MPC3024A/AC

4-axis Motion Control Card

Software Manual (V2.1)

健昇科技股份有限公司

JS AUTOMATION CORP.

新北市汐止區中興路 100 號 6 樓
6F., No.100, Zhongxing Rd.,
Xizhi Dist., New Taipei City, Taiwan
TEL：+886-2-2647-6936
FAX：+886-2-2647-6940
http://www.automation.com.tw
http://www.automation-js.com/
E-mail：control.cards@automation.com.tw

Correction record

Version	Record
1.0	wdf3024A.sys v 1.0, wdf3024A_64.sys v 1.0, MPC3024A.dll v.10, MPC3024A_64.dll
2.0	v.21 MPC3024A.dll,MPC3024A_64.dll Add new DA function
	MPC3024AC_DA_motion_config_set( )
	MPC3024AC_DA_motion_config_read( )
	MPC3024AC_DA_motion_control_set( )
	MPC3024AC_DA_motion_control_read( )
	MPC3024AC_DA_Arbitrary_Waveform_data_set( )
	MPC3024AC_DA_Arbitrary_Waveform_data_read( )
	MPC3024AC_DA_Arbitrary_Waveform_control_set( )
	MPC3024AC_DA_Arbitrary_Waveform_control_read( )
2.1	Improve the descriptions about array parameters

Contents

1. Difference between MPC3024 and MPC3024A

Although the MPC3024A is a recommended replacement of MPC3024, there exist some incompatibility between the two. The comparison of the two as follows.

hardware	software	call convention
old MPC3024	wdf3024.sys MPC3024.dll	MPC3024_
old MPC3024	wdf3024A.sys, MPC3024A.dll (32bit)	MPC3024_ (**for compatibility)
new MPC3024A	wdf3024.sys MPC3024.dll	MPC3024_ **
new MPC3024A	wdf3024A.sys, MPC3024A.dll (32bit) wdf3024A_64.sys, MPC3024A_64.dll (64 bit)	MPC3024_ (**for compatibility) **MPC3024A_** **(suggest for new design)**

** The security functions are incompatible in this case.

	Function Name	**Description**
1	MPC3024_initial( )	Initial
2	MPC3024_close( )	Close
3	MPC3024_init_card( )	Initialize parameters and auxiliary function to default value
4	MPC3024_info( )	Get the I/O address and vendor ID of card
5	MPC3024_set_pulse_outmode( )	Configure the pulse output mode
6	MPC3024_readback_pulse_outmode( )	Read back configuration of pulse output mode
7	MPC3024_set_pulse_inmode( )	Configure the multiple rate and the encoder input
8	MPC3024_readback_pulse_inmode( )	Read back configuration of pulse input mode
9	MPC3024_config_SD_PIN( )	Configure slow down input
10	MPC3024_readback_SD_PIN( )	Read back configuration of SD pin
11	MPC3024_config_PCS_PIN( )	Configure PCS(position change start) input
12	MPC3024_readback_PCS_PIN( )	Read back configuration of PCS pin
13	MPC3024_config_INP_PIN( )	Configure INP (in position) input
14	MPC3024_readback_INP_PIN( )	Read back configuration of INP pin
15	MPC3024_config_ERC_PIN( )	Configure ERC (error counter clear) output
16	MPC3024_readback_ERC_PIN( )	Read back configuration of ERC pin
17	MPC3024_config_ALM_PIN( )	Configure ALM (alarm) input
18	MPC3024_readback_ALM_PIN( )	Read back configuration of ALM pin
19	MPC3024_config_LTC_PIN( )	Configure LTC (latch) input
20	MPC3024_readback_LTC_PIN( )	Read back configuration of LTC pin
21	MPC3024_config_CMP_OUT( )	Configure CMP (compare) output
22	MPC3024_readback_CMP_OUT( )	Read back configuration of CMP_OUT
23	MPC3024_config_EL_MODE( )	Configure LS(EL) (over travel) stop mode
24	MPC3024_readback_EL_MODE( )	Read back configuration for LS(EL)
25	MPC3024_config_TTL_IO_MODE()	Configure TTL I/O mode
26	MPC3024_readback_TTL_IO_MODE( )	Read back configuration of TTL_IO
27	MPC3024_set_HOME_pin_logic( )	Configure HOME(ORG) polarity
28	MPC3024_readback_HOME_pin_logic( )	Read back configuration for HOME pin
29	MPC3024_set_EZ_pin_logic( )	Configure EZ (zero phase) polarity

30	MPC3024_readback_EZ_pin_logic( )	Read back configuration of EZ (zero phase) polarity
31	MPC3024_read_point_status( )	Read input status
32	MPC3024_write_output_point( )	Write output
33	MPC3024_save_config2_file( )	Save configuration data to file
34	MPC3024_load_config_from_file( )	Load configuration data from file
35	MPC3024_fix_speed_range( )	Set the maximum allowable speed
36	MPC3024_unfix_speed_range( )	Release the limit of maximum allowable speed
37	MPC3024_T_velocity_move( )	Velocity mode move at trapezoidal profile
38	MPC3024_S_velocity_move( )	Velocity mode move at S curve profile
39	MPC3024_velocity_change( )	To change speed on motion
40	MPC3024_dec_stop( )	Velocity mode, deceleration to stop
41	MPC3024_imd_stop( )	Velocity mode, immediate stop
42	MPC3024_emg_stop( )	Velocity mode, all axes immediate stop
43	MPC3024_read_speed( )	Read the current speed
44	MPC3024_config_home_mode( )	Select the desired homing mode
45	MPC3024_start_homing( )	To execute homing
46	MPC3024_set_current_position( )	Setup the coordinate of current point
47	MPC3024_read_current_position( )	Read the coordinate of current point
48	MPC3024_start_origin_search_homing( )	To command origin search mode homing motion
49	MPC3024_T_curve_position_move( )	Point to point move at trapezoidal acc/dec profile
50	MPC3024_S_curve_position_move( )	Point to point move at S curve profile
51	MPC3024_position_change( )	Change target position while the point to point motion is running
52	MPC3024_backlash_comp( )	Setup backlash compensation
53	MPC3024_readback_backlash_comp( )	Read back configuration of backlash compensation
54	MPC3024_suppress_vibration( )	Setup vibration suppression mode
55	MPC3024_readback_suppress_vibration( )	Read back parameters of vibration suppression mode
56	MPC3024_T_curve_move_LINE2( )	Two axes linear interpolation at trapezoidal profile
57	MPC3024_S_curve_move_LINE2( )	Two axes linear interpolation at S curve profile
58	MPC3024_T_curve_move_LINE3( )	3 axes linear interpolation at trapezoidal profile
59	MPC3024_S_curve_move_LINE3( )	3axes linear interpolation at S curve profile
60	MPC3024_T_curve_move_LINE4( )	4 axes linear interpolation at trapezoidal profile
61	MPC3024_S_curve_move_LINE4( )	4 axes linear interpolation at S curve profile
62	MPC3024_ARC2_center_move( )	Circular interpolation with the circle center and end position as parameters
63	MPC3024_ARC2_3P_move( )	Circular interpolation with current point and the other 2 points as parameters
64	MPC3024_CIR2_3P_move( )	Circular interpolation with current point and the other 2 points as parameters
65	MPC3024_ARC2_Radius_move( )	Circular interpolation with end point and radius as parameters
66	MPC3024_CIR2_Radius_move( )	Circular interpolation with radius and end position as parameters for circular trajectory
67	MPC3024_set_continuous_flag( )	Enable / disable the continuous mode
68	MPC3024_check_continuous_buffer()	To check the continuous buffer
69	MPC3024_read_motion_status( )	Read the motion status

70	MPC3024_set_event_factor( )	To enable the event for corresponding event source
71	MPC3024_read_event_flag( )	To read the event source
72	MPC3024_read_error_flag( )	To read back the status of error source
73	MPC3024_OnLine_T_curve_change( )	To change the motion parameters on the fly for single axis.
74	MPC3024_OnLine_T_curve_change_ LINE2( )	To change the motion parameters on the fly for any 2 axes linear interpolation.
75	MPC3024_OnLine_T_curve_change_ LINE3( )	To change the motion parameters on the fly for any 3 axes linear interpolation.
76	MPC3024_OnLine_T_curve_change_ LINE4( )	To change the motion parameters on the fly for 4 axes linear interpolation.
77	MPC3024_enable_IRQ( )	To enable the interrupt function.
78	MPC3024_disable_IRQ( )	To disable the interrupt function, and release the resource and close thread.
79	MPC3024_link_IRQ_process( )	Link irq service routine to driver
80	MPC3024_set_INT_source( )	To setup the error/event source that will generate interrupt at error/event occurs.
81	MPC3024_read_INT_status( )	To read back the status of interrupt event source
82	MPC3024_set_INT_mask( )	To set the interrupt mask of designated axis.
83	MPC3024_config_softlimit( )	Configure soft limit
84	MPC3024_readback_config_softlimit( )	Read back the software limit parameter
85	MPC3024_set_softlimit_data( )	Setup the coordinate data of soft limit
86	MPC3024_readback_softlimit_data( )	Read back the coordinate of software limit
87	MPC3024_enable_softlimit( )	Enable / disable software limit function
88	MPC3024_readback_enable_softlimit( )	Read back the status of enable / disable software limit
89	MPC3024_read_softlimit_flag( )	Read the software limit flag for verifying
90	MPC3024_config_pulser_mode( )	Configure the operating mode of the pulse handler
91	MPC3024_readback_pulser_mode( )	Read back the pulse handler operation mode
92	MPC3024_run_pulser_Vmove( )	Operate pulse handler as manual speed control
93	MPC3024_run_pulser_Pmove( )	Operate pulse handler as manual position control
94	MPC3024_set_pulser_counter( )	Set pulse counter
95	MPC3024_read_pulser_counter( )	Read pulse counter
96	MPC3024_set_pulser_Map( )	Map the source (pulse handler) to the target motion axis
97	MPC3024_enable_pulser_motion( )	Enable pulse handler function and the multiple rate
98	MPC3024_read_FB_counter( )	Read feedback counter
99	MPC3024_set_FB_counter( )	Set feedback counter
100	MPC3024_read_FBcounter_latch_value( )	Read feedback counter latched value
101	MPC3024_config_comparator_out( )	Configure the compare output mode
102	MPC3024_readback_comparator_out( )	Read back the configuration of the compare mode
103	MPC3024_set_comparator_data( )	Preset the value to the comparator
104	MPC3024_readback_comparator_data( )	Read back the preset comparator value
105	MPC3024_read_compare_flag( )	Read compare out flag

2. How to install the software of MPC3024A

2.1 Install the PCI driver

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

In WinXP/Win7 system you should: (take Win XP 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. Do not response to the wizard, just Install the file (..\MPC3024A_AC\Software\WinXP_7\ or if you download from website please execute the file MPC3024A_Install.exe to get the file)
6. After installation, power off
7. Power on, it's ready to use

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.

* MPC3024AC has the basic functions of MPC3024A and old hardware MPC3024, it is with the closed loop control functions as superset of them.

3. Where to find the file you need

WinXP/Win7

The directory will be located at

.. \ JS Automation \MPC3024A\API\   (header files and lib files for VB,VC,BCB,C#)

.. \ JS Automation \MPC3024A\Driver\   (backup copy of MPC3024A drivers)

.. \ JS Automation \MPC3024A\exe\   (demo program and source code)

The system driver is located at **..\system32\Drivers** and the DLL is located at **..\system**.

For your easy startup, the demo program with source code demonstrates the card functions and help file.

4. About the MPC3024A software

MPC3024A software includes a set of dynamic link library (DLL) and system driver that you can utilize to control the I/O card's function easily.

Your MPC3024A 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 MPC3024A functions within Windows' operation system environment.

4.1 What you need to get started

To set up and use your MPC3024A software, you need the following:

- MPC3024A software
- MPC3024A hardware

  Main board

  Wiring board (Option)

4.2 Software programming choices

You have several options to choose from when you are programming MPC3024A 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 MPC3024A software.

5. MPC3024A Language support

The MPC3024A software library is a DLL used with WinXP/Win7. You can use these DLL with any Windows integrating development environment that can call Windows DLLs.

5.1 Building applications with the MPC3024A software library

The MPC3024A function reference topic contains general information about building MPC3024A applications, describes the nature of the MPC3024A files used in building MPC3024A applications, and explains the basics of making applications using the following tools:

Applications tools

- Microsoft Visual C/C++
- Borland C/C++
- Microsoft Visual C#
- Microsoft Visual Basic
- Microsoft VB.net

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

5.2 MPC3024A Windows libraries

The MPC3024A for Windows function library is a DLL called **MPC3024A.dll**. Since a DLL is used, MPC3024A functions are not linked into the executable files of applications. Only the information about the MPC3024A functions in the MPC3024A 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 MPC3024A functions in MPC3024A.dll.

Header Files and Import Libraries for Different Development Environments		
Language	**Header File**	**Import Library**
Microsoft Visual C/C++	MPC3024A.h	MPC3024AVC.lib
Borland C/C++	MPC3024A.h	MPC3024ABC.lib
Microsoft Visual C#	MPC3024A.cs	
Microsoft Visual Basic	MPC3024A.bas	
Microsoft VB.net	MPC3024A.vb	

Table 1

6. Basic concepts of motion control

6.1 Classification of motion control by interface

The common used motors in motion control are step motor or servo motor. Traditionally, we control step motors by using pulse train (6.1.1) but on the other hand, servo motors can be controlled by analog voltage (6.1.2) or pulse. The un-usual type of control can be through the communication method (6.1.3).

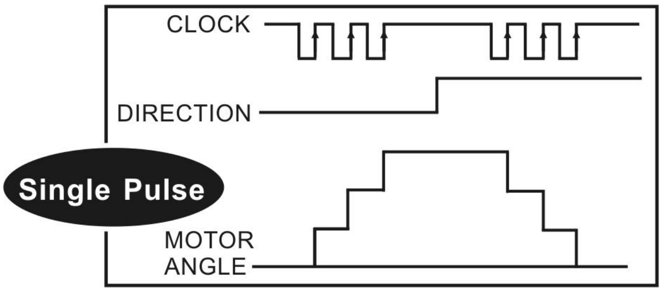
6.1.1 Pulse type motion control

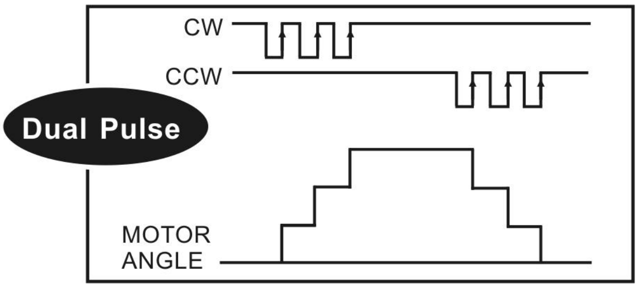
The pulse type motion control was used long ago in step motor control system. In the recent year, a new trend of digital control has moved the servo control from traditional analog control to pulse type motion control.

First, how the pulse train controls the speed and position of a motion control system? **The total pulse number is the units of distance to move and the pulse rate is the speed of motion.** In pulse type motion control, you must use a servo driver that can accept pulse train to control. The driver will close loop the feedback of the encoder of the servo motor by itself, the motion controller is just a commander.

Users can use a pulse type motion controller to control step motors or servo motors without any modification of software.

There are two control methods of pulse train, single pulse type and dual pulse type.





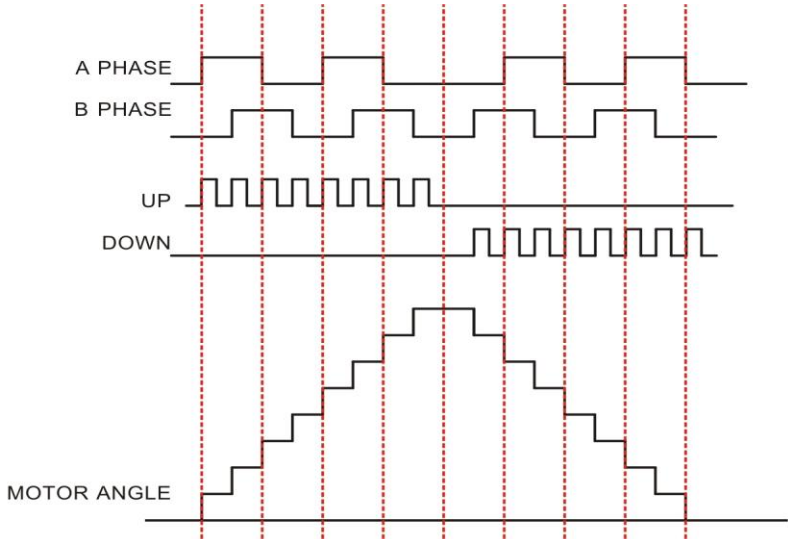
Single pulse type control use only one clock source to control speed and position and the other input is direction control. Dual pulse type control use clockwise clock to control speed and position in one direction and counter-clockwise clock for the other direction.

Let’s take a deep investigation, in single pulse control mode, if clock signal is defective (caused wire broken or short), the motor will not move at all. It seems good to protect from mal-function. But on the other hand, if the direction signal is defective, the motor will run at only one direction, this may cause hazard to equipment.

In dual phase mode, if CW is defective, there will be no counter clockwise moving, and counter-clockwise will not effect, this condition is vice versa in CCW signal defectiveness.

MPC is the pulse type motion control card and provides software selectable function to choose the control method. We suggest you to choose dual phase method for better future maintenance.

Some drivers also provides quadrature pulse input, users can use a quadrature encoder signals to control servo motor.



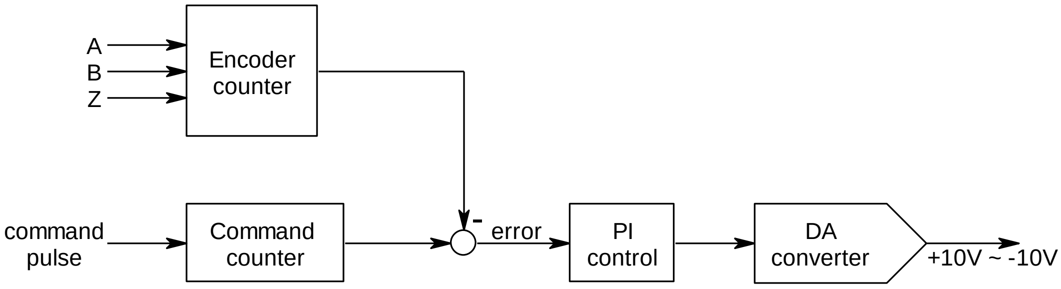
The quadrature A,B phase input also have the direction information encoded, see the above figure, the up and down clock is internally identified by the driver and the motor steps the angle as command input.

6.1.2 Voltage type motion control

The basic difference of the voltage type motion control is the driver only close loop for speed. There will be a controller which can accept the position feedback to close the position control loop.

Normally the voltage type driver accepts +10V as the clockwise rated speed input and –10V as the counter-clockwise rated speed input.

MPC3042AC* provides dual mode of motion control: pulse mode and voltage controlled close loop mode. Each servo drive will be controlled by a 17bit digital to analog converter of analog voltage from +10V to –10V dc voltage, which is driven by the error of command pulse input and encoder feedback. A PI compensator put between the error counter and D/A can be tuned for various kind of application. The following diagram shows the function blocks.



This type of control is also called as pulse reference close loop type. You can adjust the PI parameters to achieve good response and minimize position error.

*MPC3024A do not have the closed loop function.

6.1.3 Communication type motion control

A non-traditional method is communication type motion control; by RS232, RS485 or Ethernet or any kind of communication protocol. The command between motor driver and motion controller is not analog or pulses signal any more. It is a command packet which contains motion information to pass back and forth between the driver and controller. If the controller wants to directly control the speed and position of servo motor, the communication speed must high enough to up to 1000 communication per second. A single driver maybe no problem but if more servo drivers to control, this means the bandwidth should be as high as the number of servo drivers increased.

6.2 Classification of motion control by system implementation

For motion control system, the motion profile generation and control algorithm may be implemented by software or by hardware. But sometimes we can not clearly distinguish. The designers always use their best design topology to implement the system.

6.2.1 Software based motion control

For software motion control type, the motion profile generation and control algorithm heavily depends on software. The software must fast enough to calculate the profile generation and feedback control algorithm. Generally the sample rate must up to 200Hz or higher (per axis).

Some designer use a DSP as a slave processor to implement the motion control related real time task, basically it is a software type motion control system.

6.2.2 Hardware based motion control

Using dedicated hardware to implement motion control is another way, it spends very few software resource. In recent days, ASIC is so popular, an ASIC-based design of motion control system is a low cost solution.

It has no real-time problem because all motion functions are done via ASIC. Users just need to set some parameters, which ASIC requires and the motion control will be done easily. MPC card is an ASIC-based motion control card, it can be run even on early day’s PC.

6.3 Classification of motion control by application

There are 4 major types of application:

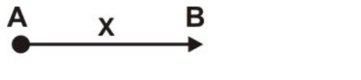
- speed control: controller controls the speed of the servo motor.
- torque control: the controller controls the torque output of the servo motor.
- tracking control: the controller controls the servo motor to follow the motion of another servo motor.
- positioning control: the controller controls the servo motor of contour motion.

Of course a mixed mode is possible.

MPC is hard ware designed for speed control and position control (point to point and linear, circular interpolation). Tracking control can also be implemented on MPC3024AC hardware.

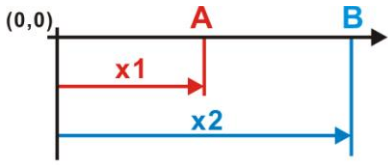
6.4 Coordinate system

The Cartesian coordinates of motion control generally divided by relative and absolute coordinate system.



B is x distance away from A

The relative coordinate system, any point’s coordinate is measured by its reference point.

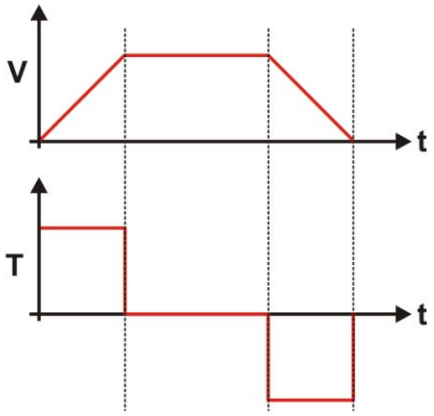


A is at x1 position
B is at x2 position

The absolute system must have a point as a origin. All the other points are measured from the origin.

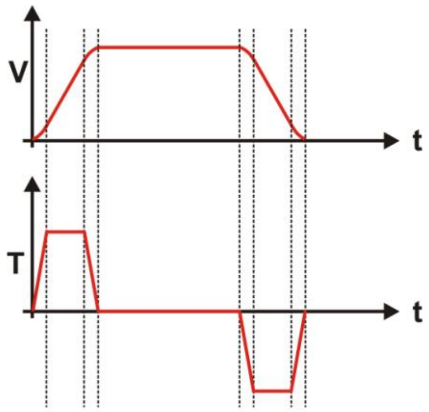
6.5 Motion profile

Motion profile is the speed to time curve of motion. Generally there are trapezoidal motion profile and S curve motion profile.



Trapezoidal profile

Trapezoidal motion profile (T curve) has a step torque curve. The machine will work under a jerk that increase the weak of mechanism.



S curve profile

The advantage of S curve profile:
- Reduces wear on mechanical components improving machine life
- Reduces system resonance and overshoot

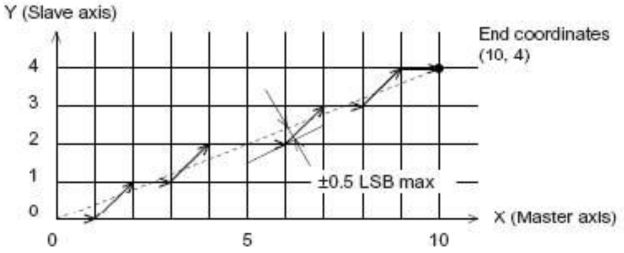
The disadvantage is:
- Requires either twice the acceleration torque or acceleration time for a S profile compared to trapezoidal motion profile

MPC card provides both motion profile function for the user application, you can estimate the system requirement to make the decision.

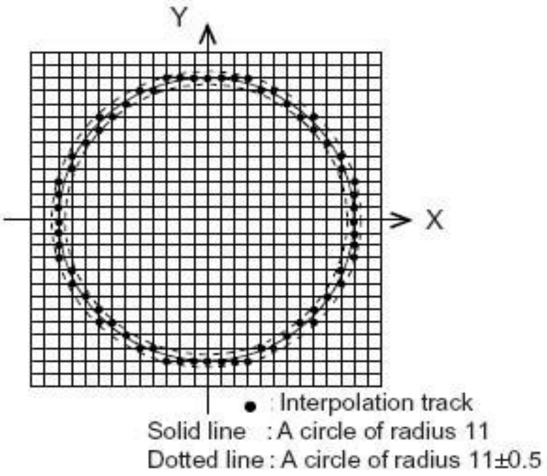
6.6 Interpolation

If you define the start and end position of line segment, the controller will go as you need at required speed and keep the position accuracy at every points it passed. This type of function is called linear interpolation function. If the trajectory is circular, we call it circular interpolation.

Linear and circular interpolations are the two most important interpolation functions. MPC card provides the hardware interpolation of both. If you want to do special curve interpolation, you can divide the curve to small line segments and using continuous function to line up the curve.



A close look of linear interpolation, say X axis is the master axis, the Y axis is slave and the composite curve try to keep the trajectory as close to the ideal curve as possible


Y
X
● : Interpolation track
Solid line : A circle of radius 11
Dotted line : A circle of radius 11±0.5

A close look of circular interpolation, the MPC hardware try to keep the circular interpolation curve close to the ideal curve and also the speed of tangential speed of the curve as user programmed.

6.7 Homing and over-travel limit

While system is power up and if the encoder is not absolute type, the system do not know where it is now. Homing function will return the mechanism to a known point and set the coordinate. There are so many homing modes available for users. MPC provides 13 homing modes to fit different requirement of applications.

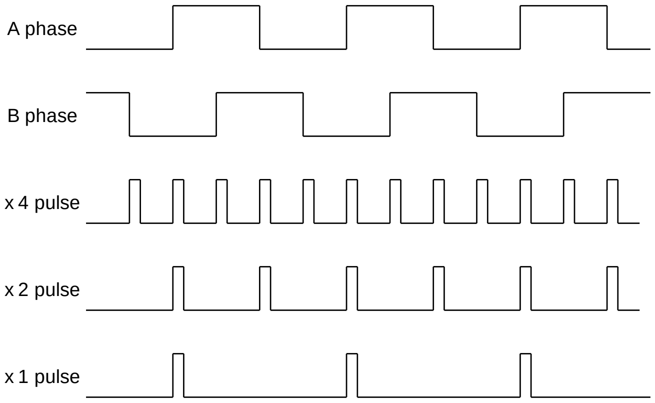
Over-travel limit switch is used under the consideration of ab-normal. If the feedback or other failure that will make the motor run out of control, the over-travel limit switches are put at the extreme position of impossible movement, once it is active, the controller must stop the motion to prevent hazard.

Over-travel limit can also implement by software, but first of all, the coordinate system must setup correctly. MPC provides both the hardware over-travel limit and software over-travel limit functions.

6.8 Feedback element of servo system

There are several types of servo motor feedback elements such as: encoder (absolute or incremental), resolver, potential meter… MPC card can only deal with incremental encoders.

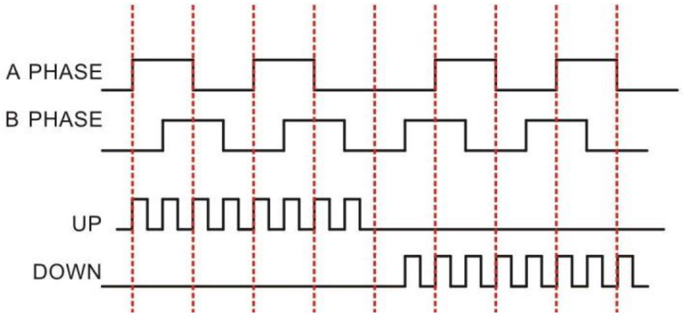
It is a device with 2 phase signals separated at 90 degree. From the input pulse we can multiply the pulse by detecting its edge by x4, x2 or x1. The figure show as follows:



The left diagram shown that A phase leads B, if we take A leads B as up count and the counting pulse of up count will depends on the multiple rate.

On the other hand, if B phase leads A phase, the counter will be down count.

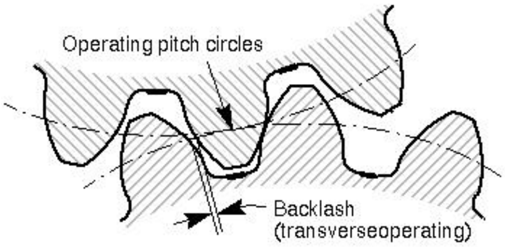
We can also discriminate the rotation direction from the phase lead or phase lag. From the following figure, if A lead B, we can decode the up pulses and if B lead A, we also can decode the down pulses. The up count or down count pulse shown at the following figure being configured as x4.



6.9 Nature of mechanism system

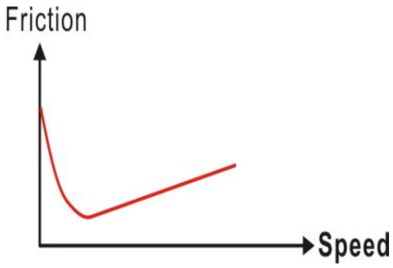
The motion control system is actually a mechatronic system (mechanical + electronics). If you want the system work perfect, you can not overlook the importance of mechanism.

6.9.1 Backlash



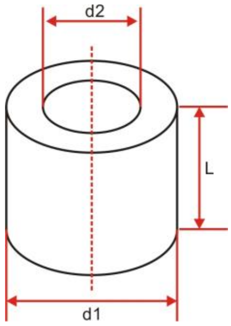
Backlash is the free motion of mechanism when the direction reversed. It is one of the important nature of mechanism. It exist in gear, screw mechanism.

6.9.2 Friction



At low speed, the static friction will dominate but at high speed, the dynamic friction will be important. The mechanism for motion control should try to keep the friction as low and smooth as possible to avoid the servo system fall into a limit cycle oscillation.

6.9.3 Inertia



Inertia is the tendency of a body to resist acceleration. It is normally proportional to mass and squared proportional to diameter.

The left cylinder inertia J will be:
J = Mass * (d1^2 + d2^2)/8
(Kg-M^2) = (Kg) * (M^2)

7. Function format and language difference

7.1 Function format

Every MPC3024A 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 MPC3024A function is the parameter **CardID** which is located the driver of MPC3024A board you want to use those given operation. The **CardID** is assigned by DIP/ROTARY SW. You can utilize multiple devices with different card CardID within one application; to do so, simply pass the appropriate **CardID** to each function.

Note: **CardID** is set by DIP/ROTARY SW (**0x0-0xF**)

7.2 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

7.3 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 MPC3024A API. Read the following sections that apply to your programming language.

Note : Be sure to include the declaration functions of MPC3024A prototypes by including the appropriate MPC3024A header file in your source code. Refer to Building Applications with the MPC3024A Software Library for the header file appropriate to your compiler.

7.3.1 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 Port function has the following format:

Status = MPC3024A_output_point_read(CardID, axis, check_factor, *state);

where **CardID** , **axis and check_factor** are input parameters, and **state** is an output parameter. Consider the following example:

u8 CardID, axis, port;
u8 state,
u32 Status;
Status = MPC3024A_set_port (CardID, axis, check_factor, state);

7.3.2 Visual basic

The file MPC3024A.bas contains definitions for constants required for obtaining card information and declared functions and variable as global variables. You should use these constants symbols in the MPC3024A.bas, do not use the numerical values.

In Visual Basic, you can add the entire MPC3024A.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 MPC3024A.bas file for your project in Visual Basic 4.0, go to the **File** menu and select the **Add File... option**. Select MPC3024A.bas, which is browsed in the MPC3024A \ API directory. Then, select **Open** to add the file to the project.

To add the MPC3024A.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** MPC3024A.bas, which is in the MPC3024A \ API directory. Then, select **Open** to add the file to the project.

7.3.3 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 MPC3024ABC.lib MPC3024A.dll

Then add the **MPC3024ABC.lib** to your project and add

#include "MPC3024A.h" to main program.

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

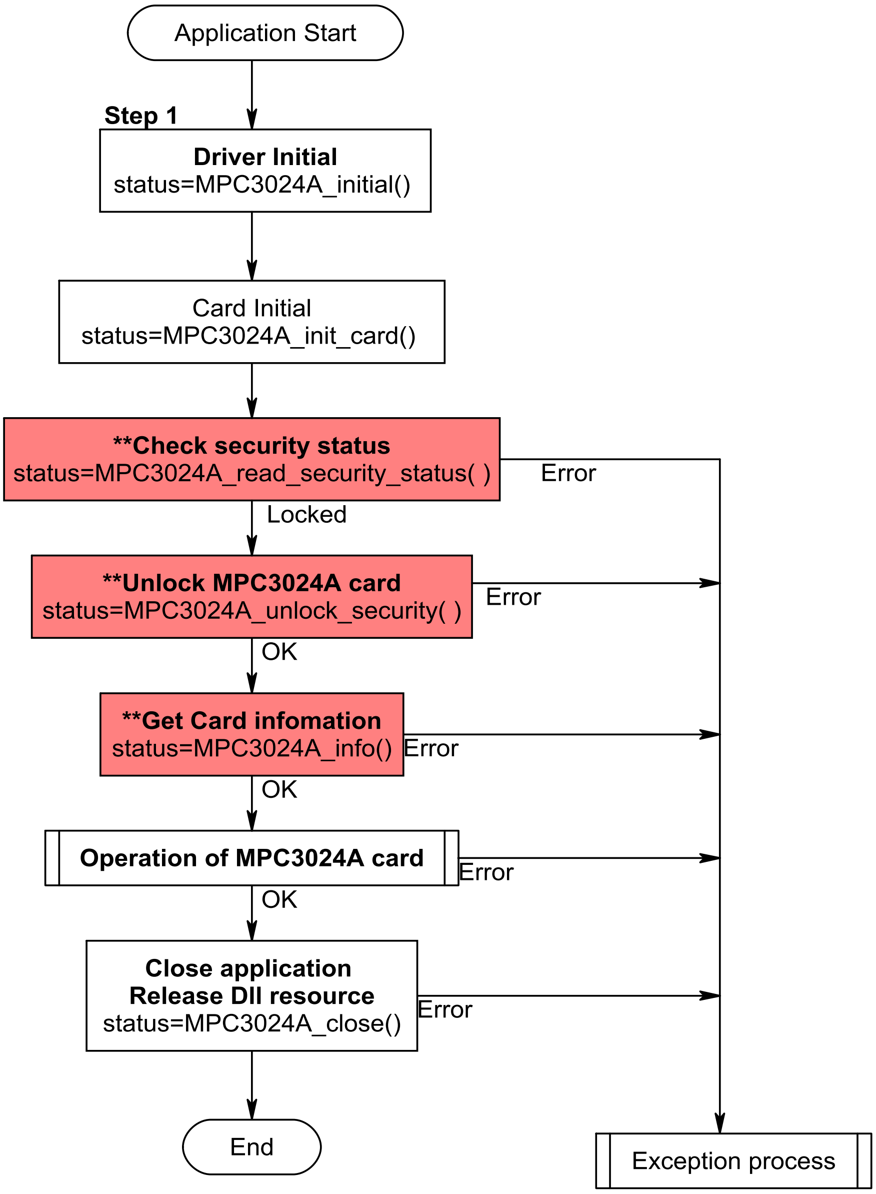
Status = MPC3024A_output_point_read(CardID, axis, check_factor, *state);

where **CardID** , **axis and check_factor** are input parameters, and **state** is an output parameter. Consider the following example:

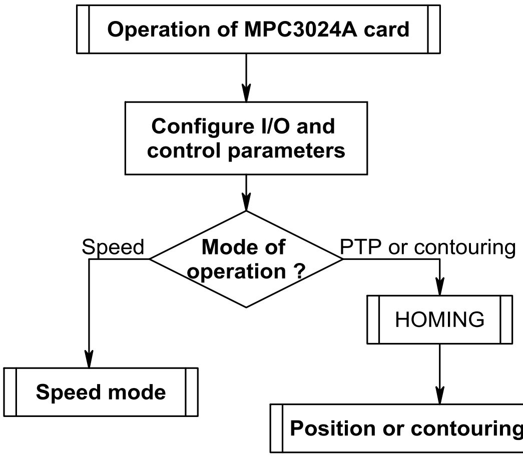
```
u8 CardID, axis, port;
u8 state;
u32 Status;
Status = MPC3024A_set_port (CardID, axis, check_factor, state);
```

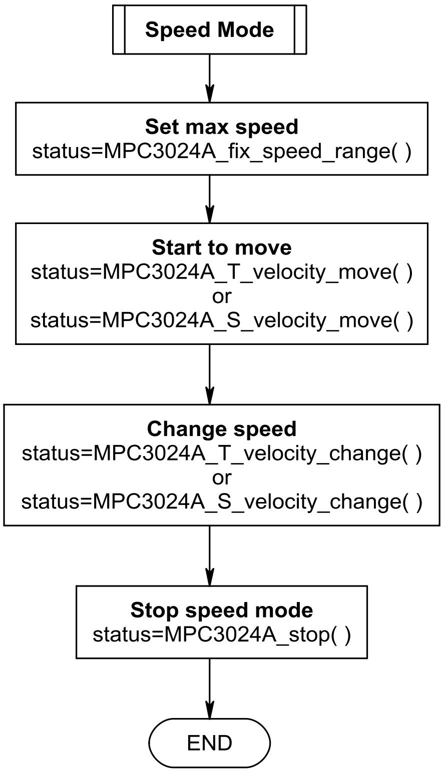
8. Flow chart of implement an application

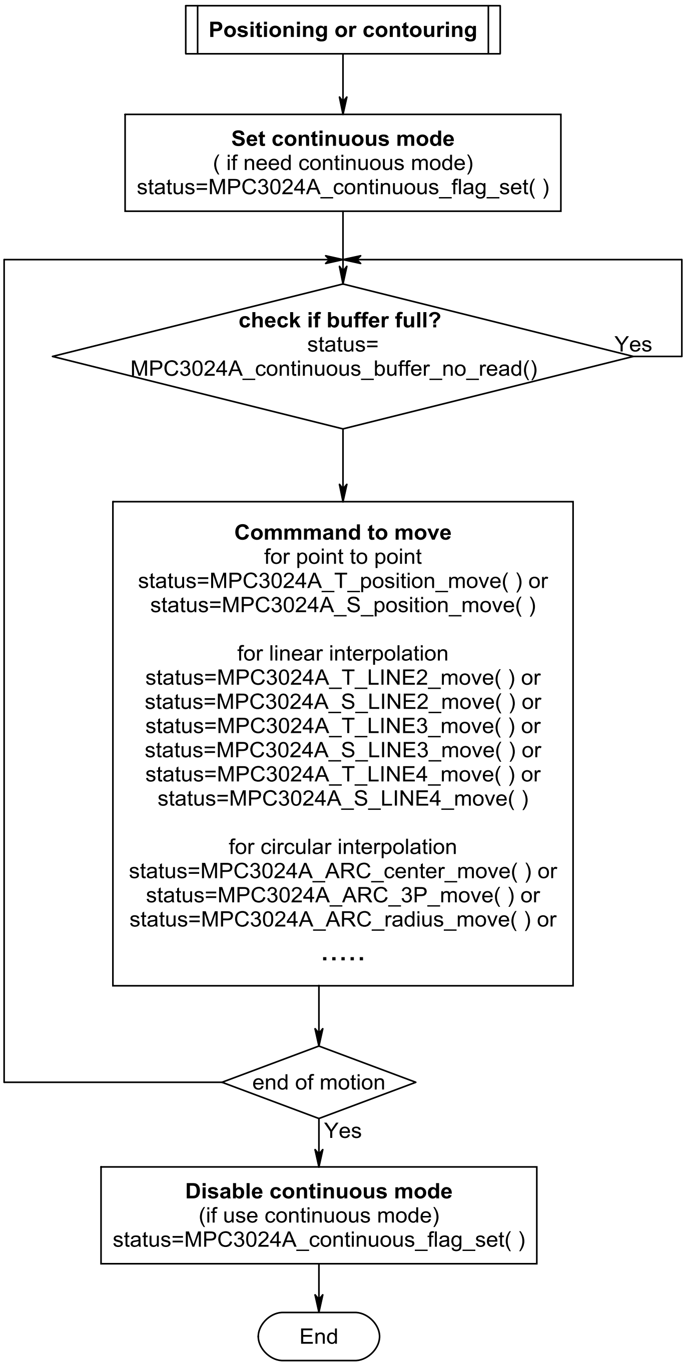
8.1 MPC3024A Flow chart of implementation



** Security function can be skipped, if you do not use such function.
** If you will use security function, you should setup security first on Demo program or other utility.


Operation of MPC3024A card
Configure I/O and control parameters
Mode of operation ?
Speed
PTP or contouring
HOMING
Speed mode
Position or contouring mode


Speed Mode
Set max speed
status=MPC3024A_fix_speed_range( )
Start to move
status=MPC3024A_T_velocity_move( )
or
status=MPC3024A_S_velocity_move( )
Change speed
status=MPC3024A_T_velocity_change( )
or
status=MPC3024A_S_velocity_change( )
Stop speed mode
status=MPC3024A_stop( )
END


Positioning or contouring
Set continuous mode
(if need continuous mode)
status=MPC3024A_continuous_flag_set( )
check if buffer full?
status=
MPC3024A_continuous_buffer_no_read()
Yes
Commmand to move
for point to point
status=MPC3024A_T_position_move( ) or
status=MPC3024A_S_position_move( )
for linear interpolation
status=MPC3024A_T_LINE2_move( ) or
status=MPC3024A_S_LINE2_move( ) or
status=MPC3024A_T_LINE3_move( ) or
status=MPC3024A_S_LINE3_move( ) or
status=MPC3024A_T_LINE4_move( ) or
status=MPC3024A_S_LINE4_move( )
for circular interpolation
status=MPC3024A_ARC_center_move( ) or
status=MPC3024A_ARC_3P_move( ) or
status=MPC3024A_ARC_radius_move( ) or
.....
end of motion
Yes
Disable continuous mode
(if use continuous mode)
status=MPC3024A_continuous_flag_set( )
End

9. Software overview and dll function

These topics describe the features and functionality of the MPC3024A boards and briefly describes the MPC3024A functions.

9.1 Initialization and close

You need to initialize system resource each time you run your application.

MPC3024A_initial( ) will do.

Once you want to close your application, call

MPC3024A_close( ) to release all the resource.

To initialize the motion and auxiliary functions at the beginning of power on,

MPC3024A_init_card( ) is a must.

If you want to know the physical address assigned by OS. use

MPC3024A_info( ) to get the address and vendor ID (for MPC3024A the vendor ID is 3024).

- **MPC3024A_initial**

Format : **u32 status = MPC3024A_initial (void)**

Purpose: Initial the MPC3024A resource when start the Windows applications.

- **MPC3024A_close**

Format : **u32 status = MPC3024A_close(void)**

Purpose: Release the MPC3024A resource when close the Windows applications.

- **MPC3024A_init_card**

Format : **u32 status = MPC3024A_init_card(u8 CardID)**

Purpose: To initialize motion function parameters and auxiliary function to default value.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

- **MPC3024A_info**

Format : **u32 status = MPC3024A_info(u8 CardID, u8 *CardType, u16 *address)**

Purpose: Read the physical I/O address assigned by O.S. and vendor ID.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Output:

Name	Type	Description
CardType	u8	0: MPC3024A 1: MPC3024AC
address	u16	physical I/O address assigned by OS

9.2 Save and reload configuration file

Motion related system parameters configured by the Motion related I/O configure and control command includes:

- pulse output mode
- encoder input mode and multiple rate
- SD pin logic and mode
- PCS pin logic and mode
- INP pin logic and mode
- ERC pin logic and mode
- ALM pin logic and mode
- LTC pin logic and mode
- CMP pin logic and mode
- EL pin logic and mode
- HOME pin logic and homing mode
- EZ pin logic
- backlash pulse number, speed and direction

All the above mentioned could be saved to file by

MPC3024A_config_file_set( )

and retrieve to the card by

MPC3024A_config_file_read( )

- **MPC3024A_config_file_set**

Format : **u32 status = MPC3024A_config_file_set(u8 CardID, char* file_name)**

Purpose: Save configuration data to file.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
file_name	char	file name of the configuration data to be saved

- **MPC3024A_config_file_read**

Format : **u32 status = MPC3024A_config_file_read(u8 CardID, char* file_name)**

Purpose: Load configuration data from file.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
file_name	char	file name of the configuration data to be saved

9.3 I/O configuration and control

Motion related input debounce time

The motion related input HOME, LS+(EL+, positive direction bover travel limit), LS-(EL-, negative direction over travel limit) are essential digital inputs to motion control. The MPC card provide software programmable digital debounce to filter out the unwanted glitch from the sensor and environment. Use

MPC3024A_debounce_set( ) to setup the debounce time and read back by ***MPC3024A_debounce_read( )***

- **MPC3024A_debounce_set**

Format : **u32 status = MPC3024A_debounce_set(u8 CardID, u8 axis, u8 debounce)**

Purpose: To configure the EL (LS+,LS-) and ORG (HOME) input debounce time.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
debounce	u8	0: no debounce 1: filter out input less than 10ms, debounce frequency at 100Hz. 2: s filter out input less than 5ms, debounce frequency at 200Hz. 3: filter out input less than 1ms, debounce frequency at 1KHz. The debounce time is applied to EL (LS+,LS-) and ORG(HOME).

- **MPC3024A_debounce_read**

Format : **u32 status = MPC3024A_debounce_read(u8 CardID, u8 axis, u8 *debounce)**

Purpose: To read back input debounce time of the EL (LS+,LS-) and ORG (HOME).

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

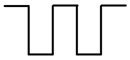
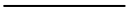
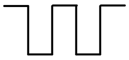
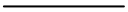
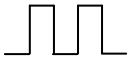
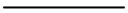
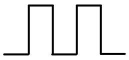
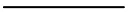
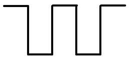
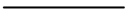
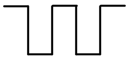
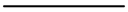
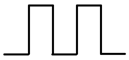
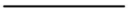
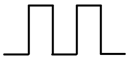
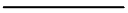
Name	Type	Description
debounce	u8	0: no debounce 1: filter out input less than 10ms, debounce frequency at 100Hz. 2: s filter out input less than 5ms, debounce frequency at 200Hz. 3: filter out input less than 1ms, debounce frequency at 1KHz. The debounce time is applied to EL (LS+,LS-) and ORG(HOME).

Motion related I/O

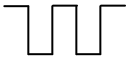
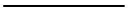
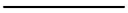
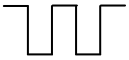
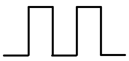
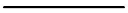
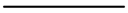
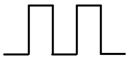
The pulse output, HOME limit switch, slow-down limit switch and over-travel limit switch are the function of motion related I/O.

To meet your driver, you should first configure the pulse output mode. There are 8 types of pulse mode for your selection:

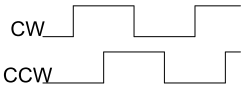
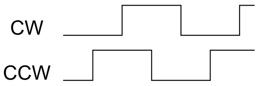
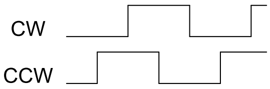
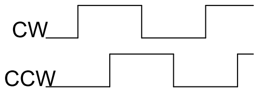
Single pulse mode: (CW pin act as pulse out, CCW pin act as direction out)

Pulse_outmode	Operation in plus direction		Operation in minus direction	
	pulse	direction	pulse	direction
0				
1				
2				
3				

Dual pulse mode: (CW for clockwise and CCW for counter-clockwise)

Pulse_outmode	Operation in plus direction		Operation in minus direction	
	CW	CCW	CW	CCW
4				
5				

Quadrature mode: (CW for A clock and CCW for B clock)

Pulse_outmode	Operation in plus direction		Operation in minus direction	
	CW	CCW	CW	CCW
6				
7				

From the above diagram both signals can also be active high or active low to drive the servo driver.

MPC3024A_pulse_outmode_set( ) to configure the pulse output type and

MPC3024A_pulse_outmode_read( ) for configuration read back.

Some time you need a slow-down limit switch at the point near HOME(ORG) or LS+(EL+),LS-(EL-) to prevent jog while LS+(EL+),LS-(EL-) or Home(ORG) activated.

MPC3024A_SD_PIN_set( ) will do.

MPC3024A_SD_PIN_read( ) for configuration read back.

To protect your system from over-travel, limit switch is common to use, configure the stop mode while it is activated by

MPC3024A_EL_mode_set( ).

MPC3024A_EL_mode_read( ) for configuration read back.

- **MPC3024A_pulse_outmode_set**

Format : **u32 status = MPC3024A_pulse_outmode_set(u8 CardID, u8 axis,**
u8 pulse_outmode)

Purpose: Set the pulse output mode for the designated axis.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
pulse_outmode	u8	0~7 (See **Note on pulse out mode**)

- **MPC3024A_pulse_outmode_read**

Format : **u32 status = MPC3024A_pulse_outmode_read(u8 CardID, u8 axis, u8* pulse_outmode)**

Purpose: Readback the pulse output mode for the designated axis.

Parameters:

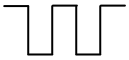
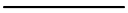
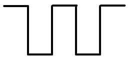
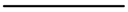
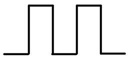
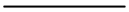
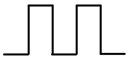
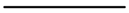
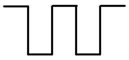
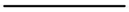
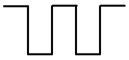
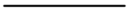
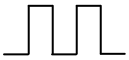
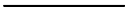
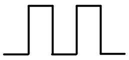
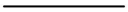
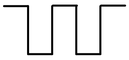
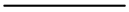
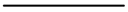
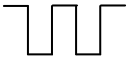
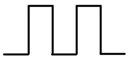
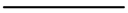
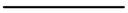
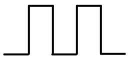
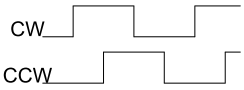
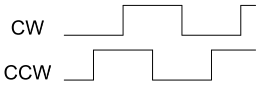
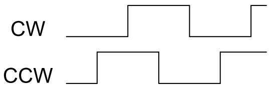
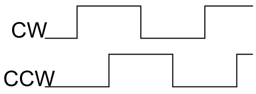
Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X axis 2: Z axis	1: Y axis 3: A axis

Output:

Name	Type	Description
pulse_outmode	u8	0~7 (See **Note on pulse out mode**)

Note on pulse out mode:

Pulse _out mode	Operation in plus direction		Operation in minus direction		Comments
	OUT pin (CW)	DIR pin (CCW)	OUT pin (CW)	DIR pin (CCW)	
0					Single pulse, Active low
1					Single pulse, Active high
2					Single pulse, Active low Inverse direction
3					Single pulse, Active high Inverse direction
4					Dual pulse Active low
5					Dual pulse Active high
6					quadrature
7					quadrature

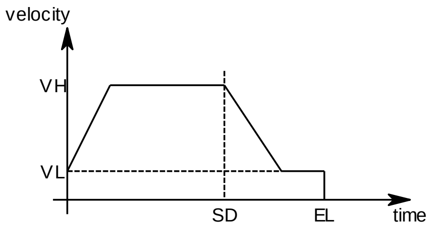
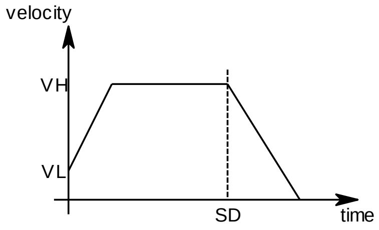
- **MPC3024A_SD_PIN_set**

Format : **u32 status = MPC3024A_SD_PIN_set(u8 CardID, u8 axis, u8 enable, u8 sd_logic, u8 sd_latch, u8 sd_mode)**

Purpose: Configure the slow down input and its mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
enable	u8	0: treat SD PIN as a general input. 1: treat SD PIN as a dedicated slow down signal input.
sd_logic	u8	0: setting the pin connects or equals to GND level make this pin active logic. 1: setting the pin floats or equals to +24V makes this signal active logic.
sd_latch	u8	0: disable SD latch function. 1: enable SD latch function. (See **Note on SD latch function**)
sd_mode	u8	0: when SD signal active motion decelerate to low speed.  1: when SD signal active motion decelerate to stop. 

- **MPC3024A_SD_PIN_read**

Format : **u32 status = MPC3024A_SD_PIN_read(u8 CardID, u8 axis, u8* enable, u8* sd_logic, u8* sd_latch, u8* sd_mode, u8 *state)**

Purpose: Readback the configuratyion of the slow down input and its mode.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X axis 2: Z axis	1: Y axis 3: A axis

Output:

Name	Type	Description
enable	u8	0: treat SD PIN as a general input. 1: treat SD PIN as a dedicated slow down signal input.
sd_logic	u8	0: setting the pin connects or equals to GND level make this pin active logic. 1: setting the pin floats or equals to +24V makes this signal active logic.
sd_latch	u8	0: disable SD latch function. 1: enable SD latch function. (See Note on SD latch function)
sd_mode	u8	0: when SD signal active motion decelerate to low speed. 1: when SD signal active motion decelerate to stop.
state	u8	state of SD pin

Note on SD latch function:

sd_latch	Description
0	disable latch, the Slow Down behavior only in SD signal input active period.
1	enable latch, once the SD signal trigger occurs the Slow Down function will be active and latched until this function disabled. Suggest to use this mode while SD signal is short.

MPC3024A_EL_mode_set

Format : **u32 status =MPC3024A_EL_mode_set (u8 CardID, u8 axis, u8 el_mode)**

Purpose: To configure the LS(EL)(end limit,over travel limit switch) mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
el_mode	u8	0: immediate stop. 1: decelerate to stop

Note:

1. On wiring board terminal marked as LS+(EL+) for positive side over travel limit.
2. On wiring board terminal marked as LS-(EL-) for negative side over travel limit.
3. Although each axis has 2 end limit (LS+(EL+),LS-(EL-)), the LS(EL) polarity can be set by one bit of dip switch on card. (i.e. the 2 LS(EL) must have the same polarity)

MPC3024A_EL_mode_read

Format : **u32 status = MPC3024A_EL_mode_read(u8 CardID, u8 axis, u8*el_mode)**

Purpose: To configure the SL(EL)(end limit,over travel limit switch) mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
el_mode	u8	0: immediate stop. 1: decelerate to stop

Servo drive related IO

The servo-on output, in-position input, error counter output and alarm input are the interface to servo driver, they are servo drive related I/O.

SVON: servo-on output from motion card to control the servo drive to activate the motor.

INP: in-position input of motion card, the signal will be active while the motor stop within the accuracy range.

ERC: error counter output from the motion card, the signal will be active according to different operation mode of HOMing, it is used to clear the remained pulses wihle homing point meets.

ALM: alarm input of motion card, this is an exception signal, it will make the motion stop.

On a pulse type servo driver, the closed loop control is implemented on the drive itself and it provides the signal to interface with the controller to get high accuracy. Normally, in-position output for controller to check the accuracy is achieved at a preset region; error counter clear input to provide the controller to clear the remained pulses after homing or any condition you need to minimize the effect of previous motion remained pulses; alarm output to claim an error occures for controller to take special actions and servo on input for controller to actuate the servo function after controller reset and servo driver ready to prevent abnormal motion during the reset period.

The MPC card servo on output is just a output point specially defined, it can be set or reset by

MPC3024A_point_set( ) (refer 9.3 I/O configuration and control TTL I/O and output controlTTL I/O and output controlTTL I/O and output controlTTL I/O and output controlTTL I/O and output control)

If your application needs in_position signal to verify the motion is completed by the driver, be sure to connect the in_position output from the servo driver to the INP input and use

MPC3024A_INP_PIN_set( ) to configure and

MPC3024A_INP_PIN_read( ) for configuration read back.

In the pulse type control system, servo driver play an important role, but during homing the motion processor detect the home (ORG) signal, the driver can not get any information but no pulse train. There maybe some remain pulses to move (in the driver input buffer). To ensure the accuracy, most servo drivers provide error counter (deviation counter) clear input for external device to clear the remained pulses. For automatic error counter clear at homing, use

MPC3024A_ERC_PIN_set( ) to configure your requirement.

MPC3024A_ERC_PIN_read( ) for configuration read back.

If your driver has alarm output and you wish to use it as ALM input to the processor,

MPC3024A_ALM_PIN_set( ) will do.

MPC3024A_ALM_PIN_read( ) for configuration read back.

- **MPC3024A_INP_PIN_set**

Format : u32 status = MPC3024A_INP_PIN_set(u8 CardID, u8 axis, u8 enable, u8 inp_logic)

Purpose: To configure the INP pin(in position input).

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
enable	u8	0: treat INP PIN as a general input. 1: treat INP PIN as a dedicated in position input.
inp_logic	u8	0: setting the pin connects or equals to GND level make this pin active logic. 1: setting the pin floats or equals to +24V makes this signal active logic.

Note: On wiring board terminal marked as INP

- **MPC3024A_INP_PIN_read**

Format : u32 status = MPC3024A_INP_PIN_read(u8 CardID, u8 axis, u8* enable, u8* inp_logic, u8 *state)

Purpose: Readback of configuration of the INP pin(in position input).

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
enable	u8	0: treat INP PIN as a general input. 1: treat INP PIN as a dedicated in position input.
inp_logic	u8	0: setting the pin connects or equals to GND level make this pin active logic. 1: setting the pin floats or equals to +24V makes this signal active logic.
state	u8	state of INP pin

Note on INP function:

Name	Description
INP	INP pin is in position function input pin. In a pulse type control system, the pulse is generated by the processor and the driver accepts the pulse train doing the motion job and feedback control. When the processor finishes the pulse generating work, do not means the servo driver finishes the positioning, the INP output of driver ensures the completeness of positioning and accuracy. If you enable INP function, the motion control will not continue even the pulse generating is complete (processor BUSY) until the INP signal received.

- **MPC3024A_ERC_PIN_set**

Format : **u32 status = MPC3024A_ERC_PIN_set(u8 CardID, u8 axis, u8 enable, u8 erc_logic, u8 erc_on_time, u8 erc_off_time)**

Purpose: To configure the ERC pin(error counter clear output).

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
enable	u8	0: treat ERC PIN as a manual error counter clear output. 1: treat ERC PIN as a automatic error counter clear output.
erc_logic	u8	0: setting the pin connects or equals to GND level make this pin active logic. 1: setting the pin floats or equals to +24V makes this signal active logic.
erc_on_time	u8	0: on time 12us 1: on time 102us 2: on time 408us 3: on time 1.6ms 4: on time 13ms 5: on time 52ms 6: on time 104ms 7: erc level out
erc_off_time	u8	0: off time 0s 1: off time 12us 2: off time 1.6ms 3: off time 104ms

Note: ERC signal will generate automatically on HOMing mode (ref. 9.5 Homing) you choose.

● MPC3024A_ERC_PIN_read

Format : **u32 status = MPC3024A_ERC_PIN_read(u8 CardID , u8 axis , u8* enable, u8* erc_logic, u8* erc_on_time, u8* erc_off_time, u8 *state)**

Purpose: To configure the ERC pin (error counter clear output).

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
enable	u8	0: treat ERC PIN as a manual error counter clear output. 1: treat ERC PIN as a automatic error counter clear output.
erc_logic	u8	0: setting the pin connects or equals to GND level make this pin active logic. 1: setting the pin floats or equals to +24V makes this signal active logic.
erc_on_time	u8	0: on time 12us 1: on time 102us 2: on time 408us 3: on time 1.6ms 4: on time 13ms 5: on time 52ms 6: on time 104ms 7: erc level out
erc_off_time	u8	0: off time 0s 1: off time 12us 2: off time 1.6ms 3: off time 104ms
state	u8	state of ERC pin

Note on ERC function:

Name	Description
ERC	ERC pin is error counter clear output pin. In a pulse type control system, the pulse is generated by the processor and the driver accepts the pulse train doing the motion job and feedback control. During homing, the processor detect the home(ORG) sensor and stop the pulse train, but the driver does not know the system is ‘homed’, the remain clock (which is accumulated in error counter) should be cleared to keep the system accuracy. While enables this function, the ERC output will be triggered automatically by the conditions met, and new motion command will not accept until the ERC output time out complete.(erc_on_time + erc_off_time). If you disable it (ie. manual control mode), use MPC3024A_output_point_set to control ERC, the active state of ERC will also stop the motion pulses. Do not use ERC as general output.

- **MPC3024A_ALM_PIN_set**

Format : **u32 status = MPC3024A_ALM_PIN_set(u8 CardID, u8 axis, u8 alm_logic, u8 alm_action)**

Purpose: To configure the ALM pin (servo driver alarm input).

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
alm_logic	u8	0: setting the pin connects or equals to GND level make this pin active logic. 1: setting the pin floats or equals to +24V makes this signal active logic.
alm_action	u8	0: immediate stop 1: decelerate to stop

- **MPC3024A_ALM_PIN_read**

Format : **u32 status = MPC3024A_ALM_PIN_read(u8 CardID, u8 axis, u8* alm_logic , u8* alm_action, u8*state)**

Purpose: Readback configuration of the ALM pin (servo driver alarm input).

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
alm_logic	u8	0: setting the pin connects or equals to GND level make this pin active logic. 1: setting the pin floats or equals to +24V makes this signal active logic.
alm_action	u8	0: immediate stop 1: decelerate to stop
state	u8	staet of ALM pin

HOMING related I/O

The HOMing position is a dedicated point that system take it as reference, the motion card clears the position counter (maybe the motion still moves) while the mechanism cross the homing point. (various kind of HOMing modes, refer 9.5 Homing)

The polarity (logic) of HOME (ORG) limit switch and encoder zero phase should be configured before homing,

MPC3024A_HOME_PIN_logic_set( )

MPC3024A_HOME_PIN_logic_read( ) for configuration read back.

MPC3024A_EZ_PIN_logic_set( ) will do.

MPC3024A_EZ_PIN_logic_read( ) for configuration read back.

Some servo driver related output and TTL output can be controlled by:

MPC3024A_point_set( ) (refer 9.3 I/O configuration and control TTL I/O and output controlTTL I/O and output controlTTL I/O and output controlTTL I/O and output controlTTL I/O and output control)

Maybe you will check the digital input status

MPC3024A_point_read( ) will give you the result.

TTL I/O and output controlTTL I/O and output controlTTL I/O and output controlTTL I/O and output controlTTL I/O and output control

- **MPC3024A_HOME_PIN_logic_set**

Format : **u32 status = MPC3024A_HOME_PIN_logic_set(u8 CardID, u8 axis,
u8 home_logic)**

Purpose: To configure the HOME(ORG) pin logic.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
home_logic	u8	0: setting the pin connects or equals to GND level make this pin active logic. 1: setting the pin floats or equals to +24V makes this signal active logic.

Note: On wiring board terminal marked as ORG for HOME signal.

- **MPC3024A_HOME_PIN_logic_read**

Format : **u32 status = MPC3024A_HOME_PIN_logic_read(u8 CardID, u8 axis,
u8* home_logic)**

Purpose: Readback configuration of the HOME(ORG) pin logic.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
home_logic	u8	0: setting the pin connects or equals to GND level make this pin active logic. 1: setting the pin floats or equals to +24V makes this signal active logic.

- **MPC3024A_EZ_PIN_logic_set**

Format : **u32 status = MPC3024A_EZ_PIN_logic_set(u8 CardID, u8 axis, u8 ez_logic)**

Purpose: To configure the EZ (Encoder Zero phase) logic.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
ez_logic	u8	0: setting the pin connects or equals to GND level make this pin active logic. 1: setting the pin floats or equals to +5V makes this signal active logic.

Note:

On wiring board terminal marked as EZ+, EZ- (differential input) for encoder zero phase input.

- **MPC3024A_EZ_PIN_logic_read**

Format : **u32 status = MPC3024A_EZ_PIN_logic_read(u8 CardID, u8 axis, u8* ez_logic)**

Purpose: To configure the EZ (Encoder Zero phase) logic.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
ez_logic	u8	0: setting the pin connects or equals to GND level make this pin active logic. 1: setting the pin floats or equals to +5V makes this signal active logic.

Motion related position counter I/O

The encoder feedback input and it working mode, position latch trigger input and position compare trigger output are the motion function related I/O. The signal comes from the JM1/2 connector,

EA+, EA-, EB+, EB-, EZ+, EZ--- encoder feedback

LTC --- position latch trigger input

CMP --- compare (equal) trigger output

For the encoder feedback input, you also have to configure the multiple rate and the physical input,

MPC3024A_pulse_inmode_set( ) will do. (ref. 9.19 Multi-function feedback counter)

MPC3024A_pulse_inmode_read( ) for configuration read back.

If your application needs to latch the encoder feedback at external trigger, use

MPC3024A_LTC_PIN_set( ) to configure the input pin.

MPC3024A_LTC_PIN_read( ) for configuration read back.

Compare function for you to generate a trigger pulse output at designed counter value, configure the output with:

MPC3024A_CMP_PIN_set( ).

MPC3024A_CMP_PIN_read( ) for configuration read back.

- **MPC3024A_LTC_PIN_set**

Format : **u32 status = MPC3024A_LTC_PIN_set(u8 CardID, u8 axis, u8 enable, u8 ltc_logic)**

Purpose: To configure the LTC pin (external trigger to latch input)**.**

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
enable	u8	0: treat LTC PIN as a general input. 1: treat LTC PIN as a dedicated external trigger to latch input.
ltc_logic	u8	0: setting the pin connects or equals to GND level make this pin active logic. 1: setting the pin floats or equals to +24V makes this signal active logic.

- **MPC3024A_LTC_PIN_read**

Format : **u32 status = MPC3024A_LTC_PIN_read(u8 CardID, u8 axis, u8* enable, u8* ltc_logic, u8* state)**

Purpose: Readback configuration of the LTC pin (external trigger to latch input).

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
enable	u8	0: treat LTC PIN as a general input. 1: treat LTC PIN as a dedicated external trigger to latch input.
ltc_logic	u8	0: setting the pin connects or equals to GND level make this pin active logic. 1: setting the pin floats or equals to +24V makes this signal active logic.
state	u8	state of LTC pin

- **MPC3024A_CMP_PIN_set**

Format : **u32 status = MPC3024A_CMP_PIN_set(u8 CardID, u8 axis, u8 cmp_mode)**

Purpose: To configure the CMP pin (compare equal output).

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
cmp_mode	u8	0: treat CMP PIN as a general output point. 1: treat CMP PIN as a dedicate output ,while comparator condition satisfied, this pin active to GND level (NMOS) or relay contactor short to COM. 2: treat CMP PIN as a dedicate output , while comparator condition satisfied, this pin active to floating level (NMOS) or relay contactor open to COM point.

- **MPC3024A_CMP_PIN_read**

Format : **u32 status = MPC3024A_CMP_PIN_read(u8 CardID, u8 axis, u8* cmp_mode, u8* state)**

Purpose: Readback configuration of the CMP pin(compare equal output).

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
cmp_mode	u8	0: treat CMP PIN as a general output point. 1: treat CMP PIN as a dedicate output ,while comparator condition satisfied, this pin active to GND level (NMOS) or relay contactor short to COM. 2: treat CMP PIN as a dedicate output , while comparator condition satisfied, this pin active to floating level (NMOS) or relay contactor open to COM point.
state	u8	state of CMP_OUT pin

TTL I/O and output control

The MPC3024A card also provides 2 nibble configurable TTL I/O (on JM3 IO7~IO0), configure it with:

MPC3024A_TTL_IO_mode_set( ).

MPC3024A_TTL_IO_mode_read( ) for configuration read back.

Some servo driver related output and TTL output can be controlled by:

MPC3024A_point_set( )

Maybe you will check the digital input status (of motion or driver related input and TTL input)

MPC3024A_point_read( ) will give you the result.

To output port dat to TTL I/O, using

MPC3024A_port_set( ) and read by

MPC3024A_port_read( )

- **MPC3024A_TTL_IO_mode_set**

Format : **u32 status = MPC3024A_TTL_IO_mode_set(u8 CardID, u8 IO_mode)**

Purpose: To configure the TTL I/O mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
IO_mode	u8	0: bit0~bit3 input, bit4~bit7 input. 1: bit0~bit3 output, bit4~bit7 input. 2: bit0~bit3 input, bit4~bit7 output. 3: bit0~bit3 output, bit4~bit7 output.

Note: On wiring board, the TTL I/O comes from/out of JM3 connector.

- **MPC3024A_TTL_IO_mode_read**

Format : **u32 status = MPC3024A_TTL_IO_mode_read(u8 CardID, u8* IO_mode)**

Purpose: Readback configuration of the TTL I/O mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Output:

Name	Type	Description
IO_mode	u8	0: bit0~bit3 input, bit4~bit7 input. 1: bit0~bit3 output, bit4~bit7 input. 2: bit0~bit3 input, bit4~bit7 output. 3: bit0~bit3 output, bit4~bit7 output.

● MPC3024A_point_set

Format : u32 status = MPC3024A_point_set(u8 CardID, u8 axis, u8 check_factor, u8 on_off)

Purpose: To set/reset output.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
check_factor	u8	0: ERC (servo error counter clear output) 1: SVON (servo on output) 2: FIN (finish output) 3: CMP (compare equal output) 4: IO_0 (TTL IO bit0 status) 5: IO_1 (TTL IO bit1 status) 6: IO_2 (TTL IO bit2 status) 7: IO_3 (TTL IO bit3 status) 8: IO_4 (TTL IO bit4 status) 9: IO_5 (TTL IO bit5 status) 10: IO_6 (TTL IO bit6 status) 11: IO_7 (TTL IO bit7 status)
on_off	u8	0: reset, inactive 1: set, active

Note on some output:

Name	Description
ERC	Ref. **Note on ERC function** MPC3024A_ERC_PIN_set
SVON	Servo on, output for user to control servo drive. At the power on stage, the driver should not operate until the motion processor is ready. Use SVON to control the driver. This is a dedicated output preserved for SVON and under control by user program, not by motion processor.
FIN	Motion finished, output for user to handshake with external control device. This is a dedicated output preserved for FIN and under control by user program, not by motion processor.
CMP	Ref. ***MPC3024A_CMP_PIN_***set

- **MPC3024A_point_read**

Format : **u32 status = MPC3024A_point_read(u8 CardID, u8 axis, u8 check_factor, u8 *state)**

Purpose: To input status.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X axis 2: Z axis	1: Y axis 3: A axis
check_factor	u8	0: SD 1: PCS 2: INP 3: ALM 4: SRDY 5: LS+(EL+) 6: LS-(EL-) 7: LTC 8: HOME(ORG) 9: EMG 10: EZ 11: ERC 12: SVON 13: FIN 14: CMP 15: CSTA 16: CSTP 17: IO_0 18: IO_1 19: IO_2 20: IO_3 21: IO_4 22: IO_5 23: IO_6 24: IO_7	(Slow Down input) (Position change start input) (In position input) (servo driver alarm input) (servo driver ready input) (positive side over travel limit switch) (negative side over travel limit switch) (external latch trigger input) (home(ORG) sensor input) (emergency input) (encoder zero phase input) (error counter output status) (servo driver on output status) (finish output) (compare equal output) (common start input) (common stop input) (TTL IO bit0 status) (TTL IO bit1 status) (TTL IO bit2 status) (TTL IO bit3 status) (TTL IO bit4 status) (TTL IO bit5 status) (TTL IO bit6 status) (TTL IO bit7 status)

Output:

Name	Type	Description
state	u8	0: in-active 1: active

- **MPC3024A_port_set**

Format : **u32 status = MPC3024A_port_set(u8 CardID, u8 data)**

Purpose: To set/reset TTL_IO port

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
data	u8	TTL/IO port data bit7: IO7 ... b0: IO0

- **MPC3024A_port_read**

Format : **u32 status = MPC3024A_port_read(u8 CardID, u8 *data)**

Purpose: To input TTL_IO port status.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Output:

Name	Type	Description
data	u8	TTL/IO port data bit7: IO7 ... b0: IO0

9.4 Velocity mode motion

Prepare for motion control

Before doing any motion movement, please make sure the over-travel protection is not active. Once any of the over-travel limit (LS+ or LS-) is active, the motion will be un-available (refer ***MPC3024A_EL_mode_set*** and the polarity can be set by on card DIP switch) . Please also notice the output pulse type of the driver you are using, adjust the output pulse mode to meet the driver. If the signal does not match (refer ***MPC3024A_pulse_outmode_set***), you can also have an unsuspected movement.

Velocity motion control is one of the functions of MPC3024A card. For safety reason or others to set the maximum speed is recommended. Use

MPC3024A_velocity_range_fix( ) to set the maximum allowable speed.

MPC3024A_velocity_range_unfix( ) to release the limit.

To have a smooth motion of velocity motion, acceleration and deceleration is required at start and stop. Use

MPC3024A_T_velocity_move( ) to move at trapezoidal profile.

MPC3024A_S_velocity_move( ) to move at S curve profile.

If you want to change speed or stop it,use

MPC3024A_velocity_change( ) to change speed.

MPC3024A_stop( ) to stop motion on any or all axes immediately or decelerate to stop.

To verify the speed use:

MPC3024A_velocity_read( ) will give you the current speed.

- **MPC3024A_velocity_range_fix**

Format : **u32 status = MPC3024A_velocity_range_fix(u8 CardID, u8 axis, i32 Vmax)**

Purpose: To set the maximum allowable speed.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
Vmax	i32	max pps (0~6553500)

- **MPC3024A_velocity_range_unfix**

Format : **u32 status = MPC3024A_velocity_range_unfix(u8 CardID, u8 axis)**

Purpose: To release the maximum allowable speed.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

- **MPC3024A_T_velocity_move**

Format : **u32 status = MPC3024A_T_velocity_move(u8 CardID, u8 axis, i32 VL, i32 VH, u32 Tacc_ms)**

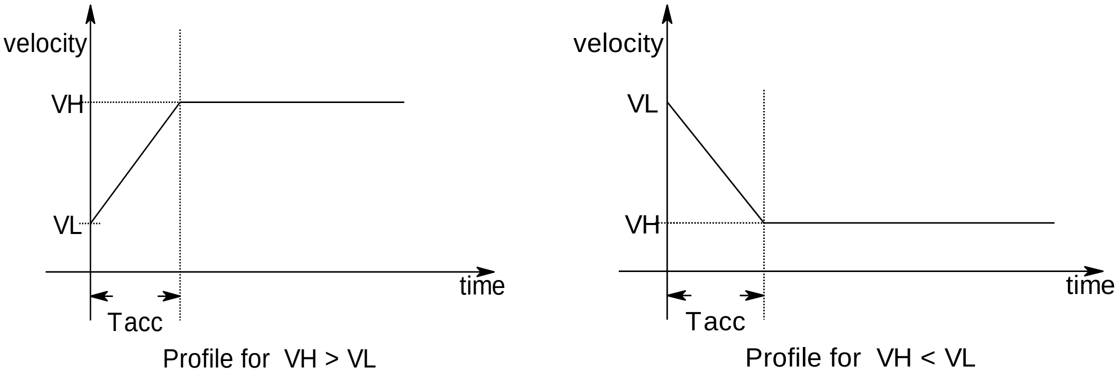
Purpose: Doing velocity mode movement at trapezoidal profile.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
VL	i32	pps, -6553500~6553500, negative value for reverse direction
VH	i32	pps, -6553500~6553500 negative value for reverse direction
Tacc_ms	u32	acc time in Milliseconds

Note on trapezoidal velocity mode:



MPC3024A_S_velocity_move

Format: **u32 status = MPC3024A_S_velocity_move(u8 CardID, u8 axis, i32 VL, i32 VH, u32 Tacc_ms, u32 TSacc_ms)**

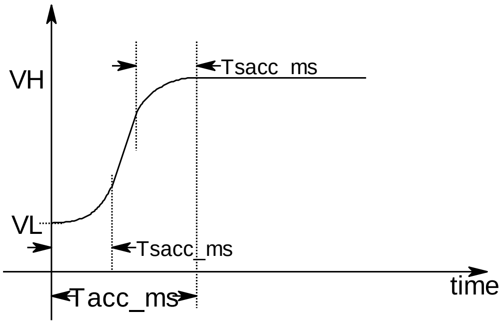
Purpose: Doing velocity mode movement at S curve profile.

Parameters:

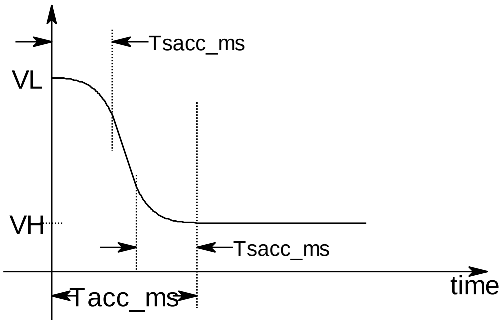
Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
VL	i32	pps, -6553500~6553500 negative value for reverse direction
VH	i32	pps, -6553500~6553500 negative value for reverse direction
Tacc_ms	u32	Milliseconds
TSacc_ms	u32	Mili-second difference of s curve range

Note on S curve velocity mode:



Profile for VH > VL



Profile for VH < VL

- **MPC3024A_velocity_change**

Format : **u32 status = MPC3024A_velocity_change(u8 CardID, u8 axis, i32 Vn, u32 Tacc_ms)**

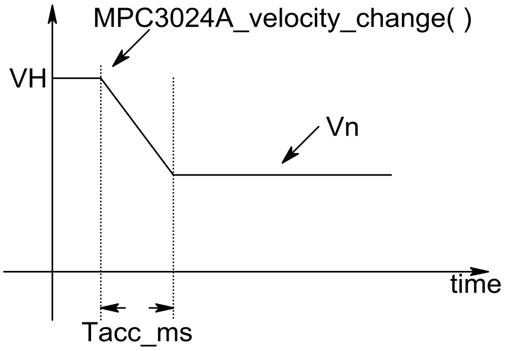
Purpose: Change speed (with the trapezoidal/S curve mode previously defined) at velocity mode.

Parameters:

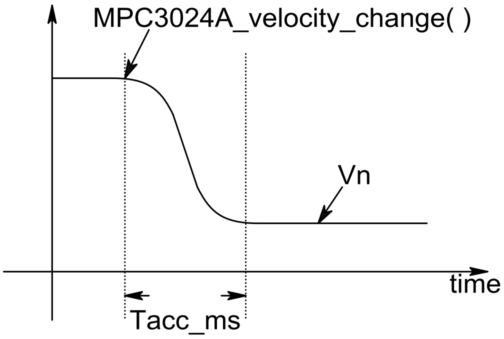
Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
Vn	i32	new speed in pps, -6553500~6553500 |Vn| ≤ Vmax (set by MPC3024A_velocity_range_fix( ))
Tacc_ms	u32	acc time in Milliseconds

Note on velocity change:



Velocity change at trapzoidal acc/dec



Velocity change at S curve acc/dec

* If you use **MPC3024A_velocity_change** to change speed, while you want to change direction, be sure to use to decrease the speed to zero before change direction. The functions **MPC3024A_S_velocity_move** and **MPC3024A_T_velocity_move** are no need to switch to zero speed.

- **MPC3024A_stop**

Format: **u32 status = MPC3024A_stop(u8 CardID, u8 axis[4], u32 Tdec_ms)**

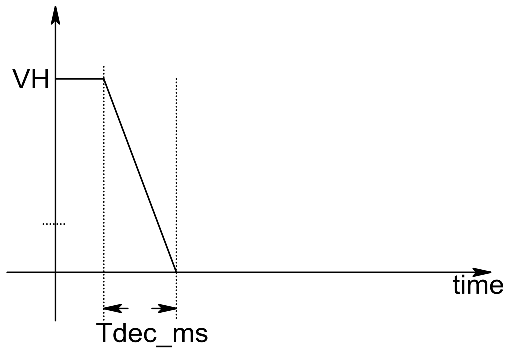
Purpose: Command to decelerate to stop or immediately stop (with the trapezoidal/S curve mode previously defined).

Parameters:

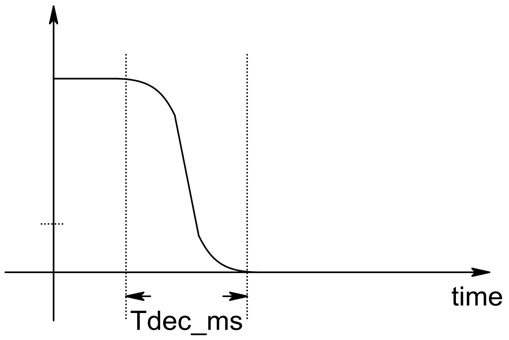
Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis[4]	u8	axis[0]: X axis, 0: not stop (normal operation) 1: stop axis[1]: Y axis, 0: not stop (normal operation) 1: stop axis[2]: Z axis, 0: not stop (normal operation) 1: stop axis[3]: A axis, 0: not stop (normal operation) 1: stop
Tdec_ms	u32	decelerate to stop time in Milliseconds if Tdec_ms=0, immediate stop

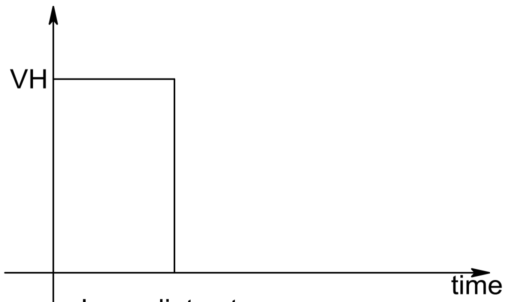
Note on decelerate to stop:



Dec stop at trapzoidal acc/dec



Dec stop at S curve acc/dec



Immediate stop

- **MPC3024A_velocity_read**

Format : **u32 status = MPC3024A_velocity_read(u8 CardID, u8 axis, f64 *speed)**

Purpose: To read the current speed.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
speed	f64	current speed in pps

Note: The old version dll, speed has the type f64, the new dll changes to i32.

9.5 Homing

Before doing any motion movement, please make sure the over-travel protection is not active. Once any of the over-travel limit (LS+ or LS-) is active, the motion will be un-available (refer ***MPC3024A_EL_mode_set*** and the polarity can be set by on card DIP switch) . Please also note the output pulse type of the driver you are using, adjust the output pulse mode to meet the driver. If the signal does not match (refer ***MPC3024A_pulse_outmode_set***), you can also have a unsuspected movement.

At the beginning of positioning or contouring control, MPC3024A (controller) must know the initial coordinate it is. Various kinds of homing modes provide you flexible choice of implementation of getting the initial coordinate.

Before homing, the polarity (logic) of HOME(ORG) limit switch and encoder zero phase should be configured, (ref. ***MPC3024A_HOME_PIN_logic_set, MPC3024A_HOME_PIN_logic_read, MPC3024A_EZ_PIN_logic_***set***, MPC3024A_EZ_PIN_logic_read)***.

At the beginning of any positioning or contouring motion control, HOMING is a must.

Use

MPC3024A_home_mode_set( ) to select the desired homing mode.

MPC3024A_home_start( ) to execute homing.

After homing you may want to initialize the coordinate of the home(ORG) position, use

MPC3024A_current_position_set( ) to setup the coordinate at any time and any point, if the motion is ready.

Any time, you want to get the coordinate,

MPC3024A_current_position_read( ) will do.

MPC3024A_home_search( ) will seek home limit switch automatically and correct the position.

Note: There are many kinds of HOMING mode you can choose but after HOMING the machine do not always stop on coordinate reading "0". It is caused by the servo driver remained pulses or HOMING mode characteristic. If you want the coordinate reading at "0" just after doing a HOMING, we suggest you use absolute move command to move to coordinate "0" position after HOMING.

Use ***MPC3024A_stop( )*** (refer MPC3024A_stop) to stop motion on any or all axes immediately or decelerate to stop.

- **MPC3024A_home_mode_set**

Format: **u32 status = MPC3024A_home_mode_set(u8 CardID, u8 axis, u8 mode, u8 EZ_count)**

Purpose: To configure the homing mode.

Parameters:

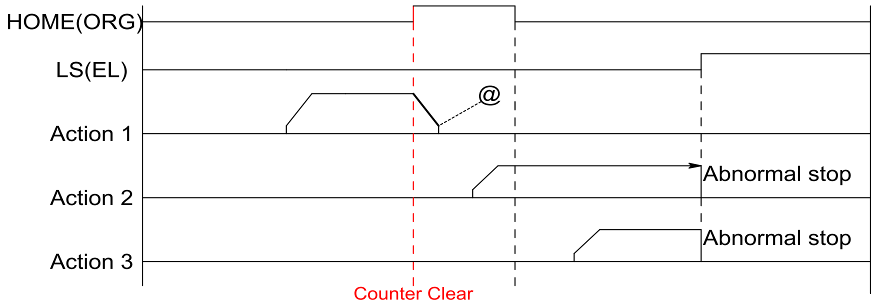
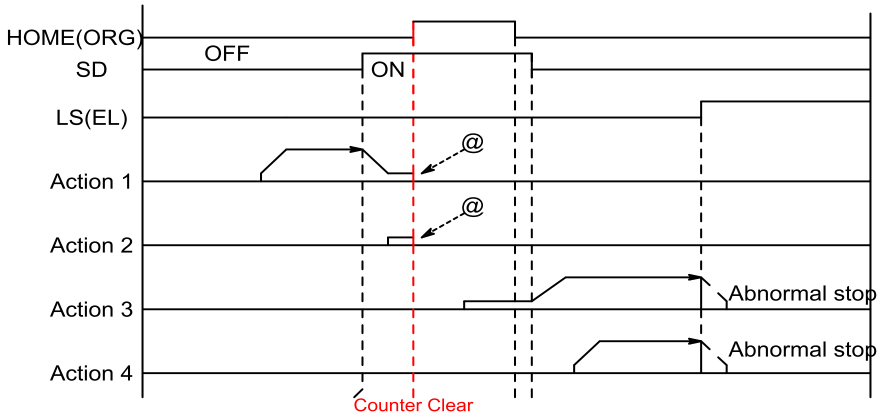
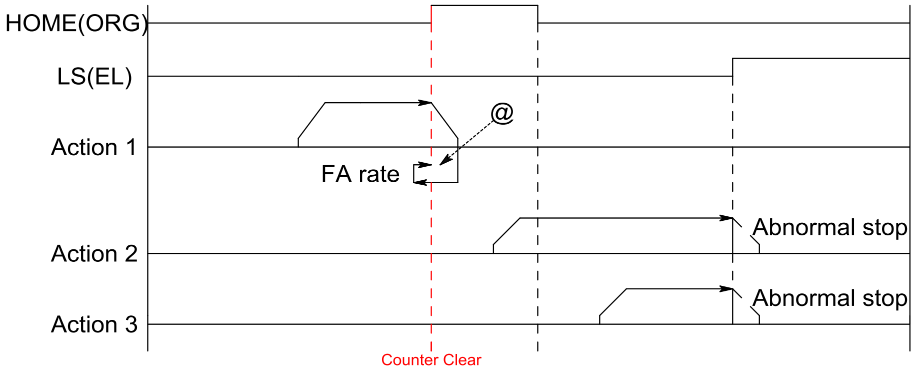
Input:

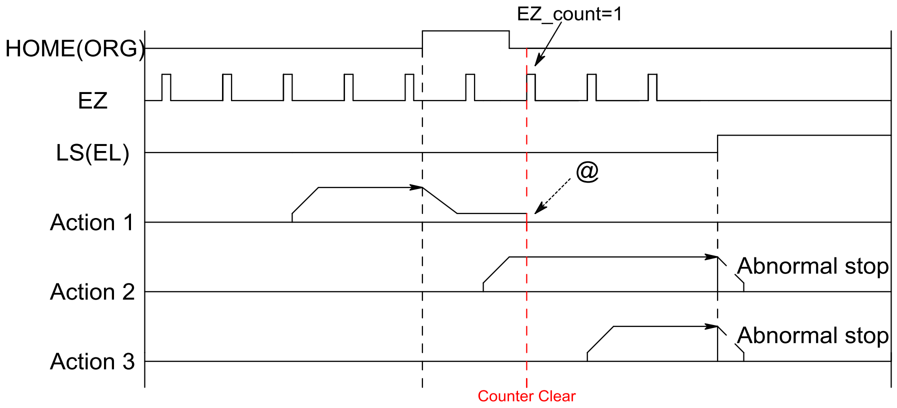
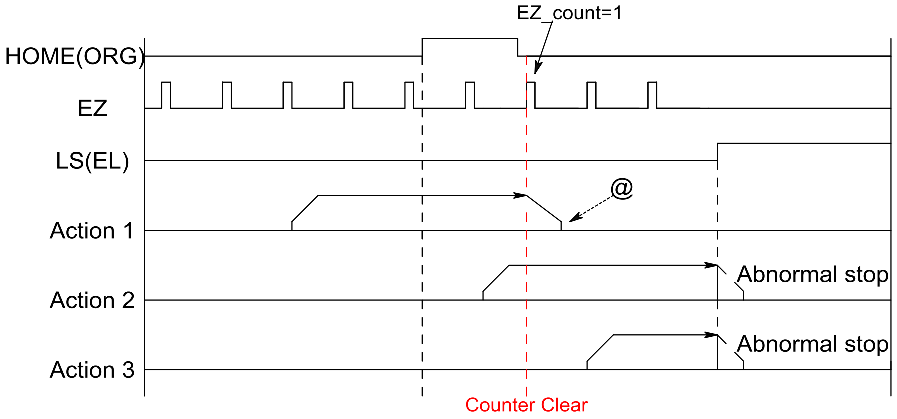
Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
mode	u8	homing mode 0~12$_{(10)}$
EZ_count	u8	Clear current position counter at the pulse numbers of zero phase input after home(ORG) switch is activated. EZ_count=0, means 1 zero phase count. … EZ_count=15 means 16 zero phase count. EZ_count maximum is 15$_{(10)}$

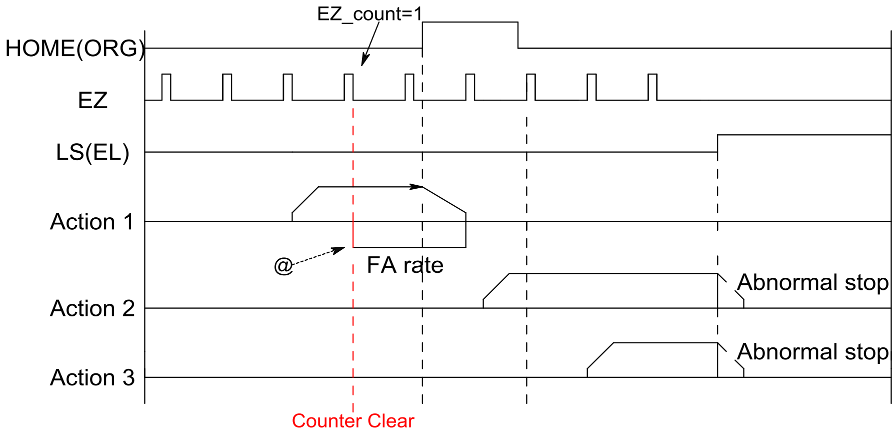
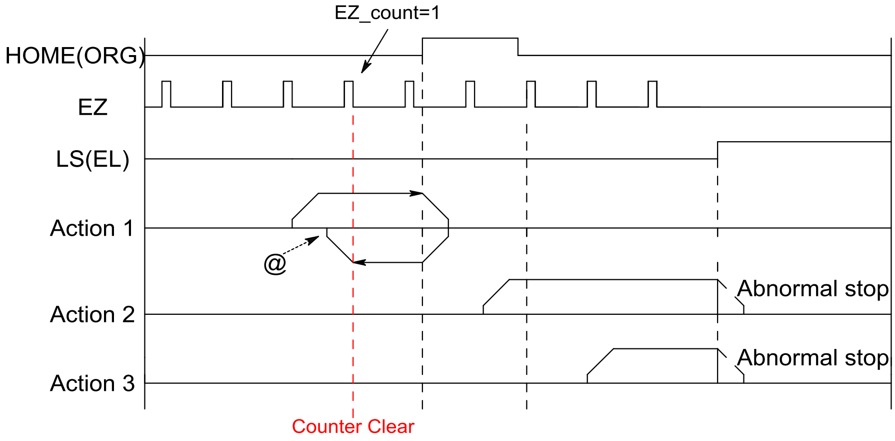
Note on homing mode:

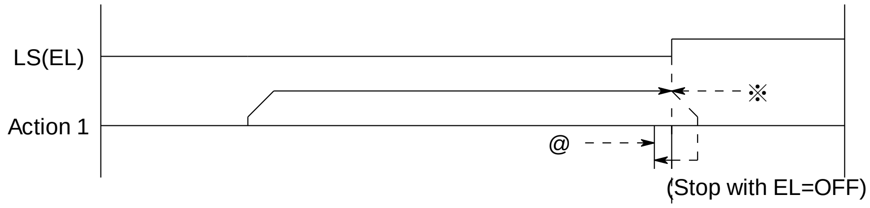
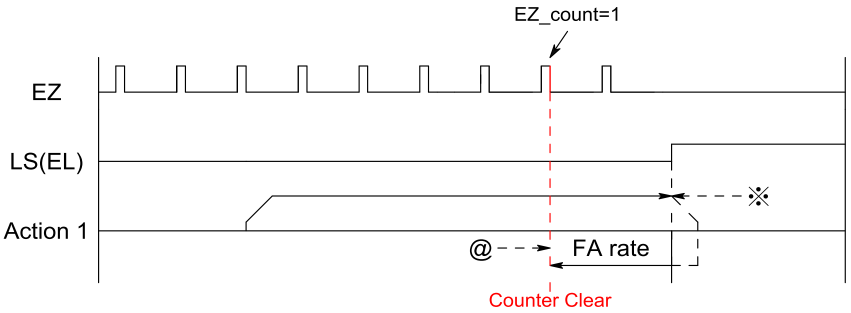
The mark @ is the position if you enable ERC pin the ERC signal will be active.

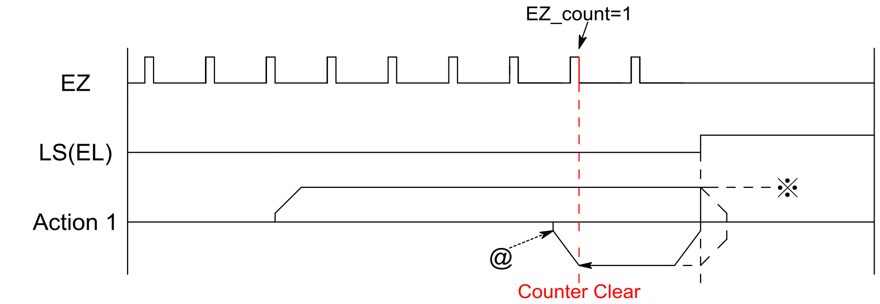
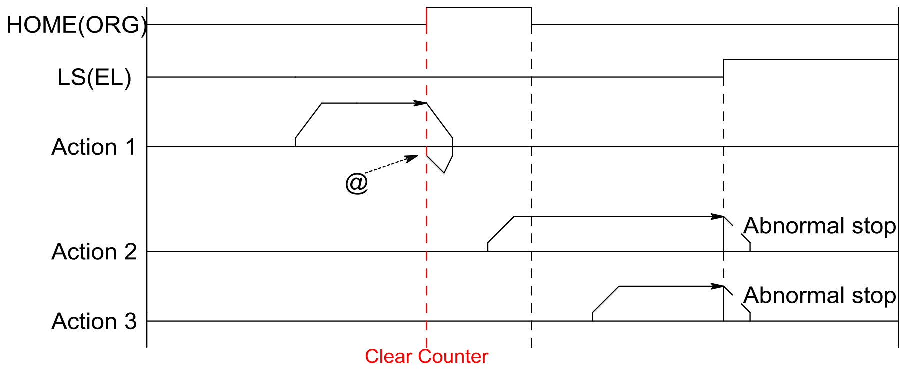
The ERC function is configured by **MPC3024A_ERC_PIN_set( ).**

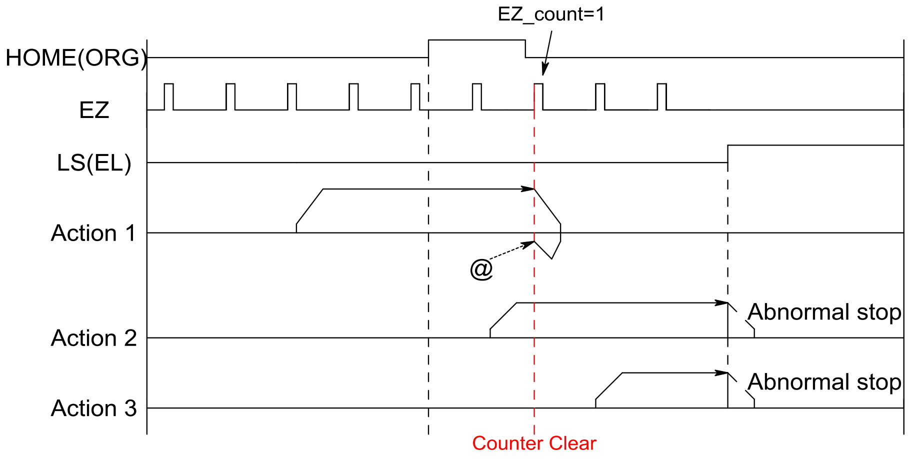
Mode	Description
0	 HOME(ORG) LS(EL) Action 1 @ Action 2 Abnormal stop Action 3 Abnormal stop Counter Clear  HOME(ORG) OFF SD ON LS(EL) @ Action 1 @ Action 2 Action 3 Abnormal stop Action 4 Abnormal stop Counter Clear Mode0: HOME(ORG) signal turning from OFF to ON causes stop after deceleration to VL speed. The counter is reset upon HOME(ORG) signal turning from OFF to ON. Note : @ position is the ERC signal output timing with “ automatic output of ERC signal “ made valid at the stop of origin return.
1	 HOME(ORG) LS(EL) @ Action 1 FA rate Action 2 Abnormal stop Action 3 Abnormal stop Counter Clear Mode1: HOME(ORG) signal turning from OFF to ON causes stop after deceleration to VL speed ,then the chip generates pulses until the HOME(ORG) signal turns from ON to OFF and after the signal turns off, it generates pulses at the RFA rate (Backlash speed) in initial direction and immediately stops when the HOME(ORG) signal turns from OFF to ON again. The counter is reset upon the HOME(ORG) signal turning from OFF to ON. Note : @ position is the ERC signal output timing with “ automatic output of ERC signal “ made valid at the stop of origin return.

2	 EZ_count=1 HOME(ORG) EZ LS(EL) @ Action 1 Abnormal stop Action 2 Abnormal stop Action 3 Counter Clear Mode2: The chip decelerates pulse output to VL when the HOME(ORG) signal turns from OFF to ON and stops immediately upon the EZ counter counting up to the preset value. The counter is reset upon the EZ counter counting up to the preset value. Note : @ position is the ERC signal output timing with “ automatic output of ERC signal “ made valid at the stop of origin return.
3	 EZ_count=1 HOME(ORG) EZ LS(EL) @ Action 1 Abnormal stop Action 2 Abnormal stop Action 3 Counter Clear Mode3: The chip decelerates to VL speed and stops pulse output upon the EZ counter counting up to the preset value after the HOME(ORG) signal turns from OFF to ON. The counter is reset upon the EZ counter counting up to the preset value. Note : @ position is the ERC signal output timing with “ automatic output of ERC signal “ made valid at the stop of origin return.

4	 EZ_count=1 HOME(ORG) EZ LS(EL) Action 1 @ FA rate Action 2 Abnormal stop Action 3 Abnormal stop Counter Clear Mode4: The chip stops after deceleration to VL speed upon the HOME(ORG) signal turning from OFF to ON and them generates pulses in reverse direction at the RFA rate (Backlash speed) before immediate stop again upon the EZ counter counting up to the preset value. The counter is reset upon the EZ counter counting up to the preset value. Note : @ position is the ERC signal output timing with “ automatic output of ERC signal “ made valid at the stop of origin return.
5	 EZ_count=1 HOME(ORG) EZ LS(EL) Action 1 @ Action 2 Abnormal stop Action 3 Abnormal stop Counter Clear Mode5: The chip stops after deceleration to VL speed upon the HOME(ORG) signal turning from OFF to ON and then generates pulses in reverse direction before stop after deceleration to VL speed upon the EZ counter counting up to the preset value. The counter is reset upon the EZ counter counting up to the preset value. Note : @ position is the ERC signal output timing with “ automatic output of ERC signal “ made valid at the stop of origin return.

6	 Mode6: The chip immediately stops pulse output (stops after deceleration if ELM (el_mode)=1) upon the LS(EL) signal turning ON and then generates pulses in reverse direction at the RFA rate (Backlash speed) before immediate stop again upon the LS(EL) signal turning off. The counter is reset when the LS(EL) signal turns off. Note : @ position is the ERC signal output timing with " automatic output of ERC signal " made valid at the stop of origin return.
7	 EZ_count=1 EZ LS(EL) ※ Action 1 @ FA rate Counter Clear Mode7: The chip immediately stops pulse output (stops after deceleration if ELM (el_mode) =1) and then generates pulses in reverse direction at the RFA rate (Backlash speed) before immediate stop again upon the EZ counter counting up to the preset value. The counter is reset at the immediate stop upon the EZ counter counting up to the preset value. Note : @ position is the ERC signal output timing with " automatic output of ERC signal " made valid at the stop of origin return.

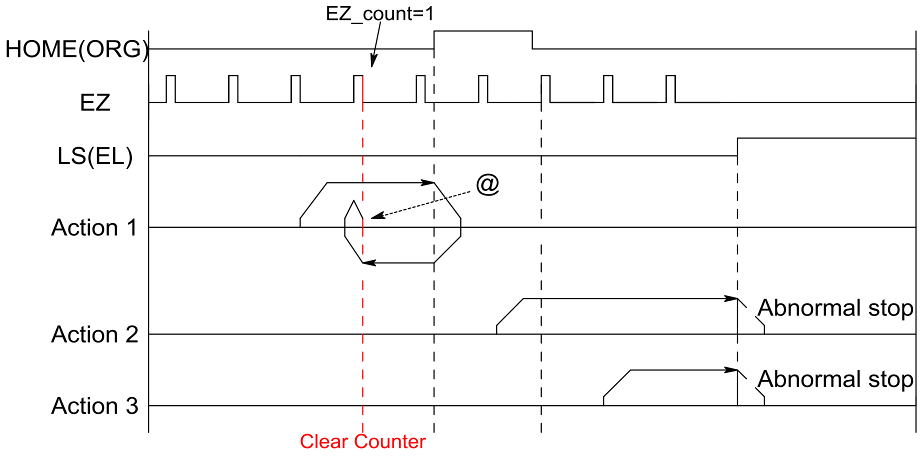
8	 EZ_count=1 EZ LS(EL) Action 1 ※ @ Counter Clear Mode8: The chip immediately stops pulse output (stops after deceleration if ELM (el_mode) =1) and then generates pulses in reverse direction before stop after deceleration to VL speed upon the EZ counter counting up to the preset value. The counter is reset upon the EZ counter counting up to the preset value. Note : @ position is the ERC signal output timing with “ automatic output of ERC signal “ made valid at the stop of origin return.
9	 HOME(ORG) LS(EL) Action 1 @ Action 2 Abnormal stop Action 3 Abnormal stop Clear Counter Mode9: After performing origin return mode 0, the chip generates pulses to return to 0 point, that is, until the FB counter counts down to 0. Note : @ position is the ERC signal output timing with “ automatic output of ERC signal “ made valid at the stop of origin return. **Special Note: This homing mode is especially useful to work in the system with the linear scale feedback**.

<table>
<tr>
<td>10</td>
<td>


Mode10:

After performing origin return mode 3, the chip generates pulses to return to 0 point, that is, until the FB counter counts down to 0.

Note : @ position is the ERC signal output timing with “ automatic output of ERC signal “ made valid at the stop of origin return.

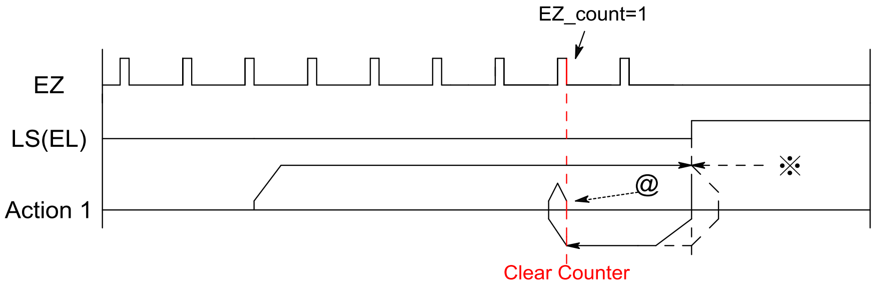
Special Note: This homing mode is especially useful to work in the system with the linear scale feedback.
</td>
</tr>
<tr>
<td>11</td>
<td>


Mode11:

After performing origin return mode 5, the chip generates pulses to return to 0 point, that is, until the FB counter counts down to 0.

Note : @ position is the ERC signal output timing with “ automatic output of ERC signal “ made valid at the stop of origin return.

Special Note: This homing mode is especially useful to work in the system with the linear scale feedback.
</td>
</tr>
</table>

12	 Mode12: After performing origin return mode 8, the chip generates pulses to return to 0 point, that is, until the FB counter counts down to 0. Note : @ position is the ERC signal output timing with “ automatic output of ERC signal “ made valid at the stop of origin return. **Special Note: This homing mode is especially useful to work in the system with the linear scale feedback**.

● MPC3024A_home_start

Format : **u32 status = MPC3024A_home_start(u8 CardID, u8 axis, i32 VL, i32 VH, u32 Tacc_ms, u8 direction)**

Purpose: To command the homing motion.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
VL	i32	pps of start speed (0~6553500)
VH	i32	pps of final speed (0~6553500)
Tacc_ms	u32	acceleration time in miliseconds
direction	u8	direction of homing 0: positive direction 1: negative direction

- **MPC3024A_current_position_set**

Format : **u32 status = MPC3024A_current_position_set(u8 CardID, u8 axis, i32 current_posi)**

Purpose: To setup the coordinate of current position.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
current_posi	i32	coordinate value, -134,217,728≦ current_posi ≦134,217,727

Note on set current position:

The current position can set only at the motion ready (not in movement).

- **MPC3024A_current_position_read**

Format : **u32 status = MPC3024A_current_position_read(u8 CardID, u8 axis, i32 *current_posi)**

Purpose: To readback the coordinate of current position.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
current_posi	i32	coordinate value, -134,217,728≦ current_posi ≦134,217,727

Note on read current position:

Current position is cleared at application initialization (3024A_initial( )) and homing.

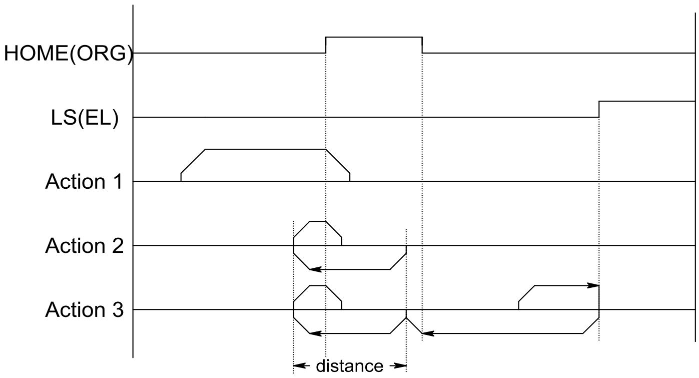
- **MPC3024A_home_search**

Format : **u32 status = MPC3024A_home_search(u8 CardID, u8 axis,**
i32 VL, i32 VH, u32 Tacc_ms, u8 direction, u32 distance)

Purpose: To command origin search mode homing motion.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
VL	i32	pps of start speed (0~6553500)
VH	i32	pps of final speed (0~6553500)
Tacc_ms	u32	acceleration time
direction	u8	direction of homing 0: positive direction 1: negative direction
distance	u32	 HOME(ORG) LS(EL) Action 1 Action 2 Action 3 distance

9.6 Backlash compensation

For accuracy positioning, the backlash compensation is required, the backlash function will compensate the backlash error only on the motion direction is changed. It will compensate before doing motion.

MPC3024A_backlash_comp_set( ) is the function you need.
MPC3024A_backlash_comp_read( ) to readback parameters.

- **MPC3024A_backlash_comp_set**

Format : u32 status = MPC3024A_backlash_comp_set(u8 CardID, u8 axis,
u16 backlash_pulse, u8 enable, u32 backlash_speed)

Purpose: To setup backlash compensation.

Parameters:

Input:

Name	Type	Description
CardID	u8	Assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
backlash_pulse	u16	backlash pulse (0 ≦backlash_pulse≦4095)
enable	u8	0: disable backlash compensation 1: enable backlash compensation from next direction change
backlash_speed	u32	backlash speed (pps) (0 ≦backlash_speed≦6553500)

- **MPC3024A_backlash_comp_read**

Format : **u32 status = MPC3024A_backlash_comp_read(u8 CardID, u8 axis, u16* backlash_pulse, u8* enable, u32* backlash_speed)**

Purpose: Readback configuration of backlash compensation.

Parameters:

Input:

Name	Type	Description	
CardID	u8	Assigned by DIP/ROTARY SW	
axis	u8	0: X axis 2: Z axis	1: Y axis 3: A axis

Output:

Name	Type	Description
backlash_pulse	u16	backlash pulse (0 ≦backlash_pulse≦4095)
enable	u8	0: disable backlash compensation 1: enable backlash compensation from next direction change
backlash_speed	u32	backlash speed (pps) (0 ≦backlash_speed≦6553500)

9.7 Conditional start of motion control

MPC3024A provides 3 kinds of conditional start of motion:

-- immediate act to start motion
-- wait for CSTA to start motion
-- wait for compare start to start motion

Immediate act

The immediate act allows you to start the motion control as your application calls the dll. It will run as soon as possible and wait for nothing.

Wait for CSTA

The wait for CSTA to start will wait for a CSTA active signal to trigger the motion, when your application calls the dll function, the motion will not run but wait for the incoming active trigger of the CSTA signal. As soon as the signal comes in the motion starts to run. It can be used to cross card control, say two motion control cards work together to implement a specific application; we wish some axes on card A will start to run simultaneously with some axes on card B. Of course if you use immediate act mode will be quite accurate as the computer runs fast, if you wish to start both axes from an external signal, you do not need a polling just configure the wait for CSTA mode to precisely start the axes on different cards.

The CSTA input pin can work as input or output depends on your command used. Normally it is work as input to accept external start trigger. The master motion control card issues the trigger command to start simutaneously. Be sure that the slave must use the wait mode in CSTA wait. If you will use the CSTA on card (refer hardware manual JM1,JM2 definition) as output to send out the CSTA trigger:

MPC3024A_CSTA_trigger( ) will do.

The MPC card also provide external trigger to stop input CSTP which normally used as input to accept external stop trigger. If you want to use CSTP as trigger stop output:

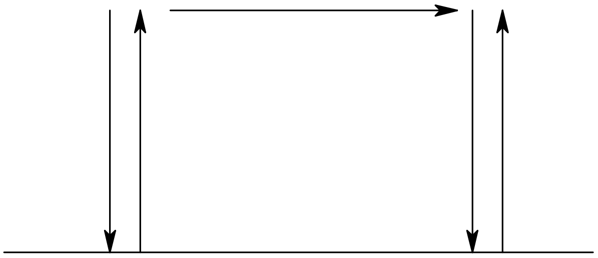
MPC3024A_CSTP_trigger( ) will do.

Wait for compare start

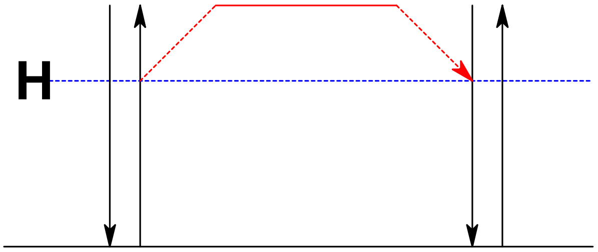
The most flexible function of conditional start is wait for compare start mode. You can have a motion command ready then wait for a specific trigger comes from the axis which will be moving or currently moving. Say a pick and place machine which has X,Y, Z 3 axes. X,Y are the coordinate of the pick and place mechanism and Z is the depth of the mechanism. Normal operation of a pick then place requires:

1. X,Y positioned to pick coordinate
2. Z drive down and pick
3. Z drive up
4. X,Y positioning to place coordinate
5. Z drive down and place
6. Z drive up

If we want to reduce the one cycle operation time, we must speed up the motion system. Is it possible to speed up without speed up the motion? If you can confirm the mechanism of Z axis will not have any conflict with other mechanism at a specific height H, you can have the X,Y start positioning while the Z axis go pass H point (the normally condition needs Z axis stop then X,Y positioning). Please refer the following figure as example: The up/ down axis is Z and the height H is confirmed as save height of movement. The left and right axis is X (we do not show Y as simplified model). The final trajectory of the pick and place mechanism will be the red line.



Normal operation of pick and place mechanizm



Speed up operation of pick and place mechanizm

To use the wait for compare start you must first choose the compare method fit for your application. There are several kinds of compare method to choose:

1: compare out at equal, and does not care direction

2: compare out at equal while counting up

3: compare out at equal while counting down

4: compare out at preset value > counter value

5: compare out at preset value < counter value

Secondly, select the source axis and the counter. Of course, any one of the available axis X, Y, Z, A, you can choose one. The counter may be

0: to compare with the current position command counter

1: to compare with the feedback counter

2: undefined (not available)

3: to compare with the pulser counter

Now you can setup the wait for compare start terms by:

MPC3024A_compare_start_set( ) and read back for verification by

MPC3024A_compare_start_read( )

The next you must setup the compare data

MPC3024A_compare_start_data_set( ) will do and read back for verification by

MPC3024A_compare_start_data_read( )

Now you can command the motion with wait parameter wait for compare start option, the motion is waiting for the signal come to trigger to run.

To verify if the compare start operation has done or not by:

MPC3024A_compare_start_flag_read( )

- **MPC3024A_CSTA_trigger**

Format : **u32 status = MPC3024A_CSTA_trigger(u8 CardID)**

Purpose: To trigger output of CSTA (START) signal from the assigned card.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

- **MPC3024A_CSTP_trigger**

Format : **u32 status = MPC3024A_CSTP_trigger(u8 CardID)**

Purpose: To trigger output of CSTP (STOP) signal from the assigned card.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

- **MPC3024A_compare_start_set**

Format : **u32 status = MPC3024A_compare_start_set(u8 CardID, u8 cmp_axis, u8 cmp_source, u8 cmp_method)**

Purpose: To configure the compare source and method of synchronous start.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
cmp_axis	u8	0: X 1: Y 2: Z 3: A
cmp_source	u8	0: to compare with the current position command counter 1: to compare with the feedback counter 2: undefined 3: to compare with the pulser counter
cmp_method	u8	1: compare out at equal, and does not care direction 2: compare out at equal while counting up 3: compare out at equal while counting down 4: compare out at preset value > counter value 5: compare out at preset value < counter value

Note: Only one compare axis can be selected for compare source.

- **MPC3024A_compare_start_read**

Format : **u32 status = MPC3024A_compare_start_read(u8 CardID, u8 cmp_axis, u8 *cmp_source, u8 *cmp_method)**

Purpose: To configure the compare source and method of synchronous start.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
cmp_axis	u8	0: X 1: Y 2: Z 3: A

Output:

Name	Type	Description
cmp_source	u8	0: to compare with the current position command counter 1: to compare with the feedback counter 2: undefined 3: to compare with the pulser counter
cmp_method	u8	1: compare out at equal, and does not care direction 2: compare out at equal while counting up 3: compare out at equal while counting down 4: compare out at preset value > counter value 5: compare out at preset value < counter value

- **MPC3024A_compare_start_data_set**

Format : **u32 status = MPC3024A_compare_start_data_set(u8 CardID, i32 cmp_data)**

Purpose: To configure the compared data.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
cmp_data	i32	The data to be compared

- **MPC3024A_compare_start_data_read**

Format : **u32 status = MPC3024A_compare_start_data_read(u8 CardID, i32 *cmp_data)**

Purpose: To configure the compared data.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

Output:

Name	Type	Description
cmp_data	i32	The data to be compared

- **MPC3024A_compare_start_flag_read**

Format : **u32 status = MPC3024A_compare_start_flag_read(u8 CardID,u8 *cmp_flag);**

Purpose: To read the compare start flag.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

Output:

Name	Type	Description
cmp_flag	u8	0: the compare condition not meet 1: the compare condition has met

9.8 Point to point motion control

Before doing any motion movement, please make sure the over-travel protection is not active. Once any of the over-travel limit (LS+ or LS-) is active, the motion will be un-available (refer ***MPC3024A_EL_mode_set*** and the polarity can be set by on card DIP switch) . Please also note the output pulse type of the driver you are using, adjust the output pulse mode to meet the driver. If the signal does not match (refer ***MPC3024A_pulse_outmode_set***), you can also have a unsuspected movement.

There are generally 2 types of motion concerning coordinate: absolute or relative (refer 6.4 Coordinate system) and also each motion command has conditional start selection (refer 9.7 conditional start of motion control).

You may control any of the 4 axes to work in point to point motion mode. Command to positioning

MPC3024A_T_position_move( ) for trapezoidal acc/dec profile.

MPC3024A_S_position_move( ) for S curve acc/dec profile.

For some special cases, you need to change target position while the point to point motion is running,

MPC3024A_position_change( ) will do. If you will change the positioning parameters on the fly, ***MPC3024A_T_onLINE_change( )*** will do in T curve mode or in S curve mode by:

MPC3024A_S_onLINE_change( )

If you will change the positioning parameters on the fly,

MPC3024A_T_onLINE_change( ) will do in T curve mode or in S curve mode by:

MPC3024A_S_onLINE_change( )

Use ***MPC3024A_stop( )*** (refer MPC3024A_stop) to stop motion on any or all axes immediately or decelerate to stop.

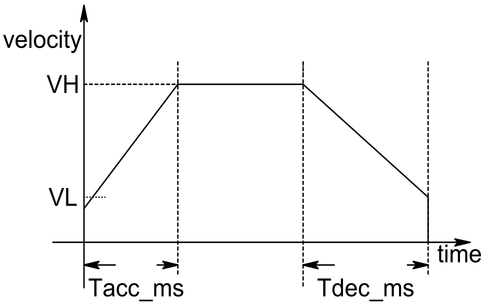
MPC3024A_T_position_move

Format : **u32 status = MPC3024A_T_position_move(u8 CardID, u8 axis, i32 Position, u8 posi_mode, i32 VL, i32 VH, u32 Tacc_ms, u32 Tdec_ms, u8 wait)**

Purpose: To point to point positioning at trapezoidal profile.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
Position	i32	relative distance to move absolute coordinate to move (-134,217,728 ≦ Position ≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 VH,VL:pps, (0 ≦ VH,VL ≦ 6553500) Tacc_ms,Tdec_ms: milliseconds.
VH	i32	
Tacc_ms	u32	
Tdec_ms	u32	
wait	u8	0: immediately act 1: wait for CSTA 2: wait for compare start

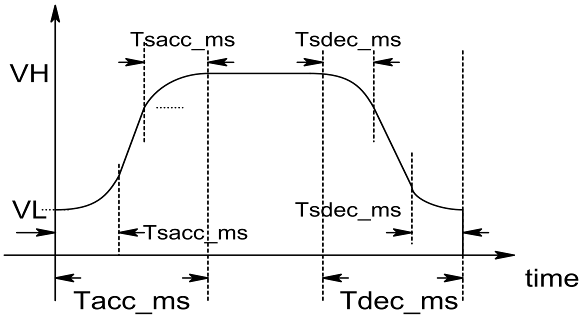
● MPC3024A_S_position_move

Format : **u32 status = MPC3024A_S_position_move(u8 CardID, u8 axis, i32 Position, u8 posi_mode, i32 VL, i32 VH, u32 Tacc_ms, u32 Tdec_ms, u32 TSacc_ms, u32 TSdec_ms, u8 wait)**

Purpose: To point to point positioning at S curve profile.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
Position	i32	0: relative distance to move 1: absolute coordinate to move (-134,217,728 ≦ Position ≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 Tsacc_ms Tsdec_ms VH VL Tsacc_ms Tsdec_ms time Tacc_ms Tdec_ms VH,VL : pps, (0 ≦ VH,VL ≦ 6553500) Tacc_ms,Tdec_ms: milliseconds. TSacc_ms,TSdec_ms: milliseconds, time of s curve range
VH	i32	
Tacc_ms	u32	
Tdec_ms	u32	
TSacc_ms	u32	
TSdec_ms	u32	
wait	u8	0: immediately act 1: wait for CSTA 2: wait for compare start

Note on point to point motion control:

1. Point to point motion control in continuous mode (MPC3024A_continuous_flag_set ( ), conti_flag=1), be sure to check continuous buffer (MPC3024A_continuous_buffer_no_read( )) until ‘remain_no’ not equal to 2,else the command will be defective.
2. In continuous mode, be sure to set maximum speed first (MPC3024A_velocity_range_fix( )).
3. In non-continuous mode(MPC3024A_continuous_flag_set( )，conti_flag=0), be sure to check (MPC3024A_motion_status_read( ); check_factor=0，ret_flag =1) to confirm the motion is ready.

MPC3024A_position_change

Format : **u32 status = MPC3024A_position_change(u8 CardID, u8 axis, i32 New_pos, u8 posi_mode)**

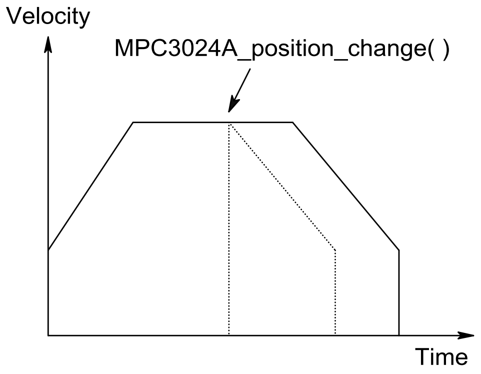
Purpose: To change positioning while point to point motion is running.

Parameters:

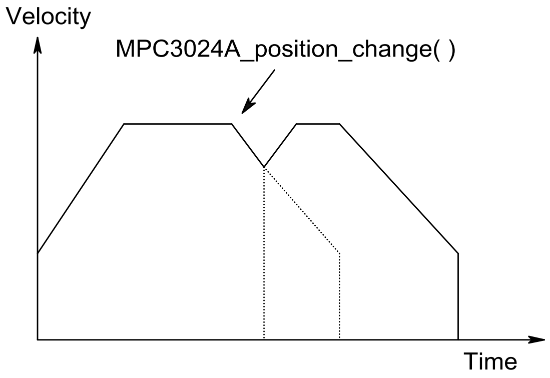
Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
New_pos	i32	new target position (-134,217,728 ≦ New_pos ≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute

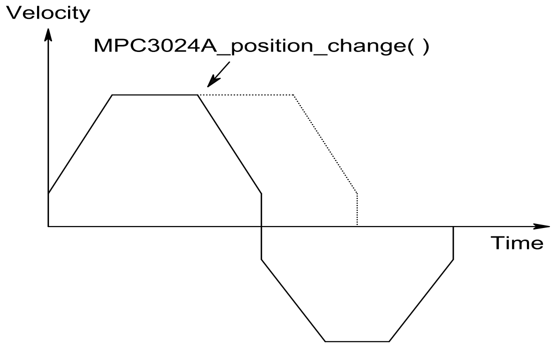
Note on position change:



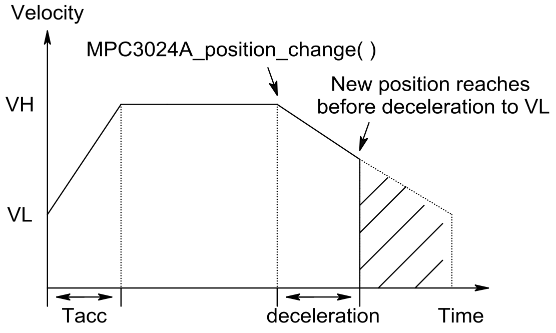
1. Command to change position at VH range



2. Command to change position at deceleration range



3. New position at different side



4. New position at mid-way of deceleration range

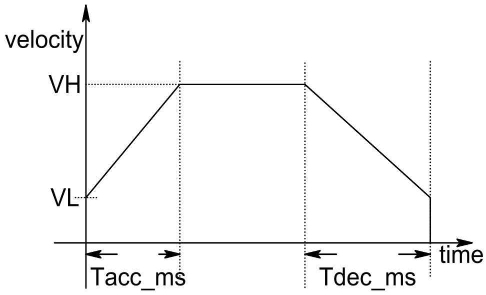
- **MPC3024A_T_onLINE_change**

Format : **u32 status = MPC3024A_T_onLINE_change(u8 CardID, u8 axis,**
i32 Position, u8 posi_mode, i32 VL, i32 VH, u32 Tacc_ms,
u32 Tdec_ms)

Purpose: To change the motion parameters on the fly.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A
Position	i32	new target position (-134,217,728 ≦ Position ≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 VH,VL: pps (0 ≦VH,VL ≦ 6553500) Tacc_ms,Tdec_ms: acc/dec time in milliseconds.
VH	i32	
Tacc_ms	u32	
Tdec_ms	u32	

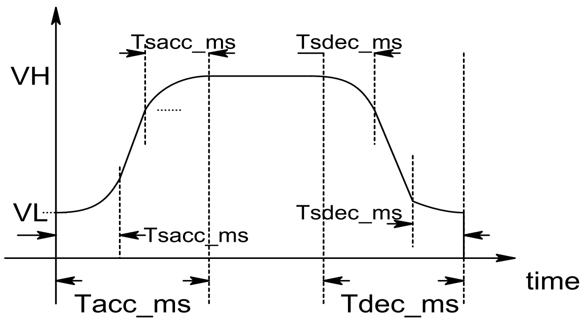
MPC3024A_S_onLINE_change

Format : u32 status = MPC3024A_S_onLINE_change(u8 CardID, u8 axis,
i32 Position, u8 posi_mode, i32 VL, i32 VH, u32 Tacc_ms,
u32 Tdec_ms, u32 TSacc_ms, u32 TSdec_ms)

Purpose: To change the motion parameters on the fly.

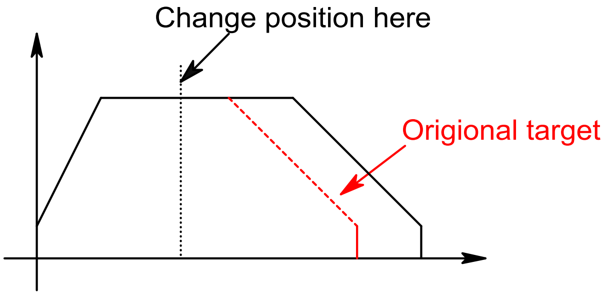
Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A
Position	i32	new target position (-134,217,728 ≦ Position ≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 VH VL Tsacc_ms Tsdec_ms Tsacc_ms Tsdec_ms Tacc_ms Tdec_ms time VH,VL : pps, (0 ≦ VH,VL ≦ 6553500) Tacc_ms,Tdec_ms: milliseconds. TSacc_ms,TSdec_ms: milliseconds, time of s curve range
VH	i32	
Tacc_ms	u32	
Tdec_ms	u32	
TSacc_ms	u32	
TSdec_ms	u32	

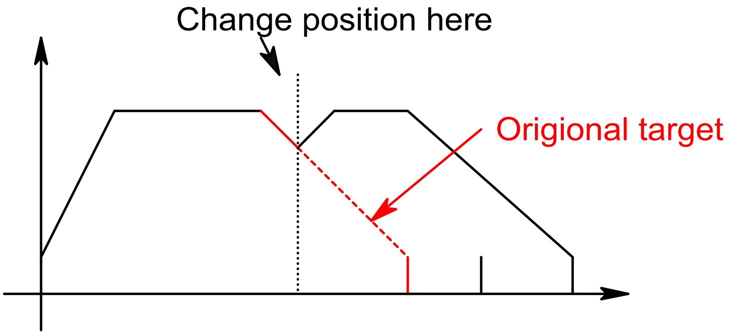
Some known conditions and suggestion

Condition1:



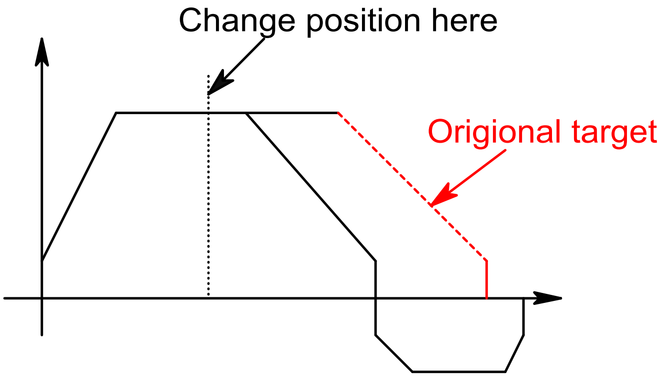
Override while motion in constant speed region.
The remained distance is smaller than the override target – original target
This is the most preferred condition.

Condition2:



Override while motion in deceleration speed region.
The remained distance is smaller than the override target – original target
Not preferred, some condition will run abnormal.

Condition3:



Override while motion in constant speed region.
The remained distance is larger than the override target – original target
Not preferred, some condition will run abnormal.

Software trigger will cause the motion continuous run, please avoid this condition.

9.9 Suppression of vibration

According to some study, the smooth positioning can be improved by adequate final pulse generation,

MPC3024A_suppress_vibration_set( ) will give less vibration at final positioning.

MPC3024A_suppress_vibration_read( ) to read back the data you set.

- **MPC3024A_suppress_vibration_set**

Format : **u32 status = MPC3024A_suppress_vibration_set(u8 CardID, u8 axis, u16 RT, u16 FT)**

Purpose: To setup vibration suppression mode.

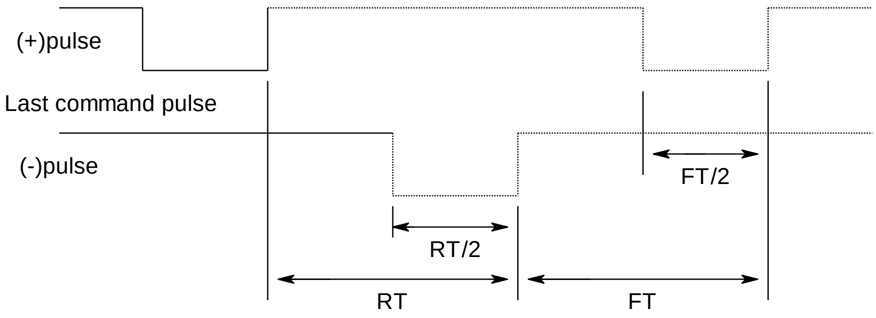
Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
RT	u16	reverse direction time, 1.6us *RT (0 ≦RT≦62500)
FT	u16	forward direction time, 1.6us *FT (0 ≦RT≦62500)

Note on vibration suppression:

The MPC3024A Card provides the function to suppress vibration at the time of stop by adding one pulse each in reverse and forward directions just after outputting all command pulses. Output timing of additional pulses is set by calling this function. The vibration suppression function is valid when the output time in reverse direction (RT) and that in forward direction (FT) are set at other that 0. Dotted lines in the figure below indicate pulses added by the vibration suppression function in the case of operation in positive direction.



- **MPC3024A_suppress_vibration_read**

Format : **u32 status = MPC3024A_suppress_vibration_read(u8 CardID, u8 axis, u16* RT, u16* FT)**

Purpose: Read back parameters of vibration suppression mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
RT	u16	reverse direction time, 1.6us *RT (0 ≦RT≦62500)
FT	u16	forward direction time, 1.6us *FT (0 ≦RT≦62500)

9.10 Position override with external trigger

If your application needs to change position from external trigger during motion period, say, packing machines need the motion control to trace the mark to position. You can configure the X axis to wait for its PCS trigger (from the mark sensor) to do a relative movement after the PCSx trigger activated. You should configure PCS(position change start) input by:

MPC3024A_PCS_PIN_set( ).

MPC3024A_PCS_PIN_read( ) for configuration read back.

After adequate configuration, you can use

MPC3024A_PCS_position_override( ) to setup the distance of overriding, now the motion srtarts (will send pulse output) but the internal control logic of positioning is suspended to wait on PCS signal, while the PCS signal is active the control logic will resume to positioning.

- **MPC3024A_PCS_PIN_set**

Format : **u32 status = MPC3024A_PCS_PIN_set(u8 CardID, u8 axis, u8 enable, u8 pcs_logic)**

Purpose: To configure the PCS pin(position change start input).

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
enable	u8	0: treat PCS PIN as a general input. 1: treat PCS PIN as a dedicated position change start input.
pcs_logic	u8	0: setting the pin connects or equals to GND level make this pin active logic. 1: setting the pin floats or equals to +24V makes this signal active logic.

Note:

1. On wiring board terminal marked as PCS
2. PCS polarity logic is very important for correct operation of position override, **it must configure as pcs_logic=0 else the PCS function may go wrong.**

- **MPC3024A_PCS_PIN_read**

Format : **u32 status = MPC3024A_PCS_PIN_read(u8 CradID, u8 axis, u8* enable, u8* pcs_logic, u8 *state)**

Purpose: Readback the configuration of the PCS pin(position change start input).

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X axis 2: Z axis	1: Y axis 3: A axis

Output:

Name	Type	Description
enable	u8	0: treat PCS PIN as a general input. 1: treat PCS PIN as a dedicated position change start input.
pcs_logic	u8	0: setting the pin connects or equals to GND level make this pin active logic. 1: setting the pin floats or equals to +24V makes this signal active logic.
state	u8	state of PCS pin

Note on PCS function:

Name	Description
PCS	PCS pin is external triggered position change function input pin. In a pulse type control system, the pulse is generated by the processor and the driver accepts the pulse train doing the motion job and feedback control. When the processor finishes the pulse generating work, do not means the servo driver finishes the positioning, the INP output of driver ensures the completeness of positioning and accuracy.

- **MPC3024A_PCS_position_override**

Format : **u32 status = MPC3024A_PCS_position_override(u8 CardID,u8 Axis,i32 distance, u8 trigger_mode);**

Purpose: To override the target position on the fly.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
distance	i32	relative distance to move (0 ≦ distance ≦ 134,217,727)
trigger_mode	u8	0: Hardware trigger (PCS pin signal turning on) 1: Software trigger (immediately override)

Note:

1. The position override is the relative distance to go of the current position on PCS active or on software override command.
2. The override is only valid while the motion is active (in motion state)
3. Even you use ***MPC3024A_PCS_position_override( )*** in software trigger mode, **you must configure PCS input as dedicated for correct software trigger function.**
4. The direction of initial position to target position must be the same as current position to override position.

9.11 Linear interpolation motion control

Before doing any motion movement, please make sure the over-travel protection is not active. Once any of the over-travel limit (LS+ or LS-) is active, the motion will be un-available (refer ***MPC3024A_EL_mode_set*** and the polarity can be set by on card DIP switch) . Please also note the output pulse type of the driver you are using, adjust the output pulse mode to meet the driver. If the signal does not match (refer ***MPC3024A_pulse_outmode_set***), you can also have a unsuspected movement.

Once you have homed and the coordinate system has setup, the linear interpolation function now is available. For the flexible combination of motion command, the MPC provides mixed coordinate system, i.e. any axis may be work in relative or absolute coordinate system.

MPC3024A_T_LINE2_move( ) for any two axes linear interpolation at trapezoidal profile.

MPC3024A_S_LINE2_move( ) for any two axes linear interpolation at S curve profile.

For any 3 axes use:

MPC3024A_T_LINE3_move( ) or

MPC3024A_S_LINE3_move( )

If the total 4 axes in linear interpolation mode, use:

MPC3024A_T_LINE4_move( ) or

MPC3024A_S_LINE4_move( )

For unified motion command

MPC3024A_T_LINE_move( ) will command the motion from 1 to 4 axes in T profile.

MPC3024A_S_LINE_move( ) will command the motion from 1 to 4 axes in S profile.

Use ***MPC3024A_stop( )*** (refer MPC3024A_stop) to stop motion on any or all axes immediately or decelerate to stop.

Note:

Conditional start of motion control is applicable to all the commands mentioned above. (refer 10.7 Conditional start of motion control) .

Note on linear interpolation with continuous mode:

1. Linear interpolation motion control in continuous mode (MPC3024A_continuous_flag_set ( ), conti_flag=1), be sure to check continuous buffer (MPC3024A_continuous_buffer_no_read( )) until ‘remain_no’ not equal to 2,else the command will be defective.
2. In non-continuous mode(MPC3024A_continuous_flag_set( )， conti_flag=0), be sure to check (MPC3024A_motion_status_read( ); check_factor=0，ret_flag =1) to confirm the motion axes are ready.
3. The remained axes maybe programmed as point to point, linear or circular interpolation mode.

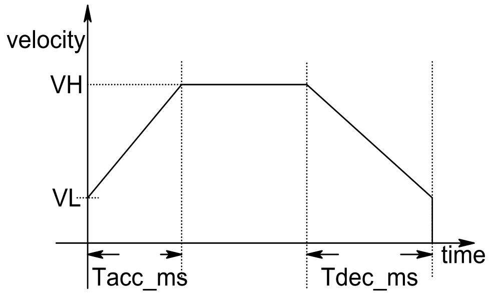
● MPC3024A_T_LINE2_move

Format : u32 status = MPC3024A_T_LINE2_move(u8 CardID, u8 line2_index,
i32 Position1, i32 Position2, u8 posi_mode[4], i32 VL, i32 VH,
u32 Tacc_ms, u32 Tdec_ms, u8 wait)

Purpose: To take linear interpolation movement with trapezoidal profile.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
line2_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
Position1	i32	target position (absolute or relative) for the first axis (-134,217,728 ≦ Position1 ≦134,217,727) for example: line2_index=2, the first axis is X
Position2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦ Position2 ≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute only valid on the corresponding axis of line2_index
VL	i32	 VH,VL:pps (0 ≦VH,VL ≦ 6553500), composite speed of motion axes. Tacc_ms,Tdec_ms: acc/dec time in milliseconds.
VH	i32	
Tacc_ms	u32	
Tdec_ms	u32	
wait	u8	0: immediately act 1: wait for CSTA 2: wait for compare start

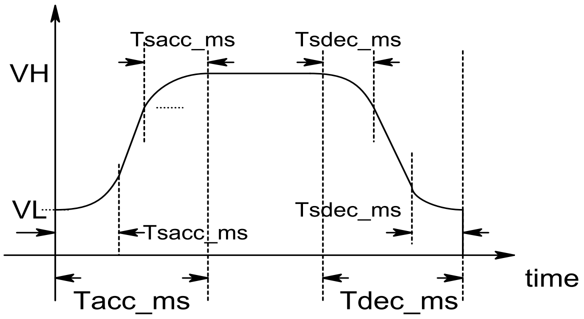
- **MPC3024A_S_LINE2_move**

Format : **u32 status = MPC3024A_S_LINE2_move(u8 CardID, u8 line2_index,**
i32 Position1, i32 Position2, u8 posi_mode[4], i32 VL, i32 VH,
u32 Tacc_ms, u32 Tdec_ms, u32 TSacc_ms, u32 TSdec_ms, u8 wait)

Purpose: To take linear interpolation movement with S curve profile.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
line2_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
Position1	i32	target position (absolute or relative) for the first axis (-134,217,728 ≦ Position1 ≦134,217,727) for example: line2_index=2, the first axis is X
Position2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦ Position2 ≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute only valid on the corresponding axis of line2_index
VL	i32	 Tsacc_ms Tsdec_ms VH VL Tsdec_ms Tsacc_ms time Tacc_ms Tdec_ms VH,VL : pps, (0 ≦ VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms,Tdec_ms: milliseconds. TSacc_ms,TSdec_ms: milliseconds, time of s curve range
VH	i32	
Tacc_ms	u32	
Tdec_ms	u32	
TSacc_ms	u32	
TSdec_ms	u32	
wait	u8	0: immediately act 1: wait for CSTA 2: wait for compare start

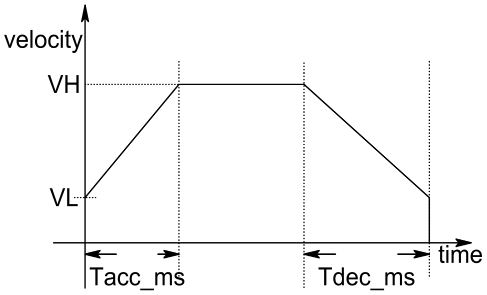
MPC3024A_T_LINE3_move

Format : **u32 status = MPC3024A_T_LINE3_move(u8 CardID, u8 line3_index,**
i32 Position1, i32 Position2, i32 Position3, u8 posi_mode[4], i32 VL,
i32 VH, u32 Tacc_ms, u32 Tdec_ms, u8 wait)

Purpose: To take linear interpolation movement with trapezoidal profile.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
line3_index	u8	0: X,Y, Z 1: X, Y, A 2: X, Z, A 3: Y, Z, A
Position1	i32	target position (absolute or relative) for the first axis (-134,217,728 ≦ Position1 ≦134,217,727) for example: line3_index=2, the first axis is X
Position2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦ Position2 ≦134,217,727) for example: line3_index=2, the second axis is Z
Position3	i32	target position (absolute or relative) for the third axis (-134,217,728 ≦ Position3 ≦134,217,727) for example: line3_index=2, the third axis is A
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute only valid on the corresponding axis of line3_index
VL	i32	 VH,VL: pps (0 ≦VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms,Tdec_ms: acc/dec time in milliseconds.
VH	i32	
Tacc_ms	u32	
Tdec_ms	u32	
wait	u8	0: immediately act 1: wait for CSTA 2: wait for compare start

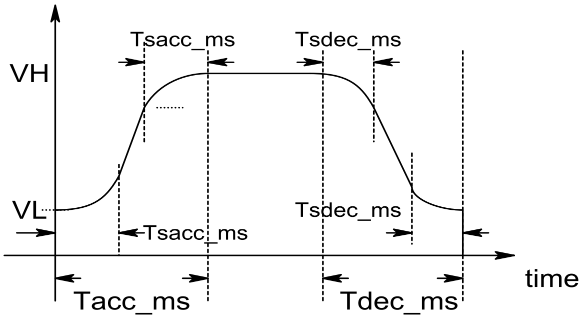
MPC3024A_S_LINE3_move

Format : **u32 status = MPC3024A_S_LINE3_move(u8 CardID, u8 line3_index, i32 Position1, i32 Position2, i32 Position3, u8 posi_mode[4], i32 VL, i32 VH, u32 Tacc_ms, u32 Tdec_ms, u32 TSacc_ms, u32 TSdec_ms, u8 wait)**

Purpose: To take linear interpolation movement with S curve profile.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
line3_index	u8	0: X,Y, Z 1: X, Y, A 2: X, Z, A 3: Y, Z, A
Position1	i32	target position (absolute or relative) for the first axis (-134,217,728 ≦ Position1 ≦134,217,727) for example: line3_index=2, the first axis is X
Position2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦ Position2 ≦134,217,727) for example: line3_index=2, the second axis is Z
Position3	i32	target position (absolute or relative) for the third axis (-134,217,728 ≦ Position3 ≦134,217,727) for example: line3_index=2, the third axis is A
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute only valid on the corresponding axis of line3_index
VL	i32	 Tsacc_ms Tsdec_ms VH VL Tsdec_ms Tsacc_ms time Tacc_ms Tdec_ms VH,VL : pps, (0 ≦ VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms,Tdec_ms: milliseconds. TSacc_ms,TSdec_ms: milliseconds, time of s curve range
VH	i32	
Tacc_ms	u32	
Tdec_ms	u32	
TSacc_ms	u32	
TSdec_ms	u32	
wait	u8	0: immediately act 1: wait for CSTA 2: wait for compare start

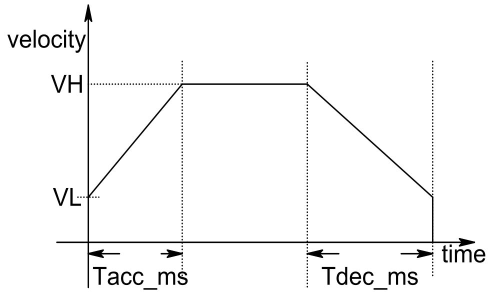
- **MPC3024A_T_LINE4_move**

Format : **u32 status = MPC3024A_T_LINE4_move(u8 CardID, i32 Position1, i32 Position2, i32 Position3, i32 Position4, u8 posi_mode[4], i32 VL, i32 VH, u32 Tacc_ms, u32 Tdec_ms, u8 wait)**

Purpose: To take linear interpolation movement with trapezoidal profile.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
Position1	i32	target position (absolute or relative) for X axis (-134,217,728 ≦ Position1 ≦134,217,727)
Position2	i32	target position (absolute or relative) for Y axis (-134,217,728 ≦ Position2 ≦134,217,727)
Position3	i32	target position (absolute or relative) for Z axis (-134,217,728 ≦ Position3 ≦134,217,727)
Position4	i32	target position (absolute or relative) for A axis (-134,217,728 ≦ Position4 ≦134,217,727)
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute
VL	i32	 VH,VL: pps (0 ≦VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms,Tdec_ms: acc/dec time in milliseconds.
VH	i32	
Tacc_ms	u32	
Tdec_ms	u32	
wait	u8	0: immediately act 1: wait for CSTA 2: **not available**

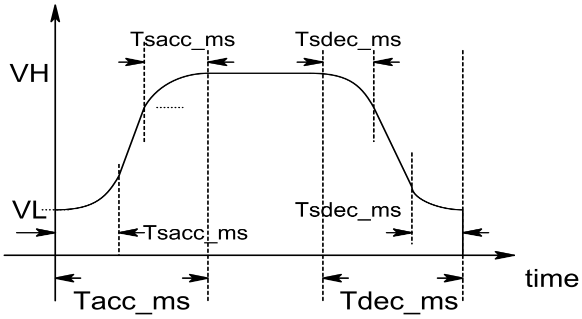
- **MPC3024A_S_LINE4_move**

Format : **u32 status = MPC3024A_S_LINE4_move(u8 CardID, i32 Position1, i32 Position2, i32 Position3, i32 Position4, u8 posi_mode[4], i32 VL, i32 VH, u32 Tacc_ms, u32 Tdec_ms, u32 TSacc_ms, u32 TSdec_ms, u8 wait)**

Purpose: To take linear interpolation movement with S curve profile.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
Position1	i32	target position (absolute or relative) for X axis (-134,217,728 ≦ Position1 ≦134,217,727)
Position2	i32	target position (absolute or relative) for Y axis (-134,217,728 ≦ Position2 ≦134,217,727)
Position3	i32	target position (absolute or relative) for Z axis (-134,217,728 ≦ Position3 ≦134,217,727)
Position4	i32	target position (absolute or relative) for A axis (-134,217,728 ≦ Position4 ≦134,217,727)
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute
VL	i32	 Tsacc_ms Tsdec_ms VH VL Tsacc_ms Tsdec_ms time Tacc_ms Tdec_ms VH,VL : pps, (0 ≦ VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms,Tdec_ms: milliseconds. TSacc_ms,TSdec_ms: milliseconds, time of s curve range
VH	i32	
Tacc_ms	u32	
Tdec_ms	u32	
TSacc_ms	u32	
TSdec_ms	u32	
wait	u8	0: immediately act 1: wait for CSTA 2: **not available**

- **MPC3024A_T_LINE_move**

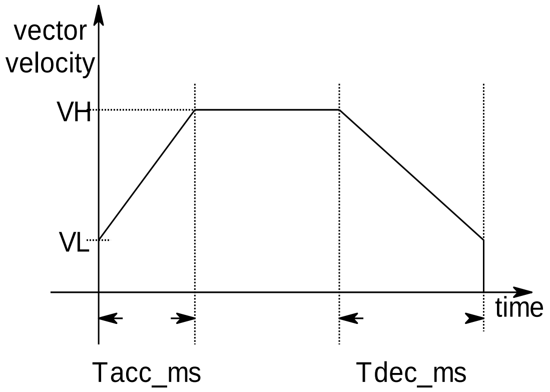
Format : **u32 status = MPC3024A_T_LINE_move (u8 CardID,**
_Tline_CMD_Type *pTLine_command,u8 wait)

Purpose: To take linear interpolation on 1~4 axes with T curve profile.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

pTLine_command	_Tline_CMD_Type	A structure pointer of motion control parameters
		struct _Tline_CMD_Type{
		u8 posi_mode[4];
		//posi_mode[0]: X axis, 0: relative, 1:absolute
		//posi_mode[1]: Y axis, 0: relative, 1:absolute
		//posi_mode[2]: Z axis, 0: relative, 1:absolute
		//posi_mode[3]: A axis, 0: relative, 1:absolute
		i32 VL;
		i32 VH;
		i32 Tacc_ms;
		i32 Tdec_ms;
		
		i32 Position[4];
		// position (absolute or relative) range
		// -134,217,728 ≦Position≦
		// 134,217,727
		// Position[0]: X Axis, position (absolute
		//or relative) range
		//Position[1]:Y Axis, position (absolute
		//or relative) range
		//Position[2]:Z Axis, position (absolute
		//or relative) range
		//Position[3]:A Axis, position (absolute
		//or relative) range
		
		u8 Axis;
		//bit0:X axis bit 1:Y axis
		//bit 2:Z axis bit 3:A axis
		} Tline_CMD_Type;
		

		VH,VL:pps, start speed (0 ≦ VL ≦ 6553500) Tacc_ms, Tdec_ms: in mini-seconds.
wait	u8	0: immediately act 1: wait for CSTA 2: wait for compare start (the compare counter source must exclude from the waiting axis)

MPC3024A_S_LINE_move

Format : **u32 status = MPC3024A_S_LINE_move (u8 CardID, _Sline_CMD_Type *pSLine_command,u8 wait)**

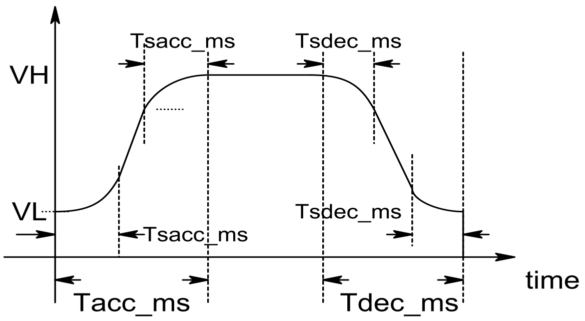
Purpose: To take linear interpolation on 1~4 axes with S curve profile.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

pSLine_command	_Sline_CMD_Type	A structure pointer of motion control parameters
		struct _Sline_CMD_Type{
		u8 posi_mode[4];
		//posi_mode[0]: X axis, 0: relative, 1:absolute
		//posi_mode[1]: Y axis, 0: relative, 1:absolute
		//posi_mode[2]: Z axis, 0: relative, 1:absolute
		//posi_mode[3]: A axis, 0: relative, 1:absolute
		
		i32 VL;
		i32 VH;
		i32 Tacc_ms;
		i32 Tdec_ms;
		u32 TSacc_ms;
		u32 TSdec_ms;
		
		i32 Position[4];
		// position (absolute or relative) range
		// -134,217,728 ≦Position≦ 134,217,727
		
		// Position[0]: X Axis, position (absolute
		//or relative) range
		//Position[1]:Y Axis, position (absolute
		//or relative) range
		//Position[2]:Z Axis, position (absolute
		//or relative) range
		//Position[3]:A Axis, position (absolute
		//or relative) range
		
		u8 Axis;
		// any bit of the following bit reads “1” means if
		//the corresponding axis is active.
		// bit 0:X axis
		// bit 1:Y axis
		// bit 2:Z axis
		// bit 3:A axis
		} Sline_CMD_Type;

		 Tsacc_ms Tsdec_ms VH VL Tsdec_ms Tsacc_ms time Tacc_ms Tdec_ms VH,VL : pps, (0 ≦ VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms,Tdec_ms: milliseconds. TSacc_ms,TSdec_ms: milliseconds, time of s curve range
wait	u8	0: immediately act 1: wait for CSTA 2: wait for compare start (the compare counter source must exclude from the waiting axis)

9.12 Circular interpolation

Before doing any motion movement, please make sure the over-travel protection is not active. Once any of the over-travel limit (LS+ or LS-) is active, the motion will be un-available (refer ***MPC3024A_EL_mode_set*** and the polarity can be set by on card DIP switch) . Please also note the output pulse type of the driver you are using, adjust the output pulse mode to meet the driver. If the signal does not match (refer ***MPC3024A_pulse_outmode_set***), you can also have a unsuspected movement.

Once you have homed and the coordinate system has setup, the circular interpolation function now is available. We can do circular interpolation on any two of the 4 axes, the MPC card provides several kinds of parameter setting to do circular interpolation.

Arc and center define a circle

Use current position as default start point, the circle center point and circle end position as parameters to define the motion. Use

MPC3024A_T_ARC_center_move( ) for T-profile and

MPC3024A_S_ARC_center_move( ) for S-profile.

3 points define a circle

Use current position as default start point, one middle point as passing through point and the end point to define the circle. For a full circle, use:

MPC3024A_T_CIR_3P_move( ) for T-profile and

MPC3024A_S_CIR_3P_move( ) for S-profile.

If only arc use:

MPC3024A_T_ARC_3P_move( ) for T-profile and

MPC3024A_S_ARC_3P_move( ) for S-profile.

Circle center point and radius define a circle

Use current position as default and end point and the radius to define a circle. For a full circle, use:

MPC3024A_T_CIR_Radius_move( ) for T-profile and

MPC3024A_S_CIR_Radius_move( ) for S-profile will do.

If only arc, use:

MPC3024A_T_ARC_Radius_move( ) for T-profile and

MPC3024A_S_ARC_Radius_move( ) for S-profile.

For the above functions, if you only need the circular interpolation work in a constant velocity mode, you can program the VH=VL and Tacc_dec_ms=0.

Use ***MPC3024A_stop( )*** (refer
MPC3024A_stop) to stop motion on any or all axes immediately or decelerate to stop.

Conditional start of motion control are applicable to all the commands mentioned above (refer 9.7 Conditional start of motion control).

Note on circular interpolation with continuous mode:

1. Circular interpolation motion control in continuous mode (MPC3024A_continuous_flag_set ( ), conti_flag=1), be sure to check continuous buffer (MPC3024A_continuous_buffer_no_read( )) until ‘remain_no’ not equal to 2,else the command will be defective.
2. In continuous mode, be sure to set maximum speed first (MPC3024A_velocity_range_fix( )) at the operation axes.
3. In non-continuous mode(MPC3024A_continuous_flag_set( )， conti_flag=0), be sure to check (MPC3024A_motion_status_read( ); check_factor=0，ret_flag =1) to confirm the motion axes are ready.
4. While any 2 axes are working in circular interpolation mode, the others can not work in circular interpolation too, but point to point or linear interpolation is permitted.
5. The function MPC3024A_ARC_3P_move( ) does not need to define the motion direction, since the trajectory point has hidden definition.

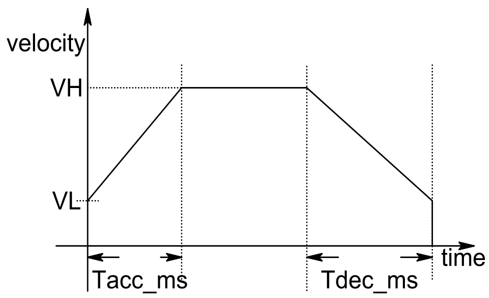
MPC3024A_T_ARC_center_move

Format : **u32 status = MPC3024A_T_ARC_center_move(u8 CardID, u8 arc2_index, i32 center1, i32 center2, i32 endp1, i32 endp2, u8 posi_mode[4], i32 VL,i32 VH, u32 Tacc_dec_ms, u8 direction, u8 wait)**

Purpose: To take circular interpolation movement with circle center and end position and the acceleration/deceleration is T profile for arc trajectrory.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X, Y 1: X, Z 2: X, A 3: Y, Z 4: Y, A 5: Z, A
center1	i32	circle center position (absolute or relative) for the first axis (-134,217,728 ≦ center1 ≦134,217,727) for example: line2_index=2, the first axis is X
center 2	i32	circle center position (absolute or relative) for the second axis (-134,217,728 ≦ center 2 ≦134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦ endp1 ≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦ endp2 ≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute only valid on the corresponding axis of arc2_index
VL	i32	 VH,VL: pps (0 ≦VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms = Tdec_ms = Tacc_dec_ms: acc/dec time in milliseconds.
VH	i32	
Tacc_dec_ms	u32	
direction	u8	0: CW direction 1: CCW direction
wait	u8	0: immediately act 1: wait for CSTA 2: wait for compare start

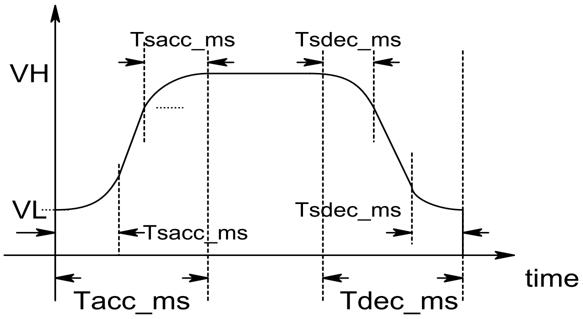
- **MPC3024A_S_ARC_center_move**

Format : **u32 status = MPC3024A_S_ARC_center_move(u8 CardID, u8 arc2_index, i32 center1, i32 center2, i32 endp1, i32 endp2, u8 posi_mode[4], i32 VL,i32 VH, u32 Tacc_dec_ms, u32 TSacc_dec_ms, u8 direction, u8 wait)**

Purpose: To take circular interpolation movement with circle center and end position and the acceleration/deceleration is S profile for arc trajectory.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X, Y 1: X, Z 2: X, A 3: Y, Z 4: Y, A 5: Z, A
center1	i32	circle center position (absolute or relative) for the first axis (-134,217,728 ≦ center1 ≦134,217,727) for example: line2_index=2, the first axis is X
center 2	i32	circle center position (absolute or relative) for the second axis (-134,217,728 ≦ center2 ≦134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦ endp1 ≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦ endp2 ≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute only valid on the corresponding axis of arc2_index
VL	i32	 VH VL Tsacc_ms Tsdec_ms Tsacc_ms Tsdec_ms Tacc_ms Tdec_ms time VH,VL : pps, (0 ≦ VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms,Tdec_ms: milliseconds. TSacc_ms = TSdec_ms = TSacc_dec_ms: milliseconds, time of s curve range
VH	i32	
Tacc_dec_ms	u32	
TSacc_dec_ms	u32	
direction	u8	0: CW direction 1: CCW direction
wait	u8	0: immediately act 1: wait for CSTA 2: wait for compare start

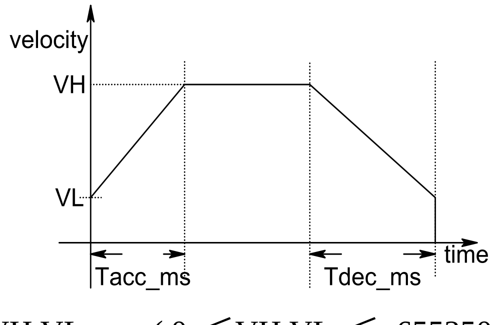
- **MPC3024A_T_CIR_3P_move**

Format : **u32 status = MPC3024A_T_CIR_3P_move(u8 CardID, u8 arc2_index,**
i32 middle1, i32 middle2, i32 endp1, i32 endp2, u8 posi_mode[4],
i32 VL, i32 VH, u32 Tacc_dec_ms, u8 wait)

Purpose: To take the current position and the middle, end position to make a circle and the circular interpolation pass through the 3 positions.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X, Y 1: X, Z 2: X, A 3: Y, Z 4: Y, A 5: Z, A
middle1	i32	middle position (absolute or relative) for the first axis (-134,217,728 ≦ middle1 ≦134,217,727) for example: line2_index=2, the first axis is X
middle 2	i32	middle position (absolute or relative) for the second axis (-134,217,728 ≦ middle2 ≦134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦ endp1 ≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦ endp2 ≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute only valid on the corresponding axis of arc2_index
VL	i32	 VH,VL: pps (0 ≦VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms = Tdec_ms = Tacc_dec_ms: acc/dec time in milliseconds.
VH	i32	
Tacc_dec_ms	u32	
wait	u8	0: immediately act 1: wait for CSTA 2: wait for compare start

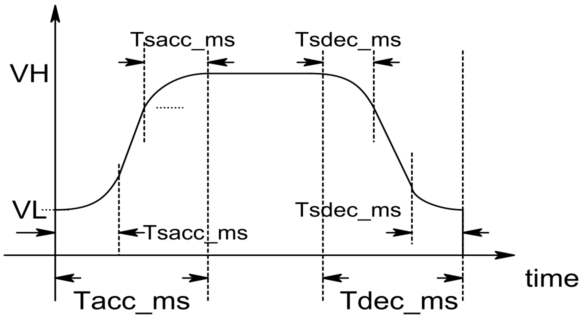
● MPC3024A_S_CIR_3P_move

Format : **u32 status = MPC3024A_S_CIR_3P_move(u8 CardID, u8 arc2_index, i32 middle1, i32 middle2, i32 endp1, i32 endp2, u8 posi_mode[4], i32 VL, i32 VH, u32 Tacc_dec_ms, u32 TSacc_dec_ms, u8 wait)**

Purpose: To take the current position and the middle, end position to make a circle and the circular interpolation pass through the 3 positions.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X, Y 1: X, Z 2: X, A 3: Y, Z 4: Y, A 5: Z, A
middle1	i32	middle position (absolute or relative) for the first axis (-134,217,728 ≦ middle1 ≦134,217,727) for example: line2_index=2, the first axis is X
middle 2	i32	middle position (absolute or relative) for the second axis (-134,217,728 ≦ middle2 ≦134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦ endp1 ≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦ endp2 ≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute only valid on the corresponding axis of arc2_index
VL	i32	 VH,VL : pps, (0 ≦ VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms,Tdec_ms: milliseconds. TSacc_ms = TSdec_ms = TSacc_dec_ms: milliseconds, time of s curve range
VH	i32	
Tacc_dec_ms	u32	
TSacc_dec_ms	u32	
wait	u8	0: immediately act 1: wait for CSTA 2: wait for compare start

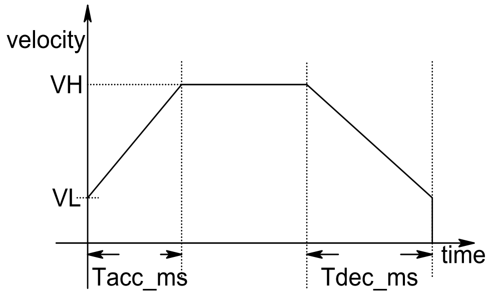
- **MPC3024A_T_ARC_3P_move**

Format : **u32 status = MPC3024A_T_ARC_3P_move(u8 CardID, u8 arc2_index,**
i32 middle1, i32 middle2, i32 endp1, i32 endp2, u8 posi_mode[4],
i32 VL, i32 VH, u32 Tacc_dec_ms, u8 wait)

Purpose: To take circular interpolation movement with current point and the other 2 points and the acceleration/deceleration is T profile for arc trajectrory.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
middle1	i32	middle position (absolute or relative) for the first axis (-134,217,728 ≦ middle1 ≦134,217,727) for example: line2_index=2, the first axis is X
middle2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦ middle2 ≦134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦ endp1 ≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦ endp2 ≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute only valid on the corresponding axis of arc2_index
VL	i32	 VH,VL: pps (0 ≦VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms = Tdec_ms = Tacc_dec_ms: acc/dec time in milliseconds.
VH	i32	
Tacc_dec_ms	u32	
wait	u8	0: immediately act 1: wait for CSTA 2: wait for compare start

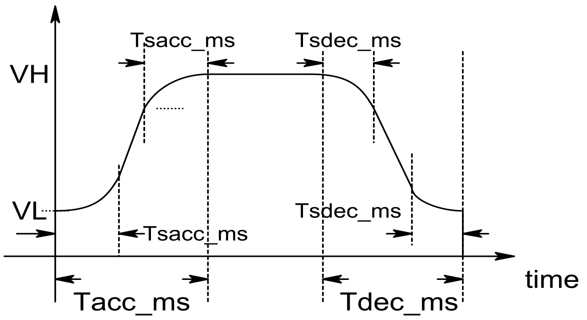
- **MPC3024A_S_ARC_3P_move**

Format : **u32 status = MPC3024A_S_ARC_3P_move(u8 CardID, u8 arc2_index,**
i32 middle1, i32 middle2, i32 endp1, i32 endp2, u8 posi_mode[4],
i32 VL, i32 VH, u32 Tacc_dec_ms, u32 TSacc_dec_ms, u8 wait)

Purpose: To take circular interpolation movement with current point and the other 2 points as the circle trajectory and the acceleration/deceleration is S profile.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
middle1	i32	middle position (absolute or relative) for the first axis (-134,217,728 ≦ middle1 ≦134,217,727) for example: line2_index=2, the first axis is X
middle2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦ middle2 ≦134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦ endp1 ≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦ endp2 ≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute only valid on the corresponding axis of arc2_index
VL	i32	 Tsacc_ms Tsdec_ms VH VL Tsdec_ms Tsacc_ms Tacc_ms Tdec_ms time VH,VL : pps, (0 ≦ VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms,Tdec_ms: milliseconds. TSacc_ms = TSdec_ms = TSacc_dec_ms: milliseconds, time of s curve range
VH	i32	
Tacc_dec_ms	u32	
TSacc_dec_ms	u32	
wait	u8	0: immediately act 1: wait for CSTA 2: wait for compare start

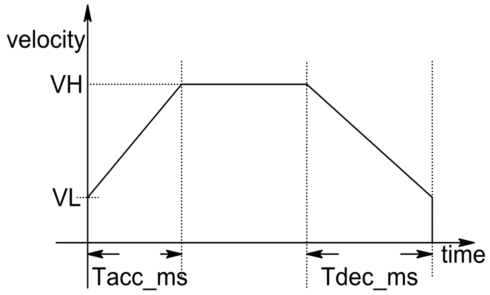
- **MPC3024A_T_CIR_Radius_move**

Format : **u32 status = MPC3024A_T_CIR_Radius_move(u8 CardID, u8 arc2_index, i32 radius, i32 endp1, i32 endp2, u8 posi_mode[4], i32 VL, i32 VH, u32 Tacc_dec_ms, u8 direction, u8 wait)**

Purpose: To take the current position and the middle, end position to make a circle and the circular interpolation pass through the 3 positions.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X, Y 1: X, Z 2: X, A 3: Y, Z 4: Y, A 5: Z, A
radius	i32	Radius of the target circle
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦ endp1 ≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦ endp2 ≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute only valid on the corresponding axis of arc2_index
VL	i32	 VH,VL: pps (0 ≦VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms = Tdec_ms = Tacc_dec_ms: acc/dec time in milliseconds.
VH	i32	
Tacc_dec_ms	u32	
direction	u8	0: CW 1: CCW refer: Note on direction paramter
wait	u8	0: immediately act 1: wait for CSTA 2: wait for compare start

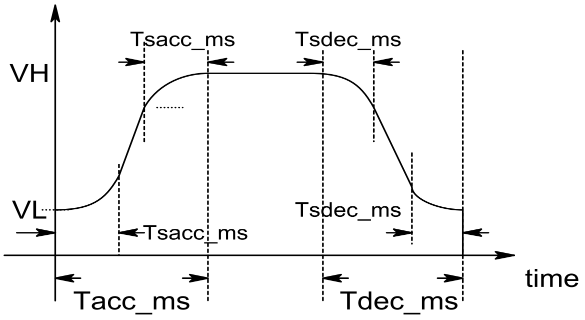
MPC3024A_S_CIR_Radius_move

Format : **u32 status = MPC3024A_S_CIR_Radius_move(u8 CardID, u8 arc2_index, i32 radius, i32 endp1, i32 endp2, u8 posi_mode[4], i32 VL, i32 VH, u32 Tacc_dec_ms, u32 TSacc_dec_ms, u8 direction, u8 wait)**

Purpose: To take the current position and the middle, end position to make a circle and the circular interpolation pass through the 3 positions.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X, Y 1: X, Z 2: X, A 3: Y, Z 4: Y, A 5: Z, A
radius	i32	Radius of the target circle
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦ endp1 ≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦ endp2 ≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute only valid on the corresponding axis of arc2_index
VL	i32	 Tsacc_ms Tsdec_ms VH VL Tsdec_ms Tsacc_ms time Tacc_ms Tdec_ms VH,VL : pps, (0 ≦ VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms,Tdec_ms: milliseconds. TSacc_ms = TSdec_ms = TSacc_dec_ms: milliseconds, time of s curve range
VH	i32	
Tacc_dec_ms	u32	
TSacc_dec_ms	u32	
direction	u8	0: CW 1: CCW refer: Note on direction paramter
wait	u8	0: immediately act 1: wait for CSTA 2: wait for compare start

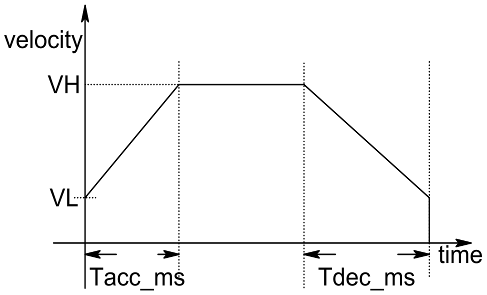
● MPC3024A_T_ARC_Radius_move

Format : **u32 status = MPC3024A_T_ARC_Radius_move(u8 CardID, u8 arc2_index,**
i32 radius, i32 endp1, i32 endp2, u8 posi_mode[4], i32 VL, i32 VH,
u32 Tacc_dec_ms, u8 direction, u8 wait)

Purpose: To take the current position and end position to make an arc at designated R**.**

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X, Y 1: X, Z 2: X, A 3: Y, Z 4: Y, A 5: Z, A
radius	i32	radius for the circle to pass currenet position and endpoint
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦ endp1 ≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦ endp2 ≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute only valid on the corresponding axis of arc2_index
VL	i32	 VH,VL: pps (0 ≦VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms = Tdec_ms = Tacc_dec_ms: acc/dec time in milliseconds.
VH	i32	
Tacc_dec_ms	u32	
direction	u8	0: CW 1: CCW refer: Note on direction paramter
wait	u8	0: immediately act 1: wait for CSTA 2: wait for compare start

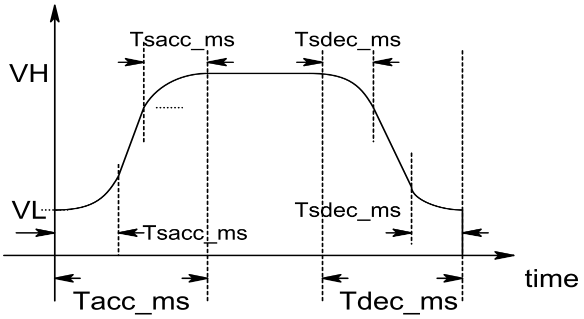
MPC3024A_S_ARC_Radius_move

Format : u32 status = MPC3024A_S_ARC_Radius_move(u8 CardID, u8 arc2_index, i32 radius, i32 endp1, i32 endp2, u8 posi_mode[4], i32 VL, i32 VH, u32 Tacc_dec_ms, u32 TSacc_dec_ms, u8 direction, u8 wait)

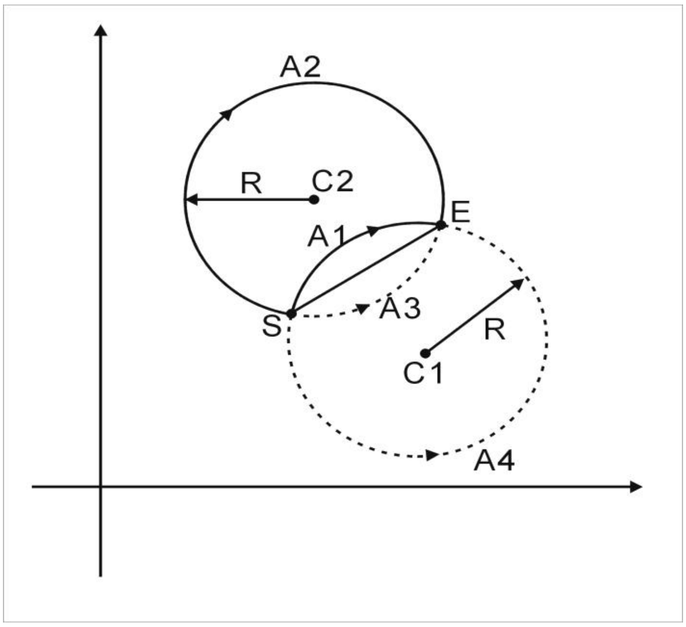
Purpose: To take the current position and end position to make an arc at designated R**.**

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X, Y 1: X, Z 2: X, A 3: Y, Z 4: Y, A 5: Z, A
radius	i32	radius for the circle to pass currenet position and endpoint
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦ endp1 ≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦ endp2 ≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute only valid on the corresponding axis of arc2_index
VL	i32	 Tsacc_ms Tsdec_ms VH VL Tsdec_ms Tsacc_ms time Tacc_ms Tdec_ms VH,VL : pps, (0 ≦ VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms,Tdec_ms: milliseconds. TSacc_ms = TSdec_ms = TSacc_dec_ms: milliseconds, time of s curve range
VH	i32	
Tacc_dec_ms	u32	
TSacc_dec_ms	u32	
direction	u8	0: CW 1: CCW refer: Note on direction paramter
wait	u8	0: immediately act 1: wait for CSTA 2: wait for compare start

Note on direction of direction parameter:



For example:
S: start point (current position)
E: end point
R: radius
Say the circle will go CW direction,
if R>0 then locus A1 will be;
if R<0 then A2 will be.

Say the circle will go CCW direction
if R>0 then locus A3 will be;
if R<0 then A4 will be.

9.13 Spiral Motion

If 2 axes doing circular interpolation with another axis doing linear interpolation synchronized, it is spiral motion. The MPC card can only provide spiral motion for X,Y doing circular interpolation and Z axis doing linear interpolation. In spiral interpolation mode the extra A axis is not available for control, it is used as auxiliary axis of spiral interpolation. Be sure not to connect A axis to drive any driver if you use spiral function in your application else you will get an un-predictable pulse output.

With different kind of circular interpolation conditions, we can have different kinds of spiral motion:

MPC3024A_T_ArcXY_LineZ_center_move( ), T acc/dec profile, spiral motion with X, Y run an arc.

MPC3024A_S_ArcXY_LineZ_center_move( ), S acc/dec profile, spiral motion with X, Y run an arc.

All just the same as circular interpolation, the arc or circle defined by different kinds of parameters, the spiral motion has different command:

Use 3 points to define the circle of spiral motion:

MPC3024A_T_CirXY_LineZ_3P_move( )

MPC3024A_S_CirXY_LineZ_3P_move( )

MPC3024A_T_ArcXY_LineZ_3P_move( )

MPC3024A_S_ArcXY_LineZ_3P_move( )

Use radius to define the circle of spiral motion:

MPC3024A_T_CirXY_LineZ_Radius_move( )

MPC3024A_S_CirXY_LineZ_Radius_move( )

MPC3024A_T_ArcXY_LineZ_Radius_move( )

MPC3024A_S_ArcXY_LineZ_Radius_move( )

Use ***MPC3024A_stop( )*** (refer MPC3024A_stop) to stop motion on any or all axes immediately or decelerate to stop.

Conditional start of motion control are applicable to all the commands mentioned above (refer 9.7 Conditional start of motion control).

Note on spiral interpolation with continuous mode:

1. Spiral interpolation motion control in continuous mode (MPC3024A_continuous_flag_set ( ), conti_flag=1), be sure to check continuous buffer (MPC3024A_continuous_buffer_no_read( )) until ‘remain_no’ not equal to 2,else the command will be defective.
2. In continuous mode, be sure to set maximum speed first (MPC3024A_velocity_range_fix( )) at the operation axes.
3. In non-continuous mode(MPC3024A_continuous_flag_set( ) ， conti_flag=0), be sure to check (MPC3024A_motion_status_read( ); check_factor=0，ret_flag =1) to confirm the motion axes are ready.

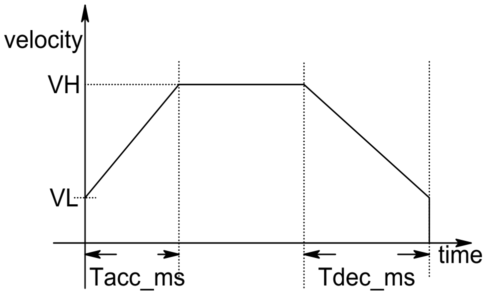
MPC3024A_T_ArcXY_LineZ_center_move

Format : **u32 status = MPC3024A_T_ArcXY_LineZ_center_move(u8 CardID, i32 centerX,**
i32 centerY, i32 endpX, i32 endpY, i32 endpZ, u8 posi_mode[4],
i32 VL, i32 VH, u32 Tacc_dec_ms, u8 direction, u8 wait)

Purpose: X,Y axes doing arc interpolation as designated parameters and Z axis doing linear interpolation synchronously.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
centerX	i32	X axis center of arc
centerY	i32	Y axis center of arc
endpX	i32	end position (absolute or relative) for the X axis (-134,217,728 ≦ endpX ≦134,217,727)
endpY	i32	end position (absolute or relative) for the Y axis (-134,217,728 ≦ endpY ≦134,217,727)
endpZ	i32	end position (absolute or relative) for the Z axis (-134,217,728 ≦ endpZ ≦134,217,727) Linear interpolation will go from current Z position to endZ position
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute
VL	i32	 VH,VL: pps (0 ≦VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms = Tdec_ms = Tacc_dec_ms: acc/dec time in milliseconds.
VH	i32	
Tacc_dec_ms	u32	
direction	u8	0: CW 1: CCW
wait	u8	0: immediately act 1: wait for CSTA 2: **not available**

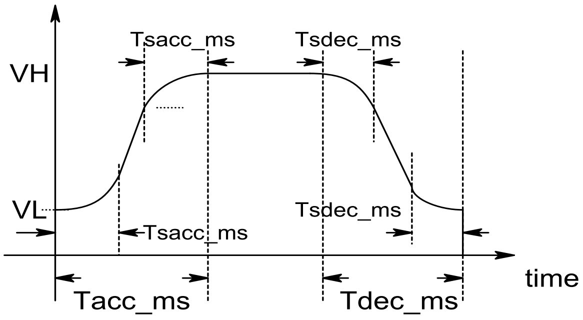
- **MPC3024A_S_ArcXY_LineZ_center_move**

Format : **u32 status = MPC3024A_S_ArcXY_LineZ_center_move(u8 CardID, i32 centerX, i32 centerY,i32 endpX, i32 endpY, i32 endpZ, u8 posi_mode[4], i32 VL, i32 VH,u32 Tacc_dec_ms, u32 TSacc_dec_ms, u8 direction, u8 wait)**

Purpose: X,Y axes doing arc interpolation as designated parameters and Z axis doing linear interpolation synchronously.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
centerX	i32	X axis center of arc
centerY	i32	Y axis center of arc
endpX	i32	end position (absolute or relative) for the X axis (-134,217,728 ≦ endpX ≦134,217,727)
endpY	i32	end position (absolute or relative) for the Y axis (-134,217,728 ≦ endpY ≦134,217,727)
endpZ	i32	end position (absolute or relative) for the Z axis (-134,217,728 ≦ endpZ ≦134,217,727) Linear interpolation will go from current Z position to endZ position
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute
VL	i32	 Tsacc_ms Tsdec_ms VH VL Tsdec_ms Tsacc_ms time Tacc_ms Tdec_ms VH,VL : pps, (0 ≦ VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms,Tdec_ms: milliseconds. TSacc_ms = TSdec_ms = TSacc_dec_ms: milliseconds, time of s curve range
VH	i32	
Tacc_dec_ms	u32	
TSacc_dec_ms	u32	
direction	u8	0: CW 1: CCW
wait	u8	0: immediately act 1: wait for CSTA 2: **not available**

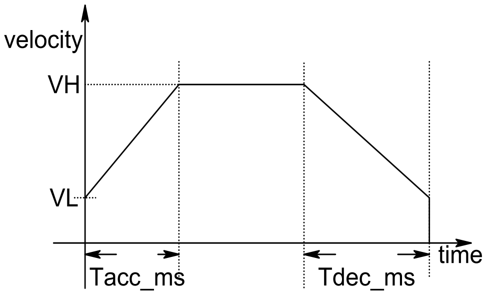
- **MPC3024A_T_CirXY_LineZ_3P_move**

Format : **u32 status = MPC3024A_T_CirXY_LineZ_3P_move(u8 CardID, i32 middleX,**
i32 middleY,i32 endpX, i32 endpY, i32 endpZ, u8 posi_mode[4],
i32 VL, i32 VH,u32 Tacc_dec_ms, u8 wait)

Purpose: X,Y axes doing circular interpolation as designated parameters and Z axis doing linear interpolation synchronously.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
middleX	i32	X axis middle point that arc will pass
middleY	i32	Y axis middle point that arc will pass
endpX	i32	end position (absolute or relative) for the X axis (-134,217,728 ≦ endpX ≦134,217,727)
endpY	i32	end position (absolute or relative) for the Y axis (-134,217,728 ≦ endpY ≦134,217,727)
endpZ	i32	end position (absolute or relative) for the Z axis (-134,217,728 ≦ endpZ ≦134,217,727) Linear interpolation will go from current Z position to endZ position
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute
VL	i32	 VH,VL: pps (0 ≦VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms = Tdec_ms = Tacc_dec_ms: acc/dec time in milliseconds.
VH	i32	
Tacc_dec_ms	u32	
wait	u8	0: immediately act 1: wait for CSTA 2: **not available**

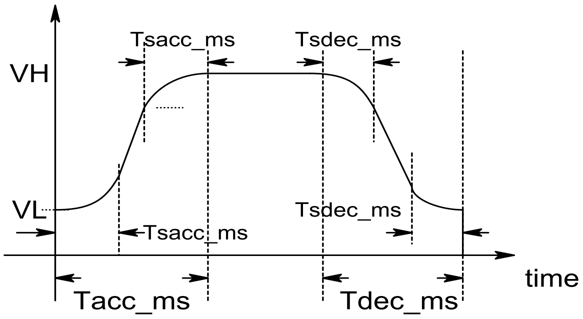
- **MPC3024A_S_CirXY_LineZ_3P_move**

Format : **u32 status = MPC3024A_S_CirXY_LineZ_3P_move(u8 CardID, i32 middleX, i32 middleY,i32 endpX, i32 endpY, i32 endpZ, u8 posi_mode[4], i32 VL, i32 VH,u32 Tacc_dec_ms, u32 TSacc_dec_ms, u8 wait)**

Purpose: X,Y axes doing circular interpolation as designated parameters and Z axis doing linear interpolation synchronously.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
middleX	i32	X axis middle point that arc will pass
middleY	i32	Y axis middle point that arc will pass
endpX	i32	end position (absolute or relative) for the X axis (-134,217,728 ≦ endpX ≦134,217,727)
endpY	i32	end position (absolute or relative) for the Y axis (-134,217,728 ≦ endpY ≦134,217,727)
endpZ	i32	end position (absolute or relative) for the Z axis (-134,217,728 ≦ endpZ ≦134,217,727) Linear interpolation will go from current Z position to endZ position
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute
VL	i32	 VH,VL : pps, (0 ≦ VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms,Tdec_ms: milliseconds. TSacc_ms = TSdec_ms = TSacc_dec_ms: milliseconds, time of s curve range
VH	i32	
Tacc_dec_ms	u32	
TSacc_dec_ms	u32	
wait	u8	0: immediately act 1: wait for CSTA 2: **not available**

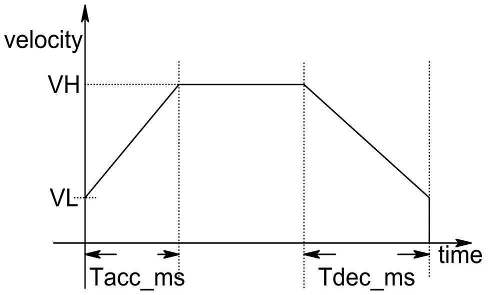
- **MPC3024A_T_ArcXY_LineZ_3P_move**

Format : **u32 status = MPC3024A_T_ArcXY_LineZ_3P_move(u8 CardID, i32 middleX, i32 middleY,i32 endpX, i32 endpY, i32 endpZ, u8 posi_mode[4], i32 VL, i32 VH,u32 Tacc_dec_ms, u8 wait)**

Purpose: X,Y axes doing arc interpolation as designated parameters and Z axis doing linear interpolation synchronously.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
middleX	i32	X axis middle point that arc will pass
middleY	i32	Y axis middle point that arc will pass
endpX	i32	end position (absolute or relative) for the X axis (-134,217,728 ≦ endpX ≦134,217,727)
endpY	i32	end position (absolute or relative) for the Y axis (-134,217,728 ≦ endpY ≦134,217,727)
endpZ	i32	end position (absolute or relative) for the Z axis (-134,217,728 ≦ endpZ ≦134,217,727) Linear interpolation will go from current Z position to endZ position
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute
VL	i32	 VH,VL: pps (0 ≦VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms = Tdec_ms = Tacc_dec_ms: milliseconds,: acc/dec time in miliseconds.
VH	i32	
Tacc_dec_ms	u32	
wait	u8	0: immediately act 1: wait for CSTA 2: **not available**

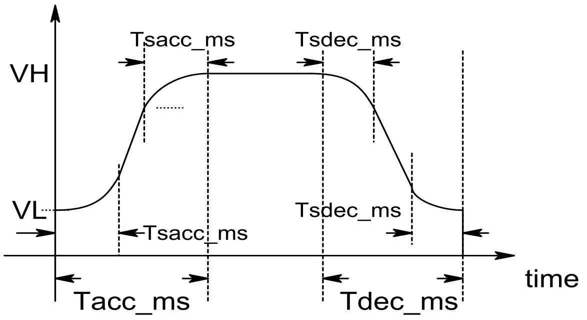
MPC3024A_S_ArcXY_LineZ_3P_move

Format : **u32 status = MPC3024A_S_ArcXY_LineZ_3P_move(u8 CardID, i32 middleX, i32 middleY,i32 endpX, i32 endpY, i32 endpZ, u8 posi_mode[4], i32 VL, i32 VH,u32 Tacc_dec_ms, u32 TSacc_dec_ms, u8 wait)**

Purpose: X,Y axes doing arc interpolation as designated parameters and Z axis doing linear interpolation synchronously.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
middleX	i32	X axis middle point that arc will pass
middleY	i32	Y axis middle point that arc will pass
endpX	i32	end position (absolute or relative) for the X axis (-134,217,728 ≦ endpX ≦134,217,727)
endpY	i32	end position (absolute or relative) for the Y axis (-134,217,728 ≦ endpY ≦134,217,727)
endpZ	i32	end position (absolute or relative) for the Z axis (-134,217,728 ≦ endpZ ≦134,217,727) Linear interpolation will go from current Z position to endpZ position
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute
VL	i32	 VH,VL : pps, (0 ≦ VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms,Tdec_ms: milliseconds. TSacc_ms = TSdec_ms = TSacc_dec_ms: milliseconds, time of s curve range
VH	i32	
Tacc_dec_ms	u32	
TSacc_dec_ms	u32	
wait	u8	0: immediately act 1: wait for CSTA 2: **not available**

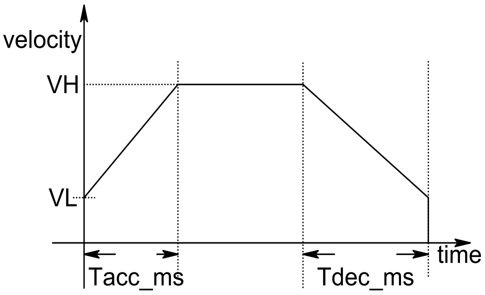
- **MPC3024A_T_CirXY_LineZ_Radius_move**

Format : **u32 status = MPC3024A_T_CirXY_LineZ_Radius_move (u8 CardID,**
i32 radius, i32 endpX, i32 endpY, i32 endpZ, u8 posi_mode[4],
i32 VL, i32 VH, u32 Tacc_dec_ms, u8 direction, u8 wait)

Purpose: X,Y axes doing circular interpolation as designated parameters and Z axis doing linear interpolation synchronously.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
radius	i32	Radius of the target circle
endpX	i32	end position (absolute or relative) for the X axis (-134,217,728 ≦ endpX ≦134,217,727)
endpY	i32	end position (absolute or relative) for the Y axis (-134,217,728 ≦ endpY ≦134,217,727)
endpZ	i32	end position (absolute or relative) for the Z axis (-134,217,728 ≦ endpZ ≦134,217,727) Linear interpolation will go from current Z position to endpZ position
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute
VL	i32	 VH,VL: pps (0 ≦VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms = Tdec_ms = Tacc_dec_ms: acc/dec time in milliseconds.
VH	i32	
Tacc_dec_ms	u32	
direction	u8	0: CW 1: CCW
wait	u8	0: immediately act 1: wait for CSTA 2: **not available**

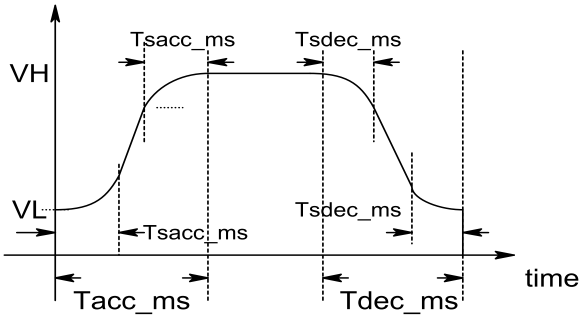
MPC3024A_S_CirXY_LineZ_Radius_move

Format : **u32 status = MPC3024A_S_CirXY_LineZ_Radius_move (u8 CardID, i32 radius, i32 endpX, i32 endpY, i32 endpZ, u8 posi_mode[4], i32 VL, i32 VH, u32 Tacc_dec_ms, u32 TSacc_dec_ms, u8 direction, u8 wait)**

Purpose: X,Y axes doing circular interpolation as designated parameters and Z axis doing linear interpolation synchronously.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
radius	i32	Radius of the target circle
endpX	i32	end position (absolute or relative) for the X axis (-134,217,728 ≦ endpX ≦134,217,727)
endpY	i32	end position (absolute or relative) for the Y axis (-134,217,728 ≦ endpY ≦134,217,727)
endpZ	i32	end position (absolute or relative) for the Z axis (-134,217,728 ≦ endpZ ≦134,217,727) Linear interpolation will go from current Z position to endpZ position
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute
VL	i32	 Tsacc_ms Tsdec_ms VH VL Tsacc_ms Tsdec_ms Tacc_ms Tdec_ms time VH,VL : pps, (0 ≦ VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms,Tdec_ms: milliseconds. TSacc_ms = TSdec_ms = TSacc_dec_ms: milliseconds, time of s curve range
VH	i32	
Tacc_dec_ms	u32	
TSacc_dec_ms	u32	
direction	u8	0: CW 1: CCW
wait	u8	0: immediately act 1: wait for CSTA 2: **not available**

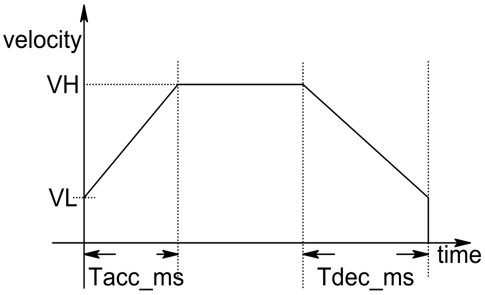
MPC3024A_T_ArcXY_LineZ_Radius_move

Format : **u32 status = MPC3024A_T_ArcXY_LineZ_Radius_move (u8 CardID,**
i32 radius, i32 endpX, i32 endpY, i32 endpZ, u8 posi_mode[4],
i32 VL, i32 VH, u32 Tacc_dec_ms, u8 direction, u8 wait)

Purpose: X,Y axes doing arc interpolation as designated parameters and Z axis doing linear interpolation synchronously.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
radius	i32	Radius of the target circle
endpX	i32	end position (absolute or relative) for the X axis (-134,217,728 ≦ endpX ≦134,217,727)
endpY	i32	end position (absolute or relative) for the Y axis (-134,217,728 ≦ endpY ≦134,217,727)
endpZ	i32	end position (absolute or relative) for the Z axis (-134,217,728 ≦ endpZ ≦134,217,727) Linear interpolation will go from current Z position to endpZ position
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute
VL	i32	 VH,VL: pps (0 ≦VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms = Tdec_ms = Tacc_dec_ms: acc/dec time in milliseconds.
VH	i32	
Tacc_dec_ms	u32	
direction	u8	0: CW 1: CCW
wait	u8	0: immediately act 1: wait for CSTA 2: **not available**

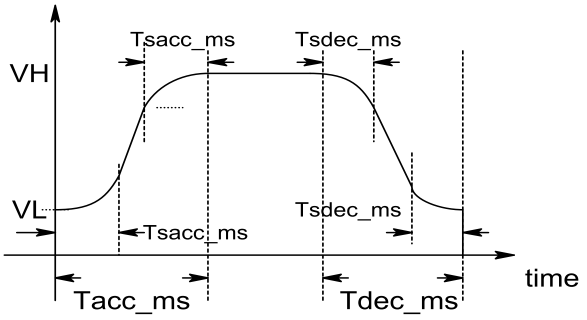
- **MPC3024A_S_ArcXY_LineZ_Radius_move**

Format : **u32 status = MPC3024A_S_ArcXY_LineZ_Radius_move (u8 CardID, i32 radius, i32 endpX, i32 endpY, i32 endpZ, u8 posi_mode[4], i32 VL, i32 VH, u32 Tacc_dec_ms, u32 TSacc_dec_ms, u8 direction, u8 wait)**

Purpose: X,Y axes doing arc interpolation as designated parameters and Z axis doing linear interpolation synchronously.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
radius	i32	Radius of the target circle
endpX	i32	end position (absolute or relative) for the X axis (-134,217,728 ≦ endpX ≦134,217,727)
endpY	i32	end position (absolute or relative) for the Y axis (-134,217,728 ≦ endpY ≦134,217,727)
endpZ	i32	end position (absolute or relative) for the Z axis (-134,217,728 ≦ endpZ ≦134,217,727) Linear interpolation will go from current Z position to endpZ position
posi_mode[4]	u8	posi_mode[0]: X axis, 0: relative, 1:absolute posi_mode[1]: Y axis, 0: relative, 1:absolute posi_mode[2]: Z axis, 0: relative, 1:absolute posi_mode[3]: A axis, 0: relative, 1:absolute
VL	i32	 Tsacc_ms Tsdec_ms VH VL Tsacc_ms Tsdec_ms time Tacc_ms Tdec_ms VH,VL : pps, (0 ≦ VH,VL ≦ 6553500)), composite speed of motion axes. Tacc_ms,Tdec_ms: milliseconds. TSacc_ms = TSdec_ms = TSacc_dec_ms: milliseconds, time of s curve range
VH	i32	
Tacc_dec_ms	u32	
TSacc_dec_ms	u32	
direction	u8	0: CW 1: CCW
wait	u8	0: immediately act 1: wait for CSTA 2: **not available**

9.14 Continuous motion function

For some applications such as gluing, you need to move continuously (with or without acce/dec at one motion to another). Another example of application is that you need to do motion at any curve profile; you can divide the curve at small segments to approach the final desired curve then command the motion at continuous mode. The on card motion chip provides 3 hardware buffers and continuous motion needs non stop of motion data to fill the buffers. Use

MPC3024A_continuous_flag_set( ) to enable / disable the continuous mode.

If you want to verify the exact number of buffer remained, use:

MPC3024A_continuous_buffer_no_read( )

MPC3024A_motion_status_read( ) for motion status read back.

Note:

It is meaningless to have a conditional wait in the continuous buffer but if you have the conditional wait at the start of continuous motion control is acceptable.

MPC3024A_continuous_flag_set

Format : **u32 status = MPC3024A_continuous_flag_set(u8 CardID, u8 axis, u8 conti_flag)**

Purpose: To set the continuous flag for new motion command.

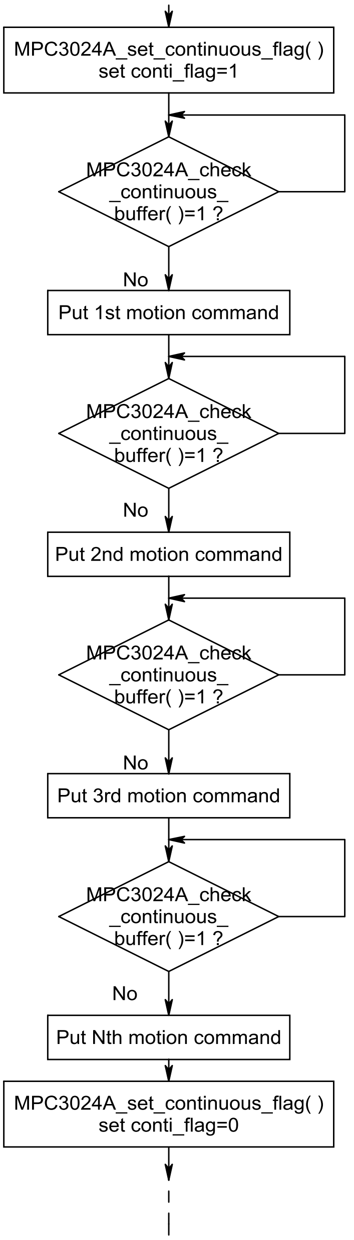
Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X axis 2: Z axis	1: Y axis 3: A axis
conti_flag	u8	0: disable continuous mode 1: enable continuous mode	

Note on using continuous mode

The sample control flow of the continuous mode application is as follows:



- **MPC3024A_continuous_buffer_no_read**

Format : **u32 status = MPC3024A_continuous_buffer_no_read(u8 CardID, u8 axis, u8 *remain_no)**

Purpose: To read how many buffered data left in continuous mode.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY switch	
axis	u8	0: X 2: Z	1: Y 3: A

Output:

Name	Type	Description
remain_no	u8	remained buffered data number

Note: There are 3 hardware continuous buffers only.

- **MPC3024A_motion_status_read**

Format : **u32 status = MPC3024A_motion_status_read(u8 CardID, u8 axis, u8 check_factor, u8 *ret_flag)**

Purpose: To read back the status of pulse command.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X 2: Z	1: Y 3: A
check_factor	u8	0: check SEND flag (pulse output flag, no pulse out=1) 1: check SPRF flag (continuous buffer flag, buffer full =1)	

Output:

Name	Type	Description
ret_flag	u8	for SEND flag 0: pulse output 1: no pulse output for SPRF flag 0: continuous buffer not full 1: continuous buffer full

9.15 Motion restart

Restart of motion is possible, if the motion is halted by software or hardware.

MPC3024A_Oneaxis_restart( ) for single axis restart.

MPC3024A_2axis_restart( ) for two axes restart.

MPC3024A_3axis_restart( ) for three axes restart.

MPC3024A_4axis_restart( ) for four axes restart.

Note: In continuous mode, restart may destroy the data in hardware buffer.

- **MPC3024A_Oneaxis_restart**

Format : u32 status = MPC3024A_Oneaxis_restart(u8 CardID, u8 axis)

Purpose: To restart the previously halted axis.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
axis	u8	0: X 1:Y 2: Z 3:A

- **MPC3024A_2axis_restart**

Format : u32 status = MPC3024A_2axis_restart(u8 CardID, u8 axis)

Purpose: To restart the previously halted 2 axes.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
axis	u8	0: XY 1: XZ 2: XA 3: YZ 4: YA 5: ZA

- **MPC3024A_3axis_restart**

Format : u32 status = MPC3024A_3axis_restart(u8 CardID, u8 axis)

Purpose: To restart the previously halted 2 axes.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
axis	u8	0: XYZ 1: XYA 2: XZA 3: YZA

- **MPC3024A_4axis_restart**

Format : **u32 status = MPC3024A_4axis_restart(u8 CardID)**

Purpose: To restart the previously halted 2 axes.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

9.16 Motion event and error status

When MPC3024A card generate an interrupt, the driver will generate an event and the user program of event process will wake up.

For the application that do not use interrupt, please use

MPC3024A_event_factor_set( ) to setup the event generated by the control card.

MPC3024A_event_flag_read( ) will give you the event generating source for your application.

MPC3024A_error_flag_read( ) will report the error conditions for your application.

- **MPC3024A_event_factor_set**

Format : **u32 status = MPC3024A_event_factor_set(u8 CardID, u8 axis, u32 event_factor)**

Purpose: To setup the event source that will generate flags at event occurs.

Parameters:

Input:

<table>
<tr><th>Name</th><th>Type</th><th colspan="3">Description</th></tr>
<tr><td>CardID</td><td>u8</td><td colspan="3">assigned by DIP/ROTARY SW</td></tr>
<tr><td>axis</td><td>u8</td><td colspan="3">0: X 1: Y
2: Z 3: A</td></tr>
<tr><td rowspan="21">event_factor</td><td rowspan="21">u32</td><td colspan="3">any bit of the following set to “1” means if the source is active, there is an interrupt will be generated.</td></tr>
<tr><td>Bit</td><td>Name</td><td>Description</td></tr>
<tr><td>bit0</td><td>IREN</td><td>Normal stop</td></tr>
<tr><td>bit1</td><td>IRNX</td><td>Successive start of the next operation</td></tr>
<tr><td>bit2</td><td></td><td>reserved</td></tr>
<tr><td>bit3</td><td></td><td>reserved</td></tr>
<tr><td>bit4</td><td>IRUS</td><td>Start of acceleration</td></tr>
<tr><td>bit5</td><td>IRUE</td><td>End of acceleration</td></tr>
<tr><td>bit6</td><td>IRDS</td><td>Start of deceleration</td></tr>
<tr><td>bit7</td><td>IRDE</td><td>End of deceleration</td></tr>
<tr><td>bit8</td><td>IRC1</td><td>Soft limit plus active</td></tr>
<tr><td>bit9</td><td>IRC2</td><td>Soft limit minus active</td></tr>
<tr><td>bit10</td><td></td><td>reserved</td></tr>
<tr><td>bit11</td><td></td><td>reserved</td></tr>
<tr><td>bit12</td><td>IRC5</td><td>Compare method satisfied</td></tr>
<tr><td>bit13</td><td></td><td>reserved</td></tr>
<tr><td>bit14</td><td>IRLT</td><td>LTC (latch) input making counter value latched</td></tr>
<tr><td>bit15</td><td></td><td>reserved</td></tr>
<tr><td>bit16</td><td>IRSD</td><td>SD (slow down)input on</td></tr>
<tr><td>bit17</td><td></td><td>reserved</td></tr>
<tr><td>bit18</td><td>IRSA</td><td>CSTA (common start) input on</td></tr>
</table>

Note:

This function is only used in the application program that **do not** use interrupt, if your application implemented with the interrupt function of the MPC3024A card, please use MPC3024A_IRQ_source_set( ) instead.

- **MPC3024A_event_flag_read**

Format : **u32 status = MPC3024A_event_flag_read(u8 CardID, u8 axis, u32 *event_flag)**

Purpose: To read back the status of event source.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A

Output:

Name	Type	Description		
event_flag	u32	while any of the following bit set to “1” means the event source is active.		
		Bit	Name	Description
		bit0	IREN	Normal stop
		bit1	IRNX	Successive start of the next operation
		bit2		reserved
		bit3		reserved
		bit4	IRUS	Start of acceleration
		bit5	IRUE	End of acceleration
		bit6	IRDS	Start of deceleration
		bit7	IRDE	End of deceleration
		bit8	IRC1	Soft limit plus active
		bit9	IRC2	Soft limit minus active
		bit10		reserved
		bit11		reserved
		bit12	IRC5	Compare method satisfied
		bit13		reserved
		bit14	IRLT	LTC (latch) input making counter value latched
		bit15		reserved
		bit16	IRSD	SD (slow down)input on
		bit17		reserved
		bit18		reserved
		bit19	IRSA	CSTA (common start) input on

Note:

This function is only used in the application program that **do not** use interrupt, if your application implemented with the interrupt function of the MPC3024A card, please use MPC3024A_IRQ_status_read( ) instead.

MPC3024A_error_flag_read

Format : **u32 status = MPC3024A_error_flag_read(u8 CardID, u8 axis, u32 *error_flag)**

Purpose: To read back the status of error source.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A

Output:

<table>
<tr><th>Name</th><th>Type</th><th colspan="3">Description</th></tr>
<tr><td rowspan="20">error_flag</td><td rowspan="20">u32</td><td colspan="3">while any of the following bit set to “1” means the error source is active.</td></tr>
<tr><td>Bit</td><td>Name</td><td>Description</td></tr>
<tr><td>bit0</td><td>ESC1</td><td>SL+ (Software Limit +) error</td></tr>
<tr><td>bit1</td><td>ESC2</td><td>SL- (Software Limit -) error</td></tr>
<tr><td>bit2</td><td></td><td>reserved</td></tr>
<tr><td>bit3</td><td></td><td>reserved</td></tr>
<tr><td>bit4</td><td>ESC5</td><td>compare action satisfied</td></tr>
<tr><td>bit5</td><td>ESPL</td><td>LS+(EL+) error</td></tr>
<tr><td>bit6</td><td>ESML</td><td>LS-(EL-) error</td></tr>
<tr><td>bit7</td><td>ESAL</td><td>ALM error</td></tr>
<tr><td>bit8</td><td>ESSP</td><td>CSTP error</td></tr>
<tr><td>bit9</td><td>ESEM</td><td>EMG error</td></tr>
<tr><td>bit10</td><td>ESSD</td><td>SD error</td></tr>
<tr><td>bit11</td><td></td><td>reserved</td></tr>
<tr><td>bit12</td><td>ESDT</td><td>Abnormal data</td></tr>
<tr><td>bit13</td><td>ESIP</td><td>Abnormal stop during interpolation</td></tr>
<tr><td>bit14</td><td>ESPO</td><td>PA/PB input counter overflow</td></tr>
<tr><td>bit15</td><td>ESAO</td><td>In-position counter exceed the counting range during interpolation</td></tr>
<tr><td>bit16</td><td>ESEE</td><td>EA/EB input error</td></tr>
<tr><td>bit17</td><td>ESPE</td><td>PA/PB input error</td></tr>
</table>

Note:

This function is only used in the application program that **do not** use interrupt, if your application implemented with the interrupt function of the MPC3024A card, please use MPC3024A_IRQ_status_read( ) instead.

9.17 Soft limit protection function

To avoid mistake of position data, software limit is the first aid before hardware limit switch protection. You must configure how to stop and the source of motion axis, use

MPC3024A_softlimit_config_set( ) to setup configuration.
MPC3024A_softlimit_config_read( ) to read back configuration.
MPC3024A_softlimit_data_set( ) to setup the coordinate data of limit.
MPC3024A_softlimit_data_read( ) to read back preset data.
MPC3024A_softlimit_enable_set( ) to enable / disable software limit function.
MPC3024A_softlimit_enable_read( ) to read back configuration.
MPC3024A_softlimit_flag_read( ) to read the software limit flag for verifying.

● MPC3024A_softlimit_config_set

Format : **u32 status = MPC3024A_softlimit_config_set(u8 CardID, u8 axis, u8 source_sel, u8 SL_action)**

Purpose: To configure the software limit axis, coordinate source and how to stop.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A
source_sel	u8	0: current position of command 1: feedback counter position
SL_action	u8	how to stop while software limit alarm 0: no processing (to be used for INT, pin output) 1: immediate stop 2: decelerate to stop

- **MPC3024A_softlimit_config_read**

Format : **u32 status = MPC3024A_softlimit_config_read(u8 CardID, u8 axis, u8* source_sel, u8* SL_action)**

Purpose: Readback the software limit parameter.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A

Output:

Name	Type	Description
source_sel	u8	0: current position of command 1: feedback counter position
SL_action	u8	how to stop while software limit alarm 0: no processing (to be used for INT, pin output) 1: immediate stop 2: decelerate to stop

- **MPC3024A_softlimit_data_set**

Format : **u32 status = MPC3024A_softlimit_data_set(u8 CardID, u8 axis, i32 P_limit, i32 N_limit)**

Purpose: To set the coordinate of software limit.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A
P_limit	i32	soft limit of positive direction (-134,217,728 ≦P_limit≦+134,217,727)
N_limit	i32	soft limit of negaitive direction (-134,217,728 ≦N_limit≦+134,217,727)

- **MPC3024A_softlimit_data_read**

Format : **u32 status = MPC304A_softlimit_data_read(u8 CardID, u8 axis, i32* P_limit, i32* N_limit)**

Purpose: Readback the coordinate of software limit.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY switch	
axis	u8	0: X 2: Z	1: Y 3: A

Output:

Name	Type	Description
P_limit	i32	soft limit of positive direction (-134,217,728 ≦P_limit≦+134,217,727)
N_limit	i32	soft limit of negaitive direction (-134,217,728 ≦N_limit≦+134,217,727)

- **MPC3024A_softlimit_enable_set**

Format : **u32 status = MPC3024A_softlimit_enable_set(u8 CardID, u8 axis, u8 ON_OFF)**

Purpose: To enable / disable software limit.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X 2: Z	1: Y 3: A
ON_OFF	u8	0: disable	1: enable

- **MPC3024A_softlimit_enable_read**

Format : **u32 status = MPC3024A_softlimit_enable_read(u8 CardID, u8 axis, u8* ON_OFF)**

Purpose: Readback the status of enable / disable software limit.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X 2: Z	1: Y 3: A

Output:

Name	Type	Description	
ON_OFF	u8	0: disable	1: enable

- **MPC3024A_softlimit_flag_read**

Format : **u32 status = MPC3024A_softlimit_flag_read(u8 CardID, u8 axis, u8 *P_limit_flag, u8 *N_limit_flag)**

Purpose: To read back software limit flag.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X 2: Z	1: Y 3: A

Output:

Name	Type	Description	
P_limit_flag	u8	0: P_limit en-active	1: P_limit active
N_limit_flag	u8	0: N_limit en-active	1: N_limit active

9.18 Manual pulser function

For the application requires pulse handler (or pulser) to work as manual control of speed or position, the MPC3024A provides extra input on JM3, the (PA1, PB1), (PA2,PB2), (PA3,PB3), (PA4, PB4) are paired input of A and B phase.

To configure the operating mode of the pulse handler with:

MPC3024A_pulser_mode_set( )

MPC3024A_pulser_mode_read( ) to read back configuration.

Concerning the pulse handler input counter, use

MPC3024A_pulser_counter_set( ) to set pulse counter, and

MPC3024A_pulser_counter_read( ) to read back the counter value.

The above mentioned functions are single function and step by step procedures they seemed tedious. If you use pulse handler as a manual input device and MPC provides an integral application mode. You just configure it by mapping the device to under control axis and enable it; now you can use the pulser to control the motion axis.

First map the pulse handler to the motion axis by

MPC3024A_pulser_map_set( )

and then enable/disable the motion function and multiple rate by using:

MPC3024A_pulser_motion_enable( )

- **MPC3024A_pulser_mode_set**

Format : **u32 status = MPC3024A_pulser_mode_set(u8 CardID, u8 axis, u8 pulser_mode, u8 direction)**

Purpose: To configure the pulse handler operation mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A
pulser_mode	u8	0: quadrature input A lead B up count, multiply by 1 1: quadrature input A lead B up count, multiply by 2 2: quadrature input A lead B up count, multiply by 4 3: count up at A phase rising edge, count down at B phase rising edge
direction	u8	override the default direction 0: as default direction 1: invert the direction

- **MPC3024A_pulser_mode_read**

Format : **u32 status = MPC3024A_pulser_mode_read(u8 CardID, u8 axis, u8* pulser_mode, u8* direction)**

Purpose: Read back the pulse handler operation mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A

Output:

Name	Type	Description
pulser_mode	u8	0: quadrature input A lead B up count, multiply by 1 1: quadrature input A lead B up count, multiply by 2 2: quadrature input A lead B up count, multiply by 4 3: count up at A phase rising edge, count down at B phase rising edge
direction	u8	override the default direction 0: as default direction 1: invert the direction

- **MPC3024A_pulser_counter_set**

Format : **u32 status = MPC3024A_pulser_counter_set(u8 CardID, u8 axis, i32 counter_value)**

Purpose: To set the pulse counter value.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A
counter_value	i32	pulse counter value to be set (-134,217,728 ≦ counter_value ≦ 134,217,727)

- **MPC3024A_pulser_counter_read**

Format : **u32 status = MPC3024A_pulser_counter_read(u8 CardID, u8 axis, i32 *counter_value)**

Purpose: To read the pulse counter value.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A

Output:

Name	Type	Description
counter_value	i32	pulse counter value (-134,217,728 ≦ counter_value ≦ 134,217,727)

- **MPC3024A_pulser_map_set**

Format : **u32 status = MPC3024A_pulser_map_set(u8 CardID, u8 axis, u8 Map_source, u8 Direction)**

Purpose: To map the source (pulse handler) to the target motion axis.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	target motion under control axis 0: Motion axis is X 1: Motion axis is Y 2: Motion axis is Z 3: Motion axis is A
Map_source	u8	pulser hardware is connected to 0: Pulse handler in X axis 1: Pulse handler in Y axis 2: Pulse handler in Z axis 3: Pulse handler in A axis
Direction	u8	0: rotate same direction with pulse handler input 1: rotate counter direction with pulse handler input

- **MPC3024A_pulser_motion_enable**

Format : **u32 status = MPC3024A_pulser_motion_enable(u8 CardID, u8 axis, u8 enable, u16 Multiple)**

Purpose: To enable pulse handler function and the multiple rate

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: Motion axis is X 1: Motion axis is Y 2: Motion axis is Z 3: Motion axis is A
enable	u8	0: disable 1: enable
Multiple	u16	The number of pulse output to motion axis for an unit of pulse handler input. (1 ≦ Multiple ≦ 1000)

Note1: This function can only be used in Windows 2000 P3 800MHz and grade-up system.

Note2: Be sure the motion pulse output is finished before using MPC3024A_pulser_motion_enable( ) function, you can check it by the value of ret_flag which is returned by calling MPC3024A_motion_status_read( ) and set check_factor=0.

Note3: Be sure to disable pulse handler function before calling any motion command, such as MPC3024A_T_position_move(), MPC3024A_S_position_move()**…**

9.19 Multi-function feedback counter

Each axis has a feedback counter on card, you can use the counter to connect to a linear scale or feedback encoder to correct or confirm the motion accuracy; MPC3024A provide external encoder input on JF1/JF2, with the differential input EA+/EA-, EB+/EB-,EZ+/EZ-. Before you use the counter, you must setup the counter input mode (refer ***MPC3024A_pulse_inmode_set***) then use:

MPC3024A_FB_counter_set( ) to preset the counter value.

MPC3024A_FB_counter_read( ) to read counter value.

If you have configure latch input function (refer ***MPC3024A_LTC_PIN_***set), use

MPC3024A_FB_counter_latch_value_read( ) to read the latched counter value.

If you have configure compare output function (refer ***MPC3024A_CMP_PIN_***set), use

MPC3024A_CMP_out_set( ) to configure the compare output mode.

MPC3024A_CMP_out_read( ) to read back configuration.

MPC3024A_CMP_data_set( ) to preset the value to the comparator.

MPC3024A_CMP_data_read( ) to read back preset value.

MPC3024A_CMP_flag_read( ) to read compare out flag for verifying the active state of the function.

- **MPC3024A_pulse_inmode_set**

Format : **u32 status = MPC3024A_pulse_inmode_set(u8 CardID, u8 axis, u8 pulse_inmode, u8 count_dir)**

Purpose: To set the encoder input mode and counting polarity.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
pulse_inmode	u8	0 ~ 3 (See **Note on pulse in mode)**
count_dir	u8	0: normal counting 1: reverse counting

Note:

The signals of position counter come from EA+/EA- for A phase, EB+/EB- for B phase and EZ+/EZ- for Z phase input, each axis has it’s own position counter.

- **MPC3024A_pulse_inmode_read**

Format : **u32 status = MPC3024A_pulse_inmode_read(u8 CardID, u8 axis,
u8* pulse_inmode, u8* count_dir)**

Purpose: Readback the parameters of the encoder input mode and counting polarity.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
pulse_inmode	u8	0 ~ 3 (See **Note on pulse in mode)**
count_dir	u8	0: normal counting 1: reverse counting

Note on pulse in mode:

pulse_inmode	Description
0 (00)	multiply by 1 and up count while phase A lead phase B
1 (01)	multiply by 2 and up count while phase A lead phase B
2 (10)	multiply by 4 and up count while phase A lead phase B
3 (11)	up count while phase A input rising down count while rising of phase B input

- **MPC3024A_FB_counter_set**

Format : **u32 status = MPC3024A_FB_counter_set(u8 CardID, u8 axis, i32 value)**

Purpose: To preset the feedback counter value.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A
value	i32	pulse counter value (-134,217,728 ≦ value ≦ 134,217,727)

- **MPC3024A_FB_counter_read**

Format : **u32 status = MPC3024A_FB_counter_read(u8 CardID, u8 axis, i32 *value)**

Purpose: To read the encoder feedback counter value.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A

Output:

Name	Type	Description
value	i32	pulse counter value (-134,217,728 ≦ value ≦ 134,217,727)

- **MPC3024A_FB_counter_latch_value_read**

Format : **u32 status = MPC3024A_FB_counter_latch_value_read(u8 CardID, u8 axis, i32 *value)**

Purpose: To read the latced value of feedback counter.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A

Output:

Name	Type	Description
value	i32	pulse counter value (-134,217,728 ≦ value ≦ 134,217,727)

- **MPC3024A_CMP_out_set**

Format : **u32 status = MPC3024A_CMP_out_set(u8 CardID, u8 axis,**
u8 cmp_source, u8 cmp_method, u8 cmp_action)

Purpose: To setup the compare mode of feedback comparator.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A
cmp_source	u8	0: to compare with the current position command counter 1: to compare with the feedback counter 2: undefined 3: to compare with the pulser counter
cmp_method	u8	1: compare out at equal, and does not care direction 2: compare out at equal while counting up 3: compare out at equal while counting down 4: compare out at preset value > counter value 5: compare out at preset value < counter value
cmp_action	u8	0: No action, use only to generate interrupt and compare output 1: immediate stop 2: decelerate to stop

- **MPC3024A_CMP_out_read**

Format : **u32 status = MPC3024A_CMP_out_read(u8 CardID, u8 axis, u8* cmp_source, u8* cmp_method, u8* cmp_action)**

Purpose: Read back the configuration of the compare mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A

Output:

Name	Type	Description
cmp_source	u8	0: to compare with the current position command counter 1: to compare with the feedback counter 2: undefined 3: to compare with the pulser counter
cmp_method	u8	1: compare out at equal, and does not care direction 2: compare out at equal while counting up 3: compare out at equal while counting down 4: compare out at preset value > counter value 5: compare out at preset value < counter value
cmp_action	u8	0: No action, use only to generate interrupt and compare output 1: immediate stop 2: decelerate to stop

- **MPC3024A_CMP_data_set**

Format : **u32 status = MPC3024A_CMP_data_set(u8 CardID, u8 axis, i32 cmp_data)**

Purpose: To preset the comparator value.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A
cmp_data	i32	comparator value to be preset (-134,217,728 ≦ cmp_data ≦ 134,217,727)

- **MPC3024A_CMP_data_read**

Format : **u32 status = MPC3024A_CMP_data_read(u8 CardID, u8 axis, i32* cmp_data)**

Purpose: Read back the preset comparator value.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A

Output:

Name	Type	Description
cmp_data	i32	comparator value to be preset (-134,217,728 ≦ cmp_data ≦ 134,217,727)

- **MPC3024A_CMP_flag_read**

Format : **u32 status = MPC3024A_CMP_flag_read(u8 CardID, u8 axis, u8 *cmp_flag)**

Purpose: To read back the compare flag.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y 2: Z 3: A

Output:

Name	Type	Description
cmp_flag	u8	0: the compare condition not meet 1: the compare condition has met

9.20 Interrupt function

Sometimes you want your application to take care of the motion while special event occurs; interrupt function is the right choice.

On MPC3024A card there are many sources to generate interrupt, select the interrupt source by

MPC3024A_IRQ_source_set( ), use

MPC3024A_IRQ_status_read( ) to read the interrupt event generating source. . At the end of interrupt service routine, you had better to clear the status buufer owing to the data will not change until the next interrupt comes in. Clear the status by:

MPC3024A_IRQ_status_clear( )

Then you should enable / disable the hardware of the interrupt source,

MPC3024A_IRQ_mask_set( ) will do and your program and hardware configuration is ready to service. To read back the IRQ mask data:

MPC3024A_IRQ_mask_read( ) will do.

Now you must tell the driver your interrupt service routine by

MPC3024A_IRQ_process_link( )

To enable the IRQ function, the global enable of the IRQ function should be set by

MPC3024A_IRQ_enable( ), now all the hardware, service routine and process is ready to response the IRQ service.

If you do not use interrupt anymore and you will close your application program, be sure to use

MPC3024A_IRQ_disable( ) to release the resource.

- **MPC3024A_IRQ_source_set**

Format : **u32 status = MPC3024A_IRQ_source_set(u8 CardID, u8 axis,
u32 REST_source_sel, u32 RIST_source_sel)**

Purpose: To setup the error/event source that will generate interrupt at error/event occurs.

Parameters:

Input:

Name	Type	Description		
CardID	u8	assigned by DIP/ROTARY SW		
axis	u8	0: X 1: Y 2: Z 3: A		
REST_source_sel	u32	any bit of the following set to “1” means if the error source is active, there is an interrupt will be generated.		
		Bit	Name	Description
		bit0	ESC1	SL+ (Software Limit +) error
		bit1	ESC2	SL- (Software Limit -) error
		bit2		reserved
		bit3		reserved
		bit4	ESC5	compare action satisfied
		bit5	ESPL	LS+(EL+) error
		bit6	ESML	LS-(EL-) error
		bit7	ESAL	ALM error
		bit8	ESSP	CSTP error
		bit9	ESEM	EMG error
		bit10	ESSD	SD error
		bit11		reserved
		bit12	ESDT	Abnormal data
		bit13	ESIP	Abnormal stop during interpolation
		bit14	ESPO	PA/PB input counter overflow
		bit15	ESAO	In-position counter exceed the counting range during interpolation
		bit16	ESEE	EA/EB input error
		bit17	ESPE	PA/PB input error
RIST_source_sel	u32	any bit of the following set to “1” means if the event source is active, there is an interrupt will be generated.		
		Bit	Name	Description
		bit0	IREN	Normal stop
		bit1	IRNX	Successive start of the next operation
		bit2		Reserved
		bit3		Reserved
		bit4	IRUS	Start of acceleration
		bit5	IRUE	End of acceleration
		bit6	IRDS	Start of deceleration
		bit7	IRDE	End of deceleration
		bit8	IRC1	Soft limit plus active
		bit9	IRC2	Soft limit minus active
		bit10		Reserved
		bit11		Reserved

		bit12	IRC5	Compare method satisfied
		bit13		Reserved
		bit14	IRLT	LTC (latch) input making counter value latched
		bit15		Reserved
		bit16	IRSD	SD (slow down)input on
		bit17		reserved
		bit18	IRSA	CSTA (common start) input on

Note:

This function is only used in the application program that **do** use interrupt function of the MPC3024A card, if you **do not** use interrupt function please use MPC3024A_event_factor_set( ) instead.

- ### MPC3024A_IRQ_status_read

Format : **u32 status = MPC3024A_IRQ_status_read(u8 CardID, u8 axis, u8 *IRQ_Status, u32 *REST, u32 *RIST)**

Purpose: To read back the status of interrupt event source.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X 2: Z	1: Y 3: A

Output:

Name	Type	Description		
IRQ_Status	u8	bit0	0: error interrupt(REST) not active 1: error interrupt(REST) active	
		bit1	0: event interrupt(RIST) not active 1: event interrupt(RIST) active	
		bit2	1: TIMER generates interrupt	
REST	u32	while any of the following bit set to “1” means the error source is active.		
		Bit	Name	Description
		bit0	ESC1	SL+ (Software Limit +) error
		bit1	ESC2	SL- (Software Limit -) error
		bit2		reserved
		bit3		reserved
		bit4	ESC5	compare action satisfied
		bit5	ESPL	LS+(EL+) error
		bit6	ESML	LS-(EL-) error
		bit7	ESAL	ALM error
		bit8	ESSP	CSTP error
		bit9	ESEM	EMG error
		bit10	ESSD	SD error
		bit11		reserved
		bit12	ESDT	Abnormal data
		bit13	ESIP	Abnormal stop during interpolation
		bit14	ESPO	PA/PB input counter overflow
		bit15	ESAO	In-position counter exceed the counting range during interpolation
		bit16	ESEE	EA/EB input error
		bit17	ESPE	PA/PB input error
RIST	u32	while any of the following bit set to “1” means the event source is active.		
		Bit	Name	Description
		bit0	IREN	Normal stop
		bit1	IRNX	Successive start of the next operation
		bit2		reserved
		bit3		reserved
		bit4	IRUS	Start of acceleration
		bit5	IRUE	End of acceleration
		bit6	IRDS	Start of deceleration

		bit7	IRDE	End of deceleration
		bit8	IRC1	Soft limit plus active
		bit9	IRC2	Soft limit minus active
		bit10		reserved
		bit11		reserved
		bit12	IRC5	Compare method satisfied
		bit13		reserved
		bit14	IRLT	LTC (latch) input making counter value latched
		bit15		reserved
		bit16	IRSD	SD (slow down)input on
		bit17		reserved
		bit18		reserved
		bit19	IRSA	CSTA (common start) input on

Note:

This function is only used in the application program that **do** use interrupt function of the MPC3024A card, if you **do not** use interrupt function please use MPC3024A_event_flag_read( ) and MPC3024A_error_flag_read( ) instead.

- **MPC3024A_IRQ_status_clear**

Format : **u32 status = MPC3024A_IRQ_status_clear(u8 CardID, u8 axis, u32 REST, u32 RIST)**

Purpose: To clear the status of interrupt event source.

Parameters:

Input:

<table>
<tr><th>Name</th><th>Type</th><th colspan="3">Description</th></tr>
<tr><td>CardID</td><td>u8</td><td colspan="3">assigned by DIP/ROTARY SW</td></tr>
<tr><td>axis</td><td>u8</td><td colspan="3">0: X 1: Y
2: Z 3: A</td></tr>
<tr><td rowspan="20">REST</td><td rowspan="20">u32</td><td colspan="3">while any of the following bit set to “1” means to reset the corresponding bit.</td></tr>
<tr><td>Bit</td><td>Name</td><td>Description</td></tr>
<tr><td>bit0</td><td>ESC1</td><td>SL+ (Software Limit +) error</td></tr>
<tr><td>bit1</td><td>ESC2</td><td>SL- (Software Limit -) error</td></tr>
<tr><td>bit2</td><td></td><td>reserved</td></tr>
<tr><td>bit3</td><td></td><td>reserved</td></tr>
<tr><td>bit4</td><td>ESC5</td><td>compare action satisfied</td></tr>
<tr><td>bit5</td><td>ESPL</td><td>LS+(EL+) error</td></tr>
<tr><td>bit6</td><td>ESML</td><td>LS-(EL-) error</td></tr>
<tr><td>bit7</td><td>ESAL</td><td>ALM error</td></tr>
<tr><td>bit8</td><td>ESSP</td><td>CSTP error</td></tr>
<tr><td>bit9</td><td>ESEM</td><td>EMG error</td></tr>
<tr><td>bit10</td><td>ESSD</td><td>SD error</td></tr>
<tr><td>bit11</td><td></td><td>reserved</td></tr>
<tr><td>bit12</td><td>ESDT</td><td>Abnormal data</td></tr>
<tr><td>bit13</td><td>ESIP</td><td>Abnormal stop during interpolation</td></tr>
<tr><td>bit14</td><td>ESPO</td><td>PA/PB input counter overflow</td></tr>
<tr><td>bit15</td><td>ESAO</td><td>In-position counter exceed the counting range during interpolation</td></tr>
<tr><td>bit16</td><td>ESEE</td><td>EA/EB input error</td></tr>
<tr><td>bit17</td><td>ESPE</td><td>PA/PB input error</td></tr>
<tr><td rowspan="16">RIST</td><td rowspan="16">u32</td><td colspan="3">while any of the following bit set to “1” means to reset the corresponding bit.</td></tr>
<tr><td>Bit</td><td>Name</td><td>Description</td></tr>
<tr><td>bit0</td><td>IREN</td><td>Normal stop</td></tr>
<tr><td>bit1</td><td>IRNX</td><td>Successive start of the next operation</td></tr>
<tr><td>bit2</td><td></td><td>reserved</td></tr>
<tr><td>bit3</td><td></td><td>reserved</td></tr>
<tr><td>bit4</td><td>IRUS</td><td>Start of acceleration</td></tr>
<tr><td>bit5</td><td>IRUE</td><td>End of acceleration</td></tr>
<tr><td>bit6</td><td>IRDS</td><td>Start of deceleration</td></tr>
<tr><td>bit7</td><td>IRDE</td><td>End of deceleration</td></tr>
<tr><td>bit8</td><td>IRC1</td><td>Soft limit plus active</td></tr>
<tr><td>bit9</td><td>IRC2</td><td>Soft limit minus active</td></tr>
<tr><td>bit10</td><td></td><td>reserved</td></tr>
<tr><td>bit11</td><td></td><td>reserved</td></tr>
<tr><td>bit12</td><td>IRC5</td><td>Compare method satisfied</td></tr>
<tr><td>bit13</td><td></td><td>reserved</td></tr>
</table>

		bit14	IRLT	LTC (latch) input making counter value latched
		bit15		reserved
		bit16	IRSD	SD (slow down)input on
		bit17		reserved
		bit18		reserved
		bit19	IRSA	CSTA (common start) input on

Note:

The interrupt status will keep until the next interrupt comes in. It is a better approach to clear the corresponding bits at the end of of the service routine.

- **MPC3024A_IRQ_mask_set**

Format : **u32 status = MPC3024A_IRQ_mask_set(u8 CardID, u8 axis, u8 on_off)**

Purpose: To set the interrupt mask of designated axis or timer.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X 2: Z 4:timer	1: Y 3: A
on_off	u8	0: disable	1: enable

- **MPC3024A_IRQ_mask_read**

Format : **u32 status = MPC3024A_IRQ_mask_read(u8 CardID, u8 axis, u8 *on_off)**

Purpose: To read the interrupt mask of designated axis or timer.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X 2: Z 4:timer	1: Y 3: A

Output:

Name	Type	Description	
on_off	u8	0: disable	1: enable

- **MPC3024A_IRQ_process_link**

Format : **status = MPC3024A_IRQ_process_link(u8 CardID,**
void (__stdcall *callbackAddr)(u8 CardID))

Purpose: Link irq service routine to driver

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP SW
callbackAddr	void	callback address of service routine

- **MPC3024A_IRQ_enable**

Format : **u32 status = MPC3024A_IRQ_enable(u8 CardID)**

Purpose: To enable the interrupt function.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

- **MPC3024A_IRQ_disable**

Format : **u32 status = MPC3024A_IRQ_disable(u8 CardID)**

Purpose: To disable the interrupt function, and release the resource and close thread.

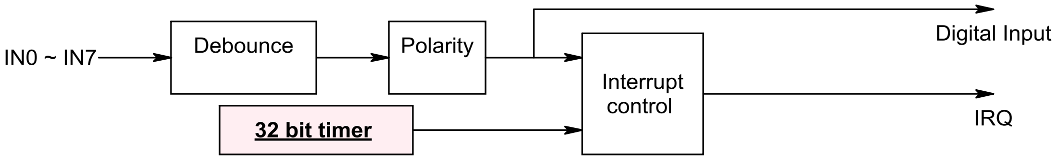
Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

9.21 Timer function

The build in 32 bit timer based on 1 us time base can be used as system clock to generate interrupt for periodical task.



To setup timer or change time constant

MPC3024A_timer_set( ) and start by

MPC3024A_timer_start( ) and stop by

MPC3024A_timer_stop( )

To read back the timer value on the fly,

MPC3024A_timer_read( ) will do.

The timer interrupt can be reached by:

MPC3024A_IRQ_mask_set( ) (refer 9.20 Interrupt function)

If you want to dedicated control the timer associated registers, use

MPC3024A_TC_set( ) to set registers and use

MPC3024A_TC_read( ) to read back settings.

- **MPC3024A_timer_set**

Format : **u32 status = MPC3024A_timer_set (u8 CardID, u8 source, u32 time_constant)**

Purpose: To setup timer operation mode or update timer

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
source	u8	0: PRELOAD 1: TIMER
time_constant	u32	Timer constant based on 1us clock

Note:

1. Time constant is based on 1us clock, period T= (time_constant +1) * 1us
2. If you enable the timer interrupt, the period T must at least longer than the system interrupt response time else the system will be hanged by excess interrupts.
3. PRELOAD is the register for timer to re-load, the value will be valid while timer count to zero and reload the data.

- **MPC3024A_timer_start**

Format : **u32 status = MPC3024A_timer_start(u8 CardID)**

Purpose: To start timer operation mode

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

- **MPC3024A_timer_stop**

Format : **u32 status = MPC3024A_timer_stop(u8 CardID)**

Purpose: To stop timer operation mode

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

- **MPC3024A_timer_read**

Format : **u32 status = MPC3024A_timer_read (u8 CardID, u8 source, u32 *time_constant)**

Purpose: To read back timer value on the fly.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
source	u8	0: PRELOAD 1: TIMER

Output:

Name	Type	Description
time_constant	u32	Timer constant based on 1us clock

- **MPC3024A_TC_set**

Format : **u32 status= MPC3024A_TC_set (u8 CardID,u8 index,u32 data)**

Purpose: To set data to timer register

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
index	u8	0: TC_CONTROL 1: PRELOAD 2: COUNTER 3: TC_IRQ_MASK
data	u32	register data to be set

Note: please refer the next segment “Note: Meaning of setting or return value of different index”

- **MPC3024A_TC_read**

Format : **u32 status= MPC3024A_TC_read (u8 CardID,u8 index,u32 *data)**

Purpose: To read data from timer register

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
index	u8	0: TC_CONTROL 1: PRELOA D 2: COUNTER 3: TC_IRQ_MASK 4: TC_IRQ _STATUS

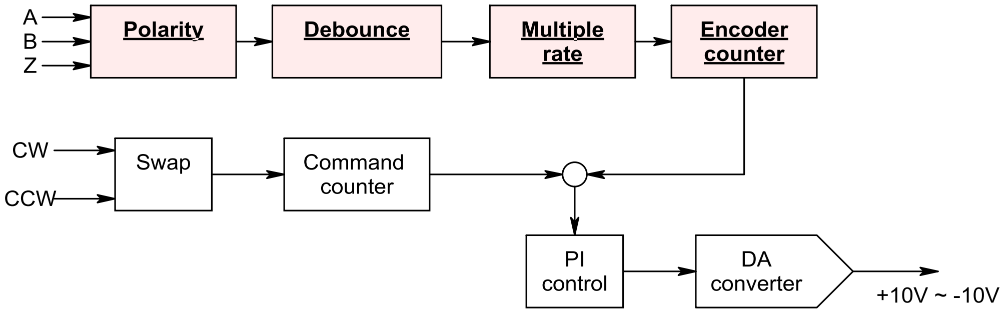
Output:

Name	Type	Description
data	u32	Data read back

Note: Meaning of setting or return value of different index

index	register	value	meaning
0	TC_CONTROL	0	STOP, stop operation of TC
		1	START, start operation of TC
1	PRELOAD	1~0xffffffff	Counter or timer or PWM preload value
2	COUNTER	1~0xffffffff	Set (write): will write preload and counter Read : will read counter on the fly
3	TC_IRQ _MASK	0	Set(Write): not allow timer to generate IRQ Read: TC IRQ mask off
		1	Set(Write): allow timer to generate IRQ Read: TC IRQ enabled
4	TC_IRQ _STATUS	0	Set(Write): no effect Read: No time up interrupt
		1	Set(Write): Reset TC_IRQ _STATUS Read: Time up interrupt occurs

9.22 Encoder counter function (**only for MPC3024AC)



The encoder input comes from the servo motor feedback which is the important information of speed and position feedback. MPC3024AC prove the encoder input a programmable debounce filter to filter out the unwanted glitches. From 512K up to 8M and the default is 1M (drop out pulse width less than 1us). Also the multiple rate of the encoder which can be x1,x2,x4 to increase the control accuracy.

MPC3024AC_encoder_mode_set( ) will set up the environment as you need.

MPC3024AC_encoder_mode_read( ) is used to read back for verification.

To fit different kinds of encoders and motion direction, the encoder polarity can be set by:

MPC3024AC_encoder_polarity_set( ) and read back to verify by:

MPC3024AC_encoder_polarity_read( )

To read the instantaneous value of the encoder state, apply

MPC3024AC_encoder_status_read( )

- **MPC3024AC_encoder_mode_set**

Format : **u32 status = MPC3024AC_encoder_mode_set(u8 CardID, u8 axis, u16 in_mode, u16 debounce_time, u16 multiple_rate)**

Purpose: To setup encoder counter operating mode

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
in_mode	u16	0 : quadrature mode A,B phase input (default) 1 : CW,CCW mode, CW<-A input, CCW <-B input 2 : Clock,Direction mode, clock<--A input, direction <-B input
debounce_time	u16	0 : debounce up to 512K (drop pulse width less than 1.95us) 1 : debounce upto 1M (drop pulse width less than 1us) (default) 2 : debounce upto 2M (drop pulse width less than 0.5us) 3 : debounce upto 4M (drop pulse width less than 0.25us) 4 : debounce upto 8M (drop pulse width less than 0.125us)
multiple_rate	u16	0: multiple rate x4 (default) 1: multiple rate x2 2: multiple rate x1

- **MPC3024AC_encoder_mode_read**

Format : **u32 status = MPC3024AC_encoder_mode_read(u8 CardID, u8 axis, u16 *in_mode, u16 *debounce_time, u16 *multiple_rate)**

Purpose: To read data from encoder mode register

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
in_mode	u16	0 : quadrature mode A,B phase input (default) 1 : CW,CCW mode, CW<-A input, CCW <-B input 2 : Clock,Direction mode, clock<--A input, direction <-B input
debouce_time	u16	0 : debounce up to 512K (drop pulse width less than 1.95us) 1 : debounce upto 1M (drop pulse width less than 1us) (default) 2 : debounce upto 2M (drop pulse width less than 0.5us) 3 : debounce upto 4M (drop pulse width less than 0.25us) 4 : debounce upto 8M (drop pulse width less than 0.125us)
multiple_rate	u16	0: multiple rate x4 (default) 1: multiple rate x2 2: multiple rate x1

- **MPC3024AC_encoder_polarity_set**

Format : **u32 status = MPC3024AC_encoder_polarity_set(u8 CardID, u8 axis, u16 polarity)**

Purpose: To setup encoder polarity

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
polarity	u16	encoder polarity b5~b0 b0 : A input polarity b1 : B input polarity b2 : Z input polarity b3 : HOME input polarity b4 : +LS input polarity b5 : -LS input polarity A bit set 0 is normal polarity and set 1 is to invert the polarity.

- **MPC3024AC_encoder_polarity_read**

Format : **u32 status = MPC3024AC_encoder_polarity_read(u8 CardID, u8 axis, u16 *polarity)**

Purpose: To read data from encoder polarity register

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
polarity	u16	encoder polarity b5~b0 b0 : A input polarity b1 : B input polarity b2 : Z input polarity b3 : HOME input polarity b4 : +LS input polarity b5 : -LS input polarity Any bit returned 0 is normal polarity and returned 1 is invert polarity.

- **MPC3024AC_encoder_status_read**

Format : **u32 status = MPC3024AC_encoder_status_read(u8 CardID, u8 axis, u16 *state)**

Purpose: To read data from encoder status register

Parameters:

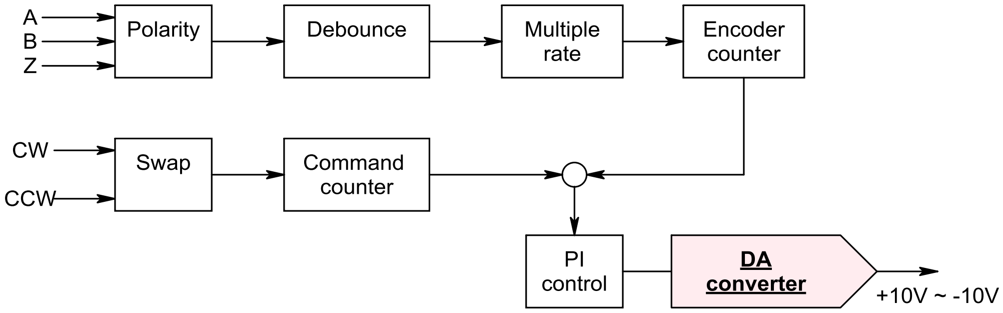
Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
state	u16	b0 : encoder A phase state b1 : encoder B phase state b2 : encoder Z phase state b3 : HOME input point b4 : +LS input point b5 : -LS input point

9.23 DA as simple digital to analog converter (**only for MPC3024AC)



In general the DA's are used as command source of the driver in closed loop PI control. You can also use DA in standalone application, take it as a 17bit -10V to +10V DA to control the device.

Use

MPC3024AC_DA_set( ) to do DA conversion.

MPC3024AC_DA_read( ) to read back the digital command value.

If you use the closed loop motion control, you cannot use the DA as standalone converter, it will automatically controlled under the PI algorithm.

- **MPC3024AC_DA_set**

Format : **u32 status = MPC3024AC_DA_set(u8 CardID, u8 axis, i32 data)**

Purpose: To set MPC3024AC card's DA data (If you do not use the closed loop control).

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
data	i32	Set conversion data. -65535 ~ 65535 (-10V ~ 10V)

Note: In PI mode, it will auto update by PI algorithm.

- **MPC3024AC_DA_read**

Format : **u32 status = MPC3024AC_DA_read(u8 CardID, u8 axis, i32 *data)**

Purpose: To read back data of DA .

Parameters:

Input:

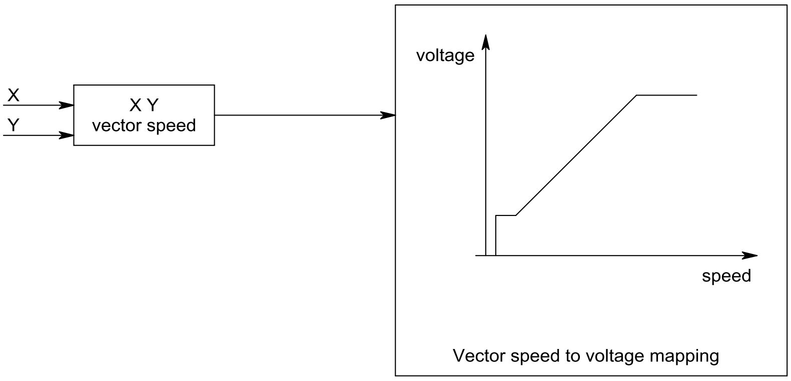
Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

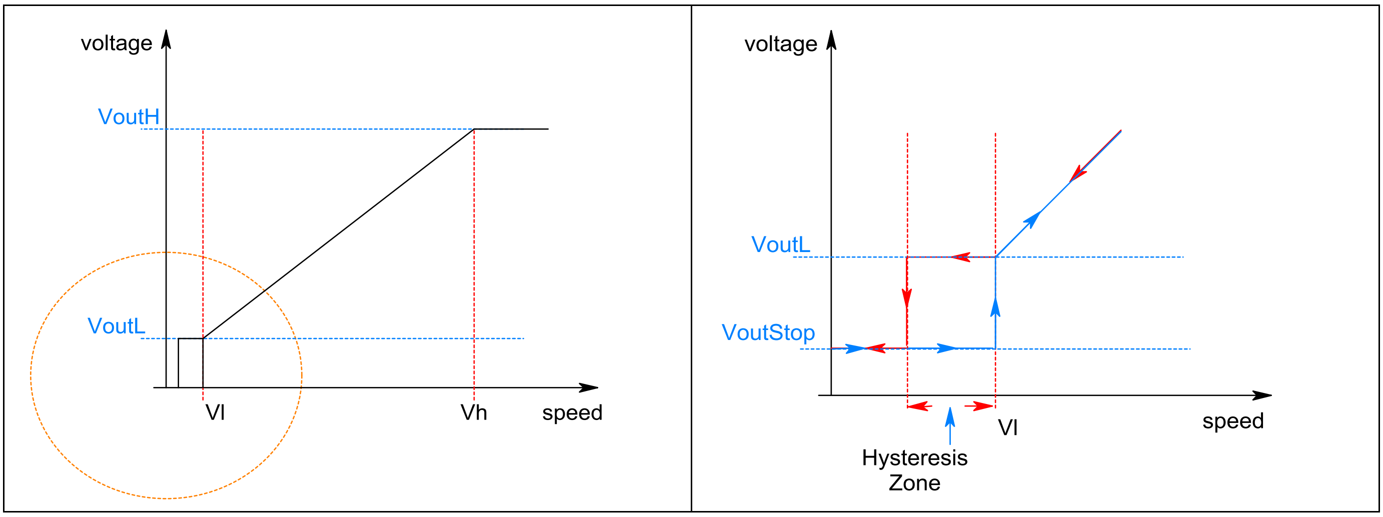
Name	Type	Description
data	i32	Data read back -65535 ~ 65535 (-10V ~ 10V)

9.24 DA as X,Y axes vector speed follower (**only for MPC3024AC)

In the laser contour cutting, normally the laser power must proportional to the cutting speed.



From the above diagram, you can see the DA output voltage follows the X,Y axes vector speed under a predefied profile. Let us take a close look at the profile definition as follows.



From the diagram, we can clearly define the Vl and Vh vector speed of X,Y axes which are the lowest speed and highest speed to map the DA output VoutL and VoutH. VoutStop is the stop speed (maybe zero voltage). From the upper right diagram, the hysteresis zone (maybe zero) can be defined.

DA0 can work as speed follower, setup the configuration by:

MPC3024AC_DA_motion_config_set( ) and read back for verification by

MPC3024AC_DA_motion_config_read( ).

The configuration parameter also includes the trigger mode, if you want to control just by software to start/ stop the speed follower function, set the trigger mode to auto start and control function start/stop by ***MPC3024AC_DA_motion_control_set( )*** and read back for verification by

MPC3024AC_DA_motion_control_read( ).

If you want to start the speed follower function by external trigger, you must set the trigger mode to input trigger then control function start/stop to wait for external trigger to start the function.

- **MPC3024AC_DA_motion_config_set**

Format : **u32 status = MPC3024AC_DA_motion_config_set(u8 CardID , _da_motion_config *config)**

Purpose: Set parameters of X,Y vector speed follower

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by jumper setting

config	_da_motion_config	struct _da_motion_config
		{
		u32 Vh; //Input high speed(pps)
		u32 Vl; //Input low speed(pps)
		u16 VoutH; //Output high voltage
		//(0~65535)
		u16 VoutL; //Output low voltage
		//(0~65535 for 0~10V)
		u16 VoutStop; //Output stop voltage
		//(0~65535 for 0~10V)
		u16 Hysteresis;//Hysteresis range
		//(0~Vl pps)
		u8 trigger_mode;
		//0 : auto start
		//1 : input trigger
		//(LTC (latch) input)
		}

Note: The DA0 is used in speed follower function and only 0-10V is available.

- **MPC3024AC_DA_motion_config_read**

Format : **u32 status = MPC3024AC_DA_motion_config_read(u8 CardID ,**
_da_motion_config *config)

Purpose: read back parameters of X,Y vector speed follower

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by jumper setting

Output:

Name	Type	Description

config	_da_motion_config	struct _da_motion_config
		{
		u32 Vh; //Input high speed(pps)
		u32 Vl; //Input low speed(pps)
		u16 VoutH; //Output high voltage
		//(0~65535)
		u16 VoutL; //Output low voltage
		//(0~65535 for 0~10V)
		u16 VoutStop; //Output stop voltage
		//(0~65535 for 0~10V)
		u16 Hysteresis;//Hysteresis range
		//(0~Vl pps)
		u8 trigger_mode;
		//0 : auto start
		//1 : input trigger
		//(LTC (latch) input)
		}

- **MPC3024AC_DA_motion_control_set**

Format : **u32 status = MPC3024AC_DA_motion_control_set(u8 CardID , u8 control)**

Purpose: to start/ stop speed follower function

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by jumper setting
control	u8	0 : disable 1 : enable

Note: To enable DA motion control function will occupy the on card timer, do not use timer functions while DA motion control enabled.

- **MPC3024AC_DA_motion_control_read**

Format : **u32 status = MPC3024AC_DA_motion_control_read(u8 CardID , u8 *control)**

Purpose: read back setting of start/ stop speed follower function

Parameters:

Input:

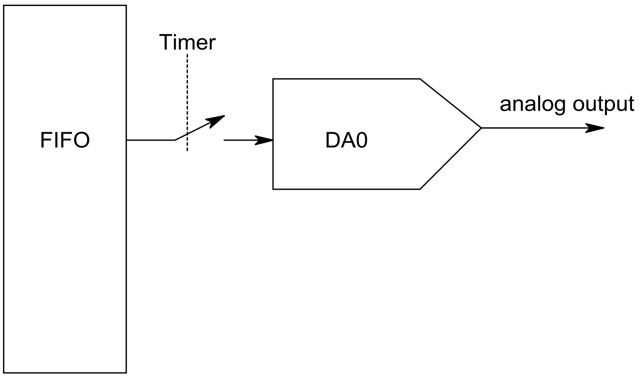
Name	Type	Description
CardID	u8	assigned by jumper setting

Output:

Name	Type	Description
control	u8	0 : disable 1 : enable

9.25 DA as arbitary waveform generator (**only for MPC3024AC)

The DA0 output can work as arbitary waveform generator by scheduling the sample period by timer and filling the FIFO to output to DA in advance.



From the above diagram, FIFO data is sampled by the timer to output data to convert to analog volatge. The working mode may be one cycle mode, it is run upto the last data then halts or repeat the waveform in always run mode. On the stop mode, no matter it is halt by command to stop or the preset stop_da_value.

Using ***MPC3024AC_DA_Arbitrary_Waveform_data_set( )*** to setup the FIFO data for waveform, run mode, stop mode and the preset stop_da_value and sampling time. By ***MPC3024AC_DA_Arbitrary_Waveform_data_read( )*** to read back for verification.

After the data and configuration setup using ***MPC3024AC_DA_Arbitrary_Waveform_control_set( )*** to control start/stop of function and read back for verication by ***MPC3024AC_DA_Arbitrary_Waveform_control_read( )***

- **MPC3024AC_DA_Arbitrary_Waveform_data_set**

Format : u32 status = MPC3024AC_DA_Arbitrary_Waveform_data_set(u8 CardID , _da_data *data , _da_data_config *config)

Purpose: Set DA arbitrary waveform data and configuration.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

data	_da_data	struct _da_data
		{
		u16 da_counter; //valid data number
		u16 da_value[10000]; //0~65535 DA value
		u8 da_sign[10000]; //0: positive voltage
		//1: negative voltage
		}
		Note:
		the maximum data number will be 10,000; each can have polarity (positive or negative voltage) attribute.
		The da_counter indicates the real meaningful data number you fill.

config	_da_data_config	Struct _da_data_config
		{
		u32 timer;
		// DAoutput sample time on
		//time base 1us, timer constant no less than 100
		
		u16 stop_da_value;
		//DA output data for stop_mode=2 (default 0V)
		
		u8 stop_da_sign;
		//polarity of stop_da_value (default 0:positive)
		u8 retrigger;
		//0 : single cycle(default)
		//1 : always run (the last data will continued with
		//the first data.
		u8 stop_mode;
		//0 : halt immediately on current data (default) ,
		//1 : halt until last data then keep at last data.
		//2 : halt immediately on stop_da_value.
		u8 trigger_mode;
		//0 : auto start
		//1 : input trigger(LTC (latch) input)
		
		}

Note:

1. The arbitrary waveform function will use timer0 as function time base. You cannot use timer during the arbitrary waveform function working.
2. Timer constant needs a value no less than 100(us) to avoid system performance lag.
3. DA0 is the dedicated output of waveform.

MPC3024AC_DA_Arbitrary_Waveform_data_read

Format : **u32 status = MPC3024AC_DA_Arbitrary_Waveform_data_read(u8 CardID , _da_data *data , _da_data_config *config)**

Purpose: Read back the DA arbitrary waveform data and configuration.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

Output:

Name	Type	Description

data	_da_data	struct _da_data
		{
		u16 da_counter; //valid data number
		u16 da_value[10000]; //0~65535 DA value
		u8 da_sign[10000]; //0: positive voltage
		//1: negative voltage
		}
		Note:
		the maximum data number will be 10,000; each can have polarity (positive or negative voltage) attribute.
		The da_counter indicates the real meaningful data number you fill.

config	_da_data_config	Struct _da_data_config
		{
		u32 timer;
		// DAoutput sample time on
		//time base 1us, timer constant no less than 100
		u16 stop_da_value;
		//DA output data for stop_mode=2 (default 0V)
		u8 stop_da_sign;
		//polarity of stop_da_value (default 0:positive)
		u8 retrigger;
		//0 : single cycle(default)
		//1 : always run (the last data will continued with
		//the first data.
		u8 stop_mode;
		//0 : halt immediately on current data (default) ,
		//1 : halt until last data then keep at last data.
		//2 : halt immediately on stop_da_value.
		u8 trigger_mode;
		//0 : auto start
		//1 : input trigger(LTC (latch) input)
		}

- **MPC3024AC_DA_Arbitrary_Waveform_control_set**

Format : **u32 status = MPC3024AC_DA_Arbitrary_Waveform_control_set(u8 CardID , u8 control)**

Purpose: DA arbitrary waveform output start/ stop

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by jumper setting
control	u8	0 : disable 1 : enable

Note:

1. DA arbitrary waveform output starts means output the waveform from the first data stored.
2. DA arbitrary waveform output stops will depend on the configuration of stop_mode set.

- **MPC3024AC_DA_Arbitrary_Waveform_control_read**

Format : **u32 status = MPC3024AC_DA_Arbitrary_Waveform_control_read(u8 CardID , u8 *control)**

Purpose: read back setting of DA arbitrary waveform start/ stop

Parameters:

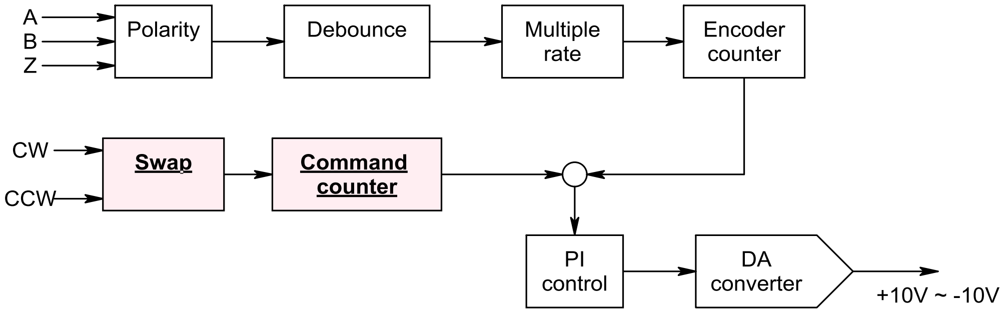
Input:

Name	Type	Description
CardID	u8	assigned by jumper setting

Output:

Name	Type	Description
control	u8	0 : disable 1 : enable

9.26 Pulse command input from internal logic (**only for MPC3024AC)



The command pulse counter is used as reference to compare with the feedback encoder counter. The command pulse is easily swapped to meet the motor direction and encoder. Be sure to adjust the feedback encoder polarity and command input swap to make the closed loop control in convergence else it will go divergence then out of control.

MPC3024AC_pulse_swap_set( ) to swap the input command signal.

MPC3024AC_pulse_swap_read( ) to read back the settings.

- ## MPC3024AC_pulse_swap_set

Format : **u32 status = MPC3024AC_pulse_swap_set(u8 CardID, u8 axis, u8 swap)**

Purpose: To setup command pulse swap register.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
swap	u8	0: normal 1: swap (exchange cw with ccw)

- ## MPC3024AC_pulse_swap_read

Format : **u32 status = MPC3024AC_pulse_swap_read(u8 CardID, u8 axis, u8 *swap)**

Purpose: To read data from command pulse swap register

Parameters:

Input:

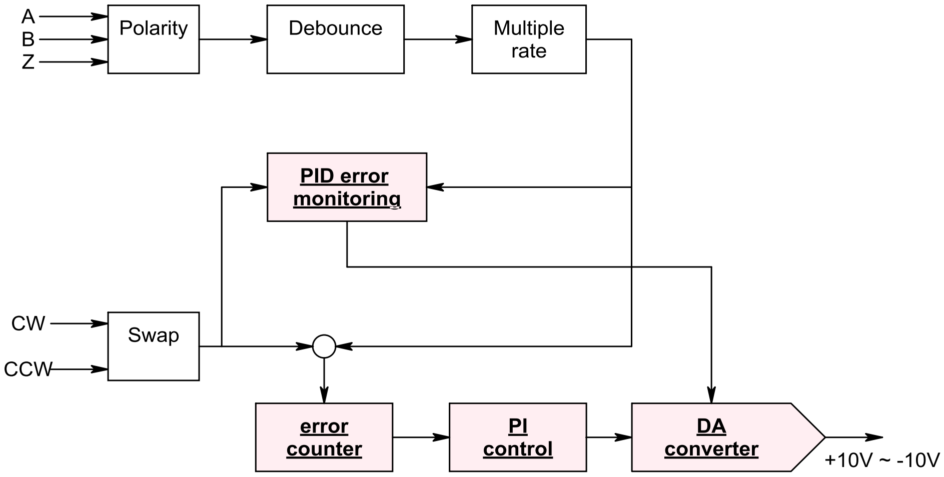
Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
swap	u8	0: normal 1: swap (exchange cw with ccw)

9.27 PI control (**only for MPC3024AC)

To the servo driver, analog voltage comes from the result of PI control register.



Before using the PID control function, you must decide what kind of PID control mode you want. The MPC3024AC provides 5 closed loop control mode:

1. 4 axes independent PI control
2. Tracking mode: Y tracks X and Z, A independent PI control
3. Tracking mode: Y, Z track X and A independent PI control
4. Tracking mode: Y tracks X and A tracks Z
5. Tracking mode: Y, Z, A track X

Depends on your application, define your motion control mode by:

MPC3024AC_PID_control_mode_set( ) and read back to verify by

MPC3024AC_PID_control_mode_read( )

Special Application Tip:

The axis you designated as follower can also use in open loop pulse mode. It is owing to motion command of the follower axis come from the master axis and the pulse generation logic now is spare.

After the mode is defined, you must setup the PI parameters. Using

MPC3024AC_PID_set( ) to set up parameters.

MPC3024AC_PID_read( ) to read back the parameters.

For the PID control mode, it needs the command pulse working in dual pulse mode and the feedback in A/B phase quadrature mode. When you have configure the encoder feedback polarity, multiple rate (ref. 9.3 MPC3024A_pulse_inmode_set( )), the command pulse swap function and confirmed the feedback and command pulse are in the same direction, you can check the feedback loop by: ***MPC3024AC_PID_error_counter_read( )*** to verify the negative feedback of encoder.

If all is in correct configuration, you can close loop the PID control by

MPC3024AC_PID_start( ) to run in analog command mode.

If you do not run anymore,

MPC3024AC_PID_stop( ) to stop the PI control and the DA can used as general purpose DA converter.

To avoid command pulse or encoder feedback broken line, a monitoring hardware is implemented to check the signal integrity, at the setup tuning stage you can disable the function to make the tuning without protection but in normal operation you should enable the error detection function to avoid abnormal of servo motion on signal failure.

MPC3024AC_PID_error_detector_set( ) to enable / disable monitoring function and read back to verify by:

MPC3024AC_PID_error_detector_read( )

Once the error occurs the DA will be cleared and the system halts, you should repair the error to recover the operation. If you do not turn off the computer, you can use

MPC3024AC_PID_error_detector_clear( ) to clear the error register and try again.

For the tracking mode control, you can verify the tracking accuracy by latch the error counter simultaneously and read back the counter to verify.

MPC3024AC_counter_simultaneous_read( ) provide the special function for you to check the performance. If the performance does not meet your requirement, please try to adjust the PI parameters to improve the performance.

- **MPC3024AC_PID_control_mode_set**

Format : **u32 status = MPC3024AC_PID_control_mode_set(u8 CardID, u8 mode)**

Purpose: To set MPC3024AC card’s PID closed loop as independent or tracking mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
mode	u8	0: independent mode (both axes are as master axis) 1: tracking mode1 (X as master and Y as slave, to track the motion) 2: tracking mode2 (X as master and Y as slave and Z as master and A as slave) 3: tracking mode3 (X as master and Y, Z as slave to track the motion) 4: tracking mode4 (X as master and Y, Z, A as slave to track the motion)

- **MPC3024AC_PID_control_mode_read**

Format : **u32 status = MPC3024AC_PID_control_mode_read(u8 CardID, u8 *mode)**

Purpose: To read back MPC3024AC card's PID closed loop as independent or tracking mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Output:

Name	Type	Description
mode	u8	0: independent mode (both axes are as master axis) 1: tracking mode1 (X as master and Y as slave, to track the motion) 2: tracking mode2 (X as master and Y as slave and Z as master and A as slave) 3: tracking mode3 (X as master and Y, Z as slave to track the motion) 4: tracking mode4 (X as master and Y, Z, A as slave to track the motion)

Note: PID control mode and axis relationship on analog and pulse output

		X	Y	Z	A
mode 0	Analog output	master	master	master	master
	Pulse output	master	master	master	master
mode 1	Analog output	master	slave	master	master
	Pulse output	master	master	master	master
mode 2	Analog output	master	slave	master	slave
	Pulse output	master	master	master	master
mode 3	Analog output	master	slave	slave	master
	Pulse output	master	master	master	master
mode 4	Analog output	master	slave	slave	slave
	Pulse output	master	master	master	master

- **MPC3024AC_PID_set**

Format : **u32 status = MPC3024AC_PID_set(u8 CardID, u8 axis, u16 P, u16 I)**

Purpose: To set MPC3024AC card's PID parameters.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
P	u16	P Gain, range 1~4095
I	u16	I Gain, 1~4095 (in millisecond) I Gain=0, no integration function

- **MPC3024AC_PID_read**

Format : **u32 status = MPC3024AC_PID_read(u8 CardID, u8 axis, u16 *P, u16 *I)**

Purpose: To read back MPC3024AC card's PID parameters

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
P	u16	P Gain, range 1~4095
I	u16	I Gain, 1~4096 (in millisecond) I Gain=0, no integration function

- **MPC3024AC_PID_error_counter_read**

Format : **u32 status = MPC3024AC_PID_error_counter_read(u8 CardID, u8 axis, i32 * value)**

Purpose: To read the feedback error data

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
value	i32	error counter data

- **MPC3024AC_PID_start**

Format : **u32 status = MPC3024AC_PID_start(u8 CardID, u8 axis)**

Purpose: Run MPC3024AC card's PID control mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Note:

You must setup the encoder input polarity, multiple rate and confirm the command pulse direction before start PID control.

During the PID control mode, you cannot use MPC3024A_pulse_swap_set( ), MPC3024AC_DA_set( ), MPC3024A_encoder_mode_set( ), MPC3024A_encoder_polarity_set( ).

- **MPC3024AC_PID_stop**

Format : **u32 status = MPC3024AC_PID_stop(u8 CardID, u8 axis)**

Purpose: Stop MPC3024AC card's PID control mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

- **MPC3024AC_PID_error_detector_set**

Format : **u32 status = MPC3024AC_PID_error_detector_set(u8 CardID, u8 axis, u8 enable)**

Purpose: Enable or disable error detect function.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
enable	u8	b0: =1, enable error detector =0, disable error detector

- **MPC3024AC_PID_error_detector_read**

Format : **u32 status = MPC3024AC_PID_error_detector_read(u8 CardID, u8 axis, u8 *state, u8 *enable)**

Purpose: Read back MPC3024AC card's PID error detector status.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
state	u8	b0: =1, encoder A phase error b1: =1, encoder B phase error b3: =1, A or B phase undetermined error b8: =over voltage without pulse command, A or B phase undetermined error
enable	u8	b0: =1, error detector enabled =0, error detector disabled

- **MPC3024AC_PID_error_detector_clear**

Format : **u32 status = MPC3024AC_PID_error_detector_clear(u8 CardID, u8 axis)**

Purpose: Clear error detector to resume error monitoring function.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

- **MPC3024AC_counter_simultaneous_read**

Format : **u32 status = MPC3024AC_counter_simultaneous_read(u8 CardID,
i32 *X_counter, i32 *Y_counter, i32 *Z_counter, i32 *A_counter)**

Purpose: Read back MPC3024AC card’s PID error counters simultaneously.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Output:

Name	Type	Description
X_counter	i32	data of PID error counter of X axis
Y_counter	i32	data of PID error counter of Y axis
Z_counter	i32	data of PID error counter of Z axis
A_counter	i32	data of PID error counter of A axis

9.28 Error conditions

These error types may indicate an internal hardware problem on the board. Chapter 11 MPC3024A Error codes summary contains a detailed listing of the error status returned by MPC3024A functions.

10. Dll list

	Function Name	Description
1.	MPC3024A_initial( )	Initial
2.	MPC3024A_close( )	Close
3.	MPC3024A_init_card( )	Initialize parameters and auxiliary function to default value
4.	MPC3024A_info( )	Get the I/O address and vendor ID of card
5.	MPC3024A_config_file_set( )	Save configuration data to file
6.	MPC3024A_config_file_read( )	Load configuration data from file
7.	MPC3024A_debounce_set( )	Setup motion related input debounce time
8.	MPC3024A_debounce_read( )	Read back motion related input debounce time
9.	MPC3024A_pulse_outmode_set( )	Configure the pulse output mode
10.	MPC3024A_pulse_outmode_read( )	Read back configuration of pulse output mode
11.	MPC3024A_SD_PIN_set( )	Configure slow down input
12.	MPC3024A_SD_PIN_read( )	Read back configuration of SD pin
13.	MPC3024A_EL_mode_set( )	Configure LS(EL) (over travel) stop mode
14.	MPC3024A_EL_mode_read( )	Read back configuration for LS(EL)
15.	MPC3024A_INP_PIN_set( )	Configure INP (in position) input
16.	MPC3024A_INP_PIN_read( )	Read back configuration of INP pin
17.	MPC3024A_ERC_PIN_set( )	Configure ERC (error counter clear) output
18.	MPC3024A_ERC_PIN_read( )	Read back configuration of ERC pin
19.	MPC3024A_ALM_PIN_set( )	Configure ALM (alarm) input
20.	MPC3024A_ALM_PIN_read( )	Read back configuration of ALM pin
21.	MPC3024A_HOME_PIN_logic_set( )	Configure HOME(ORG) polarity
22.	MPC3024A_HOME_PIN_logic_read( )	Read back configuration for HOME pin
23.	MPC3024A_EZ_PIN_logic_set( )	Configure EZ (zero phase) polarity
24.	MPC3024A_EZ_PIN_logic_read( )	Read back configuration of EZ (zero phase) polarity
25.	MPC3024A_LTC_PIN_set( )	Configure LTC (latch) input
26.	MPC3024A_LTC_PIN_read( )	Read back configuration of LTC pin
27.	MPC3024A_CMP_PIN_set( )	Configure CMP (compare) output
28.	MPC3024A_CMP_PIN_read( )	Read back configuration of CMP_OUT
29.	MPC3024A_TTL_IO_mode_set( )	Configure TTL I/O mode
30.	MPC3024A_TTL_IO_mode_read( )	Read back configuration of TTL_IO
31.	MPC3024A_point_set( )	Write point output
32.	MPC3024A_point_read( )	Read point input status
33.	MPC3024A_port_set( )	Write port output
34.	MPC3024A_port_read( )	Read port input status
35.	MPC3024A_velocity_range_fix( )	Set the maximum allowable speed
36.	MPC3024A_velocity_range_unfix( )	Release the limit of maximum allowable speed
37.	MPC3024A_T_velocity_move( )	Velocity mode move at trapezoidal profile
38.	MPC3024A_S_velocity_move( )	Velocity mode move at S curve profile
39.	MPC3024A_velocity_change( )	To change speed on motion
40.	MPC3024A_stop( )	Stop motion immediately or decelerate to stop

41.	MPC3024A_velocity_read( )	Read the current speed
42.	MPC3024A_home_mode_set( )	Select the desired homing mode
43.	MPC3024A_home_start( )	To execute homing
44.	MPC3024A_current_position_set( )	Setup the coordinate of current point
45.	MPC3024A_current_position_read( )	Read the coordinate of current point
46.	MPC3024A_home_search( )	To command origin search mode homing motion
47.	MPC3024A_backlash_comp_set( )	Setup backlash compensation
48.	MPC3024A_backlash_comp_read( )	Read back configuration of backlash compensation
49.	MPC3024A_CSTA_trigger( )	CSTA trigger output
50.	MPC3024A_CSTP_trigger( )	CSTP trigger output
51.	MPC3024A_compare_start_set( )	Setup wait for compare start
52.	MPC3024A_compare_start_read( )	Read back setup of compare start
53.	MPC3024A_compare_start_data_set( )	Setup wait for compare start coordinate data
54.	MPC3024A_compare_start_data_read( )	Read back setup of compare start coordinate data
55.	MPC3024A_compare_start_flag_read( )	Read to verify compare strt flag
56.	MPC3024A_T_position_move( )	Do point to point positioning at trapezoidal profile
57.	MPC3024A_S_position_move( )	Do point to point positioning at S profile
58.	MPC3024A_position_change( )	Change positioning while point to point motion is running
59.	MPC3024A_T_onLINE_change( )	Change the motion parameters on the fly (T profile)
60.	MPC3024A_S_onLINE_change( )	Change the motion parameters on the fly (S profile)
61.	MPC3024A_suppress_vibration_set( )	Setup vibration suppression mode
62.	MPC3024A_suppress_vibration_read( )	Read back parameters of vibration suppression mode
63.	MPC3024A_PCS_PIN_set( )	Configure PCS(position change start) input
64.	MPC3024A_PCS_PIN_read( )	Read back configuration of PCS pin
65.	MPC3024A_PCS_position_override( )	override the target position on the fly
66.	MPC3024A_T_LINE2_move( )	Two axes linear interpolation at trapezoidal profile
67.	MPC3024A_S_LINE2_move( )	Two axes linear interpolation at S curve profile
68.	MPC3024A_T_LINE3_move( )	3 axes linear interpolation at trapezoidal profile
69.	MPC3024A_S_LINE3_move( )	3axes linear interpolation at S curve profile
70.	MPC3024A_T_LINE4_move( )	4 axes linear interpolation at trapezoidal profile
71.	MPC3024A_S_LINE4_move( )	4 axes linear interpolation at S curve profile
72.	MPC3024A_T_LINE_move( )	1~4 axes linear interpolation at trapezoidal profile
73.	MPC3024A_S_LINE_move( )	1~4 axes linear interpolation at S curve profile
74.	MPC3024A_T_ARC_center_move( )	Circular interpolation with the circle center and end position as parameters (T-profile)
75.	MPC3024A_S_ARC_center_move( )	Circular interpolation with the circle center and end position as parameters (S-profile)
76.	MPC3024A_T_CIR_3P_move( )	Circular interpolation with current point and the other 2 points as parameters (T-profile)
77.	MPC3024A_S_CIR_3P_move( )	Circular interpolation with current point and the other 2 points as parameters (S-profile)
78.	MPC3024A_T_ARC_3P_move( )	Circular interpolation with current point and the other 2 points as parameters (T-profile)

79.	MPC3024A_S_ARC_3P_move( )	Circular interpolation with current point and the other 2 points as parameters (S-profile)
80.	MPC3024A_T_CIR_Radius_move( )	Circular interpolation with radius and end position as parameters for circular trajectory (T-profile)
81.	MPC3024A_S_CIR_Radius_move( )	Circular interpolation with radius and end position as parameters for circular trajectory (S-profile)
82.	MPC3024A_T_ARC_Radius_move( )	Circular interpolation with end point and radius as parameters(T-profile)
83.	MPC3024A_S_ARC_Radius_move( )	Circular interpolation with end point and radius as parameters(S-profile)
84.	MPC3024A_T_ArcXY_LineZ_center_move( )	X,Y in circular interpolation with the circle center and end position as parameters and Z linear interpolation to do spiral motion(T-profile)
85.	MPC3024A_S_ArcXY_LineZ_center_move( )	X,Y in circular interpolation with the circle center and end position as parameters and Z linear interpolation to do spiral motion (S-profile)
86.	MPC3024A_T_CirXY_LineZ_3P_move( )	X,Y in circular interpolation with current point and the other 2 points as parameters and Z linear interpolation to do spiral motion (T-profile)
87.	MPC3024A_S_CirXY_LineZ_3P_move( )	X,Y in circular interpolation with current point and the other 2 points as parameters and Z linear interpolation to do spiral motion (S-profile)
88.	MPC3024A_T_ArcXY_LineZ_3P_move( )	X,Y in circular interpolation with current point and the other 2 points as parameters and Z linear interpolation to do spiral motion (T-profile)
89.	MPC3024A_S_ArcXY_LineZ_3P_move( )	X,Y in circular interpolation with current point and the other 2 points as parameters and Z linear interpolation to do spiral motion (S-profile)
90.	MPC3024A_T_CirXY_LineZ_Radius_move( )	X,Y in circular interpolation with end point and radius as parameters and Z linear interpolation to do spiral motion (T-profile)
91.	MPC3024A_S_CirXY_LineZ_Radius_move( )	X,Y in circular interpolation with end point and radius as parameters and Z linear interpolation to do spiral motion (S-profile)
92.	MPC3024A_T_ArcXY_LineZ_Radius_move( )	X,Y in circular interpolation with end point and radius as parameters and Z linear interpolation to do spiral motion (T-profile)
93.	MPC3024A_S_ArcXY_LineZ_Radius_move( )	X,Y in circular interpolation with end point and radius as parameters and Z linear interpolation to do spiral motion (S-profile)
94.	MPC3024A_continuous_flag_set( )	Enable / disable the continuous mode
95.	MPC3024A_continuous_buffer_no_read( )	To read back the remained buffer number
96.	MPC3024A_motion_status_read( )	Read the motion status
97.	MPC3024A_Oneaxis_restart( )	To restart the previously halted axis
98.	MPC3024A_2axis_restart( )	To restart the previously halted 2 axes
99.	MPC3024A_3axis_restart( )	To restart the previously halted 2 axes.
100.	MPC3024A_4axis_restart( )	To restart the previously halted 2 axes
101.	MPC3024A_event_factor_set( )	To enable the event for corresponding event source
102.	MPC3024A_event_flag_read( )	To read the event source
103.	MPC3024A_error_flag_read( )	To read back the status of error source

104.	MPC3024A_softlimit_config_set( )	Configure soft limit
105.	MPC3024A_softlimit_config_read( )	Read back the software limit parameter
106.	MPC3024A_softlimit_data_set( )	Setup the coordinate data of soft limit
107.	MPC3024A_softlimit_data_read( )	Read back the coordinate of software limit
108.	MPC3024A_softlimit_enable_set( )	Enable / disable software limit function
109.	MPC3024A_softlimit_enable_read( )	Read back the status of enable / disable software limit
110.	MPC3024A_softlimit_flag_read( )	Read the software limit flag for verifying
111.	MPC3024A_pulser_mode_set( )	Configure the operating mode of the pulse handler
112.	MPC3024A_pulser_mode_read( )	Read back the pulse handler operation mode
113.	MPC3024A_pulser_counter_set( )	Set pulse counter
114.	MPC3024A_pulser_counter_read( )	Read pulse counter
115.	MPC3024A_pulser_map_set( )	Map the source (pulse handler) to the target motion axis
116.	MPC3024A_pulser_motion_enable( )	Enable pulse handler function and the multiple rate
117.	MPC3024A_pulse_inmode_set( )	Configure the multiple rate and the encoder input
118.	MPC3024A_pulse_inmode_read( )	Read back configuration of pulse input mode
119.	MPC3024A_FB_counter_set( )	Set feedback counter
120.	MPC3024A_FB_counter_read( )	Read feedback counter
121.	MPC3024A_FB_counter_latch_value_read( )	Read feedback counter latched value
122.	MPC3024A_CMP_out_set( )	Configure the compare output mode
123.	MPC3024A_CMP_out_read( )	Read back the configuration of the compare mode
124.	MPC3024A_CMP_data_set( )	Preset the value to the comparator
125.	MPC3024A_CMP_data_read( )	Read back the preset comparator value
126.	MPC3024A_CMP_flag_read( )	Read compare out flag
127.	MPC3024A_IRQ_source_set( )	To setup the error/event source that will generate interrupt at error/event occurs.
128.	MPC3024A_IRQ_status_read( )	To read back the status of interrupt event source
129.	MPC3024A_IRQ_status_clear( )	To clear the status of interrupt event source
130.	MPC3024A_IRQ_mask_set( )	To set the interrupt mask of designated axis.
131.	MPC3024A_IRQ_mask_read( )	To read the interrupt mask of designated axis or timer
132.	MPC3024A_IRQ_process_link( )	Link irq service routine to driver
133.	MPC3024A_IRQ_enable( )	To enable the interrupt function.
134.	MPC3024A_IRQ_disable( )	To disable the interrupt function, and release the resource and close thread.
135.	MPC3024A_timer_set( )	Setup timer parameter
136.	MPC3024A_timer_start( )	Start timer operation
137.	MPC3024A_timer_stop( )	Stop timer operation
138.	MPC3024A_timer_read( )	Read back timer value on the fly
139.	MPC3024A_TC_set( )	Set timer register
140.	MPC3024A_TC_read( )	Read timer register
141.	MPC3024AC_encoder_mode_set( )	Setup PID encoder counter operating mode
142.	MPC3024AC_encoder_mode_read( )	Read back PID encoder counter operating mode
143.	MPC3024AC_encoder_polarity_set( )	Setup PID encoder feedback polarity
144.	MPC3024AC_encoder_polarity_read( )	Read back PID encoder feedback pol
145.	MPC3024AC_encoder_status_read( )	Read back PID encoder status register

146.	MPC3024AC_DA_set( )	Set DA output
147.	MPC3024AC_DA_read( )	Read back DA output data
148.	MPC3024AC_DA_motion_config_set( )	Set parameters of X,Y vector speed follower
149.	MPC3024AC_DA_motion_config_read( )	read back parameters of X,Y vector speed follower
150.	MPC3024AC_DA_motion_control_set( )	tart/ stop speed follower function
151.	MPC3024AC_DA_motion_control_read( )	read back setting of start/ stop speed follower function
152.	MPC3024AC_DA_Arbitrary_Waveform_data_set( )	Set DA arbitrary waveform data and configuration
153.	MPC3024AC_DA_Arbitrary_Waveform_data_read( )	Read back the DA arbitrary waveform data and configuration
154.	MPC3024AC_DA_Arbitrary_Waveform_control_set( )	DA arbitrary waveform output start/ stop
155.	MPC3024AC_DA_Arbitrary_Waveform_control_read( )	read back setting of DA arbitrary waveform start/ stop
156.	MPC3024AC_pulse_swap_set( )	Swap PID pulse command input
157.	MPC3024AC_pulse_swap_read( )	Read back the PID pulse command input swap setting
158.	MPC3024AC_PID_control_mode_set( )	Setup PID as independent or tracking mode
159.	MPC3024AC_PID_control_mode_read( )	Read back PID working mode
160.	MPC3024AC_PID_set( )	Setup PID parameters
161.	MPC3024AC_PID_read( )	Read back PID parameters
162.	MPC3024AC_PID_error_counter_read( )	Read back error counter data
163.	MPC3024AC_PID_start( )	Start to run PID control
164.	MPC3024AC_PID_stop( )	Stop PID control
165.	MPC3024AC_PID_error_detector_set( )	Enable or disable error detect function
166.	MPC3024AC_PID_error_detector_read( )	Read back the error detector register
167.	MPC3024AC_PID_error_detector_clear( )	Clear error detector register and resumes error monitoring function
168.	MPC3024AC_counter_simultaneous_read( )	Read back PID error counters simultaneously

11. MPC3024A Error codes summary

11.1 MPC3024A Error codes table

Error Code	Symbolic Name	Description
0	JSDRV_NO_ERROR	Success, No error.
1	JSDRV_READ_DATA_ERROR	Driver read data error
2	JSDRV_INIT_ERROR	Driver initial error
3	JSDRV_UNLOCK_ERROR	Card is locked, must unlock before operation
6	CHIP_ERROR	Motion chip error
100	DEVICE_RW_ERROR	Device Read/Write error or no card on the system
101	JSDRV_NO_CARD	No MPC3024A card on the system.
102	JSDRV_DUPLICATE_ID	MPC3024A CardID duplicate error.
300	JSMPC_ID_ERROR	Function input parameter error. CardID setting error, CardID doesn’t match the DIP/ROTARY SW setting
301	AXIS_MAX_ERROR	axis parameter error. Parameter out of range.
302	OTHER_PAR_ERROR	Parameter error or out of range.
303	MOTION_BUSY_ERROR	Motion now is busy, no further command can accept
304	CONTINUOUS_FULL_ERROR	In continuous mode, the continuous buffer is full, no further command can accept
305	MOTION_CHANGE_ERROR	Error to use position change in continuous motion mode or motion is already (stop)
306	MOTION_SYNCHROUS_ERROR	Error during interpolation mode, while any of the action axis is error
307	FIFO_FULL_ERROR	Motion FIFO full
308	ARC3P_OVERWRITE2_LINE	It is not possible to use the designated 3 point to locate a circle and force to a line
309	READ_FILE_ERROR	File parameter does not exist or not correct while load or save configuration parameters
310	CIRCLE_ADJUST_ERROR	Circle definition error
311	CIRCLE_OVERWRITE_MIDP	Circle definition error