8	assigned by DIP (rotary) SW
SetAxis	u8	0:X,Y,Z axes 1:X,Y,A axes 2:X,Z,A axes 3:Y,Z,A axes
POS0 POS1 POS2	i32	Destination position in absolute positioning, distance to go in relative positioning.
Pos_mode	u8	0: absolute positioning mode. 1: relative positioning mode. POSn is positive for increment movement, negative for decrement movement
Vl	u32	Initial (low) speed (in pps).
Vh	u32	Final (high) speed (in pps)
Vacc	u32	Acceleration rate (in pps/sec)

Note: Relation of active axes designation and POS0,POS1 in linear motion

Designation code	Active axes	POS0	POS1	POS2
0	X,Y,Z	X	Y	Z
1	X,Y,A	X	Y	A
2	X,Z,A	X	Z	A
3	Y,Z,A	Y	Z	A

● **mpc3004 move line4**

Purpose: Command to move at 4-axes linear interpolation mode. If the call returns success, the motion is under the control of the card.

Format : **u32 Status = mpc3004_move_line4(u8 CardID, i32 POS0, i32 POS1, i32 POS2, i32 POS3, u8 Pos_mode, u32 Vl, u32 Vh, u32 Vacc)**

Input:

Name	Type	Description
CardID	u8	assigned by DIP (rotary) SW
POS0 POS1 POS2 POS3	i32	Destination position in absolute positioning, distance to go in relative positioning. POS0: X axis POS1: Y axis POS2: Z axis POS2: Z axis
Pos_mode	u8	0: absolute positioning mode. 1: relative positioning mode. POSn is positive for increment movement, negative for decrement movement
Vl	u32	Initial (low) speed (in pps).
Vh	u32	Final (high) speed (in pps)
Vacc	u32	Acceleration rate (in pps/sec)

6.5 Dll list

	Function Name	Description
1	mpc3004_initial()	Software initializaton
2	mpc3004_close()	Software close
3	mpc3004_info()	Get card information
4	mpc3004_load_steps_lines_rev()	Define the pulses per motor revolution and the moving distance per motor revolution
5	mpc3004_read_steps_lines_rev()	The pulses per motor revolution and the pitch per motor revolution readback
6	mpc3004_load_backlash()	Define the pulses of backlash compensation
7	mpc3004_read_backlash()	The pulses of backlash compensation readback
8	mpc3004_enable_zero()	Enable / disable zero phase input as reciprocal homing homed condition.
9	mpc3004_set_dir_pol()	Set axis coordinate direction
10	mpc3004_read_dir_pol()	Axis coordinate direction readback
11	mpc3004_read_axis_status()	Axis status readback
12	mpc3004_load_moving_speed()	Set moving speed
13	mpc3004_read_moving_speed()	Moving speed readback
14	mpc3004_load_home_speed()	Set homing speed
15	mpc3004_read_home_speed()	Homing speed readback
16	mpc3004_load_target_pos()	Load value to target position
17	mpc3004_read_target_pos()	Read target position
18	mpc3004_load_pos()	Load value to current position
19	mpc3004_reset_pos()	Reset current position
20	mpc3004_read_pos()	Read current position
21	mpc3004_set_act_mode()	Set active mode
22	mpc3004_read_act_mode()	Active mode readback
23	mpc3004_multi_start()	Start multiple axes simultaneously
24	mpc3004_start_motion()	Start single axis simultaneously
25	mpc3004_stop_motion()	Stop single axis simultaneously
26	mpc3004_set_free_speed_range()	Set the free speed range of designated axis.
27	mpc3004_free_speed_change()	Set/change the free speed of designated axis.
28	mpc3004_set_g_input_pol()	Set general input polarity.
29	mpc3004_read_g_input_pol()	General input polarity readback.
30	mpc3004_read_g_input()	General input port data readback
31	mpc3004_read_g_input_point()	Read general input point status readback
32	mpc3004_set_g_output_pol()	Set general output polarity
33	mpc3004_read_g_output_pol()	General output polarity readback

34	mpc3004_read_g_output()	General output port data readback
35	mpc3004_set_g_output()	Write general output command
36	mpc3004_read_g_output_point()	General output point status readback
37	mpc3004_set_g_output_point()	Write general output point command
38	mpc3004_set_limit_pol()	Set special purpose input point polarity
39	mpc3004_read_limit_pol()	Special purpose input point polarity readback
40	mpc3004_enable_limit()	Enable / disable special purpose input point
41	mpc3004_read_enable_limit()	Enable / disable special purpose input point readback
42	mpc3004_read_limit_stat()	Special purpose input point status readback
43	mpc3004_read_block_status()	Read soft limit block were normal or alarm
44	mpc3004_read_block_axis_status()	Read axes function enable/disable after software limit check
45	mpc3004_read_block_alarm_dir()	Read moving direction during block alarm generated
46	mpc3004_load_soft_limit()	Set soft limit parameter.
47	mpc3004_read_soft_limit()	Soft limit parameter readback
48	mpc3004_read_secret_status()	Read security status
49	mpc3004_open_secret_code()	Unlock the security.
50	mpc3004_set_secret_code()	Enable the security.
51	mpc3004_change_secret_code()	Change password
52	mpc3004_disable_secret_code()	Disable the security.
53	mpc3004_read_counter()	Read MPC3004 card timer
54	mpc3004_move_P2P()	Command to move at positioning mode.
55	mpc3004_move_line2()	Command to move at linear interpolation mode.
56	mpc3004_move_line3()	Command to move at 3-axes linear interpolation mode.
57	mpc3004_move_line4()	Command to move at 4-axes linear interpolation mode.

7. MPC-3004 Error codes summary

7.1 MPC3004 Error codes table

Error Code	Symbolic Name	Description
0	JSDRV_NO_ERROR	No error.
2	JSDRV_INIT_ERROR	Driver initial error
100	DEVICE_RW_ERROR	Device Read/Write error
101	JSDRV_NO_CARD	No MPC3004 card on the system.
102	JSDRV_DUPLICATE_ID	MPC3004 CardID duplicate error.
300	JSMPC_ID_ERROR	Function input parameter error. CardID setting error, CardID doesn't match the DIP SW setting
301	JSMPC_VH_GT_4M_ERROR	Function input parameter error. (VH > 4M PPS).
302	JSMPC_VL_GT_VH_ERROR	Function input parameter error. (VL > VH)
303	JSMPC_ACC_GT_4M_ERROR	Function input parameter error. (ACC > 4M PPS/Sec)
304	JSMPC_LOCK_MOTION_ERROR	You give the card more than 10 times error unlock. MPC will not function any more. Please send back to our factory to unlock.
305	JSMPC_SECRET_CODE_ERROR	Security password error and unlock fail.
306	JSMPC_AXIS_ERROR	Function input parameter error. (Axis > 3)
307	JSMPC_BIT_ERROR	Function input parameter error. (Bit > 7)
308	JSMPC_ACT_MODE_ERROR	Function input parameter error. "Mode" parameter out of range.
310	JSMPC_MULIT_START_ERROR	mpc3004_multi_start(), "ACT" parameter out of range. (ACT > 15)
311	JSMPC_BLOCK_SET_ERROR	Function input parameter error. (Block > 7)
312	JSMPC_STOP_CMD_ERROR	Function input parameter error. MODE ≠ 0(EMG_STOP) or 2(DEC_STOP)
313	JSMPC_CMD_ERROR	Stop Command error. Command to decelerate stop during stop Command during decelerate to stop, but not yet stopped.