MPC-3035

4-axis Motion Control Card

Software Manual (V1.3)

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 wdm3035.sys V1.0 , drv3035.dll V1.0, MPC3035.dll V1.1 and up
1.0->1.1	1. Correct the definition of s curve Svacc, SVdec
1.1->1.2	1. For MPC3035.dll V1.4 up. Add ***MPC3035_latch_control*** ***MPC3035_latch_mode*** ***MPC3035_read_latch_flag*** ***MPC3035_read_latch_value***
12->1.3	1. For MPC3035.dll V1.5 up Add compare segment function: ***MPC3035L_write_mask_off*** ***MPC3035L_read_mask_off*** ***MPC3035L_write_segment_control*** ***MPC3035L_read_segment_control*** ***MPC3035L_write_cmp_segment*** ***MPC3035L_read_cmp_segment*** Add X,Y simultaneous latch function ***MPC3035L_write_XY_latch*** ***MPC3035L_read_XY_latch***

Contents

1. How to install the software of MPC3035

1.1 Install the PCI driver

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

In Windows 98/2000/XP system you should: (take win2000 as example)

1. Shut down your PC, 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
   \MPC3035\Software\Win98_2K_XP\MPC3035_Install.exe
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 supplementary documents.

2. Where to find the file you need

Windows98, 2000,XP

In Windows 98, 2000,XP system, the demo program can be setup by \MPC3035\Software\Win98_2K_XP\Install\MPC3035_Install.exe

The directory will be located at

.. / JS Automation /MPC3035/API (header files and VB,VC lib files)

.. / JS Automation /MPC3035/ Driver (backup copy of MPC3035 drivers)

.. / JS Automation /MPC3035/exe (demo program and source code)

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

3. About the MPC-3035 software

MPC3035 software includes a set of dynamic link library (DLL) and system driver that you can utilize to control the I/O card's ports and points separately.

Your MPC3035 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 MPC3035 functions within Windows' operation system environment.

3.1 What you need to get started

To set up and use your MPC3035 software, you need the following:

- MPC3035 software
- MPC3035 hardware
  Main board
  Wiring board (Option)

3.2 Software programming choices

You have several options to choose from when you are programming MPC3035 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 MPC3035 software.

4. MPC3035 Language support

The MPC3035 software library is a DLL used with Windows 98/2000/XP. You can use these DLL with any Windows integrating development environment that can call Windows DLLs.

4.1 Building applications with the MPC3035 software library

The MPC3035 function reference topic contains general information about building MPC3035 applications, describes the nature of the MPC3035 files used in building MPC3035 applications, and explains the basics of making applications using the following tools:

Applications tools

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

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

4.2 MPC3035 Windows libraries

The MPC3035 for Windows function library is a DLL called **MPC3035.dll**. Since a DLL is used, MPC3035 functions are not linked into the executable files of applications. Only the information about the MPC3035 functions in the MPC3035 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 MPC3035 functions in MPC3035.dll.

Header Files and Import Libraries for Different Development Environments		
Development Environment	**Header File**	**Import Library**
Microsoft C/C++	MPC3035.h	MPC3035VC.lib
Borland C/C++	MPC3035.h	MPC3035BC.lib
Microsoft Visual Basic	MPC3035.bas	

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 or pulse (5.1.2). The un-usual type of control can be through the communication method(5.1.3).

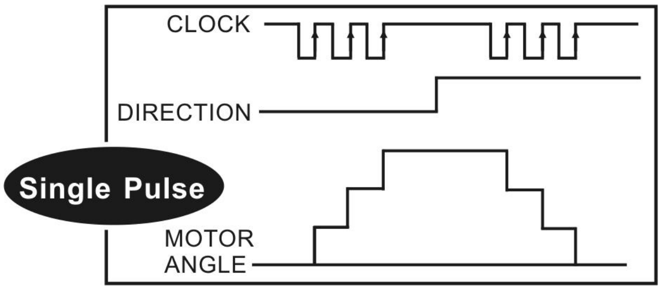
5.1.1 Pulse type motion control

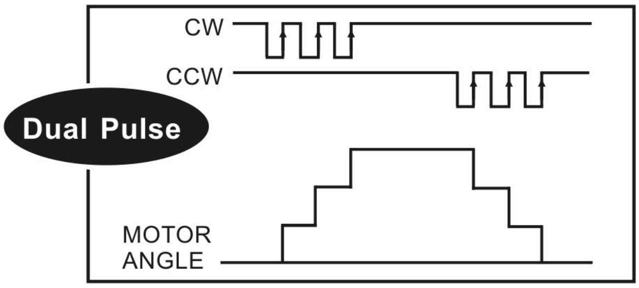
The pulse type motion control was used long ago in step motor control system. In the recent year, a new trend of digital control has moved the servo control from traditional analog control to pulse type motion control.

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.





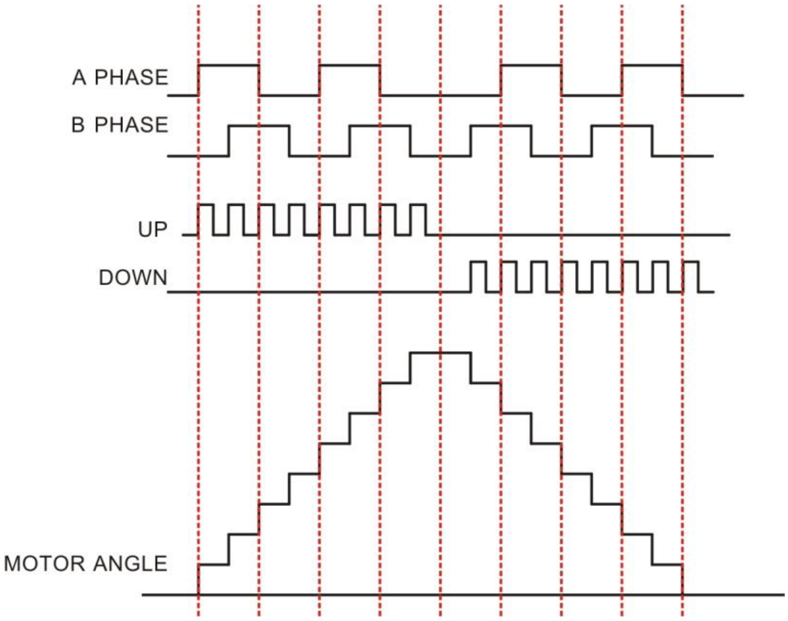
Single pulse type control use only one clock source to control speed and position and the other input is direction control. Dual pulse type control use clockwise clock to control speed and position in one direction and counter-clockwise clock for the other direction.

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 harzard to equipment.

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

MPC3035 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 idenfied 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 loop the position control mode.

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

The merit of voltage type control is the speed and position is directly controlled by controller but the complexity of wiring and system tunning is the withdrawal.

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 designer 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 is heavily depend on software. The software must fast enough to caculate the profile generation and feedback control algoritm. 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. MPC3035 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 majorly 4 types of application:

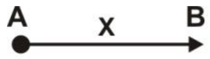
- speed control: controller controls the speed of the servo motor.
- torque control: the controller controls the torque output of the servo motor.
- tracking control: the controller controls the servo motor to follow the motion of another servo motor.
- positioning control: the controller controls the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

Of course a mixed mode is possible.

MPC3035 is hard ware designed for speed control and position control (point to point and linear, circular interpolation). Tracking control can also implemented by software.

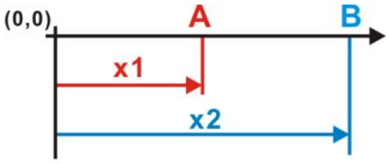
5.4 Coordinate system

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.

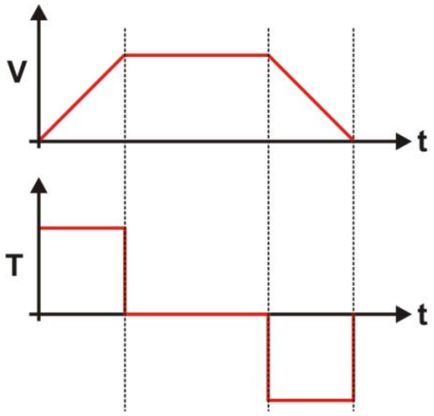


A is at x1 position
B is at x2 position

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

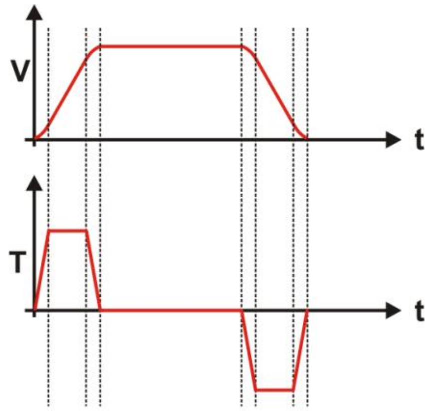
5.5 Motion profile

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 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 resonances and overshoot

The disadvantage is:

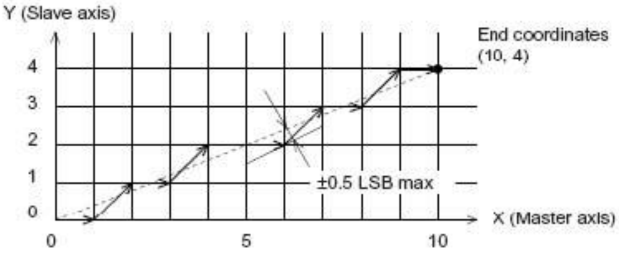
- Requires either twice the acceleration torque or acceleration time for a S profile compared to trapezoidal motion profile

MPC3035 card provides both motion profile function for the user application, you can estimate the sytem requirement to make the decision.

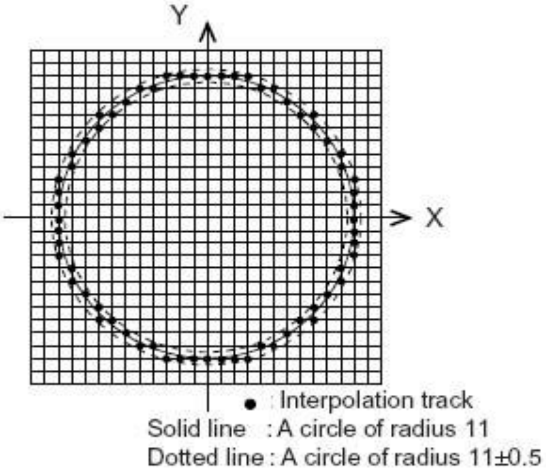
5.6 Interpolation

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 interpolation are the two most important interpolation functions. MPC3035 card provides the hardware interpolation of both. If you want to do special curve interpolation, you can devide 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 MPC3035 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

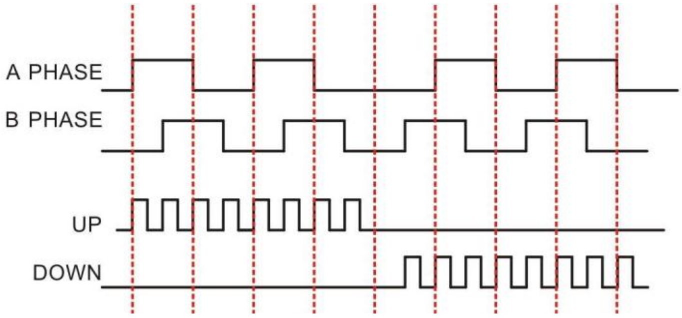
While system is power up and if the encoder is not absolute type, the system do not know where it is now. Homing function will return the mechanism to a known point and set the coordinate. There are so many homing modes available for users. MPC3035 provides 13 homing modes to fit different requirement 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. MPC3035 provides both the hardware over-travel limit and software over-travel limit functions.

5.8 Feedback element of servo system

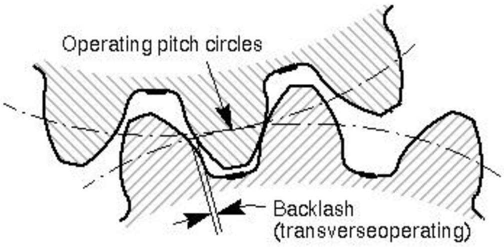
There are several types of servo motor feedback elements such as: encoder (absolute or incremental) ,resolver, potential meter… MPC3035 card can only deal with incremental encoders. It is a device with 2 phase signals separated at 90 degree. We can discriminate the rotation direction from the phase lead 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

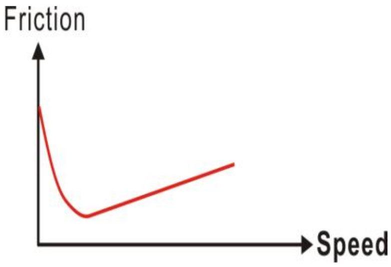
The motion control system is actually a mechtronic system (mechanical + electronics). If you want the system work perfect, you can not overlook the importance of mechanism.

5.9.1 Backlash



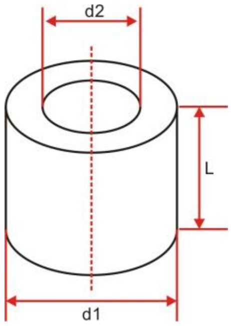
Backlash is the free motion of mechanism when the direction reversed. It is one of the important nature of mechanism. It exist in gear, screw mechanism.

5.9.2 Fraction



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 oscillatyion.

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. Basic concepts of compare function

The compare function is complex for the green hands. First of all you must know the compare function is basically applied to the counter to compare for a preset value (coordinate). If we want to compare equal condition and the counter counting on the fly while the preset value meets the counter temporary or steady value, a pulse of compare equal will generate.

6.1 Compare mode

The counter is one of the compare sources and the other source is the preset value. The preset value comes from a preset register (compare value register) the comparator logic compare with the counter value on the fly. There are 3 compare modes to choose:

One time compare mode (Single compare mode)

The desired compare value is pre-loaded, if the quadrature counter value and the compare value meet the compare condition, generate output trigger.

Auto increment compare mode

If the compare value (compare data) not only store at the preset register (compare value register) but also other subsequent data is define by a incremental value of current compare value. After one compared condition met, the preset register (compare value register) will be loaded a data which is the sum of current compare value and the incremental value to proceed the next compare.

new compare value= current compare value + auto increment value

(NOTE: the incremental value can be a minus value, this means a decrement of current compare value)

FIFO compare mode

If the compare value (compare data) not only store at the preset register (compare value register) but also other subsequent data stored at a FIFO (first in first out memory), after one compared condition met, the FIFO will supply the preset register (compare value register) a new pop up data to proceed the next compare.

new compare value = value pop up from FIFO

The compare function will continue until the FIFO is empty.

6.2 Compare method (policy)

There are 3 compare methods to choose.

-- compare equal If the preset value and the counter value are equal, generate a trigger pulse.

-- greater or equal If counter greater than or equal to the compare value, generate a trigger pulse.

-- less or equal If counter less than or equal to the compare value, generate a trigger pulse.

Take of the greater than or less than condition, both are open end conditions. Say if you want to compare with less than or equal condition, your preset register set at 100 and the counter goes from 110 to 90, this compare will generate trigger pulse and never stop until the counter go more than 100.

6.3 Trigger output width

It is apparently that you will use the CMP_OUT to trigger some device to start some tasks. Not every device is so fast to recognize the compare out pulse. A compare out pulse width (or duration) timer will extend the pulse to your need. MPC3035 card provide the compare equal pulse duration on a 1us base and 24 bit data length.

6.4 Segment mask off and external gate function

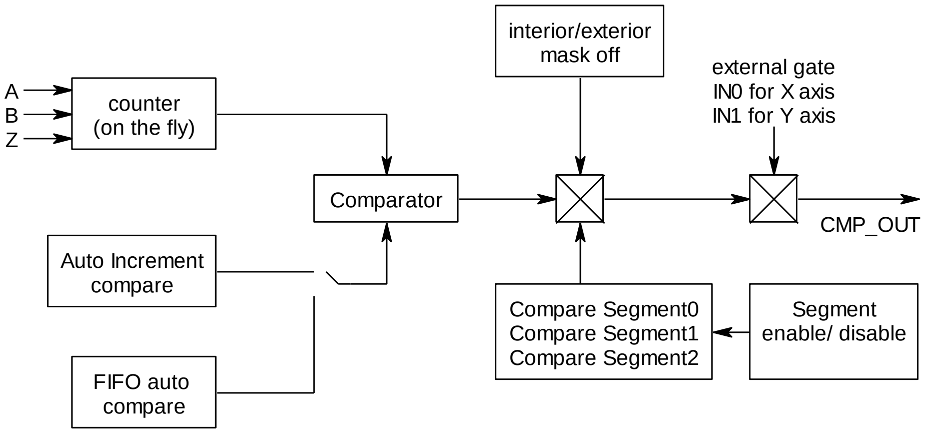


fig. 6-1 Compare mode function block diagram

The segment mask off function is only meaningful for FIFO mode and auto increment mode. The external gate control can override to disable the trigger output by external signal.

Let us begin to explain the segment mask off function from the function block diagram shown above. At the left top, the counter is counting on the fly once your configuration is done. The A, B ,Z phase input signals determines the counter value and direction.

The counter value is sent to comparator at which another comparison source is selected from FIFO or Auto increment mode. If the two coming values are met the compare method (policy), the comparator will generate a trigger to proceed with the auto increment state machine or pop out data from FIFO. But the trigger will going out as CMP_OUT signal or not depends on the other control signals.

At the right most, the CMP_OUT is the final output trigger, it is controlled by compare segment and interior/exterior mask off and external gate. The external gate signal comes from IN0 (for X axis CMP_OUT), from IN1 (for Y axis CMP_OUT). The IN0, IN1 polarity can be programmed as your physical hardware to gate the trigger signal to CMP_OUT pin.

There are total 3 segments to configure. You can set the segment at a specific coordinate, say segment0 from1,000 ~ 10,000, then enable segment0. If you set mask off to interior, the compare out pulses at interior of segment0 will be masked off and only the segment exterior can pass the compare trigger. If you set mask off at exterior, the coordinate outside of the segment0 can generate compare trigger. The segment1 and segment2 also have the same function as segment0 does. If you disable the segment function, no segment mask off function will be of the disabled segment.

7. Software overview

The MPC3035 card is composed of a base card for 4 axes motion control and a piggy back multifunction encoder counter card for 2 axes. The followings describe the features and functionality of the MPC3035 boards and briefly describes the MPC3035 dll functions.

7.1 Initialization and close

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

MPC3035_initial( ) will do.

Once you want to close your application, call

MPC3035_close( ) to release all the resource.

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

MPC3035_init_card( ) is a must.

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

MPC3035_info( ) to get the address.

If you just want to verify the flow of your application program with no MPC3035 card to plug in,

MPC3035_dll_Simu_mode( ) should be called before ***MPC3035_initial( )*** to build up a simulation environment.

System parameters (motion function parameters, motion related I/O, feedback encoder counter related I/O) configuration data can be saved to file by

MPC3035_save_config2_file( )

and retrieve to the card by

MPC3035_load_config_from_file( )

7.2 General I/O configuration and control

The MPC3035 card provide 2 nibble configurable TTL I/O, configure it with:

MPC3035_config_TTL_IO_MODE( ).

MPC3035_readback_TTL_IO_MODE( ) for configuration read back.

If you will check the TTL I/O or motion related or feedback encoder input status

MPC3035_read_point_status( ) will give you the result.

To read the related status by one command,

MPC3035_read_status( ) will do for all the motion status.

MPC3035_read_port( ) for TTL I/O port.

To set or reset the TTL I/O or motion related or feedback encoder counter output by using:

MPC3035_write_output_point( ).

MPC3035_write_port( ) for TTL I/O port.

Setup for motion control

There are single pulse mode (clock and direction control) and dual pulse mode (cw and ccw clock) you can choose to control the servo drivers. Both signal can also be active high or active low to drive the servo driver. To meet your driver requirements, you should first configure the pulse output mode with:

MPC3035_set_pulse_outmode( )

MPC3035_readback_pulse_outmode( ) 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.

MPC3035_config_SD_PIN( ) will do.

MPC3035_readback_SD_PIN( ) 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

MPC3035_config_EL_MODE( )**.**

MPC3035_readback_EL_MODE( ) for configuration read back.

The polarity of over-travel limit switch can be set by on card dip switch, please refer the hard manual on chapter 8.2 Polarity setting for over-travel limit switch.

Some packing machines need the motion control to trace the mark to positioning. This application need external sensor to trigger the motion to stop a certain distance away after trigger active. You can use the PCS(position change start) function to implement by:

MPC3035_config_PCS_PIN( ) to configure the PCS as dedicated PCS input.

MPC3035_readback_PCS_PIN( ) for configuration read back.

After adequate configuration, you can use

MPC3035_position_override( ) to setup the distance of override on PCS signal active or command to override immediately.

Servo drive related

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 remain pulses to move. To ensure the accuracy, most servo drivers provide error counter (deviation counter) clear input for external device to clear the remain pulses. For automatic error counter clear at homing, use

MPC3035_config_ERC_PIN( ) to configure your requirement.

MPC3035_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,

MPC3035_config_ALM_PIN( ) will do.

MPC3035_readback_ALM_PIN( ) for configuration read back.

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

MPC3035_config_INP_PIN( ).

MPC3035_readback_INP_PIN( ) for configuration read back.

7.3 Velocity mode motion

Velocity motion control is one of the functions of MPC3035 card. For safety reason or others to set the maximum speed is recommended. Use

MPC3035_fix_speed_range( ) to set the maximum allowable speed.

MPC3035_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 ***MPC3035_T_velocity_move( )*** to move at trapezoidal profile.

MPC3035_S_velocity_move( ) to move at S curve profile.

If you want to change speed or stop it, use

MPC3035_velocity_change( ) to change speed.

MPC3035_dec_stop( ) to have a deceleration to stop.

MPC3035_imd_stop( ) to stop specific axis immediately.

MPC3035_emg_stop( ) to stop all axes immediately.

To read back the speed on current time use:

MPC3035_read_speed( ) will give you the current speed.

7.4 Homing

At the beginning of positioning or contouring control, MPC3035 (controller) must know the initial coordinate it is. Various kinds of homing provide you flexible choice of implementation of getting the initial coordinate.

Before homing, the polarity (logic) of HOME(ORG) limit switch and encoder zero phase should be configured, use

MPC3035_set_HOME_pin_logic( ) to set up home input polarity.

MPC3035_readback_HOME_pin_logic( ) for configuration read back.

For encoder zero phase polarity,

MPC3035_set_EZ_pin_logic( ) will do.

MPC3035_readback_EZ_pin_logic( ) for configuration read back.

Once the polarity is configured, which kind of homing mode is suitable for your application, you must decide, use

MPC3035_config_home_mode( ) to select the desired homing mode.

MPC3035_start_homing( ) to execute homing.

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

MPC3035_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,

MPC3035_read_current_position( ) will do.

For some special applications, you only want to use home (ORG) limit switch as reference and the initial state to CW or CCW is unknown, use

MPC3035_start_origin_search_homing( ) will seek home (ORG) limit switch automatically and correct the position.

7.5 Point to point motion control

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

MPC3035_T_curve_position_move( ) for trapezoidal acc/dec profile.

MPC3035_S_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, *MPC3035_position_change( )* will do.

For accuracy positioning, the backlash compensation is required,

MPC3035_backlash_comp( ) is the function you need.

MPC3035_readback_backlash_comp( ) to read back parameters.

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

MPC3035_suppress_vibration( ) will give less vibration at final positioning.

MPC3035_readback_suppress_vibration( ) to read back the data you set.

7.6 Linear interpolation motion control

Once you have homed and the coordinate system has setup, the absolute or relative linear interpolation function now is available.

MPC3035_T_curve_move_LINE2( ) for any two axes linear interpolation at trapezoidal profile.

MPC3035_S_curve_move_LINE2( ) for any two axes linear interpolation at S curve profile.

For any 3 axes use:

MPC3035_T_curve_move_LINE3( ) or

MPC3035_S_curve_move_LINE3( )

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

MPC3035_T_curve_move_LINE4( ) or

MPC3035_S_curve_move_LINE4( )

For unified motion command

MPC3035_T_ LINE _move( ) will command the motion from 1 to 4 axes in T profile.

MPC3035_S_ LINE _move( ) will command the motion from 1 to 4 axes in S profile.

7.7 Circular interpolation motion control

Once you have homed and the coordinate system has setup, the circular interpolation function now is available. We can do circular interpolation on any two of the 4 axes ,if you wish to use the circle center and arc end position as parameters, use:

MPC3035_ARC2_center_move( )

MPC3035_T_ARC2_center_move( ) for T curve profile

MPC3035_S_ARC2_center_move( ) for S curve profile.

If you wish to use current point, end points and a mid-point to go through as the circle trajectory parameters, use:

MPC3035_ARC2_3P_move( )

MPC3035_T_ARC2_3P_move( ) for T curve profile

MPC3035_S_ARC2_3P_move( ) for S curve profile.

To use the radius to designate the arc

MPC3035_ARC2_Radius_move( ) will do,

MPC3035_T_ARC2_Radius_move( ) will have T curve acc/dec profile.

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

MPC3035_CIR2_3P_move( ) for constant speed motion.

MPC3035_T_CIR2_3P_move( ) for T acceleration/deceleration curve motion.

MPC3035_S_CIR2_3P_move( ) for S acceleration/deceleration curve motion.

To use the radius to designate the circle.

MPC3035_CIR2_Radius_move( ) will do,

MPC3035_T_CIR2_Radius_move( ) will have T curve acc/dec profile.

7.8 Spiral motion

Spiral motion control consist of 2 axes circular and one axis linear motion synchronously. If you do not use the A axis, you can have the spiral motion by:

MPC3035_ArcXY_LineZ_center_move( ) for X,Y going arc and Z linear interpolation and the parameters are defined by end point and circle center point.

MPC3035_ArcXY_LineZ_3P_move( ) for X,Y going arc and Z linear interpolation and the parameters are defined by end point and mid-point to go through.

MPC3035_CirXY_LineZ_3P_move( ) for X,Y going circle and Z linear interpolation and the parameters are defined by current position and other 2 points to go through.

7.9 Motion with FIFO

Some application need small line segments to simulate the curve, using the FIFO function is a good solution. But first of all, we should clarify that the motion control FIFO is softrware type, it needs some CPU resource and it is differ from the quadrature encoder counter block. The counter block's FIFO is hardware FIFO.

Once you decide to use FIFO for your motion control, use

MPC3035_enable_FIFO( ) to initialize the FIFO,

Before you fill the FIFO, check the FIFO remained size is importance, use

MPC3035_check_FIFO_buffer( )

Then fill the FIFO by:

MPC3035_T_curve_write_FIFO( ) for one axis motion control,

MPC3035_T_LINE2_write_FIFO( ) for dual axes motion control,

MPC3035_T_LINE3_write_FIFO( ) for 3 axes motion control,

MPC3035_T_LINE4_write_FIFO( ) for 4 axes motion control, then you can start the motion by ***MPC3035_Run_FIFO_CMD( )*.**

If you want to change speed on the fly,

MPC3035_set_FIFO_out_Ratio( )

The FIFO will consume as the motion go on, you must supply the FIFO to go until end of the curve, there are two methods to supply: by program scanning or by near empty interrupt to supply.

If you will supply FIFO by program scanning, just use the write FIFO and check FIFO buffer function to implement the supply function.

If you will supply at near empty interrupt, at you interrupt service routine, you must write FIFO and check FIFO buffer and at the end before exit from interrupt service routine use:

MPC3035_FIFO_EOI( ) to reset the FIFO mechanism to signal interrupt next time.

7.10 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.

MPC3035_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:

MPC3035_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:

MPC3035_T_curve_wait_Cmpsatrt( ) to synchronous start single axis T profile motion.

MPC3035_S_curve_wait_Cmpstart( ) to synchronous start single axis S profile motion.

MPC3035_T_LINE2_wait_Cmpstart( ) to synchronous start dual axes T profile linear interpolation motion.

MPC3035_S_LINE2_wait_Cmpstart( ) to synchronous start dual axes S profile linear interpolation motion.

MPC3035_T_LINE3_wait_Cmpstart( ) to synchronous start triple axes T profile linear interpolation motion.

MPC3035_S_LINE3_wait_Cmpstart( ) to synchronous start triple axes S profile linear interpolation motion.

MPC3035_T_LINE4_wait_Cmpstart( ) to synchronous start 4 axes T profile linear interpolation motion.

MPC3035_S_LINE4_wait_Cmpstart( ) to synchronous start 4 axes S profile linear interpolation motion.

MPC3035_ARC2_center_wait_Cmpstart( ) to synchronous start circular interpolation motion.

MPC3035_ARC2_3P_wait_Cmpstart( ) to synchronous start circular interpolation motion.

You can check the synchronous start flag by

MPC3035_read_compare_start_flag( )

7.11 External trigger start/stop function

The External trigger start/stop is an external input/output trigger function which can be used on multiple cards to be start/stop simultaneously. User just connect the JM1,JM2 wiring (refer hardware manual 5.2 JM1,JM2 Assignment / Definition) across the MPC cards, then the master trigger out and slave receive the command to start/stop simultaneously.

To trigger out the start signal, use

MPC3035_trigger_CSTA_pulse( ).

To trigger out the stop signal, use

MPC3035_trigger_CSTP_pulse( ).

For a T profile motion control, the slave wait for the start signal to move, use

MPC3035_T_curve_wait_CSTA( )
MPC3035_T_LINE2_wait_CSTA( )
MPC3035_T_LINE3_wait_CSTA( )
MPC3035_T_LINE4_wait_CSTA( ) for the respective motion.

For the S profile motion, use

MPC3035_S_curve_wait_CSTA( )
MPC3035_S_LINE2_wait_CSTA( )
MPC3035_S_LINE3_wait_CSTA( )
MPC3035_S_LINE4_wait_CSTA( ) for the respective motion control.

For the circular interpolation

MPC3035_ARC2_center_wait_CSTA( )
MPC3035_ARC2_3P_wait_CSTA( ) will do.

7.12 Continuous motion function

For some applications such as gluing, you need to move continuously (no acce/dec at one motion segment to another). Use

MPC3035_set_continuous_flag( ) to enable / disable the continuous mode.

Once you have switch to continuous mode, the followed motion functions you program will go continuously (without segment by segment with acc/dec).

For the motion status read back of continuous mode,

MPC3035_check_continuous_buffer( ) to check the buffer full or not for the availability of the next motion command.

To check exactly how many buffered data left,

MPC3035_read_conti_buffer_no( )
MPC3035_read_motion_status( ) for motion status read back.

7.13 Position change on the fly function

For the linear interpolation application that needs to change position or speed profile on the fly, using ***MPC3035_OnLine_T_curve_change( )*** to change for single axis,

MPC3035_OnLine_T_curve_change_LINE2( ) to change for dual axes,

MPC3035_OnLine_T_curve_change_LINE3( ) to change for triple axes and

MPC3035_OnLine_T_curve_change_LINE4( ) for the whole control axes.

7.14 Motion resume function

If the motion is halted by software or hardware, restart of motion is possible.

MPC3035_OneAxis_restart( ) for single axis restart.

MPC3035_2Axis_restart( ) for two axis restart.

MPC3035_3Axis_restart( ) for three axis restart.

MPC3035_4Axis_restart( ) for four axis restart.

7.15 Interrupt function and motion event

Sometimes you want your application to take care of the motion while special event occurs, interrupt function is the right choice. First of all enable the global IRQ function.

MPC3035_enable_IRQ( ) should program at the setup stage.

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

MPC3035_disable_IRQ( ) to release the resource.

Next tell the driver your interrupt service routine by

***MPC3035_link_IRQ_process( )*.**

On MPC-3035 card there are many sources to generate interrupt, select the interrupt source by

***MPC3035_set_INT_source( )*,** and use

MPC3035_read_INT_status( ) to read the interrupt event generating source.

Finally you should enable / disable the specific hardware of the interrupt source,

MPC3035_set_INT_mask( ) will do.

When MPC3035 card generate an interrupt, the driver will wake up the user's interrupt service routine.

For the application that do not use interrupt, but event of motion will take care by polling

MPC3035_set_event_factor( ) to setup the event generated by the control card. Then polling the event status by:

MPC3035_read_event_flag( ) to get event flag and then decide your next procedures.

MPC3035_read_error_flag( ) will report the error conditions for your application.

7.16 Soft limit protection function

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

MPC3035_config_softlimit( ) to setup configuration.

MPC3035_readback_config_softlimit( ) to read back configuration.

MPC3035_set_softlimit_data( ) to setup the coordinate data of limit.

MPC3035_readback_softlimit_data( ) to read back preset data.

MPC3035_enable_softlimit( ) to enable / disable software limit function.

MPC3035_readback_enable_softlimit( ) to read back configuration.

MPC3035_read_softlimit_flag( ) to read the software limit flag for verifying.

7.17 Manual pulser function

For the application requires pulse handler (or pulser) to work as manual control of speed or position, MPC3035 provides 2 integrated functions, first you must map the pulse handler to the motion axis by

MPC3035_set_pulser_Map( )

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

MPC3035_enable_pulser_motion( ).

Now you can manually control the servo motor in speed and position by pulse handler.

If you want to do manual functions by discrete function call, you must configure the operating mode of the pulse handler with:

MPC3035_config_pulser_mode( )

MPC3035_readback_pulser_mode( ) to read back configuration.

To operate as manual speed control,

MPC3035_run_pulser_Vmove( ) will do, and for position mode, use

MPC3035_run_pulser_Pmove( )

Concerning the pulse handler input counter, use

MPC3035_set_pulser_counter( ) to set pulse counter, and

MPC3035_read_pulser_counter( ) to read back the counter value.

7.18 Multi-function feedback counter

Each axis has a feedback counter on card, you have to configure the signals' input multiple rate and the physical input type,

MPC3035_set_pulse_inmode( ) will do.

MPC3035_readback_pulse_inmode( ) for configuration read back.

After configuration, you can manipulate the counter by:

MPC3035_read_FB_counter( ) to read counter value.

MPC3035_set_FB_counter( ) to preset the counter value.

If your application needs to latch the encoder feedback on the fly by external trigger, use

MPC3035_config_LTC_PIN( ) to configure the trigger input pin.

MPC3035_readback_LTC_PIN( ) for configuration read back.

After you have configured latch input function, use

MPC3035_read_FBcounter_latch_value( ) to read the latched counter value.

Feedback counters also provide compare function for you to generate a trigger pulse output at designated value of feedback counter, configure the output with:

MPC3035_config_CMP_OUT( ).

MPC3035_readback_CMP_OUT( ) for configuration read back.

If you have configured compare output function, for single axis comparison use

MPC3035_config_comparator_out( ) to configure the compare output mode.

MPC3035_readback_comparator_out( ) to read back configuration.

MPC3035_set_comparator_data( ) to preset the value to the comparator.

MPC3035_readback_comparator_data( ) to read back preset value.

MPC3035_read_compare_flag( ) to read compare out flag for verifying the active state of the function.

Sometimes you need to use compare out function with more than single axis, say 2 or more axes, a comparison window is more adequate than exactly one point. For each axis you want to compare use:

MPC3035_config_comparator_out_W( ) to configure the compare output mode.

MPC3035_readback_comparator_out_W( ) to read back configuration.

MPC3035_set_comparator_data_W( ) to preset the value to the comparator.

MPC3035_readback_comparator_data_W( ) to read back preset value.

MPC3035_read_compare_flag_W( ) to read compare out flag for verifying the active state of the function. But notice that you can only use either single point comparison function or window comparison function, not both of them in your application.

7.19 Software key function

Since MPC3035 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.

Before your system setup, you can use the demo program (comes with the card) to set up the password. In your application program after card initialization, unlock the card to enable the dll function. Because the password is set, the card will lock until the

MPC3035_unlock_security( ) to unlock the security.

You can also use

MPC3035_read_security_status( ) to check the current status of security.

For the users who need serial code for verification, use

MPC3035_set_serial_code( ) to setup your own serial code,

MPC3035_read_serial_code( ) to read the serial code.

If you will initialize lock in your application program, you can use

MPC3035_set_password( ) to set password and start the security function.

MPC3035_change_password( ) to change it.

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

MPC3035_clear_password( ) will reset to the virgin state.

7.20 Counter setup

The piggy back board on MPC3035 is the high speed multifunction encoder counter card. It has many advanced functions for 2 axes.

Before using the counters, set and verify A, B, Z phase input polarity by:

MPC3035L_set_input_polarity( )

MPC3035L_read_input_polarity( )

and set and verify compare output polarity by:

MPC3035L_set_output_polarity( )

MPC3035L_read_output_polarity( )

Read the phase input status by

MPC3035L_read_input_status( )

The major function of quadrature encoder/linear scale counter is to count the input pulse train accurately. Owing to multi-function design, you must choose the counter operation mode to meet the input – quadrature input counting or up/down clock counting or clock/direction counting as you need:

MPC3035L_set_counter_mode( ) will do.

If you use a quadrature encoder as signal source, be sure to configure the multiple rate by

MPC3035L_set_quadrature_times( )

7.21 Counter homing (to clear counter)

At the beginning of an application, the position of encoder / linear scale needs a reference point of coordinate. There are various modes for counter hardware homing, one-time mode for the special occasion of system coordinate setup and continuous mode can be used for cyclic motion. You can choose the homing mode and starts homing by:

MPC3035L_set_hard_homing( )

To check if the hardware completes homing by:

MPC3035L_read_hard_homing_flag( )

If anytime you want to clear counter (by software),

MPC3035L_soft_homing_command( ) will do.

7.22 Counter function

To read the counter value at any time, use

MPC3035L_read_counter( ) to get counter data on the fly.

If you want to set counter value,

MPC3035L_load_counter( ) loads counter the desired value.

To incorporate with the compare function, the counter can provide function of compare equal to trigger latch. To enable or disable the latch function

MPC3035L_latch_control ( ) will do and setup the latch mode by

MPC3035L_latch_mode ( )..

After set up these functions, once the counter equals the preset compare value internal hardware will trigger to latch counter. Just use

MPC3035L_read_latch_flag ( ) to read the flag for trigger event verification and use

MPC3035L_read_latched_value ( ) to read the counter latched value.

For some applications, both the 2 working axes will need to latch while X axis CMP_OUT trigger occurred. ***MPC3035L_write_XY_latch( )*** will enable or disable the XY latch function.

To read back the setting use:

MPC3035L_read_XY_latch( )

7.23 Compare function

Compare the counter to a preset value is a useful but special function. In application that needs to trigger external devices on the fly at specific point, the compare function is a good solution.

There are 4 modes to choose: one time mode, auto increment mode and FIFO (absolute position) mode, and no compare function mode.

Use ***MPC3035L_set_CMP_OUT_mode( )*** to setup the compare mode. If use “no compare function mode”, compare out pin can be used as general purpose output:

MPC3035L_write_output_command( ) to set/reset the output.

MPC3035L_read_output_status( ) to read back the output status.

If you will use compare mode, you must load a value for first comparison, use:

MPC3035L_load_compare_value( ) to load the compare value.

MPC3035L_read_compare_value( ) to read the compare value set.

The compare method is equal to or greater than, less than is also programmable by:

MPC3035L_set_CMP_method( ).

Auto increment compare mode

Auto increment mode will auto increase the compare value for the next comparison. If your application is to compare at regular distance, it is adequate to use:

MPC3035L_load_increase_value( ) to set the incremental distance after each compare equal.

MPC3035L_read_increase_value( ) to read back auto increment value set.

FIFO compare mode

This FIFO is different from the FIFO of motion control block. It is hardware implemented FIFO and use for counter comparison function.

If your application is not increase at regular distance, using FIFO to program the absolute position one by one is the right solution, before using the function

MPC3035L_clear_FIFO_command( ) resets the FIFO-in and FIFO-out pointer.

To load the FIFO absolute position data, use

MPC3035L_fill_FIFO_value( ) to save data to the FIFO.

If the FIFO is filled full (total 1023 depth), the full flag can be checked by

MPC3035L_read_FIFO_full_flag( )

If you want to check how many FIFO left,

MPC3035L_read_FIFO_unused_ number( ) to read the unused number.

After the compared data coincide with the compare value preset, the trigger out pin will generate a one-shot pulse and pop up the next data to the compare value register until the FIFO is empty. Using

MPC3035L_read_FIFO_empty_flag( ) to check if the FIFO is empty.

For some reasons, you want to pop up the FIFO data or verify the FIFO data,

MPC3035L_read_FIFO_value( ) will do.

Output duration

For output duration of compare output can be set by:

MPC3035L_set_CMP_oneshot_ duration( )

For some special purpose, maybe you need to mask out the trigger (to enable or disable trigger output by specific input status),

MPC3035L_set_Mask_CMP_OUT_ source( ) will give you the maximum flexibility.

Compare segment configuration and compare out mask off

For some applications, you need to disable the CMP_OUT trigger but do not effect the auto increment or FIFO operation. You can use external gate mode or the segment mask off function to disable the trigger output.

There are 3 segments on card, you can choose any one of them or use all of them as you want. First configure the one you want to use and set up the start and stop point coordinate by:

MPC3035L_write_cmp_segment( ) and read back to check by

MPC3035L_read_cmp_segment( ).

Next, the coordinate interior or exterior the start-stop points, you want to mask off the compare out

MPC3035L_write_mask_off( ) and read back by

MPC3035L_read_mask_off( )

At last enable or disable the function by:

MPC3035L_write_segment_control( ) or read back by:

MPC3035L_read_segment_control( )

After all is configured, the CMP_OUT will be mask off as you need.

7.24 Counter Interrupt function

MPC3035 card is composed of main board and daughter board. They are physically independent board. Each board has its own functions. However they share the same interrupt resource. You can select to use either main board interrupt function group or daughter board interrupt function group.

To use the interrupt service, the first step

MPC3035L_enable_IRQ( ) is required to enable the function.

MPC3035L_disable_IRQ( ) is required to disable the function.

There are 5 interrupt sources for your quick response application,

1. hardware counter clear occurred
2. counter compare condition meet
3. counter carry occurred
4. counter borrow occurred
5. FIFO empty alarm occurred

If you will fill FIFO by FIFO alarm interrupt, you must decide the threshold of the alarm by:

MPC3035L_set_FIFO_INT_no( )

After IRQ is enabled use:

MPC3035L_link_IRQ_process( ) to link your service routine to interrupt event.

MPC3035L_set_INT_mask( ) to assign which axis is allowed to generate IRQ.

MPC3035L_set_INT_source( ) to assign the interrupt source.

After you go through the procedures step by step, interrupt and your service routine is linked, once the interrupt occurred, your service routine will play as you wish.

To check the current interrupt status

MPC3035L_read_INT_status( ) will do and

MPC3035L_read_INT_ID( ) to identify interrupt axis.

7.25 Miscellaneous function

MPC3035 card has build-in 2 channels of PWM DA to output 0~10V uni-polar voltage.

MPC3035L_out_PWM_DA( ) will do.

To verify various parameters set, use:

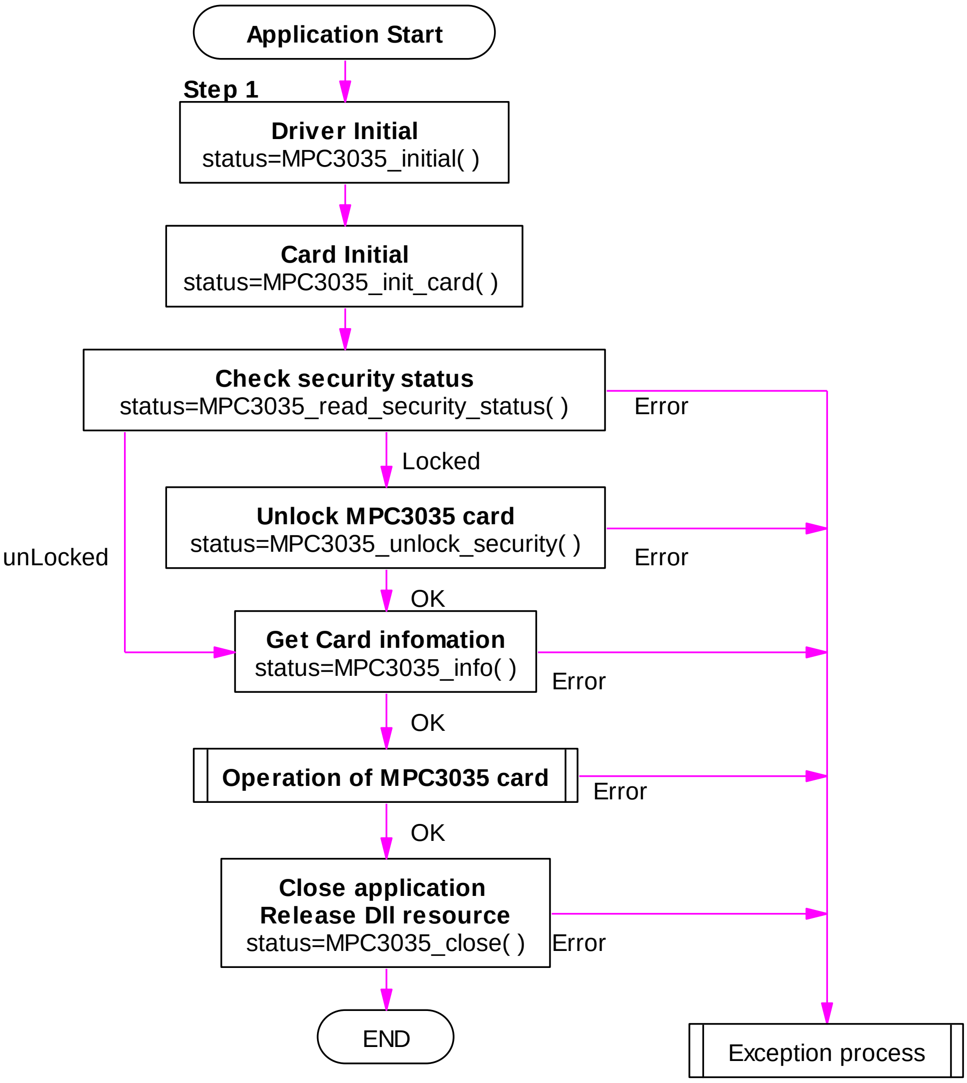
MPC3035L_read_parameters( ) to read back parameters.

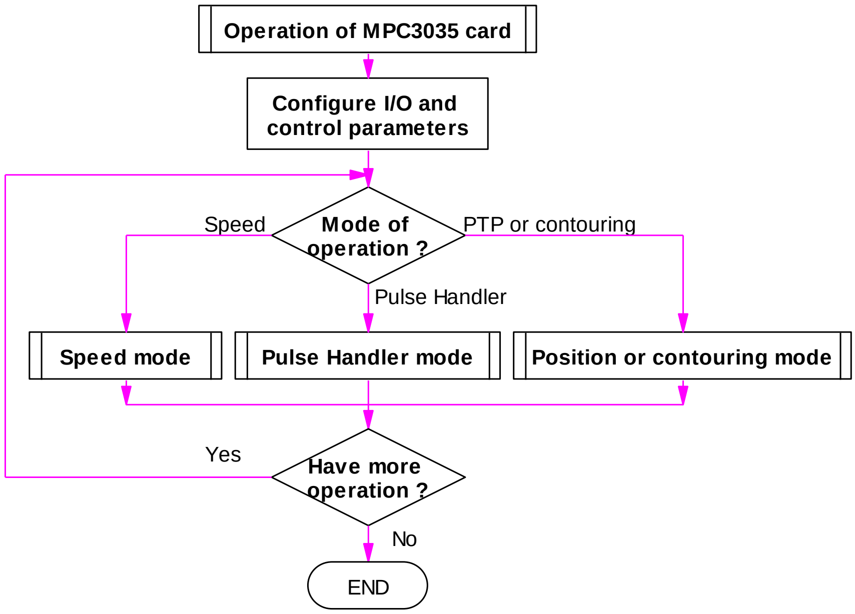
7.26 Error conditions

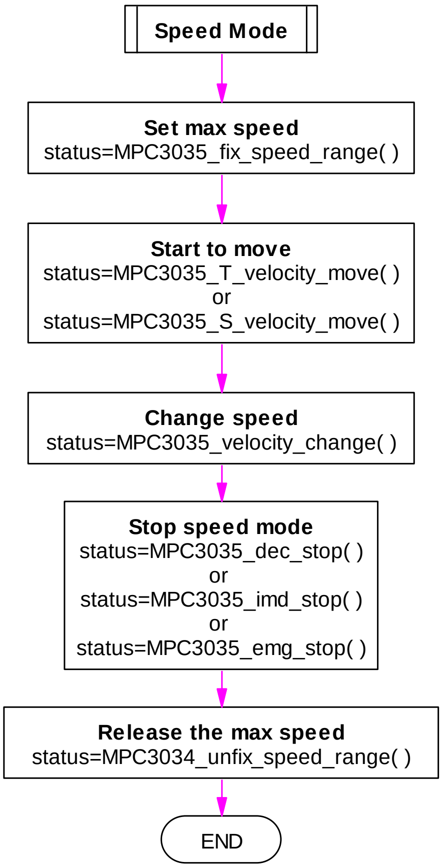
These error types may indicate an internal hardware problem on the board. Error Codes summary contains a detailed listing of the error status returned by MPC3035 functions.

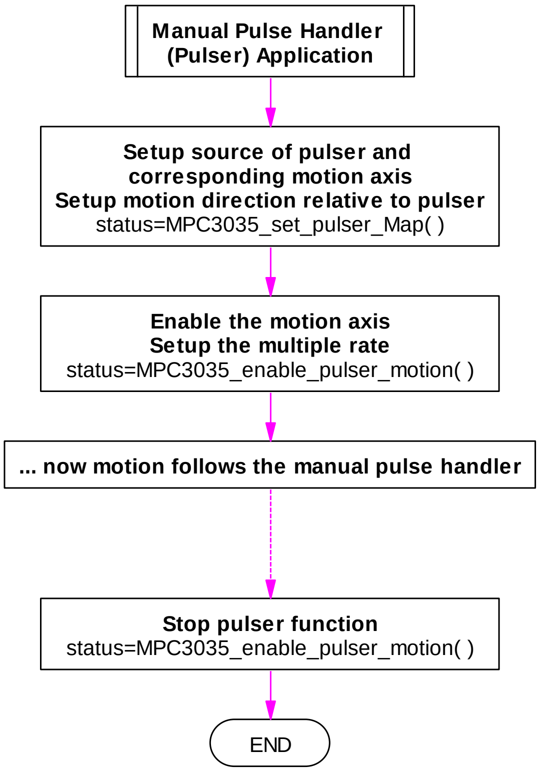
8. Flow chart of application implementation

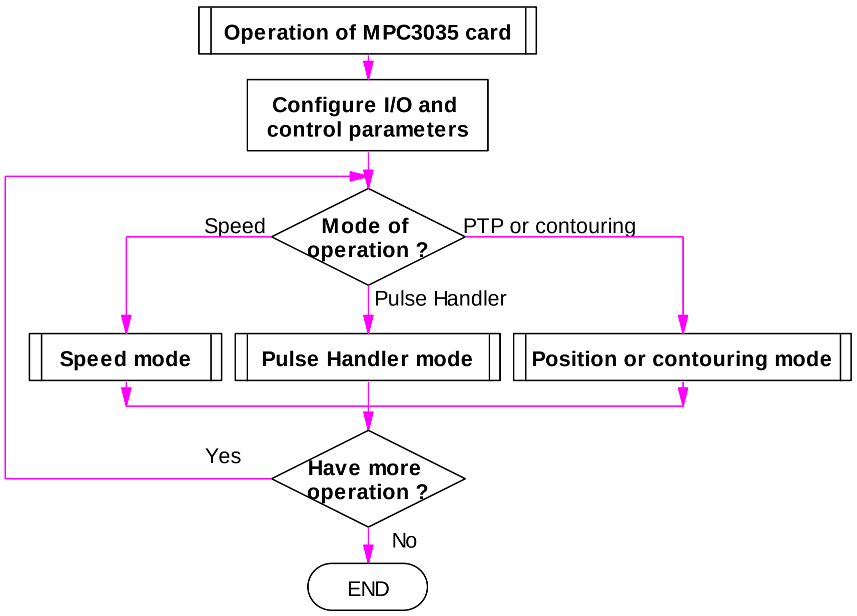
8.1 MPC3035 Flow chart of application implementation

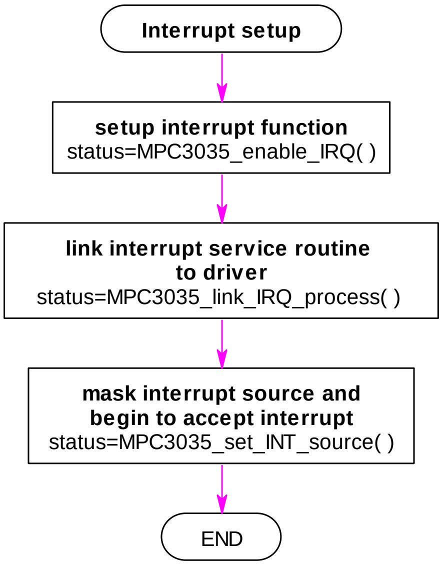


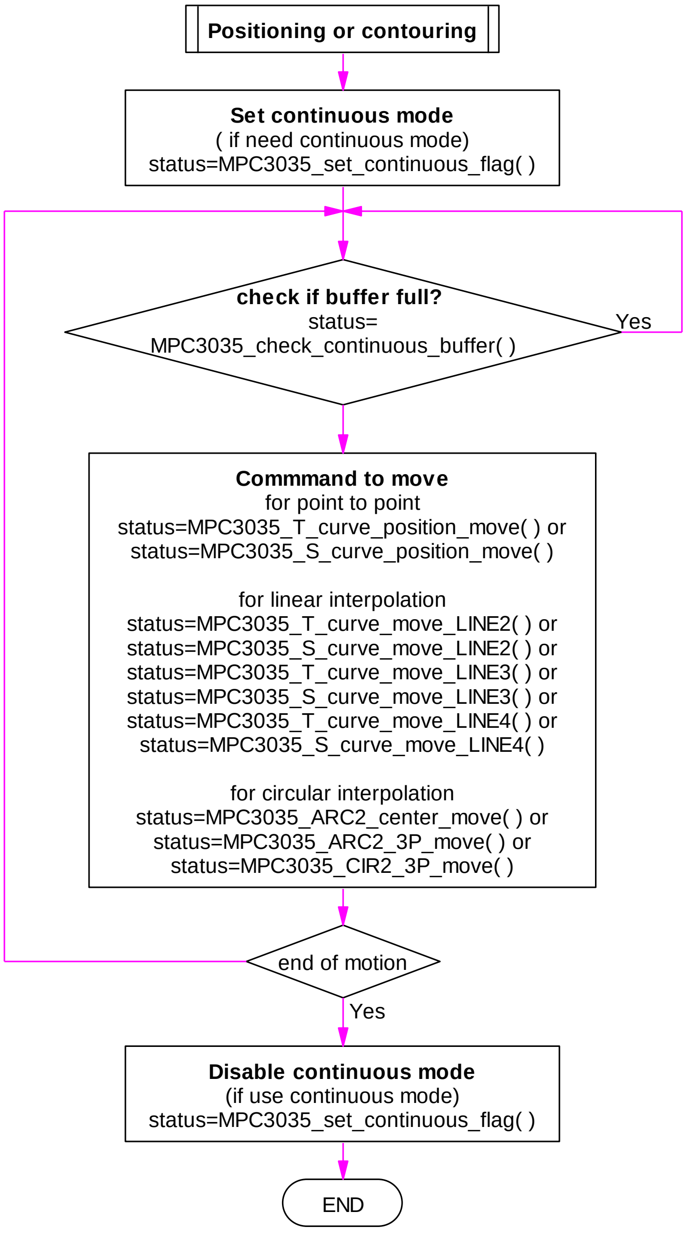

Operation of MPC3035 card
Configure I/O and control parameters
Mode of operation ?
Speed
PTP or contouring
Pulse Handler
Speed mode
Pulse Handler mode
Position or contouring mode
Have more operation ?
Yes
No
END


Speed Mode
Set max speed
status=MPC3035_fix_speed_range( )
Start to move
status=MPC3035_T_velocity_move( )
or
status=MPC3035_S_velocity_move( )
Change speed
status=MPC3035_velocity_change( )
Stop speed mode
status=MPC3035_dec_stop( )
or
status=MPC3035_imd_stop( )
or
status=MPC3035_emg_stop( )
Release the max speed
status=MPC3034_unfix_speed_range( )
END

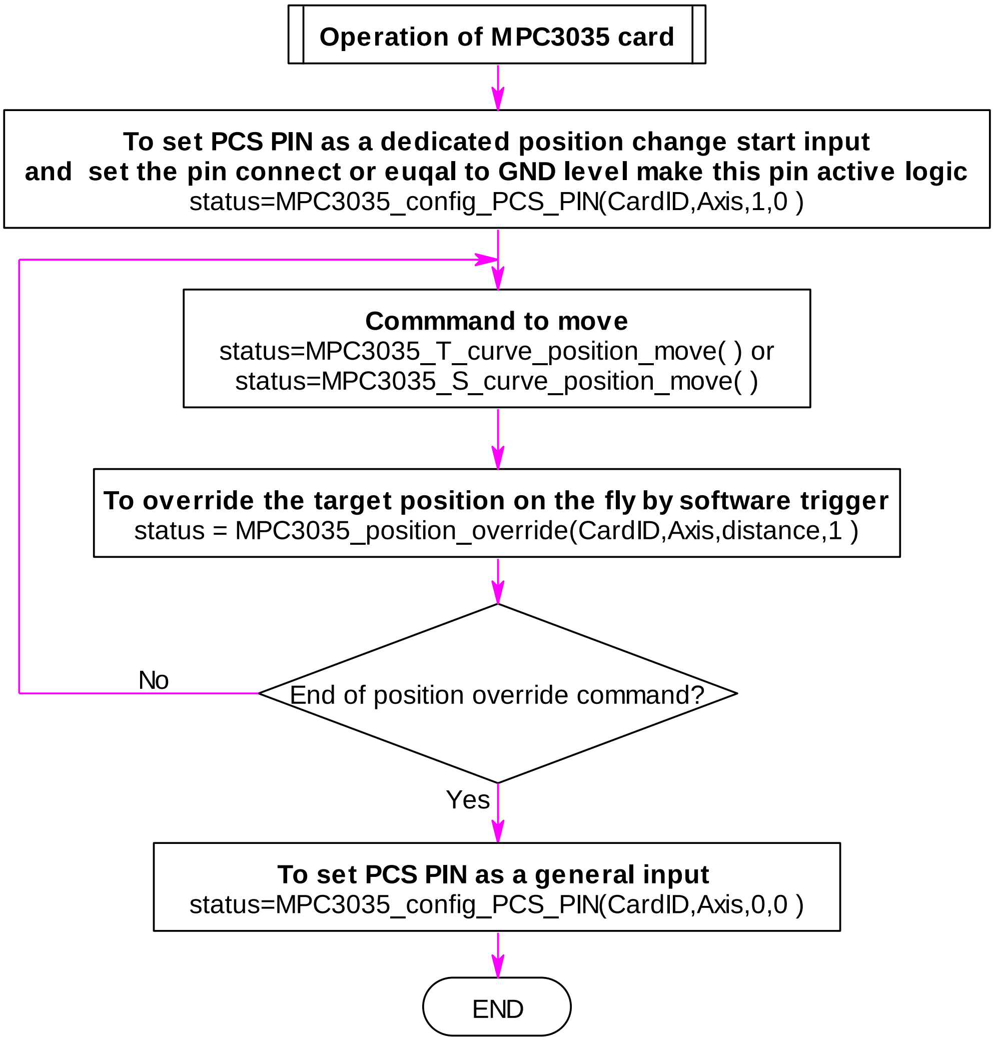

Manual Pulse Handler (Pulser) Application
Setup source of pulser and corresponding motion axis
Setup motion direction relative to pulser
status=MPC3035_set_pulser_Map( )
Enable the motion axis
Setup the multiple rate
status=MPC3035_enable_pulser_motion( )
... now motion follows the manual pulse handler
Stop pulser function
status=MPC3035_enable_pulser_motion( )
END


Operation of MPC3035 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

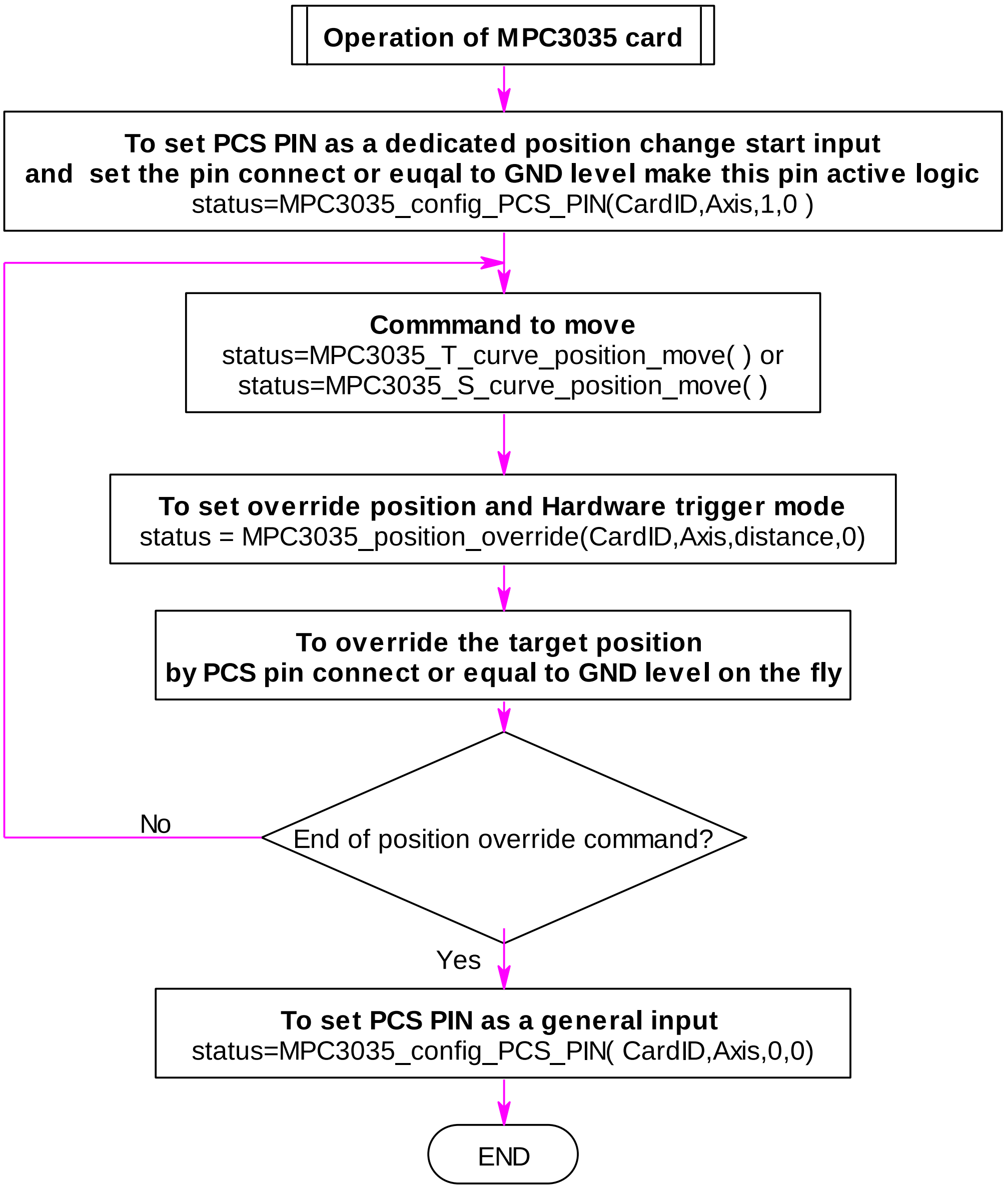

Interrupt setup
setup interrupt function
status=MPC3035_enable_IRQ( )
link interrupt service routine to driver
status=MPC3035_link_IRQ_process( )
mask interrupt source and begin to accept interrupt
status=MPC3035_set_INT_source( )
END


Positioning or contouring
Set continuous mode
(if need continuous mode)
status=MPC3035_set_continuous_flag( )
check if buffer full?
status=
MPC3035_check_continuous_buffer( )
Yes
Commmand to move
for point to point
status=MPC3035_T_curve_position_move( ) or
status=MPC3035_S_curve_position_move( )
for linear interpolation
status=MPC3035_T_curve_move_LINE2( ) or
status=MPC3035_S_curve_move_LINE2( ) or
status=MPC3035_T_curve_move_LINE3( ) or
status=MPC3035_S_curve_move_LINE3( ) or
status=MPC3035_T_curve_move_LINE4( ) or
status=MPC3035_S_curve_move_LINE4( )
for circular interpolation
status=MPC3035_ARC2_center_move( ) or
status=MPC3035_ARC2_3P_move( ) or
status=MPC3035_CIR2_3P_move( )
end of motion
Yes
Disable continuous mode
(if use continuous mode)
status=MPC3035_set_continuous_flag( )
END

8.2 Flow chart of position override by software trigger

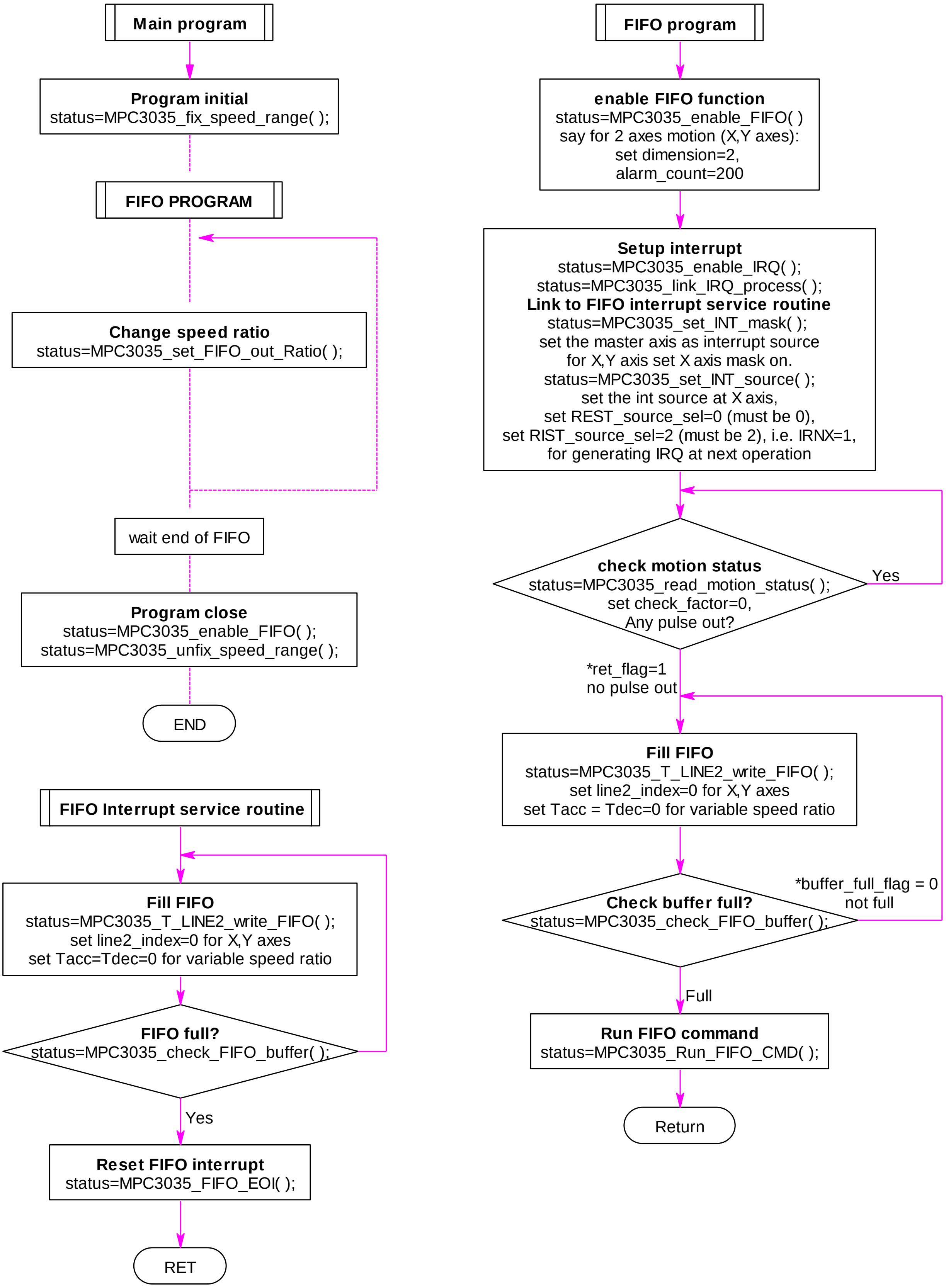

Operation of MPC3035 card
To set PCS PIN as a dedicated position change start input
and set the pin connect or euqal to GND level make this pin active logic
status=MPC3035_config_PCS_PIN(CardID,Axis,1,0)
Commmand to move
status=MPC3035_T_curve_position_move( ) or
status=MPC3035_S_curve_position_move( )
To override the target position on the fly by software trigger
status = MPC3035_position_override(CardID,Axis,distance,1)
End of position override command?
No
Yes
To set PCS PIN as a general input
status=MPC3035_config_PCS_PIN(CardID,Axis,0,0)
END

8.3 Flow chart of position override by hardware trigger

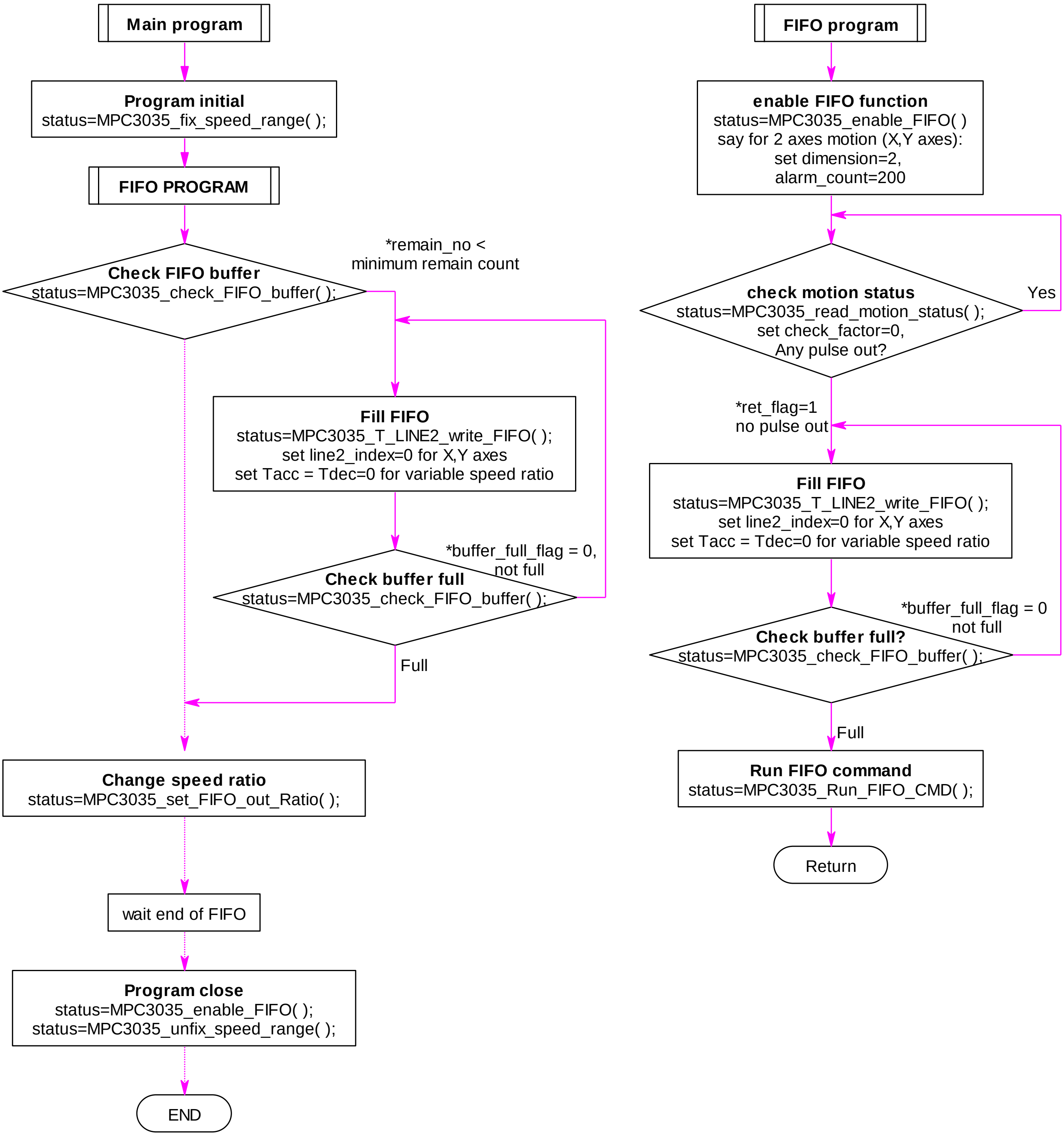

Operation of MPC3035 card
To set PCS PIN as a dedicated position change start input
and set the pin connect or euqal to GND level make this pin active logic
status=MPC3035_config_PCS_PIN(CardID,Axis,1,0)
Commmand to move
status=MPC3035_T_curve_position_move( ) or
status=MPC3035_S_curve_position_move( )
To set override position and Hardware trigger mode
status = MPC3035_position_override(CardID,Axis,distance,0)
To override the target position
by PCS pin connect or equal to GND level on the fly
End of position override command?
No
Yes
To set PCS PIN as a general input
status=MPC3035_config_PCS_PIN(CardID,Axis,0,0)
END

8.4 Control flow of FIFO application

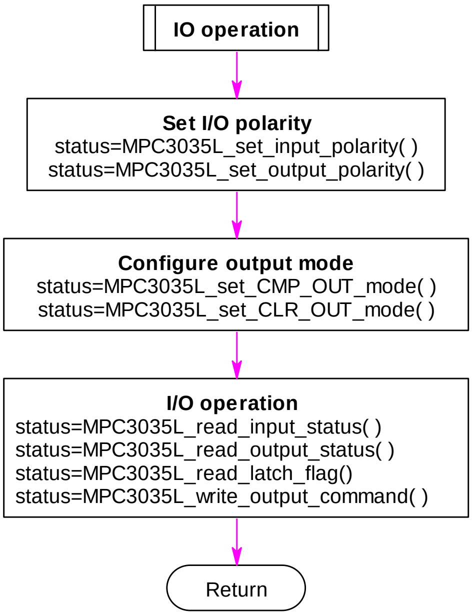
8.4.1 Using interrupt to fill FIFO

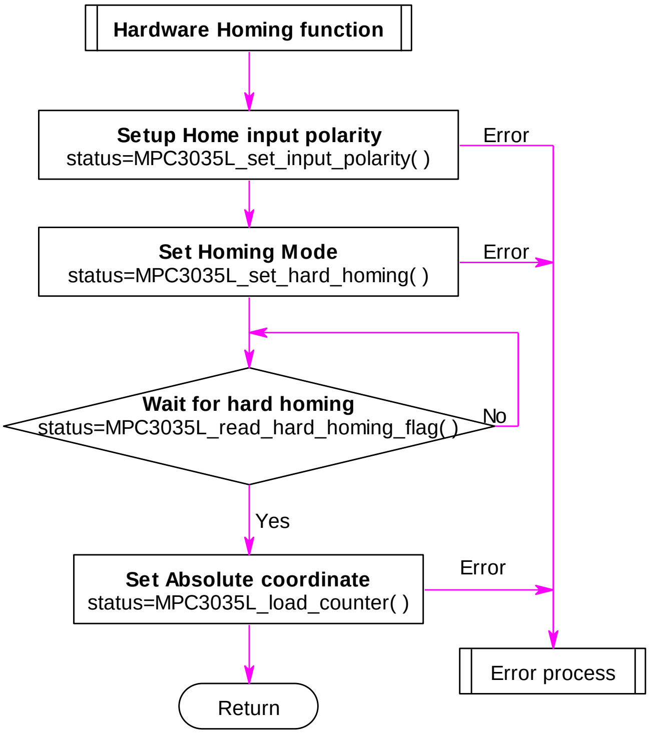

Main program
Program initial
status=MPC3035_fix_speed_range( );
FIFO PROGRAM
Change speed ratio
status=MPC3035_set_FIFO_out_Ratio( );
wait end of FIFO
Program close
status=MPC3035_enable_FIFO( );
status=MPC3035_unfix_speed_range( );
END
FIFO Interrupt service routine
Fill FIFO
status=MPC3035_T_LINE2_write_FIFO( );
set line2_index=0 for X,Y axes
set Tacc=Tdec=0 for variable speed ratio
FIFO full?
status=MPC3035_check_FIFO_buffer( );
Yes
Reset FIFO interrupt
status=MPC3035_FIFO_EOI( );
RET
FIFO program
enable FIFO function
status=MPC3035_enable_FIFO( )
say for 2 axes motion (X,Y axes):
set dimension=2,
alarm_count=200
Setup interrupt
status=MPC3035_enable_IRQ( );
status=MPC3035_link_IRQ_process( );
Link to FIFO interrupt service routine
status=MPC3035_set_INT_mask( );
set the master axis as interrupt source
for X,Y axis set X axis mask on.
status=MPC3035_set_INT_source( );
set the int source at X axis,
set REST_source_sel=0 (must be 0),
set RIST_source_sel=2 (must be 2), i.e. IRNX=1,
for generating IRQ at next operation
check motion status
status=MPC3035_read_motion_status( );
set check_factor=0,
Any pulse out?
Yes
*ret_flag=1
no pulse out
Fill FIFO
status=MPC3035_T_LINE2_write_FIFO( );
set line2_index=0 for X,Y axes
set Tacc = Tdec=0 for variable speed ratio
Check buffer full?
status=MPC3035_check_FIFO_buffer( );
*buffer_full_flag = 0
not full
Full
Run FIFO command
status=MPC3035_Run_FIFO_CMD( );
Return

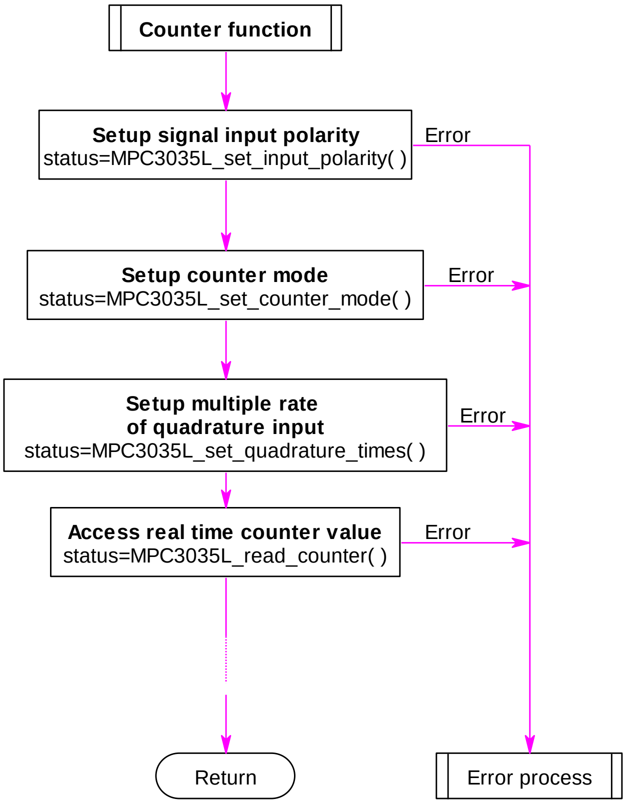
8.4.2 Using polling to fill FIFO

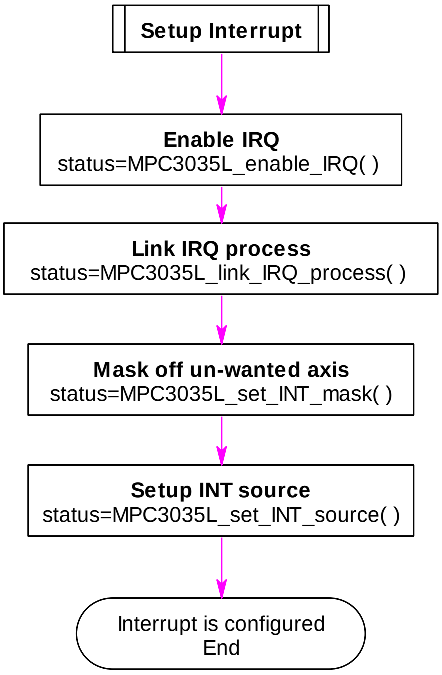

Main program
Program initial
status=MPC3035_fix_speed_range( );
FIFO PROGRAM
Check FIFO buffer
status=MPC3035_check_FIFO_buffer( );
*remain_no <
minimum remain count
Fill FIFO
status=MPC3035_T_LINE2_write_FIFO( );
set line2_index=0 for X,Y axes
set Tacc = Tdec=0 for variable speed ratio
*buffer_full_flag = 0,
not full
Check buffer full
status=MPC3035_check_FIFO_buffer( );
Full
Change speed ratio
status=MPC3035_set_FIFO_out_Ratio( );
wait end of FIFO
Program close
status=MPC3035_enable_FIFO( );
status=MPC3035_unfix_speed_range( );
END
FIFO program
enable FIFO function
status=MPC3035_enable_FIFO( )
say for 2 axes motion (X,Y axes):
set dimension=2,
alarm_count=200
check motion status
status=MPC3035_read_motion_status( );
set check_factor=0,
Any pulse out?
Yes
*ret_flag=1
no pulse out
Fill FIFO
status=MPC3035_T_LINE2_write_FIFO( );
set line2_index=0 for X,Y axes
set Tacc = Tdec=0 for variable speed ratio
*buffer_full_flag = 0
not full
Check buffer full?
status=MPC3035_check_FIFO_buffer( );
Full
Run FIFO command
status=MPC3035_Run_FIFO_CMD( );
Return

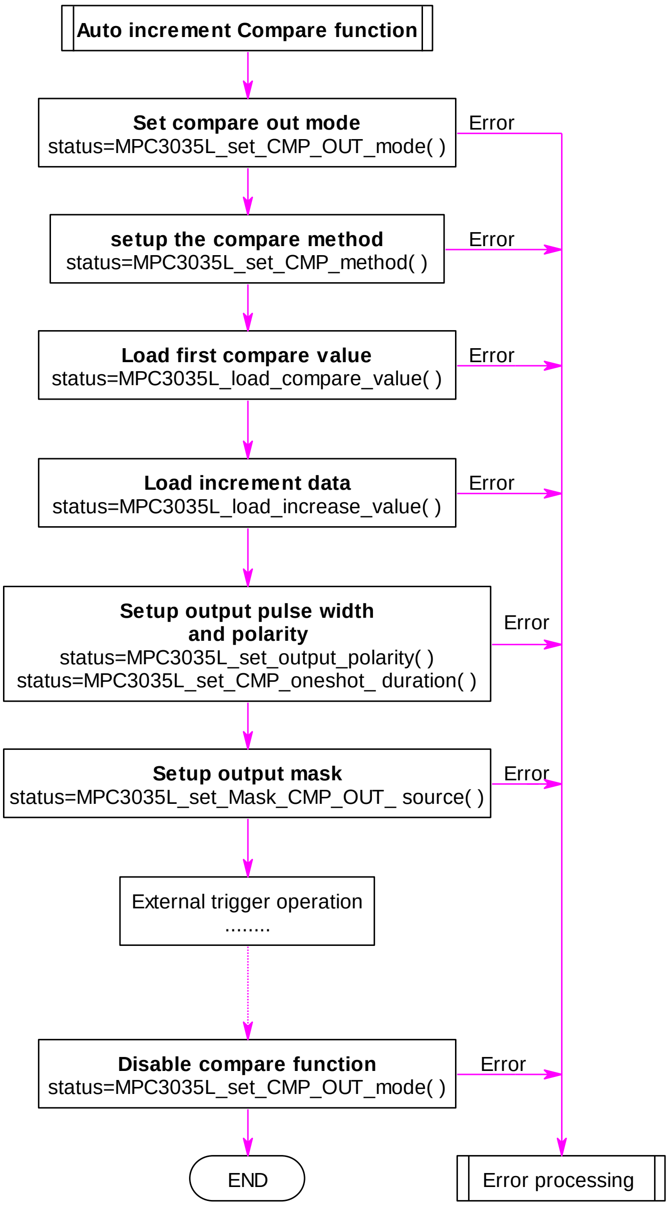
8.5 MPC3035L Flow chart of application implementation

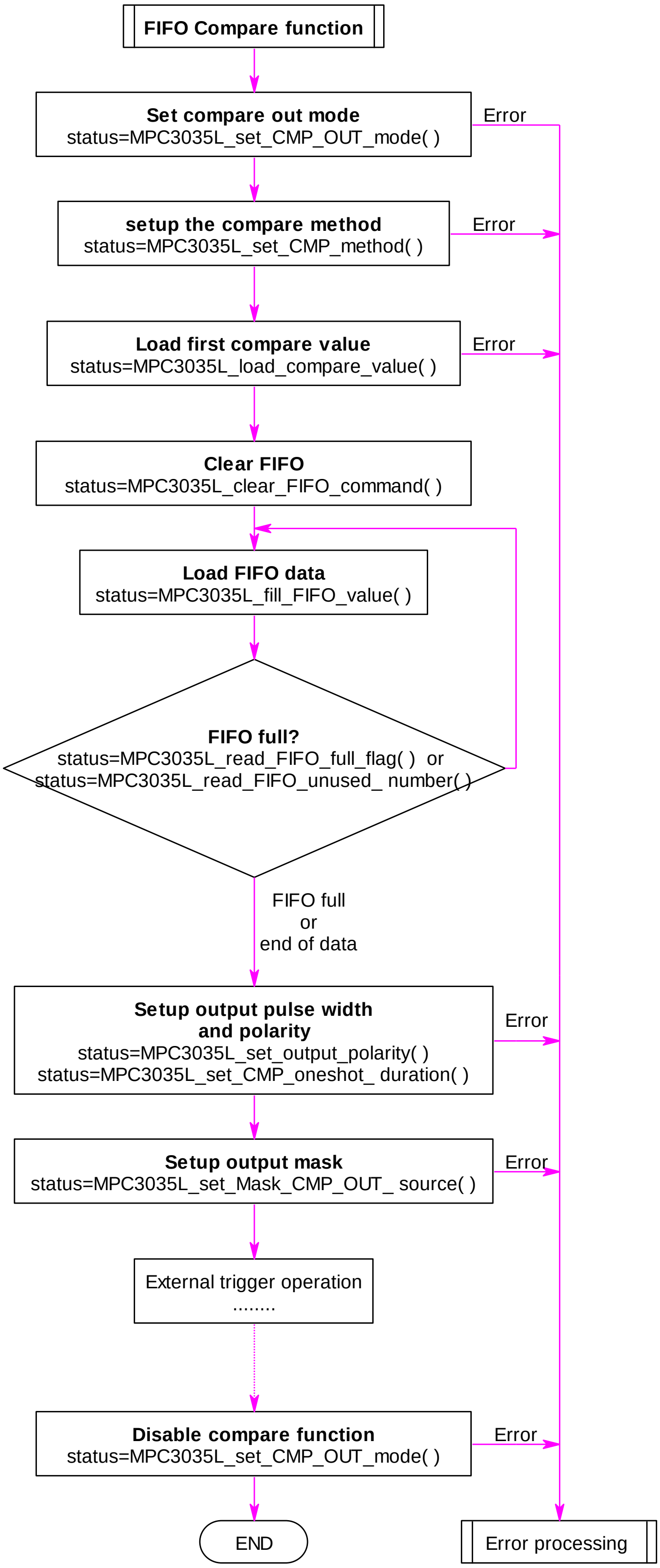

IO operation
Set I/O polarity
status=MPC3035L_set_input_polarity( )
status=MPC3035L_set_output_polarity( )
Configure output mode
status=MPC3035L_set_CMP_OUT_mode( )
status=MPC3035L_set_CLR_OUT_mode( )
I/O operation
status=MPC3035L_read_input_status( )
status=MPC3035L_read_output_status( )
status=MPC3035L_read_latch_flag()
status=MPC3035L_write_output_command( )
Return


Hardware Homing function
Setup Home input polarity
status=MPC3035L_set_input_polarity( )
Error
Set Homing Mode
status=MPC3035L_set_hard_homing( )
Error
Wait for hard homing
status=MPC3035L_read_hard_homing_flag( )
No
Yes
Set Absolute coordinate
status=MPC3035L_load_counter( )
Error
Return
Error process


Counter function
Setup signal input polarity
status=MPC3035L_set_input_polarity( )
Error
Setup counter mode
status=MPC3035L_set_counter_mode( )
Error
Setup multiple rate
of quadrature input
status=MPC3035L_set_quadrature_times( )
Error
Access real time counter value
status=MPC3035L_read_counter( )
Error
Return
Error process


Setup Interrupt
Enable IRQ
status=MPC3035L_enable_IRQ( )
Link IRQ process
status=MPC3035L_link_IRQ_process( )
Mask off un-wanted axis
status=MPC3035L_set_INT_mask( )
Setup INT source
status=MPC3035L_set_INT_source( )
Interrupt is configured
End


Auto increment Compare function
Set compare out mode
status=MPC3035L_set_CMP_OUT_mode( )
Error
setup the compare method
status=MPC3035L_set_CMP_method( )
Error
Load first compare value
status=MPC3035L_load_compare_value( )
Error
Load increment data
status=MPC3035L_load_increase_value( )
Error
Setup output pulse width
and polarity
status=MPC3035L_set_output_polarity( )
status=MPC3035L_set_CMP_oneshot_ duration( )
Error
Setup output mask
status=MPC3035L_set_Mask_CMP_OUT_ source( )
Error
External trigger operation
....
Disable compare function
status=MPC3035L_set_CMP_OUT_mode( )
Error
END
Error processing


FIFO Compare function
Set compare out mode
status=MPC3035L_set_CMP_OUT_mode( )
Error
setup the compare method
status=MPC3035L_set_CMP_method( )
Error
Load first compare value
status=MPC3035L_load_compare_value( )
Error
Clear FIFO
status=MPC3035L_clear_FIFO_command( )
Load FIFO data
status=MPC3035L_fill_FIFO_value( )
FIFO full?
status=MPC3035L_read_FIFO_full_flag( ) or
status=MPC3035L_read_FIFO_unused_ number( )
FIFO full
or
end of data
Setup output pulse width
and polarity
status=MPC3035L_set_output_polarity( )
status=MPC3035L_set_CMP_oneshot_ duration( )
Error
Setup output mask
status=MPC3035L_set_Mask_CMP_OUT_ source( )
Error
External trigger operation
....
Disable compare function
status=MPC3035L_set_CMP_OUT_mode( )
Error
END
Error processing

9. Function reference

9.1 Function format

Every MPC3035 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 MPC3035 function is the parameter **CardID** which is located the driver of MPC3035 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**)

9.2 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

9.3 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 MPC3035 API. Read the following sections that apply to your programming language.

Note : Be sure to include the declaration functions of MPC3035 prototypes by including the appropriate MPC3035 header file in your source code. Refer to Building Applications with the MPC3035 Software Library for the header file appropriate to your compiler.

9.3.1 C/C++

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

Status = MPC3035_read_point_status(CardID,Axis,check_factor,*state);

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

```
u8 CardID, axis;
u8 check_factor;
u8 state,
u32 Status;
…
Status = MPC3035_set_port (CardID, axis, check_factor, &state);
```

9.3.2 Visual basic

The file MPC3035.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 MPC3035.bas, do not use the numerical values.

In Visual Basic, you can add the entire MPC3035.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 MPC3035.bas file for your project in Visual Basic 4.0, go to the **File** menu and select the **Add File... option**. Select MPC3035.bas, which is browsed in the ..\MPC3035\API directory. Then, select **Open** to add the file to the project.

To add the MPC3035.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** MPC3035.bas, which is in the ..\MPC3035\API directory. Then, select **Open** to add the file to the project.

9.3.3 Borland C++ builder

To use Borland C++ builder as development tool, you can use the file **MPC3035BC.lib** under ..\MPC3035\API\ or generate **MPC3035BC.lib** file from the **MPC3035.dll** file by:

implib MPC3035bc.lib MPC3035.dll

Then add the **mpc3035BC.lib** to your project and add

#include "mpc3035.h" to main program.

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

Status = MPC3035_read_point_status(CardID, axis, check_factor, *state);

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

```
u8 CardID, axis;
u8 state;
u32 Status;
....
Status = MPC3035_read_point_status (CardID, axis, check_factor, &state);
```

9.4 MPC3035 Functions

Initialization and close

- **MPC3035_initial**

Format : **u32 status =MPC3035_ initial (void);**

Purpose: Initial the MPC3035 resource when start the Windows applications.

- **MPC3035_close**

Format : **u32 status =MPC3035_close (void);**

Purpose: Release the MPC3035 resource when close the Windows applications.

- **MPC3035_init_card**

Format : **u32 status =MPC3035_init_card (u8 CardID);**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

- **MPC3035_info**

Format : **u32 status =MPC3035_info(u8 CardID, 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
address	u16	physical I/O address assigned by OS

- **MPC3035_dll_Simu_mode**

Format : **u32 status =MPC3035_dll_Simu_mode(u8 enable);**

Purpose: To enable/disable simulation mode. This is designed for flow check of application program without a card plugged in.

Parameters:

Input:

Name	Type	Description
enable	u8	0: normal mode(default) 1: simulation mode

Note:

If you will run simulation to check the control flow, please apply this function before you call *MPC3035_initial( ).*

- **MPC3035_save_config2_file**

Format : **u32 status = MPC3035_save_config2_file(u8 CardID,char* file_name);**

Purpose: Save motion function parameters, motion related I/O, feedback encoder counter related I/O configuration data to file.

Parameters:

Input:

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

- **MPC3035_load_config_from_file**

Format : **u32 status = MPC3035_load_config_from_file(u8 CardID,char* file_name);**

Purpose: Load configuration data from file.

Parameters:

Input:

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

General I/O configuration and control

- **MPC3035_config_TTL_IO_MODE**

Format : **u32 status = MPC3035_config_TTL_IO_MODE(u8 CardID,u8 IO_mode);**

Purpose: To configure the TTL I/O mode.

Parameters:

Input:

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

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

- **MPC3035_readbcak_TTL_IO_MODE**

Format : **u32 status = MPC3035_readback_TTL_IO_MODE(u8 CardID, u8* IO_mode);**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

Output:

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

- **MPC3035_read_point_status**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch	
axis	u8	0: X axis 2: Z axis	1: Y axis 3: A axis
check_factor	u8	0: SD 1: PCS 2: INP 3: ALM 4: SRDY 5: LS+ (EL+) 6: LS- (EL-) 7: LTC 8: HOME(ORG) 9: EMG 10: EZ 11: ERC 12: SVON 13: FIN 14: CMP 15: CSTA 16: CSTP 17: DIO0 18: DIO1 19: DIO2 20: DIO3 21: DIO4 22: DIO5 23: DIO6 24: DIO7	(Slow Down input) (Position change start input) (In position input) (servo driver alarm input) (servo driver ready input) (positive side over travel limit switch) (negative side over travel limit switch) (external latch trigger input) (home(ORG) sensor input) (emergency input) (encoder zero phase input) (error counter output status) (servo driver on output status) (finish output) (compare equal output) (common start input) (common stop input) (TTL IO bit0 status) (TTL IO bit1 status) (TTL IO bit2 status) (TTL IO bit3 status) (TTL IO bit4 status) (TTL IO bit5 status) (TTL IO bit6 status) (TTL IO bit7 status)

Output:

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

- **MPC3035_read_status**

Format : **u32 status = MPC3035_read_status(u8 CardID,u8 Axis,u32 *data);**

Purpose: To input status.

Parameters:

Input:

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

Output:

Name	Type	Description		
data	u32	any bit of the following bit reads “1” means if the source is active.		
		Bit	Name	Description
		bit0	SD	(Slow Down input)
		bit1	PCS	(Position change start input)
		bit2	INP	(In position input)
		bit3	ALM	(servo driver alarm input)
		bit4	SRDY	(servo driver ready input)
		bit5	LS+(EL+)	(positive side over travel limit switch)
		bit6	LS-(EL-)	(negative side over travel limit switch)
		bit7	LTC	(external latch trigger input)
		bit8	HOME(ORG)	(home(ORG) sensor input)
		bit9	EMG	(emergency input)
		bit10	EZ	(encoder zero phase input)
		bit11	ERC	(error counter output status)
		bit12	SVON	(servo driver on output status)
		bit13	FIN	(finish output)
		bit14	CMP	(compare equal output)
		bit15	CSTA	(common start input)
		bit16	CSTP	(common stop input)

- **MPC3035_read_port**

Format : **u32 status = MPC3035_read_port(u8 CardID, u8 *data);**

Purpose: To input TTL_IO port status.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Output:

Name	Type	Description
data	u8	TTL/IO port data

- **MPC3035_write_output_point**

Format : **u32 status = MPC3035_write_output_point(u8 CardID,u8 Axis,u8 point_factor, u8 on_off);**

Purpose: To set/reset output point.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
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: DIO0 (TTL IO bit0 status) 5: DIO1 (TTL IO bit1 status) 6: DIO2 (TTL IO bit2 status) 7: DIO3 (TTL IO bit3 status) 8: DIO4 (TTL IO bit4 status) 9: DIO5 (TTL IO bit5 status) 10: DIO6 (TTL IO bit6 status) 11: DIO7 (TTL IO bit7 status)
on_off	u8	0: reset, inactive 1: set, active

Note on some output:

Name	Description
ERC	Ref. **Note on ERC function** MPC3035_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 driver servo on 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 motion finish and under control by user program, not by motion processor.
CMP	Ref. **Note on CMP function** MPC3035_config_CMP_OUT

- **MPC3035_write_port**

Format : **u32 status = MPC3035_write_port(u8 CardID, u8 data);**

Purpose: To set/reset TTL_IO port

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
data	u8	TTL/IO port data

Setup for motion control

- **MPC3035_set_pulse_outmode**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
pulse_outmode	u8	0~5 (See Note on pulse out mode)

Note:

On wiring board terminal marked as CW+, CW- (differential output) for cw and CCW+,CCW- for ccw signal.

- **MPC3035_readback_pulse_outmode**

Format : **u32 status = MPC3035_readback_pulse_outmode(u8 CardID,u8 Axis,**
u8* pulse_outmode);

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

Parameters:

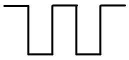
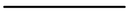
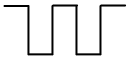
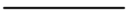
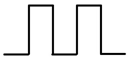
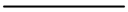
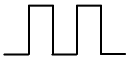
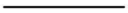
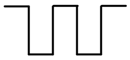
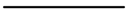
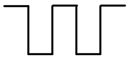
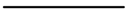
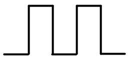
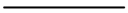
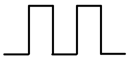
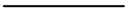
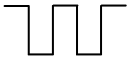
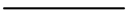
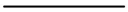
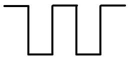
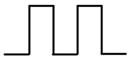
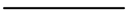
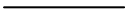
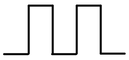
Input:

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

Output:

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

Note on pulse out mode:

Pulse_ outmode	Operation in plus direction		Operation in minus direction		Comments
	CW+ terminal of wiring board	CCW+ terminal of wiring board	CW+ terminal of wiring board	CCW+ terminal of wiring board	
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

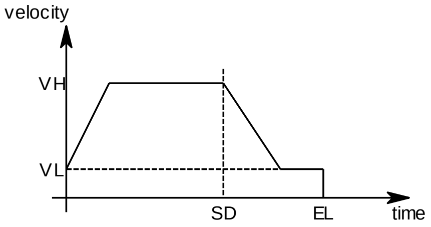
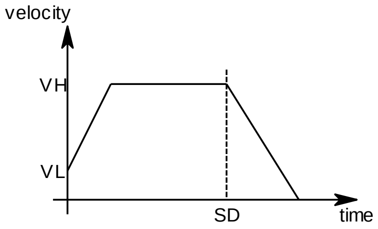
- **MPC3035_config_SD_PIN**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
enable	u8	0: treat SD PIN as a general input. 1: treat SD PIN as a dedicated slow down signal input.
sd_logic	u8	0: setting the pin 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. 

Note: On wiring board terminal marked as SD

- **MPC3035_readback_SD_PIN**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch	
axis	u8	0: X axis 2: Z axis	1: Y axis 3: A axis

Output:

Name	Type	Description
enable	u8	0: treat SD PIN as a general input. 1: treat SD PIN as a dedicated slow down signal input.
sd_logic	u8	0: setting the pin 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

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.**

MPC3035_config_EL_MODE

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
el_mode	u8	0: immediate stop. 1: decelerate to stop

Note:

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

MPC3035_readback_EL_MODE

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

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

- **MPC3035_config_PCS_PIN**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
enable	u8	0: treat PCS PIN as a general input. 1: treat PCS PIN as a dedicated position change start input.
pcs_logic	u8	0: setting the pin 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.

Note:

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

- **MPC3035_readback_PCS_PIN**

Format : **u32 status = MPC3035_readback_PCS_PIN(u8 CradID, u8 Axis, u8* enable, u8* pcs_logic,u8 *state);**

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

Parameters:

Input:

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

Output:

Name	Type	Description
enable	u8	0: treat PCS PIN as a general input. 1: treat PCS PIN as a dedicated position change start input.
pcs_logic	u8	0: setting the pin 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.
state	u8	state of PCS pin

Note on PCS function:

Name	Description
PCS	PCS pin is external triggered position change function input pin.

MPC3035_position_override

Format : **u32 status = MPC3035_position_override(u8 CardID,u8 Axis,i32 distance, u8 trigger_mode);**

Purpose: To override the target position on the fly.

Parameters:

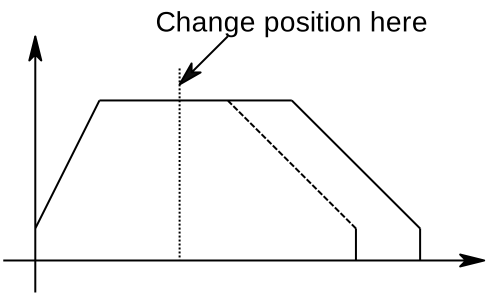
Input:

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

Note:

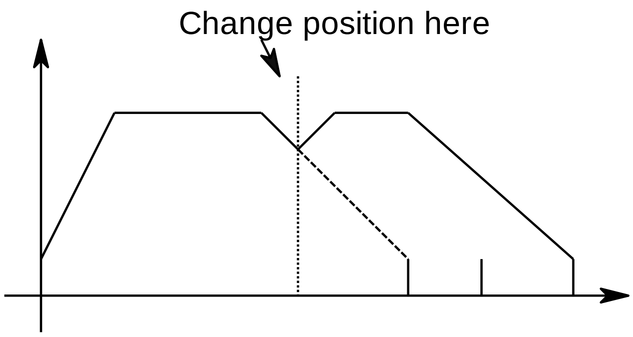
1. The position override is the relative distance to go of the current position on PCS active or on software override command.
2. The override is only valid while the motion is active (in motion state)
3. Even you use ***MPC3035_position_override( )*** in software trigger mode, **you must configure PCS input as dedicated for correct software trigger function.**
4. The direction of initial position to target position must be the same as current position to override position.
5. Some known conditions and suggestion

   Condition1:



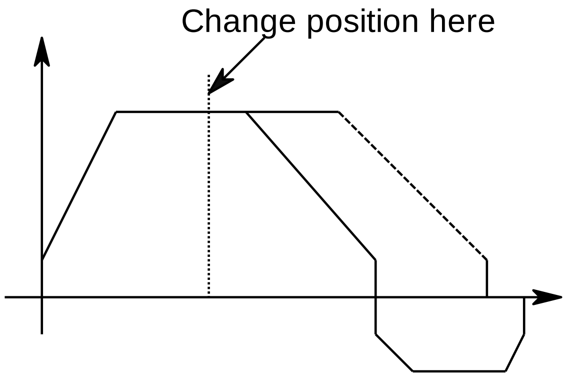
Override while motion in constant speed region.
The remained distance is smaller than the override target – original target
This is the most preferred condition.

Condition2:



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

Condition3:



Override while motion in constant speed region.
The remained distance is larger than the override target – original target
Not preferred, some condition will run abnormal.
Software trigger will cause the motion continuous run, please avoid this condition.

- **MPC3035_config_ERC_PIN**

Format : **u32 status = MPC3035_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:

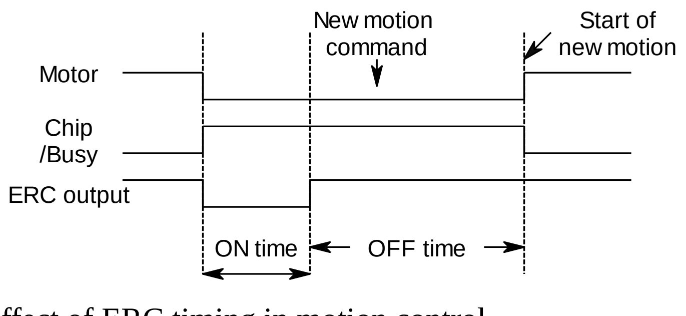
Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
enable	u8	0: treat ERC PIN as a manual error counter clear output. 1: treat ERC PIN as a automatic error counter clear output.
erc_logic	u8	0: setting the pin 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.
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 wiring board terminal marked as ERC, recommend to connect to servo driver clear error counter input, different brand driver has different terminal name such as HOLD, CL(counter clear)…

Please refer the MPC3035_config_home_mode( ) for the event of generating ERC.



Effect of ERC timing in motion control

MPC3035_readback_ERC_PIN

Format : u32 status = MPC3035_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 DIP/ROTARY switch	
axis	u8	0: X axis 2: Z axis	1: Y axis 3: A axis

Output:

Name	Type	Description	
enable	u8	0: treat ERC PIN as a manual error counter clear output. 1: treat ERC PIN as a automatic error counter clear output.	
erc_logic	u8	0: setting the pin 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.	
erc_on_time	u8	0: on time 12us 2: on time 408us 4: on time 13ms 6: on time 104ms	1: on time 102us 3: on time 1.6ms 5: on time 52ms 7: erc level out
erc_off_time	u8	0: off time 0s 2: off time 1.6ms	1: off time 12us 3: off time 104ms
state	u8	state of ERC pin	

Note on ERC function:

Name	Description
ERC	ERC pin is error counter clear output pin. In a pulse type control system, the pulse is generated by the processor and the driver accepts the pulse train doing the motion job and feedback control. During homing, the processor detect the home(ORG) sensor and stop the pulse train, but the driver does not know the system is ‘homed’, the remain clock (which is accumulated in error counter) should be cleared to keep the system accuracy. While enables this function, the ERC output will be triggered automatically by the conditions met, and new motion command will not accept until the ERC output time out complete.(erc_on_time + erc_off_time). If you disable it (ie. manual control mode), use **MPC3035_write_output_point** to control ERC**,** the active state of ERC will also stop the motion pulses. **Do not use ERC as general output.**

- **MPC3035_config_ALM_PIN**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
alm_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.
alm_action	u8	0: immediate stop 1: decelerate to stop

Note:

On wiring board terminal marked as ALM, recommend to connect to servo driver alarm output.

- **MPC3035_readback_ALM_PIN**

Format : **u32 status = MPC3035_readback_ALM_PIN(u8 CardID,u8 Axis, u8* alm_logic ,u8* alm_action,u8*state);**

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

Parameters:

Input:

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

Output:

Name	Type	Description
alm_logic	u8	0: setting the pin 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.
alm_action	u8	0: immediate stop 1: decelerate to stop
state	u8	staet of ALM pin

● MPC3035_config_INP_PIN

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
enable	u8	0: treat INP PIN as a general input. 1: treat INP PIN as a dedicated in position input.
inp_logic	u8	0: setting the pin 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.

Note: On wiring board terminal marked as INP

- **MPC3035_readback_INP_PIN**

Format : **u32 status = MPC3035_readback_INP_PIN(u8 CardID,u8 Axis,u8* enable, u8* inp_logic,u8 *state);**

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

Parameters:

Input:

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

Output:

Name	Type	Description
enable	u8	0: treat INP PIN as a general input. 1: treat INP PIN as a dedicated in position input.
inp_logic	u8	0: setting the pin 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.
state	u8	state of INP pin

Note on INP function:

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

Velocity mode motion

- **MPC3035_fix_speed_range**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
Vmax	i32	max pps (0~6553500)

- **MPC3035_unfix_speed_range**

Format : **u32 status = MPC3035_fix_speed_range(u8 CardID,u8 Axis);**

Purpose: To release the maximum allowable speed.

Parameters:

Input:

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

- **MPC3035_T_velocity_move**

Format : **u32 status = MPC3035_T_velocity_move(u8 CardID,u8 Axis,i32 VL,i32 VH, f64 Tacc);**

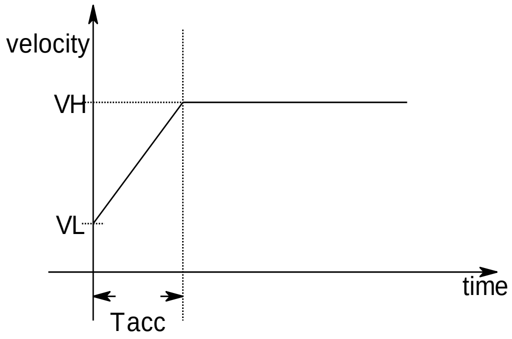
Purpose: Doing velocity mode movement at trapezoidal profile**.** The final speed will be at VH.

Parameters:

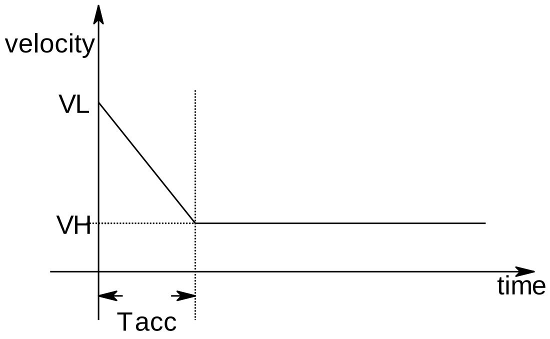
Input:

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

Note:

VH and VL must both positive or negative, you can not have one positive and the other negative.

- **MPC3035_S_velocity_move**

Format : **u32 status = MPC3035_S_velocity_move(u8 CardID,u8 Axis,i32 VL,i32 VH, f64 Tacc,u32 SVacc);**

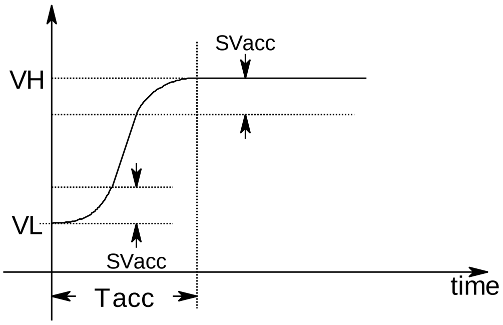
Purpose : Doing velocity mode movement at S curve profile. The final speed will be at VH.

Parameters:

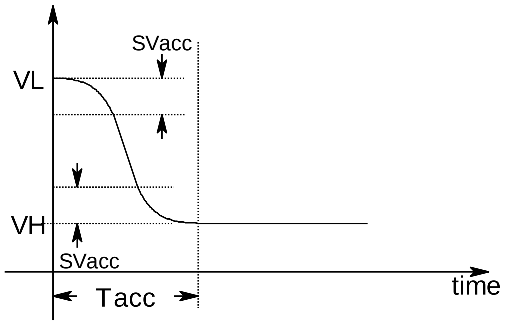
Input:

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

Note:

VH and VL must both positive or negative, you can not have one positive and the other negative.

MPC3035_velocity_change

Format : **u32 status = MPC3035_velocity_change(u8 CardID,u8 Axis,i32 Vn,f64 Tacc);**

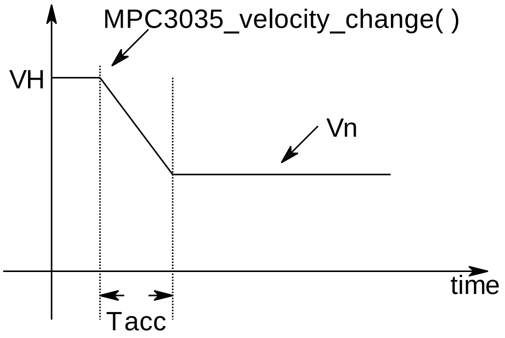
Purpose: Change speed to Vn (with the trapezoidal/S curve mode previously defined) at velocity mode.

Parameters:

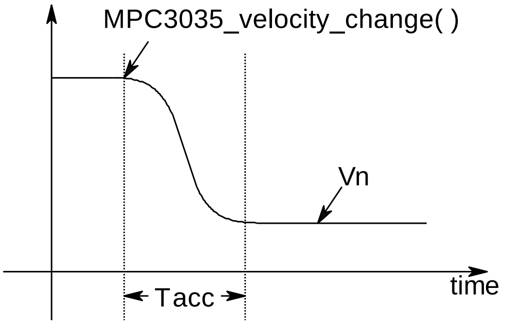
Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
Vn	i32	new speed in pps, -6553500~6553500 |Vn| ≤ Vmax (set by MPC3035_fix_speed_range( ))
Tacc	f64	acceleration time in seconds

Note on velocity change:



Velocity change at trapzoidal acc/dec



Velocity change at S curve acc/dec

* If you use **MPC3035_velocity_change** to change speed, while you want to change direction, be sure to use **MPC3035_velocity_change** to decrease the speed to zero before change direction. The functions **MPC3035_S_velocity_move** and **MPC3035_T_velocity_move** are no need to switch to zero speed.

- **MPC3035_dec_stop**

Format : **u32 status = MPC3035_dec_stop(u8 CardID,u8 Axis,f64 Tdec);**

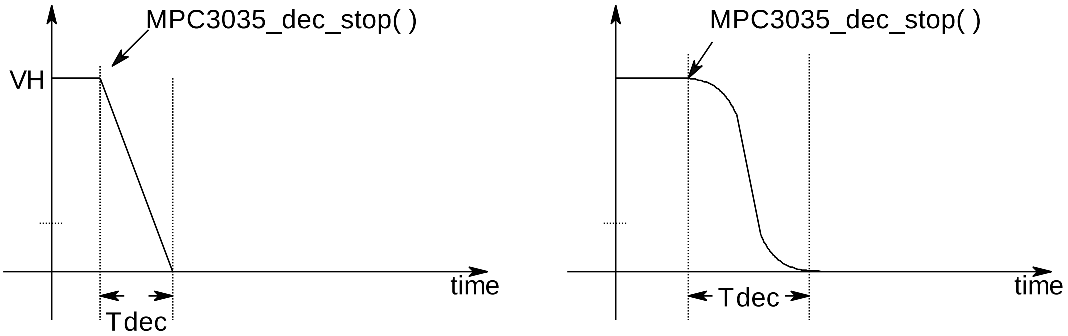
Purpose: Command to decelerate to stop (with the trapezoidal/S curve mode previously defined) at velocity mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
Tdec	f64	deceleration time in seconds

Note on decelerate to stop:



Dec stop at trapzoidal acc/dec

Dec stop at S curve acc/dec

- **MPC3035_imd_stop**

Format : **u32 status = MPC3035_imd_stop(u8 CardID,u8 Axis);**

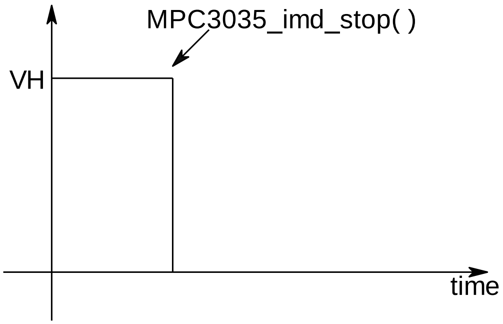
Purpose: Command the designated axis to immediate stop at velocity mode.

Parameters:

Input:

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

Note on immediate stop:



Immediate stop

- ## MPC3035_emg_stop

Format : **u32 status = MPC3035_emg_stop(u8 CardID);**

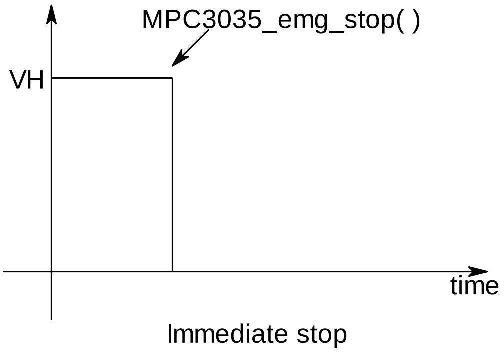
Purpose: Command all axes to immediate stop at velocity mode.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

Note on immediate stop:



- ## MPC3035_read_speed

Format : **u32 status = MPC3035_read_speed(u8 CardID,u8 Axis,f64 *speed);**

Purpose: To read the current speed.

Parameters:

Input:

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

Output:

Name	Type	Description
speed	f64	current speed in pps

Homing

- **MPC3035_set_HOME_pin_logic**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
home_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.

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

- **MPC3035_readback_HOME_pin_logic**

Format : **u32 status = MPC3035_readback_HOME_pin_logic(u8 CardID,u8 Axis, u8* home_logic);**

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

Parameters:

Input:

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

Output:

Name	Type	Description
home_logic	u8	0: setting the pin 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.

- **MPC3035_set_EZ_pin_logic**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
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.

Note:

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

- **MPC3035_readback_EZ_pin_logic**

Format : **u32 status = MPC3035_readback_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

Name	Type	Description
ez_logic	u8	0: setting the pin 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.

- **MPC3035_config_home_mode**

Format : **u32 status = MPC3035_config_home_mode(u8 CardID,u8 Axis,u8 mode, u8 EZ_count);**

Purpose: To configure the homing mode.

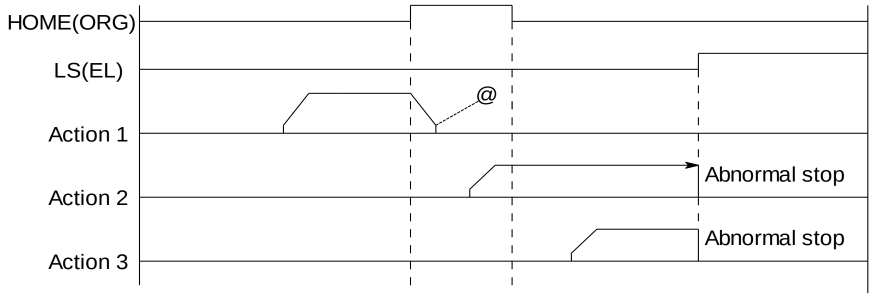
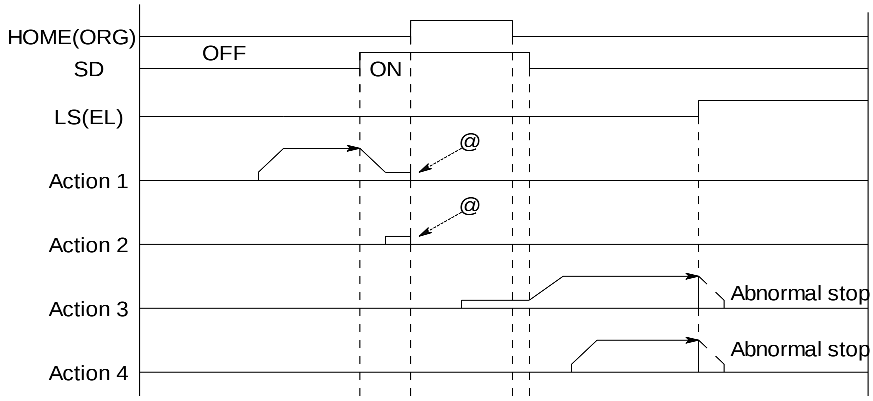
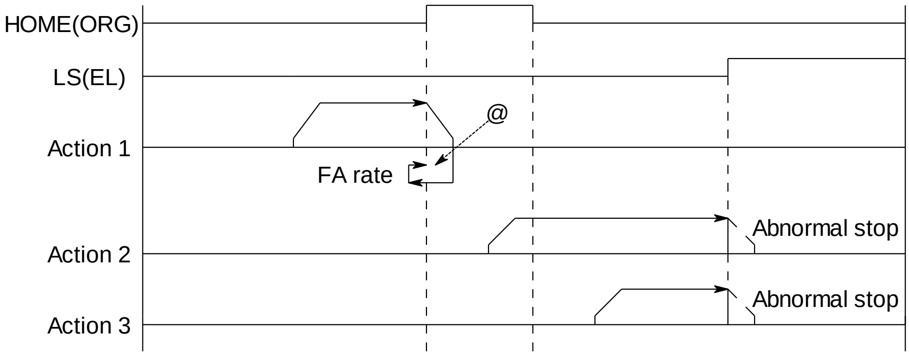
Parameters:

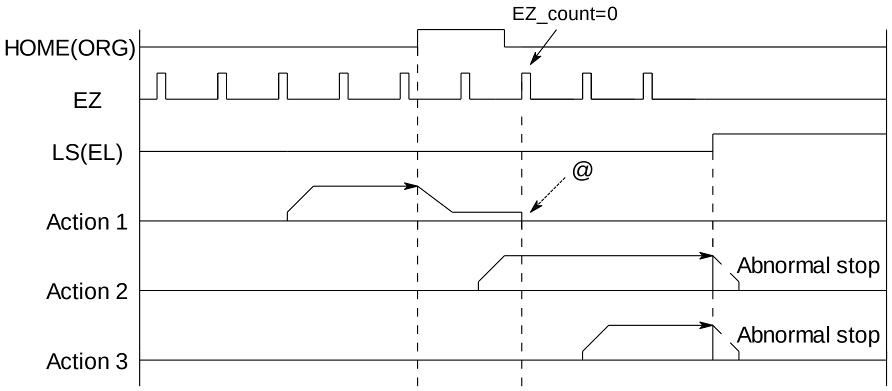
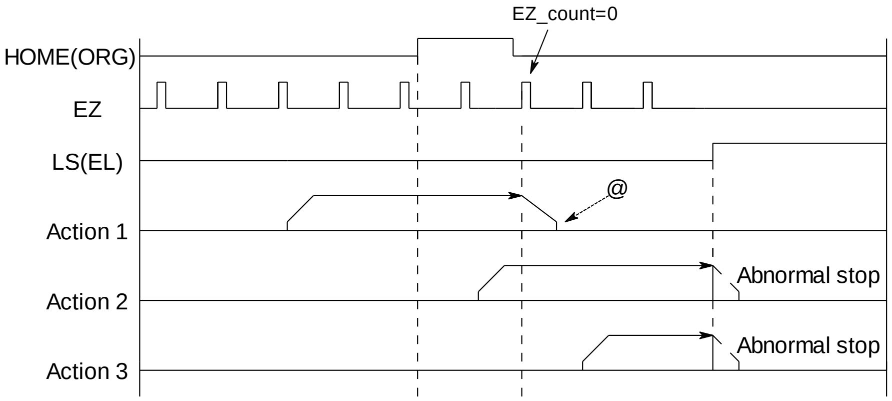
Input:

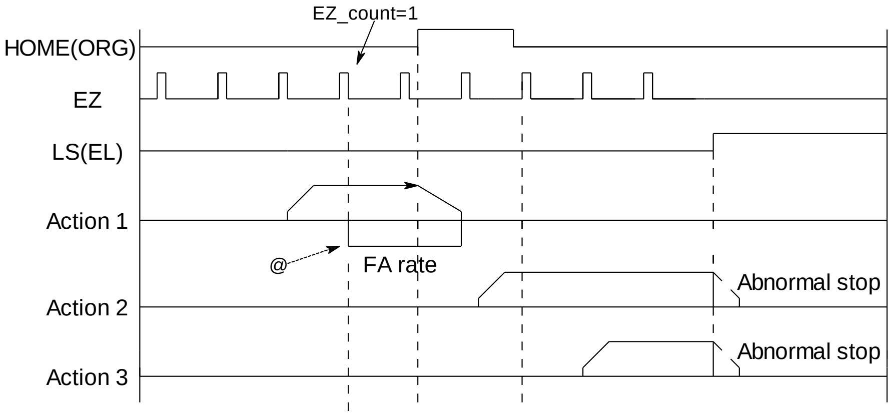
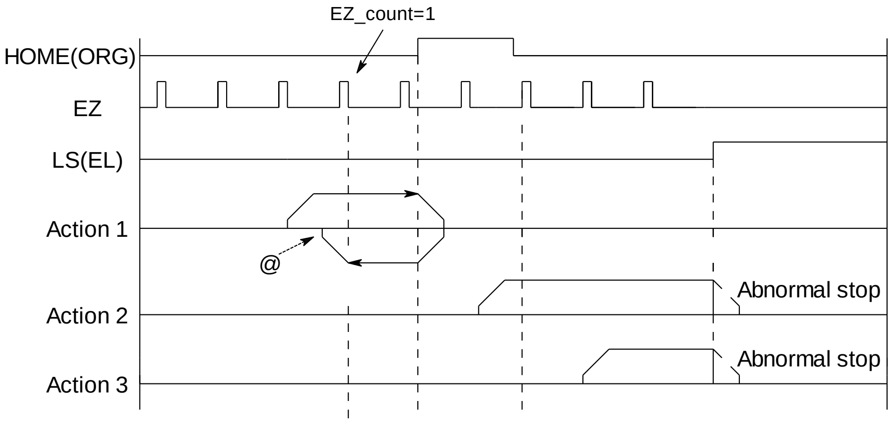
Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
mode	u8	homing mode 0~12$_{(10)}$
EZ_count	u8	Clear current position counter at the pulse numbers of zero phase input after home(ORG) switch is activated. EZ_count=0, means 1 zero phase count.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

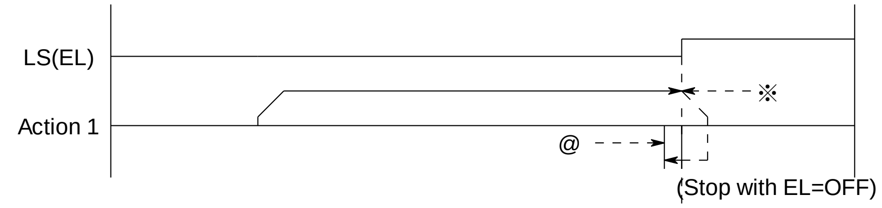
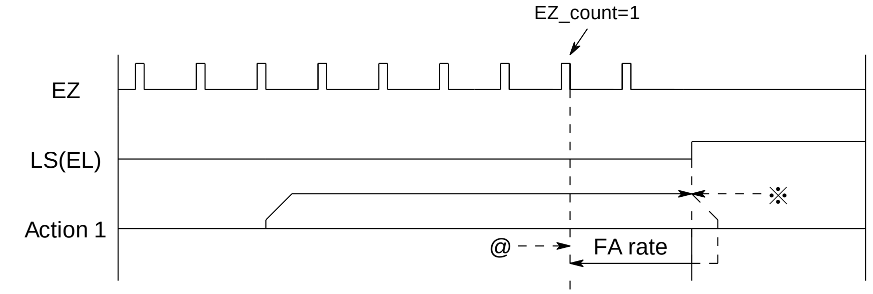
Note on homing mode:

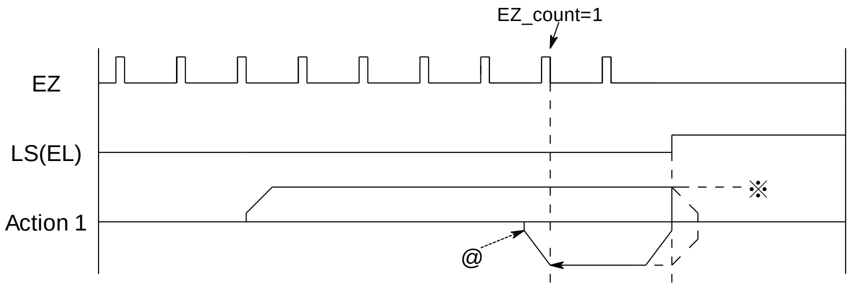
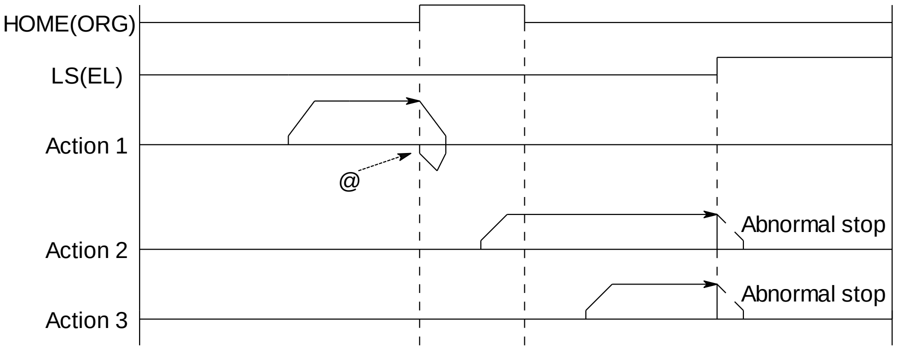
The mark @ is the homed position and if you enable ERC pin, the ERC signal will be active. The ERC function is configured by **MPC3035_config_ERC_PIN( ).**

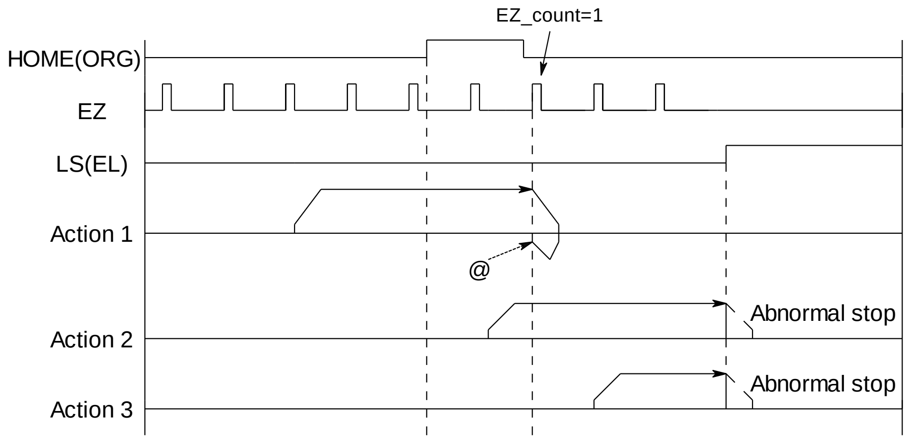
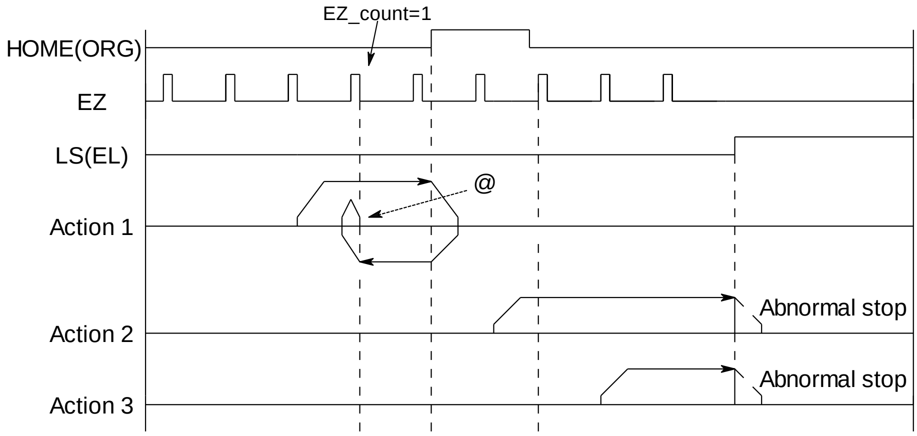
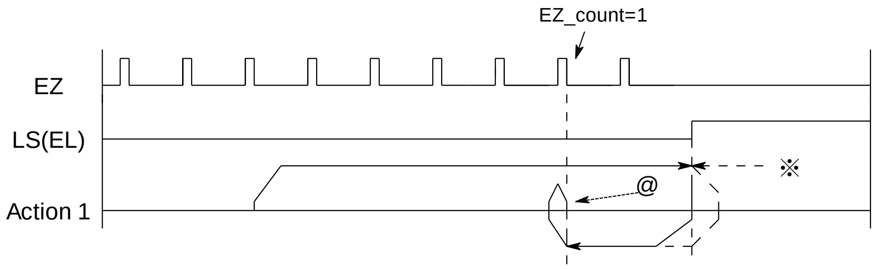
Mode	Description
0	 HOME(ORG) LS(EL) @ Action 1 Action 2 Abnormal stop Action 3 Abnormal stop  HOME(ORG) OFF SD ON LS(EL) @ Action 1 @ Action 2 Action 3 Abnormal stop Action 4 Abnormal stop Mode0: HOME(ORG) signal turning from OFF to ON causes stop after deceleration to VL speed. The current position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

2	 EZ_count=0 HOME(ORG) EZ LS(EL) @ Action 1 Abnormal stop Action 2 Abnormal stop Action 3 Mode2: The chip decelerates pulse output to VL when the HOME(ORG) signal turns from OFF to ON and stops immediately upon the EZ counter counting up to the preset value. The current position counter is reset upon the EZ counter counting up to the preset value. **Note :** @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.
3	 EZ_count=0 HOME(ORG) EZ LS(EL) @ Action 1 Abnormal stop Action 2 Abnormal stop Action 3 Mode3: The chip decelerates to VL speed and stops pulse output upon the EZ counter counting up to the preset value after the HOME(ORG) signal turns from OFF to ON. The counter is reset upon the EZ counter counting up to the preset value. **Note :** @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.

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

6	 Mode6: The chip immediately stops pulse output (stops after deceleration if ELM (el_mode)=1) upon the LS(EL) signal turning ON and then generates pulses in reverse direction at the RFA rate (Backlash speed) before immediate stop again upon the LS(EL) signal turning off. The current position counter is reset when the LS(EL) signal turns off . **Note :** @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.
7	 Mode7: The chip immediately stops pulse output (stops after deceleration if ELM (el_mode) =1) and then generates pulses in reverse direction at the RFA rate (Backlash speed) before immediate stop again upon the EZ counter counting up to the preset value. The counter is reset at the immediate stop upon the EZ counter counting up to the preset value. **Note :** @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.

8	 Mode8: The chip immediately stops pulse output (stops after deceleration if ELM (el_mode) =1) and then generates pulses in reverse direction before stop after deceleration to VL speed upon the EZ counter counting up to the preset value. The counter is reset upon the EZ counter counting up to the preset value. **Note :** @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.
9	 HOME(ORG) LS(EL) Action 1 @ Action 2 Abnormal stop Action 3 Abnormal stop Mode9: After performing origin return mode 0, the chip generates pulses to return to @ point, that is, until the FB counter counts down to 0. **Note :** @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.

10	 EZ_count=1 HOME(ORG) EZ LS(EL) Action 1 @ Action 2 Abnormal stop Action 3 Abnormal stop Mode10: After performing origin return mode 3, the chip generates pulses to return to @ point, that is, until the FB counter counts down to 0. **Note :** @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.
11	 EZ_count=1 HOME(ORG) EZ LS(EL) @ Action 1 Action 2 Abnormal stop Action 3 Abnormal stop Mode11: After performing origin return mode 5, the chip generates pulses to return to @ point, that is, until the FB counter counts down to 0. **Note :** @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.
12	 Mode12: After performing origin return mode 8, the chip generates pulses to return to 0 point, that is, until the FB counter counts down to 0. **Note :** @ position is the ERC signal output when it is configured as “ automatic output of ERC signal “.

- **MPC3035_start_homing**

Format : **u32 status = MPC3035_start_homing(u8 CardID,u8 Axis,i32 VL,i32 VH, f64 Tacc,u8 direction);**

Purpose: Command to start homing motion.

Parameters:

Input:

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

- **MPC3035_set_current_position**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
current_posi	i32	coordinate value, -134,217,728≦ current_posi ≦134,217,727

Note on set current position:

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

MPC3035_read_current_position

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis

Output:

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

Note: Current position is cleared at application initialization (initial( )) and homing.

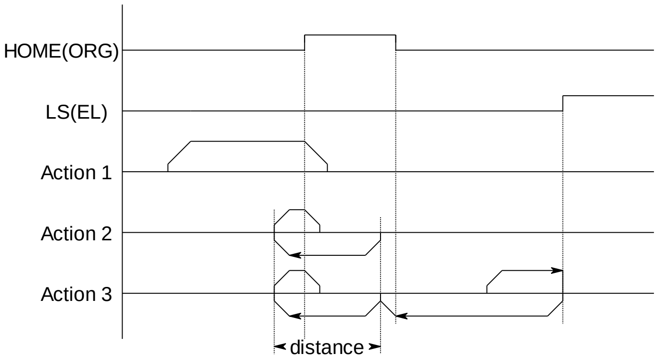
MPC3035_start_origin_search_homing

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
VL	i32	pps of start speed (0~6553500)
VH	i32	pps of final speed (0~6553500)
Tacc	f64	acceleration time
direction	u8	direction of homing 0: positive direction 1: negative direction
distance	u32	 HOME(ORG) LS(EL) Action 1 Action 2 Action 3 distance

Point to point motion control

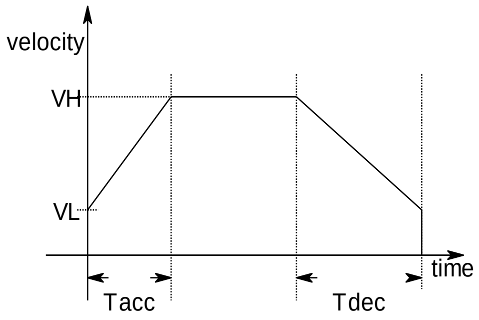
● MPC3035_T_curve_position_move

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
Position	i32	relative distance to move absolute coordinate to move (-134,217,728 ≦ Position ≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 VH,VL:pps, start speed (0 ≦ VL ≦ 6553500) Tacc,Tdec: seconds.
VH	i32	
Tacc	f64	
Tdec	f64	

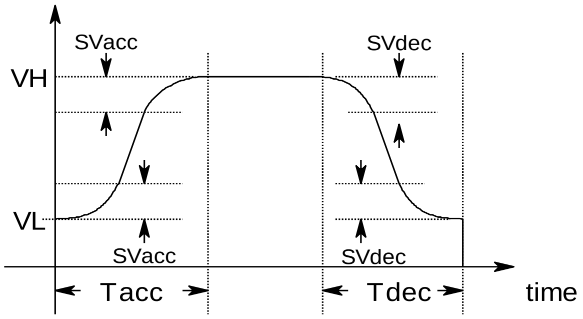
● MPC3035_S_curve_position_move

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
Position	i32	0: relative distance to move 1: absolute coordinate to move (-134,217,728 ≦ Position ≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VL	i32	 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. For point to point motion control in continuous mode (MPC3035_set_continuous_flag ( ), conti_flag=1), be sure to check continuous buffer (MPC3035_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 (MPC3035_fix_speed_range( )).
3. In non-continuous mode(MPC3035_set_continuous_flag( )，conti_flag=0), be sure to check (MPC3035_read_motion_status( ); check_factor=0，ret_flag =1) to confirm the motion is ready.

- **MPC3035_position_change**

Format : **u32 status = MPC3035_position_change(u8 CardID,u8 Axis,i32 New_pos, u8 posi_mode);**

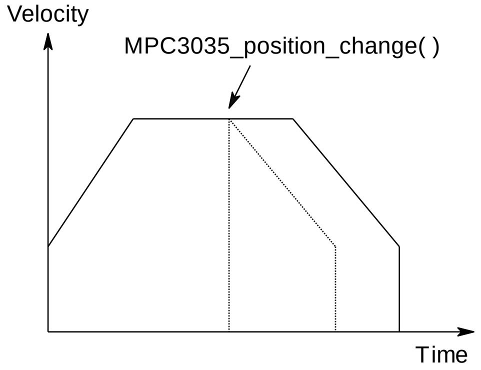
Purpose: To change positioning while point to point motion is running.

Parameters:

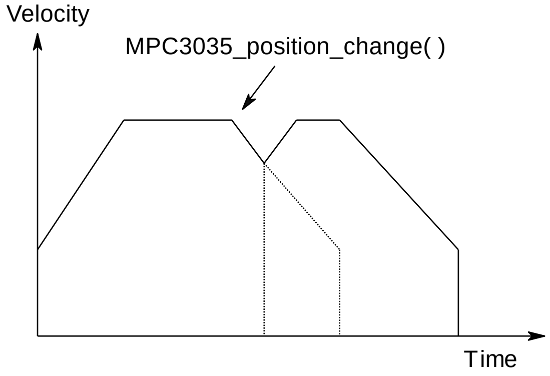
Input:

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

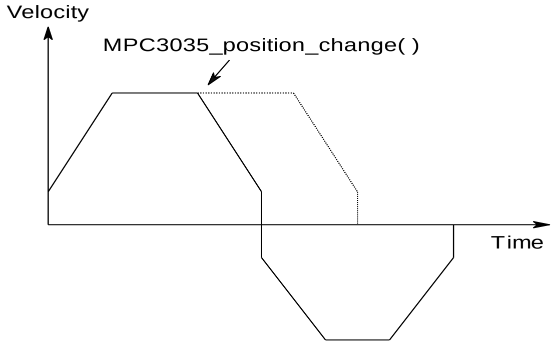
Note on position change:



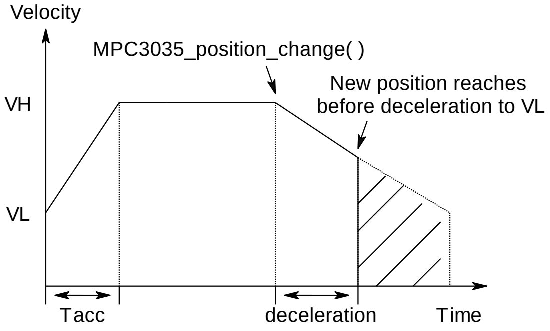
1. Command to change position at VH range



2. Command to change position at deceleration range



3. New position at different side



4. New position at mid-way of deceleration range

- **MPC3035_backlash_comp**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A 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)

Note: The backlash compensation will be made every time the moving direction changes.

- **MPC3035_readback_backlash_comp**

Format : **u32 status = MPC3035_readback_backlash_comp(u8 CardID,u8 Axis,**
u16* backlash_pulse,u8* backlash_dir,u32* backlash_speed);

Purpose: Readback configuration of backlash compensation.

Parameters:

Input:

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

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)

- **MPC3035_suppress_vibration**

Format : **u32 status = MPC3035_suppress_vibration(u8 CardID,u8 Axis,u16 RT,u16 FT);**

Purpose: To setup vibration suppression mode.

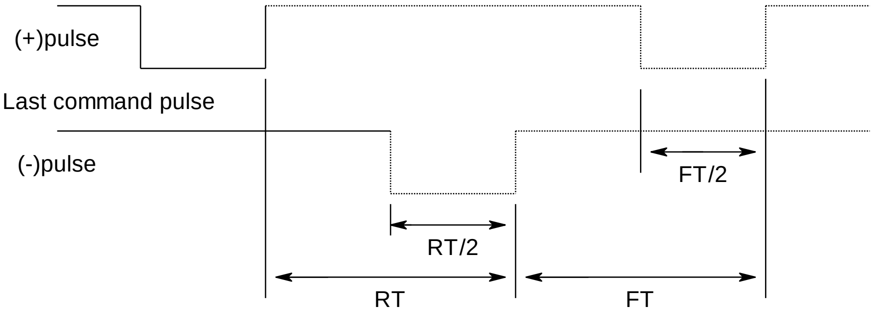
Parameters:

Input:

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

Note on vibration suppression:

The MPC3035 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.



- **MPC3035_readback_suppress_vibration**

Format : **u32 status = MPC3035_readback_suppress_vibration(u8 CardID,u8 Axis, u16* RT,u16* FT);**

Purpose: Readback parameters of vibration suppression mode**.**

Parameters:

Input:

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

Output:

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

Linear interpolation motion control

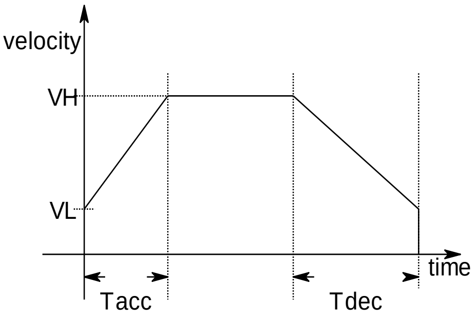
- **MPC3035_T_curve_move_LINE2**

Format : **u32 status = MPC3035_T_curve_move_LINE2(u8 CardID,u8 line2_index, 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 DIP/ROTARY switch
line2_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
Position1	i32	target position (absolute or relative) for the first axis (-134,217,728 ≦Position1≦134,217,727) for example: line2_index=2, the first axis is X
Position2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦Position2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VL	i32	 VH,VL:pps, start speed (0 ≦ VL ≦ 6553500) Tacc,Tdec: seconds.
VH	i32	
Tacc	f64	
Tdec	f64	

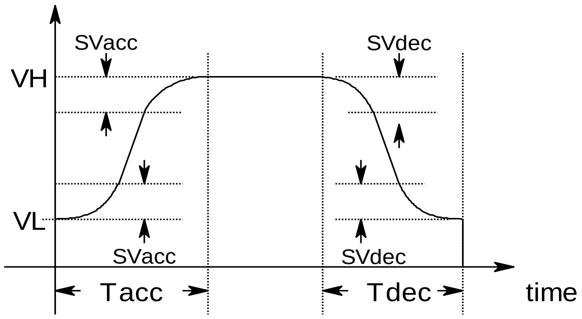
- **MPC3035_S_curve_move_LINE2**

Format : **u32 status = MPC3035_S_curve_move_LINE2(u8 CardID,u8 line2_index,**
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 DIP/ROTARY switch
line2_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
Position1	i32	target position (absolute or relative) for the first axis (-134,217,728 ≦Position1≦134,217,727) for example: line2_index=2, the first axis is X
Position2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦Position2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	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 linear interpolation:

1. For linear interpolation motion control in continuous mode (MPC3035_set_continuous_flag ( ), conti_flag=1), be sure to check continuous buffer (MPC3035_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 (MPC3035_fix_speed_range( )) at the operation axes.
3. In non-continuous mode(MPC3035_set_continuous_flag( )，conti_flag=0), be sure to check (MPC3035_read_motion_status( ); check_factor=0，ret_flag =1) to confirm the motion axes are ready.
4. The remained axes maybe programmed as point to point , linear or circular interpolation mode.

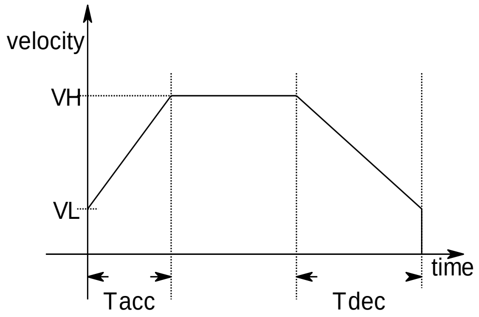
MPC3035_T_curve_move_LINE3

Format : **u32 status = MPC3035_T_curve_move_LINE3(u8 CardID,u8 line3_index,**
i32 Position1,i32 Position2,i32 Position3,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 DIP/ROTARY switch
line3_index	u8	0: X,Y, Z 1: X, Y, A 2: X, Z, A 3: Y, Z, A
Position1	i32	target position (absolute or relative) for the first axis (-134,217,728 ≦Position1≦134,217,727) for example: line3_index=2, the first axis is X
Position2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦Position2≦134,217,727) for example: line3_index=2, the second axis is Z
Position3	i32	target position (absolute or relative) for the third axis (-134,217,728 ≦Position3≦134,217,727) for example: line3_index=2, the third axis is A
posi_mode	u8	0: relative 1: absolute
VL	i32	 VH,VL:pps, start speed (0 ≦ VL ≦ 6553500) Tacc,Tdec: seconds.
VH	i32	
Tacc	f64	
Tdec	f64	

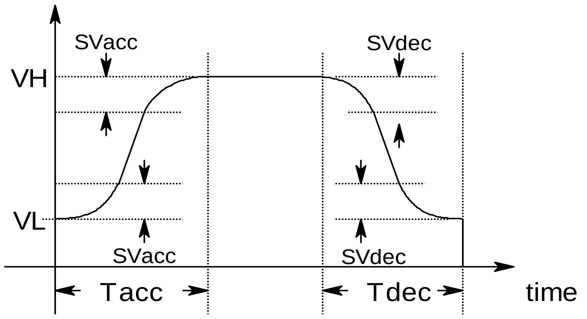
- **MPC3035_S_curve_move_LINE3**

Format : **u32 status = MPC3035_S_curve_move_LINE3(u8 CardID,u8 line3_index, i32 Position1,i32 Position2,i32 Position3,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 DIP/ROTARY switch
line3_index	u8	0: X,Y, Z 1: X, Y, A 2: X, Z, A 3: Y, Z, A
Position1	i32	target position (absolute or relative) for the first axis (-134,217,728 ≦Position1≦134,217,727) for example: line2_index=2, the first axis is X
Position2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦Position2≦134,217,727) for example: line2_index=2, the second axis is Z
Position3	i32	target position (absolute or relative) for the third axis (-134,217,728 ≦Position3≦134,217,727) for example: line2_index=2, the third axis is A
posi_mode	u8	0: relative 1: absolute
VL	i32	 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 3 axes linear interpolation:

1. For linear interpolation motion control in continuous mode (MPC3035_set_continuous_flag ( ), conti_flag=1), be sure to check continuous buffer (MPC3035_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 (MPC3035_fix_speed_range( )) at the operation axes.
3. In non-continuous mode(MPC3035_set_continuous_flag( )，conti_flag=0), be sure to check (MPC3035_read_motion_status( ); check_factor=0，ret_flag =1) to confirm the motion axes are ready.
4. The remained axes maybe programmed as point to point mode.

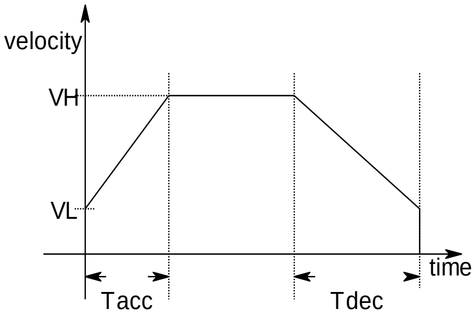
MPC3035_T_curve_move_LINE4

Format : u32 status = MPC3035_T_curve_move_LINE4(u8 CardID,i32 PositionX,
i32 PositionY,i32 PositionZ,i32 PositionA,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 DIP/ROTARY switch
PositionX	i32	target position (absolute or relative) for X axis (-134,217,728 ≦PositionX≦134,217,727)
PositionY	i32	target position (absolute or relative) for Y axis (-134,217,728 ≦PositionY≦134,217,727)
PositionZ	i32	target position (absolute or relative) for Z axis (-134,217,728 ≦PositionZ≦134,217,727)
PositionA	i32	target position (absolute or relative) for A axis (-134,217,728 ≦PositionA≦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	

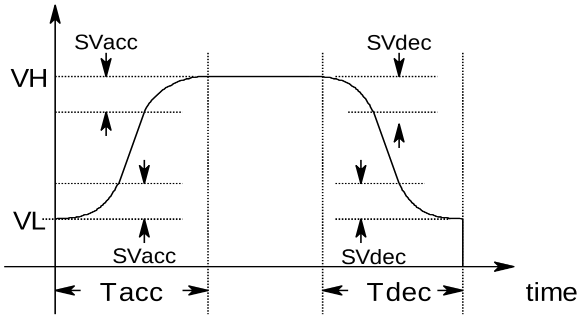
MPC3035_S_curve_move_LINE4

Format : **u32 status = MPC3035_S_curve_move_LINE4(u8 CardID,i32 PositionX, i32 PositionY,i32 PositionZ,i32 PositionA,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 DIP/ROTARY switch
PositionX	i32	target position (absolute or relative) for X axis (-134,217,728 ≦PositionX≦134,217,727)
PositionY	i32	target position (absolute or relative) for Y axis (-134,217,728 ≦PositionY≦134,217,727)
PositionZ	i32	target position (absolute or relative) for Z axis (-134,217,728 ≦PositionZ≦134,217,727)
PositionA	i32	target position (absolute or relative) for A axis (-134,217,728 ≦PositionA≦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 4 axes linear interpolation:

1. For linear interpolation motion control in continuous mode (MPC3035_set_continuous_flag ( ), conti_flag=1), be sure to check continuous buffer (MPC3035_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 (MPC3035_fix_speed_range( )) at the operation axes.
3. In non-continuous mode(MPC3035_set_continuous_flag( )，conti_flag=0), be sure to check (MPC3035_read_motion_status( ); check_factor=0，ret_flag =1) to confirm the motion axes are ready.

- **MPC3035_T_LINE_move**

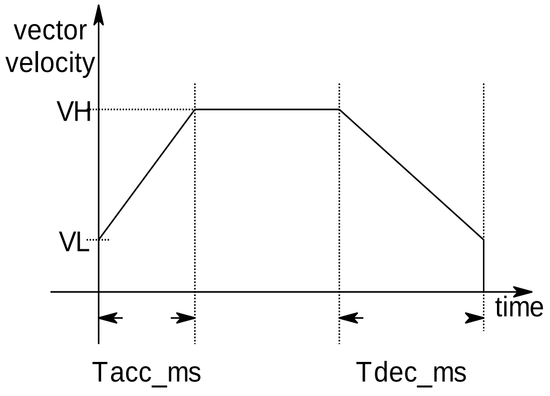
Format : **u32 status = MPC3035_T_LINE_move (u8 CardID,**
_Tline_CMD_Type *pTLine_command);

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

pTLine_command	_Tline_CMD_Type	A structure pointer of motion control parameters
		struct _Tline_CMD_Type{
		u8 posi_mode;
		//posi_mode: 0: relative, 1:absolute
		i32 VL;
		i32 VH;
		i32 Tacc_ms;
		i32 Tdec_ms;
		u8 Axis;
		//bit0:X axis bit 1:Y axis
		//bit 2:Z axis bit 3:A axis
		i32 Position[8];
		// position (absolute or relative) range
		// -134,217,728 ≦Position≦
		// 134,217,727
		Position[0]:positon of X Axis
		Position[1]:positon of Y Axis
		Position[2]:positon of Z Axis
		Position[3]:positon of A Axis
		} Tline_CMD_Type;
		
		VH,VL:pps, start speed (0 ≦ VL ≦ 6553500)
		Tacc_ms, Tdec_ms: in mini-seconds.

- ## MPC3035_S_LINE_move

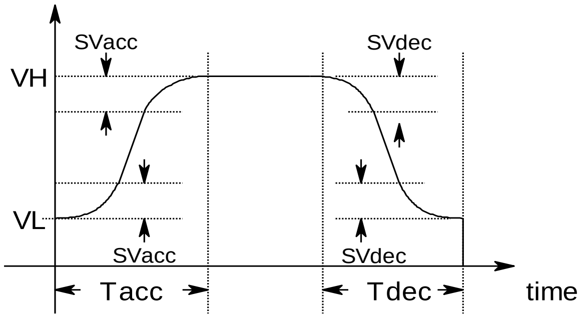
Format : u32 status = MPC3035_S_LINE_move (u8 CardID,
_Sline_CMD_Type *pSLine_command);

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

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

Circular interpolation motion control

- **MPC3035_ARC2_center_move**

Format : **u32 status = MPC3035_ARC2_center_move(u8 CardID,u8 arc2_index,**
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 for arc trajectrory.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X, Y 1: X, Z 2: X, A 3: Y, Z 4: Y, A 5: Z, A
center1	i32	circle center position (absolute or relative) for the first axis (-134,217,728 ≦center1≦134,217,727) for example: line2_index=2, the first axis is X
center2	i32	circle center position (absolute or relative) for the second axis (-134,217,728 ≦center2≦ 134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦endp1≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦endp2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VH	i32	vector velocity of circular interpolation
direction	u8	0: CW direction 1: CCW direction

- **MPC3035_T_ARC2_center_move**

Format : **u32 status = MPC3035_T_ARC2_center_move(u8 CardID,u8 arc2_index, 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 acceleration/deceleration is T profile for arc trajectrory.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X, Y 1: X, Z 2: X, A 3: Y, Z 4: Y, A 5: Z, A
center1	i32	circle center position (absolute or relative) for the first axis (-134,217,728 ≦center1≦134,217,727) for example: line2_index=2, the first axis is X
center2	i32	circle center position (absolute or relative) for the second axis (-134,217,728 ≦center2≦ 134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦endp1≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦endp2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VL	i32	Low vector speed of T profile of circular interpolation
VH	i32	High vector speed of T profile of circular interpolation
Tacc_dec	f64	Acc/Dec time of T profile of circular interpolation
direction	u8	0: CW direction 1: CCW direction

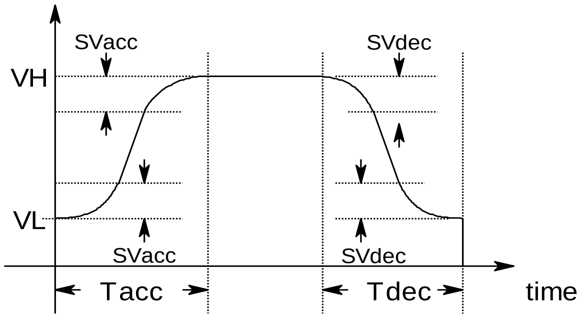
- **MPC3035_S_ARC2_center_move**

Format : **u32 status = MPC3035_S_ARC2_center_move(u8 CardID,u8 arc2_index, i32 center1,i32 center2,i32 endp1,i32 endp2,u8 posi_mode, i32 VL, i32 VH, f64 Tacc_dec, u32 SVacc_dec,u8 direction);**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X, Y 1: X, Z 2: X, A 3: Y, Z 4: Y, A 5: Z, A
center1	i32	circle center position (absolute or relative) for the first axis (-134,217,728 ≦center1≦134,217,727) for example: line2_index=2, the first axis is X
center2	i32	circle center position (absolute or relative) for the second axis (-134,217,728 ≦center2≦ 134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦endp1≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦endp2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VL	i32	 * Tacc = Tdec VH,VL are the vector velocity
VH	i32	
Tacc_dec	f64	
SVacc_dec	u32	
direction	u8	0: CW direction 1: CCW direction

- **MPC3035_ARC2_3P_move**

Format : **u32 status = MPC3035_ARC2_3P_move(u8 CardID,u8 arc2_index,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 as the arc trajectory.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
middle1	i32	middle position (absolute or relative) for the first axis (-134,217,728 ≦middle1≦134,217,727) for example: line2_index=2, the first axis is X
middle2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦middle2≦134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦endp1≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦endp2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VH	i32	vector velocity of circular interpolation

Note on circular interpolation:

1. For circular interpolation motion control in continuous mode (MPC3035_set_continuous_flag ( ), conti_flag=1), be sure to check continuous buffer (MPC3035_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 (MPC3035_fix_speed_range( )) at the operation axes.
3. In non-continuous mode(MPC3035_set_continuous_flag( )，conti_flag=0), be sure to check (MPC3035_read_motion_status( ); check_factor=0，ret_flag =1) to confirm the motion axes are ready.
4. While any 2 axes are working in circular interpolation mode, the others can not work in circular interpolation too, but point to point or linear interpolation is permitted.
5. The function MPC3035_ARC2_3P_move( ) does not need to define the motion direction, since the trajectory point has hidden definition.

- **MPC3035_T_ARC2_3P_move**

Format : **u32 status = MPC3035_T_ARC2_3P_move(u8 CardID,u8 arc2_index,**
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 acceleration/deceleration is T profile for arc trajectrory.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
middle1	i32	middle position (absolute or relative) for the first axis (-134,217,728 ≦middle1≦134,217,727) for example: line2_index=2, the first axis is X
middle2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦middle2≦134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦endp1≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦endp2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VL	i32	Low vectror speed of T profile of circular interpolation
VH	i32	High vectror speed of T profile of circular interpolation
Tacc_dec	f64	Acc/Dec time of T profile of circular interpolation

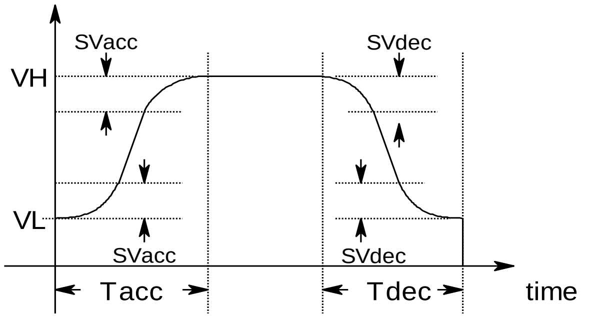
- **MPC3035_S_ARC2_3P_move**

Format : **u32 status = MPC3035_S_ARC2_3P_move(u8 CardID,u8 arc2_index,**
i32 middle1,i32 middle2,i32 endp1,i32 endp2,u8 posi_mode,
i32 VL,i32 VH, f64 Tacc_dec,u32 SVacc_dec);

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
middle1	i32	middle position (absolute or relative) for the first axis (-134,217,728 ≦middle1≦134,217,727) for example: line2_index=2, the first axis is X
middle2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦middle2≦134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦endp1≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦endp2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VL	i32	 SVacc SVdec VH VL SVacc SVdec Tacc Tdec time ** Tacc = Tdec VH,VL are the vector velocity
VH	i32	
Tacc_dec	f64	
SVacc_dec	u32	

- **MPC3035_ARC2_Radius_move**

Format : **u32 status = MPC3035_ARC2_Radius_move(u8 CardID, u8 arc2_index, i32 radius,i32 endp1,i32 endp2,u8 posi_mode,i32 VH,u8 direction)**

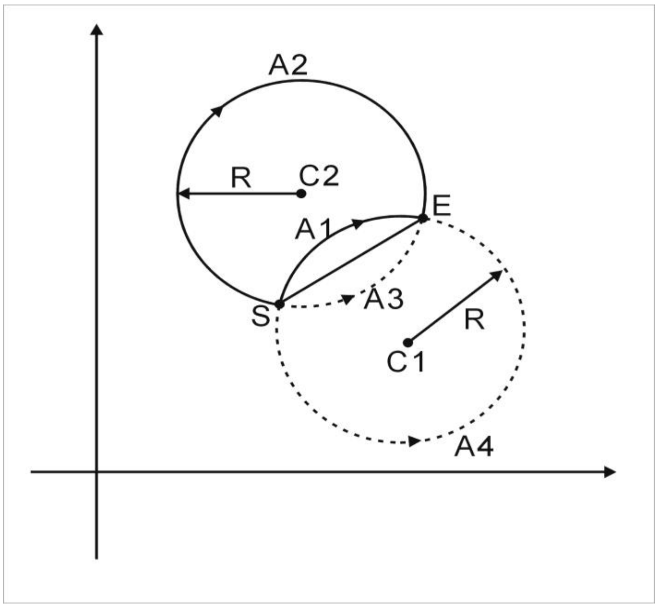
Purpose: To take the current position and end position to make a arc at designated R.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X, Y 1: X, Z 2: X, A 3: Y, Z 4: Y, A 5: Z, A
radius	i32	radius for the circle to pass currenet position and endpoint
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦endp1≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦endp2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	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.

- **MPC3035_T_ARC2_Radius_move**

Format : **u32 status = MPC3035_T_ARC2_Radius_move(u8 CardID, u8 arc2_index, i32 radius,i32 endp1,i32 endp2,u8 posi_mode,i32 VL,i32 VH, f64 Tacc_dec, u8 direction)**

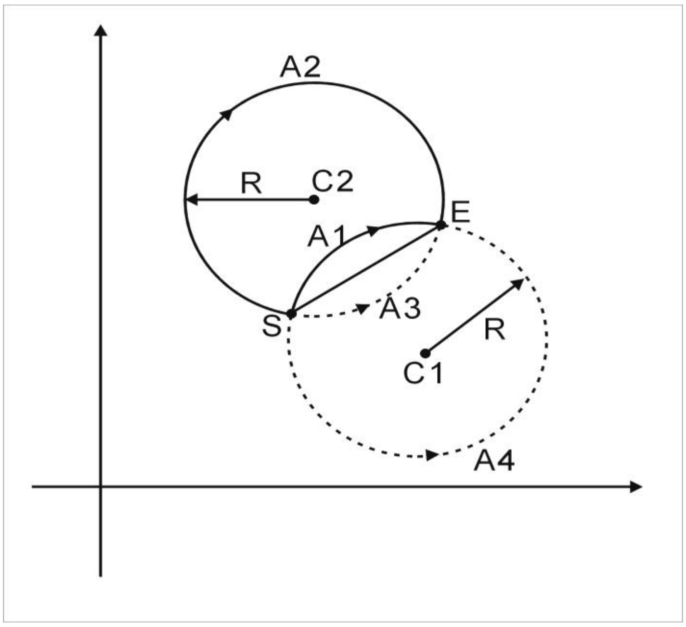
Purpose: To take the current position and end position to make a arc at designated R**.**

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY switch	
arc2_index	u8	0: X, Y 2: X, A 4: Y, A	1: X, Z 3: Y, Z 5: Z, A
radius	i32	radius for the circle to pass currenet position and endpoint	
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦endp1≦134,217,727) for example: line2_index=2, the first axis is X	
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦endp2≦134,217,727) for example: line2_index=2, the second axis is A	
posi_mode	u8	0: relative	1: absolute
VL	i32	Low vectror speed of T profile of circular interpolation	
VH	i32	High vectror speed of T profile of circular interpolation	
Tacc_dec	f64	Acc/Dec time of T profile 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.

- **MPC3035_CIR2_3P_move**

Format : **u32 status = MPC3035_CIR2_3P_move(u8 CardID,u8 arc2_index,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 DIP/ROTARY switch
arc2_index	u8	0: X, Y 1: X, Z 2: X, A 3: Y, Z 4: Y, A 5: Z, A
middle1	i32	middle position (absolute or relative) for the first axis (-134,217,728 ≦middle1≦134,217,727) for example: line2_index=2, the first axis is X
middle2	i32	middle position (absolute or relative) for the second axis (-134,217,728 ≦middle2≦134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦endp1≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦endp2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VH	i32	vector velocity of circular interpolation

- **MPC3035_T_CIR2_3P_move**

Format : **u32 status = MPC3035_T_CIR2_3P_move(u8 CardID,u8 arc2_index,**
i32 middle1,i32 middle2,i32 endp1,i32 endp2,u8 posi_mode,i32 VL,
i32 VH, f64 Tacc_dec);

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X, Y 1: X, Z 2: X, A 3: Y, Z 4: Y, A 5: Z, A
middle1	i32	middle position (absolute or relative) for the first axis (-134,217,728 ≦middle1≦134,217,727) for example: line2_index=2, the first axis is X
middle2	i32	middle position (absolute or relative) for the second axis (-134,217,728 ≦middle2≦134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦endp1≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦endp2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VL	i32	Low vectror speed of T profile of circular interpolation
VH	i32	High vectror speed of T profile of circular interpolation
Tacc_dec	f64	Acc/Dec time of T profile of circular interpolation

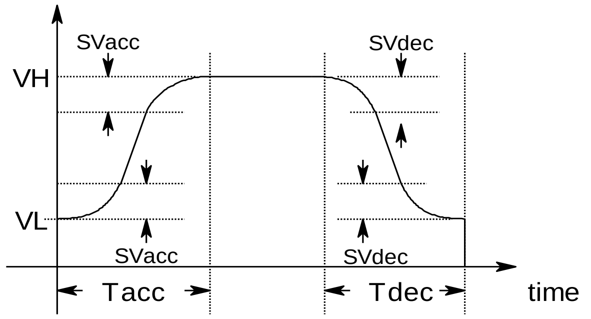
MPC3035_S_CIR2_3P_move

Format : **u32 status = MPC3035_T_CIR2_3P_move(u8 CardID,u8 arc2_index, i32 middle1,i32 middle2,i32 endp1,i32 endp2,u8 posi_mode,i32 VL, i32 VH, f64 Tacc_dec, u32 SVacc_dec);**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X, Y 1: X, Z 2: X, A 3: Y, Z 4: Y, A 5: Z, A
middle1	i32	middle position (absolute or relative) for the first axis (-134,217,728 ≦middle1≦134,217,727) for example: line2_index=2, the first axis is X
middle2	i32	middle position (absolute or relative) for the second axis (-134,217,728 ≦middle2≦134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦endp1≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦endp2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VL	i32	 ** Tacc = Tdec VH,VL are the vector velocity
VH	i32	
Tacc_dec	f64	
SVacc_dec	u32	

- **MPC3035_CIR2_Radius_move**

Format : **u32 status = MPC3035_CIR2_Radius_move(u8 CardID,u8 arc2_index, i32 radius,i32 endp1,i32 endp2,u8 posi_mode,i32 VH,u8 direction);**

Purpose: To take the current position end position and radius to make a circle.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X, Y 1: X, Z 2: X, A 3: Y, Z 4: Y, A 5: Z, A
radius	i32	Radius of the target circle
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦endp1≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦endp2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VH	i32	vectror speed of circular interpolation
direction	u8	0: CW 1: CCW

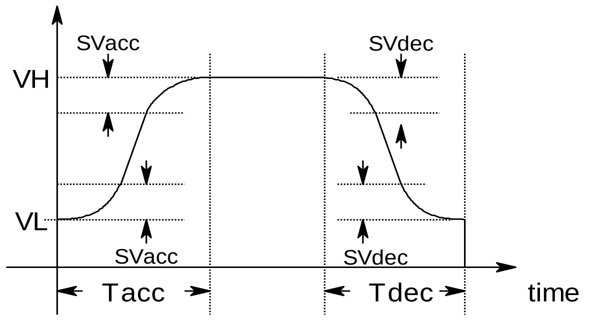
MPC3035_T_CIR2_Radius_move

Format : **u32 status = MPC3035_T_CIR2_Radius_move(u8 CardID,u8 arc2_index, 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 the middle, end position to make a circle and the circular interpolation pass through the 3 positions.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X, Y 1: X, Z 2: X, A 3: Y, Z 4: Y, A 5: Z, A
radius	i32	Radius of the target circle
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦endp1≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦endp2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VL	i32	 SVacc SVdec VH VL SVacc SVdec Tacc Tdec time ** Tacc = Tdec VH,VL are the vector velocity
VH	i32	
Tacc_dec	f64	
direction	u8	0: CW 1: CCW

Spiral motion

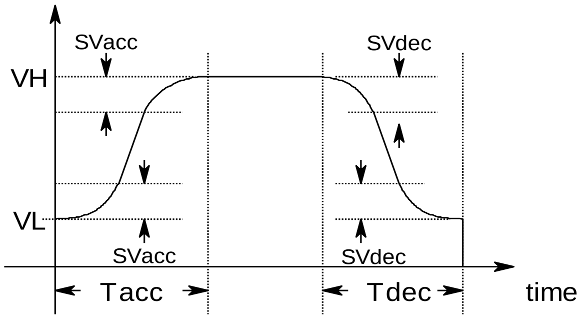
- **MPC3035_ArcXY_LineZ_center_move**

Format : **u32 status = MPC3035_ArcXY_LineZ_center_move(u8 CardID,i32 centerX, i32 centerY, i32 endpX,i32 endpY,i32 endpZ,u8 posi_mode,i32 VL, i32 VH,f64 Tacc_dec,u8 direction);**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
centerX	i32	X axis center of arc
centerY	i32	Y axis center of arc
endpX	i32	end position (absolute or relative) for the X axis (-134,217,728 ≦endpX≦134,217,727)
endpY	i32	end position (absolute or relative) for the Y axis (-134,217,728 ≦endpY≦134,217,727)
endpZ	i32	end position (absolute or relative) for the Z axis (-134,217,728 ≦endpZ≦134,217,727) Linear interpolation will go from current Z position to endZ position
posi_mode	u8	0: relative 1: absolute
VL	i32	 SVacc SVdec VH VL SVacc SVdec Tacc Tdec time ** Tacc = Tdec VL: Low speed of T profile of circular interpolation VH: High speed of T profile of circular interpolation Tacc_dec: Acc/Dec time of T profile of circular interpolation
VH	i32	
Tacc_dec	f64	
direction	u8	0: CW 1: CCW

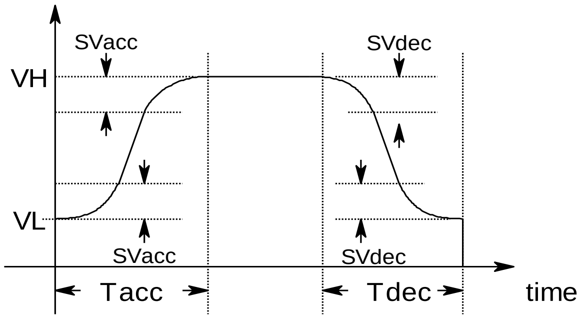
- **MPC3035_ArcXY_LineZ_3P_move**

Format : **u32 status =MPC3035_ArcXY_LineZ_3P_move(u8 CardID,i32 middleX, i32 middleY,i32 endpX,i32 endpY,i32 endpZ,u8 posi_mode,i32 VL, i32 VH,f64 Tacc_dec);**

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

Parameters:

Input:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>CardID</td><td>u8</td><td>assigned by DIP/ROTARY switch</td></tr>
<tr><td>middleX</td><td>i32</td><td>X axis middle point that arc will pass</td></tr>
<tr><td>middleY</td><td>i32</td><td>Y axis middle point that arc will pass</td></tr>
<tr><td>endpX</td><td>i32</td><td>end position (absolute or relative) for the X axis
(-134,217,728 ≦endpX≦134,217,727)</td></tr>
<tr><td>endpY</td><td>i32</td><td>end position (absolute or relative) for the Y axis
(-134,217,728 ≦endpY≦134,217,727)</td></tr>
<tr><td>endpZ</td><td>i32</td><td>end position (absolute or relative) for the Z axis
(-134,217,728 ≦endpZ≦134,217,727)
Linear interpolation will go from current Z position to endZ position</td></tr>
<tr><td>posi_mode</td><td>u8</td><td>0: relative 1: absolute</td></tr>
<tr><td>VL</td><td>i32</td><td rowspan="3">
** Tacc = Tdec
VL: Low speed of T profile of circular interpolation
VH: High speed of T profile of circular interpolation
Tacc_dec: Acc/Dec time of T profile of circular interpolation</td></tr>
<tr><td>VH</td><td>i32</td></tr>
<tr><td>Tacc_dec</td><td>f64</td></tr>
</table>

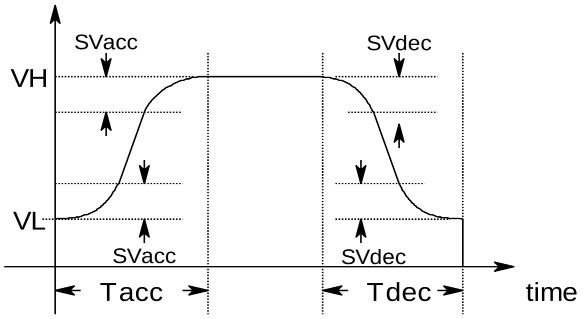
MPC3035_CirXY_LineZ_3P_move

Format : **u32 status = MPC3035_CirXY_LineZ_3P_move(u8 CardID,i32 middleX, i32 middleY,i32 endpX,i32 endpY,i32 endpZ,u8 posi_mode,i32 VL, i32 VH,f64 Tacc_dec);**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
middleX	i32	X axis middle point that arc will pass
middleY	i32	Y axis middle point that arc will pass
endpX	i32	end position (absolute or relative) for the X axis (-134,217,728 ≦endpX≦134,217,727)
endpY	i32	end position (absolute or relative) for the Y axis (-134,217,728 ≦endpY≦134,217,727)
endpZ	i32	end position (absolute or relative) for the Z axis (-134,217,728 ≦endpZ≦134,217,727) Linear interpolation will go from current Z position to endZ position
posi_mode	u8	0: relative 1: absolute
VL	i32	 SVacc SVdec VH VL SVacc SVdec Tacc Tdec time ** Tacc = Tdec VL: Low speed of T profile of circular interpolation VH: High speed of T profile of circular interpolation Tacc_dec: Acc/Dec time of T profile of circular interpolation
VH	i32	
Tacc_dec	f64	

Motion with FIFO

Note:

The FIFO function can only be used in Windows 2000/XP P3 800MHz and grade-up system. Refer the control flow chart of FIFO at section 8.4 for your application.

- ### MPC3035_enable_FIFO

Format : **u32 status = MPC3035_enable_FIFO(u8 CardID,u8 enable,u8 dimension, u32 alarm_count);**

Purpose: To enable the FIFO function.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY switch	
enable	u8	0: disable	1: enable
dimension	u8	0: null 1: one dimension FIFO (single axis) 2: 2 dimension (dual axis) 3: 3 dimension (triple axes) 4: 4 dimension (4 axes)	
alarm_count	u32	The remained data count of FIFO, while the count reached an interrupt will generate. 10 ≦ alarm_count ≦ 1948	

Note: The FIFO size is 2047 and the recommended alarm count is 200.

- ### MPC3035_check_FIFO_buffer

Format : **u32 status = MPC3035_check_FIFO_buffer(u8 CardID,u8 *buffer_full_flag, u16 *remain_no);**

Purpose: To check remained FIFO data.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

Output:

Name	Type	Description
buffer_full_flag	u8	0: buffer not full 1: buffer full
remain_no	u16	The remained data count of FIFO

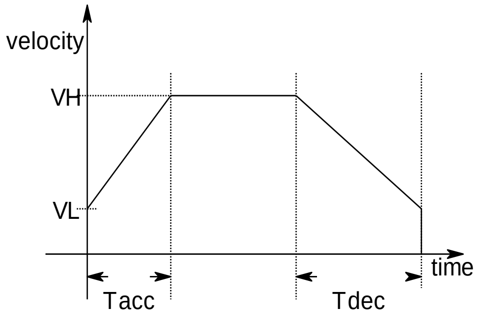
- **MPC3035_T_curve_write_FIFO**

Format : **u32 status = MPC3035_T_curve_write_FIFO(u8 CardID,u8 Axis,i32 Position,
u8 posi_mode,i32 VL,i32 VH,f64 Tacc,f64 Tdec);**

Purpose: To fill FIFO with one dimension’s datum.

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 to be buffered (-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	

Note:

1. If your application will override the speed ratio (use MPC3035_set_FIFO_out_Ratio( ) to override speed) of the motion data in FIFO, parameters Tacc and Tdec should be set to “0”.
2. The motion time per FIFO data should be greater than 0.5ms for properly operation of FIFO function, i.e. Position/VH ≧ 0.5ms.

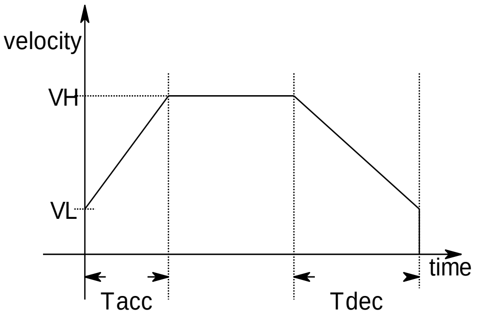
MPC3035_T_LINE2_write_FIFO

Format : **u32 status = MPC3035_T_LINE2_write_FIFO(u8 CardID,u8 line2_index, i32 Position1,i32 Position2,u8 posi_mode,i32 VL,i32 VH,f64 Tacc, f64 Tdec);**

Purpose: To fill FIFO with 2 dimensions' datum.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
line2_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
Position1	i32	new target position for first axis (-134,217,728 ≦ Position1 ≦ 134,217,727)
Position2	i32	new target position for second 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	

Note:

1. If your application will override the speed ratio (use MPC3035_set_FIFO_out_Ratio( ) to override speed) of the motion data in FIFO, parameters Tacc and Tdec should be set to "0".
2. The motion time per FIFO data should be greater than 0.5ms for properly operation of FIFO function, i.e. sqrt(Position1**2+Position2**2)/VH ≧ 0.5ms.

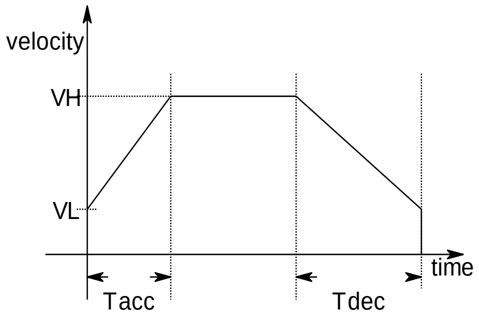
MPC3035_T_LINE3_write_FIFO

Format : **u32 status = MPC3035_T_LINE3_write_FIFO(u8 CardID,u8 line3_index, i32 Position1,i32 Position2,i32 Position3,u8 posi_mode,i32 VL, i32 VH,f64 Tacc, f64 Tdec);**

Purpose: To fill FIFO with 3 dimensions' datum.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
line3_index	u8	0: X,Y, Z 1: X, Y, A 2: X, Z, A 3: Y, Z, A
Position1	i32	new target position for first axis (-134,217,728 ≦ Position1 ≦ 134,217,727)
Position2	i32	new target position for second axis (-134,217,728 ≦ Position2 ≦ 134,217,727)
Position3	i32	new target position for third axis (-134,217,728 ≦ Position3 ≦ 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	

Note:

1. If your application will override the speed ratio (use MPC3035_set_FIFO_out_Ratio( ) to override speed) of the motion data in FIFO, parameters Tacc and Tdec should be set to "0".
2. The motion time per FIFO data should be greater than 0.5ms for properly operation of FIFO function, i.e. sqrt(Position1**2+Position2**2+ Position3**2)/VH ≧ 0.5ms.

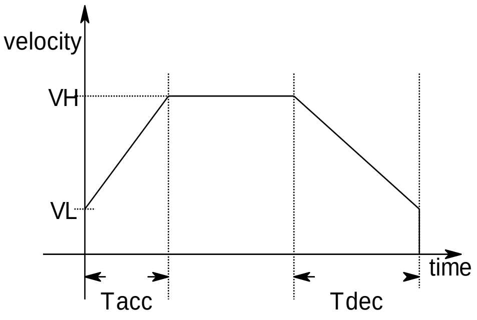
MPC3035_T_LINE4_write_FIFO

Format : **u32 status = MPC3035_T_LINE4_write_FIFO(u8 CardID,i32 PositionX, i32 PositionY,i32 PositionZ,i32 PositionA,u8 posi_mode,i32 VL, i32 VH,f64 Tacc, f64 Tdec);**

Purpose: To fill FIFO with 4 dimensions' datum.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
PositionX	i32	new target position for X axis (-134,217,728 ≦ PositionX ≦ 134,217,727)
PositionY	i32	new target position for Y axis (-134,217,728 ≦ PositionY ≦ 134,217,727)
PositionZ	i32	new target position for Z axis (-134,217,728 ≦ PositionZ ≦ 134,217,727)
PositionA	i32	new target position for A axis (-134,217,728 ≦ PositionA ≦ 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	

Note:

1. If your application will override the speed ratio (use MPC3035_set_FIFO_out_Ratio( ) to override speed) of the motion data in FIFO, parameters Tacc and Tdec should be set to "0".
2. The motion time per FIFO data should be greater than 0.5ms for properly operation of FIFO function, i.e. sqrt(PositionX**2+PositionY**2+ PositionZ**2+ PositionA**2)/VH ≧ 0.5ms.

- **MPC3035_Run_FIFO_CMD**

Format : **u32 status = MPC3035_Run_FIFO_CMD(u8 CardID);**

Purpose: Start to run the motion stored in FIFO.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

- **MPC3035_set_FIFO_out_Ratio**

Format : **u32 status = MPC3035_set_FIFO_out_Ratio(u8 CardID,u8 Percent_Ratio);**

Purpose: To override the speed of stored FIFO data.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
Percent_Ratio	u8	1 ≦ Percent_Ratio ≦ 200

Note:

1. To use speed override (MPC3035_set_FIFO_out_Ratio( )) must have Tacc, Tdec set to “0” at the previous FIFO data.
2. Although MPC3035_set_FIFO_out_Ratio( ) may override the speed of stored FIFO data, but the speed is limited by MPC3035_fix_speed_range( ), to preset an adequate maximum speed (using MPC3035_fix_speed_range( )) is recommended for speed override function.

- **MPC3035_FIFO_EOI**

Format : **u32 status = MPC3035_FIFO_EOI(u8 CardID)**

Purpose: End of interrupt of FIFO, to reset the FIFO interrupt function.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

Synchronized start motion

- **MPC3035_config_compare_start_motion**

Format : **u32 status = MPC3035_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 2: Z 3: A
cmp_source	u8	0: to compare with the current position command counter 1: to compare with the feedback counter 2: undefined 3: to compare with the pulser counter
cmp_method	u8	1: compare out at equal, and does not care direction 2: compare out at equal while counting up 3: compare out at equal while counting down 4: compare out at preset value > counter value 5: compare out at preset value < counter value

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

- **MPC3035_set_compare_start_data**

Format : **u32 status = MPC3035_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

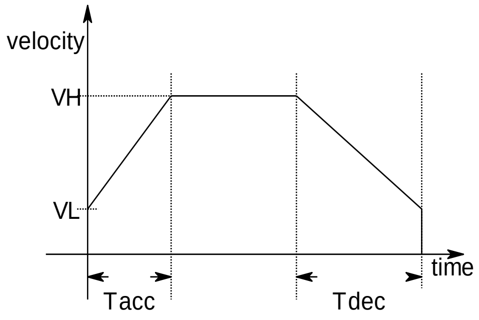
MPC3035_T_curve_wait_Cmpstart

Format : **u32 status = MPC3035_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	

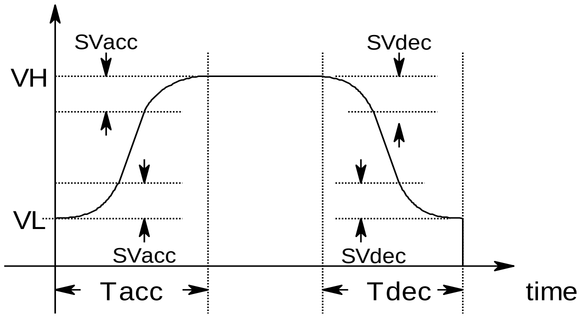
MPC3035_S_curve_wait_Cmpstart

Format : **u32 status = MPC3035_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	

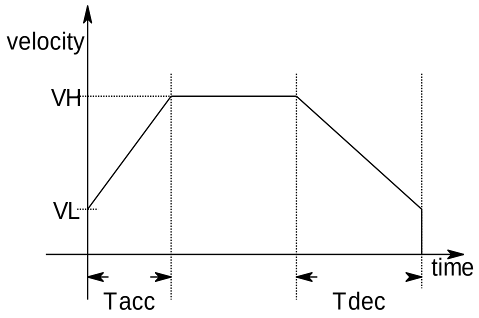
MPC3035_T_LINE2_wait_Cmpstart

Format : **u32 status = MPC3035_T_LINE2_wait_Cmpstart(u8 CardID,u8 line2_index,**
i32 Position1,i32 Position2,u8 posi_mode,i32 VL,i32 VH,f64 Tacc,
f64 Tdec);

Purpose: To setup the T profile 2 axes linear motion and wait for synchronous start signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
line2_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
Position1	i32	target position (absolute or relative) for the first axis (-134,217,728 ≦Position1≦134,217,727) for example: line2_index=2, the first axis is X
Position2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦Position2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VL	i32	 VH,VL:pps, start speed (0 ≦ VL ≦ 6553500) Tacc,Tdec: seconds.
VH	i32	
Tacc	f64	
Tdec	f64	

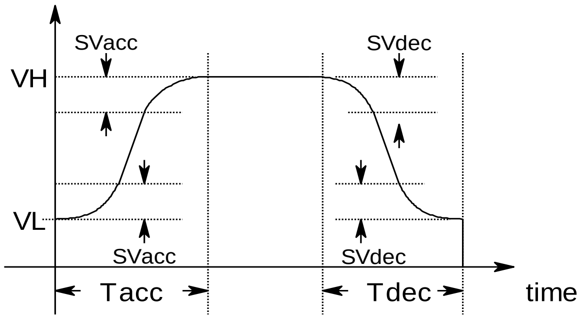
- **MPC3035_S_LINE2_wait_Cmpstart**

Format : **u32 status = MPC3035_S_LINE2_wait_Cmpstart(u8 CardID,u8 line2_index, i32 Position1,i32 Position2,u8 posi_mode,i32 VL,i32 VH,f64 Tacc, f64 Tdec, u32 SVacc,u32 SVdec);**

Purpose: To setup the S profile 2 axes linear motion and wait for synchronous start signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
line2_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
Position1	i32	target position (absolute or relative) for the first axis (-134,217,728 ≦Position1≦134,217,727) for example: line2_index=2, the first axis is X
Position2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦Position2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VL	i32	 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	

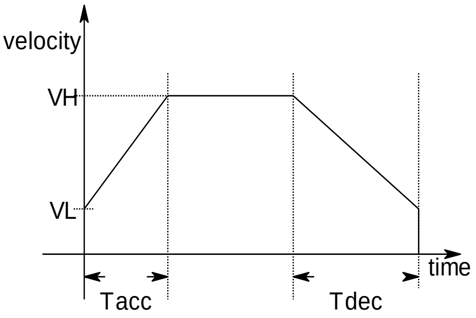
- **MPC3035_T_LINE3_wait_Cmpstart**

Format : **u32 status = MPC3035_T_LINE3_wait_Cmpstart(u8 CardID,u8 line3_index, i32 Position1,i32 Position2,i32 Position3,u8 posi_mode,i32 VL, i32 VH, f64 Tacc,f64 Tdec);**

Purpose: To setup the T profile 3 axes linear motion and wait for synchronous start signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
line3_index	u8	0: X,Y, Z 1: X, Y, A 2: X, Z, A 3: Y, Z, A
Position1	i32	target position (absolute or relative) for the first axis (-134,217,728 ≦Position1≦134,217,727) for example: line3_index=2, the first axis is X
Position2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦Position2≦134,217,727) for example: line3_index=2, the second axis is Z
Position3	i32	target position (absolute or relative) for the third axis (-134,217,728 ≦Position3≦134,217,727) for example: line3_index=2, the third axis is A
posi_mode	u8	0: relative 1: absolute
VL	i32	 VH,VL:pps, start speed (0 ≦ VL ≦ 6553500) Tacc,Tdec: seconds.
VH	i32	
Tacc	f64	
Tdec	f64	

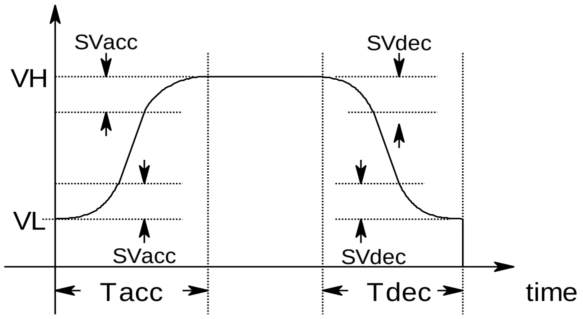
MPC3035_S_LINE3_wait_Cmpstart

Format : **u32 status = MPC3035_S_LINE3_wait_Cmpstart(u8 CardID,u8 line2_index,**
i32 Position1,i32 Position2, i32 Position3,u8 posi_mode,i32 VL,
i32 VH, f64 Tacc,f64 Tdec, u32 SVacc,u32 SVdec);

Purpose: To setup the S profile 3 axes linear motion and wait for synchronous start signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
line3_index	u8	0: X,Y, Z 1: X, Y, A 2: X, Z, A 3: Y, Z, A
Position1	i32	target position (absolute or relative) for the first axis (-134,217,728 ≦Position1≦134,217,727) for example: line2_index=2, the first axis is X
Position2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦Position2≦134,217,727) for example: line2_index=2, the second axis is Z
Position3	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦Position3≦134,217,727) for example: line2_index=2, the second axis is A
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	

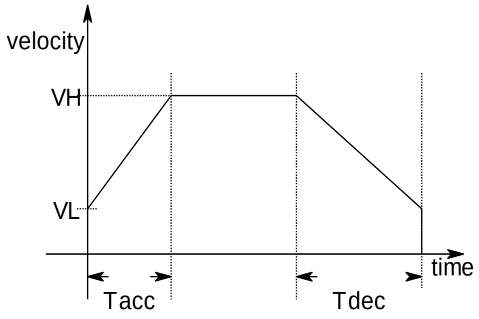
● MPC3035_T_LINE4_wait_Cmpstart

Format : **u32 status = MPC3035_T_LINE4_wait_Cmpstart(u8 CardID,i32 PositionX,**
i32 PositionY,i32 PositionZ,i32 PositionA,u8 posi_mode,i32 VL,
i32 VH, f64 Tacc,f64 Tdec);

Purpose: To setup the T profile 4 axes linear motion and wait for synchronous start signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
PositionX	i32	target position (absolute or relative) for the X axis (-134,217,728 ≦PositionX≦134,217,727)
PositionY	i32	target position (absolute or relative) for the Y axis (-134,217,728 ≦PositionY≦134,217,727)
PositionZ	i32	target position (absolute or relative) for the Z axis (-134,217,728 ≦PositionZ≦134,217,727)
PositionA	i32	target position (absolute or relative) for the A axis (-134,217,728 ≦PositionA≦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	

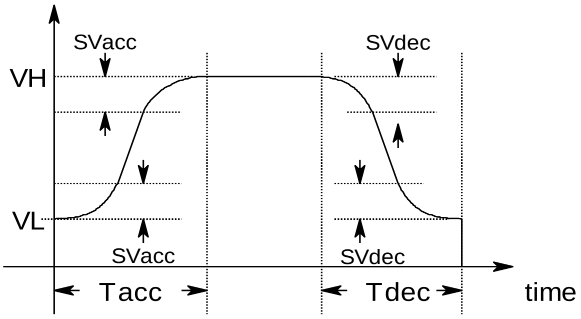
● MPC3035_S_LINE4_wait_Cmpstart

Format : **u32 status = MPC3035_S_LINE4_wait_Cmpstart(u8 CardID,i32 PositionX, i32 PositionY,i32 PositionZ,i32 PositionA,u8 posi_mode,i32 VL, i32 VH, f64 Tacc,f64 Tdec,u32 Svacc,u32 SVdec);**

Purpose: To setup the S profile 4 axes linear motion and wait for synchronous start signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
PositionX	i32	target position (absolute or relative) for the X axis (-134,217,728 ≦PositionX≦134,217,727)
PositionY	i32	target position (absolute or relative) for the Y axis (-134,217,728 ≦PositionY≦134,217,727)
PositionZ	i32	target position (absolute or relative) for the Z axis (-134,217,728 ≦PositionZ≦134,217,727)
PositionA	i32	target position (absolute or relative) for the A axis (-134,217,728 ≦PositionA≦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	

- **MPC3035_ARC2_center_wait_Cmpstart**

Format : **u32 status = MPC3035_ARC2_center_wait_Cmpstart(u8 CardID, u8 arc2_index,
i32 center1,i32 center2,i32 endp1,i32 endp2,u8 posi_mode,
i32 VH,u8 direction);**

Purpose: To setup circular interpolation movement with circle center and end position and wait for synchronous start signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X, Y　　1: X, Z 2: X, A　　3: Y, Z 4: Y, A　　5: Z, A
center1	i32	circle center position (absolute or relative) for the first axis (-134,217,728 ≦center1≦134,217,727) for example: line2_index=2, the first axis is X
center2	i32	circle center position (absolute or relative) for the second axis (-134,217,728 ≦center2≦ 134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦endp1≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦endp2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative　　1: absolute
VH	i32	vector velocity of circular interpolation
direction	u8	0: CW direction　　1: CCW direction

- **MPC3035_ARC2_3P_wait_Cmpstart**

Format : **u32 status = MPC3035_ARC2_3P_wait_Cmpstart(u8 CardID, u8 arc2_index, i32 middle1,i32 middle2,i32 endp1,i32 endp2,u8 posi_mode, i32 VH);**

Purpose: To setup circular interpolation movement with current point and the other 2 points as the circle trajectory, and wait for synchronous start signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
middle1	i32	middle position (absolute or relative) for the first axis (-134,217,728 ≦middle1≦134,217,727) for example: line2_index=2, the first axis is X
middle2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦middle2≦134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦endp1≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦endp2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VH	i32	vector velocity of circular interpolation

Note on circular interpolation:

1. Circular interpolation motion control in continuous mode (MPC3035_set_continuous_flag ( ), conti_flag=1), be sure to check continuous buffer (MPC3035_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 (MPC3035_fix_speed_range( )) at the operation axes.
3. In non-continuous mode(MPC3035_set_continuous_flag( )，conti_flag=0), be sure to check (MPC3035_read_motion_status( ); check_factor=0，ret_flag =1) to confirm the motion axes are ready.
4. In circular interpolation mode, no acceleration/deceleration is implemented.
5. While any 2 axes are working in circular interpolation mode, the others can not work in circular interpolation too, but point to point or linear interpolation is permitted.
6. The function MPC3035_ARC2_3P_move( ) does not need to define the motion direction, since the trajectory point has hidden definition.

- **MPC3035_read_compare_start_flag**

Format : **u32 status = MPC3035_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

External trigger start/stop function

- **MPC3035_trigger_CSTA_pulse**

Format : **u32 status = MPC3035_trigger_CSTA_pulse(u8 CardID);**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

- **MPC3035_trigger_CSTP_pulse**

Format : **u32 status = MPC3035_trigger_CSTP_pulse(u8 CardID);**

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

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

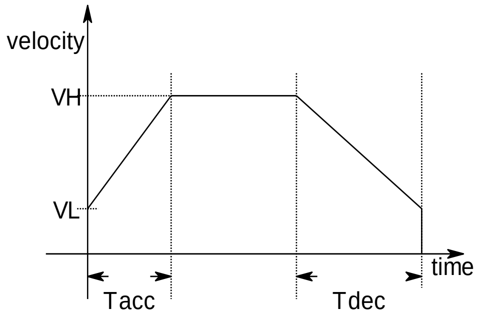
- **MPC3035_T_curve_wait_CSTA**

Format : **u32 status = MPC3035_T_curve_wait_CSTA(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 CSTA 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	

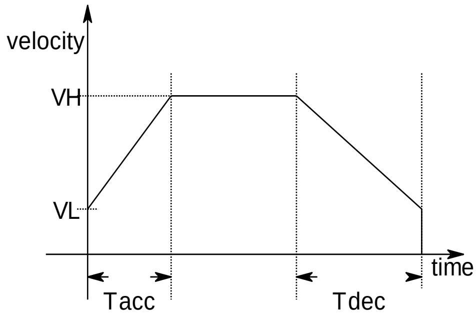
- **MPC3035_T_LINE2_wait_CSTA**

Format : **u32 status = MPC3035_T_LINE2_wait_CSTA(u8 CardID,u8 line2_index, i32 Position1,i32 Position2,u8 posi_mode,i32 VL,i32 VH,f64 Tacc, f64 Tdec);**

Purpose: To setup the T profile 2 axes linear motion and wait for CSTA signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
line2_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
Position1	i32	target position (absolute or relative) for the first axis (-134,217,728 ≦Position1≦134,217,727) for example: line2_index=2, the first axis is X
Position2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦Position2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VL	i32	 VH,VL:pps, start speed (0 ≦ VL ≦ 6553500) Tacc,Tdec: seconds.
VH	i32	
Tacc	f64	
Tdec	f64	

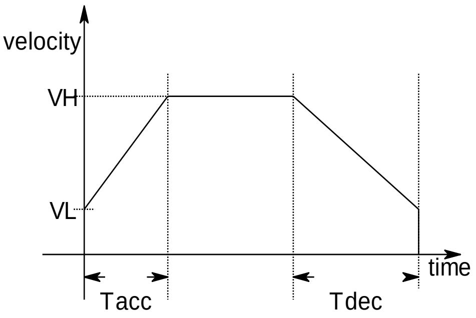
- **MPC3035_T_LINE3_wait_CSTA**

Format : **u32 status = MPC3035_T_LINE3_wait_CSTA(u8 CardID,u8 line3_index,**
i32 Position1,i32 Position2,i32 Position3,u8 posi_mode,i32 VL,
i32 VH, f64 Tacc,f64 Tdec);

Purpose: To setup the T profile 3 axes linear motion and wait for CSTA signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
Line3_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
Position1	i32	target position (absolute or relative) for the first axis (-134,217,728 ≦Position1≦134,217,727) for example: line3_index=2, the first axis is X
Position2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦Position2≦134,217,727) for example: line3_index=2, the second axis is Z
Position3	i32	target position (absolute or relative) for the third axis (-134,217,728 ≦Position3≦134,217,727) for example: line3_index=2, the third axis is A
posi_mode	u8	0: relative 1: absolute
VL	i32	 VH,VL:pps, start speed (0 ≦ VL ≦ 6553500) Tacc,Tdec: seconds.
VH	i32	
Tacc	f64	
Tdec	f64	

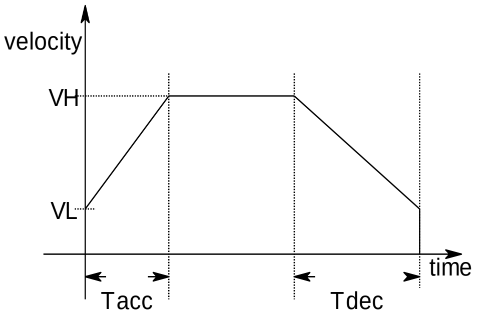
- **MPC3035_T_LINE4_wait_CSTA**

Format : **u32 status = MPC3035_T_LINE4_wait_Cmpstart(u8 CardID,i32 PositionX, i32 PositionY,i32 PositionZ,i32 PositionA,u8 posi_mode,i32 VL, i32 VH, f64 Tacc,f64 Tdec);**

Purpose: To setup the T profile 4 axes linear motion and wait for CSTA signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
PositionX	i32	target position (absolute or relative) for the X axis (-134,217,728 ≦PositionX≦134,217,727)
PositionY	i32	target position (absolute or relative) for the Y axis (-134,217,728 ≦PositionY≦134,217,727)
PositionZ	i32	target position (absolute or relative) for the Z axis (-134,217,728 ≦PositionZ≦134,217,727)
PositionA	i32	target position (absolute or relative) for the A axis (-134,217,728 ≦PositionA≦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	

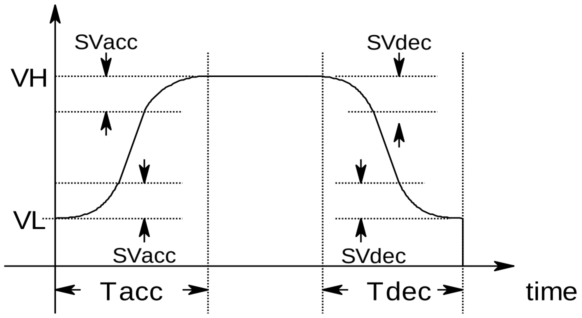
- **MPC3035_S_curve_wait_CSTA**

Format : **u32 status = MPC3035_S_curve_wait_CSTA(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 CSTA 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 : 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	

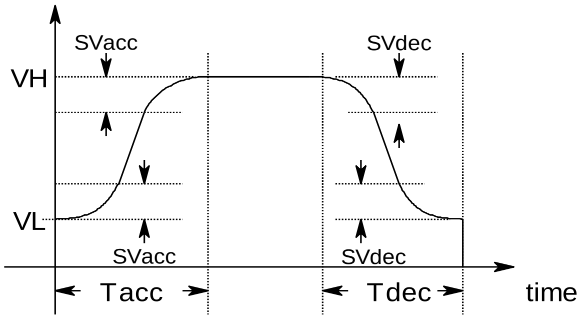
MPC3035_S_LINE2_wait_CSTA

Format : **u32 status = MPC3035_S_LINE2_wait_CSTA(u8 CardID,u8 line2_index,**
i32 Position1,i32 Position2,u8 posi_mode,i32 VL,i32 VH,f64 Tacc,
f64 Tdec, u32 SVacc,u32 SVdec);

Purpose: To setup the S profile 2 axes linear motion and wait for CSTA signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
line2_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
Position1	i32	target position (absolute or relative) for the first axis (-134,217,728 ≦Position1≦134,217,727) for example: line2_index=2, the first axis is X
Position2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦Position2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VL	i32	 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	

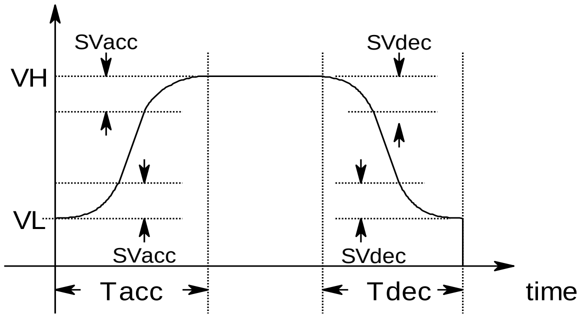
- **MPC3035_S_LINE3_wait_CSTA**

Format : **u32 status = MPC3035_S_LINE3_wait_CSTA(u8 CardID,u8 line3_index,**
i32 Position1,i32 Position2,i32 Position3,u8 posi_mode,i32 VL,
i32 VH, f64 Tacc,f64 Tdec,u32 SVacc,u32 SVdec);

Purpose: To setup the S profile 3 axes linear motion and wait for CSTA signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
line3_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
Position1	i32	target position (absolute or relative) for the first axis (-134,217,728 ≦Position1≦134,217,727) for example: line2_index=2, the first axis is X
Position2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦Position2≦134,217,727) for example: line2_index=2, the second axis is Z
Position3	i32	target position (absolute or relative) for the third axis (-134,217,728 ≦Position3≦134,217,727) for example: line2_index=2, the third axis is A
posi_mode	u8	0: relative 1: absolute
VL	i32	 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	

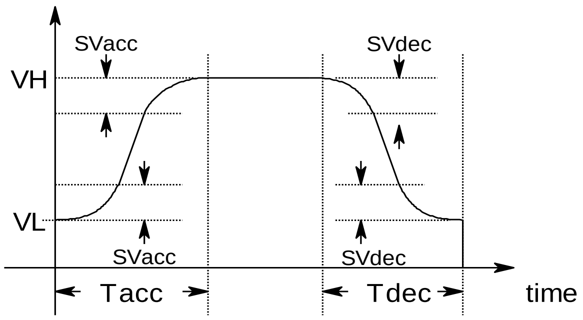
MPC3035_S_LINE4_wait_CSTA

Format : **u32 status = MPC3035_S_LINE4_wait_Cmpstart(u8 CardID,i32 PositionX,**
i32 PositionY,i32 PositionZ,i32 PositionA,u8 posi_mode,i32 VL,
i32 VH, f64 Tacc,f64 Tdec,u32 Svacc,u32 SVdec);

Purpose: To setup the S profile 4 axes linear motion and wait for CSTA signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
PositionX	i32	target position (absolute or relative) for the X axis (-134,217,728 ≦PositionX≦134,217,727)
PositionY	i32	target position (absolute or relative) for the Y axis (-134,217,728 ≦PositionY≦134,217,727)
PositionZ	i32	target position (absolute or relative) for the Z axis (-134,217,728 ≦PositionZ≦134,217,727)
PositionA	i32	target position (absolute or relative) for the A axis (-134,217,728 ≦PositionA≦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	

- **MPC3035_ARC2_center_wait_CSTA**

Format : **u32 status = MPC3035_ARC2_center_wait_CSTA(u8 CardID, u8 arc2_index,**
i32 center1,i32 center2,i32 endp1,i32 endp2,u8 posi_mode,
i32 VH,u8 direction);

Purpose: To setup circular interpolation movement with circle center and end position and wait for CSTA signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X, Y 1: X, Z 2: X, A 3: Y, Z 4: Y, A 5: Z, A
center1	i32	circle center position (absolute or relative) for the first axis (-134,217,728 ≦center1≦134,217,727) for example: line2_index=2, the first axis is X
center2	i32	circle center position (absolute or relative) for the second axis (-134,217,728 ≦center2≦ 134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦endp1≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦endp2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VH	i32	vector velocity of circular interpolation
direction	u8	0: CW direction 1: CCW direction

- **MPC3035_ARC2_3P_wait_CSTA**

Format : **u32 status = MPC3035_ARC2_3P_wait_CSTA(u8 CardID, u8 arc2_index, i32 middle1,i32 middle2,i32 endp1,i32 endp2,u8 posi_mode, i32 VH);**

Purpose: To setup circular interpolation movement with current point and the other 2 points as the circle trajectory, and wait for CSTA signal to take action.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
arc2_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
middle1	i32	middle position (absolute or relative) for the first axis (-134,217,728 ≦middle1≦134,217,727) for example: line2_index=2, the first axis is X
middle2	i32	target position (absolute or relative) for the second axis (-134,217,728 ≦middle2≦134,217,727) for example: line2_index=2, the second axis is A
endp1	i32	end position (absolute or relative) for the first axis (-134,217,728 ≦endp1≦134,217,727) for example: line2_index=2, the first axis is X
endp2	i32	end position (absolute or relative) for the second axis (-134,217,728 ≦endp2≦134,217,727) for example: line2_index=2, the second axis is A
posi_mode	u8	0: relative 1: absolute
VH	i32	vector velocity of circular interpolation

Note on circular interpolation:

1. Circular interpolation motion control in continuous mode (MPC3035_set_continuous_flag ( ), conti_flag=1), be sure to check continuous buffer (MPC3035_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 (MPC3035_fix_speed_range( )) at the operation axes.
3. In non-continuous mode(MPC3035_set_continuous_flag( )，conti_flag=0), be sure to check (MPC3035_read_motion_status( ); check_factor=0，ret_flag =1) to confirm the motion axes are ready.
4. In circular interpolation mode, no acceleration/deceleration is implemented.
5. While any 2 axes are working in circular interpolation mode, the others can not work in circular interpolation too, but point to point or linear interpolation is permitted.
6. The function MPC3035_ARC2_3P_move( ) does not need to define the motion direction, since the trajectory point has hidden definition.

Continuous motion function

- **MPC3035_set_continuous_flag**

Format : **u32 status = MPC3035_set_continuous_flag(u8 CardID,u8 Axis,u8 conti_flag);**

Purpose: To set continuous mode flag for the followed motion command.

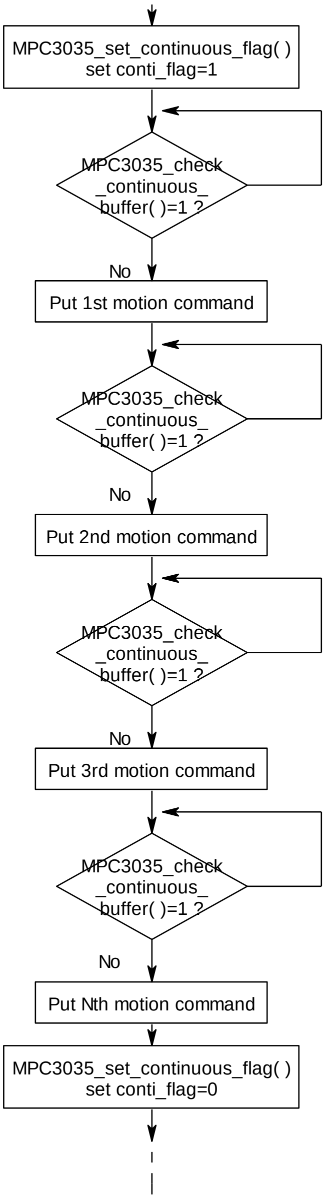
Parameters:

Input:

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

Note on using continuous mode

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


MPC3035_set_continuous_flag( )
set conti_flag=1
MPC3035_check _continuous_ buffer( )=1 ?
Yes
No
Put 1st motion command
MPC3035_check _continuous_ buffer( )=1 ?
Yes
No
Put 2nd motion command
MPC3035_check _continuous_ buffer( )=1 ?
Yes
No
Put 3rd motion command
MPC3035_check _continuous_ buffer( )=1 ?
Yes
No
Put Nth motion command
MPC3035_set_continuous_flag( )
set conti_flag=0

MPC3035_check_continuous_buffer

Format : u32 status = MPC3035_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 DIP/ROTARY switch
Axis	u8	0: X 1: Y 2: Z 3: A

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.

MPC3035_read_conti_buffer_no

Format : u32 status = MPC3035_read_conti_buffer_no(u8 CardID,u8 Axis, u8 *remain_no);

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

Parameters:

Input:

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

Output:

Name	Type	Description
remain_no	u8	remained buffered data number

*** This function is only valid for driver V2.6 and later and chip PCL6045A or version up.

- **MPC3035_read_motion_status**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch	
Axis	u8	0: X 2: Z	1: Y 3: A
check_factor	u8	0: check SEND flag (pulse output flag, no pulse out=1) 1: check SPRF flag (continuous buffer flag, buffer full =1)	

Output:

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

Position change on the fly function

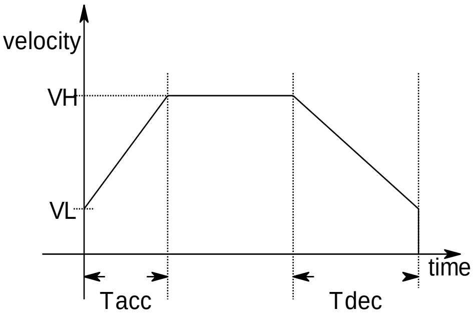
- **MPC3035_OnLine_T_curve_change**

Format : **u32 status = MPC3035_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 for single axis.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
Axis	u8	0: X 1: Y 2: Z 3: A
Position	i32	new target position (-134,217,728 ≦ Position ≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VH	u32	 VH,VL:pps, start speed (0 ≦ VL ≦ 6553500) Tacc,Tdec: seconds.
Tacc	f64	
Tdec	f64	
VH	u32	

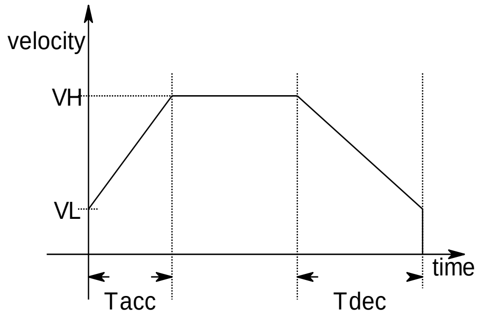
● MPC3035_OnLine_T_curve_change_LINE2

Format : **u32 status = MPC3035_OnLine_T_curve_change_LINE2(u8 CardID,**
u8 line2_index,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 any 2 axes linear interpolation.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
line2_index	u8	0: X、Y 1: X、Z 2: X、A 3: Y、Z 4: Y、A 5: Z、A
Position1	i32	new target position for first axis (-134,217,728 ≦ Position1 ≦ 134,217,727)
Position2	i32	new target position for second axis (-134,217,728 ≦ Position2 ≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VH	u32	 VH,VL:pps, start speed (0 ≦ VL ≦ 6553500) Tacc,Tdec: seconds.
Tacc	f64	
Tdec	f64	
VH	u32	

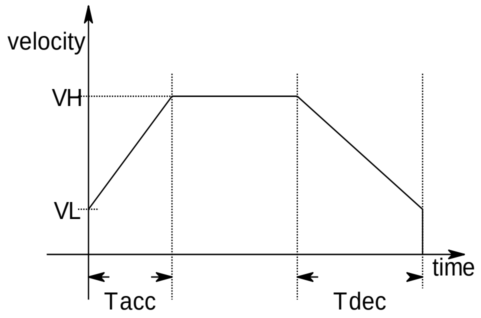
MPC3035_OnLine_T_curve_change_LINE3

Format : **u32 status = MPC3035_OnLine_T_curve_change_LINE3(u8 CardID,**
u8 line3_index,i32 Position1,i32 Position2,i32 Position3,
u8 posi_mode,i32 VL,i32 VH,f64 Tacc,f64 Tdec);

Purpose: To change the motion parameters on the fly for any 3 axes linear interpolation.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
line3_index	u8	0: X,Y, Z 1: X, Y, A 2: X, Z, A 3: Y, Z, A
Position1	i32	new target position for the first axis (-134,217,728 ≦ Position1 ≦ 134,217,727)
Position2	i32	new target position for the second axis (-134,217,728 ≦ Position2 ≦ 134,217,727)
Position3	i32	new target position for the third axis (-134,217,728 ≦ Position3 ≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VH	u32	 VH,VL:pps, start speed (0 ≦ VL ≦ 6553500) Tacc,Tdec: seconds.
Tacc	f64	
Tdec	f64	
VH	u32	

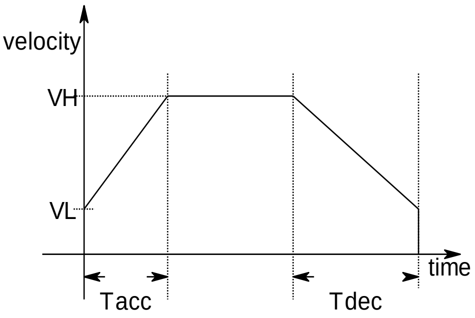
- **MPC3035_OnLine_T_curve_change_LINE4**

Format : **u32 status = MPC3035_OnLine_T_curve_change_LINE4(u8 CardID,**
i32 PositionX,i32 PositionY,i32 PositionZ,i32 PositionA,
u8 posi_mode,i32 VL,i32 VH,f64 Tacc,f64 Tdec);

Purpose: To change the motion parameters on the fly for 4 axes linear interpolation.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
PositionX	i32	new target position for the X axis (-134,217,728 ≦ PositionX ≦ 134,217,727)
PositionY	i32	new target position for the Y axis (-134,217,728 ≦ PositionY ≦ 134,217,727)
PositionZ	i32	new target position for the Z axis (-134,217,728 ≦ PositionZ ≦ 134,217,727)
PositionA	i32	new target position for the A axis (-134,217,728 ≦ PositionA ≦ 134,217,727)
posi_mode	u8	0: relative 1: absolute
VH	u32	 VH,VL:pps, start speed (0 ≦ VL ≦ 6553500) Tacc,Tdec: seconds.
Tacc	f64	
Tdec	f64	
VH	u32	

Motion resume function

● MPC3035_OneAxis_restart

Format : **u32 status = MPC3035_OneAxis_restart(u8 CardID,u8 axis);**

Purpose: To restart the previously halted axis.

Parameters:

Input:

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

***From driver V2.6 and later matched with chip PCL6045A or version up

● MPC3035_2Axis_restart

Format : **u32 status = MPC3035_2Axis_restart(u8 CardID,u8 Axis2_index);**

Purpose: To restart the previously halted 2 axes.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
Axis2_index	u8	0: X,Y 1:X,Z 2: X,A 3: Y,Z 4: Y,A 5: Z,A

*******From driver V2.6 and later matched with chip PCL6045A or version up

● MPC3035_3Axis_restart

Format : **u32 status = MPC3035_3Axis_restart(u8 CardID,u8 Axis3_index);**

Purpose: To restart the previously halted 3 axes.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
Axis3_index	u8	0: X,Y,Z 1:X,Y,A 2: X,Z,A 3: Y,Z,A

***From driver V2.6 and later matched with chip PCL6045A or version up

- **MPC3035_4Axis_restart**

Format : **u32 status = MPC3035_4Axis_restart(u8 CardID);**

Purpose: To restart the previously halted 4 axes.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

***From driver V2.6 and later matched with chip PCL6045A or version up

Interrupt function and motion event

- ### MPC3035_enable_IRQ

Format : **u32 status = MPC3035_enable_IRQ(u8 CardID,HANDLE *phEvent);**

Purpose: To enable the interrupt function.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch

Output:

Name	Type	Description
phEvent	HANDLE	returned event handle

- ### MPC3035_disable_IRQ

Format : **u32 status = MPC3035_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 switch

- ### MPC3035_link_IRQ_process

Format : **u32 status = MPC3035_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 switch
callbackAddr	void	callback address of service routine

MPC3035_set_INT_source

Format : **u32 status = MPC3035_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 switch</td></tr>
<tr><td>Axis</td><td>u8</td><td colspan="3">0: X 1: Y
2: Z 3: A</td></tr>
<tr><td>REST_source_sel</td><td>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></td><td></td><td>Bit</td><td>Name</td><td>Description</td></tr>
<tr><td></td><td></td><td>bit0</td><td>ESC1</td><td>SL+ (Software Limit +) error</td></tr>
<tr><td></td><td></td><td>bit1</td><td>ESC2</td><td>SL- (Software Limit -) error</td></tr>
<tr><td></td><td></td><td>bit2</td><td></td><td>reserved</td></tr>
<tr><td></td><td></td><td>bit3</td><td></td><td>reserved</td></tr>
<tr><td></td><td></td><td>bit4</td><td>ESC5</td><td>compare action satisfied</td></tr>
<tr><td></td><td></td><td>bit5</td><td>ESPL</td><td>LS+(EL+) error</td></tr>
<tr><td></td><td></td><td>bit6</td><td>ESML</td><td>LS-(EL-) error</td></tr>
<tr><td></td><td></td><td>bit7</td><td>ESAL</td><td>ALM error</td></tr>
<tr><td></td><td></td><td>bit8</td><td>ESSP</td><td>CSTP error</td></tr>
<tr><td></td><td></td><td>bit9</td><td>ESEM</td><td>EMG error</td></tr>
<tr><td></td><td></td><td>bit10</td><td>ESSD</td><td>SD error</td></tr>
<tr><td></td><td></td><td>bit11</td><td></td><td>reserved</td></tr>
<tr><td></td><td></td><td>bit12</td><td>ESDT</td><td>Abnormal data</td></tr>
<tr><td></td><td></td><td>bit13</td><td>ESIP</td><td>Abnormal stop during interpolation</td></tr>
<tr><td></td><td></td><td>bit14</td><td>ESPO</td><td>PA/PB input counter overflow</td></tr>
<tr><td></td><td></td><td>bit15</td><td>ESAO</td><td>In-position counter exceed the counting range during interpolation</td></tr>
<tr><td></td><td></td><td>bit16</td><td>ESEE</td><td>EA/EB input error</td></tr>
<tr><td></td><td></td><td>bit17</td><td>ESPE</td><td>PA/PB input error</td></tr>
<tr><td>RIST_source_sel</td><td>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></td><td></td><td>Bit</td><td>Name</td><td>Description</td></tr>
<tr><td></td><td></td><td>bit0</td><td>IREN</td><td>Normal stop</td></tr>
<tr><td></td><td></td><td>bit1</td><td>IRNX</td><td>Successive start of the next operation</td></tr>
<tr><td></td><td></td><td>bit2</td><td></td><td>Reserved</td></tr>
<tr><td></td><td></td><td>bit3</td><td></td><td>Reserved</td></tr>
<tr><td></td><td></td><td>bit4</td><td>IRUS</td><td>Start of acceleration</td></tr>
<tr><td></td><td></td><td>bit5</td><td>IRUE</td><td>End of acceleration</td></tr>
<tr><td></td><td></td><td>bit6</td><td>IRDS</td><td>Start of deceleration</td></tr>
<tr><td></td><td></td><td>bit7</td><td>IRDE</td><td>End of deceleration</td></tr>
<tr><td></td><td></td><td>bit8</td><td>IRC1</td><td>Soft limit plus active</td></tr>
<tr><td></td><td></td><td>bit9</td><td>IRC2</td><td>Soft limit minus active</td></tr>
<tr><td></td><td></td><td>bit10</td><td></td><td>Reserved</td></tr>
<tr><td></td><td></td><td>bit11</td><td>IRC4</td><td>Compare- method satisfied</td></tr>
<tr><td></td><td></td><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 MPC-3035 card, if you do not use interrupt function please use MPC3035_set_event_factor( ) instead.

- **MPC3035_read_INT_status**

Format : **u32 status = MPC3035_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 switch
Axis	u8	0: X 1: Y 2: Z 3: A

Output:

Name	Type	Description		
IRQ_Status	u8	bit0	0: error interrupt(REST) not active 1: error interrupt(REST) active	
		bit1	0: event interrupt(RIST) not active 1: event interrupt(RIST) active	
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 MPC-3035 card, if you do not use interrupt function please use MPC3035_read_event_flag( ) and MPC3035_read_error_flag( ) instead.

● MPC3035_set_INT_mask

Format : u32 status = MPC3035_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 switch
Axis	u8	0: X 1: Y 2: Z 3: A
on_off	u8	0: disable 1: enable

MPC3035_set_event_factor

Format : **u32 status = MPC3035_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:

Name	Type	Description		
CardID	u8	assigned by DIP/ROTARY switch		
Axis	u8	0: X 2: Z	1: Y 3: A	
event_factor	u32	any bit of the following set to “1” means if the source is active, there is an interrupt will be generated.		
		Bit	Name	Description
		bit0	IREN	Normal stop
		bit1	IRNX	Successive start of the next operation
		bit2		reserved
		bit3		reserved
		bit4	IRUS	Start of acceleration
		bit5	IRUE	End of acceleration
		bit6	IRDS	Start of deceleration
		bit7	IRDE	End of deceleration
		bit8	IRC1	Soft limit plus active
		bit9	IRC2	Soft limit minus active
		bit10		reserved
		bit11	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	IRSA	CSTA (common start) input on

Note:

This function is only used in the application program that **do not** use interrupt, if your application implemented with the interrupt function of the MPC-3035 card, please use MPC3035_set_INT_source( ) instead.

● MPC3035_read_event_flag

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
Axis	u8	0: X 1: Y 2: Z 3: A

Output:

<table>
<tr><th>Name</th><th>Type</th><th colspan="3">Description</th></tr>
<tr><td rowspan="22">event_flag</td><td rowspan="22">u32</td><td colspan="3">while any of the following bit set to “1” means the event source is active.</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>
<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></td><td>reserved</td></tr>
<tr><td>bit19</td><td>IRSA</td><td>CSTA (common start) input on</td></tr>
</table>

Note:

This function is only used in the application program that **do not** use interrupt, if your application implemented with the interrupt function of the MPC-3035 card, please use MPC3035_read_INT_status( ) instead.

MPC3035_read_error_flag

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
Axis	u8	0: X 1: Y 2: Z 3: A

Output:

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

Note:

This function is only used in the application program that **do not** use interrupt, if your application implemented with the interrupt function of the MPC-3035 card, please use MPC3035_read_INT_status( ) instead.

Soft limit protection function

- ### MPC3035_config_softlimit

Format : u32 status = MPC3035_config_softlimit(u8 CardID,u8 Axis,u8 source_sel, u8 SL_action);

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

Parameters:

Input:

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

- ### MPC3035_readback_config_softlimit

Format : u32 status = MPC3035_readback_config_softlimit(u8 CardID,u8 Axis, u8* source_sel, u8* SL_action);

Purpose: Readback the software limit parameter.

Parameters:

Input:

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

Output:

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

- **MPC3035_set_softlimit_data**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
Axis	u8	0: X 1: Y 2: Z 3: A
P_limit	i32	soft limit of positive direction (-134,217,728 ≦P_limit≦+134,217,727)
N_limit	i32	soft limit of negaitive direction (-134,217,728 ≦N_limit≦+134,217,727)

- **MPC3035_readback_softlimit_data**

Format : **u32 status = MPC3035_readback_softlimit_data(u8 CardID,u8 Axis, i32* P_limit, i32* N_limit);**

Purpose: Readback the coordinate of software limit.

Parameters:

Input:

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

Output:

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

- **MPC3035_enable_softlimit**

Format : **u32 status = MPC3035_enable_softlimit(u8 CardID,u8 Axis,u8 ON_OFF);**

Purpose: To enable / disable software limit.

Parameters:

Input:

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

- **MPC3035_readback_enable_softlimit**

Format : **u32 status = MPC3035_readback_enable_softlimit(u8 CardID,u8 Axis,**
u8* ON_OFF);

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

Parameters:

Input:

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

Output:

Name	Type	Description	
ON_OFF	u8	0: disable	1: enable

- **MPC3035_read_softlimit_flag**

Format : **u32 status = MPC3035_read_softlimit_flag(u8 CardID,u8 Axis,u8 *P_limit_flag,**
u8 *N_limit_flag);

Purpose: To read back software limit flag.

Parameters:

Input:

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

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

Manual pulser function

- **MPC3035_set_pulser_Map**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
Axis	u8	0: Motion axis is X 1: Motion axis is Y 2: Motion axis is Z 3: Motion axis is A
Map_source	u8	0: Pulse handler in X axis 1: Pulse handler in Y axis 2: Pulse handler in Z axis 3: Pulse handler in A axis
Direction	u8	0: motion rotate same direction with pulse handler 1: motion rotate counter direction with pulse handler

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

- **MPC3035_enable_pulser_motion**

Format : **u32 status = MPC3035_enable_pulser_motion(u8 CardID,u8 Axis,u8 enable, u16 Multiple);**

Purpose: To enable pulse handler function and the multiple rate

Parameters:

Input:

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

Note1:

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

Note2:

Be sure the motion pulse output is finished before using MPC3035_enable_pulser_motion( ) function, you can check it by the value of ret_flag which is returned by calling MPC3035_read_motion_status( ) and set check_factor=0.

Note3:

Be sure to disable pulse handler function before calling any motion command, such as MPC3035_T_curve_position_move(), MPC3035_S_curve_position_move()…

- **MPC3035_config_pulser_mode**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
Axis	u8	0: X 1: Y 2: Z 3: A
pulser_mode	u8	0: quadrature input A lead B up count, multiply by 1 1: quadrature input A lead B up count, multiply by 2 2: quadrature input A lead B up count, multiply by 4 3: count up at A phase rising edge, count down at B phase rising edge
direction	u8	override the default direction 0: as default direction 1: invert the direction

- **MPC3035_readback_pulser_mode**

Format : **u32 status = MPC3035_readback_pulser_mode(u8 CardID,u8 Axis, u8* pulser_mode,u8* direction);**

Purpose: Readback the pulse handler operation mode.

Parameters:

Input:

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

Output:

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

- **MPC3035_run_pulser_Vmove**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
Axis	u8	0: X 1: Y 2: Z 3: A
Maxspeed	i32	pps, the maximum pulse output that follows pulse handler input. (0≦Maxspeed≦6553500)

- **MPC3035_run_pulser_Pmove**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch	
Axis	u8	0: X 2: Z	1: Y 3: A
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)	

- **MPC3035_set_pulser_counter**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch	
Axis	u8	0: X 2: Z	1: Y 3: A
counter_value	i32	pulse counter value to be set (-134,217,728≦counter_value≦134,217,727)	

- **MPC3035_read_pulser_counter**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch	
Axis	u8	0: X 2: Z	1: Y 3: A

Output:

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

Multi-function feedback counter

- **MPC3035_set_pulse_inmode**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
pulse_inmode	u8	0 ~ 3 (See Note on pulse in mode)
count_dir	u8	0: normal counting 1: reverse counting

Note:

On wiring board terminal marked asEA+, EA- (differential for phase A input), EB+, EB- (phase B input), EZ+, EZ- (phase Z input or sometimes called phase C input)

- **MPC3035_readback_pulse_inmode**

Format : **u32 status = MPC3035_readback_pulse_inmode(u8 CardID,u8 Axis, u8* pulse_inmode,u8* count_dir);**

Purpose: Readback the parameters of the encoder input mode**.**

Parameters:

Input:

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

Output:

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

Note on pulse in mode:

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

- **MPC3035_read_FB_counter**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
Axis	u8	0: X 1: Y 2: Z 3: A

Output:

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

- **MPC3035_set_FB_counter**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
Axis	u8	0: X 1: Y 2: Z 3: A
value	i32	pulse counter value (-134,217,728 ≦ value ≦ 134,217,727)

- **MPC3035_config_LTC_PIN**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
enable	u8	0: treat LTC PIN as a general input. 1: treat LTC PIN as a dedicated external trigger to latch input.
ltc_logic	u8	0: setting the pin 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.

- **MPC3035_readback_LTC_PIN**

Format : **u32 status = MPC3035_readback_LTC_PIN(u8 CardID,u8 Axis,u8* enable, u8* ltc_logic,u8* state);**

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

Parameters:

Input:

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

Output:

Name	Type	Description
enable	u8	0: treat LTC PIN as a general input. 1: treat LTC PIN as a dedicated external trigger to latch input. Active state of LTC will latch the encoder feedback counter value on the fly. User can use ***MPC3035_read_FBcounter_latch_value( )*** to read the latched counter value
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.
state	u8	state of LTC pin

Note: On wiring board terminal marked asLTC

- **MPC3035_read_FBcounter_latch_value**

Format : **u32 status = MPC3035_read_FBcounter_latch_value(u8 CardID,u8 Axis, i32 *value);**

Purpose: To read the latced value of feedback counter.

Parameters:

Input:

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

Output:

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

- **MPC3035_config_CMP_OUT**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
axis	u8	0: X axis 1: Y axis 2: Z axis 3: A axis
cmp_mode	u8	0: treat CMP PIN as a general output point. 1: treat CMP PIN as a dedicate output ,while comparator condition satisfied, this pin active to GND level (NMOS) or relay contactor short to COM. 2: treat CMP PIN as a dedicate output , while comparator condition satisfied, this pin active to floating level (NMOS) or relay contactor open to COM point.

Note: On wiring board terminal marked asCMP

MPC3035_readback_CMP_OUT

Format : u32 status = MPC3035_readback_CMP_OUT(u8 CardID,u8 Axis,
u8* cmp_mode,u8* state);

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

Parameters:

Input:

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

Output:

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

- **MPC3035_config_comparator_out**

Format : **u32 status = MPC3035_config_comparator_out(u8 CardID,u8 Axis, u8 cmp_source,u8 cmp_method,u8 cmp_action);**

Purpose: To setup the compare mode of feedback comparator.

Parameters:

Input:

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

- **MPC3035_readback_comparator_out**

Format : **u32 status = MPC3035_readback_comparator_out(u8 CardID,u8 Axis, u8* cmp_source,u8* cmp_method,u8* cmp_action);**

Purpose: Readback the configuration of the compare mod**e.**

Parameters:

Input:

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

Output:

Name	Type	Description
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

- **MPC3035_set_comparator_data**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch	
Axis	u8	0: X 2: Z	1: Y 3: A
cmp_data	i32	comparator value to be preset (-134,217,728≦cmp_data≦134,217,727)	

- **MPC3035_readback_comparator_data**

Format : **u32 status = MPC3035_readback_comparator_data(u8 CardID,u8 Axis, i32* cmp_data);**

Purpose: Readback the preset comparator value.

Parameters:

Input:

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

Output:

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

- **MPC3035_read_compare_flag**

Format : **u32 status = MPC3035_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 DIP/ROTARY switch
Axis	u8	0: X 1: Y 2: Z 3: A

Output:

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

- **MPC3035_config_comparator_out_W**

Format : **u32 status = MPC3035_config_comparator_out_W(u8 CardID,u8 Axis,**
u8 cmp_source,u8 cmp_action);

Purpose: To setup the compare window of feedback comparator.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
Axis	u8	0: X 1: Y 2: Z 3: A
cmp_source	u8	0: to compare with the current position command counter 1: to compare with the feedback counter 2: undefined 3: to compare with the pulser counter
cmp_action	u8	0: No action, use only to generate interrupt and compare output 1: immediate stop 2: decelerate to stop

- **MPC3035_readback_comparator_out_W**

Format : **u32 status = MPC3035_readback_comparator_out_W(u8 CardID,u8 Axis,**
u8* cmp_source,u8* cmp_action);

Purpose: Readback the configuration of the window compare mode.

Parameters:

Input:

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

Output:

Name	Type	Description
cmp_source	u8	0: to compare with the current position command counter 1: to compare with the feedback counter 2: undefined 3: to compare with the pulser counter
cmp_action	u8	0: No action, use only to generate interrupt and compare output 1: immediate stop 2: decelerate to stop

- **MPC3035_set_comparator_data_W**

Format : **u32 status = MPC3035_set_comparator_data_W(u8 CardID,u8 Axis, i32 cmp_data_minus, i32 cmp_data_plus);**

Purpose: To preset the comparator value.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
Axis	u8	0: X 1: Y 2: Z 3: A
cmp_data_minus	i32	window comparator lower limit value to be preset (-134,217,728≦cmp_data_minus≦134,217,727)
cmp_data_plus	i32	window comparator upper limit value to be preset (-134,217,728≦cmp_data_plus≦134,217,727)

- **MPC3035_readback_comparator_data_W**

Format : **u32 status = MPC3035_readback_comparator_data_W(u8 CardID,u8 Axis, i32* cmp_data_minus, i32* cmp_data_plus);**

Purpose: Readback the preset comparator value.

Parameters:

Input:

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

Output:

Name	Type	Description
cmp_data_minus	i32	window comparator lower limit value to be preset (-134,217,728≦cmp_data_minus≦134,217,727)
cmp_data_plus	i32	window comparator upper limit value to be preset (-134,217,728≦cmp_data_plus≦134,217,727)

- **MPC3035_read_compare_flag_W**

Format : **u32 status = MPC3035_read_compare_flag_W(u8 CardID,u8 Axis, u8 *cmp_flag_minus, u8 *cmp_flag_plus);**

Purpose: To read back the compare flag**.**

Parameters:

Input:

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

Output:

Name	Type	Description
cmp_flag_minus	u8	0: current position is less than lower limit value 1: current position is larger than lower limit value
cmp_flag_plus	u8	0: current position is larger than upper limit value 1: current position is less than upper limit value

Software key function

- **MPC3035_unlock_security**

Format : **u32 status = MPC3035_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 switch
password[5]	u16	The password previous set

- **MPC3035_read_security_status**

Format : **u32 status = MPC3035_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 switch

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.

- **MPC3035_set_serial_code**

Format : **u32 status = MPC3035_set_serial_code(u8 CardID, u8 index_no, u16 serial_code);**

Purpose: To setup the serial code.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
index_no	u8	index number of serial code (0~5)
serial_code	u16	serial code to be write

- **MPC3035_read_serial_code**

Format : **u32 status = MPC3035_read_serial_code(u8 CardID, u8 index_no, u16* serial_code);**

Purpose: To read the serial code.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY switch
index_no	u8	index number of serial code (0~5)

Output:

Name	Type	Description
serial_code	u16	serial code read

Note :

Serial code does not concern with the operation of DLL's, it is just provide a mechanism for the user to control the software for his own purpose.

- **MPC3035_set_password**

Format : **u32 status = MPC3035_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 switch
password[5]	u16	Password, 5 words

Note on password:

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

- **MPC3035_change_password**

Format : **u32 status = MPC3035_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 switch
Oldpassword [5]	u16	The previous password
password[5]	u16	The new password to be set

- **MPC3035_clear_password**

Format : **u32 status = MPC3035_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 switch
password[5]	u16	The password previous set

Counter setup

- ## MPC3035L_set_input_polarity

Format : **u32 status = MPC3035L_set_input_polarity (u8 CardID, u8 axis, u8 point, u8 polarity)**

Purpose: To set MPC3035L card's input point polarity.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axes	u8	0: X	1: Y
point	u8	0: A_PHASE input point 1: B_PHASE input point 2: Z_PHASE input point 3: N/A 4: N/A 5: N/A 6: INn (general) input point	
polarity	u8	0: normal. (default) 1: invert.	

Note:

X axis:	MPC3035L 25pin Female D type definition
A_PHASE input	Differential input XA+ (pin 1) XA- (pin 14)
B_PHASE input	Differential input XB+ (pin 2) XB- (pin 15)
Z_PHASE input	Differential input XZ+ (pin 3) XZ- (pin 16)
INx	IN0 (pin 7)
Y axis:	
A_PHASE input	Differential input YA+ (pin 4) XA- (pin 17)
B_PHASE input	Differential input YB+ (pin 5) XB- (pin 18)
Z_PHASE input	Differential input YZ+ (pin 6) XZ- (pin 19)
INy	IN1 (pin 20)

- **MPC3035L_read_input_polarity**

Format : **u32 status = MPC3035L_read_input_polarity (u8 CardID, u8 axis, u8 point, u8 *polarity)**

Purpose: To read back polarity of input point.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axes	u8	0: X	1: Y
point	u8	0: A_PHASE input point 1: B_PHASE input point 2: Z_PHASE input point 3: N/A 4: N/A 5: N/A 6: INn (general) input point	

Output:

Name	Type	Description
polarity	u8	0: normal. (default) 1: invert.

- **MPC3035L_set_output_polarity**

Format : **u32 status = MPC3035L_set_output_polarity (u8 CardID, u8 axis, u8 point, u8 polarity)**

Purpose: To set MPC3035L card's output polarity.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axes	u8	0: X	1: Y
point	u8	0: always use 0	
polarity	u8	0: normal. (default) 1: invert.	

Note:

X axis:	MPC3035L 25pin Female D type definition
Output	Differential output XOUT+ (pin 1) XOUT- (pin 14)
Y axis:	
Output	Differential output YOUT+ (pin 1) YOUT- (pin 14)

MPC3035L_read_output_polarity

Format : u32 status = MPC3035L_read_output_polarity (u8 CardID, u8 axis, u8 point, u8 *polarity)

Purpose: To read back polarity of output point.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axes	u8	0: X	1: Y
point	u8	0: always use 0	

Output:

Name	Type	Description
polarity	u8	0: normal. (default) 1: invert.

MPC3035L_read_input_status

Format : u32 status = MPC3035L_read_input_status (u8 CardID, u8 axis, u8 point, u8 *state)

Purpose: To read MPC3035L card’s input status.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axes	u8	0: X	1: Y
point	u8	0: A_PHASE input point 1: B_PHASE input point 2: Z_PHASE input point 3: N/A 4: N/A 5: N/A 6: INn (general) input point 7: Z_PHASE trigger toggled flag	

Output:

Name	Type	Description
state	u8	0: 0V, GND level. 1: 5V, high level.

Note:

The returned state is the physical input xor input polarity.

Say physical input of A_PHASE is GND level, the polarity you set is normal, then you will get the input status at 0. But if you set the polarity to invert, then the input status will be 1.

- **MPC3035L_set_counter_mode**

Format : **u32 status = MPC3035L_set_counter_mode (u8 CardID, u8 axis, u8 mode)**

Purpose: Set MPC3035L card counter mode.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
mode	u8	0: A,B phases input quadrature up count mode(if A lead B).(default) 1: A,B phases input quadrature down count mode(if A lead B). 2: A input is CLOCK,B input is DIRECTION,up count mode. 3: A input is CLOCK,B input is DIRECTION,down count mode. 4: A input is UP CLOCK,B input is DOWN CLOCK,dual clock mode. 5: A input is DOWN CLOCK,B input is UP CLOCK,dual clock mode.	

Note:

You can change the A, B phase connection or set the counting mode in quadrature mode to change the counting direction.

- **MPC3035L_set_quadrature_times**

Format : **u32 status = MPC3035L_set_quadrature_times (u8 CardID, u8 axis, u8 times)**

Purpose: Set MPC3035L card quadrature decoding rate as the counter mode setting in A/B phase quadrature mode.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
times	u8	0: x 4 (default) 1: x 2 2: x 1	

Homing function

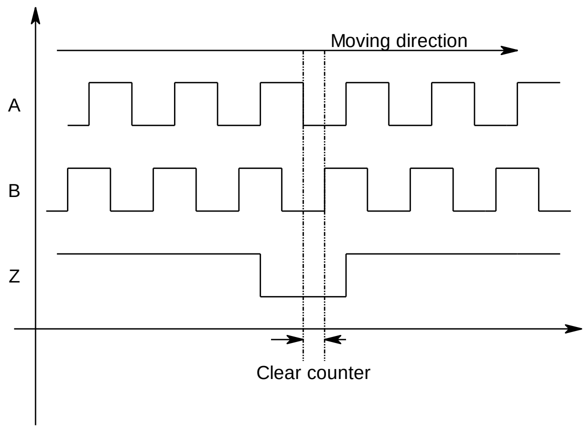
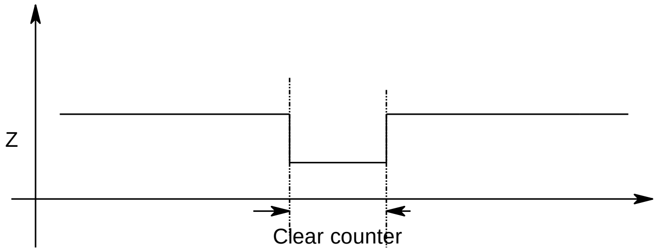
- **MPC3035L_set_hard_homing**

Format : **u32 status = MPC3035L_set_hard_homing (u8 CardID, u8 axis, u8 mode)**

Purpose: Set hardware homing mode (hardware clear counter).

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axes	u8	0: X	1: Y
mode	u8	hardware homing mode 0: Normal operation (default) 1: Clear counter while A,B,Z, signal are "LOW" simultaneously. This mode will keep continuously until you switch to other homing mode.  5: Clear counter at Z input active low. This mode will keep continuously until you switch to other homing mode.  6: Clear counter while A,B,Z, signal are "LOW" simultaneously. Once the counter cleared, this command will also cleared to "0" (Normal mode). A: Clear counter at Z input active low. Once the counter cleared, this command will also cleared to "0" (Normal mode).	

Note: The signal mentioned is the result of physical input xor the polarity.

- **MPC3035L_read_hard_homing_flag**

Format : **u32 status = MPC3035L_read_hard_homing_flag (u8 CardID, u8 axis, u8 *flag)**

Purpose: Read MPC3035L card's hardware homing flag.

Parameters:

Input:

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

Output:

Name	Type	Description
flag	u8	0: no operation. 1:hardware homing happened, when read, the flag will reset to 0

Note:

The homing flag only can read once each time hardware homing happens; because the function will clear the flag after it is read.

- **MPC3035L_soft_homing_command**

Format : **u32 status = MPC3035L_soft_homing_command (u8 CardID, u8 axis)**

Purpose: To clear counter.

Parameters:

Input:

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

Counter function

- **MPC3035L_read_counter**

Format : **u32 status = MPC3035L_read_counter (u8 CardID, u8 axis, i32 *value)**

Purpose: To read MPC3035L card counter. The 32bit counter value will return.

Parameters:

Input:

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

Output:

Name	Type	Description
value	i32	32bit counter value. (-2,147,483,648 ~ 2,147,483,647)

- **MPC3035L_load_counter**

Format : **u32 status = MPC3035L_load_counter (u8 CardID, u8 axis, i32 value)**

Purpose: To load value into MPC3035L card counter.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axes	u8	0: X	1: Y
value	i32	32bit counter value to be load. (-2,147,483,648 ~ 2,147,483,647)	

Note:

Use MPC3035L_load_counter( ) to set current position after the hardware homing means you define the coordinate origion (at certain point).

- **MPC3035L_latch_control**

Format : **u32 status = MPC3035L_latch_control (u8 CardID, u8 Axis, u8 control)**

Purpose: To enable/disable latch function.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
Axes	u8	0: X	1: Y
control	u8	0:disable latch function. 1:enable latch function. (Default value)	

- **MPC3035L_latch_mode**

Format : **u32 status = MPC3035L_latch_mode (u8 CardID, u8 Axis, u8 mode)**

Purpose: To assign the hardware trigger input function.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
Axes	u8	0: X	1: Y
mode	u8	0 : continuous trigger latch (counter) mode. (default) 1: one-shot trigger latch mode. **Once triggered the further trigger will be disabled. New trigger should be enable by MPC3035L_latch_control( ).**	

Note:

If you want to latch counter data on the fly

1. enable latch function (MPC3035L***_latch_control***)
2. select latch mode (MPC3035L***_latch_control***)
3. hardware now is waiting the compare equal trigger, once the trigger occurs at its active edge, immediately the counter value will be latched by hardware mechanism, simultaneously interrupt request may occur.

- **MPC3035L_read_latch_flag**

Format : **u32 status = MPC3035L_read_latch_flag (u8 CardID, u8 axis, u8 *flag)**

Purpose: To read latch flag

Parameters:

Input:

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

Output:

Name	Type	Description
flag	u8	0:no operation. 1:occurrence of latch trigger event.

Note: To read the latch flag will reset latch flag to no operation state.

- **MPC3035L_read_latched_value**

Format : **u32 status = MPC3035L_read_latched_value (u8 CardID, u8 axis, i32 *value)**

Purpose: To read **MPC3035L** counter latched value.

Parameters:

Input:

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

Output:

Name	Type	Description
value	i32	32bit counter latched value. (-2,147,483,648 ~ 2,147,483,647)

- **MPC3035L_write_XY_latch**

Format : **u32 status = MPC3035L_write_XY_latch (u8 CardID ,u8 value)**

Purpose: To disable or enable the X,Y axes simutaneous latch by X compare trigger.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
value	u8	0:disable 1:enable

- **MPC3035L_read_XY_latch**

Format : **u32 status = MPC3035L_read_XY_latch (u8 CardID,u8 *value)**

Purpose: To read back the flag of disable or enable setting of the X,Y axes simutaneous latch by X compare trigger.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Output:

Name	Type	Description
value	u8	0:disable 1:enable

Compare function

MPC3035L_set_CMP_OUT_mode

Format : **u32 status = MPC3035L_set_CMP_OUT_mode (u8 CardID, u8 axis, u8 mode)**

Purpose: To set the CMP_OUT (compare out) mode.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axes	u8	0: X	1: Y
mode	u8	0: take CMP_OUT output point as general purpose output. (no compare function) 1: one time compare mode during the counter value meet the preset compare value, a pulse is active at CMP_OUT point. 2: auto increment mode during the counter value meet the preset compare value a pulse is active at CMP_OUT point. The next compare value auto increased. (refer MPC3035L_load_increment_value) 3: auto load from FIFO mode during the counter meet the preset compare value, a pulse is active at CMP_OUT point. The next compare value auto loaded from FIFO (X axis only). 4. auto load from FIFO mode, During the counter meet the preset compare value, a toggle output at CMP_OUT point. The next compare value auto loaded from FIFO. (X axis only)	

Note:

1. Both the “auto increment mode” and “auto load from FIFO mode” must have load the compare value as the first compare data to enter the specific working mode.
2. The output pulse duration time is defined by *MPC3035L_set_CMP_oneshot_duration( ).*
3. The CMP_OUT is differential out on D 25 connector defined as XOUT+, XOUT- fro X axis, YOUT+, YOUT- fro Y axis,

MPC3035L_write_output_command

Format : **u32 status = MPC3035L_write_output_command (u8 CardID, u8 axis, u8 point, u8 on_off)**

Purpose: To write MPC3035L card's output point.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axes	u8	0: X	1: Y
		2: Z	3: A
point	u8	0: always use 0	
on_off	u8	0: 0V, GND level. 1: 5V, high level.	

Note:

The physical output will be the command output xor output polarity.

Say you want the physical output go to GND level, the polarity you set is normal, then you should command the on_off at 0. But if you set the polarity to invert, then you must command on_off at 1.

MPC3035L_read_output_status

Format : **u32 status = MPC3035L_read_output_status (u8 CardID, u8 axis, u8 point, u8 *state)**

Purpose: To read back status of output point.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axes	u8	0: X	1: Y
point	u8	0: always use 0	

Output:

Name	Type	Description
state	u8	0: 0V, GND level. 1: 5V, high level.

Note: The returned state is the previous command of on_off.

- **MPC3035L_load_compare_value**

Format : **u32 status = MPC3035L_load_compare_value (u8 CardID, u8 axis, i32 value)**

Purpose: To set the compare value.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axes	u8	0: X	1: Y
value	i32	32 bit value (-2,147,483,648 ~ 2,147,483,647) to be compared with counter.	

- **MPC3035L_read_compare_value**

Format : **u32 status = MPC3035L_read_compare_value (u8 CardID, u8 axis, i32 *value)**

Purpose: To read the compare value.

Parameters:

Input:

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

Output:

Name	Type	Description
value	i32	32 bit value (-2,147,483,648 ~ 2,147,483,647) to be compared with counter.

- **MPC3035L_set_CMP_method**

Format : **u32 status = MPC3035L_set_CMP_method (u8 CardID,u8 Axis,**
u8 CMP_method)

Purpose: To set the compare method of compare output function.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axes	u8	0: X	1: Y
CMP_method	u8	0: equal. 1: counter Greater or Equal compare value. 2: counter Less or Equal compare value.	

Note:

Thefunction is only valid positive interger zone or negative integer zone, if sign changed (cross zero) will not work correctly.

Auto increment compare mode

- **MPC3035L_load_increase_value**

Format : **u32 status = MPC3035L_load_increase_value (u8 CardID, u8 axis, i32 value)**

Purpose: To load the increase value.

Next compare value = current compare value + increase value

At compare out mode 2 (auto increment mode), next compare value will be loaded after compare out trigger.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axes	u8	0: X	1: Y
value	i32	32 bit value (-2,147,483,648 ~ 2,147,483,647) to be compared with counter.	

- **MPC3035L_read_increase_value**

Format : **u32 status = MPC3035L_read_increase_value (u8 CardID, u8 axis, i32 *value)**

Purpose: To read the incremental data.

Parameters:

Input:

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

Output:

Name	Type	Description
value	i32	32 bit value (-2,147,483,648 ~ 2,147,483,647) to be compared with counter.

FIFO compare mode

- **MPC3035L_clear_FIFO_command**

Format : **u32 status = MPC3035L_clear_FIFO_command(u8 CardID,u8 Axis)**

Purpose: To clear (reset) FIFO.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axes	u8	0: always use 0

Note: X axis only.

- **MPC3035L_fill_FIFO_value**

Format : **u32 status = MPC3035L_fill_FIFO_value(u8 CardID,u8 Axis,i32 value)**

Purpose: To fill (load) FIFO.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axes	u8	0: always use 0
value	i32 (28bit valid)	data to be push into FIFO (-134,217,728 ≦value≦ 134,217,727)

Note:

1. X axis only.
2. To fill the FIFO, FIFO-in pointer will increase by one.
   Total 1023 available data space.

- **MPC3035L_read_FIFO_full_flag**

Format : **u32 status =MPC3035L_read_FIFO_full_flag(u8 CardID,u8 Axis,u8 *flag)**

Purpose: Read FIFO full flag

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axes	u8	0: always use 0

Output:

Name	Type	Description
flag	u8	0: FIFO not full 1: FIFO full

Note:

1. Please check FIFO full flag before push any data into FIFO.
2. X axis only.

- **MPC3035L_read_FIFO_unused_number**

Format : **u32 status =MPC3035L_read_FIFO_unused_number(u8 CardID,u8 Axis, i32*value)**

Purpose: Read FIFO unused space (number)

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axes	u8	0: always use 0

Output:

Name	Type	Description
value	i32	unsused space (number) total maximum space is 1023 data

Note: X axis only.

- **MPC3035L_read_FIFO_empty_flag**

Format : **u32 status =MPC3035L_read_FIFO_empty_flag(u8 CardID,u8 Axis,u8 *flag)**

Purpose: Read FIFO empty flag

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axes	u8	0: always use 0

Output:

Name	Type	Description
flag	u8	0: not empty 1: empty

Note: X axis only.

- **MPC3035L_read_FIFO_value**

Format : **u32 status =MPC3035L_read_FIFO_value(u8 CardID,u8 Axis, i32 *value)**

Purpose: Read data from top of FIFO

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
Axes	u8	0: always use 0

Output:

Name	Type	Description
value (28bit valid)	i32	data at the top of FIFO (-134,217,728 ≦value≦ 134,217,727)

Note:

1. The FIFO-out pointer will increase by one, i.e. one data pop out from FIFO.

2. X axis only.

Output duration

- **MPC3035L_set_CMP_oneshot_duration**

Format : **u32 status = MPC3035L_set_CMP_oneshot_duration (u8 CardID,u8 Axis, i32Value)**

Purpose: To set the one shot duration time for compare out.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axes	u8	0: X	1: Y
value	i32	Duration time = value * 1us 1<= value <= 16777215	

- **MPC3035L_set_Mask_CMP_OUT_source**

Format : **u32 status = MPC3035L_set_Mask_CMP_OUT_source(u8 CardID,u8 Axis, u8 source_sel)**

Purpose: To set the mask source of CMP_OUT .

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axes	u8	0: X	1: Y
source_sel	u8	0: no mask to CMP_OUT. 1: N/A. 2: use general INn to gate CMP_OUT. (CMP_OUT is enable/disabled by INn)	

Note: For each axis CMP_OUT and its gate

axis	Compare output	Gate input
X	XOUT	IN0
Y	YOUT	IN1

Name	Input Satus	Output
INn	0	Output disable
	1	Output enable

Compare segment configuration and compare out mask off

Note: The segment function is only valid for X axis.

- **MPC3035L_write_cmp_segment**

Format : **u32 status = MPC3035L_write_cmp_segment(u8 CardID,u8 index,u32 start,u32 stop)**

Purpose: To write the X axis segment coordinate.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
index	u8	0: Segment 0 1: Segment 1 2: Segment 2
start	u32	Start of mask off segment
stop	u32	Stop of mask off segment

- **MPC3035L_read_cmp_segment**

Format : **u32 status = MPC3035L_read_cmp_segment (u8 CardID,u8 index,u32 *start,u32 *stop)**

Purpose: To read the X axis segment coordinate.

Parameters:

Input:

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

Output:

Name	Type	Description
start	u32	Start of mask off segment
stop	u32	Stop of mask off segment

- **MPC3035L_write_mask_off**

Format : **u32 status = MPC3035L_write_mask_off (u8 CardID, u8 attribute)**

Purpose: To write the mask off attribute.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
attribute	u8	0: mask off interior 1: mask off exterior

- **MPC3035L_read_mask_off**

Format : **u32 status = MPC3035L_read_mask_off (u8 CardID,u8 *attribute)**

Purpose: To read back the segment interior or exterior attribute.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Output:

Name	Type	Description
attribute	u8	0: mask off interior 1: mask off exterior

- **MPC3035L_write_segment_control**

Format : **u32 status = MPC3035L_write_segment_control (u8 CardID,u8 index, u8 control)**

Purpose: To write the X axis segment control.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
index	u8	0: Segment 0 1: Segment 1 2: Segment 2
control	u8	0:disable 1:enable

- **MPC3035L_read_segment_control**

Format : **u32 status = MPC3035L_read_segment_control (u8 CardID,u8 index,u8 *control)**

Purpose: To read the X axis segment control.

Parameters:

Input:

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

Output:

Name	Type	Description
control	u8	0:disable 1:enable

Interrupt function

- ### MPC3035L_enable_IRQ

Format : **u32 status = MPC3035L_enable_IRQ(u8 CardID,HANDLE *phEvent)**

Purpose: To enable interrupt. This is a function to initialize IRQ resource only, you must do first of all.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Output:

Name	Type	Description
phEvent	HANDLE	The returned handle of event

- ### MPC3035L_disable_IRQ

Format : **u32 status = MPC3035L_disable_IRQ(u8 CardID)**

Purpose: To disable interrupt.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

- ### MPC3035L_set_FIFO_INT_no

Format : **u32 status = MPC3035L_set_FIFO_INT_no (u8 CardID,u8 Axis, u16 ALM_number)**

Purpose: Set refill FIFO alarm number condition to result in an interrupt

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
axis	u8	0: always use 0
ALM_number	u16	0 < ALM_number < 1023

Note: The default ALM_number= 100 , X axis only.

- **MPC3035L_link_IRQ_process**

Format : **u32 status = MPC3035L_link_IRQ_process**
(u8 CardID,,void (__stdcall *callbackAddr(u8 CardID))

Purpose: To link the interrupt event with the callback function.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
callbackAddr	void	the address of your callback function

- **MPC3035L_set_INT_mask**

Format : **u32 status = MPC3035L_set_INT_mask(u8 CardID,u8 Axis,u8 on_off)**

Purpose: To mask off the un-wanted interrupt 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	

- **MPC3035L_set_INT_source**

Format : **u32 status = MPC3035L_set_INT_source(u8 CardID,u8 Axis,u8 source_sel)**

Purpose: To set interrupt source

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
source_sel	u8	bit0: N/A. bit1: Interrupt on hardware counter clear. bit2: Interrupt on counter value equals counter compare value. bit3: Interrupt on counter carry occurred. bit4: Interrupt on counter borrow occurred. bit5: Interrupt on FIFO refill number equal to FIFO alarm number (X aixs only)	

- **MPC3035L_read_INT_status**

Format : **u32 status = MPC3035L_read_INT_status(u8 CardID,u8 Axis,u8 *Int_status)**

Purpose: To read the interrupt source.

Parameters:

Input:

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

Output:

Name	Type	Description
Int_status	u8	bit0:N/A. bit1:1, interrupt on hardware counter clear. bit2:1, interrupt on counter value equals counter compare value. bit3:1, interrupt on counter carry occurred. bit4:1, interrupt on counter borrow occurred. bit 5:1, interrupt on FIFO refill number equal to FIFO alarm number (X axis only).

- **MPC3035L_read_INT_ID**

Format : **u32 status = MPC3035L_read_INT_ID(u8 CardID,u8 *ID)**

Purpose: To read the interrupt source axis.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Output:

Name	Type	Description
ID	u8	bit0: 1, X axis generate interrupt, else 0. bit1:1, Y axis generate interrupt, else 0.

Miscellaneous function

- **MPC3035L_out_PWM_DA**

Format : **u32 status = MPC3035L_out_PWM_DA(u8 CardID, u8 axis,u16 DA_value)**

Purpose: Output the value to PWM DA**.**

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by rotary switch	
axes	u8	0: X	1: Y
DA_value	u16	0~255 data, DA value will be 0Vdc ~ 10Vdc	

Note:

The PWM DA is single end analog output on D 25 connector defined as DA1 for X axis, DA2 for Y axis.

- **MPC3035L_read_parameters**

Format : **u32 status = MPC3035L_read_parameters (u8 CardID, u8 axis, u8 parameter_no, i32 *value)**

Purpose: To read parameters currently set.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axes	u8	0: X	1: Y
Parameter_no	u8	0: homing mode (MPC3035L_set_hard_homing ()) 1: counter mode (MPC3035L_set_counter_mode ()) 2: quadrature times (MPC3035L_set_quadrature_times ()) 3: latch mode (MPC3035L _latch_mode ()) 4: latch control (MPC3035L _latch_control ()) 5: N/A. 6: N/A. 7: compare out mode (MPC3035L_set_compare_out_mode()) 8: INT_MASK (MPC3035L_set_INT_mask()) 9: INT_SOURCE (MPC3035L_set_INT_source()) 10: N/A. 11:compare duration constant (MPC3035L_set_CMP_oneshot_duration) 12: mask compare output source select (MPC3035L_set_Mask_CMP_OUT_source) 13: CMP_METHOD (MPC3035L_set_CMP_method()) 14: FIFO_INT_NO (MPC3035L_set_FIFO_INT_no(), X axis only)	

Output:

Name	Type	Description
value	i32	32 bit value (-2,147,483,648 ~ 2,147,483,647)

9.5 Dll list

	Function Name	Description
1.	MPC3035_initial( )	MPC3035 Initial
2.	MPC3035_close( )	MPC3035 Close
3.	MPC3035_init_card( )	Initialize parameters and auxiliary function to default value
4.	MPC3035_info( )	Get the I/O address and vendor ID of card
5.	MPC3035_dll_Simu_mode( )	Enter/exit simulation mode
6.	MPC3035_save_config2_file( )	Save configuration data to file
7.	MPC3035_load_config_from_file( )	Load configuration data from file
8.	MPC3035_config_TTL_IO_MODE( )	Configure TTL I/O mode
9.	MPC3035_readback_TTL_IO_MODE( )	Read back configuration of TTL_IO
10.	MPC3035_read_point_status( )	Read input status
11.	MPC3035_read_status( )	To input status
12.	MPC3035_read_port( )	To input TTL_IO port status
13.	MPC3035_write_output_point( )	Write output
14.	MPC3035_write_port( )	To set/reset TTL_IO port
15.	MPC3035_set_pulse_outmode( )	Configure the pulse output mode
16.	MPC3035_readback_pulse_outmode( )	Read back configuration of pulse output mode
17.	MPC3035_config_SD_PIN( )	Configure slow down input
18.	MPC3035_readback_SD_PIN( )	Read back configuration of SD pin
19.	MPC3035_config_EL_MODE( )	Configure LS(EL) (over travel) stop mode
20.	MPC3035_readback_EL_MODE( )	Read back configuration for LS(EL)
21.	MPC3035_config_PCS_PIN( )	Configure PCS(position change start) input
22.	MPC3035_readback_PCS_PIN( )	Read back configuration of PCS pin
23.	MPC3035_position_override( )	Set and enable the PCS distance.
24.	MPC3035_config_ERC_PIN( )	Configure ERC (error counter clear) output
25.	MPC3035_readback_ERC_PIN( )	Read back configuration of ERC pin
26.	MPC3035_config_ALM_PIN( )	Configure ALM (alarm) input
27.	MPC3035_readback_ALM_PIN( )	Read back configuration of ALM pin
28.	MPC3035_config_INP_PIN( )	Configure INP (in position) input
29.	MPC3035_readback_INP_PIN( )	Read back configuration of INP pin
30.	MPC3035_fix_speed_range( )	Set the maximum allowable speed
31.	MPC3035_unfix_speed_range( )	Release the limit of maximum allowable speed
32.	MPC3035_T_velocity_move( )	Velocity mode move at trapezoidal profile
33.	MPC3035_S_velocity_move( )	Velocity mode move at S curve profile
34.	MPC3035_velocity_change( )	To change speed on motion
35.	MPC3035_dec_stop( )	Velocity mode, deceleration to stop
36.	MPC3035_imd_stop( )	Velocity mode, immediate stop
37.	MPC3035_emg_stop( )	Velocity mode, all axes immediate stop
38.	MPC3035_read_speed( )	Read the current speed
39.	MPC3035_set_HOME_pin_logic( )	Configure HOME (ORG) polarity
40.	MPC3035_readback_HOME_pin_logic( )	Read back configuration for HOME (ORG) pin
41.	MPC3035_set_EZ_pin_logic( )	Configure EZ (zero phase) polarity
42.	MPC3035_readback_EZ_pin_logic( )	Read back configuration of EZ (zero phase) polarity
43.	MPC3035_config_home_mode( )	Select the desired homing mode
44.	MPC3035_start_homing( )	To execute homing

45.	MPC3035_set_current_position( )	Setup the coordinate of current point
46.	MPC3035_read_current_position( )	Read the coordinate of current point
47.	MPC3035_start_origin_search_homing( )	To command origin search mode homing motion
48.	MPC3035_T_curve_position_move( )	Point to point move at trapezoidal acc/dec profile
49.	MPC3035_S_curve_position_move( )	Point to point move at S curve profile
50.	MPC3035_position_change( )	Change target position while the point to point motion is running
51.	MPC3035_backlash_comp( )	Setup backlash compensation
52.	MPC3035_readback_backlash_comp( )	Read back configuration of backlash compensation
53.	MPC3035_suppress_vibration( )	Setup vibration suppression mode
54.	MPC3035_readback_suppress_vibration( )	Read back parameters of vibration suppression mode
55.	MPC3035_T_curve_move_LINE2( )	Two axes linear interpolation at trapezoidal profile
56.	MPC3035_S_curve_move_LINE2( )	Two axes linear interpolation at S curve profile
57.	MPC3035_T_curve_move_LINE3( )	3 axes linear interpolation at trapezoidal profile
58.	MPC3035_S_curve_move_LINE3( )	3axes linear interpolation at S curve profile
59.	MPC3035_T_curve_move_LINE4( )	4 axes linear interpolation at trapezoidal profile
60.	MPC3035_S_curve_move_LINE4( )	4 axes linear interpolation at S curve profile
61.	MPC3035_T _LINE_move( )	To take linear interpolation on 1~4 axes with T curve profile
62.	MPC3035_S _LINE_move( )	To take linear interpolation on 1~4 axes with S curve profile
63.	MPC3035_ARC2_center_move( )	Circular interpolation with the circle center and end position as parameters for arc trajectory
64.	MPC3035_T_ARC2_center_move( )	T trajectory circular interpolation with the circle center and end position as parameters for arc trajectory
65.	MPC3035_S_ARC2_center_move( )	S trajectory circular interpolation with the circle center and end position as parameters for arc trajectory
66.	MPC3035_ARC2_3P_move( )	Circular interpolation with current point and the other 2 points as parameters for arc trajectory
67.	MPC3035_T_ARC2_3P_move( )	T trajectory circular interpolation with current point and the other 2 points as parameters for arc trajectory
68.	MPC3035_S_ARC2_3P_move( )	S trajectory circular interpolation with current point and the other 2 points as parameters for arc trajectory
69.	MPC3035_ARC2_Radius_move( )	Circular interpolation with radius and end position as parameters for arc trajectory
70.	MPC3035_T_ARC2_Radius_move( )	T trajectory circular interpolation with radius and end position as parameters for T profile acc/dec arc trajectory
71.	MPC3035_CIR2_3P_move( )	Circle with current point and the other 2 points as parameters to pass through
72.	MPC3035_T_CIR2_3P_move( )	T trajectory circle with current point and the other 2 points as parameters to pass through
73.	MPC3035_S_CIR2_3P_move( )	S trajectory circle with current point and the other 2 points as parameters to pass through
74.	MPC3035_CIR2_Radius_move( )	Circular interpolation with radius and end position as parameters for circular trajectory
75.	MPC3035_T_CIR2_Radius_move( )	Circular interpolation with radius and end position as

		parameters for T profile acc/dec circular trajectory
76.	MPC3035_ArcXY_LineZ_center_move( )	X,Y axes doing arc interpolation as designated parameters and Z axis doing linear interpolation synchronously.
77.	MPC3035_ArcXY_LineZ_3P_move( )	X,Y axes doing arc interpolation as designated parameters and Z axis doing linear interpolation synchronously.
78.	MPC3035_CirXY_LineZ_3P_move( )	X,Y axes doing circular interpolation as designated parameters and Z axis doing linear interpolation synchronously.
79.	MPC3035_enable_FIFO( )	To enable the FIFO function
80.	MPC3035_check_FIFO_buffer( )	To check remained FIFO data
81.	MPC3035_T_curve_write_FIFO( )	To fill FIFO with one dimension’s datum
82.	MPC3035_T_LINE2_write_FIFO( )	To fill FIFO with 2 dimensions’ datum
83.	MPC3035_T_LINE3_write_FIFO( )	To fill FIFO with 3 dimensions’ datum
84.	MPC3035_T_LINE4_write_FIFO( )	To fill FIFO with 4 dimensions’ datum
85.	MPC3035_Run_FIFO_CMD( )	Start to run the motion stored in FIFO
86.	MPC3035_set_FIFO_out_Ratio( )	To override the speed of stored FIFO data
87.	MPC3035_FIFO_EOI( )	End of interrupt of FIFO, to restart the FIFO interrupt function
88.	MPC3035_config_compare_start_motion( )	To configure the compare source and method of synchronous start
89.	MPC3035_set_compare_start_data( )	To configure the compared data
90.	MPC3035_T_curve_wait_Cmpstart( )	To setup the T profile motion and wait for synchronous start signal to take action.
91.	MPC3035_S_curve_wait_Cmpstart( )	To setup the S profile motion and wait for synchronous start signal to take action.
92.	MPC3035_T_LINE2_wait_Cmpstart( )	To setup the T profile 2 axes linear motion and wait for synchronous start signal to take action.
93.	MPC3035_S_LINE2_wait_Cmpstart( )	To setup the S profile 2 axes linear motion and wait for synchronous start signal to take action.
94.	MPC3035_T_LINE3_wait_Cmpstart( )	To setup the T profile 3 axes linear motion and wait for synchronous start signal to take action.
95.	MPC3035_S_LINE3_wait_Cmpstart( )	To setup the S profile 3 axes linear motion and wait for synchronous start signal to take action.
96.	MPC3035_T_LINE4_wait_Cmpstart( )	To setup the T profile 4 axes linear motion and wait for synchronous start signal to take action.
97.	MPC3035_S_LINE4_wait_Cmpstart( )	To setup the S profile 4 axes linear motion and wait for synchronous start signal to take action.
98.	MPC3035_ARC2_center_wait_Cmpstart( )	To setup circular interpolation movement with circle center and end position and wait for synchronous start signal to take action.
99.	MPC3035_ARC2_3P_wait_Cmpstart( )	To setup circular interpolation movement with current point and the other 2 points as the circle trajectory, and wait for synchronous start signal to take action.
100.	MPC3035_read_compare_start_flag( )	To read the compare start flag.
101.	MPC3035_trigger_CSTA_pulse( )	To trigger out the CSTA (START) signal from the assigned card.
102.	MPC3035_trigger_CSTP_pulse( )	To trigger out the CSTP (STOP) signal from the

		assigned card.
103.	MPC3035_T_curve_wait_CSTA( )	To setup the T profile motion and wait for CSTA signal to take action.
104.	MPC3035_T_LINE2_wait_CSTA( )	To setup the T profile 2 axes linear motion and wait for CSTA signal to take action.
105.	MPC3035_T_LINE3_wait_CSTA( )	To setup the T profile 3 axes linear motion and wait for CSTA signal to take action.
106.	MPC3035_T_LINE4_wait_CSTA( )	To setup the T profile 4 axes linear motion and wait for CSTA signal to take action.
107.	MPC3035_S_curve_wait_CSTA( )	To setup the S profile motion and wait for CSTA signal to take action.
108.	MPC3035_S_LINE2_wait_CSTA( )	To setup the S profile 2 axes linear motion and wait for CSTA signal to take action.
109.	MPC3035_S_LINE3_wait_CSTA( )	To setup the S profile 3 axes linear motion and wait for CSTA signal to take action.
110.	MPC3035_S_LINE4_wait_CSTA( )	To setup the S profile 4 axes linear motion and wait for CSTA signal to take action.
111.	MPC3035_ARC2_center_wait_CSTA( )	To setup circular interpolation movement with circle center and end position and wait for CSTA signal to take action.
112.	MPC3035_ARC2_3P_wait_CSTA( )	To setup circular interpolation movement with current point and the other 2 points as the circle trajectory, and wait for CSTA signal to take action.
113.	MPC3035_set_continuous_flag( )	Enable / disable the continuous mode
114.	MPC3035_check_continuous_buffer( )	To check the continuous buffer
115.	MPC3035_read_conti_buffer_no( )	To read how many buffered data left in continuous mode
116.	MPC3035_read_motion_status( )	Read the motion status
117.	MPC3035_OnLine_T_curve_change( )	To change the motion parameters on the fly for single axis.
118.	MPC3035_OnLine_T_curve_change_ LINE2( )	To change the motion parameters on the fly for any 2 axes linear interpolation.
119.	MPC3035_OnLine_T_curve_change_ LINE3( )	To change the motion parameters on the fly for any 3 axes linear interpolation.
120.	MPC3035_OnLine_T_curve_change_ LINE4( )	To change the motion parameters on the fly for 4 axes linear interpolation.
121.	MPC3035_OneAxis_restart( )	To restart the previously halted axis
122.	MPC3035_2Axis_restart( )	To restart the previously halted 2 axes
123.	MPC3035_3Axis_restart( )	To restart the previously halted 3 axes
124.	MPC3035_4Axis_restart( )	To restart the previously halted 4 axes
125.	MPC3035_enable_IRQ( )	To enable the interrupt function.
126.	MPC3035_disable_IRQ( )	To disable the interrupt function, and release the resource and close thread.
127.	MPC3035_link_IRQ_process( )	Link irq service routine to driver
128.	MPC3035_set_INT_source( )	To setup the error/event source that will generate interrupt at error/event occurs.
129.	MPC3035_read_INT_status( )	To read back the status of interrupt event source
130.	MPC3035_set_INT_mask( )	To set the interrupt mask of designated axis.
131.	MPC3035_set_event_factor( )	To enable the event for corresponding event source
132.	MPC3035_read_event_flag( )	To read the event source

133.	MPC3035_read_error_flag( )	To read back the status of error source
134.	MPC3035_config_softlimit( )	Configure soft limit
135.	MPC3035_readback_config_softlimit( )	Read back the software limit parameter
136.	MPC3035_set_softlimit_data( )	Setup the coordinate data of soft limit
137.	MPC3035_readback_softlimit_data( )	Read back the coordinate of software limit
138.	MPC3035_enable_softlimit( )	Enable / disable software limit function
139.	MPC3035_readback_enable_softlimit( )	Read back the status of enable / disable software limit
140.	MPC3035_read_softlimit_flag( )	Read the software limit flag for verifying
141.	MPC3035_set_pulser_Map( )	Map the source (pulse handler) to the target motion axis
142.	MPC3035_enable_pulser_motion( )	Enable pulse handler function and the multiple rate
143.	MPC3035_config_pulser_mode( )	Configure the operating mode of the pulse handler
144.	MPC3035_readback_pulser_mode( )	Read back the pulse handler operation mode
145.	MPC3035_run_pulser_Vmove( )	Operate pulse handler as manual speed control
146.	MPC3035_run_pulser_Pmove( )	Operate pulse handler as manual position control
147.	MPC3035_set_pulser_counter( )	Set pulse counter
148.	MPC3035_read_pulser_counter( )	Read pulse counter
149.	MPC3035_set_pulse_inmode( )	Configure the multiple rate and the encoder input
150.	MPC3035_readback_pulse_inmode( )	Read back configuration of pulse input mode
151.	MPC3035_read_FB_counter( )	Read feedback counter
152.	MPC3035_set_FB_counter( )	Set feedback counter
153.	MPC3035_config_LTC_PIN( )	Configure LTC (latch) input
154.	MPC3035_readback_LTC_PIN( )	Read back configuration of LTC pin
155.	MPC3035_read_FBcounter_latch_value( )	Read feedback counter latched value
156.	MPC3035_config_CMP_OUT( )	Configure CMP (compare) output
157.	MPC3035_readback_CMP_OUT( )	Read back configuration of CMP_OUT
158.	MPC3035_config_comparator_out( )	Configure the compare output mode
159.	MPC3035_readback_comparator_out( )	Read back the configuration of the compare mode
160.	MPC3035_set_comparator_data( )	Preset the value to the comparator
161.	MPC3035_readback_comparator_data( )	Read back the preset comparator value
162.	MPC3035_read_compare_flag( )	Read compare out flag
163.	MPC3035_config_comparator_out_W( )	Configure the window compare output mode
164.	MPC3035_readback_comparator_out( ) _W	Read back the configuration of the window compare mode
165.	MPC3035_set_comparator_data_W( )	Preset the value to the window comparator
166.	MPC3035_readback_comparator_data_W( )	Read back the preset window comparator value
167.	MPC3035_read_compare_flag_W( )	Read window compare out flag
168.	MPC3035_unlock_security( )	Unlock security for further operation
169.	MPC3035_read_security_status( )	Check the current status of security
170.	MPC3035_set_serial_code( )	To setup the serial code
171.	MPC3035_read_serial_code( )	To read the serial code
172.	MPC3035_set_password( )	Set password and start the security function
173.	MPC3035_change_password( )	Change password
174.	MPC3035_clear_password( )	Clear password
175.	MPC3035L_set_input_polarity( )	set MPC3035L card’s input point polarity.
176.	MPC3035L_read_input_polarity( )	read back polarity of input point.

177.	MPC3035L_set_output_polarity( )	set MPC3035L card’s output polarity.
178.	MPC3035L_read_output_polarity( )	read back polarity of output point.
179.	MPC3035L_read_input_status( )	read MPC3035L card’s input status
180.	MPC3035L_set_counter_mode( )	Set MPC3035L card counter mode.
181.	MPC3035L_set_quadrature_times( )	Set MPC3035L card quadrature decoding rate as the counter mode setting in A/B phase quadrature mode.
182.	MPC3035L_set_hard_homing( )	Set hardware homing mode (hardware clear counter).
183.	MPC3035L_read_hard_homing_flag( )	Read MPC3035L card’s hardware homing flag.
184.	MPC3035L_soft_homing_command( )	To clear counter.
185.	MPC3035L_read_counter( )	read MPC3035L card counter. The 32bit counter value will return.
186.	MPC3035L_load_counter( )	load value into MPC3035L card counter.
187.	MPC3035L_latch_control	To enable/disable latch function
188.	MPC3035L_latch_mode	To assign the compare equal trigger input function
189.	MPC3035L_read_latch_flag	To read latch flag, which accused by compare equal trigger
190.	MPC3035L_read_latch_value	Read MPC3035 counter latched value
191.	MPC3035L_write_XY_latch( )	Set XY simutaneous latch enable
192.	MPC3035L_read_XY_latch( )	Read XY simutaneous latch enable flag
193.	MPC3035L_set_CMP_OUT_mode( )	set the CMP_OUT (compare out) mode.
194.	MPC3035L_write_output_command( )	write MPC3035L card’s output point.
195.	MPC3035L_read_output_status( )	read back status of output point
196.	MPC3035L_load_compare_value( )	set the compare value.
197.	MPC3035L_read_compare_value( )	read the compare value.
198.	MPC3035L_set_CMP_method( )	set the compare method of compare output function.
199.	MPC3035L_load_increase_value( )	load the increase value.
200.	MPC3035L_read_increase_value( )	read the incremental data
201.	MPC3035L_clear_FIFO_command( )	clear (reset) FIFO
202.	MPC3035L_fill_FIFO_value( )	fill (load) FIFO.
203.	MPC3035L_read_FIFO_full_flag( )	Read FIFO full flag
204.	MPC3035L_read_FIFO_unused_number( )	Read FIFO unused space (number)
205.	MPC3035L_read_FIFO_empty_flag( )	Read FIFO empty flag
206.	MPC3035L_read_FIFO_value( )	Read data from top of FIFO
207.	MPC3035L_set_CMP_oneshot_duration( )	set the one shot duration time for compare out.
208.	MPC3035L_set_Mask_CMP_OUT_source( )	set the mask source of CMP_OUT
209.	MPC3035L_write_mask_off( )	Write mask off attribute
210.	MPC3035L_read_mask_off( )	Read mask off attribute
211.	MPC3035L_write_cmp_segment	Write segment coordinate
212.	MPC3035L_read_cmp_segment	Read segment coordinate
213.	MPC3035L_write_segment_control	Wrie segment enable/disable
214.	MPC3035L_read_segment_control	Read segment enable/disable
215.	MPC3035L_enable_IRQ( )	To enable interrupt.
216.	MPC3035L_disable_IRQ( )	To disable interrupt.
217.	MPC3035L_set_FIFO_INT_no( )	Set refill FIFO alarm number condition to result in an interrupt

218.	MPC3035L_link_IRQ_process( )	To link the interrupt event with the callback function.
219.	MPC3035L_set_INT_mask( )	To mask off the un-wanted interrupt axis
220.	MPC3035L_set_INT_source( )	To set interrupt source
221.	MPC3035L_read_INT_status( )	To read the interrupt source.
222.	MPC3035L_read_INT_ID( )	To read the interrupt source axis.
223.	MPC3035L_out_PWM_DA( )	Output the value to PWM DA.
224.	MPC3035L_read_parameters( )	To read parameters currently set.

10. MPC-3035 Error codes summary

10.1 MPC3035 Error codes table

Error Code	Symbolic Name	Description
Error in wdm3035.sys		
0	JSDRV_NO_ERROR	Success, No error.
1	JSDRV_READ_DATA_ERROR	
2	JSDRV_INIT_ERROR	Driver initial error
Error in drv3035.dll		
100	DEVICE_RW_ERROR	Device Read/Write error or no card on the system
101	JSDRV_NO_CARD	No MPC3035 card on the system.
102	JSDRV_DUPLICATE_ID	MPC3035 CardID duplicate error.
Error in MPC3035.dll of motion block		
3	JSDRV_UNLOCK_ERROR	Card is locked, must unlock before operation
4	JSDRV_LOCK_COUNTER_ERROR	Unlock with wrong password more than 10 times
5	JSDRV_SET_SECURITY_ERROR	Error during password setting, maybe old password exit
300	JSMPC_ID_ERROR	Function input parameter error. CardID setting error, CardID doesn’t match the DIP/ROTARY switch setting
301	MPC3035_AXIS_MAX_ERROR	Axis parameter error. Parameter out of range.
302	MPC3035_OTHER_PAR_ERROR	Parameter error or out of range.
303	MPC3035_MOTION_BUSY_ERROR	Motion now is busy, no further command can accept
304	MPC3035_CONTINUOUS_FULL_ERROR	In continuous mode, the continuous buffer is full, no further command can accept
305	MPC3035_MOTION_CHANGE_ERROR	Error to use position change in continuous motion mode or motion is already (stop) Fail of change position or speed profile on the fly
306	MPC3035_MOTION_SYNCHROUS_ERROR	Error during interpolation mode, while any of the action axis is error
308	MPC3035_ARC3P_OVERWRITE2_LINE	It is not possible to use the desinated 3 point to locate a circle and force to a line
309	MPC3035_READ_FILE_ ERROR	File parameter does not exist or not correct while load or save configuration parameters

Error in MPC3035.dll of encoder counter block		
1301	JSLSI_COUNTER_MODE_ERROR	MPC3035L_set_counter_mode(), "mode" parameter invalid.
1302	JSLSI_QUADRATURE_TIMES_ERROR	MPC3035L_set_quadrature_times(),"time" parameter invalid.
1305	JSLSI_POINT_ERROR	"point" parameter out of range.
1306	JSLSI_AXIS_ERROR	"axis" parameter out of range.
1308	JSLSI_HOME_MODE_ERROR	MPC3035L_set_hard_homing(),"mode" parameter invalid.
1309	JSLSI_POLARITY_ERROR	"polarity" parameter out of range.
1310	JSLSI_ON_OFF_ERROR	"on_off" parameter out of range.
1312	JSLSI_COMPARE_OUT_MODE_ERROR	setting of compare out mode error.
1313	JSLSI_FIFO_FULL_ERROR	push into new data while FIFO full.
1314	JSLSI_FIFO_EMPTY_ERROR	pop out data while FIFO empty.
1315	JSLSI_OTHER_PAR_ERROR	Other un-specified paramater error