_7443_version_info:

Let users readback version infomation.

@ **Syntax**

C/C++ (DOS, Windows 95/NT)

```
I16 _7443_initial(I16 *existCards);
I16 _7443_close(void);
I16 _7443_get_irq_channel(I16 cardNo, U16 *irq_no );
I16 _7443_get_base_addr(I16 cardNo, U16 *base_addr );
I16 _7443_delay_time(I16 AxisNo,Double MiniSec);
I16 _7443_config_from_file(char *filename);
I16 _7443_version_info(I16 CardNo, U16 *HardwareInfo, U16
 *SoftwareInfo, U16 *DriverInfo);
```

Visual Basic (Windows 95/NT)

```
B_7443_initial (existCards As Integer) As Integer
B_7443_close () As Integer
B_7443_get_irq_channel (ByVal CardNo As Integer, irq_no As Integer) As
        Integer
B_7443_get_base_addr (ByVal CardNo As Integer, base_addr As Integer)
        As Integer
B_7443_delay_time (ByVal AxisNo As Integer, ByVal MiniSec As Double)
        As Integer
B_7443_config_from_file(ByVal filename As string)as integer
B_7443_version_info (ByVal CardNo As Integer, HardwareInfo As Integer,
        SoftwareInfo As Integer, DriverInfo As Integer) As Integer
```

@ **Argument**

***existCards**: numbers of existing PPCI7443 cards
cardNo: The PPCI7443 card index number.
***irq_no**: Irq number of specified PPCI7443 card.
***base_addr**: base address of specified PPCI7443 card
***Filename**: The specified filename recording the configuration of PPCI7443. This file must be created by PPCI7443 Utility.
AxisNo: axis number designated to move or stop.
MiniSec: Delay time(unit of ms)
***Hardwareinfo**: Hardware version readback
***SoftwareInfo**: Software library version readback
***DriverInfo**: Device driver version readback

@ **Return Code**

ERR_NoErroe
ERR_NoCardFound
ERR_PCIBiosNotExist
ERR_ConigFileOpenError

6.4 Pulse Input/Output Configuration

@ Name

_7443_set_pls_outmode – Set the configuration for pulse command output.
_7443_set_pls_iptmode – Set the configuration for feedback pulse input.
_7443_set_feedback_src – Enable/Disable the external feedback pulse input

@ Description

_7443_set_pls_outmode:

Configure the output modes of command pulse. There are 6 modes for command pulse output.

_7443_set_pls_iptmode:

Configure the input modes of external feedback pulse. There are four types for feedback pulse input. Note that this function makes sense only when **Src** parameter in **_7443_set_feedback_src ()** function is enabled.

_7443_set_feedback_src:

If external encoder feedback is available in the system, set the **Src** parameter in this function to *Enabled* state. Then internal 28-bit up/down counter will count according configuration of **_*7443*_set_pls_iptmode()** function. Or the counter will count the command pulse output.

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _7443_set_pls_outmode(I16 AxisNo, I16 pls_outmode);
I16 _7443_set_pls_iptmode(I16 AxisNo, I16 pls_iptmode, I16 pls_logic);
I16 _7443_set_feedback_src(I16 AxisNo, I16 Src);
```

Visual Basic (Windows 95/NT)

```
B_7443_set_pls_outmode (ByVal AxisNo As Integer, ByVal pls_outmode
        As Integer) As Integer
B_7443_set_pls_iptmode (ByVal AxisNo As Integer, ByVal pls_iptmode As
        Integer, ByVal pls_logic As Integer) As Integer
B_7443_set_feedback_src (ByVal AxisNo As Integer, ByVal Src As Integer)
        As Integer
```

@ **Argument**

AxisNo: axis number designated to configure pulse Input/Output.

pls_outmode: setting of command pulse output mode

Value	Meaning	
0	OUT/DIR	OUT Falling edge, DIR+ is high level
1	OUT/DIR	OUT Rising edge, DIR+ is high level
2	OUT/DIR	OUT Falling edge, DIR+ is low level
3	OUT/DIR	OUT Rising edge, DIR+ is low level
4	CW/CCW	Falling edge
5	CW/CCW	Rising edge

pls_iptmode: setting of encoder feedback pulse input mode

Value	Meaning
0	1X A/B
1	2X A/B
2	4X A/B
3	CW/CCW

pls_logic: Logic of encoder feedback pulse
pls_logic=0, Normal low.
pls_logic=1, Normal high

Src: Counter source

Value	Meaning
0	External Feedback
1	Command pulse

@ **Return Code**

ERR_NoError

6.5 Velocity mode motion

@ Name

_7443_tv_move – Accelerate an axis to a constant velocity with trapezoidal profile
_7443_sv_move – Accelerate an axis to a constant velocity with S-curve profile
_7443_v_change –Change speed on the fly
_7443_sd_stop –Decelerate to stop
_7443_emg_stop –Immediately stop
_7443_fix_speed_range – Define the speed range
_7443_unfix_speed_range – Release the speed range constrain
_7443_get_current_speed – Get current speed
_7443_verify_speed – get speed profile's minimum and maximum acc/dec time

@ Description

_7443_tv_move:

This function is to accelerate an axis to the specified constant velocity with trapezoidal profile. The axis will continue to travel at a constant velocity until the velocity is changed or the axis is commanded to stop. The direction is determined by the sign of velocity parameter.

_7443_sv_move:

This function is to accelerate an axis to the specified constant velocity with S-curve profile. The axis will continue to travel at a constant velocity until the velocity is changed or the axis is commanded to stop. The direction is determined by the sign of velocity parameter.

_7443_v_change:

This function changes the moving velocity with trapezoidal profile or S-curve profile. Before calling this function, it is necessary to define the speed range by _7443_fix_speed_range. _7443_v_change is also applicable on pre-set motion. Note: The velocity profile is decided by original motion profile. When using in S-curve, please set the motion to be pure S-curve. There are some limitations for this function, please refer to section 4.6.1 before use it.

_7443_sd_stop:

This function is used to decelerate an axis to stop with trapezoidal profile or S-curve profile. This function is also useful when ***preset move*** (both trapezoidal and S-curve motion), ***manual move*** or ***home return*** function is performed. Note: The velocity profile is decided by original motion profile.

_7443_emg_stop:

This function is used to immediately stop an axis. This function is also useful when ***preset move*** (both trapezoidal and S-curve motion), ***manual move*** or ***home return*** function is performed.

_7443_fix_speed_range

This function is used to define the speed range. It should be called

before starting motion that may contains velocity changing.

_7443_unfix_speed_range

This function is used to Release the speed range constrain.

_7443_get_current_speed

This function is used to read current pulse output rate of specified axis. It is applicable in any time and any operating mode.

_7443_verify_speed

Find a speed profile's minimum and maximum acc/time time.

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _7443_tv_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _7443_sv_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc, F64
      SVacc);
I16 _7443_v_change(I16 AxisNo, F64 NewVel, F64 Tacc);
I16 _7443_sd_stop(I16 AxisNo,F64 Tdec);
I16 _7443_emg_stop(I16 AxisNo);
F64 _7443_fix_speed_range(I16 AxisNo, F64 MaxVel);
I16 _7443_unfix_speed_range(I16 AxisNo);
I16 _7443_get_current_speed(I16 AxisNo, F64 *speed);
F64 _7443_verify_speed(F64 StrVel,F64 MaxVel,F64 *minAccT,F64
      *maxAccT, F64 MaxSpeed);
```

Visual Basic (Windows 95/NT)

```
B_7443_tv_move (ByVal AxisNo As Integer, ByVal StrVel As Double, ByVal
      MaxVel As Double, ByVal Tacc As Double) As Integer
B_7443_sv_move (ByVal AxisNo As Integer, ByVal StrVel As Double,
      ByVal MaxVel As Double, ByVal Tacc As Double, ByVal SVacc As
      Double) As Integer
B_7443_v_change (ByVal AxisNo As Integer, ByVal NewVel As Double,
      ByVal TimeSecond As Double) As Integer
B_7443_sd_stop (ByVal AxisNo As Integer, ByVal Tdec As Double) As
      Integer
B_7443_emg_stop (ByVal AxisNo As Integer) As Integer
B_7443_fix_speed_range (ByVal AxisNo As Integer, ByVal MaxVel As
      Double) As Integer
B_7443_unfix_speed_range (ByVal AxisNo As Integer) As Integer
B_7443_get_current_speed (ByVal AxisNo As Integer, Speed As Double)
      As Integer
B_7443_verify_speed  (ByVal StrVel As Double, ByVal MaxVel As Double,
      minAccT As Double, maxAccT As Double, ByVal MaxSpeed As
      Double) As Double
```

@ **Argument**

AxisNo: axis number designated to move or stop.
StrVel: starting velocity in unit of pulse per second
MaxVel: maximum velocity in unit of pulse per second
Tacc: specified acceleration time in unit of second
SVacc: specified velocity interval in which S-curve acceleration is performed.
Note: SVacc = 0, for pure S-curve
NewVel: New velocity in unit of pulse per second
Tdec: specified deceleration time in unit of second
***Speed:** Variable to save current speed.
(speed range: 0~6553500)
minAccT: Minimum acceleration time .
maxAcct: Maximum acceleration time
MaxSpeed: The speed set by Fix_Speed

@ **Return Code**

ERR_NoError
ERR_SpeedError
ERR_SpeedChangeError
ERR_SlowDownPointError
ERR_AxisAlreadyStop

6.6 Single Axis Position Mode

@ Name

_7443_start_tr_move – Begin a relative trapezoidal profile move
_7443_start_ta_move – Begin an absolute trapezoidal profile move
_7443_start_sr_move – Begin a relative S-curve profile move
_7443_start_sa_move – Begin an absolute S-curve profile move
_7443_set_move_ratio –Set the ratio of command pulse and feedback pulse.
_7443_p_change – Change position on the fly
_7443_set_pcs_logic –Set the logic of PCS (Position Change Signal) pin
_7443_set_sd_pin –Set SD/PCS pin
_7443_backlash_comp – Set backlash compensating pulse for compensation
_7443_suppress_vibration – Set vibration suppressing timing
_7443_set_idle_pulse – Set suppress vibration idle pulse counts

@ Description

General: *The moving direction is determined by the sign of* ***Pos*** *or* ***Dist*** *parameter. If the moving distance is too short to reach the specified velocity, the controller will automatically lower the MaxVel ,and the Tacc, Tdec, SVacc, SVdec, will also become shorter while the dV/dt(acceleration / deceleration) and d(dV/dt)/dt (jerk) keep unchanged.*

_7443_start_tr_move:

This function causes the axis to accelerate from a starting velocity, slew at constant velocity, and decelerate to stop at the relative distance with trapezoidal profile. The acceleration and deceleration time is specified independently. It won't let the program wait for motion completion but immediately return control to the program.

_7443_start_ta_move :

This function causes the axis to accelerate from a starting velocity, slew at constant velocity, and decelerate to stop at the specified absolute position with trapezoidal profile. The acceleration and deceleration time is specified independently. It won't let the program wait for motion completion but immediately return control to the program.

_7443_start_sr_move:

This function causes the axis to accelerate from a starting velocity, slew at constant velocity, and decelerate to stop at the relative distance with S-curve profile. The acceleration and deceleration time is specified independently. It won't let the program wait for motion completion but immediately return control to the program.

_7443_start_sa_move :

This function causes the axis to accelerate from a starting velocity, slew at constant velocity, and decelerate to stop at the specified absolute position with S-curve profile. The acceleration and deceleration time is specified independently. It won't let the program wait for motion completion but immediately return control to the program.

_7443_set_move_ratio :

This function configures scale factors for the specified axis. Usually, the axes only need scale factors if their mechanical resolutions are different. For example, if the resolution of feedback sensors is two times resolution of command pulse, then ***ratio = 2***.

_7443_p_change

This function is used to change target position on the fly. There are some limitations for this function. Please refer to section 4.6.2 before use it.

_7443_set_pcs_logic :

This function is used to set the logic of Position Change Signal (pcs). The PCS share the same pin with SD signal. Only when the SD/PCS pin was set to PCS by _7443_set_sd_pin, this _7443_set_pcs_logic function becomes effective.

_7443_set_sd_pin :

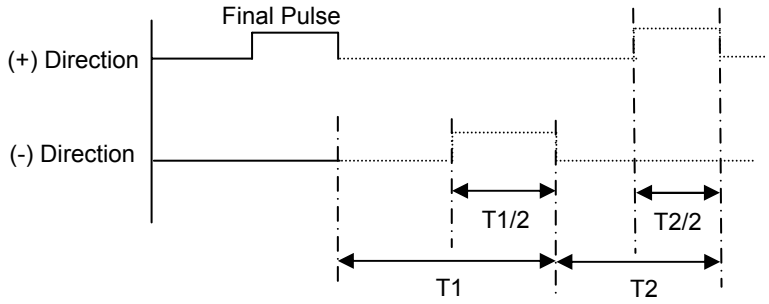
This function is used to set the operating mode of SD pin. The SD pin may be used either as Slow-Down signal input or as Position Change Signal (PCS) input. Please refer to section 4.3.1

_7443_backlash_comp :

Whenever direction change is occurred, The PPCI7443 outputs backlash corrective pulses before sending commands. This function is used set the compensation pulse numbers.

_7443_suppress_vibration

This function is used to suppress vibration of mechanical system by outputting single pulse for negative direction and then single pulse for positive direction right after completion of command movement.



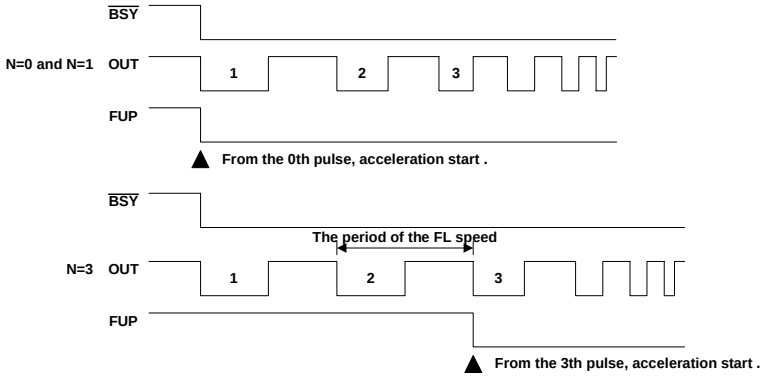
_7443_set_idle_pulse :

The idling pulse to control the vibration of the machine is set up. Acceleration is made to start after number pulse idling is outputted at a start speed when a movement starts.

Attention :

Set up 2 - 7 in the setup value when you use this function.

Set up 0 or 1 when you don't use it.



@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _7443_start_tr_move(I16 AxisNo, F64 Dist, F64 StrVel, F64 MaxVel,
        F64 Tacc,F64 Tdec);
I16 _7443_start_ta_move(I16 AxisNo, F64 Pos, F64 StrVel, F64 MaxVel,
        F64 Tacc, F64 Tdec);
I16 _7443_start_sr_move(I16 AxisNo, F64 Dist, F64 StrVel, F64 MaxVel,
        F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _7443_start_sa_move(I16 AxisNo, F64 Pos, F64 StrVel, F64 MaxVel,
        F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _7443_set_move_ratio(I16 AxisNo, F64 move_ratio);
I16 _7443_p_change(I16 AxisNo, F64 NewPos);
I16 _7443_set_pcs_logic(I16 AxisNo, I16 pcs_logic);
I16 _7443_set_sd_pin(I16 AxisNo, I16 Type);
I16 _7443_backlash_comp(I16 AxisNo, I16 BCompPulse);
I16 _7443_suppress_vibration(I16 AxisNo, U16 T1, U16 T2);
I16 _7443_set_idle_pulse(I16 AxisNo, I16 idl_pulse);
```

Visual Basic (Windows 95/NT)

B_7443_start_tr_move (ByVal AxisNo As Integer, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_ta_move (ByVal AxisNo As Integer, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_sr_move (ByVal AxisNo As Integer, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_start_sa_move (ByVal AxisNo As Integer, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_set_move_ratio (ByVal AxisNo As Integer, ByVal move_ratio As Double) As Integer

B_7443_p_change (ByVal AxisNo As Integer, ByVal NewPos As Double) As Integer

B_7443_set_pcs_logic (ByVal AxisNo As Integer, ByVal pcs_logic As Integer) As Integer

B_7443_set_sd_pin (ByVal AxisNo As Integer, ByVal Type As Integer) As Integer

B_7443_backlash_comp (ByVal AxisNo As Integer, ByVal BCompPulse As Integer, ByVal ForwardTime As Integer) As Integer

B_7443_suppress_vibration (ByVal AxisNo As Integer, ByVal ReserveTime As Integer, ByVal ForwardTime As Integer) As Integer

B_7443_set_idle_pulse(ByVal AxisNo As Integer, ByVal idl_pulse As Integer);

@ **Argument**

AxisNo: axis number designated to move or change position.
Dist: specified relative distance to move
Pos: specified absolute position to move
StrVel: starting velocity of a velocity profile in unit of pulse per second
MaxVel: maximum velocity of a velocity profile in unit of pulse per second
Tacc: specified acceleration time in unit of second
Tdec: specified deceleration time in unit of second
SVacc: specified velocity interval in which S-curve acceleration is performed.
Note: SVacc = 0, for pure S-curve
SVdec: specified velocity interval in which S-curve deceleration is performed.
Note: SVdec = 0, for pure S-curve
Move_ratio: ratio of (feedback resolution)/(command resolution) , should not be 0
NewPos: specified new absolute position to move
pcs_logic: Specify he pcs logic.
Value = 0: low active ,
Value = 1: high active
Type: define the SD pin usage
Value = 0 : SD pin as SD signal
Value = 1: SD pin as PCS signal
BcompPulse: Specified number of corrective pulses
T1: Specified Reverse Time
T2: Specified Forward Time
Idl_pulse: Idl_pulse=0~7

@ **Return Code**

ERR_NoError
ERR_SpeedError
ERR_PChangeSlowDownPointError
ERR_MoveRatioError

6.7 Linear Interpolated Motion

@ Name
_7443_start_tr_move_xy – Begin a relative 2-axis linear interpolation for X & Y, with trapezoidal profile,
_7443_start_ta_move_xy – Begin a absolute 2-axis linear interpolation for X & Y, with trapezoidal profile,
_7443_start_sr_move_xy – Begin a relative 2-axis linear interpolation for X & Y, with S-curve profile,
_7443_start_sa_move_xy – Begin a absolute 2-axis linear interpolation for X & Y, with S-curve profile,
_7443_start_tr_move_zu – Begin a relative 2-axis linear interpolation for Z & U, with trapezoidal profile,
_7443_start_ta_move_zu – Begin a absolute 2-axis linear interpolation for Z & U, with trapezoidal profile,
_7443_start_sr_move_zu – Begin a relative 2-axis linear interpolation for Z & U, with S-curve profile,
_7443_start_sa_move_zu – Begin a absolute 2-axis linear interpolation for Z & U, with S-curve profile,
_7443_start_tr_line2 – Begin a relative 2-axis linear interpolation for any 2 axes, with trapezoidal profile,
_7443_start_sr_line2 – Begin a relative 2-axis linear interpolation for any 2 axes,, with S-curve profile
_7443_start_ta_line2 – Begin a absolute 2-axis linear interpolation for any 2 axes,, with trapezoidal profile
_7443_start_sa_line2 – Begin a absolute 2-axis linear interpolation for any 2 axes,, with S-curve profile,
_7443_start_tr_line3 – Begin a relative 3-axis linear interpolation with trapezoidal profile,
_7443_start_sr_line3 – Begin a relative 3-axis linear interpolation with S-curve profile
_7443_start_ta_line3 – Begin a absolute 3-axis linear interpolation with trapezoidal profile
_7443_start_sa_line3 – Begin a absolute 3-axis linear interpolation with S-curve profile,
_7443_start_tr_line4 – Begin a relative 4-axis linear interpolation with trapezoidal profile,
_7443_start_sr_line4 – Begin a relative 4-axis linear interpolation with S-curve profile
_7443_start_ta_line4 – Begin a absolute 4-axis linear interpolation with trapezoidal profile
_7443_start_sa_line4 – Begin a absolute 4-axis linear interpolation with S-curve profile,
_7443_set_line_move_mode – Set continuous line interpolation mode
_7443_set_axis_option – Choose the interpolation speed mode

@ **Description**

Functions	No. of interpolating axes	Speed Profile	Relative/Absolute	Target Axes
_7443_start_tr_move_xy	2	Trapezoidal	R	Axis 0 & 1
_7443_start_ta_move_xy	2	Trapezoidal	A	Axis 0 & 1
_7443_start_sr_move_xy	2	S-curve	R	Axis 0 & 1
_7443_start_sa_move_xy	2	S-curve	A	Axis 0 & 1
_7443_start_tr_move_zu	2	Trapezoidal	R	Axis 2 & 3
_7443_start_ta_move_zu	2	Trapezoidal	A	Axis 2 & 3
_7443_start_sr_move_zu	2	S-curve	R	Axis 2 & 3
_7443_start_sa_move_zu	2	S-curve	A	Axis 2 & 3
_7443_start_tr_line2	2	Trapezoidal	R	Any 2 of 4
_7443_start_ta_ line2	2	Trapezoidal	A	Any 2 of 4
_7443_start_sr_ line2	2	S-curve	R	Any 2 of 4
_7443_start_sa_ line2	2	S-curve	A	Any 2 of 4
_7443_start_tr_ line3	3	Trapezoidal	R	Any 3 of 4
_7443_start_ta_ line3	3	Trapezoidal	A	Any 3 of 4
_7443_start_sr_ line3	3	S-curve	R	Any 3 of 4
_7443_start_sa_ line3	3	S-curve	A	Any 3 of 4
_7443_start_tr_ line4	4	Trapezoidal	R	Any 4 of 4
_7443_start_ta_ line4	4	Trapezoidal	A	Any 4 of 4
_7443_start_sr_ line4	4	S-curve	R	Any 4 of 4
_7443_start_sa_ line4	4	S-curve	A	Any 4 of 4

_7443_set_line_move_tmode :

There is continuous interpolation mode in the line interpolation of 4 axes from 2 axes. An interpolation movement is continued until a position of a stop is not a distance but a stop command is written when this mode is set up in the continuance. It becomes the continuous interpolation mode when the mode parameter of the _7443_set_line_mode() command is set up in "1".It becomes the positioning interpolation mode when "0" is set up.

@ **Syntax**

C/C++ (DOS, Windows 95/NT)

I16 _7443_start_tr_move_xy(I16 CardNo, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);

I16 _7443_start_ta_move_xy(I16 CardNo, F64 PosX, F64 PosY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);

I16 _7443_start_sr_move_xy(I16 CardNo, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

I16 _7443_start_sa_move_xy(I16 CardNo, F64 PosX, F64 PosY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

I16 _7443_start_tr_move_zu(I16 CardNo, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _7443_start_ta_move_zu(I16 CardNo, F64 PosX, F64 PosY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _7443_start_sr_move_zu(I16 CardNo, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _7443_start_sa_move_zu(I16 CardNo, F64 PosX, F64 PosY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _7443_start_tr_line2(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _7443_start_ta_line2(I16 CardNo, I16 *AxisArray, F64 PosX, F64 PosY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _7443_start_sr_line2(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _7443_start_sa_line2(I16 CardNo, I16 *AxisArray, F64 PosX, F64 PosY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _7443_start_tr_line3(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY, F64 DistZ, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _7443_start_ta_line3(I16 CardNo, I16 *AxisArray, F64 PosX, F64 PosY, F64 PosZ, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _7443_start_sr_line3(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY, F64 DistZ, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _7443_start_sa_line3(I16 CardNo, I16 *AxisArray, F64 PosX, F64 PosY, F64 PosZ, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _7443_start_tr_line4(I16 CardNo, F64 DistX, F64 DistY, F64 DistZ, F64 DistU, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _7443_start_ta_line4(I16 CardNo, F64 PosX, F64 PosY, F64 PosZ, F64 PosU, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _7443_start_sr_line4(I16 CardNo, F64 DistX, F64 DistY, F64 DistZ, F64 DistU, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _7443_start_sa_line4(I16 CardNo, F64 PosX, F64 PosY, F64 PosZ, F64 PosU, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _7443_set_line_move_mode(I16 AxisNo, I16 Mode);
I16 _7443_set_axis_option(I16 AxisNo, I16 option);

Visual Basic (Windows 95/NT)

B_7443_start_tr_move_xy (ByVal CardNo As Integer, ByVal Dist As Double, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer
B_7443_start_ta_move_xy (ByVal CardNo As Integer, ByVal Pos As Double, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer
B_7443_start_sr_move_xy (ByVal CardNo As Integer, ByVal Dist As Double, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_start_sa_move_xy (ByVal CardNo As Integer, ByVal Pos As Double, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_start_tr_move_zu (ByVal CardNo As Integer, ByVal Dist As Double, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_ta_move_zu (ByVal CardNo As Integer, ByVal Pos As Double, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_sr_move_zu (ByVal CardNo As Integer, ByVal Dist As Double, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_start_sa_move_zu (ByVal CardNo As Integer, ByVal Pos As Double, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_start_tr_line2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_ta_line2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal PosX As Double, ByVal PosY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_sr_line2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_start_sa_line2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal PosX As Double, ByVal PosY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_start_tr_line3 (ByVal CardNo As Integer, AxisArray As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal DistZ As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_ta_line3 (ByVal CardNo As Integer, AxisArray As Integer, ByVal PosX As Double, ByVal PosY As Double, ByVal PosZ As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_sr_line3 (ByVal CardNo As Integer, AxisArray As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal DistZ As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_start_sa_line3 (ByVal CardNo As Integer, AxisArray As Integer, ByVal PosX As Double, ByVal PosY As Double, ByVal PosZ As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_start_tr_line4 (ByVal CardNo As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal DistZ As Double, ByVal DistU As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_ta_line4 (ByVal CardNo As Integer, ByVal PosX As Double, ByVal PosY As Double, ByVal PosZ As Double, ByVal PosU As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_sr_line4 (ByVal CardNo As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal DistZ As Double, ByVal DistU As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_start_sa_line4 (ByVal CardNo As Integer, ByVal PosX As Double, ByVal PosY As Double, ByVal PosZ As Double, ByVal PosU As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_set_line_move_mode (ByVal AxisNo As Integer, ByVal Mode As Integer) As Integer

B_7443_set_axis_option (ByVal AxisNo As Integer, ByVal option1 As Integer) As Integer

@ **Argument**

CardNo: Card number designated to perform linear interpolation
DistX: specified relative distance of axis 0 to move
DistY: specified relative distance of axis 1 to move
DistZ: specified relative distance of axis 2 to move
DistU: specified relative distance of axis 3 to move
PosX: specified absolute position of axis 0 to move
PosY: specified absolute position of axis 1 to move
PosZ: specified absolute position of axis 2 to move
PosU: specified absolute position of axis 3 to move
StrVel: starting velocity of a velocity profile in unit of pulse per second
MaxVel: starting velocity of a velocity profile in unit of pulse per second
Tacc: specified acceleration time in unit of second
Tdec: specified deceleration time in unit of second
SVacc: specified velocity interval in which S-curve acceleration is performed.
Note: SVacc = 0, for pure S-curve
SVdec: specified velocity interval in which S-curve deceleration is performed.
Note: SVdec = 0, for pure S-curve
AxisArray: Array of axis number to perform interpolation.
Example: Int AxisArray[2] = {0,2}; // axis 0 & 2
Int AxisArray[3] = {0,1,3}; // axis 0,1,3
Note: AxisArray[n] must be smaller than AxisArray[m], if n<m.
Mode: InterPoration mode
Mode = 0 : positioning line interpolation mode
Mode = 1 : Continuous line interpolation mode
Option1: 0=default line move mode
1=Composite speed constant mode

@ **Return Code**

ERR_NoError
ERR_SpeedError
ERR_AxisArrayErrot

6.8 Circular Interpolation Motion

@ Name

_7443_start_r_arc_xy – Begin a relative circular interpolation for X & Y
_7443_start_a_arc_xy – Begin a absolute circular interpolation for X & Y
_7443_start_r_arc_zu – Begin a relative circular interpolation for Z & U
_7443_start_a_arc_zu – Begin a absolute circular interpolation for Z & U
_7443_start_r_arc2 – Begin a relative circular interpolation for any 2 axes
_7443_start_a_arc2 – Begin a absolute circular interpolation for any 2 axes

_7443_start_tr_arc_xyu – Begin a Trapezoidal relative circular interpolation
_7443_start_ta_arc_xyu – Begin a Trapezoidal absolute circular interpolation
_7443_start_sr_arc_xyu – Begin a S-curve relative circular interpolation
_7443_start_sa_arc_xyu – Begin a S-curve absolute circular interpolation
_7443_start_tr_arc_yzu – Begin a Trapezoidal relative circular interpolation
_7443_start_ta_arc_yzu – Begin a Trapezoidal absolute circular interpolation
_7443_start_sr_arc_yzu – Begin a S-curve relative circular interpolation
_7443_start_sa_arc_yzu – Begin a Trapezoidal absolute circular interpolation

_7443_start_tr_arc2 – Begin a Trapezoidal relative circular interpolation
_7443_start_ta_arc2 – Begin a Trapezoidal absolute circular interpolation
_7443_start_sr_arc2– Begin a S-curve relative circular interpolation
_7443_start_sa_arc2– Begin a S-curve absolute circular interpolation
_7443_start_tr_arc_xy – Begin a Trapezoidal relative circular interpolation
_7443_start_ta_arc_xy – Begin a Trapezoidal absolute circular interpolation
_7443_start_tr_arc_zu – Begin a Trapezoidal relative circular interpolation
_7443_start_ta_arc_zu – Begin a Trapezoidal absolute circular interpolation
_7443_start_sr_arc_xy – Begin a S-curve relative circular interpolation
_7443_start_sa_arc_xy – Begin a S-curve absolute circular interpolation
_7443_start_sr_arc_zu – Begin a S-curve relative circular interpolation
_7443_start_sa_arc_zu – Begin a S-curve absolute circular interpolation

@ Description

Function	Relative/ Absolute	Speed Profile	Target Axes	Hardware version bit 12
_7443_start_r_arc_xy	R	Flat	Axis 0 & 1	0 or 1
_7443_start_a_arc_xy	A	Flat	Axis 0 & 1	0 or 1
_7443_start_r_arc_zu	R	Flat	Axis 2 & 3	0 or 1
_7443_start_a_arc_zu	A	Flat	Axis 2 & 3	0 or 1
_7443_start_r_arc2	R	Flat	Any 2 of 4	0 or 1
_7443_start_a_arc2	A	Flat	Any 2 of 4	0 or 1
_7443_start_tr_arc_xyu	R	Trapezoidal	Axis 0 & 1	0 or 1
_7443_start_ta_arc_xyu	A	Trapezoidal	Axis 0 & 1	0 or 1
_7443_start_sr_arc_xyu	R	S-curve	Axis 1 & 2	0 or 1
_7443_start_sa_arc_xyu	A	S-curve	Axis 1 & 2	0 or 1
_7443_start_tr_arc_xy	R	Trapezoidal	Axis 0 & 1	1
_7443_start_ta_arc_xy	A	Trapezoidal	Axis 0 & 1	1

_7443_start_sr_arc_xy	R	S-curve	Axis 0 & 1	1
_7443_start_sa_arc_xy	A	S-curve	Axis 0 & 1	1
_7443_start_tr_arc_zu	R	Trapezoidal	Axis 2 & 3	1
_7443_start_ta_arc_zu	A	Trapezoidal	Axis 2 & 3	1
_7443_start_sr_arc_zu	R	S-curve	Axis 2 & 3	1
_7443_start_sa_arc_zu	A	S-curve	Axis 2 & 3	1
_7443_start_tr_arc2	R	Trapezoidal	Any 2 of 4	1
_7443_start_ta_arc2	A	Trapezoidal	Any 2 of 4	1
_7443_start_sr_arc2	R	S-curve	Any 2 of 4	1
_7443_start_sa_arc2	A	S-curve	Any 2 of 4	1

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _7443_start_r_arc_xy(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64
      OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
I16 _7443_start_a_arc_xy(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey,
      I16 DIR, F64 MaxVel);
I16 _7443_start_r_arc_zu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64
      OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
I16 _7443_start_a_arc_zu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey,
      I16 DIR, F64 MaxVel);
I16 _7443_start_r_arc2(I16 CardNo, I16 *AxisArray, F64 OffsetCx, F64
      OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
I16 _7443_start_a_arc2(I16 CardNo, I16 *AxisArray, F64 Cx, F64 Cy, F64
      Ex, F64 Ey, I16 DIR, F64 MaxVel);

I16 _7443_start_tr_arc_xyu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64
      OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel, F64 MaxVel, F64
      Tacc);
I16 _7443_start_ta_arc_xyu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey,
      I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _7443_start_sr_arc_xyu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64
      OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc,
      F64 SVacc);
I16 _7443_start_sa_arc_xyu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey,
      I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc, F64 SVacc);
I16 _7443_start_tr_arc_yzu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64
      OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel, F64 MaxVel, F64
      Tacc);
I16 _7443_start_ta_arc_yzu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey,
      I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _7443_start_sr_arc_yzu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64
      OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc,
      F64 SVacc);
I16 _7443_start_sa_arc_yzu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey,
      I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc, F64 SVacc);
```

I16 _7443_start_tr_arc2(I16 CardNo, I16 *AxisArray, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel,F64 MaxVel, F64 Tacc,F64 Tdec);
I16 _7443_start_ta_arc2(I16 CardNo, I16 *AxisArray, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc,F64 Tdec);
I16 _7443_start_sr_arc2(I16 CardNo, I16 *AxisArray, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel,F64 MaxVel, F64 Tacc,F64 Tdec,F64 SVacc,F64 SVdec);
I16 _7443_start_sa_arc2(I16 CardNo, I16 *AxisArray, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc,F64 Tdec,F64 SVacc,F64 SVdec);

I16 _7443_start_tr_arc_xy(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec);
I16 _7443_start_ta_arc_xy(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec);
I16 _7443_start_tr_arc_zu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec);
I16 _7443_start_ta_arc_zu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec);

I16 _7443_start_sr_arc_xy(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel,F64 MaxVel, F64 Tacc,F64 Tdec,F64 SVacc,F64 SVdec);
I16 _7443_start_sa_arc_xy(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel,F64 MaxVel, F64 Tacc,F64 Tdec,F64 SVacc,F64 SVdec);
I16 _7443_start_sr_arc_zu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel,F64 MaxVel, F64 Tacc,F64 Tdec,F64 SVacc,F64 SVdec);
I16 _7443_start_sa_arc_zu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel,F64 MaxVel, F64 Tacc,F64 Tdec,F64 SVacc,F64 SVdec);

Visual Basic (Windows 95/NT)

B_7443_start_a_arc_xy (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer
B_7443_start_r_arc_xy (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer
B_7443_start_a_arc_zu (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer
B_7443_start_r_arc_zu (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer

B_7443_start_a_arc2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer

B_7443_start_r_arc2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer

B_7443_start_tr_arc_xyu (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Integer

B_7443_start_ta_arc_xyu (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Integer

B_7443_start_sr_arc_xyu (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Integer

B_7443_start_sa_arc_xyu (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal SVacc As Double) As Integer

B_7443_start_tr_arc_yzu (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Integer

B_7443_start_ta_arc_yzu (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Integer

B_7443_start_sr_arc_yzu (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Integer

B_7443_start_sa_arc_yzu (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal SVacc As Double) As Integer

B_7443_start_tr_arc2 (ByVal CardNo As Integer, AxisArray As Double, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_ta_arc2 (ByVal CardNo As Integer, AxisArray As Double, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_sr_arc2 (ByVal CardNo As Integer, AxisArray As Double, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_start_sa_arc2 (ByVal CardNo As Integer, AxisArray As Double, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_start_tr_arc_xy (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_ta_arc_xy (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_tr_arc_zu (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_ta_arc_zu (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_7443_start_sr_arc_xy (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_start_sa_arc_xy (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_start_sr_arc_zu (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_7443_start_sa_arc_zu (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

@ **Argument**

CardNo: Card number designated to perform linear interpolation
OffsetCx: X-axis offset to center
OffsetCy: Y-axis offset to center
OffsetEx: X-axis offset to end of arc
OffsetEy: Y-axis offset to end of arc
Cx: specified X-axis absolute position of center
Cy: specified Y-axis absolute position of center
Ex: specified X-axis absolute position end of arc
Ey: specified Y-axis absolute position end of arc
DIR: Specified direction of arc, CW:0, CCW:1
StrVel: starting velocity of a velocity profile in unit of pulse per second
MaxVel: Tangential velocity in unit of pulse per second
Tacc: specified acceleration time in unit of second
Tdec: specified deceleration time in unit of second
SVacc: specified velocity interval in which S-curve acceleration is performed.
Note: SVacc = 0, for pure S-curve
SVdec: specified velocity interval in which S-curve deceleration is performed.
Note: SVdec = 0, for pure S-curve
AxisArray: Array of axis number to perform interpolation.
Example: Int AxisArray[2] = {0,2}; // axis 0 & 2
Int AxisArray[2] = {1,3}; // axis 1 & 3
Note: AxisArray[0] must be smaller than AxisArray[1]

@ **Return Code**

ERR_NoError
ERR_SpeedError
ERR_AxisArrayErrot

6.9 Home Return Mode

@ Name

_7443_set_home_config – Set the configuration for home return.
_7443_home_move – Perform a home return move.
_7443_escape_home – Escape Home Function
_7443_home_search –Auto-Search Home Switch (Without ORGOffset)
_7443_auto_home_search –Auto-Search Home Switch (With ORGOffset)

@ Description

_7443_set_home_config:

Configure the home return mode, origin & index signal(EZ) logic, EZ count and ERC output options for home_move() function. Refer to Section 4.1.8 for the setting of home_mode control.

_7443_home_move:

This function will cause the axis to perform a home return move according to the setting of **_7443_set_home_config()** function. The direction of moving is determined by the sign of velocity parameter(svel, mvel). Since the stopping condition of this function is determined by *home_mode* setting, user should take care to select the initial moving direction. Or user should take care to handle the condition when limit switch is touched or other conditions that is possible causing the axis to stop. Executing v**_stop()** function during **home_move()** can also cause the axis to stop.

_7443_escape_home:

After homing, use this function to leave home switch

_7443_home_search:

Auto-Search Home Switch.

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _7443_set_home_config(I16 AxisNo, I16 home_mode, I16 org_logic,
        I16 ez_logic, I16 ez_count, I16 erc_out);
I16 _7443_home_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _7443_escape_home(I16 AxisNo, F64 SrVel,F64 MaxVel,F64 Tacc);
I16 _7443_home_search(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _7443_auto_home_search(I16 AxisNo, F64 StrVel, F64 MaxVel, F64
        Tacc, F64 ORGOffset);
```

Visual Basic (Windows 95/NT)

```
B_7443_set_home_config (ByVal AxisNo As Integer, ByVal home_mode As
        Integer, ByVal org_logic As Integer, ByVal ez_logic As Integer,
        ByVal ez_count As Integer, ByVal erc_out As Integer) As Integer
B_7443_home_move (ByVal AxisNo As Integer, ByVal StrVel As Double,
        ByVal MaxVel As Double, ByVal Tacc As Double) As Integer
B_7443_escape_home(ByVal AxisNo As Integer, ByVal SrVel As Double,
        ByVal MaxVel As Double, ByVal Tacc As Double) As Integer
B_7443_home_search(ByVal AxisNo As Integer, ByVal StrVel As Double,
        ByVal MaxVel As Double, ByVal Tacc As Double)As Integer
```

B_7443_auto_home_search(ByVal AxisNo As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal ORGOffset As Double)As Integer

@ **Argument**

AxisNo: axis number designated to configure and perform home returning
home_mode: stopping modes for home return, 0~12
(Please refer to section 4.1.8)
org_logic: Action logic configuration for ORG signal
org_logic=0, active low;
org_logic=1, active high
EZ_logic: Action logic configuration for EZ signal
EZ_logic=0, active low;
EZ_logic=1, active high.
ez_count: 0~15 (Please refer to section 4.1.8)
erc_out: Set ERC output options.
erc_out =0, no erc out;
erc_out =1, erc out when homing finish
StrVel: starting velocity of a velocity profile in unit of pulse per second
MaxVel: starting velocity of a velocity profile in unit of pulse per second
Tacc: specified acceleration time in unit of second
ORGOffset: The escape pulse amounts when home search touching the ORG singal

@ **Return Code**

ERR_NoError

6.10 Manual Pulser Motion

@ Name

_7443_set_pulser_iptmode - set the input signal modes of pulser
_7443_pulser_vmove – manual pulser v_move
_7443_pulser_pmove – manual pulser p_moce
_7443_pulser_home_move – manual pulser home move
_7443_set_pulser_ratio –Set manual pulser ratio for actual output pulse rate.
_7443_pulser_r_line2 –Pulser mode for 2-axis linear interpolation
_7443_pulser_r_arc2 –Pulser mode for 2-axis arc interpolation

@ Description

_7443_set_pulser_iptmode:

This function is used to configure the input mode of manual pulser.

_7443_pulser_vmove:

As this command is written, the axis begins to move the axis according to manual pulser input. The axis will output one pulse when receive one pulse from pulser, until the ***sd_stop*** or ***emg_stop*** command is written.

_7443_pulser_pmove:

As this command is written, the axis begins to move the axis according to manual pulser input. The axis will output one pulse when receive one pulse from pulser, until the ***sd_stop*** or ***emg_stop*** command is written or the output pulse number reach dist.

_7443_pulser_home_move:

As this command is written, the axis begins to move the axis according to manual pulser input. The axis will output one pulse when receive one pulse from pulser, until the ***sd_stop*** or ***emg_stop*** command is written or the home move finish.

_7443_set_pulser_ratio:

Set manual pulser ratio for actual output pulse rate. The formula for pulser output rate is

Output Pulse Speed=(PA_PB Speed) * 4 * (PMG+1)*PDV/2048
The PDV=0~10 Divide Factor
The PMG=0~4 Multi Factor

_7443_set_pulser_ratio:

Pulser mode for 2-axis linear interpolation (relative mode only).

_7443_pulser_r_arc2:

Pulser mode for 2-axis arc interpolation (relative mode only)

@ **Syntax**

C/C++ (DOS, Windows 95/NT)

```
I16 _7443_set_pulser_iptmode(I16 AxisNo,I16 InputMode, I16 Inverse);
I16 _7443_pulser_vmove(I16 AxisNo, F64 SpeedLimit);
I16 _7443_pulser_pmove(I16 AxisNo, F64 Dist, F64 SpeedLimit);
I16 _7443_pulser_home_move(I16 AxisNo, I16 HomeType, F64
    SpeedLimit);
I16 _7443_set_pulser_ratio(I16 AxisNo,I16 PDV, I16 PMG);
I16 _7443_pulser_r_line2(I16 CardNo,I16 *AxisArray, F64 DistX, F64 DistY,
    F64 SpeedLimit);
I16 _7443_pulser_r_arc2(I16 CardNo, I16 *AxisArray, F64 OffsetCx, F64
    OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
```

Visual Basic (Windows 95/NT)

```
B_7443_set_pulser_iptmode (ByVal AxisNo As Integer, ByVal InputMode
    As Integer, ByVal Inverse As Integer) As Integer
B_7443_pulser_vmove (ByVal AxisNo As Integer, ByVal SpeedLimit As
    Double) As Integer
B_7443_pulser_pmove (ByVal AxisNo As Integer, ByVal Dist As Double,
    ByVal SpeedLimit As Double) As Integer
B_7443_pulser_home_move (ByVal AxisNo As Integer, ByVal HomeType
    As Integer, ByVal SpeedLimit As Double) As Integer
B_7443_set_pulser_ratio(ByVal AxisNo As Integer, ByVal PDV As Integer,
    ByVal PMG As Integer) As Integer
B_7443_pulser_r_line2(ByVal CardNo As Integer, AxisArray As Integer,
    ByVal DistX As Double, ByVal DistY As Double, ByVal SpeedLimit
    As Double) As Integer
B_7443_pulser_r_arc2(ByVal CardNo As Integer, AxisArray As Integer,
    ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal
    OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As
    Integer, ByVal MaxVel As Double) As Integer
```

@ **Argument**

AxisNo: axis number designated to start manual move
InputMode: setting of manual pulser input mode from PA and PB pins
ipt_mode=0, 1X AB phase type pulse input.
ipt_mode=1, 2X AB phase type pulse input.
ipt_mode=2, 4X AB phase type pulse input.
ipt_mode=3, CW/CCW type pulse input.
Inverse: Reverse the moving direction from pulse direction
Inverse =0, no inverse
Inverse =1, Reverse moving direction
SpeedLimit: The maximum speed in pulser move.
For example, if SpeedLimit is set to be 100 pps, then the axis can move at fastest 100 pps , even the input pulser signal rate is more then 100 pps.
Dist: specified relative distance to move
HomeType: specified home move type
HomeType =0, Command Origin.(that means axis stops when command counter becomes '0')
HomeType =1, ORG pin.
PDV, PMG: Divide and Multi Factor.
PDV=0~10 Divide Factor
PMG=0~4 Multi Factor
The Output Pulse Speed=(PA_PB Speed) * 4 * (PMG+1)*PDV/2048
DistX: specified relative distance of axis 0 to move
DistY: specified relative distance of axis 1 to move
OffsetCx: X-axis offset to center
OffsetCy: Y-axis offset to center
OffsetEx: X-axis offset to end of arc
OffsetEy: Y-axis offset to end of arc
DIR: Specified direction of arc, CW:0 , CCW:1
SpeedLimit: Maximum tangential velocity in unit of pulse per second
MaxVel: Maximum tangential velocity in unit of pulse per second

@ **Return Code**

ERR_NoError
ERR_PulserHomeTypeError

6.11 Motion Status

@ Name
_7443_motion_done – Return the motion status

@ Description
_7443_motion_done:
Return the motion status of PPCI7443.

@ Syntax

C/C++ (DOS, Windows 95/NT)
I16 _7443_motion_done(I16 AxisNo);

Visual Basic (Windows 95/NT)
B_7443_motion_done (ByVal AxisNo As Integer) As Integer

@ Argument
AxisNo: axis number designated to start manual move

@ Return Value

0	Stop
1	Reserved
2	Reserved
3	Reserved
4	Wait other axis
5	Wait ERC finished
6	Wait DIR Change
7	Backlash compensating
8	Wait PA/PB
9	In home special speed motion
10	In start velocity motion
11	In acceleration
12	In Max velocity motion
13	In deceleration
14	Wait INP
15	Other axis us still moving

6.12 Motion Interface I/O

@ Name

_7443_set_alm – Set alarm logic and operating mode
_7443_set_el – Set EL logic and operating mode
_7443_set_inp– Set Inp logic and operating mode
_7443_set_erc– Set ERC logic and timing
_7443_set_servo – Set state of general purpose output pin
_7443_set_sd – Set SD logic and operating mode

@ Description

_7443_set_alm_logic:

Set the active logic of **ALARM** signal input from servo driver. Two reacting modes are available when **ALARM** signal is active.

_7443_set_el:

Set the reacting modes of **EL** signal.

_7443_set_inp_logic:

Set the active logic of **In-Position** signal input from servo driver. Users can select whether they want to enable this function. Default state is disabled.

_7443_set_erc:

You can set the logic and on time of ERC by this function.

_7443_set_servo:

You can set the ON-OFF state of SVON signal by this function. The default value is 1(OFF), which means the SVON is open to GND.

_7443_set_sd_logic:

Set the active logic, latch control and operating mode of **SD** signal input from mechanical system. Users can select whether they want to enable this function. Default state is disabled.

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _7443_set_alm(I16 AxisNo, I16 alm_logic, I16 alm_mode);
I16 _7443_set_el(I16 AxisNo, I16 el_mode);
I16 _7443_set_inp(I16 AxisNo, I16 inp_enable, I16 inp_logic);
I16 _7443_set_erc(I16 AxisNo, I16 erc_logic, I16 erc_on_time);
I16 _7443_set_servo(I16 AxisNo, I16 on_off);
I16 _7443_set_sd(I16 AxisNo, I16 enable, I16 sd_logic, I16 sd_latch, I16
    sd_mode);
```

Visual Basic (Windows 95/NT)

```
B_7443_set_alm (ByVal AxisNo As Integer, ByVal alm_logic As Integer,
    ByVal alm_mode As Integer) As Integer
B_7443_set_el (ByVal AxisNo As Integer, ByVal el_mode As Integer) As
    Integer
B_7443_set_inp (ByVal AxisNo As Integer, ByVal inp_enable As Integer,
    ByVal inp_logic As Integer) As Integer
B_7443_set_erc (ByVal AxisNo As Integer, ByVal erc_logic As Integer,
    ByVal erc_on_time As Integer) As Integer
B_7443_set_servo (ByVal AxisNo As Integer, ByVal On_Off As Integer) As
    Integer
```

B_7443_set_sd (ByVal AxisNo As Integer, ByVal enable As Integer, ByVal sd_logic As Integer, ByVal sd_latch As Integer, ByVal sd_mode As Integer) As Integer

@ **Argument**

AxisNo: axis number designated to configure
alm_logic: setting of active logic for ALARM signal
alm_logic=0, active LOW.
alm_logic=1, active HIGH.
alm_mode: reacting modes when receiving ALARM signal.
alm_mode=0, motor immediately stops(Default)
alm_mode=1, motor decelerates then stops.
el_mode: reacting modes when receiving EL signal.
el_mode=0, motor immediately stops.(Default)
el_mode=1, motor decelerates then stops.
inp_enable: INP function enable/disable
inp_enable=0, Disabled (Default)
inp_enable=1, Enabled
inp_logic: setting of active logic for INP signal
inp_logic=0, active LOW.
inp_logic=1, active HIGH.
erc_logic: setting of active logic for ERC signal
erc_logic=0, active LOW.
erc_logic=1, active HIGH.
erc_on_time: Setting of time length of ERC active

erc_on_time=0	12us
erc_on_time=1	102us
erc_on_time=2	409us
erc_on_time=3	1.6ms
erc_on_time=4	13ms
erc_on_time=5	52ms
erc_on_time=6	104ms

on_off: ON-OFF state of SVON signal
on_off = 0 , ON
on_off = 1 , OFF
enable: Enable/disable the SD signal.
enable=0, Disabled (Default)
enable=1, Enabled
sd_logic: setting of active logic for SD signal
sd_logic=0, active LOW.
sd_logic=1, active HIGH.
sd_latch: setting of latch control for SD signal
sd_latch=0, do not latch.
sd_latch=1, latch.
sd_mode: setting the reacting mode of SD signal
sd_mode=0, slow down only
sd_mode=1, slow down then stop

@ **Return Code**

ERR_NoError

6.13 Motion I/O Monitoring

@ Name

_7443_get_io_status –Get all the motion I/O status of PPCI7443

@ Description

_7443_get_io_status:

Get all the I/O status for each axis. The definition for each bit is as following:

Bit	Name	Description
0	RDY	RDY pin input
1	ALM	Alarm Signal
2	+EL	Positive Limit Switch
3	-EL	Negative Limit Switch
4	ORG	Origin Switch
5	DIR	DIR output
6	Reserved	
7	PCS	PCS signal input
8	ERC	ERC pin output
9	EZ	Index signal
10	Reserved	
11	Latch	Latch signal input
12	SD	Slow Down signal input
13	INP	In-Position signal input
14	SVON	Servo-ON output status

@ Syntax

C/C++ (DOS, Windows 95/98/NT)

I16 _7443_get_io_status(I16 AxisNo, U16 *io_sts);

Visual Basic (Windows 95/NT)

B_7443_get_io_status (ByVal AxisNo As Integer, io_sts As Integer) As Integer

@ Argument

AxisNo: axis number for I/O control and monitoring

***io_status**: I/O status word. Where “1’ is ON and “0” is OFF. ON/OFF state is read based on the corresponding set logic.

@ Return Code

ERR_NoError

6.14 Interrupt Control

@ Name
_7443_int_control – Enable/Disable INT service
_7443_set_int_factor – Set INT factor
_7443_int_enable – Enable event (For Window only)
_7443_int_disable – Disable event (For Window only)
_7443_get_int_status – Get INT Status (For Window only)
_7443_link_interrupt – Set link to interrupt call back function (For Window only)
_7443_get_int_type – Get INT type (For DOS only)
_7443_enter_isr – Enter interrupt service routine (For DOS only)
_7443_leave_isr – Leave interrupt service routine (For DOS only)
_7443_get_event_int – Get event status (For DOS only)
_7443_get_error_int – Get error status (For DOS only)
_7443_get_irq_status – Get IRQ status (For DOS only)
_7443_not_my_irq – Not My IRQ (For DOS only)
_7443_isr0~9, a, b – Interrupt service routine (For DOS only)
_7443_set_axis_stop_int – enable axis stop int
_7443_mask_axis_stop_int – mask axis stop int

@ Description

_7443_int_control:

This function is used to enable interrupt generating to host PC.

_7443_set_int_factor:

This function allows users to select factors to initiate the event int. The error can never be masked once the interrupt service is turn on by _7443_int_control().

The int status of PPCI7443 is composed of two independent parts: **error_int_status** and **event_int_status**. The event_*int*_status recodes the motion and comparator event under **normal operation**, and this kind of INT status can be masked by **_7443_set_int_factor()**. The error_int_status is for abnormal stop of PPCI7443, for example: EL, ALM …etc. This kind of INT cannot be masked. The following is the definition of these two int_status. By setting the relative bit as 1, PPCI7443 can generate INT signal to host PC.

Bit	Description
0	Normal Stop
1	Next command continued
2	Command pre-register 2 is empty
3	(Reserved)
4	Acceleration Start
5	Acceleration End
6	Deceleration Start
7	Deceleration End
8	(Reserved)
9	(Reserved)
10	(Reserved)
11	General Comparator compared
12	Compared triggered for axis 0,1
13	(Reserved)
14	Latched for axis2,3
15	ORG on
16	SD on
17	(Reserved)
18	(Reserved)
19	CSTA, Sync. Start on
20~31	(Reserved)

_7443_int_enable : (For Window only.)

This function is used to assign the window INT event.

_7443_int_disable: (For Window only.)

This function is used to disable the window INT event.

_7443_get_int_status: (For Window only.)

This function allows user to identify what cause the interrupt signal. After user gets this value, the status register will be cleared to 0. The return value is two 32 bits unsigned integers. The first one is for error_int_status, which is not able to mask by _7443_set_int_factor(). The definition for bit of error_int_status is as following:

error_int_status : can not be masked	
Bit	Interrupt Factor
0	+SL Stop
1	-SL Stop
2	(Reserved)
3	General Comparator Stop
4	(Reserved)
5	+EL
6	-EL
7	ALM
8	(Reserved)
9	(Reserved)
10	SD on then stop
11	(Reserved)
12	Interpolation Error and stop
13	Other Axis stop on Interpolation
14	Pulser input buffer overflow and stop
15	Interpolation counter overflow
16	Encoder input signal error
17	Pulser input signal error
18~31	(Reserved)

The second is for event_int_status, which can be masked by _7443_set_int_factor(). The definition for bit of event_int_status is as following:

event_int_status : can be masked by function call ***_7443_int_*factor()**	
Bit	Description
0	Normal Stop
1	Next command continued
2	Continuous pre-register is empty and allow users to fill new command
3	(Reserved)
4	Acceleration Start
5	Acceleration End
6	Deceleration Start
7	Deceleration End
8	(Reserved)
9	(Reserved)
10	Step-losing occur
11	General Comparator compared
12	Compared triggered for axis 0,1
13	(Reserved)
14	Latched for axis2,3

15	ORG on
16	SD on
17	(Reserved)
18	(Reserved)
19	CSTA, Sync. Start on
17~31	(Reserved)

_7443_link_interrupt: (For **Window** only.)

This function is used to link interrupt call back function.

_7443_get_int_type: **(** This function is for **DOS** only **)**

This function is used to detect which kind of INT occurred.

_7443_enter_isr: **(** This function is for **DOS** only **)**

This function is used to inform system that process is now entering interrupt service routine.

_7443_leave_isr: **(This function is for DOS only)**

This function is used to inform system that process is now leaving interrupt service routine.

_7443_get_event_int: **(** This function is for **DOS** only **)**

This function is used to get event_int_status.

_7443_get_error_int: **(** This function is for **DOS** only **)**

This function is used to get error_int_status.

_7443_get_irq_status: **(** This function is for **DOS** only **)**

This function allows user to confirm if the designated card generates the INT signal to host PC.

_7443_not_my_irq: **(** This function is for **DOS** only **)**

This function must be called after knowing not the designated card generates the INT signal to host PC.

_7443_isr0, _7443_isr1, _7443_isr2, _7443_isr3, ….. _7443_isr9,
_7443_isra, _7443_isrb: **(** Theses function is for **DOS** only **)**

Individual Interrupt service routine for card 0~11.

_7443_set_axis_stop_int

This function will enable an axis stop interrupt factor. Once it is enabled, the interrupt will happen no matter it is normal stop or error stop. This interrupt condition can be turned on or off accompanied every motion command by setting _7443_mask_axis_stop_int(). This kind of interrupt condition is different from _7443_set_int_factor(). It can be controlled in each motion command, very useful in continuous motion when users need only final command has interrupt.

_7443_mask_axis_stop_int

This function will affect axis stop interrupt factor which is set by _7443_set_axis_stop_int().

@ Syntax

C/C++ (DOS)

```
I16 _7443_int_control(U16 cardNo, U16 intFlag );
I16 _7443_set_int_factor(I16 AxisNo, U32 int_factor );
I16 _7443_get_int_type(I16 AxisNo, U16 *int_type);
I16 _7443_enter_isr(I16 AxisNo);
I16 _7443_leave_isr(I16 AxisNo);
I16 _7443_get_event_int(I16 AxisNo, U32 *event_int);
I16 _7443_get_error_int(I16 AxisNo, U32 *error_int);
I16 _7443_get_irq_status(U16 cardNo, U16 *sts);
I16 _7443_not_my_irq(I16 CardNo);
void interrupt _7443_isr0 (void);
void interrupt _7443_isr1 (void);
void interrupt _7443_isr2 (void);
void interrupt _7443_isr3 (void);
void interrupt _7443_isr4 (void);
void interrupt _7443_isr5 (void);
void interrupt _7443_isr6 (void);
void interrupt _7443_isr7 (void);
void interrupt _7443_isr8 (void);
void interrupt _7443_isr9 (void);
void interrupt _7443_isra (void);
void interrupt _7443_isrb (void);
```

C/C++ (Windows 95/98/NT)

```
I16 _7443_int_control(U16 cardNo, U16 intFlag );
I16 _7443_set_int_factor(I16 AxisNo, U32 int_factor );
I16 _7443_int_enable(I16 CardNo, HANDLE *phEvent);
I16 _7443_int_disable(I16 CardNo);
I16 _7443_get_int_status(I16 AxisNo, U32 *error_int_status, U32
      *event_int_status );
I16 _7443_link_interrupt(I16 CardNo, void ( __stdcall *callbackAddr)(I16
      IntAxisNoInCard));
I16 _7443_set_axis_stop_int(I16 AxisNo, I16 axis_stop_int);
I16 _7443_mask_axis_stop_int(I16 AxisNo, I16 int_disable);
```

Visual Basic (Windows 95/NT)

```
B_7443_int_control (ByVal CardNo As Integer, ByVal intFlag As Integer) As
      Integer
B_7443_set_int_factor (ByVal AxisNo As Integer, ByVal int_factor As Long)
      As Integer
B_7443_int_enable (ByVal CardNo As Integer, phEvent As Long) As
      Integer
B_7443_int_disable (ByVal CardNo As Integer) As Integer
B_7443_get_int_status (ByVal AxisNo As Integer, error_int_status As Long,
      event_int_status As Long) As Integer
B_7443_link_interrupt (ByVal CardNo As Integer, ByVal lpCallBackProc As
      Long) As Integer
B_7443_mask_axis_stop_int (ByVal AxisNo As Integer, ByVal int_disable
      As Integer) As Integer
B_7443_set_axis_stop_int (ByVal AxisNo As Integer, ByVal axis_stop_int
      As Integer) As Integer
```

@ **Argument**

cardNo: card number 0,1,2,3…
AxisNo: axis number 0,1,2,3,4…
intFlag: int flag, 0 or 1 (0: Disable, 1:Enable)
int_factor: interrupt factor, refer to previous table
***int_type:** Interrupt type, (1: error int, 2: event int, 3: both happened)
***event_int:** event_int_status, , refer to previous table
***error_int:** error_int_status, refer to previous table
***sts:** (0: not this card's IRQ, 1: this card's IRQ)
***phEvent:** event handler (Windows)
***error_int_status**: refer to previous table
***event_int_status:** refer to previous table
int_disable: (0:make axis stop interrupt active, 1:make axis stop interrupt in-active)
axis_stop_int: (0: disable axis stop interrupt factor, 1: enable axis stop interrupt factor)

@ **Return Code**

ERR_NoError
ERR_EventNotEnableYet
ERR_LinkIntError
ERR_CardNoErrot

6.15 Position Control and Counters

@ Name

_7443_get_position – Get the value of feedback position counter
_7443_set_position – Set the feedback position counter
_7443_get_command – Get the value of command position counter
_7443_set_command – Set the command position counter
_7443_get_error_counter – Get the value of position error counter
_7443_reset_error_counter – Reset the position error counter
_7443_get_general_counter – Get the value of general counter
_7443_set_general_counter – Set the general counter
_7443_get_target_pos – Get the value of target position recorder
_7443_reset_target_pos – Reset target position recorder
_7443_get_rest_command – Get remaining pulse till end of motion
_7443_check_rdp – Get the ramping down point data

@ Description

_7443_get_position():

This function is used to read the value of feedback position counter. Note, this value has already been processed by move ratio. If move ratio is 0.5, than the value read will be twice as the counter value. The source of feedback counter is selectable by function **_7443_set_feedback_src()** to be external EA/EB or pulse output of PPCI7443.

_7443_set_position():

This function is used to change the feedback position counter to the specified value. Note, the value to be set will be processed by move ratio. If move ratio is 0.5, than the set value will be twice as given value.

_7443_get_command():

This function is used to read the value of command position counter. The source of command position counter is the pulse output of PPCI7443.

_7443_set_command():

This function is used to change the value of command position counter.

_7443_get_error_counter():

This function is used to read the value of position error counter.

_7443_reset_error_counter():

This function is used to clear position error counter.

_7443_get_general_counter():

This function is used to read the value of general counter.

_7443_set_general_counter():

This function is used to set the counting source of and change the value of general counter. (By default, the source is pulser input.)

_7443_get_target_pos():

This function is used to read the value of target position recorder. The target position recorder is maintained by PPCI7443 software driver. It records the position to settle down for current running motion.

_7443_reset_target_pos():

This function is used to set new value for target position recorder. It is necessary to call this function when home return completion or when new feedback counter value is set by function _7443_set_position().

_7443_get_rest_command():

This function is used to read remaining pulse counts till end of current motion.

_7443_check_rdp():

This function is used to read the ramping down point data. The ramping down point is the position where deceleration starts. The data is stored as pulse count, and it cause the axis start to decelerate when remaining pulse count reach the data.

@ Syntax

C/C++ (DOS, Windows 95/98/NT)

```
I16 _7443_get_position(I16 AxisNo, F64 *pos);
I16 _7443_set_position(I16 AxisNo, F64 pos);
I16 _7443_get_command(I16 AxisNo, I32 *cmd);
I16 _7443_set_command(I16 AxisNo, I32 cmd);
I16 _7443_get_error_counter(I16 AxisNo, I16 *error_counter);
I16 _7443_reset_error_counter(I16 AxisNo);
I16 _7443_get_general_counter(I16 AxisNo, F64 *CntValue);
I16 _7443_set_general_counter(I16 AxisNo,I16 CntSrc, F64 CntValue);
I16 _7443_get_target_pos(I16 AxisNo, F64 *T_pos);
I16 _7443_reset_target_pos(I16 AxisNo, F64 T_pos);
I16 _7443_get_rest_command(I16 AxisNo, I32 *rest_command);
I16 _7443_check_rdp(I16 AxisNo, I32 *rdp_command);
```

Visual Basic (Windows 95/NT)

B_7443_get_position (ByVal AxisNo As Integer, Pos As Double) As Integer
B_7443_set_position (ByVal AxisNo As Integer, ByVal Pos As Double) As Integer
B_7443_get_command (ByVal AxisNo As Integer, cmd As Long) As Integer
B_7443_set_command (ByVal AxisNo As Integer, ByVal cmd As Long) As Integer
B_7443_get_error_counter (ByVal AxisNo As Integer, error_counter As Integer) As Integer
B_7443_reset_error_counter (ByVal AxisNo As Integer) As Integer
B_7443_get_general_counter (ByVal AxisNo As Integer, CntValue As Double) As Integer
B_7443_set_general_counter (ByVal AxisNo As Integer, ByVal CntSrc As Integer, ByVal CntValue As Double) As Integer
B_7443_get_target_pos (ByVal AxisNo As Integer, Pos As Double) As Integer
B_7443_reset_target_pos (ByVal AxisNo As Integer, ByVal Pos As Double) As Integer
B_7443_get_rest_command (ByVal AxisNo As Integer, rest_command As Long) As Integer
B_7443_check_rdp (ByVal AxisNo As Integer, rdp_command As Long) As Integer

@ **Argument**

AxisNo: Axis number
Pos, *Pos: Feedback position counter value,
range: -134217728~134217727
cmd, *cmd: Command position counter value,
range: -134217728~134217727
error_counter, *error_counter: Position error counter value,
range: -32768~32767
T_pos, *T_pos: Target position recorder value,
T_ range: -134217728~134217727
CntValue, * CntValue: General counter value,
range: -134217728~134217727
rest_command, *rest_command: Rest pulse count till end,
range: -134217728~134217727
rdp_command, *rdp_command: Ramping down point data
range: 0~167777215
CntSrc: Source of general counter
0 : command
1: EA/EB
2: PA/PB (Default)
3: CLK/2

@ **Return Code**

ERR_NoError
ERR_PosOutofRange

6.16 Position Compare and Latch

@ Name

_7443_set_ltc_logic – Set the LTC logic
_7443_get_latch_data – Get latched counter data
_7443_set_soft_limit – Set soft limit
_7443_enable_soft_limit – Enable soft limit function
_7443_disable_soft_limit – Disable soft limit function
_7443_set_error_counter_check – Step-losing detection setup
_7443_set_general_comparator – Set general-purposed comparator
_7443_set_trigger_comparator – Set trigger comparator
_7443_set_trigger_type – Set the trigger output type
_7443_check_compare_data – Check current comparator data
_7443_check_compare_status – Check current comparator status
_7443_set_auto_compare – Set comparing data source for auto loading
_7443_build_compare_function – Build compare data via constant interval
_7443_build_compare_table – Build compare data via compare table
_7443_cmp_v_change – Speed change by comparator
_7443_set_enable_inp – Set latch signal

@ Description

_7443_set_ltc_logic():

This function is used to set the logic of latch input. This function is applicable only for last two axes in every PPCI7443 card.

_7443_get_latch_data():

After the latch signal arrived, this function is used to read the latched value of counters.

_7443_set_soft_limit():

This function is used to set the value of soft limit.

_7443_enable_soft_limit(),_7443_disable_soft_limit():

These two functions are used to enable/disable the soft limit function. Once enabled, the action of soft limit will be exactly the same as physical limit.

_7443_set_error_counter_check():

This function is used to enable the step losing checking facility. By giving an tolerance value, the PPCI7443 will generate an interrupt (event_int_status , bit 10) when position error counter exceed tolerance.

_7443_set_general_comparator():

This function is used to set the source and comparing value for general comparator. When the source counter value reached the comparing value, the PPCI7443 will generate an interrupt (event_int_status , bit 11).

_7443_set_trigger_comparator():

This function is used to set the comparing method and value for trigger comparator. When the feedback position counter value reached the comparing value, the PPCI7443 will generate trigger a

pulse output via **CMP** and an interrupt (event_int_status , bit 12) will also be sent to host PC. If _7443_set_auto_compare is used, then comparing value set by this function will be ignored automatically. *Note: it is applicable only for first two axes in every PPCI7443 card.*

_7443_set_trigger_type():

This function is used to set the trigger output mode.
In hardware A2 version, it is used for setting the output pulse as one shot or constant on.
In hardware A3 version, it is used for setting the output pulse as normal high or normal low.

_7443_check_compare_data():

This function is used to get current comparing data of designated comparator.

_7443_check_compare_status():

This function is used to get status of all comparator. When some comparator comes into existence, the relative bit of cmp_sts will become '1', otherwise '0'.

_7443_set_auto_compare():

This function is used to set the comparing data source of trigger comparator. The source can be either a function or a table.

_7443_build_compare_function():

This function is used to build comparing function by defining the start / end point and interval. There is no limitation on the max number of comparing data. It will automatically load a final point after user's end point. That is (end point + Interval x total points) x move ratio
Note: Please turn off all interrupt function, when triggering is running.

_7443_build_compare_table():

This function is used to build comparing table by defining data array. The size of array is limited to 1024, when using RAM mode.
Note: Please turn off all interrupt function, when triggering is running.

_7443_cmp_v_change():

This function is used to setup comparator velocity change function. It is in fact a V_change function but acts when general comparator comes into existence. When this function is issued, the parameter "CmpAction" of **_7443_set_general_comparator()** must be set '3'. The compare data is also set by **_7443_set_general_comparator()**. While, the remain distance, the compare point's velocity , the new velocity and the acceleration time are set by **_7443_cmp_v_change()**.

_7443_set_enable_inp():

This function is used to setup latched signal

@ Syntax

C/C++ (DOS, Windows 95/98/NT)

I16 _7443_set_ltc_logic(I16 AxisNo_2or3, I16 ltc_logic);
I16 _7443_get_latch_data(I16 AxisNo, I16 LatchNo, F64 *Pos);
I16 _7443_set_soft_limit(I16 AxisNo, I32 PLimit, I32 NLimit);
I16 _7443_disable_soft_limit(I16 AxisNo);
I16 _7443_enable_soft_limit(I16 AxisNo, I16 Action);
I16 _7443_set_error_counter_check(I16 AxisNo, I16 Tolerance, I16 On_Off);
I16 _7443_set_general_comparator(I16 AxisNo, I16 CmpSrc, I16 CmpMethod, I16 CmpAction, F64 Data);
I16 _7443_set_trigger_comparator(I16 AxisNo, I16 CmpSrc, I16 CmpMethod, F64 Data);
I16 _7443_set_trigger_type(I16 AxisNo, I16 TriggerType);
I16 _7443_check_compare_data(I16 AxisNo, I16 CompType, F64 *Pos);
I16 _7443_check_compare_status(I16 AxisNo, U16 *cmp_sts);
I16 _7443_set_auto_compare(I16 AxisNo ,I16 SelectSrc);
I16 _7443_cmp_v_change(I16 AxisNo, F64 Res_dist, F64 oldvel, F64 newvel, F64 AccTime)
I16 _7443_set_enable_inp(I16 AxisNo, I16 inp_enable);

C/C++ (Windows 95/98/NT)

I16 _7443_build_compare_function(I16 AxisNo, F64 Start, F64 End, F64 Interval, I16 Device);
I16 _7443_build_compare_table(I16 AxisNo, F64 *TableArray, I32 Size, I16 Device);

C/C++ (Dos)

I16 _7443_build_compare_function(I16 AxisNo, F64 Start, F64 End, F64 Interval);
I16 _7443_build_compare_table(I16 AxisNo, F64 *TableArray, I16 Size);

Visual Basic (Windows 95/NT)

B_7443_set_ltc_logic (ByVal AxisNo As Integer, ByVal ltc_logic As Integer) As Integer
B_7443_get_latch_data (ByVal AxisNo As Integer, ByVal Counter As Integer, Pos As Double) As Integer
B_7443_set_soft_limit (ByVal AxisNo As Integer, ByVal PLimit As Long, ByVal NLimit As Long) As Integer
B_7443_disable_soft_limit (ByVal AxisNo As Integer) As Integer
B_7443_enable_soft_limit (ByVal AxisNo As Integer, ByVal Action As Integer) As Integer
B_7443_set_error_counter_check (ByVal AxisNo As Integer, ByVal Tolerance As Integer, ByVal On_Off As Integer) As Integer
B_7443_set_general_comparator (ByVal AxisNo As Integer, ByVal CmpSrc As Integer, ByVal CmpMethod As Integer, ByVal CmpAction As Integer, ByVal Data As Double) As Integer
B_7443_set_trigger_comparator (ByVal AxisNo As Integer, ByVal CmpSrc As Integer, ByVal CmpMethod As Integer, ByVal Data As Double) As Integer
B_7443_set_trigger_type (ByVal AxisNo As Integer, ByVal TriggerType As Integer) As Integer
B_7443_check_compare_data (ByVal AxisNo As Integer, ByVal CompType As Integer, Pos As Double) As Integer

B_7443_check_compare_status (ByVal AxisNo As Integer, cmp_sts As Integer) As Integer
B_7443_set_auto_compare (ByVal AxisNo As Integer, ByVal SelectSrc As Integer) As Integer
B_7443_build_compare_function (ByVal AxisNo As Integer, ByVal Start As Double, ByVal End As Double, ByVal Interval As Double, ByVal Device As Integer) As Integer
B_7443_build_compare_table (ByVal AxisNo As Integer, TableArray As Double, ByVal Size As Long, ByVal Device As Integer) As Integer
B_7443_cmp_v_change(ByVal AxisNo, ByVal Res_dist as Double, ByVal oldvel as Double, ByVal newvel as Double, ByVal AccTime as Double)
B_7443_set_enable_inp (ByVal AxisNo As Integer, ByVal inp_enable As Integer) As Integer

@ **Argument**

AxisNo_2or3: Axis number, for last two axes in one card
Itc_logic: 0 means active low, 1 means active high
AxisNo: Axis number
LatchNo or **Counter:** Specified Counter to latch
Counter = 1 , Command counter
Counter = 2 , Feedback counter
Counter = 3 , Error Counter
Counter = 4 , General Counter
Pos: Latched counter value,
PLimit: Soft limit value in positive direction
NLimit: Soft limit value in negative direction
Action: The reacting method of soft limit
Action =0, INT only
Action =1, Immediately stop
Action =2, slow down then stop
Action =3, reserved
Tolerance: The tolerance of step-losing detection
On_Off: Enable / Disable step-losing detection
On_Off =0, Disable
On_Off =1, Enable
CmpSrc: The comparing source counter
CmpSrc =0, Command Counter
CmpSrc =1, Feedback Counter
CmpSrc =2, Error Counter
CmpSrc =3, General Counter
CmpMethod: The comparing method
CmpMethod =0, No compare
CmpMethod =1, CmpValue=Counter (Directionless)
CmpMethod =2, CmpValue=Counter (+Dir)
CmpMethod =3, CmpValue=Counter (-Dir)
CmpMethod =4, CmpValue>Counter
CmpMethod =5, CmpValue<Counter

CmpAction: The reacting mode when comparison comes into exist
CmpAction =0, INT only
CmpAction =1, Immediately stop
CmpAction =2, slow down then stop
CmpAction =3, speed change
Data: Comparing value,
TriggerType: Selection of type of trigger output mode
Hardware Version A2
TriggerType =0, one shoot (default)
TriggerType =1, constant high
Hardware Version A3
TriggerType =0, normal high (default)
TriggerType =1, normal low
CompType: Selection of type of comparator
CompType =1, + Soft Limit
CompType =2, - Soft Limit
CompType =3, Error Counter Comparator Value
CompType =4, General Comparator Value
CompType =5, Trigger Output Comparator Value
cmp_sts: status of comparator

Bit	Meaning
0	+Softlimit On
1	-SoftLimit On
2	Error counter comparator On
3	General comparator On
4	Trigger comparator On (for 0 , 1 axis only)

SelectSrc: The comparing data source
SelectSrc =0, disable auto compare
SelectSrc =1, use FIFO
Start: Start point of compare function
End: End point of compare function
Interval: Interval of compare function
TableArray: Array of comparing data
Size: Size of table array
Device: Selection of reload device for comparator data
Device =1, FIFO
Res_dist: The remain distance from the compare point. After comparison, the original target position will be ignored, and the axis will keep moving the Res_dist.
oldvel: The velocity at compare point. User must specify it manually.
newvel: The new velocity.
AccTime: The acceleration time.
Inp_enable: Latch source
Inp_enable =0, LTC Pin
Inp_enable =1, ORG Pin
Inp_enable =2, Counter4
Inp_enable =3, Counter5

@ **Return Code**

ERR_NoError
ERR_CompareNoError
ERR_CompareMethodError
ERR_CompareAxisError
ERR_CompareTableSizeError
ERR_CompareFunctionError
ERR_CompareTableNotReady
ERR_CompareLineNotReady
ERR_HardwareCompareAxisWrong
ERR_AutocompareSourceWrong
ERR_CompareDeviceTypeError

6.17 Continuous motion

@ Name

_7443_set_continuous_move – Enable continuous motion
_7443_check_continuous_buffer – check if the buffer is empty

@ Description

_7443_set_continuous_move():

This function is necessary to placed before and after continuous motion.

_7443_check_continuous_buffer():

This function is used to detect if the command pre-register is empty or not. Once the command pre-register is empty, user may write next motion command into it. Otherwise , the new command will overwrite previous in 2nd command pre-register.

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _7443_set_continuous_move(I16 AxisNo, I16 conti_flag);
I16 _7443_check_continuous_buffer(I16 AxisNo);
```

Visual Basic (Windows 95/NT)

```
B_7443_set_continuous_move (ByVal AxisNo As Integer, ByVal conti_flag
        As Integer) As Integer
B_7443_check_continuous_buffer (ByVal AxisNo As Integer) As Integer
```

@ Argument

AxisNo: axis number designated
conti_flag: Flag for continuous motion
conti_flag = 0, one-shoot motion, end of continuous motion
conti_flag = 1, continuous motion, start of continuous motion

@ Return Value

ERR_NoError

Return value of _7443_check_continuous_buffer():
Hardware version bit 12=0
0: Continuous register 2 is empty
1: Continuous register 2 is in-use
Return value of _7443_check_continuous_buffer():
Hardware version bit 12=1
0: all command registers are empty
1: command register is in-use
2: command register 1 is in-use
3: command register 2 is in-use

6.18 Multiple Axes Simultaneous Operation

@ Name

_7443_set_tr_move_all – Multi-axis simultaneous operation setup.
_7443_set_ta_move_all – Multi-axis simultaneous operation setup.
_7443_set_sr_move_all – Multi-axis simultaneous operation setup.
_7443_set_sa_move_all – Multi-axis simultaneous operation setup.
_7443_start_move_all – Begin a multi-axis trapezoidal profile motion
_7443_stop_move_all –Simultaneously stop Multi-axis motion
_7443_set_sync_option – Other sync. motion setting
_7443_set_sync_stop_mode – Setting the stop mode of CSTOP signal

@ Description

Theses functions are related simultaneous operation of multi-axis even in different cards. The simultaneous multi-axis operation means to start or stop moving specified axes at the same time. The move axes are specified by parameter **"AxisArray"** and the number of axes are defined by parameter "***TotalAxes"*** in **_7443_set_tr_move_all()**.

When properly setup with **_7443_set_xx_move_all()**, the function **_7443_start_move_all()** will cause all specified axes to begin trapezoidal relative moving, and _7443_stop_move_all() will stop them. Both functions guarantee that motion Start/Stop on all specified axes at the same time. **Note** that it is necessary to make connections according to Section 3.14 on CN4 if these two functions are needed.

The following code demos how to utilize these functions. This code moves axis 0 and axis 4 to distance 8000.0 and 120000.0 respectively. If we choose velocities and accelerations that are proportional to the ratio of distances, then the axes will arrive at their endpoints at the same time (simultaneous motion).

```
int main()
{
        I16   axes[2] = {0, 4};
        F64  dist[2] = {8000.0, 12000.0},
             str_vel[2]={0.0, 0.0},
             max_vel[2]={4000.0, 6000.0},
             Tacc[2]={0.04, 0.06},
             Tdec[2]= {0.04, 0.06};

_7443_set_tr_move_all(2, axes, dist, str_vel, max_vel, Tacc, Tdec);
_7443_start_move_all(axes[0]);

return   ERR_NoError;
}
```

_7443_set_sync_option()

It has many functions. It lets two or more different command groups start at the same time. For example, if you want a 2-axis linear interpolation and a 1-axis single motion start at the same time, you can turn on this option before command starts. This function also can be used on waiting another command's finish signal then start. For example, axis1 must start after axis2 is done.

_7443_set_sync_stop_mode()

It has two option for stop types: One is immediately stop and the other is slow down to stop. When the _7443_stop_move_all() or CSTOP signal is used, the axes will stop according to this setting.

@ Syntax

C/C++ (DOS, Windows 95/NT)

I16 _7443_set_tr_move_all(I16 TotalAxes, I16 *AxisArray, F64 *DistA, F64 *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA);

I16 _7443_set_sa_move_all(I16 TotalAx, I16 *AxisArray, F64 *PosA, F64 *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA, F64 *SVaccA, F64 *SVdecA);

I16 _7443_set_ta_move_all(I16 TotalAx, I16 *AxisArray, F64 *PosA, F64 *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA);

I16 _7443_set_sr_move_all(I16 TotalAx, I16 *AxisArray, F64 *DistA, F64 *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA, F64 *SVaccA, F64 *SVdecA);

I16 _7443_start_move_all(I16 FirstAxisNo);

I16 _7443_stop_move_all(I16 FirstAxisNo);

I16 _7443_set_sync_option(I16 AxisNo, I16 sync_stop_on, I16 cstop_output_on, I16 sync_option1, I16 sync_option2);

I16 _7443_set_sync_stop_mode(I16 AxisNo, I16 stop_mode);

Visual Basic (Windows 95/NT)

B_7443_set_tr_move_all(ByVal TotalAxes As Integer, AxisArray As Integer, DistA As Double, StrVelA As double, MaxVelA As double, TaccA As double, TdecA As double);

B_7443_set_sa_move_all(ByVal TotalAxes As Integer, AxisArray As Integer, PosA As Double, StrVelA As double, MaxVelA As double, TaccA As double, TdecA As double, SVaccA As double, SVdecA As Double);

B_7443_set_ta_move_all(ByVal TotalAxes As Integer, AxisArray As Integer, PosA As Double, StrVelA As double, MaxVelA As double, TaccA As double, TdecA As double);

B_7443_set_sr_move_all(ByVal TotalAxes As Integer, AxisArray As Integer, DistA As Double, StrVelA As double, MaxVelA As double, TaccA As double, TdecA As double, SVaccA As double, SVdecA As Double);

B_7443_start_move_all(ByVal FirstAxisNo As Integer);

B_7443_stop_move_all(ByVal FirstAxisNo As Integer);

B_7443_set_sync_option (ByVal AxisNo As Integer, ByVal sync_stop_on As Integer, ByVal cstop_output_on As Integer, ByVal sync_option1 As Integer, ByVal sync_option2 As Integer) As Integer

B_7443_set_sync_stop_mode (ByVal AxisNo As Integer, ByVal stop_mode As Integer) As Integer

@ **Argument**

TotalAxes: number of axes for simultaneous motion, 1~48.
*** AxisArray**: specified axes number array designated to move.
*** DistA**: specified position array in unit of pulse
*** StrVelA:** starting velocity array in unit of pulse per second
*** MaxVelA :** maximum velocity array in unit of pulse per second
*** TaccA:** acceleration time array in unit of second
*** TdecA:** deceleration time array in unit of second
*** SVaccA**: specified velocity interval array in which S-curve acceleration is performed.
*** SVdecA**: specified velocity interval array in which S-curve deceleration is performed.
FirstAxisNo : the first element in AxisArray.
Sync_stop_on: Axis will stop if the CSTOP singal is on
Cstop_output_on: CSTOP singal will output when abnormal stop (ALM,EL..etc)
Sync_option1: Choose command start type
0: default (immediately start)
1: waiting _7443_start_move_all() or CSTA singal
2: Reserved
3: Check Sync_option2's condition to start
Sync_option2: for example
0: default (useless)
1: after Axis0 stops
2: after Axis1 stops
4: after Axis2 stops
8: after Axis3 stops
5: after Axis0 and Axis2 stop
15: Axis0~Axis3 stop
stop_mode: 0: immediately stop
1: slow down to stop

@ **Return Code**

ERR_NoError
ERR_SpeedError

6.19 General-purposed TTL output

@ Name

_7443_d_output – Digital Output
_7443_get_dio_status – Get DIO status

@ Description

_7443_d_output():

Set the on_off status for general-purposed TTL Digital output pin.

_7443_get_dio_status():

Read status of all digital output pin.

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _7443_d_output(I16 CardNo, I16 Ch_No, I16 value);
I16 _7443_get_dio_status(I16 CardNo, U16 *dio_sts);
```

Visual Basic (Windows 95/NT)

```
B_7443_d_output (ByVal CardNo As Integer, ByVal Ch_No As Integer,
        ByVal value As Integer) As Integer
B_7443_get_dio_status (ByVal CardNo As Integer, dio_sts As Integer) As
        Integer
```

@ Argument

CardNo: Designated card number
Ch_No: Designated channel number 0~5
Value: On-Off Value for output
Value =0, output OFF
Value =1, output ON
dio_status: Digital output status
bit0~bit5 for channel 0~5 , respectively

@ Return Value

ERR_NoError
ERR_DioNoError

7

Connection Example

This chapter shows some connection examples between PPCI7443 and servo drivers and stepping drivers.

7.1 General Description of Wiring

CN1: Receives +24V power from external power supply.

CN2 :Main connection between PPCI7443 and pulse input servo driver or stepping driver.

CN3: Receive pulse command from manual pulser.

CN4: Connector for simultaneously start or stop multiple PPCI7443 cards.

CN5: TTL digital output.

Figure 7.1 shows how to integrate PPCI7443 with a physical system.

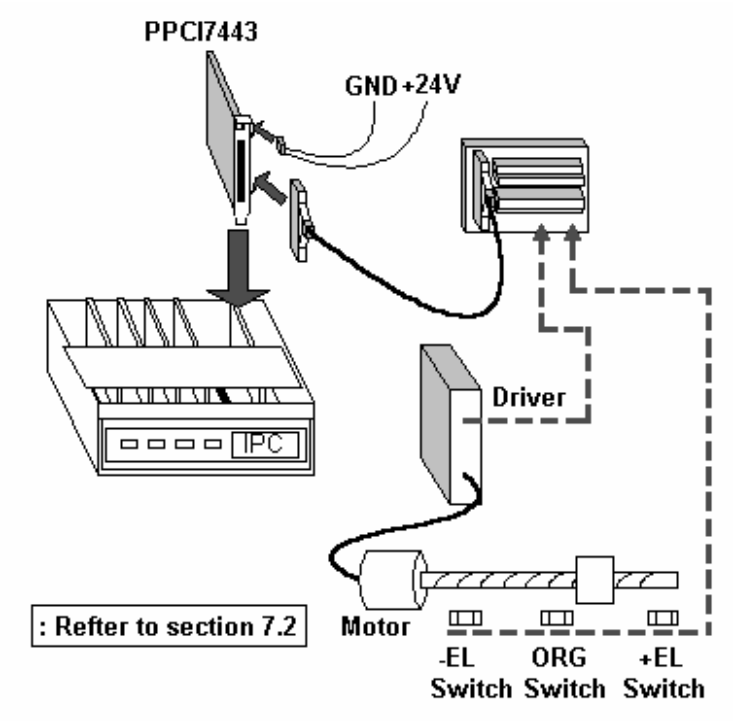


Figure 7.1 System Integration with PPCI7443

7.2 Connection Example with Servo Driver

In this section, we use ***Panasonic Servo Driver*** as an example to show how to connect it with ***PPCI7443***. Figure 7.2 show the wiring.

Note that:

1. For convenience' sake , the drawing shows connections for one axis only.
2. Default pulse output mode is **OUT/DIR** mode; default input mode is 1X **AB phase** mode. Anyway, user can set to other mode by software function.
3. Since most general purpose servomotor driver can operates in ***Torque Mode; Velocity Mode; Position mode.*** For linking with PPCI7443, user should set the operating mode to **Position** Mode.

Wiring of PPCI7443 with Panasonic MSD

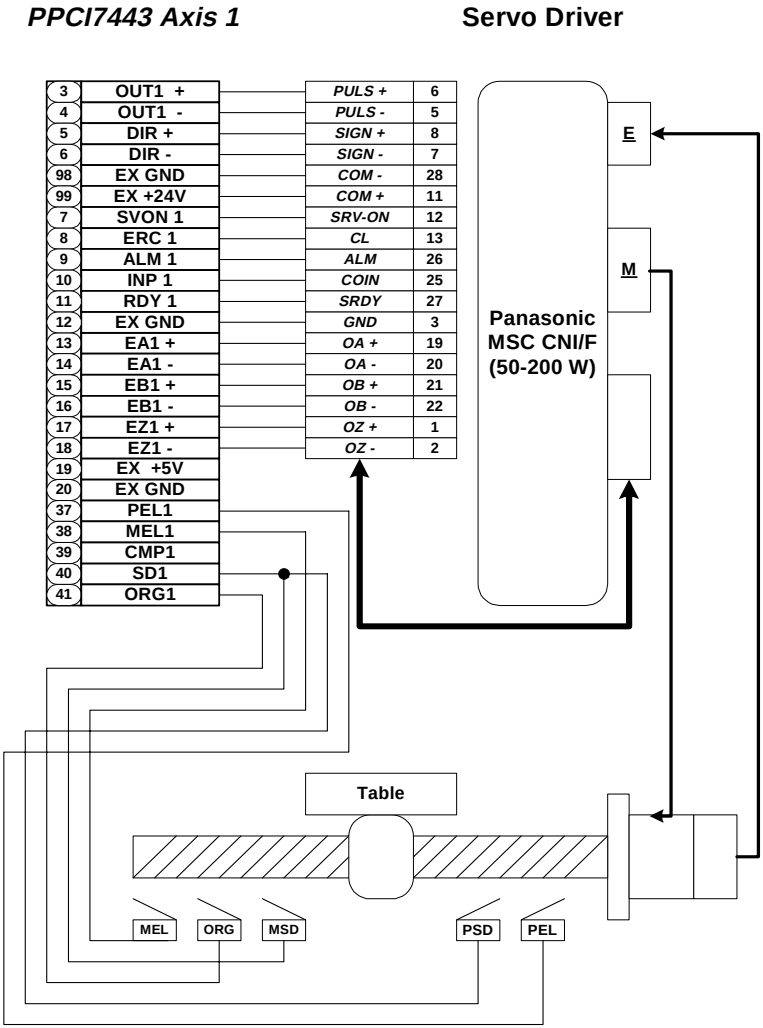


Figure 7.2 Connection of PPCI7443 with Panasonic Driver

Wiring of PPCI7443 with SANYO AC Servo PY2

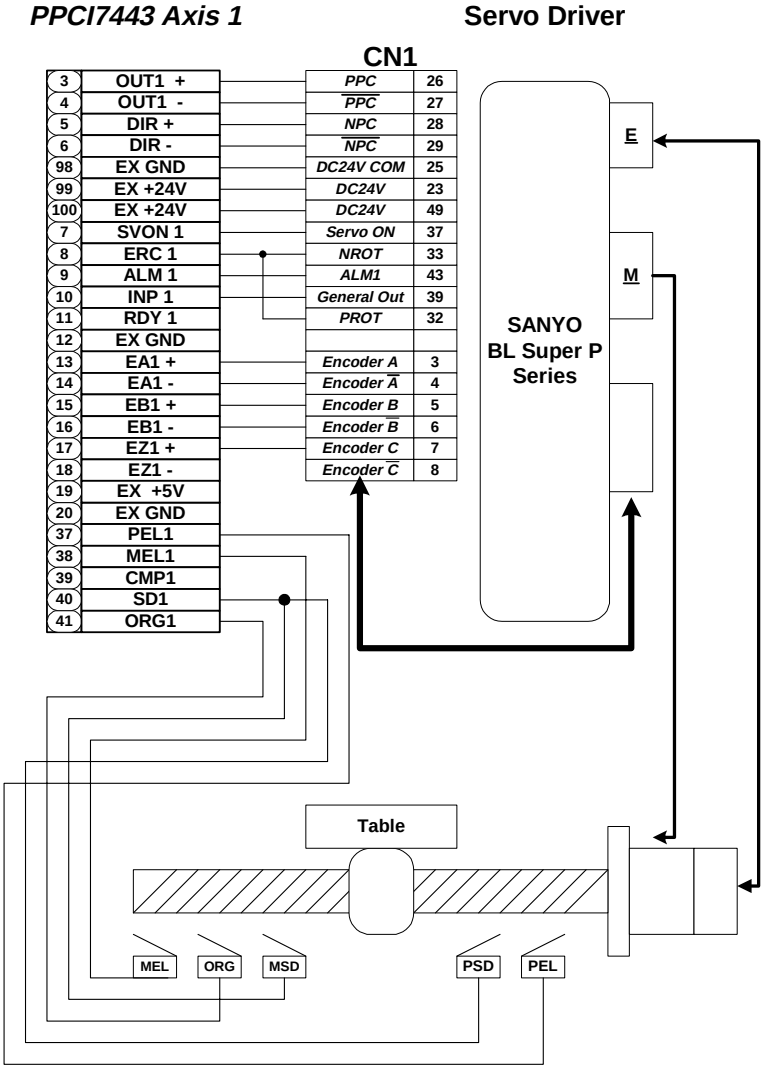


Figure 7.3 Connection of PPCI7443 with SANYO Driver