MPC3042A/3042AL

2-axis Motion Control Card

Software Manual (V3.0)

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	for driver v1.0 up
V1.0->V2.0	Add PI Control function v2.0 up
V2.0->V2.1	Modify the order of the contents (flow chart)
V2.1 ->V2.2	Dll add ***MPC3042A_clear_INT_status( )***
V2.2 ->V3.0	disable the software key function with return value always true

Contents

1. How to install the driver and utilities of MPC3042A/3042AL card

1.1 Install the card

Please follow the following steps to install your new card.

In WinXP/7 and up 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 (..\MPC3042A_AL\Software\WinXP_7\ or if you download from website please execute the file MPC3042A_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.

Note: MPC3042AL share the same driver and dll with MPC3042A but the function of motion DA and digital PI is invalid. MPC3042AL is direct function replacement of MPC3042.

Note: Dll functions named with prefix MPC3042_ is compatible with previous version MPC3042 and new functions will be named with MPC3042A_

2. Where to find the file you need

WinXP/7 and up

The directory will be located at

.. \ JS Automation \MPC3042A\API\   (header files and lib files for VB,VC,BCB,C#)

.. \ JS Automation \MPC3042A\Driver\   (backup copy of MPC3042A drivers)

.. \ JS Automation \MPC3042A\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.

3. About the MPC3042A software

MPC3042A software includes a set of dynamic link library (DLL) and system driver that you can utilize to control the motion card's functions.

Your MPC3042A 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 MPC3042A functions within Windows' operation system environment.

3.1 What you need to get started

To set up and use your MPC3042A software, you need the following:

- MPC3042A software
- MPC3042A hardware

  Main board

  Wiring board (Option)

3.2 Software programming choices

You have several options to choose from when you are programming MPC3042A 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 MPC3042A software.

4. MPC3042A Language support

The MPC3042A software library is a DLL used with WinXP/7 and up. You can use these DLL with any Windows integrating development environment that can call Windows DLLs.

4.1 Building applications with the MPC3042A software library

The MPC3042A function reference topic contains general information about building MPC3042A applications, describes the nature of the MPC3042A files used in building MPC3042A 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.

4.2 MPC3042A Windows libraries

The MPC3042A for Windows function library is a DLL called **MPC3042A.dll**. Since a DLL is used, MPC3042A functions are not linked into the executable files of applications. Only the information about the MPC3042A functions in the MPC3042A 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 MPC3042A functions in MPC3042A.dll.

Header Files and Import Libraries for Different Development Environments		
Language	**Header File**	**Import Library**
Microsoft Visual C/C++	MPC3042A/MPC3042.h	MPC3042AVC.lib
Borland C/C++	MPC3042A/MPC3042.h	MPC3042ABC.lib
Microsoft Visual C#	MPC3042A.cs	
Microsoft Visual Basic	MPC3042A.bas	
Microsoft VB.net	MPC3042A.vb	

Table 1

5. Basic concepts of motion control

5.1 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 (5.1.1) but on the other hand, servo motors can be controlled by analog voltage (5.1.2) or pulse. The un-usual type of control can be through the communication method(5.1.3).

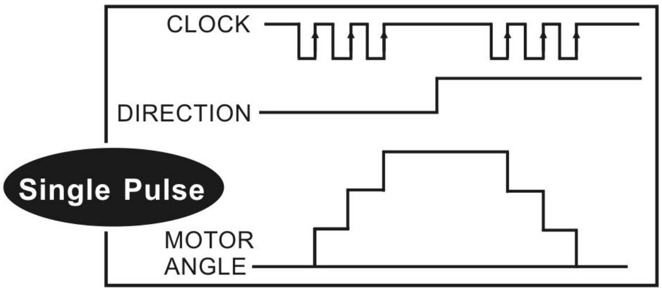
5.1.1 Pulse type motion control

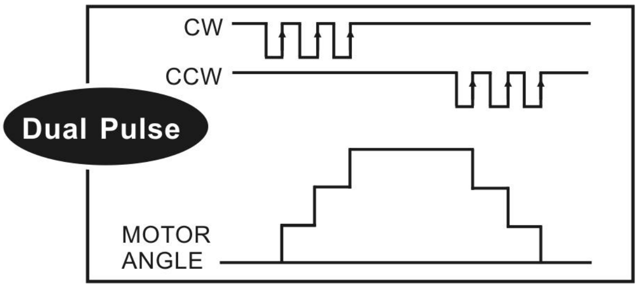
The pulse type motion control was used long ago in step motor control system. In the recent year, a new trend of digital control has moved the servo control from traditional analog control to pulse type motion control.

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.





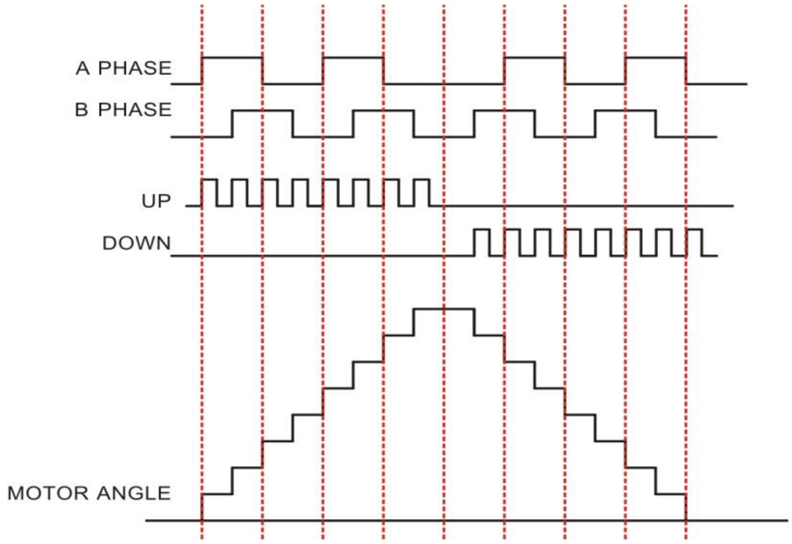
Single pulse type control use only one clock source to control speed and position and the other input is direction control. Dual pulse type control use clockwise clock to control speed and position in one direction and counter-clockwise clock for the other direction.

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.



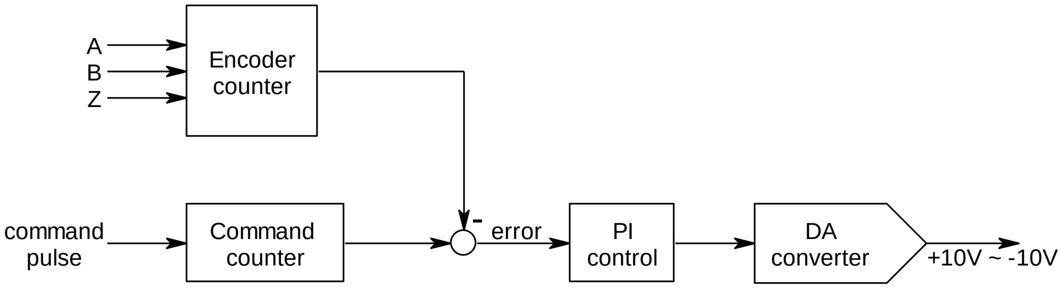
The quadrature A,B phase input also have the direction information encoded, see the above figure, the up and down clock is internally identified by the driver and the motor steps the angle as command input.

5.1.2 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.

MPC3042A 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.

5.1.3 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.

5.2 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.

5.2.1 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.

5.2.2 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.

5.3 Classification of motion control by application

There are 4 major types of application:

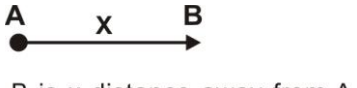
- speed control: controller controls the speed of the servo motor.
- torque control: the controller controls the torque output of the servo motor.
- tracking control: the controller controls the servo motor to follow the motion of another servo motor.
- positioning control: the controller controls the 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 MPC3042A hardware.

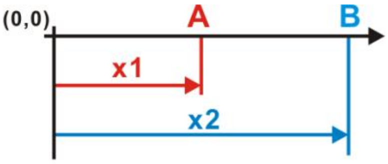
5.4 Coordinate system

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.

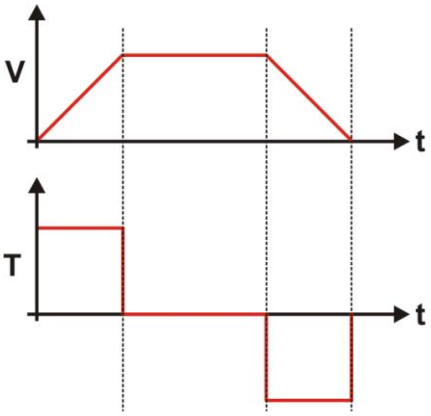


A is at x1 position
B is at x2 position

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

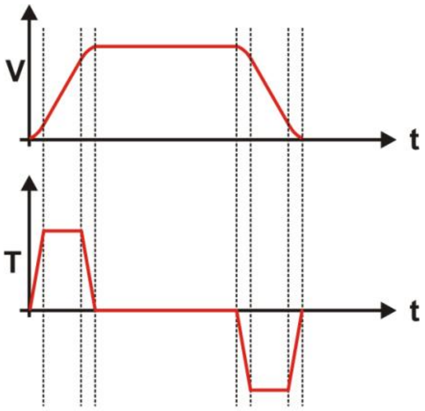
5.5 Motion profile

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

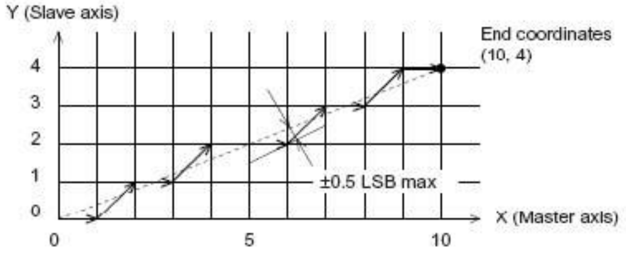
The disadvantage is:
- Requires either twice the acceleration torque or acceleration time for a S profile compared to trapezoidal motion profile

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

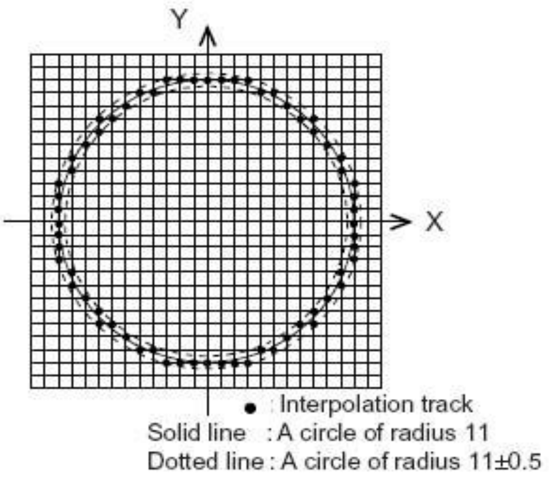
5.6 Interpolation

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.

5.7 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.

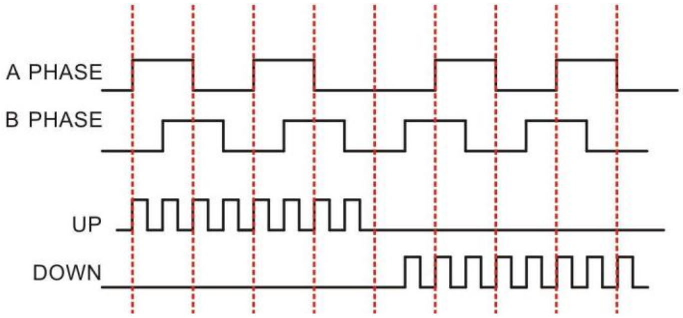
Over-travel limit switch is used under the consideration of ab-normal. If the feedback or other failure that will make the motor run out of control, the over-travel limit switches are put at the extreme position of impossible movement, once it is active, the controller must stop the motion to prevent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

5.8 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.

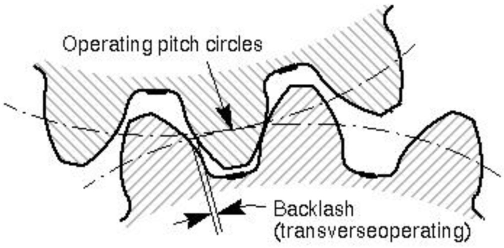
It is a device with 2 phase signals separated at 90 degree. We can discriminate the rotation direction from the phase lead or phase lag. From the following figure, if A lead B, we can decode the up pulses and if B lead A, we also can decode the down pulses.



5.9 Nature of mechanism system

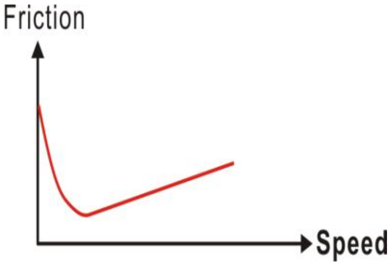
The motion control system is actually a mechtronic system (mechanical + electronics). If you want the system work perfect, you can not overlook the importance of mechanism.

5.9.1 Backlash



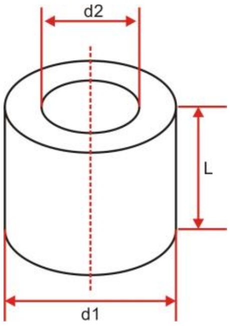
Backlash is the free motion of mechanism when the direction reversed. It is one of the important nature of mechanism. It exist in gear, screw mechanism.

5.9.2 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.

5.9.3 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)

6. Function format and language difference

6.1 Function format

Every MPC 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 MPC function is the parameter **CardID** which is located the driver of MPC board you want to use those given operation. The **CardID** is assigned by DIP/ROTARY switch. 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 switch **(0x0-0xF)**

6.2 Variable data types

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

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

Table 2

6.3 Programming language considerations

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

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

6.3.1 C/C++

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

Status = MPC3042A_DIO_polarity_read(CardID, port, polarity);

//u32 status = MPC3042A_DIO_polarity _read(u8 CardID, u8 port, u8 *polarity);

where **CardID** and **port** are input parameters, and **polarity** is an output parameter. Consider the following example:

```
    u8 CardID=2,port=0;    //card set at ID=2, input port=0
    u8 polarity,
    u32 Status;
…
Status =MPC3042A_DIO_polarity_read(CardID, port, &polarity);
```

6.3.2 Visual basic

The file MPC3042A.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 MPC3042A.bas, do not use the numerical values.

In Visual Basic, you can add the entire MPC3042A.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 MPC3042A.bas file for your project in Visual Basic 4.0, go to the **File** menu and select the **Add File..**. **option.** Select MPC3042A.bas, which is browsed in the ..\MPC3042A\API directory. Then, select **Open** to add the file to the project.

To add the MPC3042A.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** MPC3042A.bas, which is in the ..\MPC3042A\API directory. Then, select **Open** to add the file to the project.

6.3.3 Borland C++ builder

To use Borland C++ builder as development tool, you can use the **file MPC3042ABC.lib** under ..\MPC3042A\API\ or generate **MPC3042ABC.lib** file from the **MPC3042A.dll** file by:

implib MPC3042ABC.lib MPC3042A.dll

Then add the **MPC3042ABC.lib** to your project and add

#include "MPC3042A.h" to main program.

Now you may use the dll functions in your program. For example, the DIO point input function has the following format:

Status = MPC3042A_DIO_polarity_read(CardID, port, polarity);

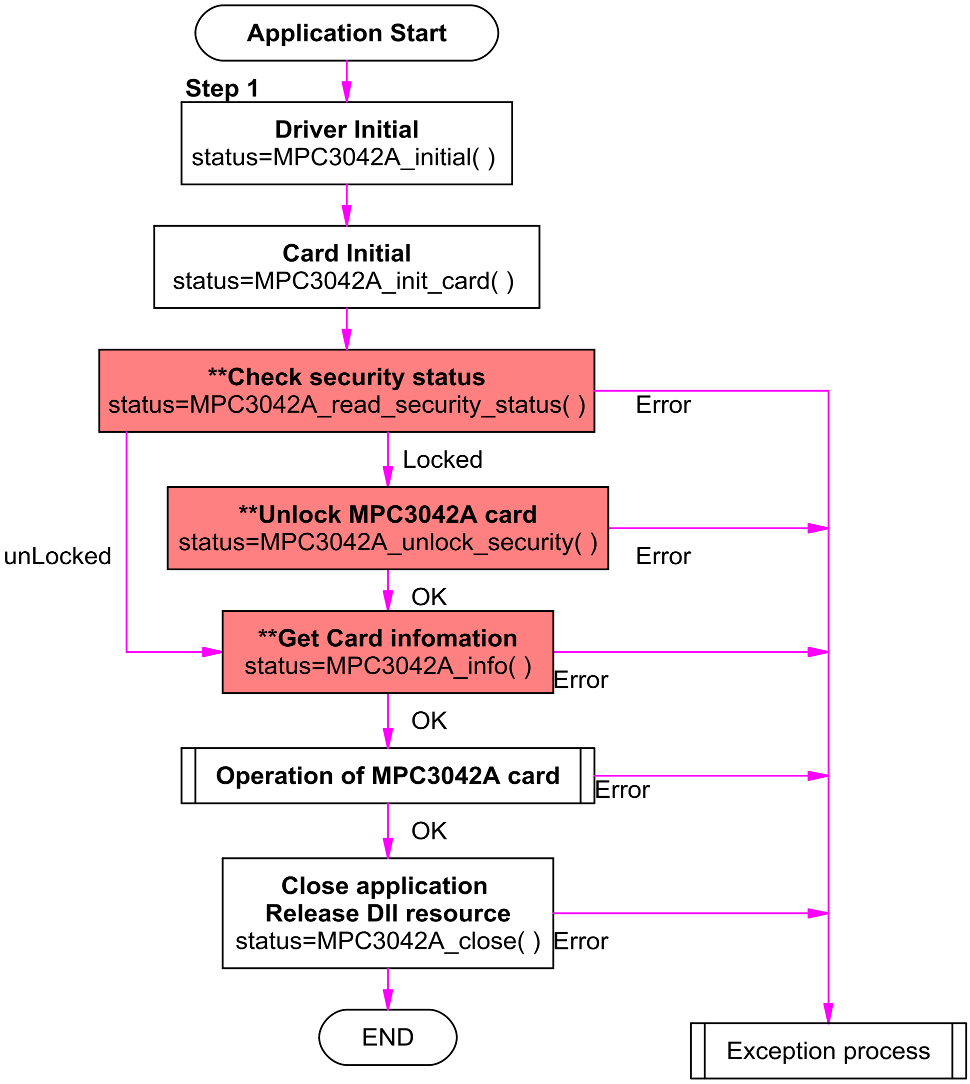
```
        //u32 status = MPC3042A_DIO_polarity_read(u8 CardID, u8 port, u8 *polarity);
```

where **CardID** and **port** are input parameters, and **polarity** is an output parameter. Consider the following example:

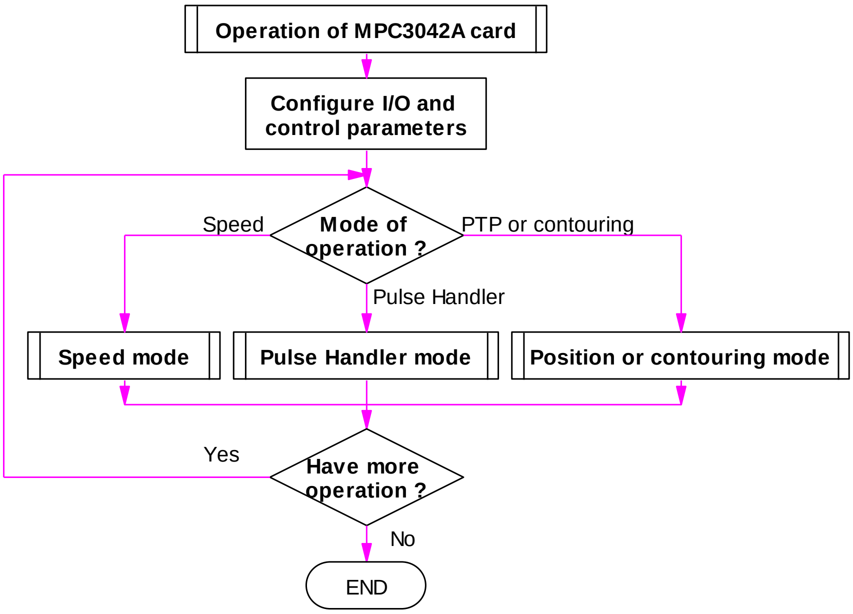
```
    u8 CardID=2,port=0;    //card set at ID=2, input port=0
    u8 polarity,
    u32 Status;
    …
Status = MPC3042A_DIO_polarity_read(CardID, port, &polarity);
```

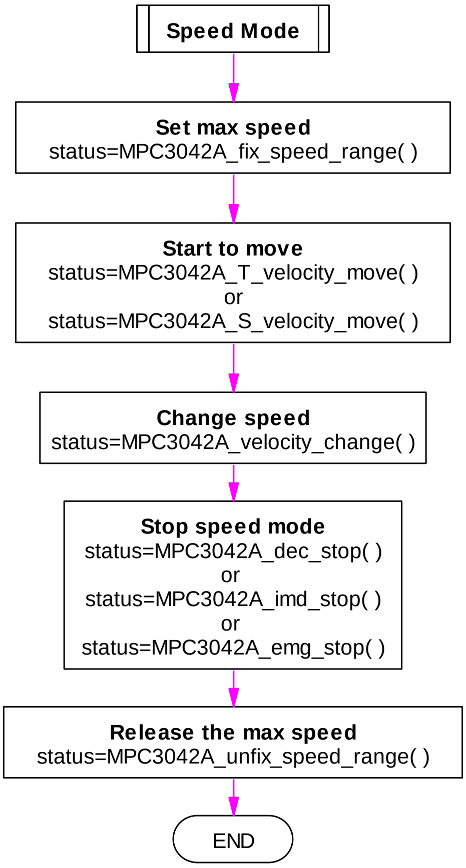
7. Flow chart of application implementation

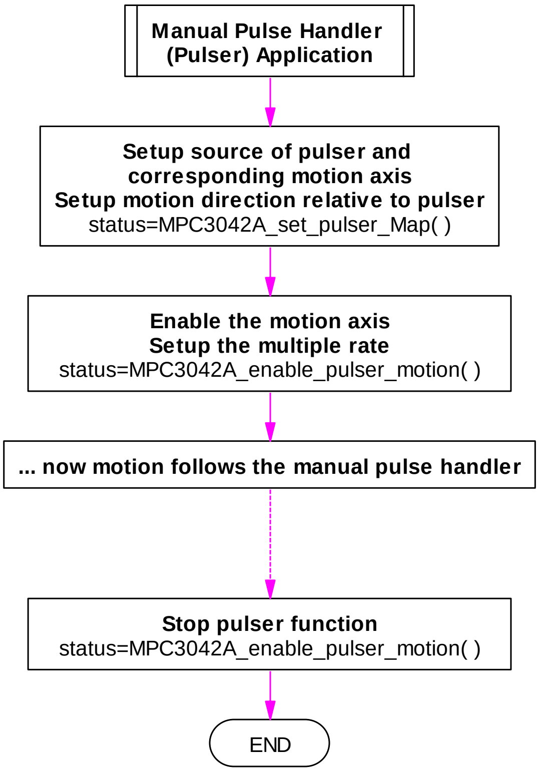
7.1 MPC3042A Flow chart of application implementation

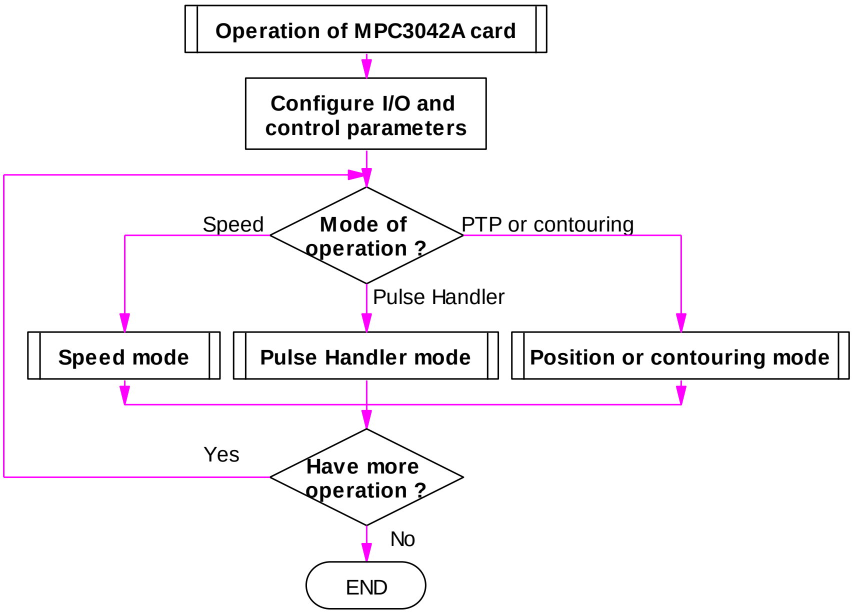


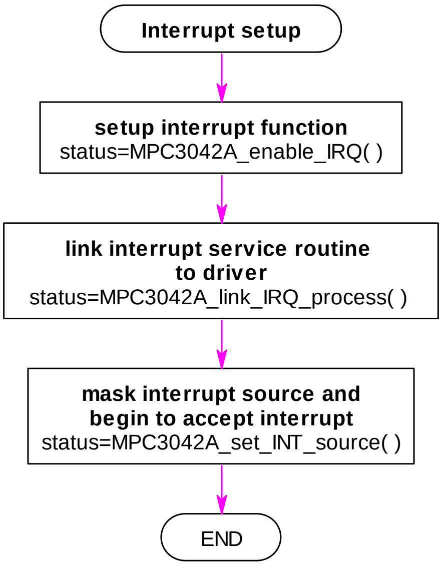
** Security function can be skipped, if you do not use such function.
** If you will use security function, you should setup security first on Demo program or other utility.

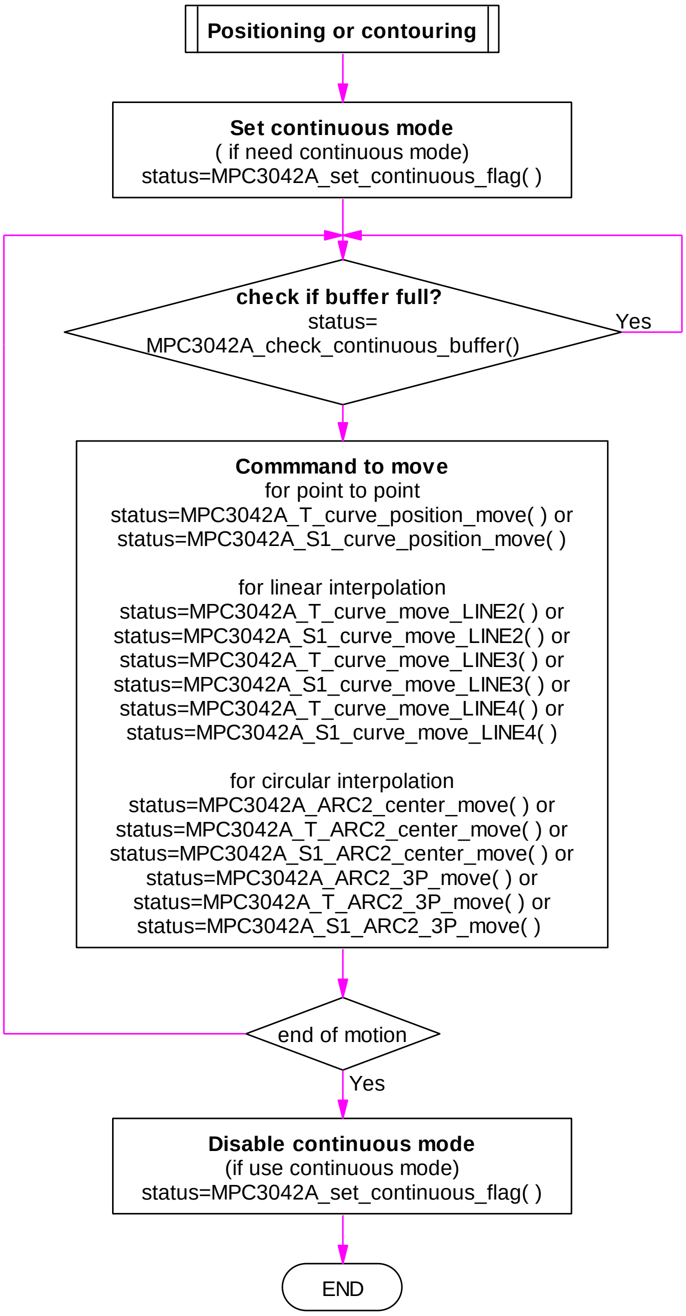

Operation of MPC3042A card
Configure I/O and control parameters
Mode of operation ?
Speed
PTP or contouring
Pulse Handler
Speed mode
Pulse Handler mode
Position or contouring mode
Yes
Have more operation ?
No
END


Speed Mode
Set max speed
status=MPC3042A_fix_speed_range( )
Start to move
status=MPC3042A_T_velocity_move( )
or
status=MPC3042A_S_velocity_move( )
Change speed
status=MPC3042A_velocity_change( )
Stop speed mode
status=MPC3042A_dec_stop( )
or
status=MPC3042A_imd_stop( )
or
status=MPC3042A_emg_stop( )
Release the max speed
status=MPC3042A_unfix_speed_range( )
END


Manual Pulse Handler (Pulser) Application
Setup source of pulser and corresponding motion axis
Setup motion direction relative to pulser
status=MPC3042A_set_pulser_Map( )
Enable the motion axis
Setup the multiple rate
status=MPC3042A_enable_pulser_motion( )
... now motion follows the manual pulse handler
Stop pulser function
status=MPC3042A_enable_pulser_motion( )
END


Operation of MPC3042A card
Configure I/O and control parameters
Mode of operation ?
Speed
PTP or contouring
Pulse Handler
Speed mode
Pulse Handler mode
Position or contouring mode
Have more operation ?
Yes
No
END


Interrupt setup
setup interrupt function
status=MPC3042A_enable_IRQ( )
link interrupt service routine to driver
status=MPC3042A_link_IRQ_process( )
mask interrupt source and begin to accept interrupt
status=MPC3042A_set_INT_source( )
END


Positioning or contouring
Set continuous mode
(if need continuous mode)
status=MPC3042A_set_continuous_flag( )
check if buffer full?
status=
MPC3042A_check_continuous_buffer()
Yes
Commmand to move
for point to point
status=MPC3042A_T_curve_position_move( ) or
status=MPC3042A_S1_curve_position_move( )
for linear interpolation
status=MPC3042A_T_curve_move_LINE2( ) or
status=MPC3042A_S1_curve_move_LINE2( ) or
status=MPC3042A_T_curve_move_LINE3( ) or
status=MPC3042A_S1_curve_move_LINE3( ) or
status=MPC3042A_T_curve_move_LINE4( ) or
status=MPC3042A_S1_curve_move_LINE4( )
for circular interpolation
status=MPC3042A_ARC2_center_move( ) or
status=MPC3042A_T_ARC2_center_move( ) or
status=MPC3042A_S1_ARC2_center_move( ) or
status=MPC3042A_ARC2_3P_move( ) or
status=MPC3042A_T_ARC2_3P_move( ) or
status=MPC3042A_S1_ARC2_3P_move( )
end of motion
Yes
Disable continuous mode
(if use continuous mode)
status=MPC3042A_set_continuous_flag( )
END

8. Software overview and dll function

These topics describe the features and functionality of the MPC3042A boards and describes the details of MPC3042A functions.

8.1 Initialization and close

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

MPC3042A_initial( ) will do.

Once you want to close your application, call

MPC3042A_close( ) to release all the resource.

The motion control card can be initialized at default state by

MPC3042A_init_card( ) or initial to a customized state by configure each function at your will after MPC3042A_initial( ).

To get the physical address assigned by O.S., use

MPC3042A_info( ) will return the address of designated card ID.

- **MPC3042A_initial**

Format : **u32 status =MPC3042A_initial(void)**

Purpose: Initial the MPC3042A resource when start the Windows applications.

- **MPC3042A_close**

Format : **u32 status =MPC3042A_close (void);**

Purpose: Release the MPC3042A resource when close the Windows applications.

- **MPC3042A_init_card**

Format : **u32 status =MPC3042A_init_card(u8 CardID);**

Purpose: To initialize the registers of motion to a factory default state.

Parameters:

Input:

Name	Type	Description
CardID	u8	Card ID assigned by rotary switch

- **MPC3042A_info**

Format : **u32 status =MPC3042A_info(u8 CardID, u8 *CardType, u16 *address);**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

Output:

Name	Type	Description
CardType	u8	0: MPC3042AL 1: MPC3042A 2: MPC3042
address	u16	physical I/O address assigned by OS

8.2 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

MPC3042A_save_config2_file( )

and retrieve to the card by

MPC3042A_load_config_from_file( )

● MPC3042A_save_config2_file

Format : **u32 status = MPC3042A_save_config2_file(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

● MPC3042A_load_config_from_file

Format : **u32 status = MPC3042A_load_config_from_file(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

8.3 Motion related I/O configure and control

Motion related I/O

To meet your servo/step driver, you should first configure the pulse output mode with:

MPC3042A_set_pulse_outmode( )

MPC3042A_readback_pulse_outmode( ) for configuration read back.

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

MPC3042A_set_pulse_inmode( ) will do.

MPC3042A_readback_pulse_inmode( ) 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.

MPC3042A_config_SD_PIN( ) will do.

MPC3042A_readback_SD_PIN( ) for configuration read back.

If your application needs to change position from external trigger during motion period, you should configure PCS(position change start) input by:

MPC3042A_config_PCS_PIN( ).

MPC3042A_readback_PCS_PIN( ) for configuration read back.

If your application needs in position signal to verify if 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

MPC3042A_config_INP_PIN( ) to configure.

MPC3042A_readback_INP_PIN( ) for configuration read back.

In the pulse type control system, servo driver play a 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 remained pulses to move. 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

MPC3042A_config_ERC_PIN( ) to configure your requirement.

MPC3042A_readback_ERC_PIN( ) for configuration read back.

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

MPC3042A_config_ALM_PIN( ) will do.

MPC3042A_readback_ALM_PIN( ) for configuration read back.

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

MPC3042A_config_LTC_PIN( ) to configure the trigger input pin.

MPC3042A_readback_LTC_PIN( ) for configuration read back.

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

MPC3042A_config_CMP_OUT( ).

MPC3042A_readback_CMP_OUT( ) 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

MPC3042A_config_EL_MODE( ).

MPC3042A_readback_EL_MODE( ) for configuration read back.

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

MPC3042A_set_HOME_pin_logic( )

MPC3042A_readback_HOME_pin_logic( ) for configuration read back.

Both the EL (over-travel limit switch) and the ORG (home limit switch) on MPC3042A/AL has new function for input debounce. You can program debounce time at 100Hz, 200Hz ,1Khz or no debounce by:

MPC3042A_debounce_set( ) and read back for verification by:

MPC3042A_debounce_read( )

The encoder input Z-phase is an important reference point of position, the Z phase with the A,B phase at a special relationship defines the encoder original point, the Z phase polarity can be used to change for different type of encoder to meet the motion card requirement.

MPC3042A_set_EZ_pin_logic( ) will do.

MPC3042A_readback_EZ_pin_logic( ) for configuration read back.

- **MPC3042A_set_pulse_outmode**

Format : **u32 status = MPC3042A_set_pulse_outmode(u8 CardID, u8 axis, u8 pulse_outmode)**

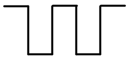
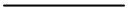
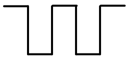
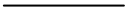
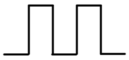
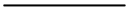
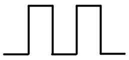
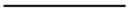
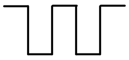
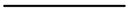
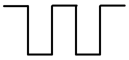
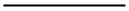
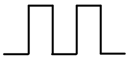
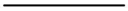
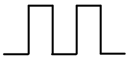
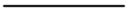
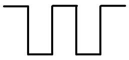
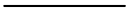
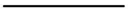
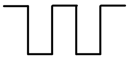
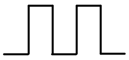
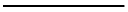
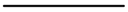
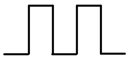
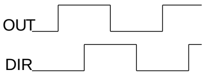
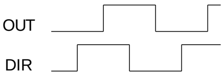
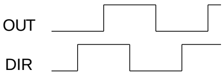
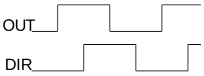
Purpose: Set the pulse output mode for the designated axis.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis
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

- **MPC3042A_readback_pulse_outmode**

Format : **u32 status = MPC3042A_readback_pulse_outmode(u8 CardID,u8 axis, u8* pulse_outmode)**

Purpose: Read back the pulse output mode for the designated axis.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis

Output:

Name	Type	Description
pulse_outmode	u8	0~5 (See Note on pulse out mode)

- **MPC3042A_set_pulse_inmode**

Format : **u32 status = MPC3042A_set_pulse_inmode(u8 CardID, u8 axis, u8 pulse_inmode, u8 count_dir)**

Purpose: To set the encoder input mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis
pulse_inmode	u8	b3~b0: 0:multiply by 1 and up count while phase A lead phase B 1:multiply by 2 and up count while phase A lead phase B 2:multiply by 4 and up count while phase A lead phase B 3:up count while phase A input rising (as CW) down count while rising of phase B input (as CCW) b7~b4: 0: filter out duration less than 1.95us signal, counter bandwidth less than 512K. 1: filter out duration less than 1us signal (default), counter bandwidth less than 1M. 2: filter out duration less than 0.5us signal, counter bandwidth less than 2M. 3: filter out duration less than 0.25us signal, counter bandwidth less than 4M. 4: filter out duration less than 0.125us signal, counter bandwidth less than 8M.
count_dir	u8	0: normal counting 1: reverse counting

- **MPC3042A_readback_pulse_inmode**

Format : **u32 status = MPC3042A_readback_pulse_inmode(u8 CardID, u8 axis, u8 *pulse_inmode, u8 *count_dir)**

Purpose: Read back the parameters of the encoder input mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis

Output:

Name	Type	Description
pulse_inmode	u8	b3~b0: 0:multiply by 1 and up count while phase A lead phase B 1:multiply by 2 and up count while phase A lead phase B 2:multiply by 4 and up count while phase A lead phase B 3:up count while phase A input rising (as CW) down count while rising of phase B input (as CCW) b7~b4: 0: filter out duration less than 1.95us signal, counter bandwidth less than 512K. 1: filter out duration less than 1us signal (default), counter bandwidth less than 1M. 2: filter out duration less than 0.5us signal, counter bandwidth less than 2M. 3: filter out duration less than 0.25us signal, counter bandwidth less than 4M. 4: filter out duration less than 0.125us signal, counter bandwidth less than 8M.
count_dir	u8	0: normal counting 1: reverse counting

- **MPC3042A_config_SD_PIN**

Format : **u32 status = MPC3042A_config_SD_PIN(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 rotary switch
axis	u8	0: X axis 1: Y 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 connect or equal to GND level make this pin active logic. 1: setting the pin floating or equal 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 decelerates to low speed. 1: when SD signal active motion decelerates to stop.

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.

- **MPC3042A_readback_SD_PIN**

Format : **u32 status = MPC3042A_readback_SD_PIN(u8 CardID, u8 axis, u8 *enable, u8 *SD_logic, u8 *SD_latch, u8 *SD_mode, u8 *state)**

Purpose: Read back the configuration of the slow down input and its mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y 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 connect or equal to GND level make this pin active logic. 1: setting the pin floating or equal 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

- **MPC3042A_config_PCS_PIN**

Format : **u32 status = MPC3042A_config_PCS_PIN(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 rotary switch
axis	u8	0: X axis 1: Y 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 equal to GND level make this pin active logic. 1: setting the pin floating or equal to +24v makes this signal active logic.

Note on PCS function:

Name	Description
PCS	PCS pin is external triggered position change function input pin. It can be used to change the target position of motion on the fly.

- **MPC3042A_readback_PCS_PIN**

Format : **u32 status = MPC3042A_readback_PCS_PIN(u8 CradID, u8 axis, u8 *enable, u8 *PCS_logic, u8 *state)**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y 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 equal to GND level make this pin active logic. 1: setting the pin floating or equal to +24v makes this signal active logic.
state	u8	state of PCS pin

- **MPC3042A_config_INP_PIN**

Format : **u32 status = MPC3042A_config_INP_PIN(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 rotary switch
axis	u8	0: X axis 1: Y 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 equal to GND level make this pin active logic. 1: setting the pin floating or equal to +24v makes this signal active logic.

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.

- **MPC3042A_readback_INP_PIN**

Format : **u32 status = MPC3042A_readback_INP_PIN(u8 CardID, u8 axis, u8 *enable, u8 *INP_logic, u8 *state)**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y 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 equal to GND level make this pin active logic. 1: setting the pin floating or equal to +24v makes this signal active logic.
state	u8	state of INP pin

- **MPC3042A_config_ERC_PIN**

Format : **u32 status = MPC3042A_config_ERC_PIN(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 rotary switch
axis	u8	0: X axis 1: Y 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 connectsor equal to GND level make this pin active logic. 1: setting the pin floating or equal 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 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 remained 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 **MPC3042A_write_output_point** to control ERC**,** the active state of ERC will also stop the motion pulses. **Do not use ERC as general output.**

- **MPC3042A_readback_ERC_PIN**

Format : **u32 status = MPC3042A_readback_ERC_PIN(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 rotary switch
axis	u8	0: X axis 1: Y 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 equal to GND level make this pin active logic. 1: setting the pin floating or equal 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

- **MPC3042A_config_ALM_PIN**

Format : **u32 status = MPC3042A_config_ALM_PIN(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 rotary switch
axis	u8	0: X axis 1: Y axis
ALM_logic	u8	0: setting the pin connects or equal to GND level make this pin active logic. 1: setting the pin floating or equal to +24v makes this signal active logic.
ALM_action	u8	0: immediate stop 1: decelerate to stop

- **MPC3042A_readback_ALM_PIN**

Format : **u32 status = MPC3042A_readback_ALM_PIN(u8 CardID, u8 axis, u8 *ALM_logic ,u8 *ALM_action, u8 *state)**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis

Output:

Name	Type	Description
ALM_logic	u8	0: setting the pin connects or equal to GND level make this pin active logic. 1: setting the pin floating or equal to +24v makes this signal active logic.
ALM_action	u8	0: immediate stop 1: decelerate to stop
state	u8	state of ALM pin

- **MPC3042A_config_LTC_PIN**

Format : **u32 status = MPC3042A_config_LTC_PIN(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 rotary switch
axis	u8	0: X axis 1: Y 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 connect or equal to GND level make this pin active logic. 1: setting the pin floating or equal to +24v makes this signal active logic.

- **MPC3042A_readback_LTC_PIN**

Format : **u32 status = MPC3042A_readback_LTC_PIN(u8 CardID, u8 axis, u8 *enable, u8 *LTC_logic, u8 *state)**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y 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 equal to GND level make this pin active logic. 1: setting the pin floating or equal to +24v makes this signal active logic.
state	u8	state of LTC pin

- **MPC3042A_config_CMP_OUT**

Format : **u32 status = MPC3042A_config_CMP_OUT(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 rotary switch
axis	u8	0: X axis 1: Y 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.

- **MPC3042A_readback_CMP_OUT**

Format : **u32 status =MPC3042A_readback_CMP_OUT(u8 CardID, u8 axis,**
u8 *CMP_mode, u8 *state)

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y 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

- **MPC3042A_config_EL_MODE**

Format : **u32 status = MPC3042A_config_EL_MODE(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 rotary switch
axis	u8	0: X axis 1: Y axis
EL_mode	u8	0: immediate stop. 1: decelerate to stop

Note on LS(EL):

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)

- **MPC3042A_readback_EL_MODE**

Format : **u32 status = MPC3042A_readback_EL_MODE(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 rotary switch
axis	u8	0: X axis 1: Y axis

Output:

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

- **MPC3042A_set_HOME_pin_logic**

Format : **u32 status = MPC3042A_set_HOME_pin_logic(u8 CardID, u8 axis, u8 HOME_logic)**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y
HOME_logic	u8	0: setting the pin connects or equal to GND level make this pin active logic. 1: setting the pin floating or equal to +24v makes this signal active logic.

- **MPC3042A_readback_HOME_pin_logic**

Format : **u32 status = MPC3042A_readback_HOME_pin_logic(u8 CardID, u8 axis, u8 *HOME_logic)**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis

Output:

Name	Type	Description
HOME_logic	u8	0: setting the pin connects or equal to GND level make this pin active logic. 1: setting the pin floating or equal to +24v makes this signal active logic.

- **MPC3042A_debounce_set**

Format : **u32 status = MPC3042A_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
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).**

- **MPC3042A_debounce_read**

Format : **u32 status = MPC3042A_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

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).**

- **MPC3042A_set_EZ_pin_logic**

Format : **u32 status = MPC3042A_set_EZ_pin_logic(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 rotary switch
axis	u8	0: X axis 1: Y axis
EZ_logic	u8	0: setting the pin connects or equal to GND level make this pin active logic. 1: setting the pin floating or equal to +5V makes this signal active logic.

- **MPC3042A_readback_EZ_pin_logic**

Format : **u32 status = MPC3042A_readback_EZ_pin_logic(u8 CardID, u8 axis, u8 *EZ_logic)**

Purpose: To read back configuration of the EZ (Encoder Zero phase) logic.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis

Output:

Name	Type	Description
EZ_logic	u8	0: setting the pin connect or equal to GND level make this pin active logic. 1: setting the pin floating or equal to +5V makes this signal active logic.

TTL I/O and output control

The TTL I/O on JM4 and photo-isolated I/O on ADP9201_JM1

- **MPC3042A_write_output_point**

Format : **u32 status = MPC3042A_write_output_point(u8 CardID, u8 axis, u8 point_factor,u8 on_off)**

Purpose: To set/reset output.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis
point_factor	u8	0: ERC (servo error counter clear output) 1: SVON (servo on output) 2: FIN (finish output) 3: CMP (compare equal output) 4: OUT0 (Photo-isolated output bit0 status) 5: OUT1 (Photo-isolated output bit1 status) 6: OUT2 (Photo-isolated output bit2 status) 7: OUT3 (Photo-isolated output bit3 status) 8: OUT4 (Photo-isolated output bit4 status) 9: OUT5 (Photo-isolated output bit5 status) 10: OUT6 (Photo-isolated output bit6 status) 11: OUT7 (Photo-isolated output bit7 status)
on_off	u8	0: reset, inactive 1: set, active

Note on some output:

Name	Description
ERC	Ref. **Note on ERC function** MPC3042A_config_ERC_PIN
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. **Note on CMP function** MPC3042A_config_CMP_OUT

MPC3042A_read_point_status

Format : **u32 status = MPC3042A_read_point_status(u8 CardID, u8 axis, u8 check_factor, u8 *state)**

Purpose: To input status.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by rotary switch	
axis	u8	0: X axis	1: Y axis
check_factor	u8	0: SD	(Slow Down input)
		1: PCS	(Position change start input)
		2: INP	(In position input)
		3: ALM	(servo driver alarm input)
		4: SRDY	(servo driver ready input)
		5: LS+(EL+)	(positive side over travel limit switch)
		6: LS-(EL-)	(negative side over travel limit switch)
		7: LTC	(external latch trigger input)
		8: HOME(ORG)	(home(ORG) sensor input)
		9: EMG	(emergency input)
		10: EZ	(encoder zero phase input)
		11: ERC	(error counter output status)
		12: SVON	(servo driver on output status)
		13: FIN	(finish output)
		14: CMP	(compare equal output)
		15: STA	(start input)
		16: not available	
		17: DI0	(JM4 TTL input DI0 status)
		18: DI1	(JM4 TTL input DI1 status)
		19: DI2	(JM4 TTL input DI2 status)
		20: DI3	(JM4 TTL input DI3 status)
		21: DI4	(JM4 TTL input DI4 status)
		22: DI5	(JM4 TTL input DI5 status)
		23: DI6	(JM4 TTL input DI6 status)
		24: DI7	(JM4 TTL input DI7 status)
			The followings on ADP9201_JM1
		25: IN0	(Photo-isolated input bit0)
		26: IN1	(Photo-isolated input bit1)
		27: IN2	(Photo-isolated input bit2)
		28: IN3	(Photo-isolated input bit3)
		29: IN4	(Photo-isolated input bit4)
		30: IN5	(Photo-isolated input bit5)
		31: IN6	(Photo-isolated input bit6)
		32: IN7	(Photo-isolated input bit7)

		33: OUT0	(Photo-isolated output bit0)
		34: OUT1	(Photo-isolated output bit1)
		35: OUT2	(Photo-isolated output bit2)
		36: OUT3	(Photo-isolated output bit3)
		37: OUT4	(Photo-isolated output bit4)
		38: OUT5	(Photo-isolated output bit5)
		39: OUT6	(Photo-isolated output bit6)
		40: OUT7	(Photo-isolated output bit7)

Output:

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

Note: For the general purpose input and output, the axis parameter is of no use.

8.4 Digital I/O port

For the isolated digital I/O port on ADP9201_JM1, the input or output polarity can be set by:

MPC3042A_DIO_polarity_set( ) to set digital input/output port logic polarity, and read back setting by:

MPC3042A_DIO_polarity_read( ).

To eliminate the input noise, debounce filter is a good solution. MPC3042A card provides software input debounce circuit, before using the digital input, selecting an adequate filter frequency by:

MPC3042A_DIO_debounce_set( ) and read back setting by

MPC3042A_DIO_debounce_read( ).

The digital I/O can be set/reset or verified by:

MPC3042A_DIO_set( ) to set/reset a port;

MPC3042A_DIO_read( ) to read back for verification.

MPC3042A_DIO_bit_set( ) to set/reset a specific bit;

MPC3042A_DIO_bit_read( ) to read back a specific bit for verification.

- **MPC3042A_DIO_polarity_set**

Format : **u32 status = MPC3042A_DIO_polarity_set(u8 CardID, u8 port, u8 polarity);**

Purpose: To set the polarity of photo-isolated ports.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
port	u8	0: input 1: output
polarity	u8	0: normal 1: invert

- **MPC3042A_DIO_polarity_read**

Format : **u32 status = MPC3042A_DIO_polarity_read(u8 CardID, u8 port, u8*polarity);**

Purpose: To read back the polarity of photo-isolated ports.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
port	u8	0: input 1: output

Output:

Name	Type	Description
polarity	u8	0: normal 1:invert

- **MPC3042A_DIO_debounce_set**

Format : **u32 status = MPC3042A_DIO_debounce_set(u8 CardID, u8 debounce_time);**

Purpose: To set the debounce time of photo-isolated input ports.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
debounce_time	u8	0: no debounce 1: 100 hz(default), filter out less than 10ms glitch 2: 200 hz, filter out less than 5ms glitch 3: 1khz, filter out less than 1ms glitch

- **MPC3042A_DIO_debounce_read**

Format : **u32 status = MPC3042A_DIO_debounce_read(u8 CardID, u8*debounce_time);**

Purpose: To read back debounce time of photo-isolated input ports.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

Output:

Name	Type	Description
debounce_time	u8	0: no debounce 1: 100 hz(default), filter out less than 10ms glitch 2: 200 hz, filter out less than 5ms glitch 3: 1khz, filter out less than 1ms glitch

- **MPC3042A_DIO_set**

Format : **u32 status = MPC3042A_DIO_set(u8 CardID, u8 data);**

Purpose: To set the polarity of photo-isolated ports.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
data	u8	b7~b0: b0: 0, reset OUT0 1, set OUT0 … b7: 0, reset OUT7 1, set OUT7

- **MPC3042A_DIO_read**

Format : **u32 status = MPC3042A_DIO_read(u8 CardID, u8 port, u8 *data);**

Purpose: To read back the polarity of photo-isolated ports.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
port	u8	0: input 1: output

Output:

Name	Type	Description
data	u8	b7~b0: (depends on output or input) b0: 0, reset OUT0 or IN0 1, set OUT0 or IN0 … b7: 0, reset OUT7 or IN7 1, set OUT7 or IN7

- **MPC3042A_DIO_bit_set**

Format : **u32 status = MPC3042A_DIO_bit_set(u8 CardID, u8 bit, u8 data);**

Purpose: To set the output bit of photo-isolated DIO.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
bit	u8	7: OUT7 … 0: OUT0
data	u8	0: reset 1: set

- **MPC3042A_DIO_bit_read**

Format : **u32 status = MPC3042A_DIO_bit_read(u8 CardID, u8 port, u8 bit, u8 *data);**

Purpose: To read back the bit data of photo-isolated DIO.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
port	u8	0: input 1: output
bit	u8	7: OUT7 or IN7 (depends on output or input) … 0: OUT0 or IN0

Output:

Name	Type	Description
data	u8	0: reset 1: set

8.5 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 ***MPC3042A_config_EL_MODE*** 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 ***MPC3042A_set_pulse_outmode***), you can also have a unsuspected movement.

Velocity motion control is one of the functions of MPC3042A card. For safety reason or others to set the maximum speed is recommended. Use

MPC3042A_fix_speed_range( ) to set the maximum allowable speed.

MPC3042A_unfix_speed_range( ) to release the limit.

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

MPC3042A_T_velocity_move( ) to move at trapezoidal profile.

MPC3042A_S_velocity_move( ) or

MPC3042A_S1_velocity_move( ) to move at S curve profile. (S1 profile is defined by Tsacc)

If you want to change speed or stop it, use

MPC3042A_velocity_change( ) to change speed.

For any axis you want to stop,

MPC3042A_dec_stop( ) to have a deceleration to stop.

MPC3042A_imd_stop( ) to stop immediately.

Use

MPC3042A_emg_stop( ) to stop all the axes immediately

To verify the speed use:

MPC3042A_read_speed( ) will give you the current speed.

- **MPC3042A_fix_speed_range**

Format : **u32 status = MPC3042A_fix_speed_range(u8 CardID,u8 axis,i32 Vmax)**

Purpose: To set the maximum allowable speed.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis
Vmax	i32	max pps (0~6553500)

MPC3042A_unfix_speed_range

Format : **u32 status = MPC3042A_unfix_speed_range(u8 CardID,u8 axis)**

Purpose: To release the maximum allowable speed.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis

MPC3042A_T_velocity_move

Format : **u32 status = MPC3042A_T_velocity_move(u8 CardID,u8 axis,i32 VL,i32 VH, f64 Tacc)**

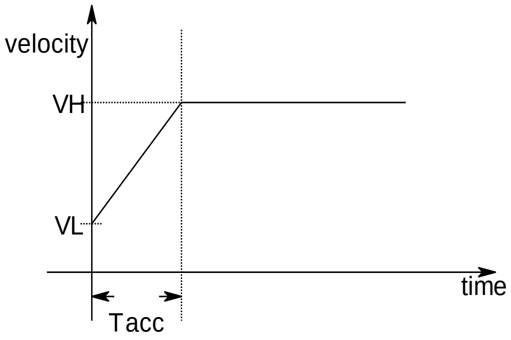
Purpose: Doing velocity mode movement at trapezoidal profile.

Parameters:

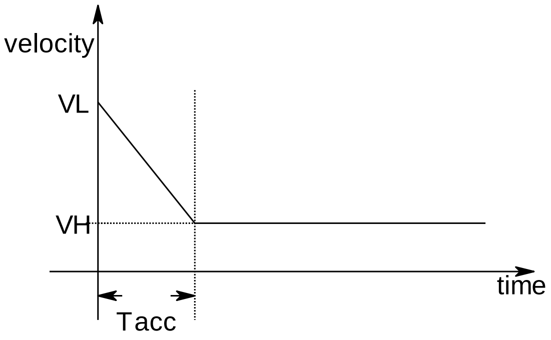
Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis
VL	i32	pps, -6553500~6553500, negative value for reverse direction
VH	i32	pps, -6553500~6553500 negative value for reverse direction
Tacc	f64	acc time in seconds

Note on trapezoidal velocity mode:



Profile for VH > VL



Profile for VH < VL

MPC3042A_S_velocity_move

Format : **u32 status = MPC3042A_S_velocity_move(u8 CardID,u8 axis,i32 VL,i32 VH, f64 Tacc,u32 SVacc)**

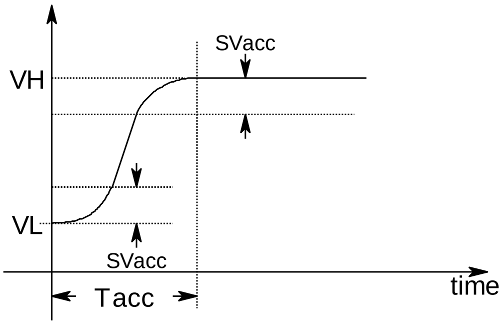
Purpose: Doing velocity mode movement at S curve profile.

Parameters:

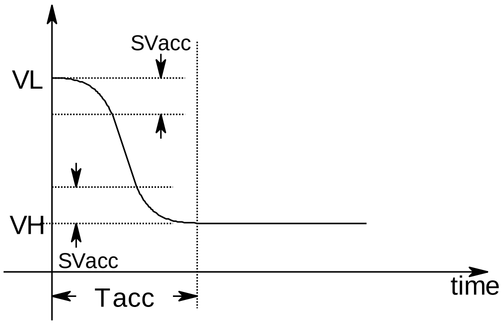
Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis
VL	i32	pps, -6553500~6553500 negative value for reverse direction
VH	i32	pps, -6553500~6553500 negative value for reverse direction
Tacc	f64	seconds
Svacc	u32	frequency difference of s curve range, 0 ≤ Svacc ≤ 0.5(VH-VL)

Note on S curve velocity mode:



Profile for VH > VL



Profile for VH < VL

MPC3042A_S1_velocity_move

Format : **u32 status = MPC3042A_S1_velocity_move(u8 CardID,u8 axis,i32 VL,i32 VH, u32 Tacc_ms,u32 Tsacc_ms)**

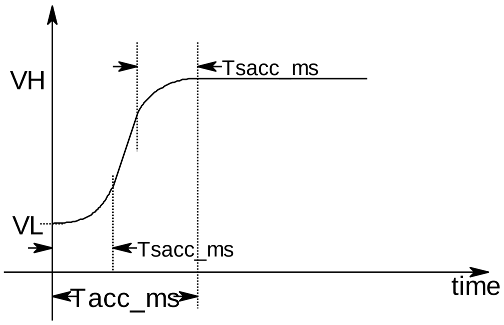
Purpose: Doing velocity mode movement at S curve profile.

Parameters:

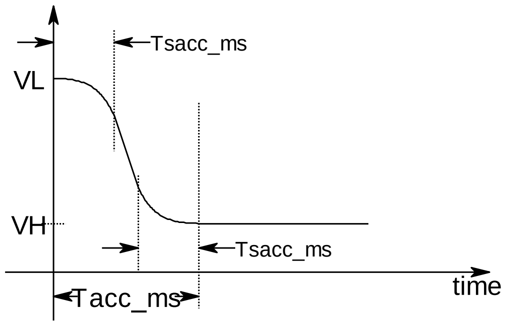
Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis
VL	i32	pps, -6553500~6553500 negative value for reverse direction
VH	i32	pps, -6553500~6553500 negative value for reverse direction
Tacc_ms	u32	mili-seconds
Tsacc_ms	u32	mili-seconds

Note on S curve velocity mode:



Profile for VH > VL



Profile for VH < VL

MPC3042A_velocity_change

Format : **u32 status = MPC3042A_velocity_change(u8 CardID,u8 axis,i32 Vn,f64 Tacc)**

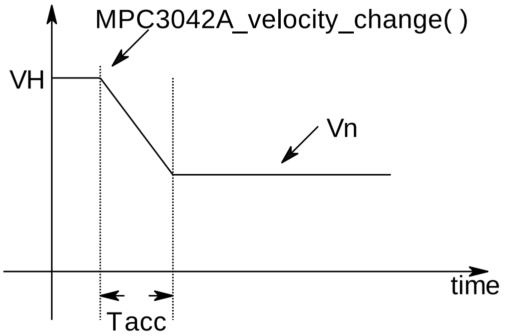
Purpose: Change speed (with the trapezoidal/S curve mode previously defined) at velocity mode.

Parameters:

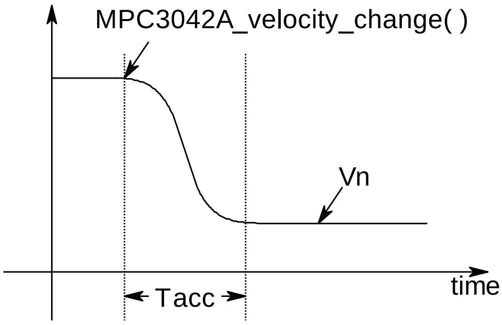
Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis
Vn	i32	new speed in pps, -6553500~6553500 |Vn| ≤ Vmax (set by MPC3042A_fix_speed_range( ))
Tacc	f64	acc time in seconds

Note on velocity change:



Velocity change at trapzoidal acc/dec



Velocity change at S curve acc/dec

Note:

If you use MPC3042A_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 MPC3042A_S_velocity_move and MPC3042A_T_velocity_move are no need to switch to zero speed.

- **MPC3042A_dec_stop**

Format : **u32 status = MPC3042A_dec_stop(u8 CardID,u8 axis,f64 Tdec)**

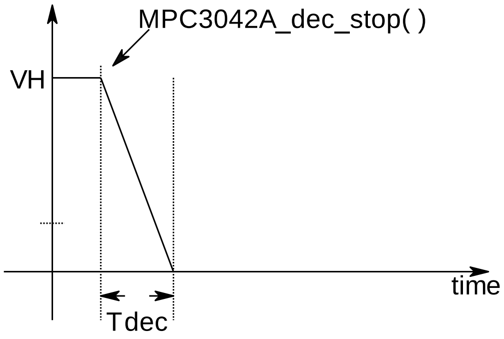
Purpose: Command to decelerate to stop (with the trapezoidal/S curve mode previously defined) at velocity mode.

Parameters:

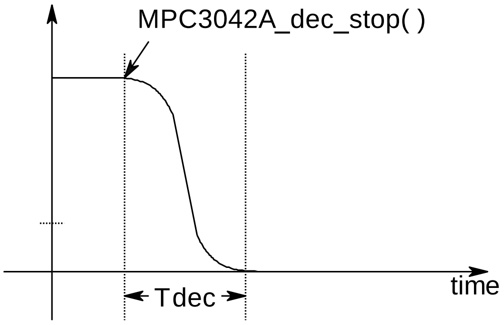
Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis
Tdec	f64	dec time in seconds

Note on decelerate to stop:



Dec stop at trapzoidal acc/dec



Dec stop at S curve acc/dec

- **MPC3042A_imd_stop**

Format : **u32 status = MPC3042A_imd_stop(u8 CardID,u8 axis)**

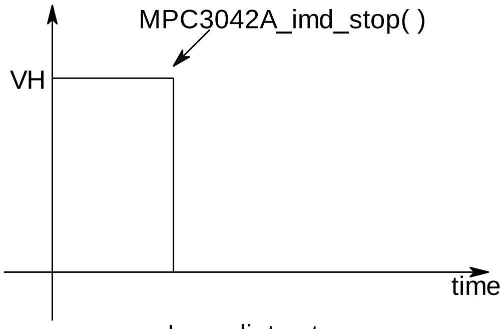
Purpose: Command to immediate stop.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis

Note on immediate stop:



Immediate stop

- **MPC3042A_emg_stop**

Format : **u32 status = MPC3042A_emg_stop(u8 CardID)**

Purpose: Command to emgerency stop. (i.e. All axes immediately stop)

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch

- **MPC3042A_read_speed**

Format : **u32 status = MPC3042A_read_speed(u8 CardID,u8 axis,f64 *speed)**

Purpose: To read the current speed.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis

Output:

Name	Type	Description
speed	f64	current speed in pps

8.6 Homing

Refer 7.5 Velocity mode motion, Prepare for motion control to setup the pre-requisit conditions.

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

Use

MPC3042A_config_home_mode( ) to select the desired homing mode.

MPC3042A_start_homing( ) to execute homing.

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

MPC3042A_set_current_position( ) to setup the coordinate at any time and any point, if the motion is ready.

Any time, you want to get the coordinate,

MPC3042A_read_current_position( ) will do.

MPC3042A_start_origin_search_homing( ) will seek home (ORG) limit switch automatically and correct the position.

- **MPC3042A_config_home_mode**

Format : **u32 status = MPC3042A_config_home_mode(u8 CardID,u8 axis,u8 mode, u8 EZ_count)**

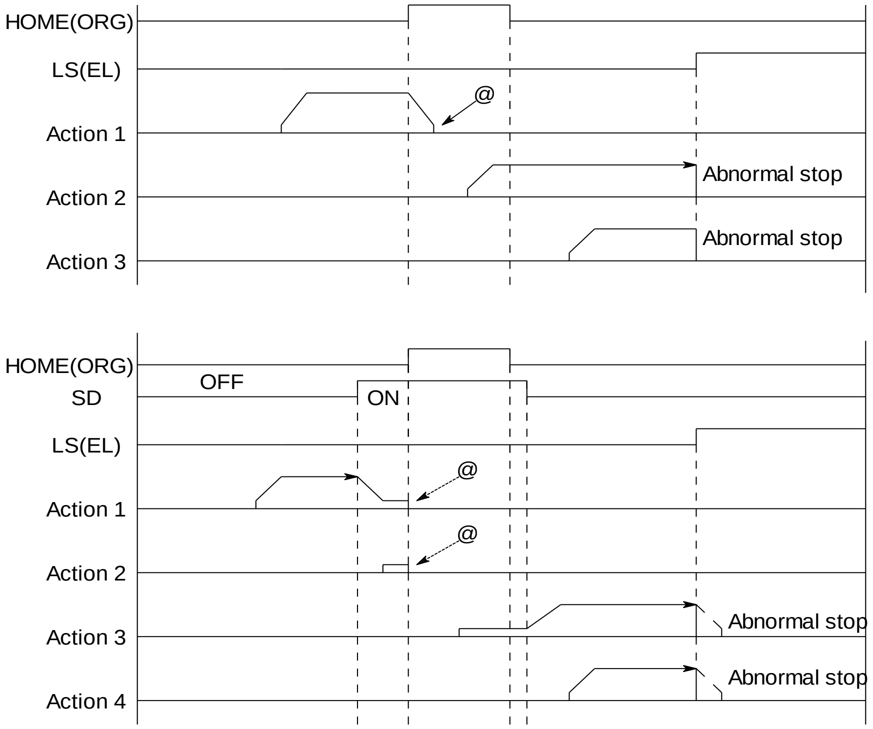
Purpose: To configure the homing mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis
mode	u8	homing mode 0~12$_{(10)}$
EZ_count	u8	Homed position after the pulse numbers of zero input while home (ORG) switch activated. 0~15$_{(10)}$ The counter number is the EZ_count value+1.

Mode 0 Description



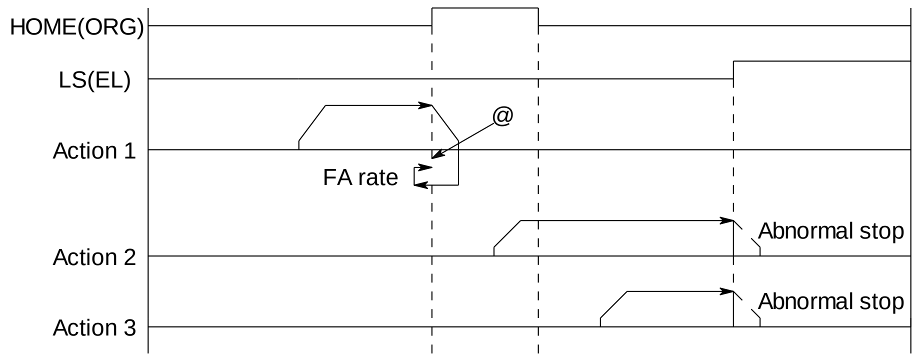
Mode0:

1.The motion will begin to decelerate to VL then stop on the input signal of HOME(ORG) signal turning from OFF to ON.

2. The current position counter will reset upon HOME(ORG) signal turning from OFF to ON transition. At the stop position, the counter value maybe not “zero”.

3.@ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.

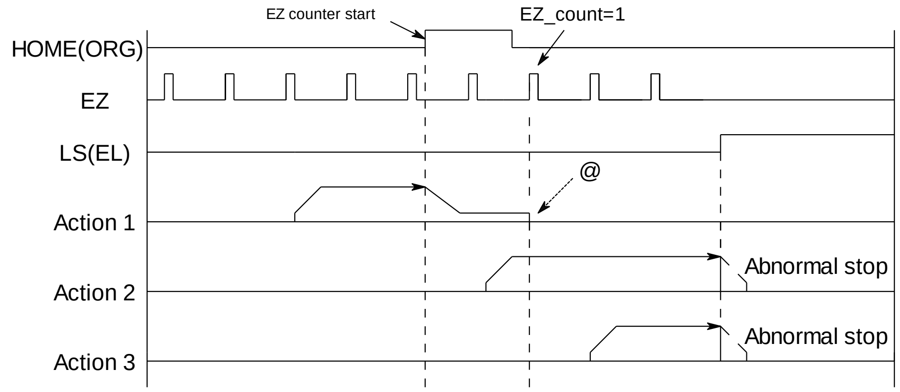
Mode 1 Description



Mode1:

1. The motion will begin to decelerate to VL on the input signal of HOME(ORG) switch signal turning from OFF to ON, then moves in reverse direction until the HOME(ORG) signal turns from ON to OFF and after the signal turns off, it moves at the FA rate (Backlash speed) in initial direction and immediately stops when the HOME(ORG) signal turns from OFF to ON again.
2. The current position counter will reset upon the HOME(ORG) signal turning from OFF to ON.
3. @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.

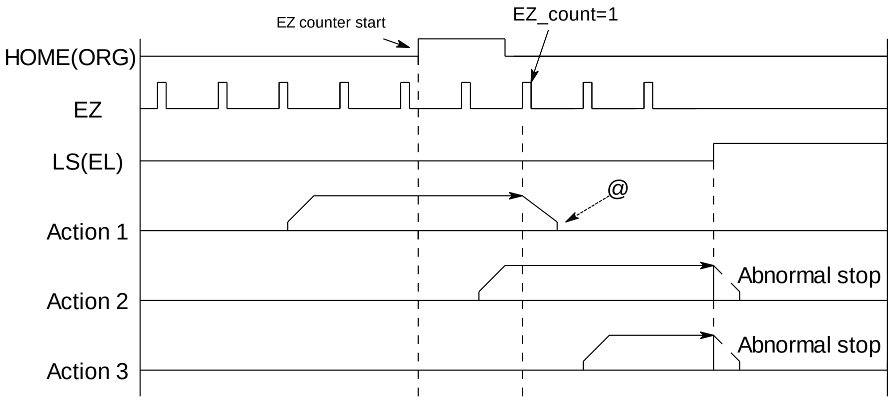
Mode 2 Description



Mode2:

1. The motion begin to decelerate to VL on the HOME(ORG) signal turns from OFF to ON (start EZ counter) and stops immediately upon the EZ counter counting up to the preset value.
2. The current position counter is reset upon the EZ counter counting up to the preset value.
3. @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.

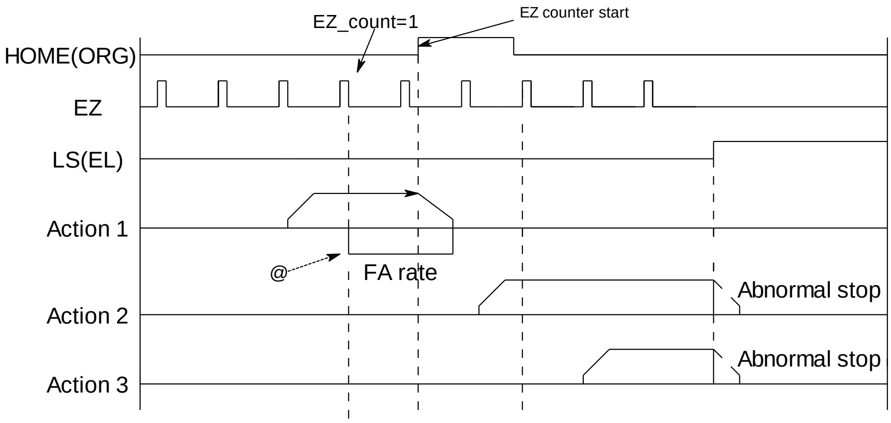
Mode 3 Description



Mode3:

1. The motion begin to decelerate to VL speed then stops after the and the EZ counter counting up to the preset value.
2. The counter is reset upon the EZ counter counting up to the preset value. At the stop position, the counter value maybe not “zero”.
3. @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.

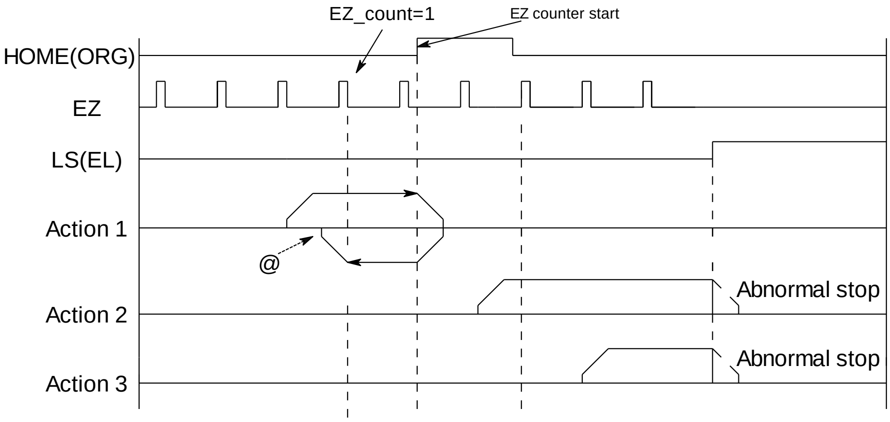
Mode 4 Description



Mode4:

1. The motion begin to decelerate to VL speed upon the HOME(ORG) signal turning from OFF to ON and then moves in reverse direction at the FA rate (Backlash speed) and stop immediately on the EZ counter counting up to the preset value.
2. The current position counter is reset upon the EZ counter counting up to the preset value.
3. @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.

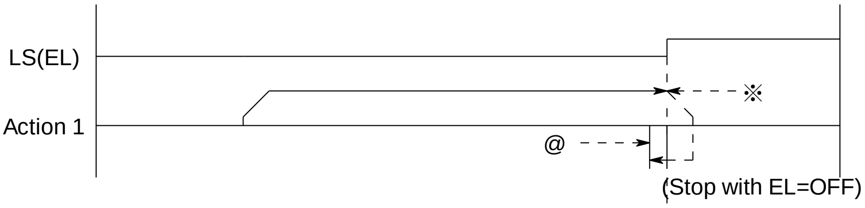
Mode 5 Description



Mode5:

1. The motion begin to decelerate to VL speed upon the HOME(ORG) signal turning from OFF to ON and then moves in reverse direction. It will decelerate to stop on the EZ counter counting up to the preset value.
2. The current position counter is reset upon the EZ counter counting up to the preset value.
3. @ position is the ERC signal output when it is configured as " automatic output of ERC signal ".

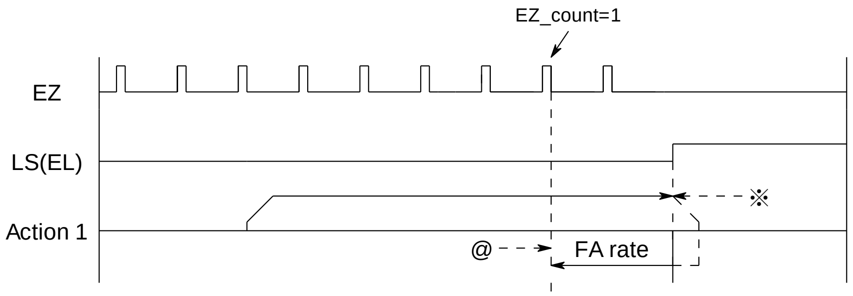
Mode 6 Description



Mode6:

1. The motion immediately stops or decelerate to stop (refer: ***MPC_EL_config_set*** , ELl_mode=1) upon the LS(EL) signal turning ON and then moves in reverse direction at the FA rate (Backlash speed). It will immediately stop upon the LS(EL) signal turning from ON to OFF.
2. The current position counter is reset when the LS(EL) signal turns from ON to OFF .
3. @ position is the ERC signal output when it is configured as " automatic output of ERC signal ".

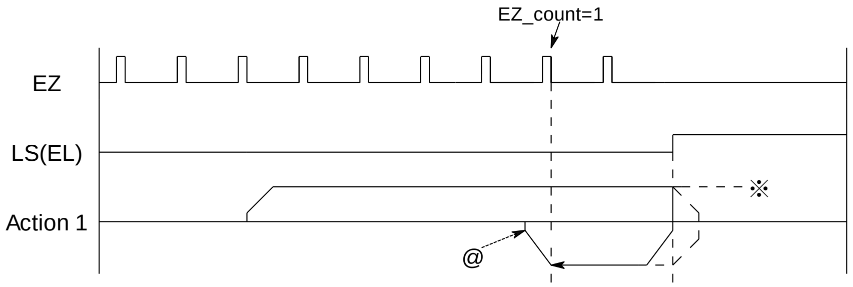
Mode 7 Description



Mode7:

1. The motion immediately stops or decelerate to stop (refer: ***MPC_EL_config_set*** , ELl_mode=1) upon the LS(EL) signal turning ON and then moves in reverse direction at the FA rate (Backlash speed). It will immediately stop upon the EZ counter counting up to the preset value and the LS(EL) signal has turned from ON to OFF.
2. The current position counter is reset at the immediate stop upon the EZ counter counting up to the preset value.
3. @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.

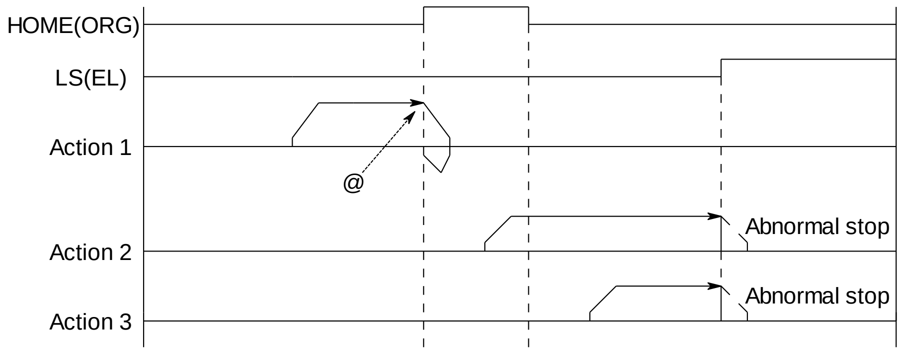
Mode 8 Description



Mode8:

1. The motion immediately stops or decelerate to stop (refer: ***MPC_EL_config_set*** , ELl_mode=1) upon the LS(EL) signal turning ON then moves in reverse direction. It will begin to decelerate to VL to stop on the EZ counter counting up to the preset value.
2. The current position counter is reset upon the EZ counter counting up to the preset value and the LS(EL) signal has turned from ON to OFF. At the stop position, the counter value maybe not “zero”.
3. @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.

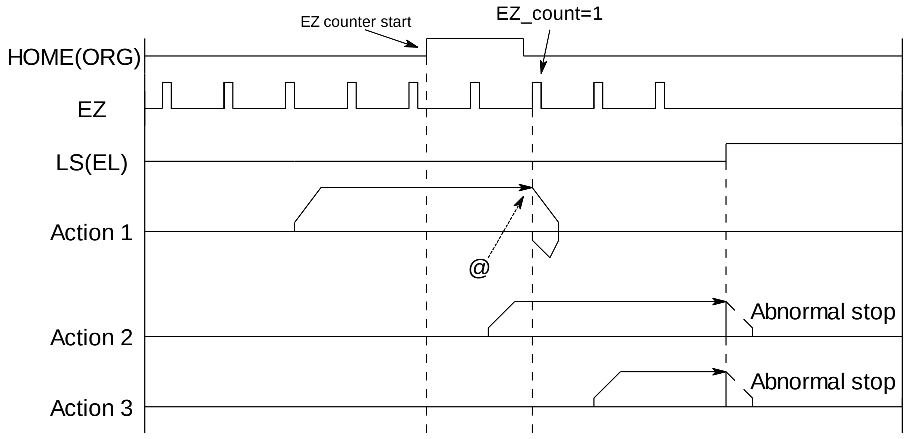
Mode 9 Description



Mode9:

1. After doing homing mode 0 then motion reverse the direction and immediately stop on position counter count down to 0.
2. The current position counter is reset upon@ point.
3. During homing complete, @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.

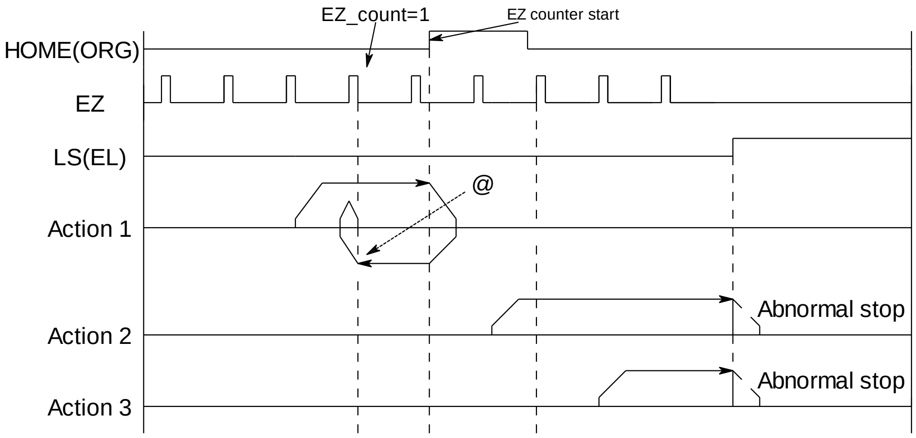
Mode 10 Description



Mode10:

1. After doing homing mode 3 then motion reverse the direction and immediately stop on the position counter count down to 0.
2. The current position counter is reset upon@ point.
3. During homing complete, @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.

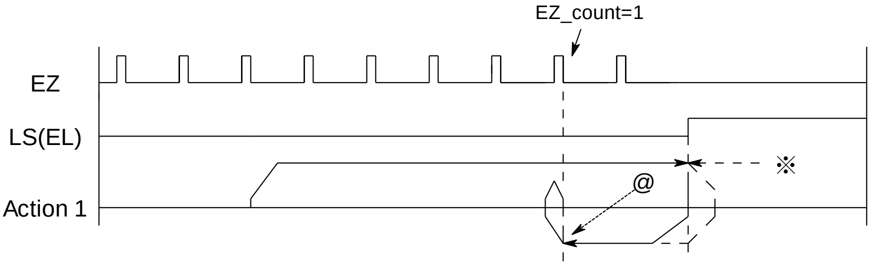
Mode 11 Description



Mode11:

1. After doing homing mode 5, then motion reverse the direction and immediately stop on the position counter count down to 0.
2. The counter is reset upon@ point.
3. During homing complete, @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.

Mode 12 Description



Mode12:

1. After doing homing mode 8, then motion reverse the direction and immediately stop on the position counter count down to 0.
2. The counter is reset upon@ point.
3. During homing complete, @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.

- **MPC3042A_start_homing**

Format: **u32 status = MPC3042A_start_homing(u8 CardID,u8 axis,i32 VL,i32 VH, f64 Tacc,u8 direction)**

Purpose: To command the homing motion.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis
VL	i32	pps of start speed (0~6553500)
VH	i32	pps of final speed (0~6553500)
Tacc	f64	acceleration time
direction	u8	direction of homing 0: positive direction 1: negative direction

- **MPC3042A_set_current_position**

Format : **u32 status = MPC3042A_set_current_position(u8 CardID,u8 axis, i32 current_posi)**

Purpose: To setup the coordinate of current position.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y 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).

● MPC3042A_read_current_position

Format : **u32 status = MPC3042A_read_current_position(u8 CardID,u8 axis, i32 *current_posi)**

Purpose: To readback the coordinate of current position.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y 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 (initial( )) and homing.

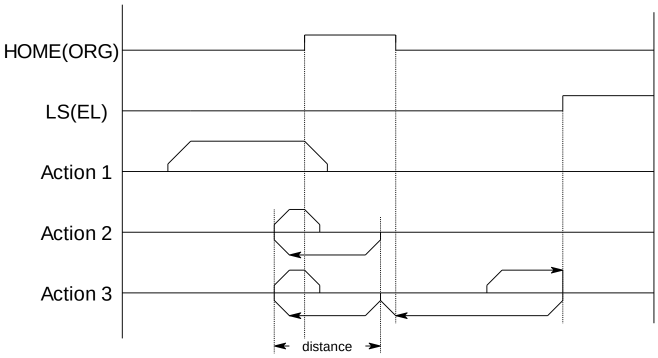
● MPC3042A_start_origin_search_homing

Format : **u32 status = MPC3042A_start_origin_search_homing(u8 CardID,u8 axis,i32 VL, i32 VH,f64 Tacc,u8 direction,u32 distance)**

Purpose: To command origin search mode homing motion.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis
VL	i32	pps of start speed (0~6553500)
VH	i32	pps of final speed (0~6553500)
Tacc	f64	acceleration time
direction	u8	direction of homing 0: positive direction 1: negative direction
distance	u32	

8.7 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.

MPC3042A_backlash_comp( ) is the function to setup compensation.

MPC3042A_readback_backlash_comp( ) to read back the backlash parameter.

- **MPC3042A_backlash_comp**

Format : **u32 status = MPC3042A_backlash_comp(u8 CardID,u8 axis, u16 backlash_pulse, u8 backlash_dir,u32 backlash_speed)**

Purpose: To setup backlash compensation.

Parameters:

Input:

Name	Type	Description
CardID	u8	Assigned by rotary switch
axis	u8	0: X axis 1: Y axis
backlash_pulse	u16	backlash pulse (0 ≦backlash_pulse≦4095)
backlash_dir	u8	0: the first compensation is negative direction 1: the first compensation is positive direction
backlash_speed	u32	backlash speed (pps) (0 ≦backlash_speed≦6553500)

- **MPC3042A_readback_backlash_comp**

Format : **u32 status = MPC3042A_readback_backlash_comp(u8 CardID,u8 axis, u16* backlash_pulse, u8* backlash_dir, u32* backlash_speed)**

Purpose: Read back configuration of backlash compensation.

Parameters:

Input:

Name	Type	Description
CardID	u8	Assigned by rotary switch
axis	u8	0: X axis 1: Y axis

Output:

Name	Type	Description
backlash_pulse	u16	backlash pulse (0 ≦backlash_pulse≦4095)
backlash_dir	u8	0: the first compensation is negative direction 1: the first compensation is positive direction
backlash_speed	u32	backlash speed (pps) (0 ≦backlash_speed≦6553500)

8.8 Point to point motion control

Refer 7.5 Velocity mode motion, Prepare for motion control to setup the pre-requisit conditions.

You may control any of the 2 axes to work in point to point motion mode. Command to positioning

MPC3042A_T_curve_position_move( ) for trapezoidal acc/dec profile.

MPC3042A_S_curve_position_move( ) or

MPC3042A_S1_curve_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,

MPC3042A_position_change( ) will do.

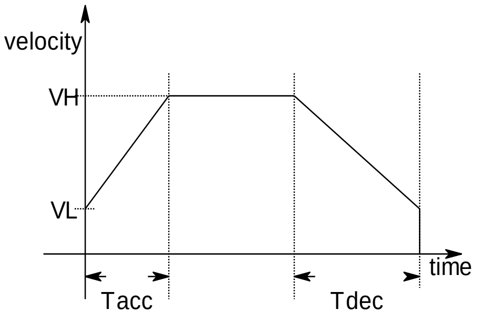
- **MPC3042A_T_curve_position_move**

Format : **u32 status = MPC3042A_T_curve_position_move(u8 CardID,u8 axis,i32 position, u8 posi_mode,i32 VL,i32 VH,f64 Tacc,f64 Tdec)**

Purpose: To point to point positioning at trapezoidal profile.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y 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, start speed (0 ≦ VL ≦ 6553500) Tacc,Tdec: seconds.
VH	i32	
Tacc	f64	
Tdec	f64	

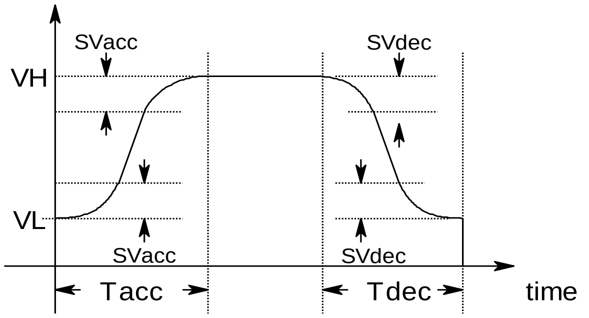
- **MPC3042A_S_curve_position_move**

Format : **u32 status = MPC3042A_S_curve_position_move(u8 CardID,u8 axis,i32 position, u8 posi_mode, i32 VL, i32 VH, f64 Tacc, f64 Tdec, u32 SVacc, u32 SVdec)**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y 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	 SVacc SVdec VH VL SVacc SVdec Tacc Tdec time VH,VL : pps, (0 ≦ VH ≦ 6553500) Tacc,Tdec: seconds. SVacc,SVdec: frequency difference of s curve range, 0≦Svacc(Svdec)≦1/2(VH-VL)
VH	i32	
Tacc	f64	
Tdec	f64	
SVacc	u32	
SVdec	u32	

Note on point to point motion control:

1. Point to point motion control in continuous mode (MPC3042A_set_continuous_flag ( ), conti_flag=1), be sure to check continuous buffer (MPC3042A_check_continuous_buffer( )) until 'full' not equal 1, else the command will be defective.
2. In continuous mode, be sure to set maximum speed first (MPC3042A_fix_speed_range( )).
3. In non-continuous mode(MPC3042A_set_continuous_flag( )，conti_flag=0), be sure to check (MPC3042A_read_motion_status( ); check_factor=0，ret_flag =1) to confirm the motion is ready.

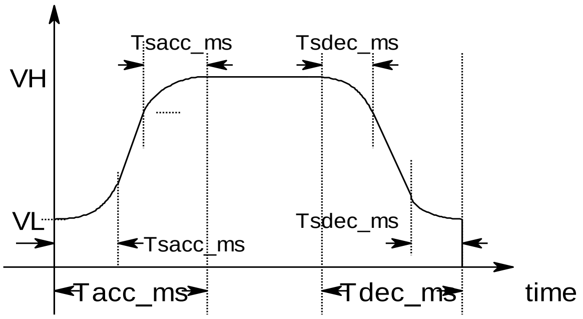
- **MPC3042A_S1_curve_position_move**

Format : **u32 status = MPC3042A_S1_curve_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)

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y 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	 VH VL Tsacc_ms Tsdec_ms Tsacc_ms Tsdec_ms Tacc_ms Tdec_ms time VH,VL : pps, (0 ≦ VH ≦ 6553500) Tacc_ms,Tdec_ms: mili-seconds. Tsacc_ms,Tsdec_ms: mili-seconds.
VH	i32	
Tacc_ms	u32	
Tdec_ms	u32	
Tsacc_ms	u32	
Tsdec_ms	u32	

Tsdec_ms

Note on point to point motion control:

1. Point to point motion control in continuous mode (MPC3042A_set_continuous_flag ( ), conti_flag=1), be sure to check continuous buffer (MPC3042A_check_continuous_buffer( )) until ‘full’ not equal 1, else the command will be defective.
2. In continuous mode, be sure to set maximum speed first (MPC3042A_fix_speed_range( )).
3. In non-continuous mode(MPC3042A_set_continuous_flag( )，conti_flag=0), be sure to check (MPC3042A_read_motion_status( ); check_factor=0，ret_flag =1) to confirm the motion is ready.

MPC3042A_position_change

Format : **u32 status = MPC3042A_position_change(u8 CardID,u8 axis,i32 new_pos, u8 posi_mode)**

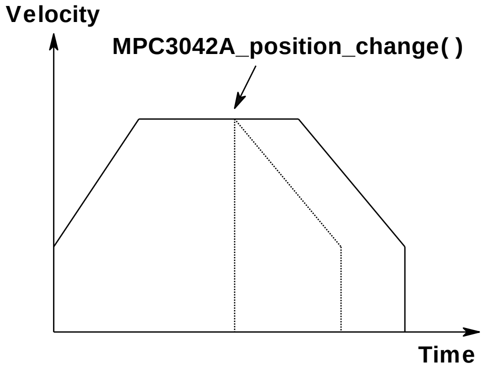
Purpose: To change positioning while point to point motion is running.

Parameters:

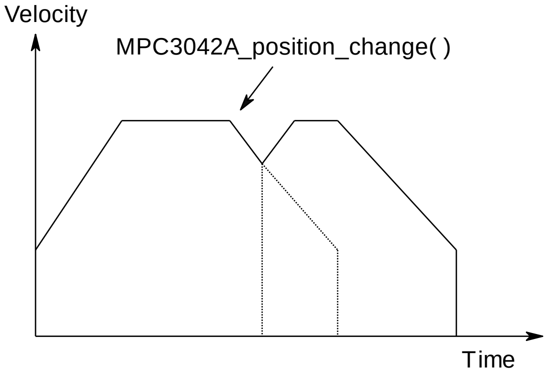
Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y axis
new_pos	i32	new target position (-134,217,728 ≦new_pos≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute

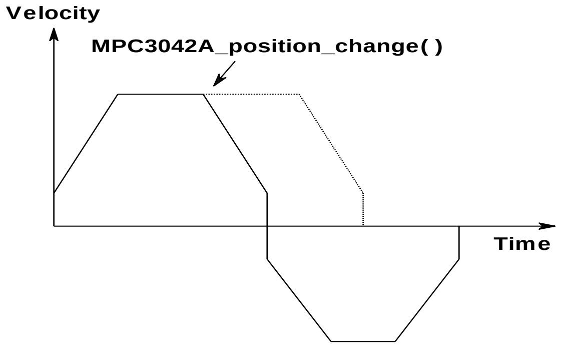
Note on position change:



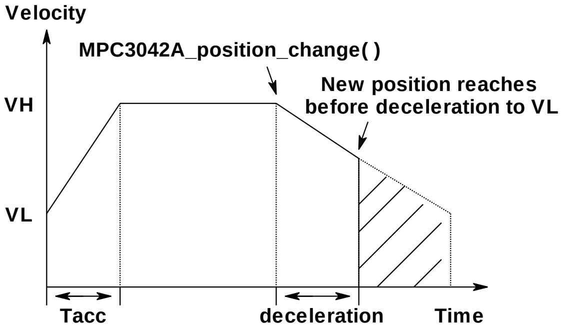
1. Command to change position at VH range



2. Command to change position at deceleration range



3. New position at different side


Velocity
MPC3042A_position_change( )
New position reaches before deceleration to VL
VH
VL
Tacc
deceleration
Time

4. New position at mid-way of deceleration range

8.9 Suppression of vibration

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

MPC3042A_suppress_vibration( ) will give less vibration at final positioning.

MPC3042A_readback_suppress_vibration( ) to read back the data you set.

- **MPC3042A_suppress_vibration**

Format : **u32 status = MPC3042A_suppress_vibration(u8 CardID,u8 axis,u16 RT,u16 FT)**

Purpose: To setup vibration suppression mode.

Parameters:

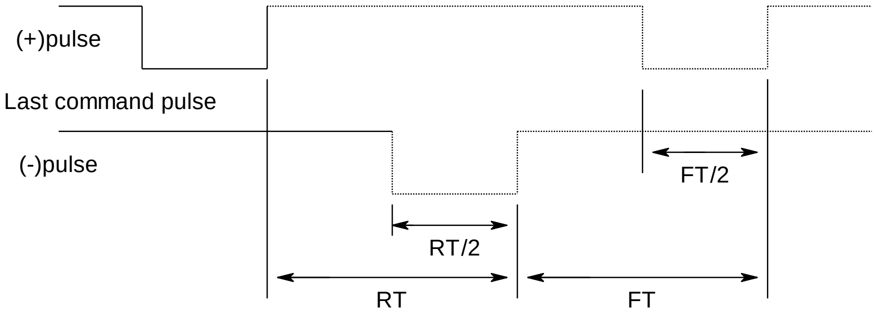
Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X axis 1: Y 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 MPC3042A 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.



- **MPC3042A_readback_suppress_vibration**

Format : **u32 status = MPC3042A_readback_suppress_vibration(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 rotary switch
axis	u8	0: X axis 1: Y 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)

8.10 Linear interpolation

Once you have homed and configured the motion profile, the linear interpolation function now is available.

MPC3042A_T_curve_move_LINE2( ) for any two axes linear interpolation at trapezoidal profile.

MPC3042A_S_curve_move_LINE2( ) or

MPC3042A_S1_curve_move_LINE2( ) for any two axes linear interpolation at S curve profile.

For the linear interpolation application that needs to change position or speed profile on the fly, using

MPC3042A_OnLine_T_curve_change( ) to change for single axis,

MPC3042A_OnLine_T_curve_change_LINE2( ) to change for dual axes.

Note on applying linear interpolation command on continuous mode:

1. All the commends mentioned above are applicable to continuous mode.
2. Linear interpolation motion control in continuous mode (while MPC3042A_set_continuous_flag ( ), conti_flag=1), be sure to check continuous buffer (MPC3042A_check_continuous_buffer( )) until ‘full’ not equal 1, else the command will be defective.
3. In continuous mode, be sure to set maximum speed first (MPC3042A_fix_speed_range( )) at the operation axes.
4. In non-continuous mode(MPC3042A_set_continuous_flag( )，conti_flag=0), be sure to check (MPC3042A_read_motion_status( ); check_factor=0，ret_flag =1) to confirm the motion axes are ready.

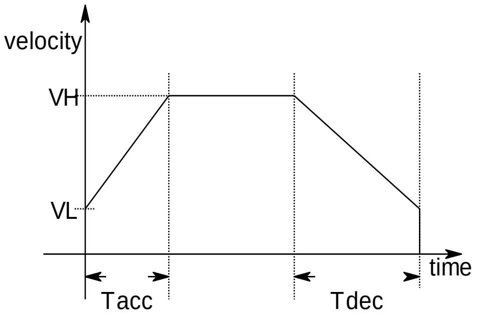
MPC3042A_T_curve_move_LINE2

Format : **u32 status = MPC3042A_T_curve_move_LINE2(u8 CardID,i32 Position1, i32 Position2,u8 posi_mode,i32 VL,i32 VH,f64 Tacc,f64 Tdec)**

Purpose: To take linear interpolation movement with trapezoidal profile.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
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)
posi_mode	u8	0: relative 1: absolute
VL	i32	 VH,VL:pps, start speed (0 ≦ VL ≦ 6553500) Tacc,Tdec: seconds.
VH	i32	
Tacc	f64	
Tdec	f64	

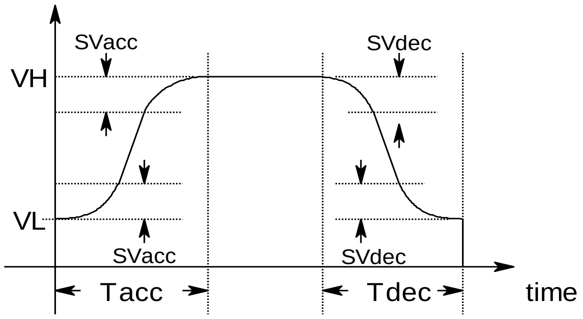
- **MPC3042A_S_curve_move_LINE2**

Format : **u32 status = MPC3042A_S_curve_move_LINE2(u8 CardID,i32 Position1, i32 Position2,u8 posi_mode,i32 VL,i32 VH,f64 Tacc,f64 Tdec, u32 SVacc,u32 SVdec)**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
Position1	i32	target position (absolute or relative) for the X axis (-134,217,728 ≦Position1≦ 134,217,727)
Position2	i32	target position (absolute or relative) for the Y axis (-134,217,728 ≦Position2≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 SVacc SVdec VH VL SVacc SVdec Tacc Tdec time VH,VL : pps, (0 ≦ VH ≦ 6553500) Tacc,Tdec: seconds. SVacc,SVdec: Frequency difference of s curve range , 0≦Svacc(Svdec)≦1/2(VH-VL)
VH	i32	
Tacc	f64	
Tdec	f64	
SVacc	u32	
SVdec	u32	

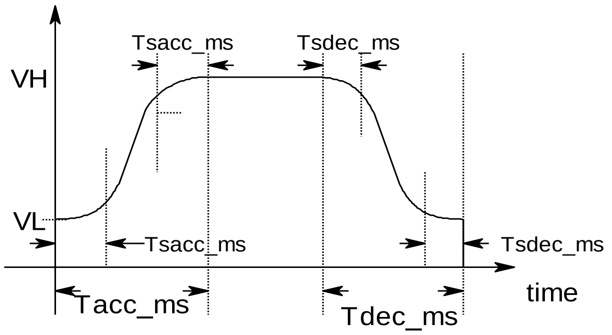
- **MPC3042A_S1_curve_move_LINE2**

Format : **u32 status = MPC3042A_S1_curve_move_LINE2(u8 CardID,i32 Position1,
i32 Position2, u8 posi_mode, i32 VL, i32 VH, u32 Tacc_ms,
u32 Tdec_ms, u32 Tsacc_ms, u32 Tsdec_ms)**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
Position1	i32	target position (absolute or relative) for the 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)
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 ≦ 6553500) Tacc: mili-seconds of total acc time. Tdec: mili-seconds of total dec time. Tsacc: mili-seconds of s curve region acc time. Tsdec: mili-seconds of s curve region dec time.
VH	i32	
Tacc_ms	u32	
Tdec_ms	u32	
Tsacc_ms	u32	
Tsdec_ms	u32	

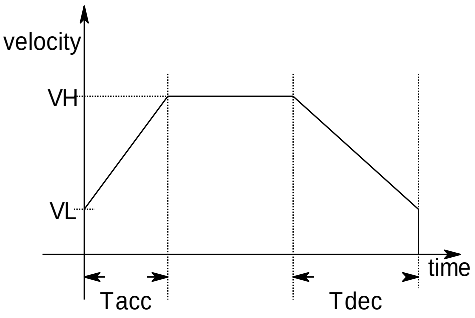
- **MPC3042A_OnLine_T_curve_change**

Format : **u32 status = MPC3042A_OnLine_T_curve_change(u8 CardID,u8 axis, i32 Position,u8 posi_mode,i32 VL,i32 VH,f64 Tacc,f64 Tdec)**

Purpose: To change the motion parameters on the fly.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X 1: Y
Position	i32	new target position (-134,217,728 ≦Position≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 VH,VL:pps, start speed (0 ≦ VL ≦ 6553500) Tacc,Tdec: seconds.
VH	i32	
Tacc	f64	
Tdec	f64	

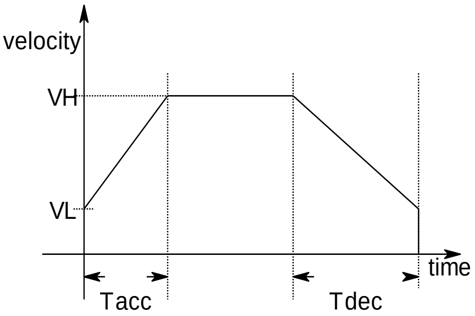
MPC3042A_OnLine_T_curve_change_LINE2

Format : **u32 status = MPC3042A_OnLine_T_curve_change_LINE2(u8 CardID, i32 Position1,i32 Position2,u8 posi_mode,i32 VL,i32 VH,f64 Tacc, f64 Tdec)**

Purpose: To change the motion parameters on the fly for linear interpolation.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
Position1	i32	new target position for X axis (-134,217,728 ≦Position1≦ 134,217,727)
Position2	i32	new target position for Y axis (-134,217,728 ≦Position2≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 VH,VL:pps, start speed (0 ≦ VL ≦ 6553500) Tacc,Tdec: seconds.
VH	i32	
Tacc	f64	
Tdec	f64	

8.11 Synchronized start motion

In some applications the motion needs to start on the occasion of specific conditions, mostly a predefined point or angle occurs. In lathe application, the thread cutting is the application of this kind, cutter begins to cut thread at a predefined angle.

MPC3042A_config_compare_start_motion( ) is used to configure the compare source and the compare condition to trigger the start of motion.

The compared data is configured by:

MPC3042A_set_compare_start_data( )

After you have setup the data, you must decide what kind of motion you will take at the synchronous start, maybe single axis, dual ,triple even 4 axes, linear or circular at your will. Use:

MPC3042A_T_curve_wait_Cmpstart( ) to synchronous start single axis T profile motion.

MPC3042A_S_curve_wait_Cmpstart( ) to synchronous start single axis S profile motion.

MPC3042A_S1_curve_wait_Cmpstart( ) to synchronous start single axis S1 profile motion.

You can check the synchronous start flag by

MPC3042A_read_compare_start_flag( )

● MPC3042A_config_compare_start_motion

Format : u32 status = MPC3042A_config_compare_start_motion(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
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 select for compare source.

MPC3042A_set_compare_start_data

Format : **u32 status = MPC3042A_set_compare_start_data(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

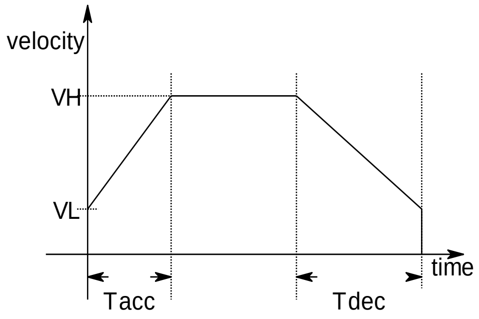
MPC3042A_T_curve_wait_Cmpstart

Format : **u32 status = MPC3042A_T_curve_wait_Cmpstart(u8 CardID,u8 Axis,**
i32 Position, u8 posi_mode,i32 VL,i32 VH,f64 Tacc,f64 Tdec);

Purpose: To setup the T profile motion and wait for synchronous start signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
Axis	u8	0: X 1: Y 2: Z 3: A
Position	i32	target position (absolute or relative) for motion (-134,217,728 ≦Position≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 VH,VL:pps, start speed (0 ≦ VL ≦ 6553500) Tacc,Tdec: seconds.
VH	i32	
Tacc	f64	
Tdec	f64	

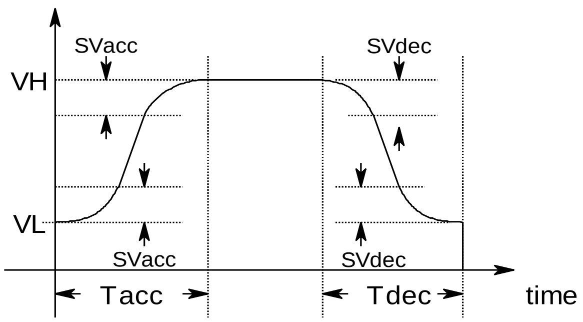
MPC3042A_S_curve_wait_Cmpstart

Format : **u32 status = MPC3042A_S_curve_wait_Cmpstart(u8 CardID,u8 Axis,**
i32 Position,u8 posi_mode,i32 VL,i32 VH,f64 Tacc,f64 Tdec,
u32 SVacc, u32 SVdec);

Purpose: To setup the S profile motion and wait for synchronous start signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
Axis	u8	0: X 1: Y 2: Z 3: A
Position	i32	target position (absolute or relative) for motions (-134,217,728 ≦Position≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 SVacc SVdec VH VL SVacc SVdec Tacc Tdec time VH,VL : pps, (0 ≦ VH ≦ 6553500) Tacc,Tdec: seconds. SVacc,SVdec: frequency difference of s curve range , 0≦Svacc(Svdec)≦1/2(VH-VL)
VH	i32	
Tacc	f64	
Tdec	f64	
SVacc	u32	
SVdec	u32	

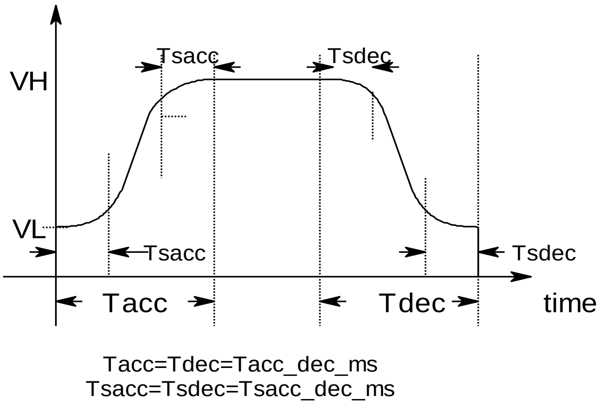
- **MPC3042A_S1_curve_wait_Cmpstart**

Format : **u32 status = MPC3042A_S1_curve_wait_Cmpstart(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 setup the S profile motion and wait for synchronous start signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
Axis	u8	0: X 1: Y 2: Z 3: A
Position	i32	target position (absolute or relative) for motions (-134,217,728 ≦Position≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 VH VL Tsacc Tsdec Tsacc Tsdec Tacc Tdec time Tacc=Tdec=Tacc_dec_ms Tsacc=Tsdec=Tsacc_dec_ms Tacc_ms,Tdec_ms: total acc/dec time in mili-second Tsacc_ms, Tsdec_ms: s curve portion time in mili-second
VH	i32	
Tacc_ms	u32	
Tdec_ms	u32	
Tsacc_ms	u32	
Tsdec_ms	u32	

- **MPC3042A_read_compare_start_flag**

Format : **u32 status = MPC3042A_read_compare_start_flag(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

8.12 Circular interpolation

Once you have homed and configured the motion profile, the circular interpolation function now is available. If you wish to use the circle center and end position as parameters, use:

MPC3042A_ARC2_center_move( ) to move an arc.

MPC3042A_T_ARC2_center_move( ) with T type acceleration and deceleration.

MPC3042A_S1_ARC2_center_move( ) with S type acceleration and deceleration.

If you wish to use current point and the other 2 points as the circle trajectory parameters, use:

MPC3042A_ARC2_3P_move( ) to move an arc.

MPC3042A_T_ARC2_3P_move( ) with T type acceleration and deceleration.

MPC3042A_S1_ARC2_3P_move( ) with S type acceleration and deceleration.

If you want to have a circle defined by 3 points (circle pass through the 3 pre-defined point)

MPC3042A_CIR2_3P_move( ) for constant speed motion for full circle.

MPC3042A_T_CIR2_3P_move( ) with T type acceleration and deceleration.

MPC3042A_S1_CIR2_3P_move( ) with S type acceleration and deceleration.

If you use current position and end position with a radius, you can have circular interpolation to move an arc:

MPC3042A_ARC2_Radius_move( ) without acc/dec.

MPC3042A_T_ARC2_Radius_move( ) with T type acceleration and deceleration.

MPC3042A_S1_ARC2_Radius_move( ) with S type acceleration and deceleration.

To move a circle:

MPC3042A_CIR2_Radius_move( ) without acc/dec.

MPC3042A_T_CIR2_Radius_move( ) with T type acceleration and deceleration.

MPC3042A_S1_CIR2_Radius_move( ) with S type acceleration and deceleration.

Note on circular interpolation:

1. All the commands mentioned above are applicable to continuous mode.
2. Circular interpolation motion control in continuous mode (MPC3042A_set_continuous_flag ( ), conti_flag=1), be sure to check continuous buffer (MPC3042A_check_continuous_buffer( )) until 'full' not equal 1, else the command will be defective.
3. In continuous mode, be sure to set maximum speed first (MPC3042A_fix_speed_range( )) at the operation axes.
4. In non-continuous mode(MPC3042A_set_continuous_flag( ), conti_flag=0), be sure to check (MPC3042A_read_motion_status( ); check_factor=0, ret_flag =1) to confirm the motion axes are ready.

- **MPC3042A_ARC2_center_move**

Format : **u32 status = MPC3042A_ARC2_center_move(u8 CardID, i32 center1, i32 center2, i32 endp1, i32 endp2, u8 posi_mode, i32 VH, u8 direction)**

Purpose: To take circular interpolation movement with circle center and end position parameter as arc trajectory.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
center1	i32	circle center position (absolute or relative) for X axis (-134,217,728 ≦center1≦ 134,217,727)
center2	i32	circle center position (absolute or relative) for Y axis (-134,217,728 ≦center2≦ 134,217,727)
endp1	i32	end position (absolute or relative) for X axis (-134,217,728 ≦endp1≦ 134,217,727)
endp2	i32	end position (absolute or relative) for Y axis (-134,217,728 ≦endp2≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VH	i32	vector velocity of circular interpolation
direction	u8	0: CW direction 1: CCW direction

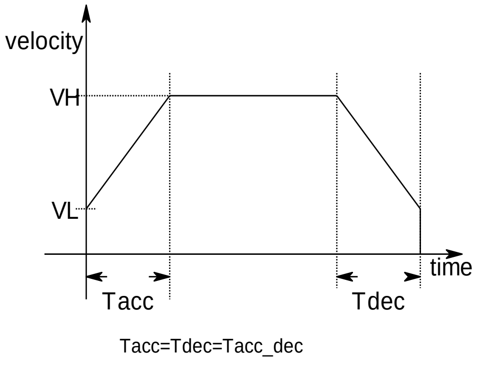
MPC3042A_T_ARC2_center_move

Format : **u32 status = MPC3042A_T_ARC2_center_move(u8 CardID, i32 center1,**
i32 center2, i32 endp1, i32 endp2, u8 posi_mode, i32 VL, i32 VH,
f64 Tacc_dec, u8 direction);

Purpose: To take circular interpolation movement with circle center and end position and the T type acceleration/deceleration as arc trajectory.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
center1	i32	circle center position (absolute or relative) for X axis (-134,217,728 ≦center1≦ 134,217,727)
center2	i32	circle center position (absolute or relative) for Y axis (-134,217,728 ≦center2≦ 134,217,727)
endp1	i32	end position (absolute or relative) for X axis (-134,217,728 ≦endp1≦ 134,217,727)
endp2	i32	end position (absolute or relative) for Y axis (-134,217,728 ≦endp2≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 velocity VH VL time Tacc Tdec Tacc=Tdec=Tacc_dec Tacc_dec: seconds of acc or dec time
VH	i32	
Tacc_dec	f64	
direction	u8	0: CW direction 1: CCW direction

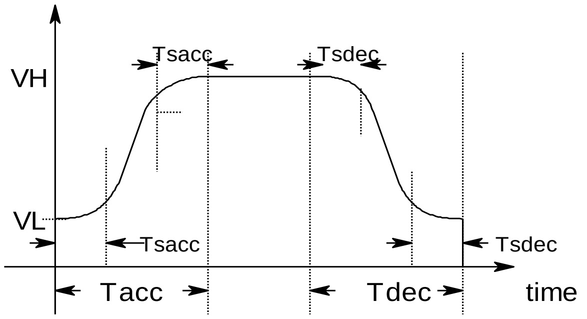
- **MPC3042A_S1_ARC2_center_move**

Format : **u32 status = MPC3042A_S1_ARC2_center_move(u8 CardID, i32 center1,
i32 center2, i32 endp1, i32 endp2, u8 posi_mode, i32 VL, i32 VH,
u32 Tacc_dec_ms, u32 Tsacc_dec_ms, u8 direction);**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
center1	i32	circle center position (absolute or relative) for X axis (-134,217,728 ≦center1≦ 134,217,727)
center2	i32	circle center position (absolute or relative) for Y axis (-134,217,728 ≦center2≦ 134,217,727)
endp1	i32	end position (absolute or relative) for X axis (-134,217,728 ≦endp1≦ 134,217,727)
endp2	i32	end position (absolute or relative) for Y axis (-134,217,728 ≦endp2≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 VH VL Tsacc Tsdec Tsacc Tsdec Tacc Tdec time Tacc=Tdec=Tacc_dec_ms Tsacc=Tsdec=Tsacc_dec_ms
VH	i32	
Tacc_dec_ms	u32	
Tsacc_dec_ms	u32	
direction	u8	0: CW direction 1: CCW direction

- **MPC3042A_ARC2_3P_move**

Format : **u32 status = MPC3042A_ARC2_3P_move(u8 CardID, i32 middle1, i32 middle2, i32 endp1, i32 endp2, u8 posi_mode, i32 VH)**

Purpose: To take circular interpolation movement with current point and the other 2 points for the arc trajectory.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
middle1	i32	middle position (absolute or relative) for X axis (-134,217,728 ≦middle1≦ 134,217,727)
middle2	i32	middle position (absolute or relative) for Y axis (-134,217,728 ≦middle2≦ 134,217,727)
endp1	i32	end position (absolute or relative) for X axis (-134,217,728 ≦endp1≦ 134,217,727)
endp2	i32	end position (absolute or relative) for Y axis (-134,217,728 ≦endp2≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VH	i32	vector velocity of circular interpolation

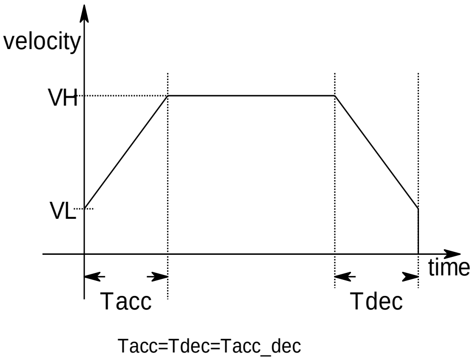
- **MPC3042A_T_ARC2_3P_move**

Format : **u32 status = MPC3042A_T_ARC2_3P_move(u8 CardID, i32 middle1, i32 middle2, i32 endp1, i32 endp2, u8 posi_mode, i32 VL, i32 VH, f64 Tacc_dec);**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
middle1	i32	middle position (absolute or relative) for X axis (-134,217,728 ≦middle1≦ 134,217,727)
middle2	i32	middle position (absolute or relative) for Y axis (-134,217,728 ≦middle2≦ 134,217,727)
endp1	i32	end position (absolute or relative) for X axis (-134,217,728 ≦endp1≦ 134,217,727)
endp2	i32	end position (absolute or relative) for Y axis (-134,217,728 ≦endp2≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 velocity VH VL time Tacc Tdec Tacc=Tdec=Tacc_dec Tacc_dec: seconds of acc or dec time
VH	i32	
Tacc_dec	f64	

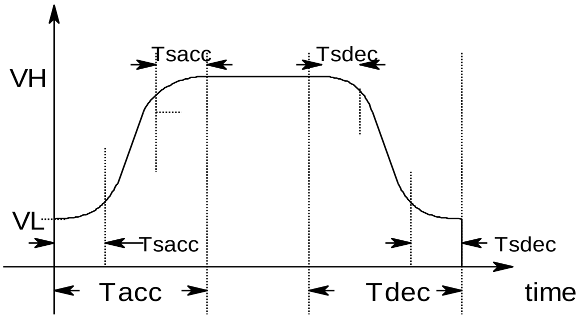
- **MPC3042A_S1_ARC2_3P_move**

Format : **u32 status = MPC3042A_S1_ARC2_3P_move(u8 CardID, i32 middle1, i32 middle2, i32 endp1, i32 endp2, u8 posi_mode, i32 VL, i32 VH, u32 Tacc_dec_ms, u32 Tsacc_dec_ms);**

Purpose: To take circular interpolation movement with S type profile and with current point and the other 2 points for the arc trajectory.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
middle1	i32	middle position (absolute or relative) for X axis (-134,217,728 ≦middle1≦ 134,217,727)
middle2	i32	middle position (absolute or relative) for Y axis (-134,217,728 ≦middle2≦ 134,217,727)
endp1	i32	end position (absolute or relative) for X axis (-134,217,728 ≦endp1≦ 134,217,727)
endp2	i32	end position (absolute or relative) for Y axis (-134,217,728 ≦endp2≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 VH VL Tsacc Tsdec Tsacc Tsdec Tacc Tdec time Tacc=Tdec=Tacc_dec_ms Tsacc=Tsdec=Tsacc_dec_ms
VH	i32	
Tacc_dec_ms	u32	
Tsacc_dec_ms	u32	

- **MPC3042A_CIR2_3P_move**

Format : **u32 status = MPC3042A_CIR2_3P_move(u8 CardID, i32 middle1, i32 middle2, i32 endp1, i32 endp2, u8 posi_mode, i32 VH)**

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 rotary switch
middle1	i32	middle position (absolute or relative) for X axis (-134,217,728 ≦middle1≦ 134,217,727)
middle2	i32	middle position (absolute or relative) for Y axis (-134,217,728 ≦middle2≦ 134,217,727)
endp1	i32	end position (absolute or relative) forX axis (-134,217,728 ≦endp1≦ 134,217,727)
endp2	i32	end position (absolute or relative) for Y axis (-134,217,728 ≦endp2≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VH	i32	vector velocity of circular interpolation

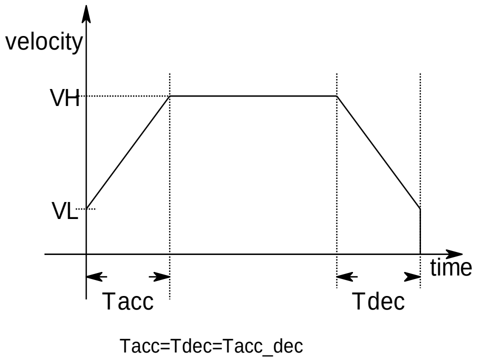
- **MPC3042A_T_CIR2_3P_move**

Format : **u32 status = MPC3042A_T_CIR2_3P_move(u8 CardID, i32 middle1, i32 middle2, i32 endp1, i32 endp2, u8 posi_mode, i32 VL, i32 VH, f64 Tacc_dec);**

Purpose: To take circular interpolation movement with current point and the other 2 points and the T type acceleration/deceleration profile for the circle trajectory.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
middle1	i32	middle position (absolute or relative) for X axis (-134,217,728 ≦middle1≦ 134,217,727)
middle2	i32	middle position (absolute or relative) for Y axis (-134,217,728 ≦middle2≦ 134,217,727)
endp1	i32	end position (absolute or relative) for X axis (-134,217,728 ≦endp1≦ 134,217,727)
endp2	i32	end position (absolute or relative) for Y axis (-134,217,728 ≦endp2≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 velocity VH VL time Tacc Tdec Tacc=Tdec=Tacc_dec Tacc_dec: seconds of acc or dec time
VH	i32	
Tacc_dec	f64	

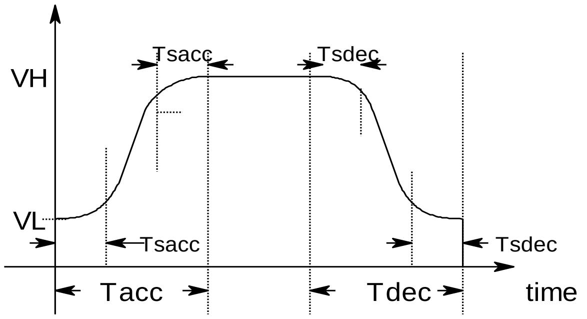
- **MPC3042A_S1_CIR2_3P_move**

Format : **u32 status = MPC3042A_S1_CIR2_3P_move(u8 CardID, i32 middle1, i32 middle2, i32 endp1, i32 endp2, u8 posi_mode, i32 VL, i32 VH, u32 Tacc_dec_ms, u32 Tsacc_dec_ms);**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
middle1	i32	middle position (absolute or relative) for X axis (-134,217,728 ≦middle1≦ 134,217,727)
middle2	i32	middle position (absolute or relative) for Y axis (-134,217,728 ≦middle2≦ 134,217,727)
endp1	i32	end position (absolute or relative) for X axis (-134,217,728 ≦endp1≦ 134,217,727)
endp2	i32	end position (absolute or relative) for Y axis (-134,217,728 ≦endp2≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 VH VL Tsacc Tsdec Tsacc Tsdec Tacc Tdec time Tacc=Tdec=Tacc_dec_ms Tsacc=Tsdec=Tsacc_dec_ms
VH	i32	
Tacc_dec_ms	u32	
Tsacc_dec_ms	u32	

- **MPC3042A_ARC2_Radius_move**

Format : **u32 status = MPC3042A_ARC2_Radius_move(u8 CardID, i32 radius,i32 endp1, i32 endp2,u8 posi_mode,i32 VH,u8 direction)**

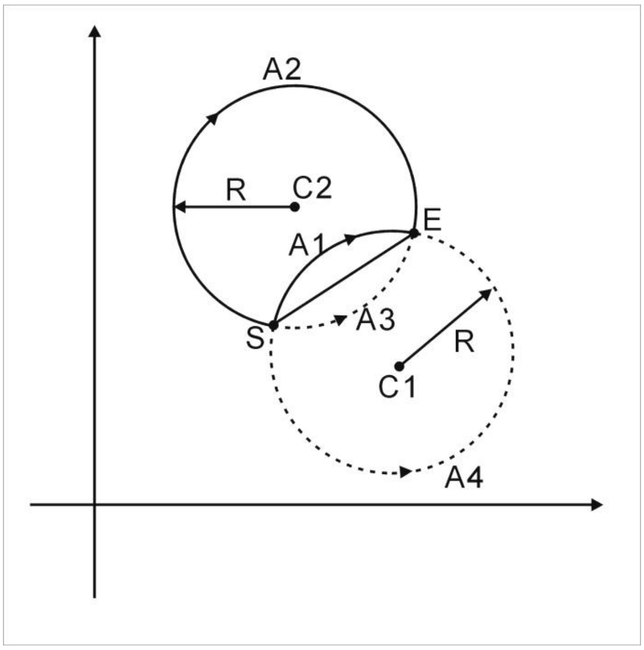
Purpose: To take the current position and end position to make an arc at designated R.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
radius	i32	radius for the circle to pass currenet position and endpoint
endp1	i32	end position (absolute or relative) for X axis (-134,217,728 ≦endp1≦ 134,217,727)
endp2	i32	end position (absolute or relative) for Y axis (-134,217,728 ≦endp2≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VH	i32	vector velocity of circular interpolation
direction	u8	0: CW 1: CCW

Note:



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.

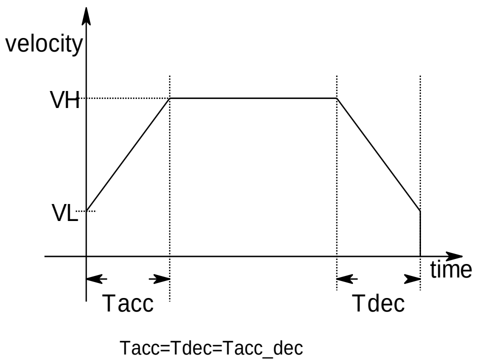
MPC3042A_T_ARC2_Radius_move

Format : **u32 status = MPC3042A_T_ARC2_Radius_move(u8 CardID, i32 radius, i32 endp1, i32 endp2, u8 posi_mode, i32 VL, i32 VH, f64 Tacc_dec, u8 direction)**

Purpose: To take the current position and end position to make an arc at designated R with T type acceleration/deceleration profile.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
radius	i32	radius for the circle to pass currenet position and endpoint
endp1	i32	end position (absolute or relative) for X axis (-134,217,728 ≦endp1≦ 134,217,727)
endp2	i32	end position (absolute or relative) for Y axis (-134,217,728 ≦endp2≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 Tacc_dec: seconds of acc or dec time
VH	i32	
Tacc_dec	f64	
direction	u8	0: CW 1: CCW

MPC3042A_S1_ARC2_Radius_move

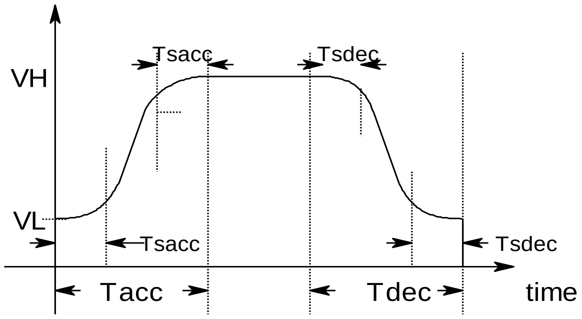
Format : **u32 status = MPC3042A_S1_ARC2_Radius_move(u8 CardID, i32 radius,**
i32 endp1, i32 endp2, u8 posi_mode, i32 VL, i32 VH,
u32 Tacc_dec_ms, u32 Tsacc_dec_ms, u8 direction)

Purpose: To take the current position and end position to make an arc at designated R with S type acceleration/deceleration profile.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
radius	i32	radius for the circle to pass currenet position and endpoint
endp1	i32	end position (absolute or relative) for X axis (-134,217,728 ≦endp1≦ 134,217,727)
endp2	i32	end position (absolute or relative) for Y axis (-134,217,728 ≦endp2≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute

VL	i32	
		Tacc=Tdec=Tacc_dec_ms
		Tsacc=Tsdec=Tsacc_dec_ms

VH	i32	
Tacc_dec_ms	u32	
Tsacc_dec_ms	u32	
direction	u8	0: CW 1: CCW

- **MPC3042A_CIR2_Radius_move**

Format : **u32 status = MPC3042A_CIR2_Radius_move(u8 CardID, i32 radius,i32 endp1, i32 endp2,u8 posi_mode,i32 VH,u8 direction)**

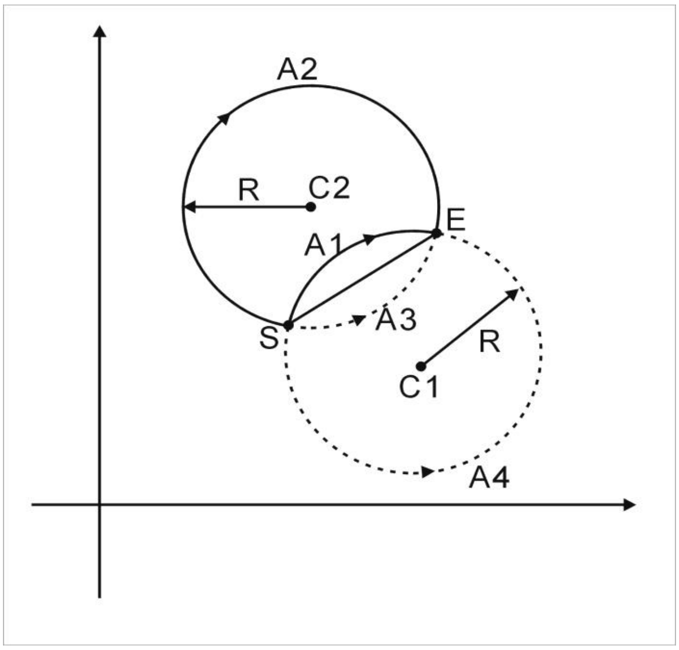
Purpose: To take the current position and end position to make a cycle at designated R.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
radius	i32	radius for the circle to pass currenet position and endpoint
endp1	i32	end position (absolute or relative) for X axis (-134,217,728 ≦endp1≦ 134,217,727)
endp2	i32	end position (absolute or relative) for Y axis (-134,217,728 ≦endp2≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VH	i32	vector velocity of circular interpolation
direction	u8	0: CW 1: CCW

Note:



For example:
S: start point (current position)
E: end point
R: radius

Say the circle will go CW direction,
if R>0 then the circle will go through A1 ;
if R<0 then A2 will be.

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

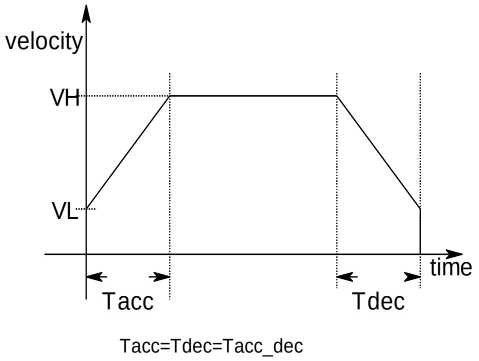
● MPC3042A_T_CIR2_Radius_move

Format : **u32 status = MPC3042A_T_CIR2_Radius_move(u8 CardID, i32 radius, i32 endp1, i32 endp2, u8 posi_mode, i32 VL, i32 VH, f64 Tacc_dec, u8 direction)**

Purpose: To take the current position and end position to make a cycle at designated R with T type acceleration/deceleration profile.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
radius	i32	radius for the circle to pass currenet position and endpoint
endp1	i32	end position (absolute or relative) for X axis (-134,217,728 ≦endp1≦ 134,217,727)
endp2	i32	end position (absolute or relative) for Y axis (-134,217,728 ≦endp2≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 Tacc_dec: seconds of acc or dec time
VH	i32	
Tacc_dec	f64	
direction	u8	0: CW 1: CCW

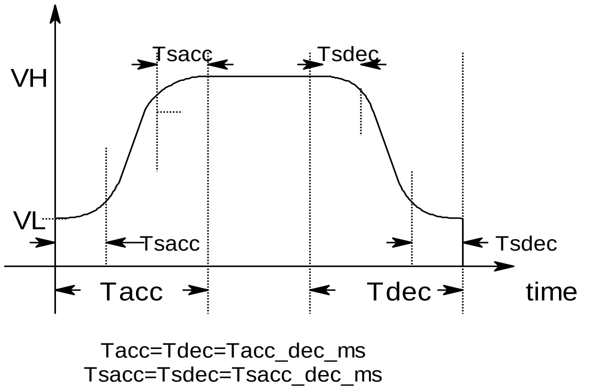
MPC3042A_S1_CIR2_Radius_move

Format : **u32 status = MPC3042A_S1_CIR2_Radius_move(u8 CardID, i32 radius,**
i32 endp1, i32 endp2, u8 posi_mode, i32 VL, i32 VH,
u32 Tacc_dec_ms, u32 Tsacc_dec_ms, u8 direction)

Purpose: To take the current position and end position to make a cycle at designated R with S type acceleration/deceleration profile.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
radius	i32	radius for the circle to pass currenet position and endpoint
endp1	i32	end position (absolute or relative) for X axis (-134,217,728 ≦endp1≦ 134,217,727)
endp2	i32	end position (absolute or relative) for Y axis (-134,217,728 ≦endp2≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	
VH	i32	
Tacc_dec_ms	u32	
Tsacc_dec_ms	u32	
direction	u8	0: CW 1: CCW

8.13 Continuous motion

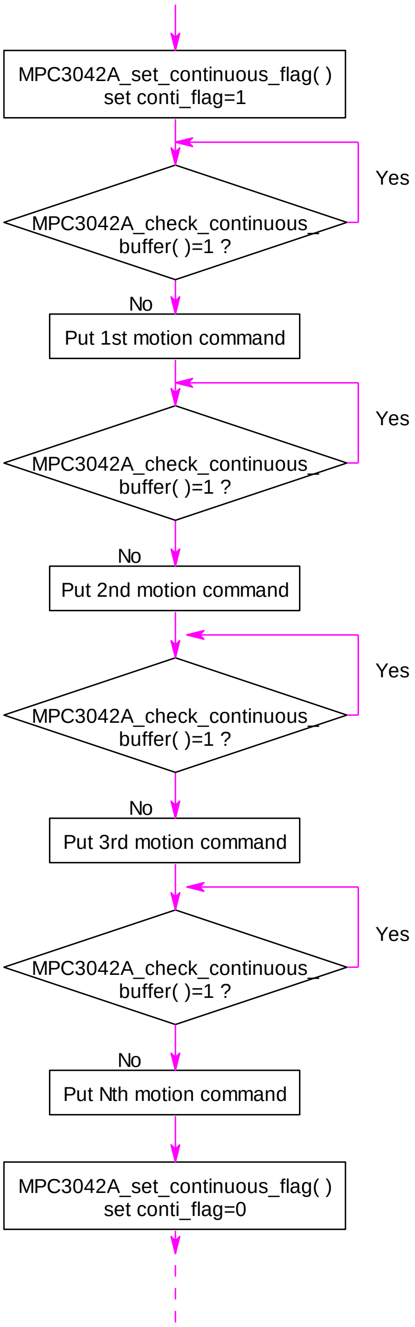
For some applications such as gluing, you need to move continuously (without any stop between segment to segment). MPC3042A provides 3 hardware buffers for motion related registers; the motion command can go to next without any discontinuity, use:

MPC3042A_set_continuous_flag( ) to enable / disable the continuous mode.

For the motion status read back of continuous mode,

MPC3042A_check_continuous_buffer( ) to check the buffer full or not for the availability of the next motion command.

MPC3042A_read_motion_status( ) for motion status read back.


MPC3042A_set_continuous_flag( )
set conti_flag=1
Yes
MPC3042A_check_continuous_buffer( )=1 ?
No
Put 1st motion command
Yes
MPC3042A_check_continuous_buffer( )=1 ?
No
Put 2nd motion command
Yes
MPC3042A_check_continuous_buffer( )=1 ?
No
Put 3rd motion command
Yes
MPC3042A_check_continuous_buffer( )=1 ?
No
Put Nth motion command
MPC3042A_set_continuous_flag( )
set conti_flag=0

- **MPC3042A_set_continuous_flag**

Format : **u32 status = MPC3042A_set_continuous_flag(u8 CardID, u8 axis, u8 conti_flag)**

Purpose: To read back the continuous flag for checking the availability of new motion command.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by rotary switch	
axis	u8	0: X axis	1: Y axis
conti_flag	u8	0: disable continuous mode 1: enable continuous mode	

- **MPC3042A_check_continuous_buffer**

Format : **u32 status = MPC3042A_check_continuous_buffer(u8 CardID,u8 axis, u8 *buffer_full_flag)**

Purpose: To read continuous buffer flag for checking if the buffer is full.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X 1: Y

Output:

Name	Type	Description
buffer_full_flag	u8	0: buffer not full, the card may accept command from PC 1: buffer full, no further command can accept until it is not full.

Note:

The motion command for continuous mode is the same as the normal mode but to ensure the validity of the hardware buffer is required.

- **MPC3042A_read_motion_status**

Format : **u32 status = MPC3042A_read_motion_status(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 rotary switch
axis	u8	0: X 1: Y
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

8.14 Motion restart

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

MPC3042A_OneAxis_restart( ) for single axis restart.

MPC3042A_2Axis_restart( ) for two axis restart.

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

- **MPC3042A_OneAxis_restart**

Format : **u32 status = MPC3042A_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

- **MPC3042A_2Axis_restart**

Format : **u32 status = MPC3042A_2Axis_restart(u8 CardID);**

Purpose: To restart the previously halted 2 axes.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

8.15 Motion event and error status

For the program to take care of special condition interested, please use

MPC3042A_set_event_factor( ) to setup the event generated by the control card.

MPC3042A_read_event_flag( ) will give you the event generating source for your application.

MPC3042A_read_error_flag( ) will report the error conditions for your application.

- ### MPC3042A_set_event_factor

Format : **u32 status = MPC3042A_set_event_factor(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 rotary switch</td></tr>
<tr><td>axis</td><td>u8</td><td colspan="3">0: X 1: Y</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>

MPC3042A_read_event_flag

Format : **u32 status = MPC3042A_read_event_flag(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 rotary switch
axis	u8	0: X 1: Y

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

MPC3042A_read_error_flag

Format : **u32 status = MPC3042A_read_error_flag(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 rotary switch
axis	u8	0: X 1: Y

Output:

Name	Type	Description		
error_flag	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

8.16 Soft limit protection function

For the motion control system, the protection of available motion area is traditionally protected by the limit switches. The over-travel limit switches will stop motion as your configuration setup but you do not have the flexibility to change the protection area on the fly. The software limit enables you to change by program, you can protect the motion at dynamic bases as you need without change or adjust the hardware over-travel limit switches. Special note should be taken, **it is not designed to replace the hardware over-travel limit switches**.

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 coordinate system, use

MPC3042A_config_softlimit( ) to setup configuration.
MPC3042A_readback_config_softlimit( ) to read back configuration.
MPC3042A_set_softlimit_data( ) to setup the coordinate data of limit.
MPC3042A_readback_softlimit_data( ) to read back preset data.
MPC3042A_enable_softlimit( ) to enable / disable software limit function.
MPC3042A_readback_enable_softlimit( ) to read back configuration.
MPC3042A_read_softlimit_flag( ) to read the software limit flag for verifying.

- **MPC3042A_config_softlimit**

Format : **u32 status = MPC3042A_config_softlimit(u8 CardID, u8 axis, u8 source_sel, u8 SL_action)**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X 1: Y
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

- **MPC3042A_readback_config_softlimit**

Format : **u32 status = MPC3042A_readback_config_softlimit(u8 CardID, u8 axis, u8* source_sel, u8* SL_action)**

Purpose: Read back the software limit parameter.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X 1: Y

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

- **MPC3042A_set_softlimit_data**

Format : **u32 status = MPC3042A_set_softlimit_data(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 rotary switch
axis	u8	0: X 1: Y
P_limit	i32	soft limit of positive direction (-134,217,728 ≦P_limit≦ +134,217,727)
N_limit	i32	soft limit of negative direction (-134,217,728 ≦N_limit≦ +134,217,727)

- **MPC3042A_readback_softlimit_data**

Format : **u32 status = MPC3042A_readback_softlimit_data(u8 CardID, u8 axis, i32* P_limit, i32* N_limit)**

Purpose: To read back the coordinate of software limit.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X 1: Y

Output:

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

- **MPC3042A_enable_softlimit**

Format : **u32 status = MPC3042A_enable_softlimit(u8 CardID, u8 axis, u8 ON_OFF)**

Purpose: To enable / disable software limit.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X 1: Y
ON_OFF	u8	0: disable 1: enable

- **MPC3042A_readback_enable_softlimit**

Format : **u32 status = MPC3042A_readback_enable_softlimit(u8 CardID, u8 axis, u8* ON_OFF)**

Purpose: Read back the status of enable / disable software limit.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X 1: Y

Output:

Name	Type	Description
ON_OFF	u8	0: disable 1: enable

- **MPC3042A_read_softlimit_flag**

Format : **u32 status = MPC3042A_read_softlimit_flag(u8 CardID, u8 axis, u8 *P_limit_flag, u8 *N_limit_flag)**

Purpose: To read back software limit flag for verification of events.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X 1: Y

Output:

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

8.17 Manual pulser function

Manual pulser (pulse handler) is used for manual adjust of motion position. It is another type of encoder, normally it has 100 pulse per revolution. A ideal pulse handler control function will run motion to follows the pulse handler speed without losing any incoming pulses.

MPC3042A provides an integrated function to do the speed and position control of pulse handler input. If you use pulse handler as a manual input device, you can map the pulse handler to the motion axis by configure the operating mode of the pulse handler with:

MPC3042A_config_pulser_mode( )

MPC3042A_readback_pulser_mode( ) to read back configuration.

The compound tracking speed and position function can be set by:

MPC3042A_set_pulser_Map( ) and then enable the motion function and multiple rate by:

MPC3042A_enable_pulser_motion( ), then it will track the speed and position of incoming pulses.

If you do not need the compound function, to operate as manual speed control,

MPC3042A_run_pulser_Vmove( ) will do, and for position mode, use

MPC3042A_run_pulser_Pmove( )

Concerning the pulse handler input counter, use

MPC3042A_set_pulser_counter( ) to set pulse counter, and

MPC3042A_read_pulser_counter( ) to read back the counter value.

- **MPC3042A_config_pulser_mode**

Format : **u32 status = MPC3042A_config_pulser_mode(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 rotary switch
axis	u8	0: X 1: Y
pulser_mode	u8	b3~b0: 0:multiply by 1 and up count while phase A lead phase B 1:multiply by 2 and up count while phase A lead phase B 2:multiply by 4 and up count while phase A lead phase B 3:up count while phase A input rising (as CW) down count while rising of phase B input (as CCW) b7~b4: 0: no debounce 1: debounce, filter out pulse higher than 6.5M
direction	u8	override the default direction 0: as default direction 1: invert the direction

- **MPC3042A_readback_pulser_mode**

Format : **u32 status = MPC3042A_readback_pulser_mode(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 rotary switch
axis	u8	0: X 1: Y

Output:

Name	Type	Description
pulser_mode	u8	b3~b0: 0:multiply by 1 and up count while phase A lead phase B 1:multiply by 2 and up count while phase A lead phase B 2:multiply by 4 and up count while phase A lead phase B 3:up count while phase A input rising (as CW) down count while rising of phase B input (as CCW b7~b4: 0: no debounce 1: debounce, filter out pulse higher than 6.5M
direction	u8	0: as default direction 1: invert the direction

- **MPC3042A_set_pulser_Map**

Format : **u32 status = MPC3042A_set_pulser_Map(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 rotary switch
axis	u8	0: Motion axis is X 1: Motion axis is Y
Map_source	u8	0: Pulse handler in X axis 1: Pulse handler in Y axis
Direction	u8	0: rotate same direction with pulse handler input 1: rotate counter direction with pulse handler input

- **MPC3042A_enable_pulser_motion**

Format : **u32 status = MPC3042A_enable_pulser_motion(u8 CardID, u8 axis, u8 enable, u16 Multiple)**

Purpose: To enable/disable pulse handler function and the multiple rate

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: Motion axis is X 1: Motion axis is Y
enable	u8	0: disable 1: enable
Multiple	u16	The number of pulse output to motion axis for an unit of pulse handler input.

Notes:

1. This function can only be used in Windows 2000 P3 800MHz and grade-up system.
2. Before using MPC3042A_enable_pulser_motion( ) function, you must confirm the previous motion is completed. You can check it by the value of ret_flag which is returned by calling MPC3042A_read_motion_status( ) and set check_factor=0.
3. Be sure to disable pulse handler function to stop the pulse handler function before calling any motion command.

- **MPC3042A_run_pulser_Vmove**

Format : **u32 status = MPC3042A_run_pulser_Vmove(u8 CardID,u8 axis,i32 Maxspeed)**

Purpose: To command velocity motion mode and the speed follows the pulse handler input.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X 1: Y
Maxspeed	i32	pps, the maximum pulse output that follows pulse handler input. (0 ≦Maxspeed≦ 6553500)

- **MPC3042A_run_pulser_Pmove**

Format : **u32 status = MPC3042A_run_pulser_Pmove(u8 CardID, u8 axis, i32 Position, u8 posi_mode, i32 Maxspeed)**

Purpose: To command position motion mode and the speed and pulse output follows the pulse handler input, the final position is assigned at parameter Position.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X 1: Y
Position	i32	final position of position move function (-134,217,728 ≦Position≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
Maxspeed	i32	pps, the maximum pulse output that follows pulse handler input. (0 ≦Maxspeed≦ 6553500)

- **MPC3042A_set_pulser_counter**

Format : **u32 status = MPC3042A_set_pulser_counter(u8 CardID, u8 axis, i32 counter_value)**

Purpose: To set the pulse counter value.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X 1: Y
counter_value	i32	pulse counter value to be set (-134,217,728 ≦counter_value≦ 134,217,727)

- **MPC3042A_read_pulser_counter**

Format : **u32 status = MPC3042A_read_pulser_counter(u8 CardID,u8 axis,**
i32 *counter_value)

Purpose: To read the pulse counter value.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X 1: Y

Output:

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

8.18 Multi-function feedback counter

MPC3042A also provide feedback counters (each axis has a feedback counter on card), which also have associate functions such as comparator function, external trigger latch function. Before you use the counter, you must setup the counter input mode (refer ***MPC3024A_set_pulse_inmode)***) then use the following functions :

MPC3042A_read_FB_counter( ) to read counter value.

MPC3042A_set_FB_counter( ) to preset the counter value.

If you have configure latch input function (MPC3042A_config_LTC_PIN( )), use

MPC3042A_read_FBcounter_latch_value( ) to read the latched counter value.

If you have configure compare output function (MPC3042A_config_CMP_OUT( )), use

MPC3042A_config_comparator_out( ) to configure the compare output mode.

MPC3042A_readback_comparator_out( ) to read back configuration.

MPC3042A_set_comparator_data( ) to preset the value to the comparator.

MPC3042A_readback_comparator_data( ) to read back preset value.

MPC3042A_read_compare_flag( ) to read compare out flag for verifying the active state of the function.

- **MPC3042A_read_FB_counter**

Format : **u32 status = MPC3042A_read_FB_counter(u8 CardID, u8 axis, i32 *value)**

Purpose: To read the encoder feedback counter value.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X 1: Y

Output:

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

- **MPC3042A_set_FB_counter**

Format : **u32 status = MPC3042A_set_FB_counter(u8 CardID, u8 axis, i32 value)**

Purpose: To preset the feedback counter value.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X 1: Y
value	i32	pulse counter value (-134,217,728 ≦value≦ 134,217,727)

- **MPC3042A_read_FBcounter_latch_value**

Format : **u32 status = MPC3042A_read_FBcounter_latch_value(u8 CardID, u8 axis, i32 *value)**

Purpose: To read the latched value of feedback counter.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X 1: Y

Output:

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

Note:

You have to configure latch input function (MPC3042A_config_LTC_PIN( )) properly.

- **MPC3042A_config_comparator_out**

Format : **u32 status = MPC3042A_config_comparator_out(u8 CardID,u 8 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 rotary switch
axis	u8	0: X 1: Y
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

Note:

You must have configured compare output function (MPC3042A_config_CMP_OUT( )).

- **MPC3042A_readback_comparator_out**

Format : **u32 status = MPC3042A_readback_comparator_out(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 rotary switch
axis	u8	0: X 1: Y

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

- **MPC3042A_set_comparator_data**

Format : **u32 status = MPC3042A_set_comparator_data(u8 CardID, u8 axis, i32 cmp_data)**

Purpose: To preset the comparator value.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X 1: Y
cmp_data	i32	comparator value to be preset (-134,217,728 ≦cmp_data≦ 134,217,727)

- **MPC3042A_readback_comparator_data**

Format : **u32 status = MPC3042A_readback_comparator_data(u8 CardID, u8 axis, i32* cmp_data)**

Purpose: Read back the preset comparator value.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X 1: Y

Output:

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

- **MPC3042A_read_compare_flag**

Format : **u32 status = MPC3042A_read_compare_flag(u8 CardID, u8 axis, u8 *cmp_flag)**

Purpose: To read back the compare flag.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
axis	u8	0: X 1: Y

Output:

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

8.19 PWM DA

For some application needs one axis speed control, to use FVC-01 (pulse to voltage module) is one of the solutions. On MPC3042A card, it provides an extra PWM DA channel, the range is from 0Vdc to 10Vdc unipolar at 8 bit resolution. It is suitable for speed control (such as spindle speed control). The connector for PWM DA is JM1.

Use

MPC3042A_out_PWM_DA( ) to control the output voltage.

- **MPC3042A_out_PWM_DA**

Format : u32 status = MPC3042A_out_PWM_DA(u8 CardID, u16 DA_value)

Purpose: Output the value to PWM DA.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by rotary switch
DA_value	u16	0~255 data, DA value will be 0Vdc ~ 10Vdc

8.20 Interrupt function

Sometimes you want your application to take care of the motion while special event occurs, interrupt function is the right choice. First of all you must hook the interrupt service routine to the driver:

MPC3042A_link_IRQ_process( ).

On MPC3042A card there are many source to generate interrupt, select the interrupt source by

MPC3042A_set_INT_source( ), use

You can mask off or temporary mask off unwanted hardware of the interrupt source by,

MPC3042A_set_INT_mask( ) will do and your program is waiting a interrupt to service.

To read the interrupt status by:

MPC3042A_read_INT_status( ) 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:

MPC3042A_clear_INT_status( )

For the PI control function block DIO and timer interrupt can be controlled by:,

MPC3042A_IRQ_mask_set( ) to mask off the undesired interrupt source (DIO or timer).

MPC3042A_IRQ_mask_read( ) to read back the mask. To read back the IRQ status,

MPC3042A_IRQ_status_read( ) will do and it also clears the interrupt status

Finally, you can enable or disable the interrupt by:

MPC3042A_enable_IRQ( ) to enable the IRQ function..

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

MPC3042A_disable_IRQ( ) to release the resource.

- **MPC3042A_link_IRQ_process**

Format : u32 status = MPC3042A_link_IRQ_process (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

MPC3042A_set_INT_source

Format : **u32 status = MPC3042A_set_INT_source(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:

<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</td></tr>
<tr><td rowspan="20">REST_source_sel: error interrupt source select</td><td rowspan="20">u32</td><td colspan="3">any bit of the following set to “1” means if the error 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>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="15">RIST_source_sel: event interrupt source select</td><td rowspan="15">u32</td><td colspan="3">any bit of the following set to “1” means if the event 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>IRC4</td><td>Compare- method satisfied</td></tr>
<tr><td>bit12</td><td>IRC5</td><td>Compare (compare+) method satisfied</td></tr>
</table>

		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 MPC3042A card, if you do not use interrupt function please use MPC3042A_set_event_factor( ) instead.

- **MPC3042A_set_INT_mask**

Format : **u32 status = MPC3042A_set_INT_mask(u8 CardID, u8 axis, u8 on_off);**

Purpose: To set the interrupt mask of designated axis.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X 1: Y
on_off	u8	0: disable 1: enable

MPC3042A_read_INT_status

Format : **u32 status = MPC3042A_read_INT_status(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 1: Y

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	
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	IRC4	Compare- method satisfied
		bit12	IRC5	Compare (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 MPC3042A card, if you do not use interrupt function please use MPC3042A_read_event_flag( ) and MPC3042A_read_error_flag( ) instead.

MPC3042A_clear_INT_status

Format : **u32 status = MPC3042A_clear_INT_status(u8 CardID, u8 axis,**
u32 REST, u32 RIST);

Purpose: To reset 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</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>
</table>

RIST	u32	while any of the following bit set to “1” means means to reset the corresponding bit.		
		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	IRC4	Compare- method satisfied
		bit12	IRC5	Compare (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:

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.

- **MPC3042A_IRQ_mask_set**

Format : **u32 status = MPC3042A_IRQ_mask_set (u8 CardID,u8 source, u8 mask)**

Purpose: Mask off interrupt source of DIO IN07~IN00 or timer

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
source	u8	0: DIO block 1: timer block
mask	u8	DIO block: b7: set 1, IN07 can generate interrupt else can not. … b0: set 1, IN00 can generate interrupt else can not. Timer block: b0=1, enable timer cross zero to generate interrupt, else disable.

- **MPC3042A_IRQ_mask_read**

Format : **u32 status = MPC3042A_IRQ_mask_read (u8 CardID,u8 source,u8 *mask)**

Purpose: read back interrupt mask of IN07~IN00 or timer

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
source	u8	0: DIO block 1: timer/counter block

Output:

Name	Type	Description
mask	u8	DIO block: b7: set 1, IN07 can generate interrupt else can not. … b0: set 1, IN00 can generate interrupt else can not. Timer block: b0=1, enable timer cross zero to generate interrupt, else disable.

- **MPC3042A_IRQ_status_read**

Format : **u32 status = MPC3042A_IRQ_status_read(u8 CardID,u8 source,**
u8 *Event_Status)

Purpose: To read back the interrupt status to identify the source

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by Rotary SW
source	u8	0: DIO block 1: timer block

Output:

Name	Type	Description
Event_Status	u8	DIO block: b7: =1, IN07 generates interrupt … b0: =1, IN00 generates interrupt Timer block: b0: =1 TIMER generates interrupt

Note:

1. Status read back will also clear the on board status register.
2. The status will reflect the on board digital input or timer count up status are irrelevant to the IRQ_MASK

- **MPC3042A_enable_IRQ**

Format : **u32 status = MPC3042A_enable_IRQ(u8 CardID, HANDLE *phEvent);**

Purpose: To enable the interrupt function.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Output:

Name	Type	Description
phEvent	HANDLE	returned event handle

- **MPC3042A_disable_IRQ**

Format : **u32 status = MPC3042A_disable_IRQ(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

8.21 Software key function

From the dll version 3.0 and later, we remove the software key function owing to some customers complained about the card locked on some unknown occasion. We only remain the functions to comply with the existing programs but the returned value always true.

Since MPC3042A is a general purpose card, anyone who can buy from JS automation corp. or her distributors. Your program is the fruit of your intelligence, un-authorized copy maybe prevent by the security function enabled.

You can use

MPC3042A_set_password( ) to set password and start the security function.

MPC3042A_change_password( ) to change it.

If you don't want to use security function after the password being setup,

MPC3042A_clear_password( ) will reset to the virgin state.

Once the password is set, any function call of the dll's (except for the security functions) will be blocked until the

MPC3042A_unlock_security( ) unlock the security.

You can also use

MPC3042A_read_security_status( ) to check the current status of security.

- **MPC3042A_set_password**

Format : u32 status = MPC3042A_set_password(u8 CardID,u16 password[5]);

Purpose: To set password and if the password is not all "0", security function will be enabled.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
password[5]	u16	Password, 5 words

Note on password:

If the password is all "0", the security function is disabled.

- **MPC3042A_change_password**

Format : u32 status = MPC3042A_change_password(u8 CardID,u16 Oldpassword[5], u16 password[5]);

Purpose: To replace old password with new password.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
Oldpassword [5]	u16	The previous password
password[5]	u16	The new password to be set

- **MPC3042A_clear_password**

Format : **u32 status = MPC3042A_clear_password(u8 CardID,u16 password[5]);**

Purpose: To clear password, to set password to all “0”, i.e. disable security function.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
password[5]	u16	The password previous set

- **MPC3042A_unlock_security**

Format : **u32 status = MPC3042A_unlock_security(u8 CardID,u16 password[5]);**

Purpose: To unlock security function and enable the further operation of this card.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
password[5]	u16	The password previous set

- **MPC3042A_read_security_status**

Format : **u32 status = MPC3042A_read_security_status(u8 CardID,u8 *lock_status,**
u8 *security_enable);

Purpose: To read security status for checking if the card security function is unlocked.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Output:

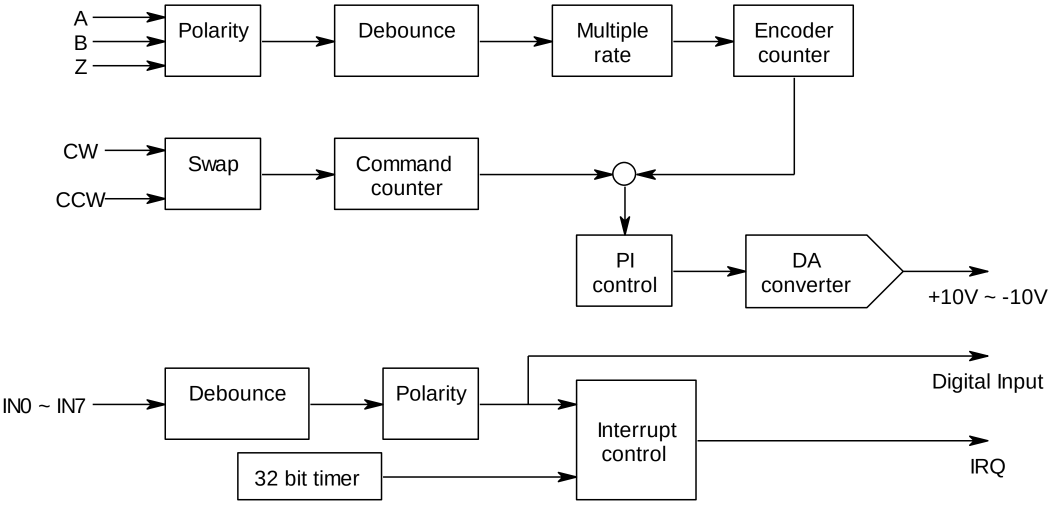
Name	Type	Description
lock_status	u8	0: security unlocked 1: locked 2: dead lock (must return to original maker to unlock)
security_enable	u8	0: security function disabled 1: security function enabled

Note on security status:

The security should be unlocked before using any other function of the card, and any attempt to unlock with the wrong passwords more than 10 times will cause the card at dead lock status. Any further operation even with the correct password will not unlock the card. The only way is to send back to the card distributor or the original maker to unlock to virgin state.

8.22 Close loop PI control and associate functions (MPC3042A only)

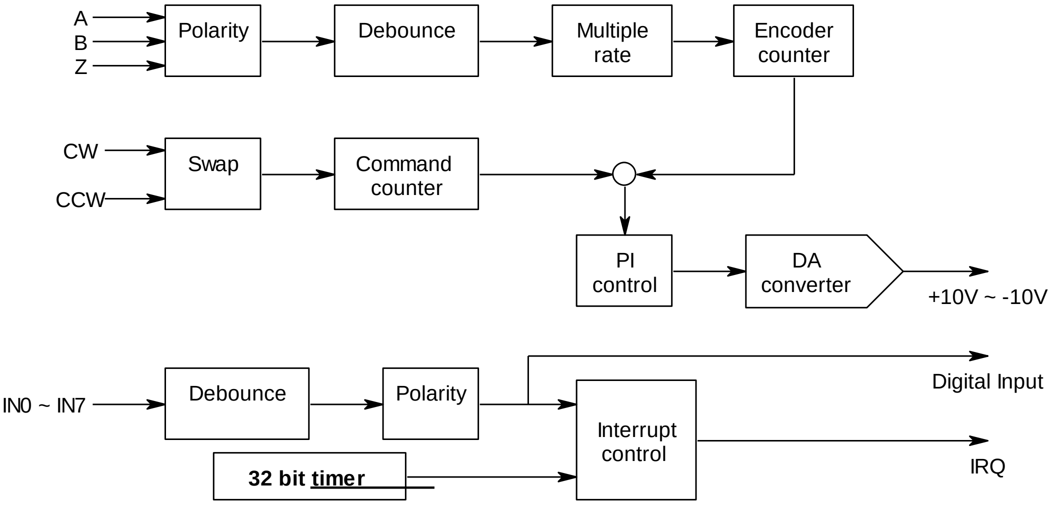
The MPC3042A has build-in digital hardware of PI control and 17 bit DA converter to do the close loop motion control.



The above diagram is the digital PI control block and the timer. MPC3042A provides easy to use function for the functions.

8.23 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

MPC3042A_timer_set( ) and start by

MPC3042A_timer_start( ) and stop by

MPC3042A_timer_stop( )

The timer interrupt can be reached by:

MPC3042A_IRQ_mask_set( ) (refer 8.20 Interrupt function)

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

MPC3042A_TC_set( ) to set registers and use

MPC3042A_TC_read( ) to read back settings.

- **MPC3042A_timer_set**

Format : u32 status = MPC3042A_timer_set (u8 CardID, u32 time_constant)

Purpose: To setup timer operation mode or update timer

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
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.

- **MPC3042A_timer_start**

Format : **u32 status = MPC3042A_timer_start (u8 CardID)**

Purpose: To start timer operation mode

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

- **MPC3042A_timer_stop**

Format : **u32 status = MPC3042A_timer_stop (u8 CardID)**

Purpose: To stop timer operation mode

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

- **MPC3042A_TC_set**

Format : **u32 status=MPC3042A_TC_set (u8 CardID, u8 index, u32 data)**

Purpose: To load data to timer related registers

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
index	u8	0: TC_CONTROL 1: PRELOAD 2: TIMER
data	u32	For TC_CONTROL 0: stop timer operation 1: timer run For PRELOAD or TIMER Data is the constant to be load

Note:

PRELOAD is the register for timer to re-load, the value will be valid while timer count to zero and reload the data.

- **MPC3042A_TC_read**

Format : **u32 status=MPC3042A_TC_read (u8 CardID,u8 index,u32 *data)**

Purpose: To read data from timer related registers

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
index	u8	0: TC_CONTROL 1: PRELOAD 2: TIMER

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	0~1	0:timer stops operation 1: timer runs
1	PRELOAD	1~0xffffffff	timer preload value
2	TIMER	1~0xffffffff	Timer value on the fly

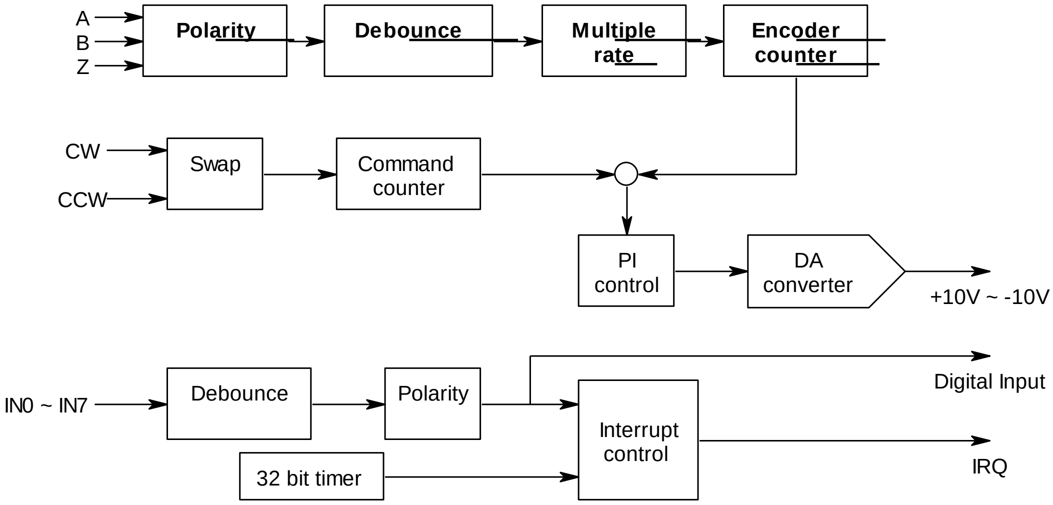
Note:

For example, you want to watch the timer counting on the fly, use

MPC3042A_TC_read (CardID, index, *data) //CardID as you assign, index=2

To read back the timer value.

8.24 Encoder counter function (only valid for MPC3042A)



The encoder input comes from the servo motor feedback which is the important information of speed and position feedback. MPC3042A 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.

MPC3042A_encoder_mode_set( ) will set up the environment as you need.

MPC3042A_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:

MPC3042A_encoder_polarity_set( ) and read back to verify by:

MPC3042A_encoder_polarity_read( )

To read the instantaneous value of the encoder state, apply

MPC3042A_encoder_status_read( )

- **MPC3042A_encoder_mode_set**

Format : **u32 status=MPC3042A_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
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

- **MPC3042A_encoder_mode_read**

Format : **u32 status=MPC3042A_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

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

- **MPC3042A_encoder_polarity_set**

Format : **u32 status=MPC3042A_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
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.

- **MPC3042A_encoder_polarity_read**

Format : **u32 status=MPC3042A_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

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.

- **MPC3042A_encoder_status_read**

Format : **u32 status=MPC3042A_encoder_status_read (u8 CardID, u8 axis, u16 *state)**

Purpose: To read data from encoder status register

Parameters:

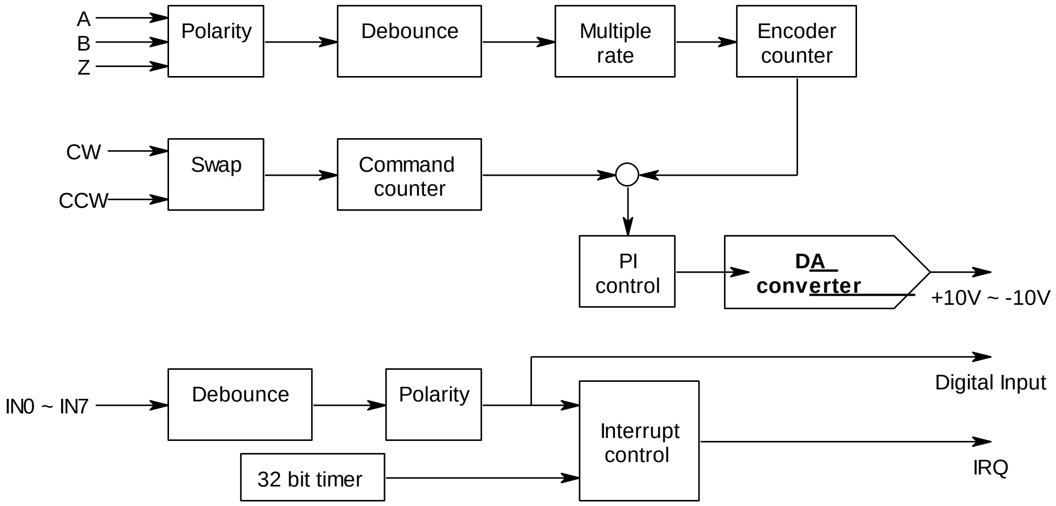
Input:

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

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

8.25 Digital to analog converter (only valid for MPC3042A)



The DA can be used as stand alone application, take it as a 17bit -10V to +10V DA to control the device.

Use

MPC3042A_DA_set( ) to do DA conversion.

MPC3042A_DA_read( ) to read back the digital command value.

- **MPC3042A_DA_set**

Format : u32 status = MPC3042A_DA_set(u8 CardID,u8 axis, i32 data)

Purpose: To set MPC3042A card's DA data.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0:X axis 1:Y axis
data	i32	Set conversion data. -65536 ~ 65535 (-10V ~ 10V)

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

- **MPC3042A_DA_read**

Format : **u32 status = MPC3042A_DA_read (u8 CardID,u8 axis,i32 *data)**

Purpose: To read back data of DA .

Parameters:

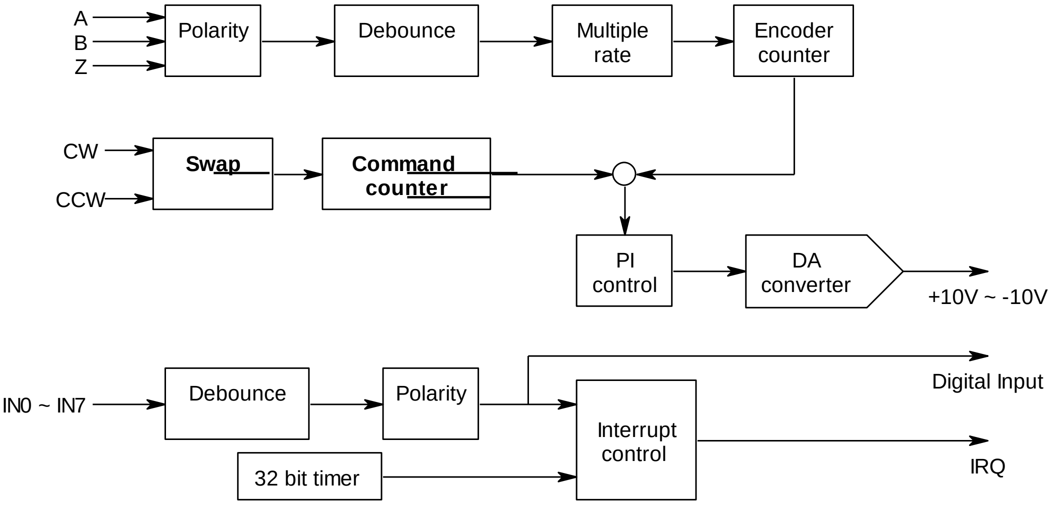
Input:

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

Output:

Name	Type	Description
data	i32	Data read back -65536 ~ 65535 (-10V ~ 10V)

8.26 Pulse command input from internal logic (only valid for MPC3042A)



The command pulse counter is used as reference to compare with the feed back encoder counter. The command pulse is easily swapped to meet the motor direction and encoder.

MPC3042A_pulse_swap_set( ) to swap the input command signal.

MPC3042A_pulse_swap_read( ) to read back the settings.

- **MPC3042A_pulse_swap_set**

Format : **u32 status=MPC3042A_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
swap	u8	0: normal 1: swap (exchange cw with ccw)

- **MPC3042A_pulse_swap_read**

Format : **u32 status=MPC3042A_pulse_swap_read (u8 CardID, u8 axis, u8 *swap)**

Purpose: To read data from command pulse swap register

Parameters:

Input:

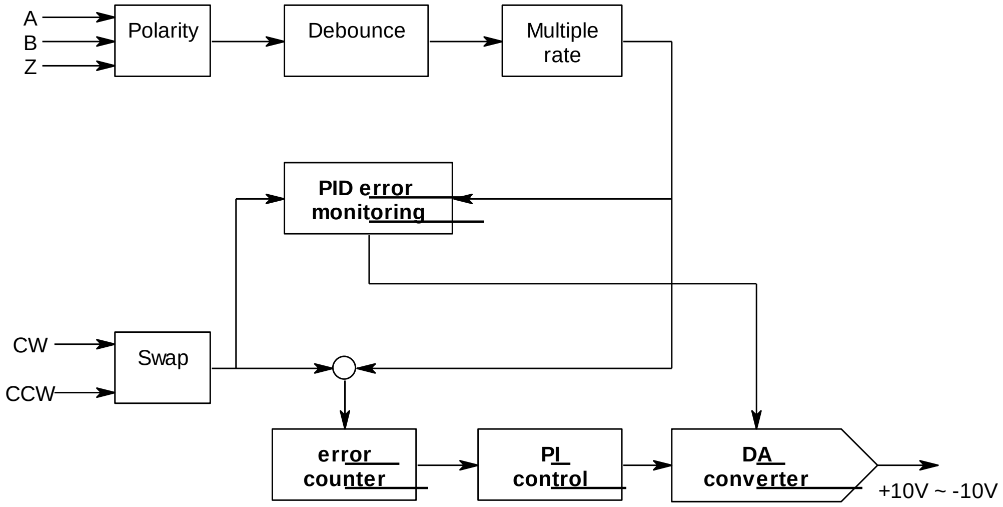
Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: X axis 1: Y axis

Output:

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

8.27 PI control (only valid for MPC3042A)

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 MPC3042A provide 2 axis independent PI mode, it is used for the normal motion control. The other mode is tracking mode, the Y axis will following the X axis motion (Y axis tracks the X axis). Define your motion control mode by:

MPC3042A_PID_control_mode_set( ) and read back to verify by

MPC3042A_PID_control_mode_read( )

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

MPC3042A_PID_set( ) to set up parameters.

MPC3042A_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. 7.3 MPC3042A_set_pulse_inmode( ))and the command pulse swap function and confirmed the feedback and command pulse are in the same direction, you can check the feedback loop by:

MPC3042A_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

MPC3042A_PID_start( ) to run in analog command mode.

If you do not run anymore,

MPC3042A_PID_stop( ) to stop the PI control and 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.

MPC3042A_PID_error_detector_set( ) to enable / disable monitoring function and read back to verify by:

MPC3042A_PID_error_detector_read( )

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

MPC3042A_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.

MPC3042A_counter_simultaneous_read( ) provide the special function for you to check the performance. If the performance is not meet your requirement, please try to adjust the PI parameters to improve the performance.

- **MPC3042A_PID_control_mode_set**

Format : **u32 status = MPC3042A_PID_control_mode_set(u8 CardID, u8 enable)**

Purpose: To set MPC3042A card’s PID closed loop as independent or tracking mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
enable	u8	0: independent mode (both X,Y are as master axis) 1: dependent mode (X as master and Y as slave to track the motion)

- **MPC3042A_PID_control_mode_read**

Format : **u32 status = MPC3042A_PID_control_mode_read(u8 CardID, u8 *enable)**

Purpose: To read back MPC3042A 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
enable	u8	0: independent mode (both X,Y are as master axis) 1: dependent mode (X as master and Y as slave to track the motion)

- **MPC3042A_PID_set**

Format : **u32 status = MPC3042A_PID_set (u8 CardID, u8 axis, u16 P,u16 I)**

Purpose: To set MPC3042A card’s PID parameters.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: input port 1: output port
P	u16	P Gain, range 1~4095
I	u16	I Gain, 1~4095 (in mili-second) I Gain=0, no integration function

- **MPC3042A_PID_read**

Format : **u32 status = MPC3042A_PID_read (u8 CardID, u8 axis, u16 *P, u16 *I)**

Purpose: To set MPC3042A card’s PID

Parameters:

Input:

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

Output:

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

- **MPC3042A_PID_error_counter_read**

Format : **u32 status = MPC3042A_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

Output:

Name	Type	Description
value	i32	error counter data

- **MPC3042A_PID_start**

Format : **u32 status = MPC3042A_PID_start (u8 CardID, u8 axis)**

Purpose: Run MPC3042A 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

Note:

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

In the PID control mode, you can not use MPC3042A_pulse_swap_set( ), MPC3042A_DA_set( ), MPC3042A_encoder_mode_set( ), MPC3042A_encoder_polarity_set( ).

- **MPC3042A_PID_stop**

Format : **u32 status = MPC3042A_PID_stop (u8 CardID, u8 axis)**

Purpose: Stop MPC3042A 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

- **MPC3042A_PID_error_detector_set**

Format : **u32 status = MPC3042A_PID_error_detector_set(u8 CardID, u8 axis, u8 enable)**

Purpose: Enable or disable error detector.

Parameters:

Input:

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

- **MPC3042A_PID_error_detector_read**

Format : **u32 status = MPC3042A_PID_error_detector_read(u8 CardID, u8 axis,u8 *state, u8 *enable)**

Purpose: Read back MPC3042A 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

Output:

Name	Type	Description
state	u8	b0: =1, encoder A phase error b1: =1, encoder B phase error b2: =1, DA run over voltage without pulse command b3: =1, A or B phase undetermined error
enable	u8	b0: =1, error detector enabled =0, error detector disabled

- **MPC3042A_PID_error_detector_clear**

Format : **u32 status = MPC3042A_PID_error_detector_clear(u8 CardID, u8 axis)**

Purpose: Clear error detector to resume monitoring function.

Parameters:

Input:

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

- **MPC3042A_counter_simultaneous_read**

Format : **u32 status = MPC3042A_counter_simultaneous_read(u8 CardID,i32 *x_counter, i32 *y_counter)**

Purpose: Read back MPC3042A card’s PID error counter 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

8.28 Error conditions

These error types may indicate an internal hardware problem on the board. Error Codes summary contains a detailed listing of the error status returned by MPC3042A functions.

9. Dll list

	Function Name	Description
1.	MPC3042A_initial( )	MPC3042A Initial
2.	MPC3042A_close( )	MPC3042A Close
3.	MPC3042A_init_card( )	Initialize parameters and auxiliary function to default value
4.	MPC3042A_info( )	Get the address, card type of designated card ID
5.	MPC3042A_save_config2_file( )	Save configuration data to file
6.	MPC3042A_load_config_from_file( )	Load configuration data from file
7.	MPC3042A_set_pulse_outmode( )	Configure the pulse output mode
8.	MPC3042A_readback_pulse_outmode( )	Read back configuration of pulse output mode
9.	MPC3042A_set_pulse_inmode( )	Configure the multiple rate and the encoder input
10.	MPC3042A_readback_pulse_inmode( )	Read back configuration of pulse input mode
11.	MPC3042A_config_SD_PIN( )	Configure slow down input
12.	MPC3042A_readback_SD_PIN( )	Read back configuration of SD pin
13.	MPC3042A_config_PCS_PIN( )	Configure PCS(position change start) input
14.	MPC3042A_readback_PCS_PIN( )	Read back configuration of PCS pin
15.	MPC3042A_config_INP_PIN( )	Configure INP (in position) input
16.	MPC3042A_readback_INP_PIN( )	Read back configuration of INP pin
17.	MPC3042A_config_ERC_PIN( )	Configure ERC (error counter clear) output
18.	MPC3042A_readback_ERC_PIN( )	Read back configuration of ERC pin
19.	MPC3042A_config_ALM_PIN( )	Configure ALM (alarm) input
20.	MPC3042A_readback_ALM_PIN( )	Read back configuration of ALM pin
21.	MPC3042A_config_LTC_PIN( )	Configure LTC (latch) input
22.	MPC3042A_readback_LTC_PIN( )	Read back configuration of LTC pin
23.	MPC3042A_config_CMP_OUT( )	Configure CMP (compare) output
24.	MPC3042A_readback_CMP_OUT( )	Read back configuration of CMP_OUT
25.	MPC3042A_config_EL_MODE( )	Configure LS(EL) (over travel) stop mode
26.	MPC3042A_readback_EL_MODE( )	Read back configuration for LS(EL)
27.	MPC3042A_set_HOME_pin_logic( )	Configure HOME (ORG) polarity
28.	MPC3042A_readback_HOME_pin_logic( )	Read back configuration for HOME (ORG) pin
29.	MPC3042A_debounce_set( )	Set debounce time of motion related digital input
30.	MPC3042A_debounce_read( )	Read back debounce time of motion related digital input
31.	MPC3042A_set_EZ_pin_logic( )	Configure EZ (zero phase) polarity
32.	MPC3042A_readback_EZ_pin_logic( )	Read back configuration of EZ (zero phase) polarity
33.	MPC3042A_write_output_point( )	Write output
34.	MPC3042A_read_point_status( )	Read input status
35.	MPC3042A_DIO_polarity_set( )	Set DIO port polarity
36.	MPC3042A_DIO_polarity_read( )	Read back DIO port polarity
37.	MPC3042A_DIO_debounce_set( )	Set DIO debounce time
38.	MPC3042A_DIO_debounce_read( )	Read back DIO debounce time
39.	MPC3042A_DIO_set( )	Set DIO port
40.	MPC3042A_DIO_read( )	Read back DIO port
41.	MPC3042A_DIO_bit_set( )	Set DIO bit
42.	MPC3042A_DIO_bit_read( )	Read back DIO bit

43.	MPC3042A_fix_speed_range( )	Set the maximum allowable speed
44.	MPC3042A_unfix_speed_range( )	Release the limit of maximum allowable speed
45.	MPC3042A_T_velocity_move( )	Velocity mode move at trapezoidal profile
46.	MPC3042A_S_velocity_move( )	Velocity mode move at S curve profile
47.	MPC3042A_S1_velocity_move( )	Velocity mode move at S curve profile
48.	MPC3042A_velocity_change( )	To change speed on motion
49.	MPC3042A_dec_stop( )	Velocity mode, deceleration to stop
50.	MPC3042A_imd_stop( )	Velocity mode, immediate stop
51.	MPC3042A_emg_stop( )	Velocity mode, emergency stop all axes
52.	MPC3042A_read_speed( )	Read the current speed
53.	MPC3042A_config_home_mode( )	Select the desired homing mode
54.	MPC3042A_start_homing( )	To execute homing
55.	MPC3042A_set_current_position( )	Setup the coordinate of current point
56.	MPC3042A_read_current_position( )	Read the coordinate of current point
57.	MPC3042A_start_origin_search_homing( )	Seek home (ORG) limit switch automatically and correct the position
58.	MPC3042A_backlash_comp( )	Setup backlash compensation
59.	MPC3042A_readback_backlash_comp( )	Read back configuration of backlash compensation
60.	MPC3042A_T_curve_position_move( )	Point to point move at trapezoidal acc/dec profile
61.	MPC3042A_S_curve_position_move( )	Point to point move at S curve profile
62.	MPC3042A_S1_curve_position_move( )	Point to point move at S curve profile
63.	MPC3042A_position_change( )	Change target position while the point to point motion is running
64.	MPC3042A_suppress_vibration( )	Setup vibration suppression mode
65.	MPC3042A_readback_suppress_vibration( )	Read back parameters of vibration suppression mode
66.	MPC3042A_T_curve_move_LINE2( )	Two axes linear interpolation at trapezoidal profile
67.	MPC3042A_S_curve_move_LINE2( )	Two axes linear interpolation at S curve profile
68.	MPC3042A_S1_curve_move_LINE2( )	Two axes linear interpolation at S curve profile
69.	MPC3042A_OnLine_T_curve_change( )	Online motion parameters change for 1 axis
70.	MPC3042A_OnLine_T_curve_change_LINE2( )	Online motion parameters change for 2 axis
71.	MPC3042A_config_compare_start_motion( )	To configure the compare source and method of synchronous start
72.	MPC3042A_set_compare_start_data( )	To configure the compared data
73.	MPC3042A_T_curve_wait_Cmpstart( )	To setup the T profile motion and wait for synchronous start signal to take action.
74.	MPC3042A_S_curve_wait_Cmpstart( )	To setup the S profile motion and wait for synchronous start signal to take action.
75.	MPC3042A_S1_curve_wait_Cmpstart( )	To setup the S1 profile motion and wait for synchronous start signal to take action.
76.	MPC3042A_read_compare_start_flag( )	To read the compare start flag.
77.	MPC3042A_ARC2_center_move( )	Circular interpolation with the circle center and end position parameter as arc trajectory
78.	MPC3042A_T_ARC2_center_move( )	Circular interpolation with the circle center and end position and the T type acceleration/deceleration as arc trajectory
79.	MPC3042A_S1_ARC2_center_move( )	Circular interpolation with the circle center and end position and the S type acceleration/deceleration for arc trajectory

80.	MPC3042A_ARC2_3P_move( )	Circular interpolation with current point and the other 2 points for the arc trajectory
81.	MPC3042A_T_ARC2_3P_move( )	Circular interpolation with current point and the other 2 points and T type the acceleration/deceleration for arc trajectory
82.	MPC3042A_S1_ARC2_3P_move( )	Circular interpolation with S type profile and with current point and the other 2 points for the arc trajectory
83.	MPC3042A_CIR2_3P_move( )	the current position and the middle, end position to make a circle and the circular interpolation pass through the 3 positions.
84.	MPC3042A_T_CIR2_3P_move( )	Circular interpolation with current point and the other 2 points and the T type acceleration/deceleration profile for the circle trajectory
85.	MPC3042A_S1_CIR2_3P_move( )	Circular interpolation with current point and the other 2 points and the S type acceleration/deceleration profile for the circle trajectory
86.	MPC3042A_ARC2_Radius_move( )	the current position and end position to make an arc at designated R.
87.	MPC3042A_T_ARC2_Radius_move( )	the current position and end position to make an arc at designated R with T type acceleration/deceleration profile.
88.	MPC3042A_S1_ARC2_Radius_move( )	the current position and end position to make an arc at designated R with S type acceleration/deceleration profile.
89.	MPC3042A_CIR2_Radius_move( )	the current position and end position to make a cycle at designated R.
90.	MPC3042A_T_CIR2_Radius_move( )	the current position and end position to make a cycle at designated R with T type acceleration/deceleration profile
91.	MPC3042A_S1_CIR2_Radius_move( )	the current position and end position to make a cycle at designated R with S type acceleration/deceleration profile.
92.	MPC3042A_set_continuous_flag( )	Enable / disable the continuous mode
93.	MPC3042A_check_continuous_buffer( )	To check the continuous buffer
94.	MPC3042A_read_motion_status( )	Read the motion status
95.	MPC3042A_OneAxis_restart( )	Single axis restart
96.	MPC3042A_2Axis_restart( )	Two axes restart
97.	MPC3042A_set_event_factor( )	To enable the event for corresponding event source
98.	MPC3042A_read_event_flag( )	To read the event source
99.	MPC3042A_read_error_flag( )	To read the error condition flag
100.	MPC3042A_config_softlimit( )	Configure soft limit
101.	MPC3042A_readback_config_softlimit( )	Read back the software limit parameter
102.	MPC3042A_set_softlimit_data( )	Setup the coordinate data of soft limit
103.	MPC3042A_readback_softlimit_data( )	Read back the coordinate of software limit
104.	MPC3042A_enable_softlimit( )	Enable / disable software limit function
105.	MPC3042A_readback_enable_softlimit( )	Read back the status of enable / disable software limit
106.	MPC3042A_read_softlimit_flag( )	Read the software limit flag for verifying

107.	MPC3042A_config_pulser_mode( )	Configure the operating mode of the pulse handler
108.	MPC3042A_readback_pulser_mode( )	Read back the pulse handler operation mode
109.	MPC3042A_set_pulser_Map( )	Map the pulser input to motion axis
110.	MPC3042A_enable_pulser_motion( )	Enable motion function and multiple rate
111.	MPC3042A_run_pulser_Vmove( )	Operate pulse handler as manual speed control
112.	MPC3042A_run_pulser_Pmove( )	Operate pulse handler as manual position control
113.	MPC3042A_set_pulser_counter( )	Set pulse counter
114.	MPC3042A_read_pulser_counter( )	Read pulse counter
115.	MPC3042A_read_FB_counter( )	Read feedback counter
116.	MPC3042A_set_FB_counter( )	Set feedback counter
117.	MPC3042A_read_FBcounter_latch_value( )	Read feedback counter latched value
118.	MPC3042A_config_comparator_out( )	Configure the compare output mode
119.	MPC3042A_readback_comparator_out( )	Read back the configuration of the compare mode
120.	MPC3042A_set_comparator_data( )	Preset the value to the comparator
121.	MPC3042A_readback_comparator_data( )	Read back the preset comparator value
122.	MPC3042A_read_compare_flag( )	Read compare out flag
123.	MPC3042A_out_PWM_DA( )	Control PWM DA output
124.	MPC3042A_link_IRQ_process( )	Link interrupt service routine
125.	MPC3042A_set_INT_source( )	Select the interrupt source
126.	MPC3042A_set_INT_mask( )	Enable / disable the hardware of the interrupt source
127.	MPC3042A_read_INT_status( )	Read the interrupt event generating source
128.	MPC3042A_clear_INT_status( )	Clear the interrupt event generating source
129.	MPC3042A_IRQ_mask_set( )	Mask off the undesired interrupt source (DIO or timer)
130.	MPC3042A_IRQ_mask_read( )	Read back the mask of the interrupt source (DIO or timer)
131.	MPC3042A_IRQ_status_read( )	Read back the IRQ status (DIO or timer)
132.	MPC3042A_enable_IRQ( )	Enable the IRQ function
133.	MPC3042A_disable_IRQ( )	Disable the IRQ function and release the resource
134.	MPC3042A_set_password( )	Set password and start the security function
135.	MPC3042A_change_password( )	Change password
136.	MPC3042A_clear_password( )	Clear password
137.	MPC3042A_unlock_security( )	Unlock security for further operation
138.	MPC3042A_read_security_status( )	Check the current status of security
139.	MPC3042A_timer_set( )	Set timer constant
140.	MPC3042A_timer_start( )	Start timer function
141.	MPC3042A_timer_stop( )	Stop timer function
142.	MPC3042A_TC_set( )	Set the timer associated registers
143.	MPC3042A_TC_read( )	Read the timer associated registers
144.	MPC3042A_encoder_mode_set( )	Setup encoder counter operating mode
145.	MPC3042A_encoder_mode_read( )	Read back encoder counter operating mode
146.	MPC3042A_encoder_polarity_set( )	Setup encoder polarity
147.	MPC3042A_encoder_polarity_read( )	Read back encoder polarity
148.	MPC3042A_encoder_status_read( )	Read data from encoder status register
149.	MPC3042A_DA_set( )	Do DA data conversion
150.	MPC3042A_DA_read( )	Read back DA data

151.	MPC3042A_pulse_swap_set( )	Setup command pulse swap register
152.	MPC3042A_pulse_swap_read( )	Read back command pulse swap register
153.	MPC3042A_PID_control_mode_set( )	Setup PID control mode
154.	MPC3042A_PID_control_mode_read( )	Read back PID control mode
155.	MPC3042A_PID_set( )	Setup PID parameters
156.	MPC3042A_PID_read( )	Read back PID parameters
157.	MPC3042A_PID_error_counter_read( )	Read back error counter of command and encoder
158.	MPC3042A_PID_start( )	Start PID control
159.	MPC3042A_PID_stop( )	Stop PID control
160.	MPC3042A_PID_error_detector_set( )	Setup PID error counter status parameter
161.	MPC3042A_PID_error_detector_read( )	Read back PID error counter status parameter
162.	MPC3042A_PID_error_detector_clear( )	Setup PID error counter status to clear
163.	MPC3042A_counter_simultaneous_read( )	Simultaneously reads x axis and the y axis feedback counter.

10. MPC3042A Error codes summary

10.1 MPC3042A Error codes table

Error Code	Symbolic Name	Description
0	NO_ERROR	Success, No error.
1	READ_DATA_ERROR	Driver read data error
2	INIT_ERROR	Driver initial error
3	UNLOCK_ERROR	Unlock error
4	LOCK_COUNTER_ERROR	Unlock error over 10 times
5	SET_SECURITY_ERROR	Set security error
6	CHIP_ERROR	Select Chip error.
100	RW_ERROR	Device Read/Write error or no card on the system
101	NO_CARD	No MPC3042A card on the system.
102	DUPLICATE_ID	MPC3042A CardID duplicate error.
103	NOT_INSTALL	Driver not installed or bad installation
300	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
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	End point do not on the circle
311	CIRCLE_OVERWRITE_MIDP	Middle point do not on the circle
400	INDEX_ERROR	TC register index error
401	CONSTANT_ERROR	TC register variable error
402	TC_CONTROL_ERROR	Run timer is error.
501	PORT_ERROR	Function input parameter error. Parameter out of range.
502	POINT_ERROR	Function point parameter error. Parameter out of range.
503	DEBOUNCE_MODE_ERROR	Bad debounce time parameter
601	DA_ERROR	Bad DA parameter
602	PID_ERROR	Bad PID parameter
700	SOURCE_ERROR	Bad source parameter