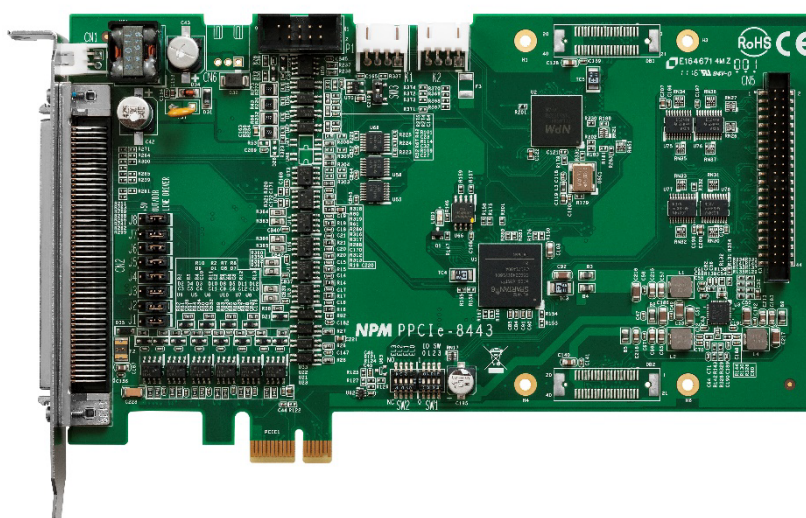




## 4-axis Control Board (PCI-Express)

PPC1e-8443

User's Manual



# INDEX

1. Introduction.....	1
1.1 How to use this manual.....	1
1.1.1 Symbol description.....	1
1.2. Handling the product.....	4
1.2.1 Storing.....	4
1.2.2 Unpacking.....	4
1.2.3 Safety.....	4
1.3. Product Warranty.....	7
1.3.1 Warranty period.....	7
1.3.2 Warranty scope.....	7
1.4. Notice.....	7
1.5. Confirmation.....	7
2. Outline.....	8
2.1. Features.....	10
2.2. Specification.....	11
2.3. Software Supporting.....	13
2.3.1 Programming Library.....	13
2.3.2 PCIe-8443 Utility.....	13
3. Installation.....	14
3.1. Accessories.....	14
3.2. PCIe-8443 Dimensions.....	15
3.3. Hardware Installation.....	16
3.3.1 Hardware Configuration.....	16
3.3.2 PCIe Slot Selection.....	16
3.3.3 Installation Procedures.....	16
3.3.4 Trouble Shooting.....	16
3.3.5 Precautions (Sleep function, Fast startup function),.....	16
3.4. Software Driver Installation.....	17
3.4.1 Precautions for Installation to Windows 7.....	17
3.4.2 Installation procedure for 32-bit Windows.....	17
3.4.3 Installation Procedure for 64-bit Windows.....	19
3.5. CN1 Pin Assignment: Emergency Input.....	20
3.6. CN2 Pin Assignment: Main Connector.....	20
3.7. P1 Pin Assignment: Manual Pulser Input.....	22
3.8. K1/K2 Pin Assignment: Simultaneous Start/Stop.....	22
3.9. CN5 Pin Assignment: General Purpose Input/Output.....	23

3.10. Jumper Setting for Pulse Output .....	25
3.11. SW1 Board Index setting .....	25
3.12. SW2 Switch setting for the logic of EL .....	26
3.13. SW3 Switch Setting to Enable Emergency Input.....	26
4. Signal Connections .....	27
4.1. Pulse Output Signals OUT and DIR .....	28
4.2. Encoder Feedback Signals: EA, EB, and EZ.....	29
4.3. Origin Signal ORG (Home return) .....	31
4.4. End-Limit Signals: PEL and MEL .....	32
4.5. Ramping-Down & PCS .....	33
4.6. In-Position Signal: INP.....	34
4.7. Alarm Signal: ALM .....	35
4.8. Deviation Counter Clear Signal: ERC.....	36
4.9. General Purpose Output Signal SVON .....	37
4.10. General Purpose Input Signal: RDY.....	37
4.11. Position Comparator Output: CMP.....	38
4.12. Position Latch Input: LTC.....	39
4.13. Pulser Input Signals: PA and PB .....	40
4.14. Simultaneous Start/Stop Signals: STA and STP .....	42
4.15. Emergency Input EMG .....	43
4.16. Extended General-Purpose Input / Output: EDI and EDO .....	43
4.17. Power Supply Configuration .....	44
5. Operation Theorem .....	45
5.1. Motion control mode .....	45
5.1.1 Output Pulse Mode .....	46
5.1.2 Velocity Mode Operation .....	48
5.1.3 Positioning Operation for Single Axis .....	49
5.1.4 S-curve Profile Acceleration / Deceleration Operation .....	51
5.1.5 Linear Interpolation for Two to Four Axes .....	53
5.1.6 Circular Interpolation for Two Axes .....	57
5.1.7 Circular Interpolation with Acceleration / Deceleration Time.....	58
5.1.8 Helical Interpolation .....	59
5.1.9 The Relationship between Velocity and Acceleration Time .....	60
5.1.10 Continuous Operation .....	62
5.1.11 Home Return Operation (Origin Return) .....	66
5.1.12 Manual pulser operation .....	72
5.1.13 Timer Mode.....	72
5.1.14 Pulser Interpolation.....	72
5.2. The Motor Driver Interface.....	73

5.2.1 INP .....	73
5.2.2 ALM .....	74
5.2.3 ERC.....	75
5.2.4 SVON and RDY .....	75
5.3. Mechanical Input Interface and I/O Status.....	76
5.3.1 SD / PCS .....	76
5.3.2 EL .....	77
5.3.3 ORG .....	77
5.3.4 EMG.....	77
5.4. Counters .....	78
5.4.1 Command Position Counter .....	78
5.4.2 Feedback Position Counter.....	78
5.4.3 Position Error Counter.....	80
5.4.4 General Purpose Counter .....	81
5.4.5 Target Position Recorder .....	81
5.5. Multiple PPCle-8443 operation .....	82
5.6. Change Position or Speed on-the-fly (Override Function).....	83
5.6.1 Change Speed on-the-fly (Speed Override).....	83
5.6.2 Change Position on-the-fly (Position Override).....	88
5.7. Comparator and Latch .....	90
5.7.1 Comparator of PPCle-8443 .....	90
5.7.2 Position Comparator.....	91
5.7.3 Position Latch .....	93
5.8. Backlash Compensator and Vibration Suppression .....	93
5.9. Software Limit Function .....	94
5.10. Interrupt Control .....	95
5.11. Idling Control .....	98
6. PPCle-8443 Utility.....	99
6.1. Execute PPCle-8443 Utility .....	99
6.2. About PPCle-8443 Utility.....	99
6.3. PPCle8443 Utility Screen Introduction .....	100
6.3.1 Board ID Switch Enable / Disable Screen.....	100
6.3.2 Main Screen.....	100
6.3.3 Interface I/O Configuration Screen.....	101
6.3.4 Pulse I/O and interrupt configuration screen.....	102
6.3.5 Operation screen .....	103
7. Function Library.....	107
7.1. List of Functions.....	107
7.2. C/C++ Programming Library .....	113
7.3. Initialization .....	114

7.4. Pulse Input/Output Configuration.....	117
7.5. Velocity Mode Operation .....	119
7.6. Single Axis Position Operation .....	122
7.7. Linear Interpolation Operation.....	127
7.8. Circular Interpolation Operation .....	132
7.9. Helical Interpolation Operation .....	137
7.10. Home Return Mode (Origin Return) .....	140
7.11. Manual Pulser Operation .....	142
7.12. Motion Status .....	145
7.13. Motion Interface I/O.....	146
7.14. Motion I/O Monitoring.....	149
7.15. Interrupt Operation .....	150
7.16. Position Controls and Counters .....	154
7.17. Position comparator and Latch.....	157
7.18. Continuous Operation .....	162
7.19. Multiple Axes Simultaneous Operation .....	163
7.20. Extended General-Purpose Input/Output .....	166
7.21. Error Code List .....	168

# 1. Introduction

Thank you for choosing our 4-axis control board (PCI-Express) PPCle-8443.

This manual describes the specifications, functions, connections, and usages of 4-axis control board (PCI-Express) PPCle-8443.

Be sure to read this manual thoroughly and keep it handy in order to use the product appropriately.

## 1.1 How to use this manual

1. Reproduction of this manual in whole or in part without permission is prohibited by the Copyright Act.
2. The contents of this manual are subject to change without the prior notice along with the improvement of performance and quality.
3. Although this manual is produced with the utmost care, please contact our sales representative if there are any questions, errors or omissions.

### 1.1.1 Symbol description

#### 1.1.1.1 Physical damage level

In this manual, the physical damage level is defined as follows.

- Serious injury  
Those that might cause aftereffects such as loss of sight, wound, burn, electric shock, fracture, poisoning, or those requiring hospitalization or long-term outpatient treatment.
- Minor injury  
Those not requiring hospitalization or long-term outpatient treatment. (Other than "serious injury" above)

#### 1.1.1.2 Hazardous level

The product is designed with the top priority for the safety of operators. However, due to the nature of the product, there are risks that cannot be eliminated. In this manual, the seriousness and level of these risks are divided into three categories: "Danger," "Warning," and "Caution." Be sure to read and understand the symbols descriptions thoroughly before operating or performing maintenance work on the product.

"Danger", "Warning", and "Caution" are indicated in the order of severity of hazard: (danger > warning > caution), and the meanings are described underneath.



**D a n g e r**

"Danger" indicates that it might cause an imminent risk that could result in the death or serious injury of the operator during operations of this product.



**W a r n i n g**

"Warning" indicates that it may result in the death or serious injury of the operator during operations of this product.



**C a u t i o n**

"Caution" indicates that it may result in minor injury of the operator during operations of this product.

**C a u t i o n**

"Caution" without warning symbol indicates that the operator is not likely to be injured, but it can cause damage or result in a malfunction to this product, your equipment, or your instruments.

In addition to the hazardous level classifications described above, the following notations are also used.

**I m p o r t a n c e**

"Importance" indicates the information and contents that must be known particularly in operations and maintenance works of this product.

**R e m a r k s**

"Remarks" initiates the useful information or contents for operations and maintenance works of this product.

### 1.1.1.3 Warning symbol

In this manual, the following symbols are added along with the notations "Danger," "Warning," "Caution," and "Importance" to indicate the warning contents in an easy-to-understand manner.



This symbol indicates that a high voltage may be applied.  
Failure to confirm safety or mishandling of this product might cause a risk of electric shock, burn, or death.



This symbol indicates that some parts have a high surface temperature, and the mishandling can cause a risk of burns.



This symbol indicates that mishandling may cause a fire.



This symbol indicates "prohibited" actions that must not be performed in the operation and the maintenance work of this product.



This symbol indicates "mandatory" actions that must be performed in the operation and the maintenance work of this product.



## 1.2. Handling the product

### 1.2.1 Storing

Store the product in an environment where condensation does not occur at a temperature of  $-20^{\circ}\text{C}$  to  $+80^{\circ}\text{C}$ .

### 1.2.2 Unpacking

Check if the following items are included when unpacking.

- PCIe-8443 body 1 unit
- Emergency stop input cable (for CN1)



### 1.2.3 Safety



This section describes basic safety precautions for safer operations.

Follow the instructions below when using the product.

Failure to comply with the items may result in injuries or disasters.

#### 1.2.3.1 Precautions for transportation and installation

 <b>C a u t i o n</b>	
	<ul style="list-style-type: none"> <li>• Since this is a precision instrument, do not drop it or apply a strong impact to it.</li> <li>• Be careful not to overload the product as it may cause the load to collapse.</li> <li>• Do not step on or place heavy loads on this product. It may cause an injury or a damage to the product.</li> <li>• Be sure to wire correctly and securely. Failure to do so may cause the motor to run out of control that result in injuries or failures.</li> <li>• Do not mis-wire in pin connectons, which may cause failures.</li> <li>• Various connecting cables must be firmly fixed in the vicinity and no tensile stress must be applied to them. Failure to do so may result in failures.</li> <li>• Please use the end limit signal and emergency stop signal for safety when wiring as necessary.</li> <li>• When attaching the board to a PC, align the connectors on the PC side with the card edge of the board, and then insert it slowly and carefully.</li> </ul> <p>If the card is misaligned or slewed position, the edge of the board may be damaged, resulting in failures.</p>

 <b>C a u t i o n</b>	
	<ul style="list-style-type: none"> <li>• Be sure to follow the installation method. Failures may occur. (For details, see section 3.3 in this manual.)</li> <li>• Do not allow foreign objects to fall on or enter the board as it may result in failures or fire.</li> </ul>



### C a u t i o n



- Do not install this product in places where corrosive gas, oil droplets, dust, water vapor, metal powder, etc. are present. Failures may occur.
- Do not use a power supply other than the specified one to prevent failures.
- Do not install this product in places where severe vibration exists or where sealed. Failures may occur.
- Do not install or remove this product while the power is ON, which may result in failures.
- Do not apply excessive force that may distort the board during installation, which may result in failures.



### C a u t i o n



- Please do not touch the product during energization or for a while after turning off the power. There is a risk of burns due to the high temperature parts.

## 1.2.3.2 Precaution in operation








### C a u t i o n



- Check and adjust each set values before operations.
- Be sure not to make any adjustments or changes that are not described in this manual as the operation can be unstable.
- Set an emergency stop circuit externally so that the operation stops immediately, and the power can be cut off.
- Check the operation of a motor fixed and disconnected from the mechanical system before installing the motor in your machine. There is a risk of damage to the machine or injury.
- When an alarm occurs, eliminate the cause, ensure the safety, and reset the alarm before restarting (for details, see section 4.7). There is a risk of injury.
- Use a noise filter to minimize the effect of electromagnetic interference. There is a risk of electromagnetic interference with electronic devices used near the servo drivers.
- Insert the connector firmly all the way to the base. Do not insert or remove the connector with wet hands.
- If smoke, unusual smells or noises are noticed, turn off the power immediately.
- The codes listed in this manual are examples.  
When programming, please use it after implementing processing such as input value and return value confirmations as necessary.

### 1.2.3.3 Maintenance Precautions

 <b>D a n g e r</b>	
 	<ul style="list-style-type: none"><li>• Do not conduct an inspection of the product while it is energized. It may result in an electric shock.</li><li>• Terminat (Turn OFF) the input power supply and PC power supply, and wait at least three minutes before conducting inspections.</li><li>• Do not disassemble, modify, or repair in any way that is not described in this manual.</li></ul>

 <b>C a u t i o n</b>	
	<ul style="list-style-type: none"><li>• Do not install or remove the product or wire while it is being supplied with electric power. Failures may occur.</li><li>• Do not disassemble, convert or repair the product, which are not described in the manual.</li></ul>

## 1.3. Product Warranty

This content is the warranty of the product purchased from Nippon Pulse Motor.

When the product is purchased from a supplier other than NPM, please contact the supplier regarding the product's warranty.

### 1.3.1 Warranty period

The warranty period is one year from the date of delivery to an assigned place.

### 1.3.2 Warranty scope

If any defect is found in a product during the warranty period under the normal use following this document, NPM will repair or replace the product without charge.

However, the following cases are not covered by the warranty even during the warranty period.

- 1) Products modified or repaired by anyone other than NPM or a person authorized by NPM.
- 2) Defects that result from dropping after the delivery or mishandling in transit.
- 3) Natural deterioration, wearing, and fatigue of components.
- 4) Defects result from any usage other than the original described in this manual.
- 5) Defects result from natural disaster or force majeure such as fires, earthquakes, lightning strikes, winds, floods, salts or electrical surges.
- 6) Defects or damages result from a cause that is not the fault of NPM.

Free repairs will only be conducted at NPM locations; no repairs will be made by business trips.

Warranty period of repaired product is the same as the warranty period before repair.

This warranty covers the product itself. The detriments or damages induced by the product failure etc. will not be covered by the warranty.

## 1.4. Notice

This document aims to describe the details of functions of the product. It does not warrant fitness for a particular purpose of the customer. Also, the examples of applications and circuit diagrams in this manual are included only for your reference. Please confirm the features and the safety of device or equipment before use.

## 1.5. Confirmation

Please do not use this product in the following conditions. If you need to use in the following conditions, please contact our sales representatives:

1. Any equipment that may require a high reliability or a safety, such as nuclear facilities, electricity or gas supply systems, transportation facilities, vehicles, various safety systems, medical equipment, etc.
2. Any equipment that may directly affect human survival or property.
3. Usages under conditions or circumstances that are not specified in the catalog, manual, etc.

For applications that may cause serious damages to a human life or property due to failure of this product, ensure high reliability and safety by redundant design.

## 2. Outline

PPCLe-8443 is an advanced 4 axes motion controller board with PCI Express interface. It can generate high frequency pulses (6.5 Mpps) to drive stepping motors and servo motors with pulse train input. It provides 2-axis circular, linear interpolation between 2 to 4 axes and continuous interpolation with velocity continuity. Also, changing positions and speeds on the fly are available in single axis operation. Multiple PPCLe-8443 up to 12 boards can be used in one system. Incremental encoder interface with all four axes provides the ability to correct positioning errors generated by inaccurate mechanical transmissions. And with the help of on-board FIFO, PPCLe-8443 can perform a precise and extremely fast position comparison and a trigger function without consuming CPU resource. In addition, mechanical sensor interface, servo motor interface and general-purpose I/O signals are provided for system integration.

Figure 2-1 in the next page shows the function block diagram of PPCLe-8443. PPCLe-8443 uses one ASIC (PCL6046) to perform 4 axes motion control, and the ASIC is made by Nippon Pulse Motor. The motion control functions include linear and S-curve acceleration / deceleration, circular interpolation between two axes, linear interpolation between 2 ~ 4 axes, continuous motion, in positioning, and 13 home return modes. Since these functions needing complex computations are done internally in the ASIC, the PC's CPU is free to supervise and perform other tasks.

PPCLe-8443 Utility, Microsoft Windows based software, is for supporting your application developments. It is very helpful for debugging a motion control system during the design phase of a project. The on-screen monitor shows all installed axis information and I/O signals status of PPCLe-8443. In addition to PPCLe-8443 Utility, Windows version function library are included using programmers using C++ and Visual Basic language. Several sample programs are provided to illustrate how to use the function library.

Figure 2-2 is a flowchart to show a recommended process to develop an application using this manual. Please also see the relative chapters for the details of each process.

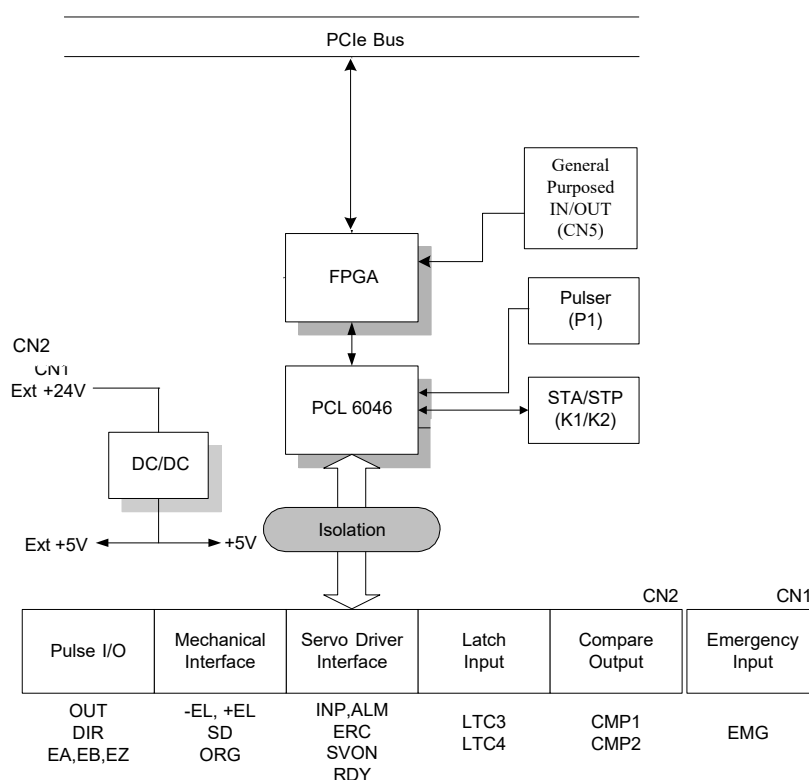


Figure 2-1 Block Diagram of PPCLe-8443

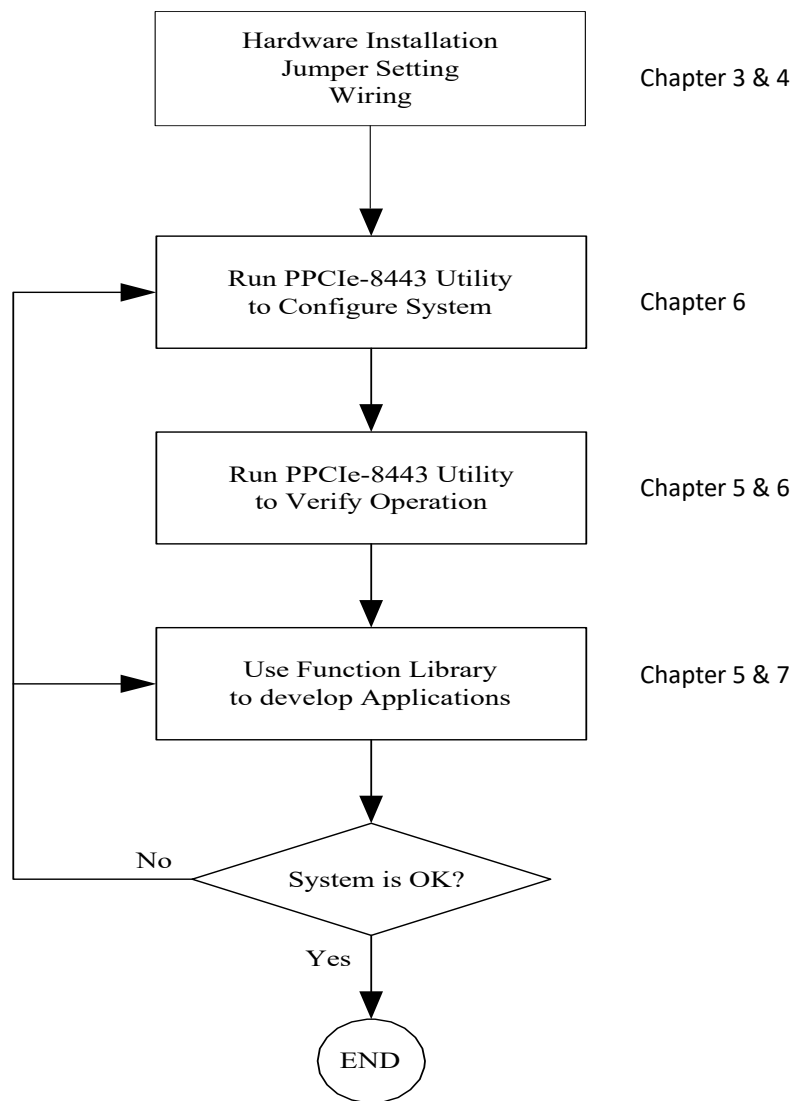


Figure 2-2 Flowchart of building an application

## 2.1. Features



The following shows the main features of PPCle-8443.

- PCIe-Bus plug and play
- 4 axes of step and direction pulse outputs for controlling stepping motors and servo motors
- Maximum output frequency of 6.5 Mpps
- Pulse output options: OUT / DIR, CW / CCW, or AB phase
- Maximum encoder input frequency of 6.5 Mpps at 4 x AB phase
- Maximum pulser input frequency of 6.5 Mpps at 4 x AB phase and CW / CCW
- Programmable acceleration and deceleration time
- Trapezoidal and S-curve velocity profiles for all modes
- Circular interpolation by any 2 axes
- Linear interpolation by any 2 to 4 axes
- Continuous interpolation for contour following motion
- Change position and speed on the fly
- Change speed by comparator condition
- 13 home return modes
- Hardware backlash compensator and vibration suppression
- Software end-limits for each axis without hardware switches.
- 32-bit up / down counters for incremental encoder feedback.
- Home switch, index signal (EZ), positive and negative end-limit switches interface provided for all axes.
- Axes high speed position latch input
- Axes position compare trigger output with 4K FIFO auto-loading
- Maximum trigger output frequency of 25 KHz
- All digital input and output signals are 2500 Vrms isolated (excluding SAT, STP, EDI, and EDO)
- Programmable interrupt sources.
- Simultaneous start / stop motion on multiple axes.
- Manual pulser input interface.
- Software supports maximum 12 boards of PPCle-8443 (48 axes) in one system.
- Compact, half size PCB.
- PPCle-8443 Utility, Microsoft Windows based application development software.
- PPCle-8443 Library and Utility for Windows7/8/10 (32 / 64 Bit) DLL, and sample programs (VC, VB, and C#) are included.

## 2.2. Specification

Applicable motors		Pulse input type Servo motors, Stepping motors
Bus interface		PCI-Express
Built-in LSI		PCL6046 (Nippon Pulse Motor)
Performance	Number of controllable axes	4
	Max pulse output frequency	6.5 Mpps (Constant speed operation, Linear/S-curve acceleration/deceleration operation)
	Reference clock	19.6608 MHz
	Positioning pulse setting range	-2,147,483,648 ~ +2,147,483,647 (32 bit)
	Pulse rate setting range (pulse ratio Multiplier)	0.1 pps ~ 6,553.5 pps (Multiplier = 0.1 ) 1 pps ~ 65,535 pps (Multiplier = 1 ) 100 pps ~ 6,553,500 pps (Multiplier = 100)
Motion control	I/O Signals	Command pulse output: OUT and DIR (each axis) Incremental encoder signals input: EA, EB (each axis) Encoder Z-phase input : EZ (each axis) Mechanical input: + EL, - EL, SD / PCS, ORG (each axis) Servo driver I/F : INP, ALM, ERC (each axis) Position latch input: LTC (Axis 2, Axis 3) Comparator output: CMP (Axis 0, Axis 1) General-purpose output: SVON (each axis) General-purpose input: : RDY (each axis) Pulser signal input: PA, PB (All axis in common, motion axis selected by software) Simultaneous Start / Stop signals I/O : STA, STP Emergency input: EMG Photo coupler insulation, insulation voltage:2500 Vrms (excluding STA, STP, EDI, and EDO)
	Extended general-purpose input / output	Input 16 points, Output 16 points (EDI, EDO)
General specification	Axis control connector	SCSI-type 100-pin connector
	Operating Temperature	0 °C ~ 50 °C
	Storage Temperature	-20 °C ~ 80 °C
	Humidity	5 % ~ 85 % non-condensing



	Environment/Standard	<p>RoHS Directive: Identified by the following marks.</p> <div>  <p>RoHS Directive: 2011/65/EU (2015/863/EU is not included)</p> </div> <div>  <p>RoHS Directive: 2011/65/EU (2015/863/EU is included)</p> </div> <p>CE marking (EN 55022: 2010 /AC: 2011 EN 61000-3-2: 2014 EN 61000-3-3: 2013 EN 55024: 2010)</p>
	Power consumption	<p>Bus power supply (input) : + 12 V DC <math>\pm</math> 5 %, 250 mA Max  External power supply(input) : + 24 V DC <math>\pm</math> 5 %, 500 mA Max  External power supply(output) : + 5 V DC <math>\pm</math> 5 %, 500 mA Max</p>
	Dimensions	185 mm (L) X 98.4 mm (H)

## 2.3. Software Supporting

### 2.3.1 Programming Library

NPM provides Windows7/8/10 (32 / 64 bit) DLL for PPCle-8443.

Please download the board setup program from our website and install it on the PC. It will be installed under the Program Files folder.

### 2.3.2 PPCle-8443 Utility

Windows-based utility softwares are available for customers to set up boards, motors and systems. It can help users to debug their hardwares and softwares.

Parameter settings such as I/O logic and signal specification set up using this software can also be used in customers' programs.

This software is installed by board setup program.

See Chapter 6 for details.

## 3. Installation

This chapter describes how to install PPCle-8443. Please follow the steps below to install it correctly:

- Check the accessories (Section 3.1)
- Check the PC board (Section 3.2)
- Install the hardware (Section 3.3)
- Install the software driver (Section 3.4)
- Understand the I/O signal connectors (Chapter 4) and the operation methods (Chapter 5)
- Confirm the connectors' pin assignments (Section 3.5 to 3.13) and wire them.

### 3.1. Accessories

In addition to PPCle-8443 Motion Control board, the package includes the following item:

Emergency input cable (for CN1)

If the item is missing or is damaged, contact the supplier of the product. Please keep the accessories and the carton box in the case you ship or store the product in the future.

## 3.2. PPCle-8443 Dimensions

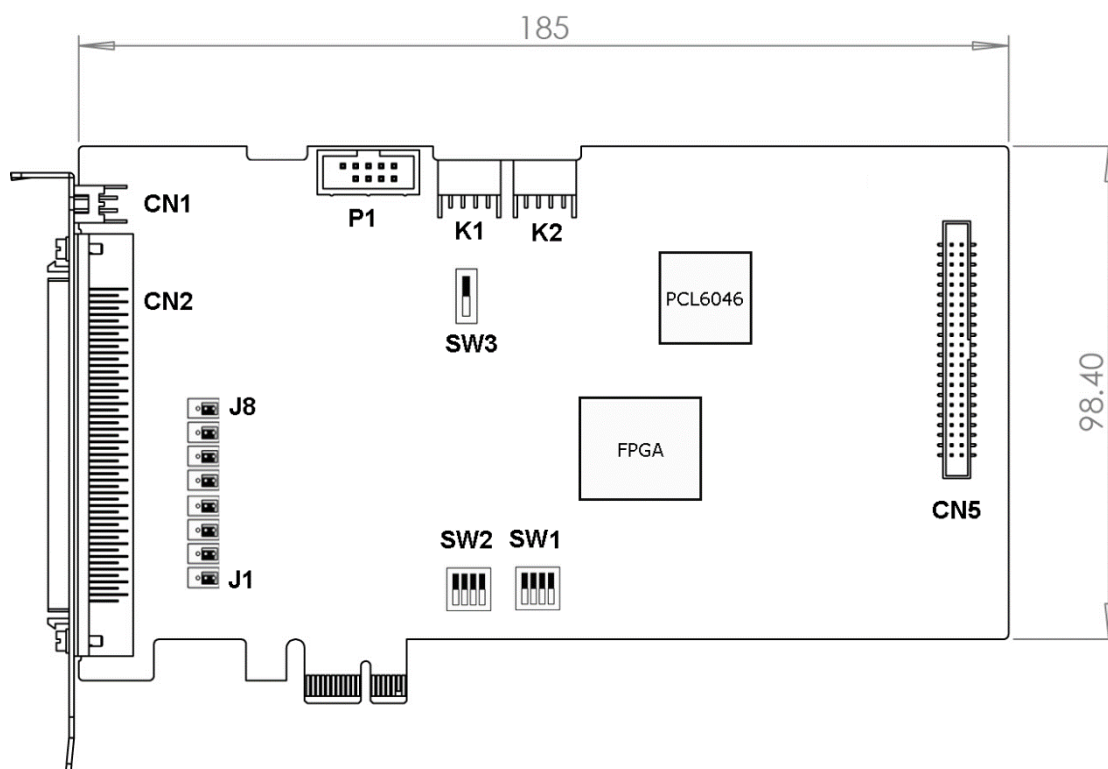
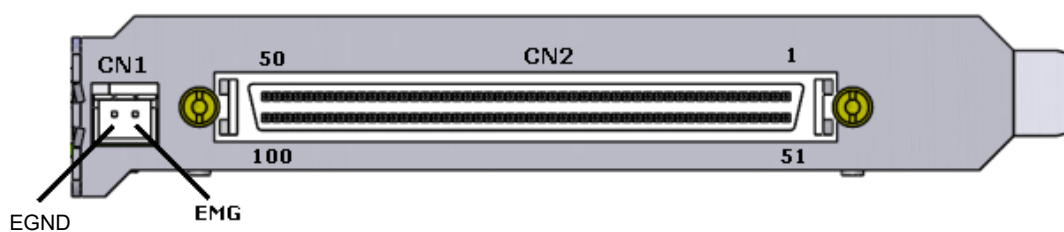


Figure 3.2-1 PCB Layout of PPCle-8443

- CN1: Emergency Input Connector
- CN2: Input / Output Signal Connector
- CN5: Extended General-Purpose Input/ Output Connector
- P1: Manual Pulser Signal Connector
- K1/K2: Simultaneous Start / Stop Connector
- SW1: Board ID setting Switch
- SW2: End Limit Logic Selection Switch
- SW3: Emergency Input Enable Selection Switch
- J1 ~ J8: Pulse Output Type Selection Jumper



## **3.3. Hardware Installation**

### **3.3.1 Hardware Configuration**

PPCLe-8443 is a plug and play PCIe controller board. The memory usage (I/O port locations) of the PCIe board is assigned by system BIOS. The address assignment is done board-by-board basis in the system.

### **3.3.2 PCIe Slot Selection**

PPCLe-8443 board can be used in any PCIe slot.

### **3.3.3 Installation Procedures**

1. Read through this manual, and setup jumpers according to your application.
2. After turning off the PC, then unplug the power cable which is connected to the PC.
3. Remove the computer cover.
4. Select a PCIe expansion slot to insert the board.
5. Before handling PPCLe-8443, discharge any static buildup on your body by touching the metal case of the computer. Hold the edges and do not touch the components on the board.
6. Insert the board into the PCIe slot you selected.
7. Secure the board in place at the rear panel of the system with the screws removed from the slot.

### **3.3.4 Trouble Shooting**

If the system does not start up or causes an unstable operation although you inserted the board in the correct place, please check the system BIOS manual. It may be caused by an interrupt conflict.

### **3.3.5 Precautions (Sleep function, Fast startup function),**

This board and software do not support the sleep function and fast startup function of OS.  
Please use them with turning the sleep function and fast startup function OFF.

## 3.4. Software Driver Installation

### 3.4.1 Precautions for Installation to Windows 7.

When the OS of the PC is Windows 7, it is necessary to conduct Windows Update (support information: KB 3033929) in terms of SHA-2 compatibility:

Japanese version

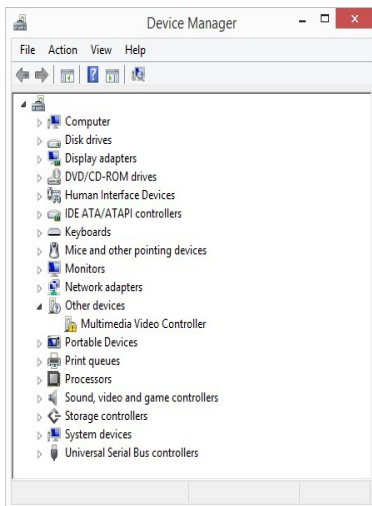
<http://www.microsoft.com/ja-jp/download/details.aspx?id=46078>

English version

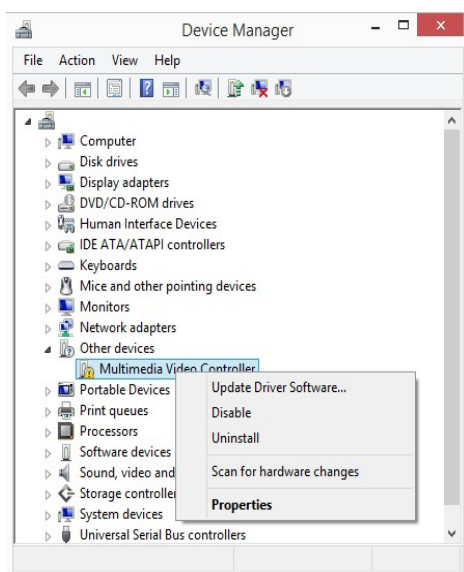
<https://www.microsoft.com/en-US/download/details.aspx?id=46078>

### 3.4.2 Installation procedure for 32-bit Windows

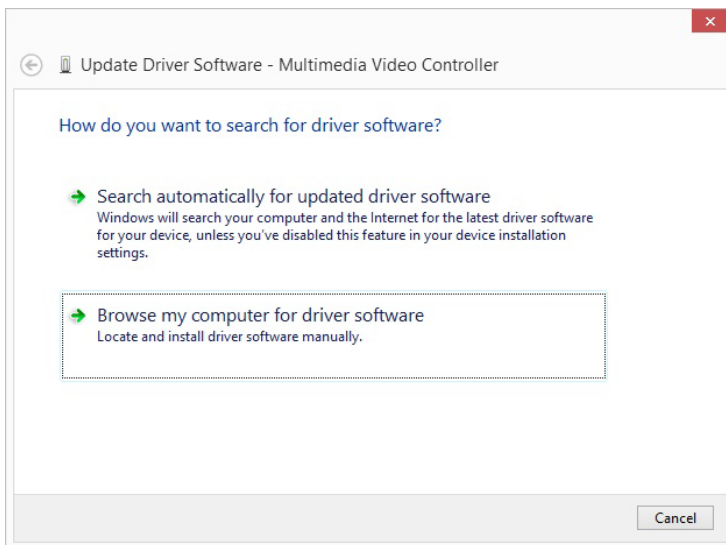
- 1) Execute the setup.exe (32 bit) with “Run as administrator”
- 2) After finishing the set-up, open “Device Manager”, and find “Multimedia Video Controller” under “Other devices”.



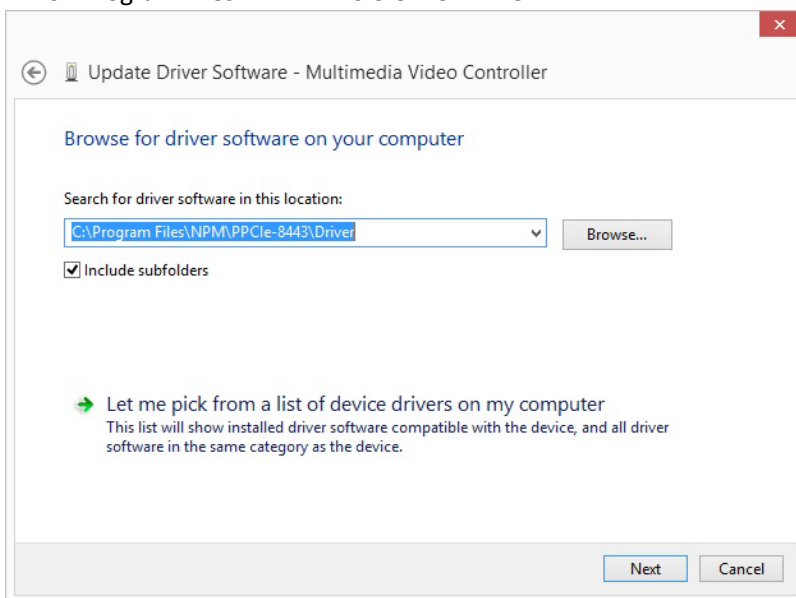
- 3) Select “Update Driver Software” from the Right-Click Menu.



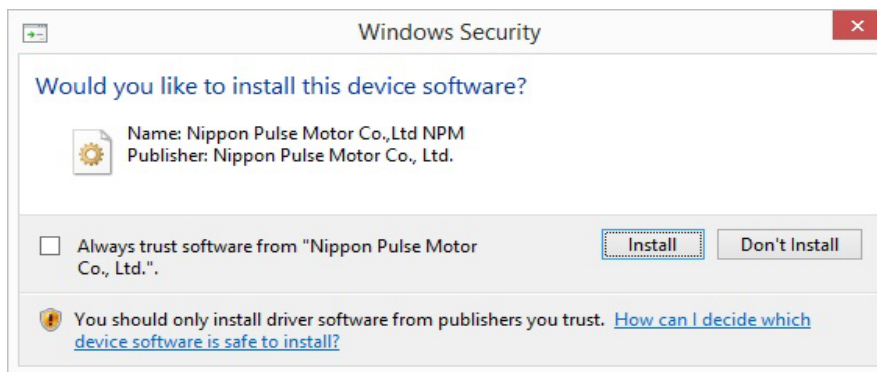
4) Click “Browse my computer for driver software”.



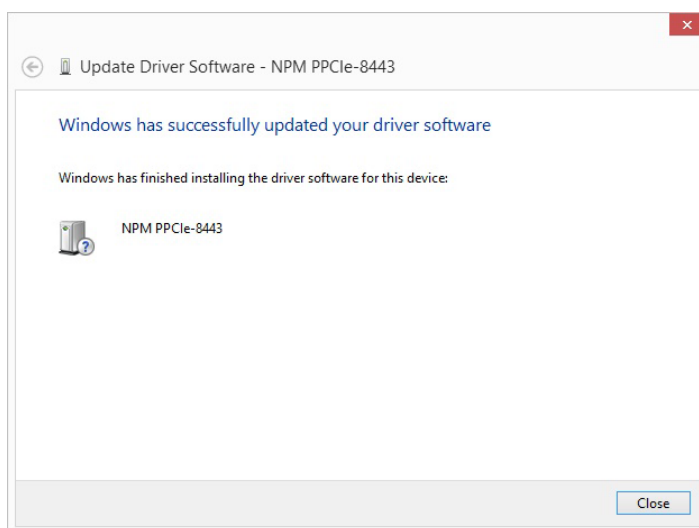
5) Click “Browse” button and enter the following driver folder name:  
“C:\Program Files\NPM\PPCLe-8443\Driver”



- 6) When the driver installation is executed, the following screen will be displayed. Then, click "Install".  
("You do not have to check Always trust software from "Nippon Pulse Motor Co., Ltd.")



- 7) When the driver installation is completed normally, the following screen will be displayed.



### 3.4.3 Installation Procedure for 64-bit Windows

- 1) Execute setup.exe (64 bit) with "Run as administrator".
- 2) When the software set-up is completed, the driver installation wizard will be executed continuously. Follow the instructions on the screen to complete the installation of the driver.



### 3.5. CN1 Pin Assignment: Emergency Input

Board side connector: 2317RJ-02 (NELTRON)

Cable side connector: A2501H02-2P (JOWLE)

No.	Name	Function
1	EGND	Ext. Power ground
2	EMG	Emergency signal

Note: CN1 is a plug-in terminal board with no screw.

### 3.6. CN2 Pin Assignment: Main Connector

CN2 is a major connector for motion control I/O signals.

Board side connector: 0-5787082-9 (AMP)

Cable side connector: Z1-013100 (All Best Electronics)

No.	Name	I/O	Function	No.	Name	I/O	Function
1	VDD	O	+5V power supply output	51	VDD	O	+5V power supply output
2	EGND	-	Ext. power ground	52	EGND	-	Ext. power ground
3	OUT0+	O	Pulse signal (+)	53	OUT2+	O	Pulse signal (+)
4	OUT0-	O	Pulse signal (-)	54	OUT2-	O	Pulse signal (-)
5	DIR0+	O	Dir. signal (+)	55	DIR2+	O	Dir. signal (+)
6	DIR0-	O	Dir. signal (-)	56	DIR2-	O	Dir. signal (-)
7	SVON0	O	Servo On/Off	57	SVON2	O	Servo On/Off
8	ERC0	O	Dev. ctr, clr. Signal	58	ERC2	O	Dev. ctr, clr. signal
9	ALM0	I	Alarm signal	59	ALM2	I	Alarm signal
10	INP0	I	In-position signal	60	INP2	I	In-position signal
11	RDY0	I	Multi-purpose Input signal	61	RDY2	I	Multi-purpose Input signal
12	EGND	-	Ext. power ground	62	EGND	-	Ext. power ground
13	EA0+	I	Encoder A-phase (+)	63	EA2+	I	Encoder A-phase (+)
14	EA0-	I	Encoder A-phase (-)	64	EA2-	I	Encoder A-phase (-)
15	EB0+	I	Encoder B-phase (+)	65	EB2+	I	Encoder B-phase (+)
16	EB0-	I	Encoder B-phase (-)	66	EB2-	I	Encoder B-phase (-)
17	EZ0+	I	Encoder Z-phase (+)	67	EZ2+	I	Encoder Z-phase (+)
18	EZ0-	I	Encoder Z-phase (-)	68	EZ2-	I	Encoder Z-phase (-)
19	VDD	O	+5V power supply output	69	VDD	O	+5V power supply output
20	EGND	-	Ext. power ground	70	EGND	-	Ext. power ground
21	OUT1+	O	Pulse signal (+)	71	OUT3+	O	Pulse signal (+)
22	OUT1-	O	Pulse signal (-)	72	OUT3-	O	Pulse signal (-)
23	DIR1+	O	Dir. signal (+)	73	DIR3+	O	Dir. signal (+)
24	DIR1-	O	Dir. signal (-)	74	DIR3-	O	Dir. signal (-)
25	SVON1	O	Servo On/Off	75	SVON3	O	Servo On/Off
26	ERC1	O	Dev. ctr, clr. Signal	76	ERC3	O	Dev. ctr, clr. signal
27	ALM1	I	Alarm signal	77	ALM3	I	Alarm signal

No.	Name	I/O	Function	No.	Name	I/O	Function
28	INP1	I	In-position signal	78	INP3	I	In-position signal
29	RDY1	I	Multi-purpose Input signal	79	RDY3	I	Multi-purpose Input signal
30	EGND	-	Ext. power ground	80	EGND	-	Ext. power ground
31	EA1+	I	Encoder A-phase (+)	81	EA3+	I	Encoder A-phase (+)
32	EA1-	I	Encoder A-phase (-)	82	EA3-	I	Encoder A-phase (-)
33	EB1+	I	Encoder B-phase (+)	83	EB3+	I	Encoder B-phase (+)
34	EB1-	I	Encoder B-phase (-)	84	EB3-	I	Encoder B-phase (-)
35	EZ1+	I	Encoder Z-phase (+)	85	EZ3+	I	Encoder Z-phase (+)
36	EZ1-	I	Encoder Z-phase (-)	86	EZ3-	I	Encoder Z-phase (-)
37	PEL0	I	End limit signal (+)	87	PEL2	I	End limit signal (+)
38	MEL0	I	End limit signal (-)	88	MEL2	I	End limit signal (-)
39	CMP0	O	Position compare output 0	89	LTC2	I	Position latch input 2
40	SD/PCS0	I	Ramp-down signal 0	90	SD/PCS2	I	Ramp-down signal 2
41	ORG0	I	Origin signal	91	ORG2	I	Origin signal
42	EGND	-	Ext. power ground	92	EGND	-	Ext. power ground
43	PEL1	I	End limit signal (+)	93	PEL3	I	End limit signal (+)
44	MEL1	I	End limit signal (-)	94	MEL3	I	End limit signal (-)
45	CMP1	O	Position compare output 1	95	LTC3	I	Position latch input 3
46	SD/PCS1	I	Ramp-down signal 1	96	SD/PCS3	I	Ramp-down signal 3
47	ORG1	I	Origin signal	97	ORG3	I	Origin signal
48	EGND	-	Ext. power ground	98	EGND	-	Ext. power ground
49	EGND	-	Ext. power ground	99	E_24V	-	Isolation power Input, +24V
50	EGND	-	Ext. power ground	100	E_24V	-	Isolation power Input, +24V

Note: It is necessary to supply 24 V external power supply to E\_24 V terminal. See “Power supply configuration” (Section 4.17).

### 3.7. P1 Pin Assignment: Manual Pulser Input

The signals on P1 are for manual pulser input.

Board side connector: 23N6960-10S10B-01G-V10-G (JIH)

Cable side connector: 110M10-HA12A (KEENTOP)

No.	Name	Function (Axis)
1	EX + 5 V	Isolated Power + 5 V Output
2	PA+	Pulser A+ phase signal input
3	PA–	Pulser A– phase signal input
4	PB+	Pulser B+ phase signal input
5	PB–	Pulser B– phase signal input
6	EGND	External Ground
7	N/A	Not Available
8	N/A	Not Available
9	N/A	Not Available

Note: EX + 5V is isolated from the bus power supply. See the “Power supply configuration” (Section 4.17).

### 3.8. K1/K2 Pin Assignment: Simultaneous Start/Stop

K1/K2 are connectors for simultaneously start/stop signals for multiple axes of multiple boards.

Board side connector: 2317RJ-04 (NELTRON)

Cable side connector: 2318HJ-04 (NELTRON)

No.	Name	Function
1	N/A	N/A
2	STA	Simultaneous start signal input/output
3	STP	Simultaneous stop signal input/output
4	GND	Power ground

Note: GND terminals are connected to the PCIe bus power supply GND. See the “Power supply configuration” (Section 4.17).

### 3.9. CN5 Pin Assignment: General Purpose Input/Output

The signals on CN5 connector are for general purpose input/output.

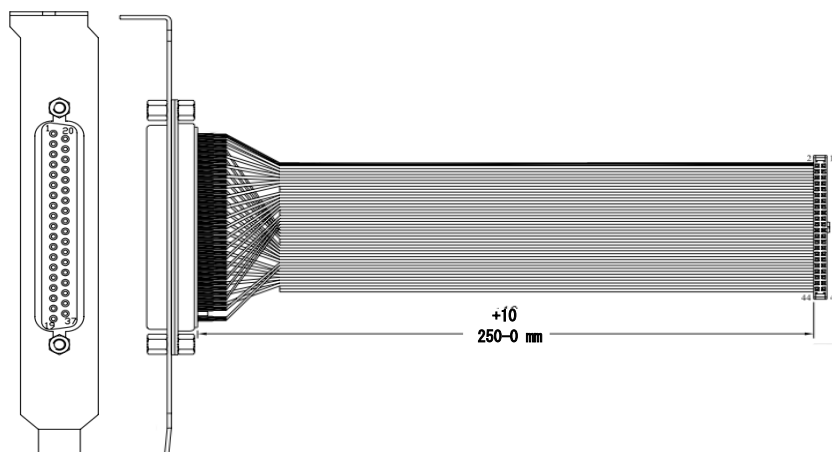
Board side connector: 23N6850-44M10B-01G-7.6-C (JVE)

Cable side connector: MFC-44 (YELE)

No.	Name	I/O	Function	No.	Name	I/O	Function
1	GND	--	Power ground	2	GND	--	Power ground
3	EDI0	I	Digital Input 0	4	EDI1	I	Digital Input 1
5	EDI2	I	Digital Input 2	6	EDI3	I	Digital Input 3
7	EDI4	I	Digital Input 4	8	EDI5	I	Digital Input 5
9	VCC	O	Power + 3.3 V	10	GND	--	Power ground
11	EDI6	I	Digital Input 6	12	EDI7	I	Digital Input 7
13	EDI8	I	Digital Input 8	14	EDI9	I	Digital Input 9
15	EDI10	I	Digital Input 10	16	EDI11	I	Digital Input 11
17	GND	--	Power ground	18	GND	--	Power ground
19	EDI12	I	Digital Input 12	20	EDI13	I	Digital Input 13
21	EDI14	I	Digital Input 14	22	EDI15	I	Digital Input 15
23	EDO0	O	Digital Output 0	24	EDO1	O	Digital Output 1
25	EDO2	O	Digital Output 2	26	EDO3	O	Digital Output 3
27	GND	--	Power ground	28	GND	--	Power ground
29	EDO4	O	Digital Output 4	30	EDO5	O	Digital Output 5
31	EDO6	O	Digital Output 6	32	EDO7	O	Digital Output 7
33	EDO8	O	Digital Output 8	34	EDO9	O	Digital Output 9
35	GND	--	Power ground	36	VCC	O	Power + 3.3 V
37	EDO10	O	Digital Output 10	38	EDO11	O	Digital Output 11
39	EDO12	O	Digital Output 12	40	EDO13	O	Digital Output 13
41	EDO14	O	Digital Output 14	42	EDO15	O	Digital Output 15
43	GND	--	Power ground	44	GND	--	Power ground

Note: The VCC (power supply+ 3.3 V) is supplied directly by the PCIe bus power supply. The GND terminal is connected to the PCIe bus power supply GND. See the “Power supply configuration” (Section 4.17).

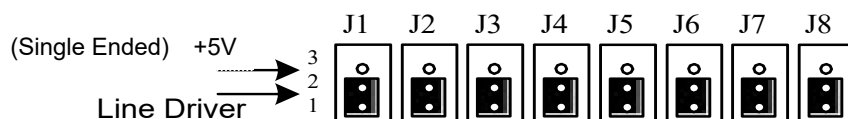
The terminal assignment of D-SUB 37P connector on the extension cable (optional) is listed as follows:



No.	Name	I/O	Function	No.	Name	I/O	Function
1	GND	--	Power ground	20	GND	--	Power ground
2	EDI0	I	Digital Input 0	21	EDO0	O	Digital Output 0
3	EDI1	I	Digital Input 1	22	EDO1	O	Digital Output 1
4	EDI2	I	Digital Input 2	23	EDO2	O	Digital Output 2
5	EDI3	I	Digital Input 3	24	EDO3	O	Digital Output 3
6	EDI4	I	Digital Input 4	25	EDO4	O	Digital Output 4
7	EDI5	I	Digital Input 5	26	EDO5	O	Digital Output 5
8	EDI6	I	Digital Input 6	27	EDO6	O	Digital Output 6
9	EDI7	I	Digital Input 7	28	EDO7	O	Digital Output 7
10	EDI8	I	Digital Input 8	29	EDO8	O	Digital Output 8
11	EDI9	I	Digital Input 9	30	EDO9	O	Digital Output 9
12	EDI10	I	Digital Input 10	31	EDO10	O	Digital Output 10
13	EDI11	I	Digital Input 11	32	EDO11	O	Digital Output 11
14	EDI12	I	Digital Input 12	33	EDO12	O	Digital Output 12
15	EDI13	I	Digital Input 13	34	EDO13	O	Digital Output 13
16	EDI14	I	Digital Input 14	35	EDO14	O	Digital Output 14
17	EDI15	I	Digital Input 15	36	EDO15	O	Digital Output 15
18	GND	--	Power ground	37	NC	-	-
19	VCC	O	Power + 3.3 V				

### 3.10. Jumper Setting for Pulse Output

J1~J8 is used to set the signal type of pulse output signals (DIR and OUT). The output signal types can be selected either differential line driver output or single ended output. See section 3.1 for the details of jumper settings. The default setting is the differential line driver mode.



### 3.11. SW1 Board Index setting

SW1 is used to set a board index. For example, if SW1 is set to ON and the others are to OFF, the board index will be 1. The index value can be set from 0 to 11. The values from 12 to 15 are unable to use. The default setting is 0. See the below table for the details.

Setting of SW1 switch



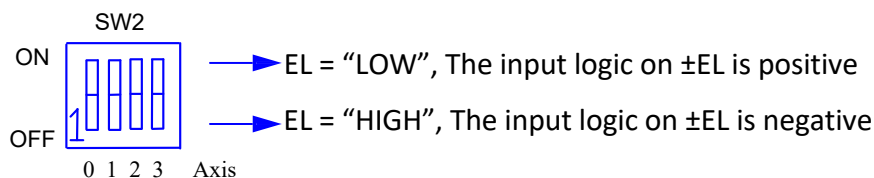
Board ID	SW1 Setting(ON=1)			
	1	2	3	4
0	OFF	OFF	OFF	OFF
1	ON	OFF	OFF	OFF
2	OFF	ON	OFF	OFF
3	ON	ON	OFF	OFF
4	OFF	OFF	ON	OFF
5	ON	OFF	ON	OFF
6	OFF	ON	ON	OFF
7	ON	ON	ON	OFF
8	OFF	OFF	OFF	ON
9	ON	OFF	OFF	ON
10	OFF	ON	OFF	ON
11	ON	ON	OFF	ON

### 3.12. SW2 Switch setting for the logic of EL

SW2 switch is used to set the EL limit switch type. It will be "HIGH" when you switch off. The default setting of EL switch type is "LOW".

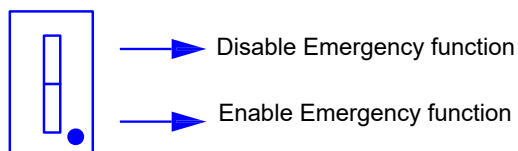
For your safety, please set so that the end limit becomes active when a breakdown or a disconnection occurs.

Setting of SW2 switch on board



### 3.13. SW3 Switch Setting to Enable Emergency Input

SW3 switch is used to enable the Emergency signal from CN1. The default setting is "Disable Emergency function". For details, see section 4.15.



## 4. Signal Connections

In this chapter, connections of all of the I/O signals are described. See the contents of this chapter before wiring PPCle-8443 and the motor drivers.

This chapter describes the following items:

Section 4.1	Pulse output signals OUT and DIR
Section 4.2	Encoder feedback signals EA, EB and EZ
Section 4.3	Origin signal ORG
Section 4.4	End-limit signals PEL and MEL
Section 4.5	Ramping-down & PCS
Section 4.6	In-position signal INP
Section 4.7	Alarm signal ALM
Section 4.8	Deviation counter clear signal ERC
Section 4.9	General-purpose output signal SVON
Section 4.10	General-purpose input signal RDY
Section 4.11	Position compare output terminal: CMP
Section 4.12	Position latch input terminal: LTC
Section 4.13	Pulser input signals PA and PB
Section 4.14	Simultaneous start/stop signals STA and STP
Section 4.15	Emergency input EMG
Section 4.16	General purpose Input / Output EDI and EDO
Section 4.17	Power supply configuration



## 4.1. Pulse Output Signals OUT and DIR

As for the pulse output signals, the differential signal output for OUT signal and DIR signal is available for each of 4 axes. For the signal logic settings of OUT signals and DIR signals, see subsection 5.1.1. Pulse output signal of CN2 is as shown in the table below:

CN2 Terminal No.	Signal Name	Description	Axis No.
3	OUT0+	Pulse signals (+)	0
4	OUT0-	Pulse signals (-)	0
5	DIR0+	Direction signal (+)	0
6	DIR0-	Direction signal (-)	0
21	OUT1+	Pulse signals (+)	1
22	OUT1-	Pulse signals (-)	1
23	DIR1+	Direction signal (+)	1
24	DIR1-	Direction signal (-)	1
53	OUT2+	Pulse signals (+)	2
54	OUT2-	Pulse signals (-)	2
55	DIR2+	Direction signal (+)	2
56	DIR2-	Direction signal (-)	2
71	OUT3+	Pulse signals (+)	3
72	OUT3-	Pulse signals (-)	3
73	DIR3+	Direction signal (+)	3
74	DIR3-	Direction signal (-)	3

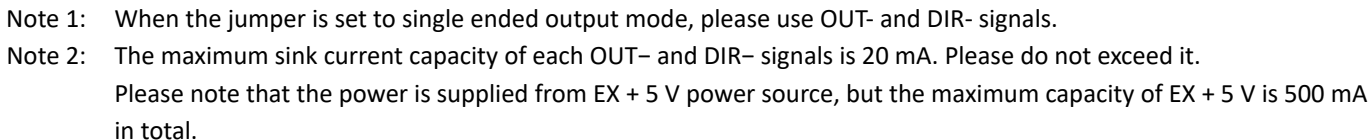
For OUT signal and DIR signal, either Line driver output or single-ended output can be selected with jumpers. The correspondence of jumpers J1 to J8 to each signal is as shown in the table below:

Output Signal	Corresponding Jumper
OUT0+	J1
DIR0+	J2
OUT1+	J3
DIR1+	J4
OUT2+	J5
DIR2+	J6
OUT3+	J7
DIR3+	J8

.....

1-2 short: Line driver output

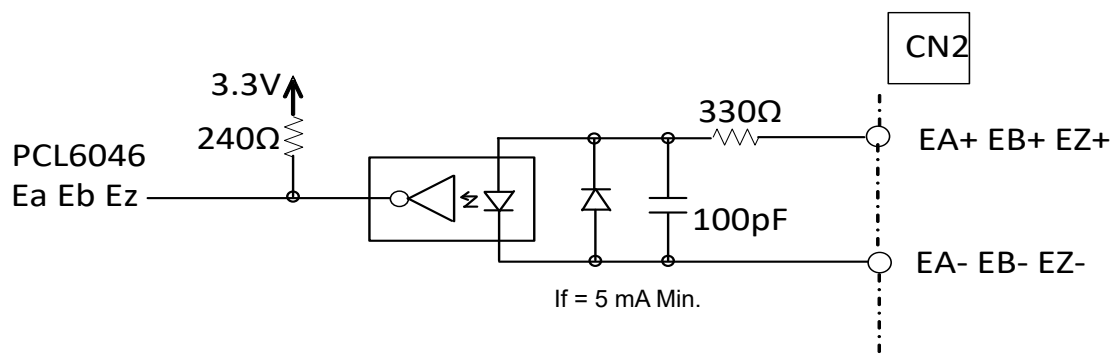
2-3 short: Single-ended output



Encoder feedback signals include EA, EB, and EZ signals. Each axis has 6 signal terminals for differential pairs of A-phase (EA), B-phase (EB), and Z-phase (EZ) input. EA and EB are used for position counting and EZ is used for Z phase input (zero position index). Signal names, terminal numbers and axis number are shown in the following table.

CN2 Terminal No	Signal Name	Axis No.	CN2 Terminal No	Signal Name	Axis No.
17	EZ0+	0	67	EZ2+	2
18	EZ0–	0	68	EZ2–	2
35	EZ1+	1	85	EZ3+	3
36	EZ1–	1	86	EZ3–	3

The input circuits of EA, EB, and EZ signals are as follows:

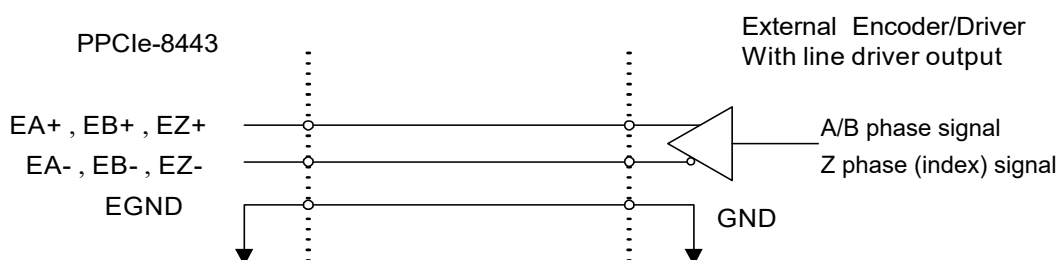


The differential pair signals (EA+, EA-), (EB+, EB-), (EZ+, EZ-) of the encoder input must have a potential difference of at least 3.5 V or higher. Therefore, please pay attention to the driving capability when connecting with encoder feedback or motor driver feedback. The differential signal pairs will be converted to digital signals EA, EB, EZ to connect with PCL 6046 chip.

The following are the two examples of connecting an input circuit with an external encoder or motor driver.

#### Connection to Line driver output

To drive the encoder input, the driver output must provide at least 3.5 V across the differential pairs with 6 mA driving capability. The GND levels of the two sides need to be tied together.

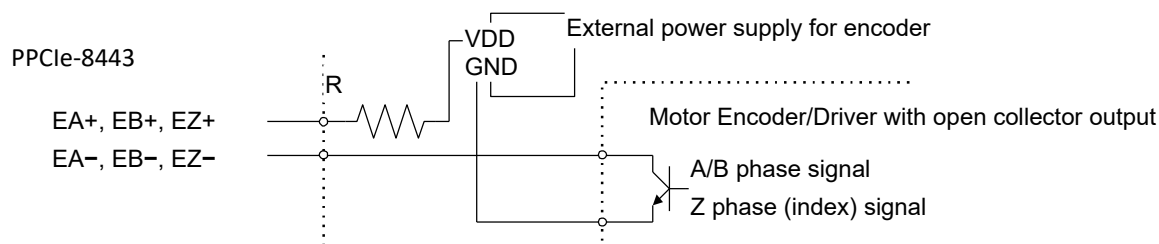


#### Connection to Open Collector Output

To connect with open collector output, an external power supply is necessary. The connection between PPCle-8443, the encoder, and the power supply is shown in the following diagram. Please note that the external current limit resistor R is necessary to protect the PPCle-8443 input circuit in accordance with the signal voltage. The following table shows the suggested resistor value according to the encoder power supply.

Encoder Power (VDD)	External Resistor R
+ 5 V	0 (None)
+ 12 V	1.8 kΩ
+ 24 V	4.3 kΩ

If = 6 mA Max



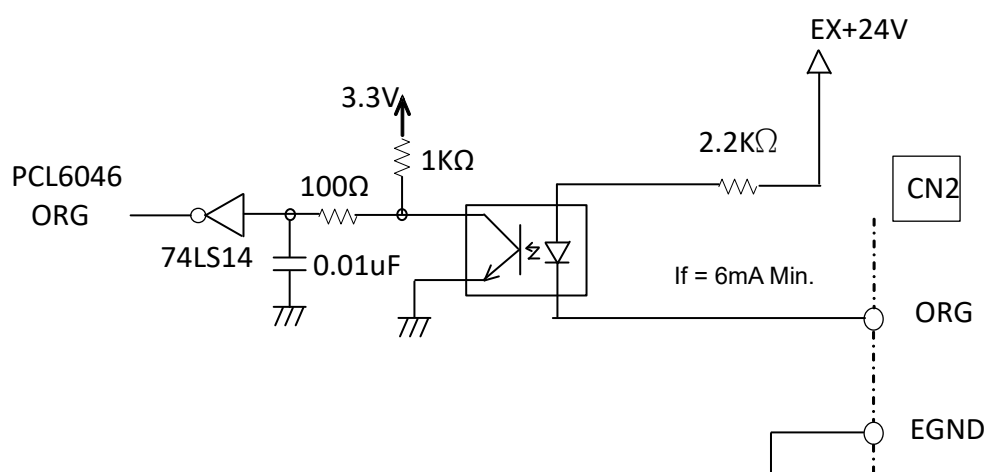
For more detail operations of encoder feedback signals, see subsection 5.4.2.

### 4.3. Origin Signal ORG (Home return)

The origin signals (ORG0 ~ ORG3) are used as input signal for the origin of a mechanism. The relative signal names, terminal numbers and axis numbers are shown in the following table:

CN2 Terminal No	Signal Name	Axis No
41	ORG0	0
47	ORG1	1
91	ORG2	2
97	ORG3	3

The input circuit of ORG signal is shown as follows. Usually, ORG signal is used to indicate the mechanical origin position of an axis. Internally, a CR filter circuit is built-in to prevent noise issues.



When the motion controller is operated in the Home return mode, ORG signal is used to stop control output signals (OUT and DIR signals). For detail operations of ORG, see subsection 5.3.3.



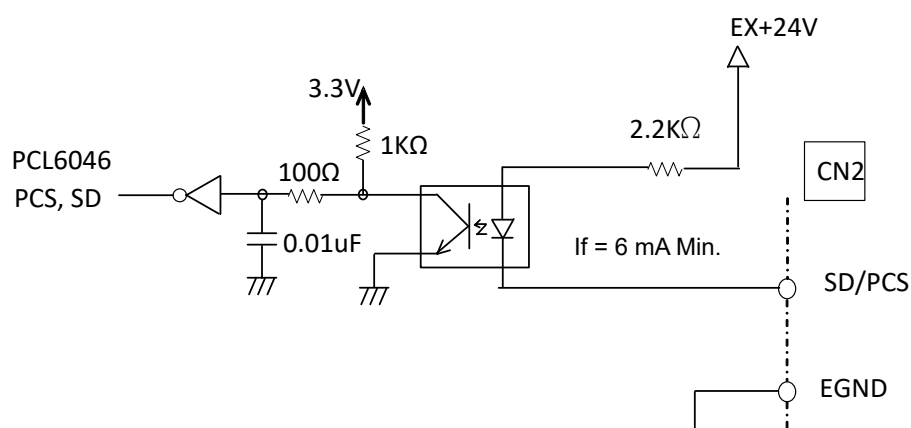
## 4.5. Ramping-Down & PCS

There are SD / PCS signals for each of 4 axes. The relative signal name, terminal number and axis number are shown in the following table:

CN2 Terminal No	Signal Name	Axis No
40	SD / PCS0	0
46	SD / PCS1	1
90	SD / PCS2	2
96	SD / PCS3	3

The input circuit of the signals is shown as follows.

Normally, SD signal is used to input a ramp down signal that decelerates the motor speed from high speed to low speed. Internally, the CR filter circuit is built-in to prevent noise issues. For details of SD/ PCS signals, see subsection 5.3.1.

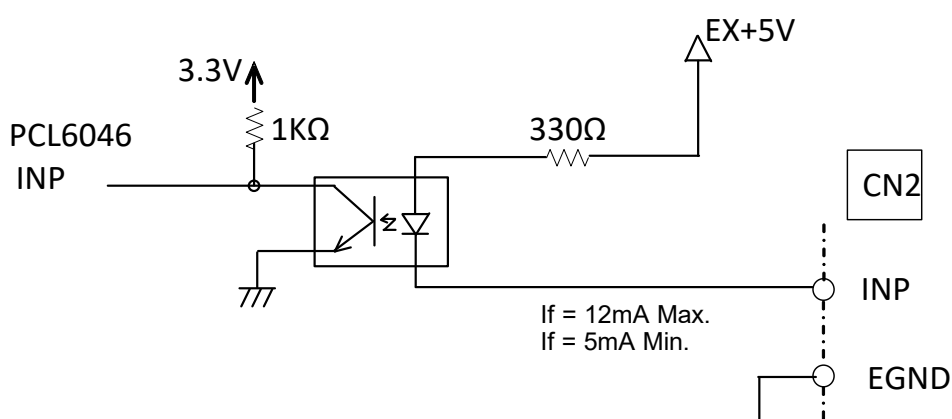


## 4.6. In-Position Signal: INP

In-position signal (INP) from a servo motor driver indicates the deviation between command pulse and feedback pulse is within the set range, and input the positioning completion output signal of the motor driver. The relative signal name, terminal number and axis number are shown in the following table:

CN2 Terminal No	Signal Name	Axis
10	INP0	0
28	INP1	1
60	INP2	2
78	INP3	3

The input circuit of the INP signals is shown in the following diagram



The in-position signal is usually input from a servomotor driver. For the INP signal operation, see subsection 5.2.1.

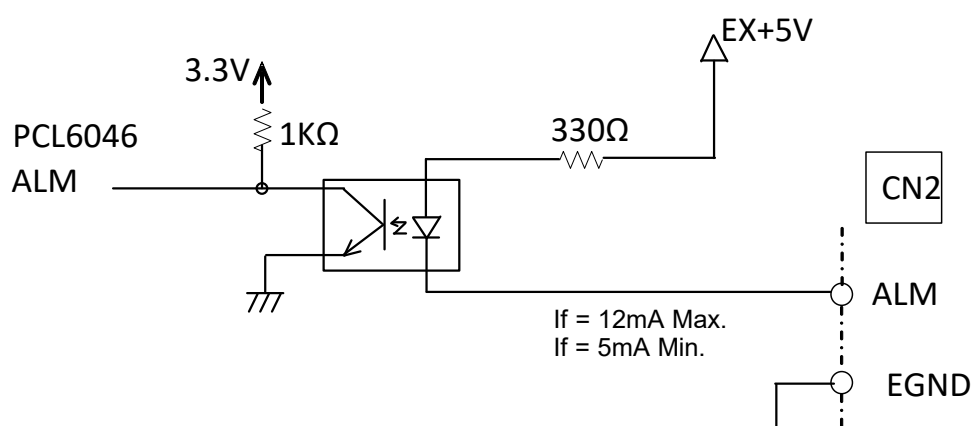
## 4.7. Alarm Signal: ALM

The alarm signal ALM is used to indicate the alarm status from the servo driver. The relative signal name, terminal number and axis number are shown in the following table:

CN2 Terminal No	Signal Name	Axis No
9	ALM0	0
27	ALM1	1
59	ALM2	2
77	ALM3	3

The input circuit of an alarm signal is shown in the following diagram. The ALM signals are usually output from servomotor drivers.

For more details of ALM operation, see subsection 5.2.2.





## 4.8. Deviation Counter Clear Signal: ERC

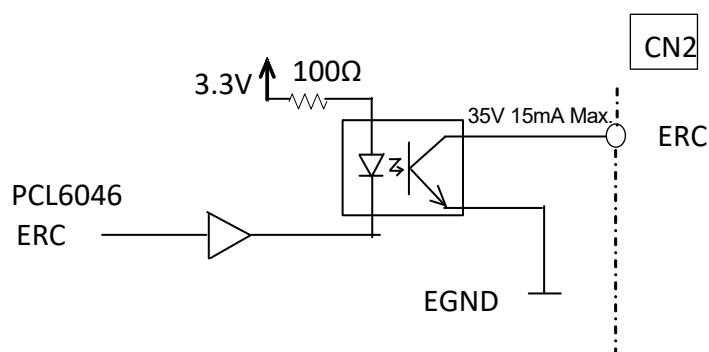
The deviation counter clear (ERC) signal is active in the following three situations:

1. When a home return operation is completed
2. When an error input signal such as end-limit switch, alarm switch, or emergency switch, is turned ON.
3. When an emergency stop command is issued by software.

The relative signal names, terminal numbers and axis numbers are shown in the following table:

CN2 Terminal No	Signal Name	Axis No
8	ERC0	0
26	ERC1	1
58	ERC2	2
76	ERC3	3

The ERC signal is used to clear the deviation counter of a servomotor driver. The ERC output circuit is an open collector output circuit, and it has the maximum 35 V output power with 15 mA driving capability. For more details of ERC operation, see subsection 5.2.3.

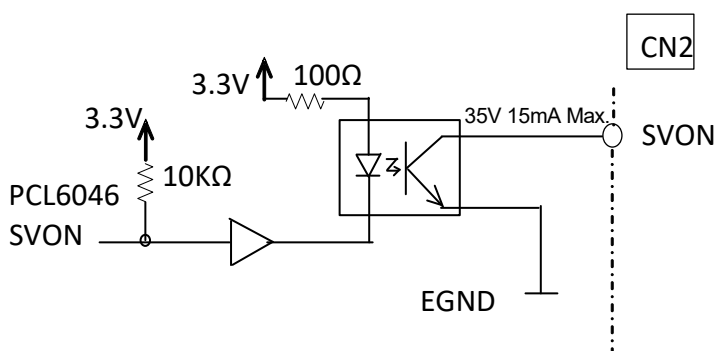


## 4.9. General Purpose Output Signal SVON

The SVON signals are mainly used for servomotor-on control; however they can be used as general- purpose output signals. The relative signal name, terminal number and axis number are shown in the following table:

CN2 Terminal No	Signal Name	Axis No
7	SVON0	0
25	SVON1	1
57	SVON2	2
75	SVON3	3

The SVON output circuit is an open collector output circuit, and it has the maximum 35 V output power with 15 mA driving capability. For more details of SVON operation, see subsection 5.2.4.

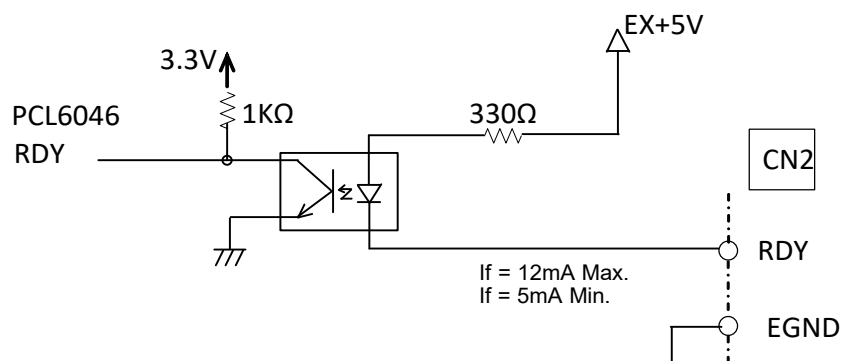


## 4.10. General Purpose Input Signal: RDY

The RDY signals are mainly used for motor driver ready input (preparation ready), however, they can be used as general purpose input signals. The relative signal name, terminal number and axis number are shown in the following table:

CN2 Terminal No	Signal Name	Axis No
11	RDY0	0
29	RDY1	1
61	RDY2	2
79	RDY3	3

The input circuit of RDY signal is shown in the following diagram:



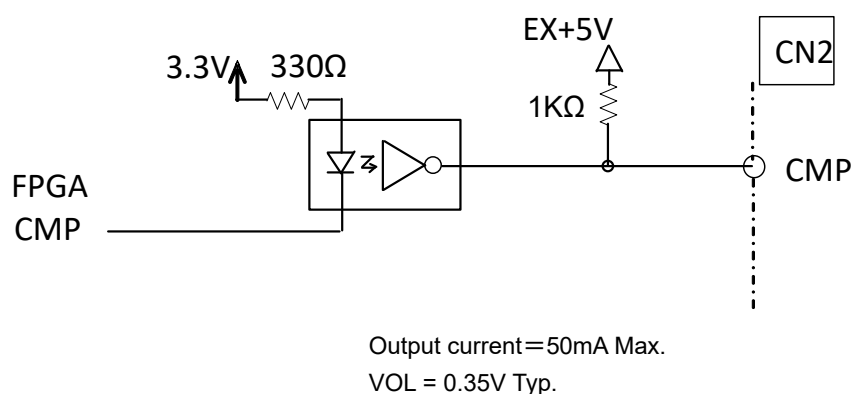
## 4.11. Position Comparator Output: CMP

The PPCle-8443 provides two position comparison output channels, and they are only available for the first two axes (Axis 0 and Axis 1). A pulse signal will be generated as a comparing output when the encoder counter matches the pre-set value set by a user.

The relative signal name, terminal number and axis number are shown in the following table:

CN2 Terminal No	Signal Name	Axis No
39	CMP0	0
45	CMP1	1

The output circuit of CMP signal is shown in the following diagram:



Note: CMP output specification can be set either as normal low (rising edge) or as normal high (falling edge). The default setting is normal high. See function `_8443_set_trigger_type()` in section 7.17 for details. Also, CMP terminal can be used as a general purpose output.

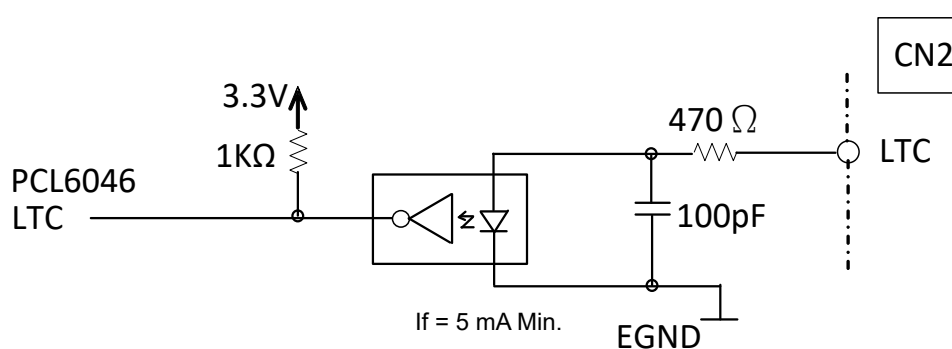
## 4.12. Position Latch Input: LTC

The PPCle-8443 provides two position latch input channels, and they are only available for the last two axes, (Axis 2 and Axis 3). The LTC signal can trigger to latch a counter value, so that you can obtain a precise position by hardware without going through software.

The relative signal name, terminal number and axis number are shown in the following table:

CN2 Terminal No	Signal Name	Axis No
89	LTC2	2
95	LTC3	3

The input circuit of LTC signal is shown in the following diagram. Please use input voltage range of 0 ~ 5 V.

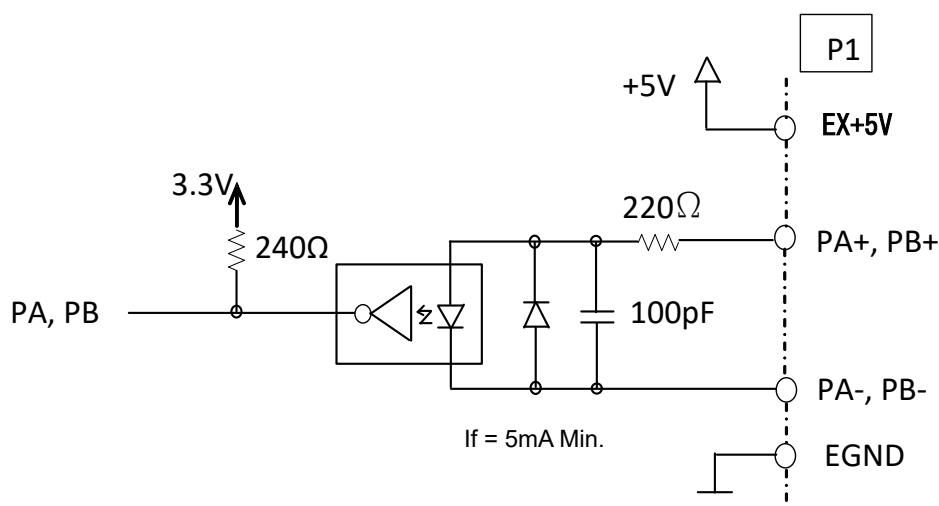


## 4.13. Pulser Input Signals: PA and PB

PPCLe-8443 can accept input signals from a pulser through P1 connector. The signal specification of the pulser signal is 90 degree phase difference (A/B phase) input, which is the same as encoder signal. The motor can be operated following the positioning pulses generated manually by the pulser.

P1 Terminal No	Signal Name
2	PA +
3	PA -
4	PB +
5	PB -

PA and PB terminals of the connector P1 are directly connected to PA and PB terminals of PCL6046. The interface circuits are shown as follows:



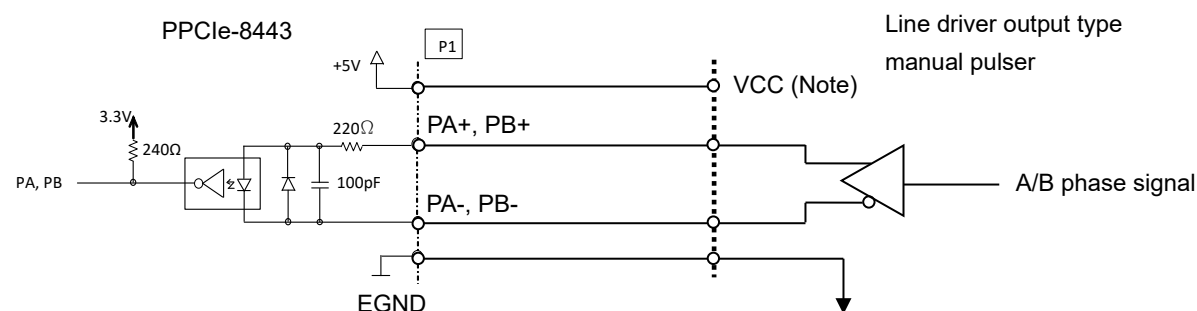
Please note that the voltage across every differential pair of pulser input signals (PA+, PA-) and (PB+, PB-) should be at least 3 V or higher.

You can select which axes to be operated by the function `_8443_disable_pulser_input()`. See section 7.11 for details.

### Connection with a manual pulser

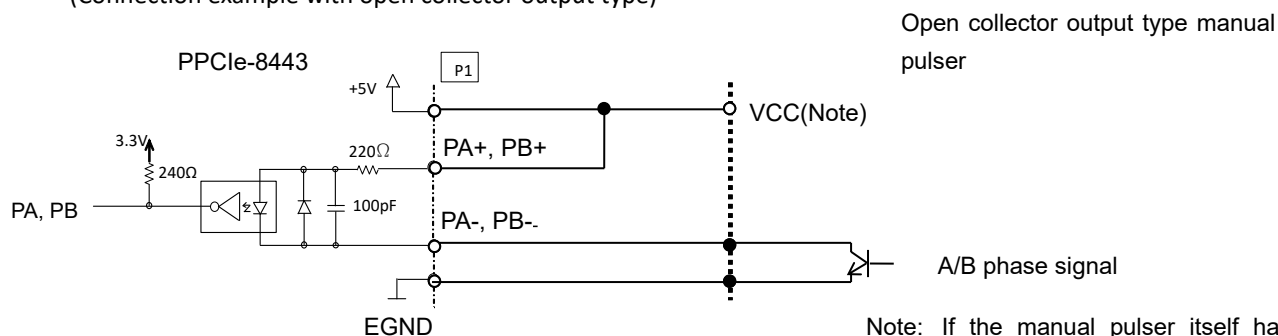
To operate a pulser input of PPCle-8443 board, at least 3 V of differential voltage is required in the differential pair of the differential driver output. Please make GNDs on both sides in common.

(Connection example with the line driver output type)



Note: If the manual pulser itself has a power supply, it is not necessary to connect the power supply line (VCC) with it.

(Connection example with open collector output type)



Note: If the manual pulser itself has a power supply, it is not necessary to connect the power supply line (VCC) with it.

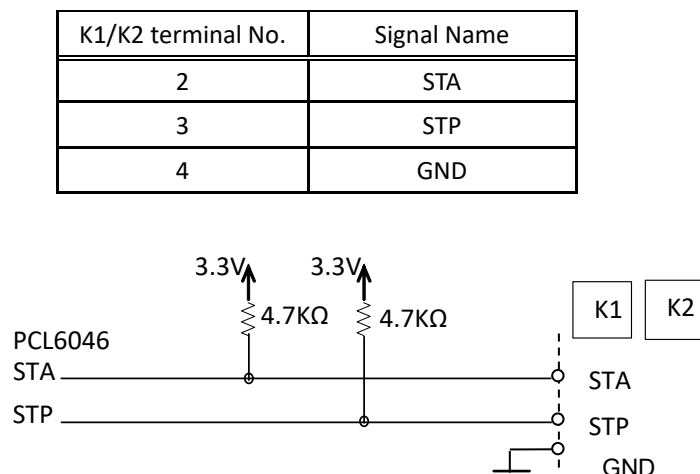
See subsection 5.1.12 for operating method using a manual pulser.

For connection, we recommend using a shielded cable or a twisted cable in terms of noise immunity.

## 4.14. Simultaneous Start/Stop Signals: STA and STP

The PPCle-8443 provides STA and STP signals, which enable simultaneous start/stop of motions with multiple axes. The STA and STP signals are on connectors K1 and K2.

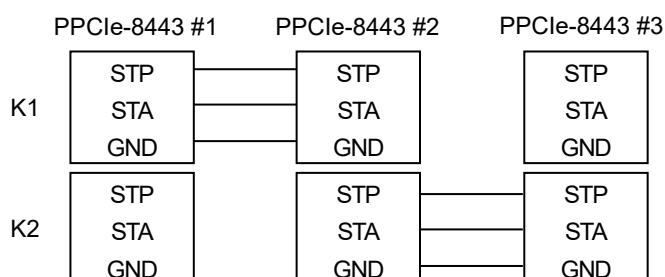
The following diagram shows the on-board circuits. STA and STP signals of the other boards are tied together respectively. The internal circuit is shown in the figure below.



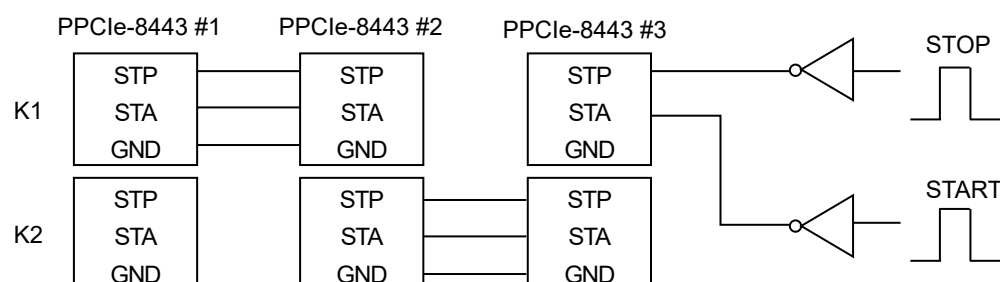
The STP and STA signals are both for input and output. By the software control, signals can be generated from any one of the PCL6046 chip on board, and other chip can perform start and stop simultaneously by receiving the signals if properly programmed.

You can also use an external open collector or switch to drive the STA / STP signals for simultaneous start / stop.

If there are two or more PPCle-8443 boards, cascade K1 / K2 connectors of all boards for simultaneous start/stop control. Since the signals are connected in the board, connect K1 / K2 as follows.



To make an external signal initiate simultaneous start or stop, connect the 7406 (open collector) or the equivalent circuit as follows:

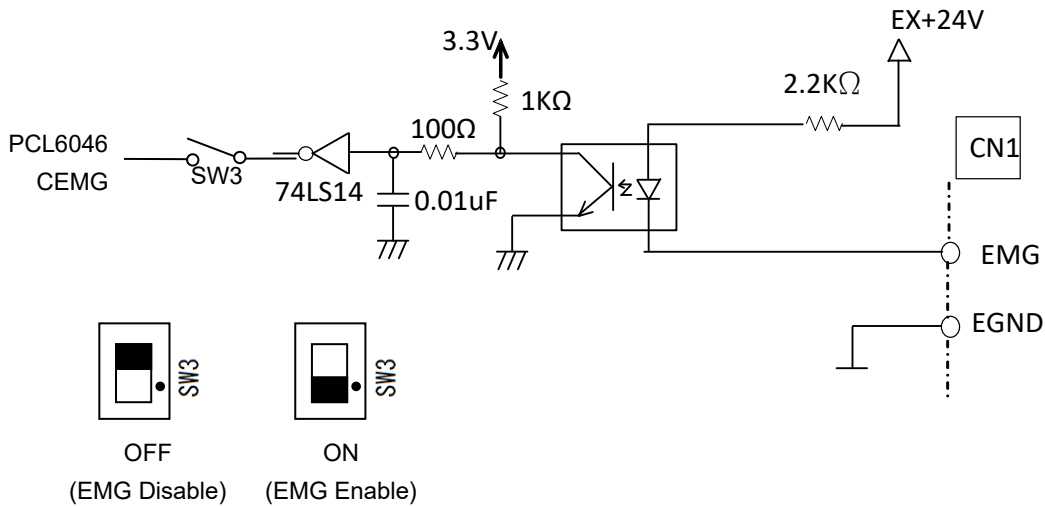


Note: STA / STP signals will turn ON by setting it to 0.8 V or less with reference to the GND signals of No.4 terminal in K1 / K2 connector. It is the PCIe bus power supply GND. For the specification of the GND signal, please check the power supply configuration (Section 4.17).

## 4.15. Emergency Input EMG

PPCLe-8443 provides an emergency input, EMG, in CN1. Once EMG is input, all axes will stop immediately to prevent the damage of the machine.

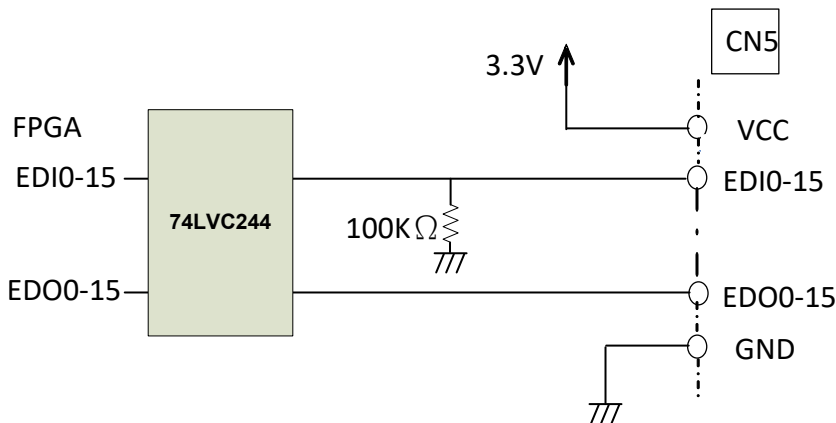
You can use SW3 to set either enable / disable the emergency signal from CN1. The input circuit of EMG signal is shown in the following diagram. Internally, a CR filter circuit is built in to prevent noise issues.



## 4.16. Extended General-Purpose Input / Output: EDI and EDO

PPCLe-8443 provides 16 /16 general-purpose digital input / output in CN5. The relative signal name, terminal number and axis number are shown in the following table.

The circuit of general-purpose input/output signals is shown in the following diagram:



The specifications of digital input and output of general-purpose signals are listed as follows:

[EDI]

Input Current: 24 mA (max);  $V_{IH}$  = 2.0 V (min);  $V_{IL}$  = 0.8 V (max);  $V_I$  = 5.5 V (max)

[EDO]

Output Current: 24 mA (max);  $V_{OH}$  = 3.3 V (max);  $V_{OL}$  = 0 V (min)

Note: EDI / EDO are GND based signals of PCIe bus power supply. See the power supply configuration (Section 4.17).

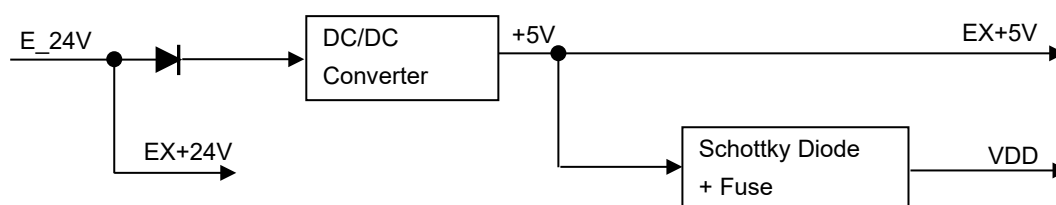


## 4.17. Power Supply Configuration

The power supply of PPCle-8443 is shown as follows:

Signal	Function	Description
E_24V	24 V External power input	Externally supplied 24 V power supply Input voltage: + 24 V DC $\pm$ 5%, Required power capacity: 500 mA Max Insulated from the bus power supply.
EX+24V	24 V Internal power supply	24 V power supply used for external signal input supplied from external 24 V input power supply. Insulated from the bus power supply.
VDD	+ 5 V Power supply output	5 V power supply that can be used externally. It is generated from an external 24 V power supply input. Output voltage: + 5 V DC $\pm$ 5%, Output capacity : VDD and EX + 5 V; total 500 mA Max Insulated from the bus power supply. Overcurrent and reverse voltage protection circuit is built-in.
EX+5V	+ 5 V Power supply output	5 V power supply that can be used externally. It is generated from an external 24 V power supply input. Output voltage: + 5 V DC $\pm$ 5%, Output capacity : VDD and EX + 5 V; Total 500 mA Max Insulated from the bus power supply. No protection circuit.
EGND	External power supply GND	GND for external power supply(EX+24V and EX+5V) Insulated from the bus power supply.
VCC	+ 3.3 V bus power supply	3.3 V for board internal control supplied from the bus. Used with CN5 for extended general purpose input / output.
B12V	+ 12 V bus power supply	12 V for board internal control supplied from the bus. Power supply capacity: 250 mA Max No output to the outside.
GND	Bus power supply GND	GND for the board internal control supplied from the bus Used with extended general-purpose input / output CN5 and K1 / K2 for simultaneous start / stop

Note 1: Both VDD and EX + 5 V are generated from the external power supply 24 V input, and they are 5 V voltage which can be used externally. However overcurrent and reverse voltage protection circuit is built in only in VDD.



Note 2: When connecting an external device (a driver or an external encoder) to the 5 V power supply, pay attention to the power capacity.

Note 3: When using the bus power supply, pay extra attention to the noise inflows in wiring or power supply capacities.

## 5. Operation Theorem

This chapter describes the detail operations of the PPCle-8443 as follows:

Section 5.1:	Motion control mode
Section 5.2:	The Motor Driver Interface (INP, ERC, ALM, SVON, RDY)
Section 5.3:	Mechanical Input Interface and I/O Status (SD / PCS, EL, ORG)
Section 5.4:	Counters (EA, EB, EZ)
Section 5.5:	Multiple PPCle-8443 operation
Section 5.6:	Change Position or Speed On The Fly (Override Function)
Section 5.7:	Comparator and Latch
Section 5.8:	Backlash Compensator and Vibration Suppression
Section 5.9:	Software Limit Function
Section 5.10:	Interrupt Control
Section 5.11:	Idling Control

### 5.1. Motion control mode

In this section, the pulse output signal configurations and the motion control modes are described.

Subsection 5.1.1	Output Pulse Mode
Subsection 5.1.2	Velocity Mode Operation
Subsection 5.1.3	Positioning Operation for Single Axis
Subsection 5.1.4	S-curve Profile Acceleration / Deceleration Operation
Subsection 5.1.5	Linear Interpolation for Two to Four Axes
Subsection 5.1.6	Circular Interpolation for Two Axes
Subsection 5.1.7	Circular Interpolation with Acceleration / Deceleration Time
Subsection 5.1.8	Helical Interpolation
Subsection 5.1.9	The Relationship between Velocity and Acceleration Time
Subsection 5.1.10	Continuous Operation
Subsection 5.1.11	Home Return Operation (Origin Return)H
Subsection 5.1.12	Manual pulser operation
Subsection 5.1.13	Timer Mode
Subsection 5.1.14	Pulser Interpolation

### 5.1.1 Output Pulse Mode

PPC1e-8443 uses a pulse command to control servo/stepper motors via drivers. The pulse command consists of two signals: OUT and DIR. There are three output pulse modes: 1) Common pulse output mode (OUT/DIR), 2) 2 pulse output mode (CW/CCW type pulse output), and 3) 90 degree phase difference output.

The software function: `_8443_set_pls_outmode()` is used to program the output pulse mode. The mode vs. signal type of OUT and DIR terminal are shown in the following table:

Mode	OUT terminal	DIR terminal
Common pulse mode (OUT/DIR)	Pulse signal	Direction signal (Level)
2 pulse mode (CW/CCW)	CW signal	CCW signal
90-degree phase difference (A/B phase) mode	90-degree phase difference signal	90-degree phase difference signal

The interface characteristics of these signals can be set either in differential line driver output or single-ended output. See section 4.1 for the jumper setting for the output.

Three types of pulse output modes are explained below.

On this board, the direction in which the command pulse counts up is CW, and the direction to count down is CCW. Please note that the direction of motor rotation and the direction of CW and CCW of this board may be different depending on the motor manufacturer.

#### Common pulse mode (OUT / DIR)

In this mode, OUT signal will be the command pulse.

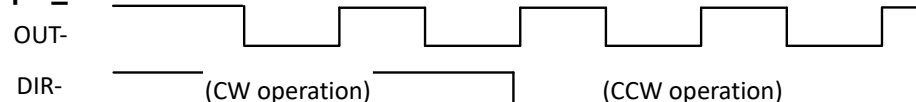
The number of OUT pulse signal will represent the relative “distance” or “position”, and the frequency of the OUT pulse signal will represent the command for “speed” or “velocity”.

The DIR signal will represent the command direction of positive (+) or negative (-).

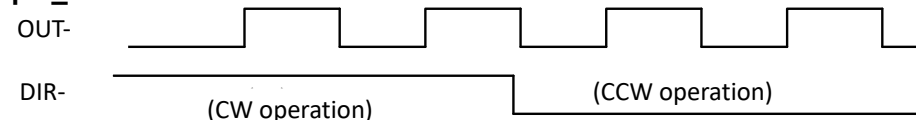
The following figures show the minus side output waveforms of differential signals.

You can select the logic of pulse signal from the following:

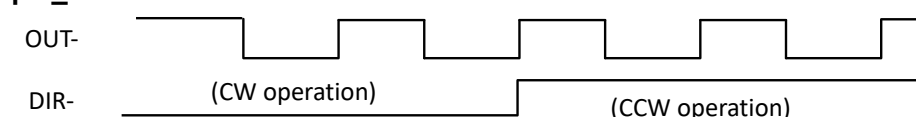
#### pls\_outmode = 0:



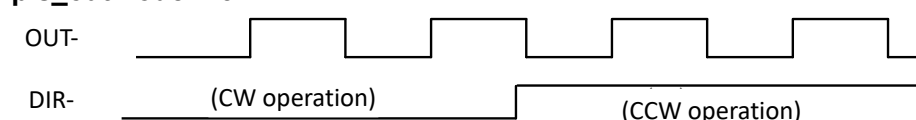
#### pls\_outmode = 1:



#### pls\_outmode = 2:



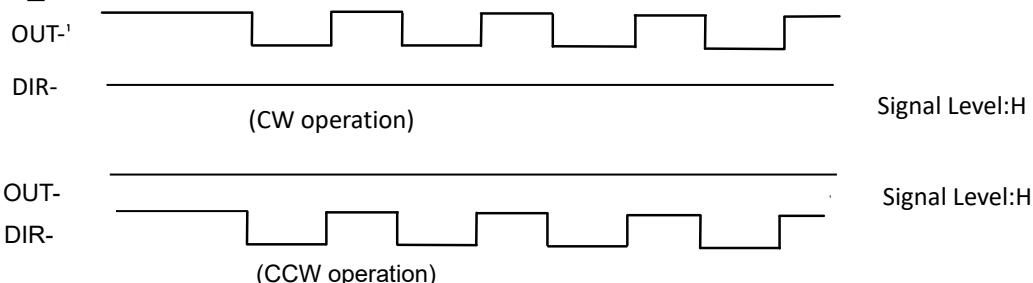
#### pls\_outmode = 3:



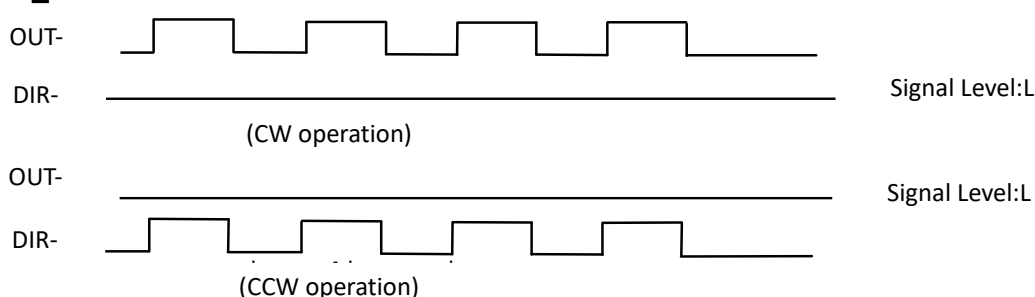
## 2 pulse mode (CW/CCW)

In this mode, the waveform of the OUT and DIR terminals represent CW (clockwise) and CCW (counter clockwise) command pulse output respectively. Pulse output from CW terminal makes a motor move in positive direction (CW), while pulse output from CCW terminal makes a motor move in negative direction (CCW). The following diagram shows the output waveforms of differential signals CW and CCW commands.

### pls\_outmode = 4:



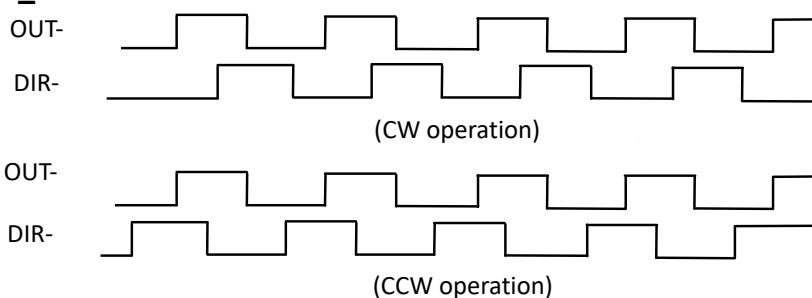
### pls\_outmode = 5:



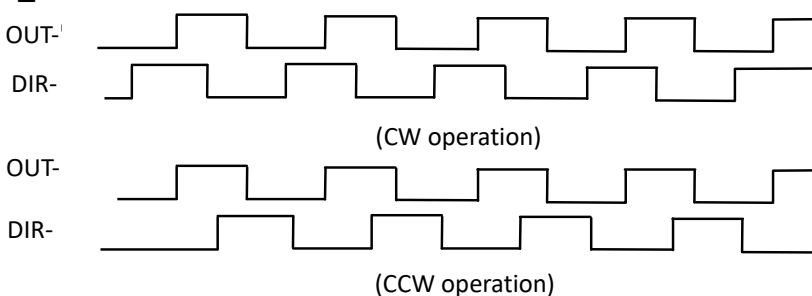
## 90-degree phase difference (A / B phase) mode

In this mode, OUT signal is output by 90 degree phase ahead (behind) of the DIR signal. “Ahead” or “behind” of phase difference between the two signals will determine the direction of motor rotation. When the OUT signal is 90 degree phase ahead of DIR signal in mode 6, the motor is running in CW direction. In opposite, when the OUT signal is 90 degree phase behind of DIR signal in mode 7, the motor is running in CW direction. The following figures show the minus side output waveforms of differential signals.

### pls\_outmode = 6:



### pls\_outmode = 7:



#### Related functions:

**\_8443\_set\_pls\_outmode():** See section 7.4.

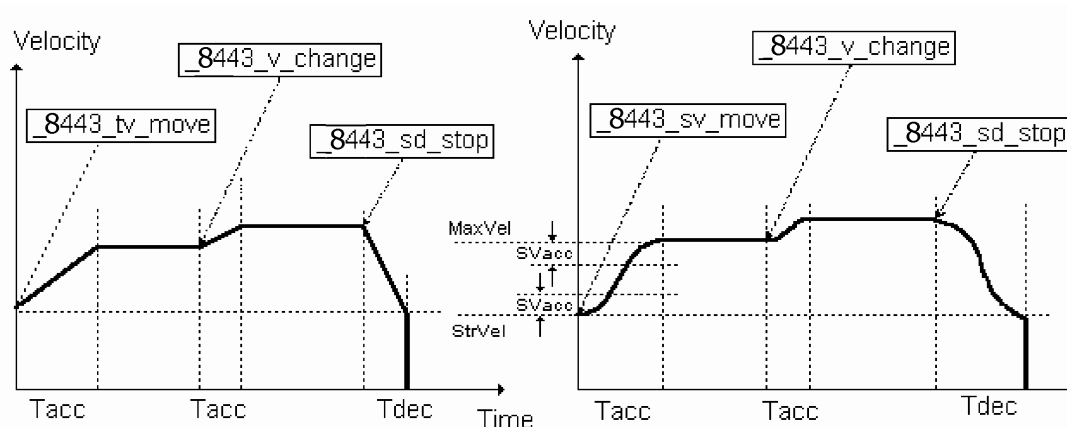
## 5.1.2 Velocity Mode Operation

This mode is used to operate one axis motor in velocity mode. Output pulse trains accelerate from the starting velocity (StrVel) to the specified constant velocity (MaxVel). **\_8443\_tv\_move()** function is used to accelerate constantly while **\_8443\_sv\_move()** function is used to accelerate with S-curve accel/decel. The pulse output rate will be maintained at maximum velocity until another velocity command is set or stop command is issued.

**\_8443\_v\_change()** is used to change speed during operations. Before this function is applied, be sure to call **\_8443\_fix\_speed\_range()**. See section 5.6 for more detail explanation. The **\_8443\_sd\_stop()** is used to decelerate and stop. **\_8443\_emg\_stop()** function is used to immediately stop the motion.

The speed change function and the speed setting of stop function follow the setting of the initial operation command such as tv\_move or sv\_move function. The velocity profiles with the functions are shown as follows:

Note: The speed change function and stop functions can be used for **positioning operation** (for single axis positioning; see subsection 5.1.3 and S-curve profile operation, see subsection 5.1.4) or **home return operation** (see subsection 5.1.11).



### Related functions:

**\_8443\_tv\_move(), \_8443\_sv\_move(), \_8443\_v\_change(),**  
**\_8443\_sd\_stop(), \_8443\_emg\_stop(), \_8443\_fix\_speed\_range(),**  
**\_8443\_unfix\_speed\_range()**

: See section 7.5

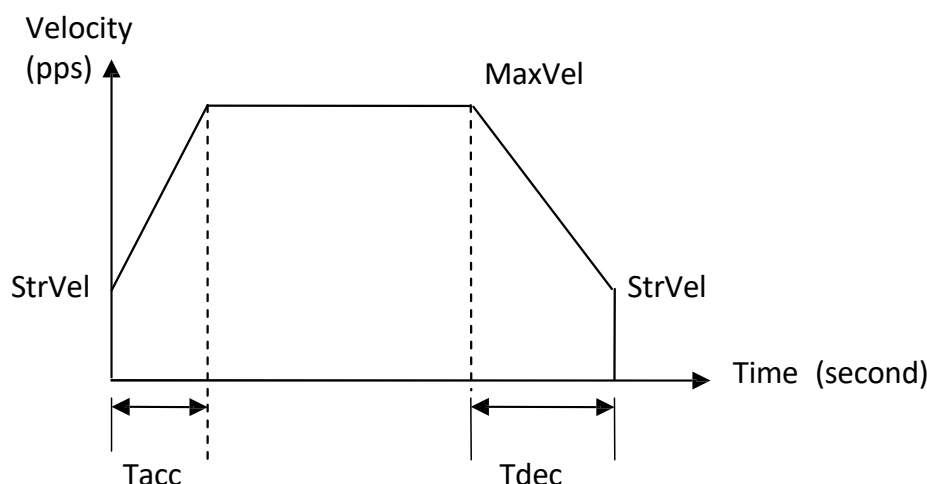
### 5.1.3 Positioning Operation for Single Axis

This mode is used to move an axis to a specified position (or distance) with a specified velocity profile. Two kinds of motion profile, absolute and relative motions, can be performed.

In absolute mode, a target position is specified. In relative mode, relative distance (number of pulses) to the target position is specified.

In both absolute and relative mode, acceleration and deceleration can be set in different values. `_8443_motion_done()` function is used to check if the movement is completed.

The following figure shows the trapezoidal profile.



There are two trapezoidal point-to-point positioning functions supported by PPCle-8443.

In `_8443_start_ta_move()` function, the absolute target position is given in the unit of pulse. The physical length or angle of one movement is dependent on the resolution (moving amount per one pulse). In the absolute positioning mode, the value of the feedback counter is referenced and then commanded.

In advance, the external encoder feedback source need to be set by `_8443_set_feedback_src()` function. And the ratio between the command pulses and the external feedback pulses must be appropriately set by `_8443_set_move_ratio()` function.

In `_8443_start_tr_move()` function, the moving position (distance) in the relative mode is set in the unit of pulse.

Unsymmetrical trapezoidal velocity profile ( $T_{acc}$  is not equal to  $T_{dec}$ ) can be specified in both `_8443_start_ta_move()` and `_8443_start_tr_move()` functions.

The  $StrVel$  and  $MaxVel$  parameters are given in the unit of pulse per second (pps). The  $T_{acc}$  and  $T_{dec}$  parameters are given in the unit of second represent accel/decel time respectively. You have to know the physical meaning of “one pulse” to calculate the physical value of the relative velocity or acceleration parameters. The following formula gives the basic relationship between these parameters.

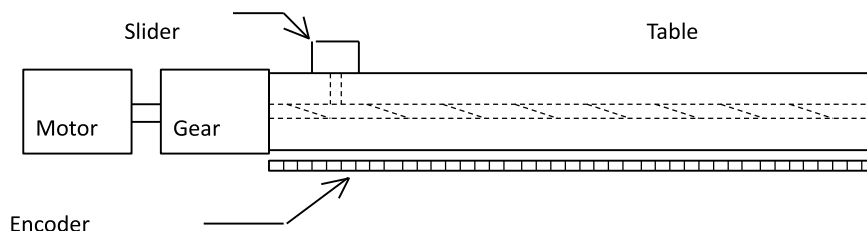
$$MaxVel = StrVel + accel * T_{acc};$$

$$StrVel = MaxVel + decel * T_{dec};$$

Accel/decel represents the acceleration / deceleration rate in unit of pps /sec. The area inside the trapezoidal profile represents the moving distance. The unit of velocity setting is pulses per second (pps). Usually, the unit of velocity in the manual of motor or driver is in rounds per minute (rpm). A simple conversion is necessary to match between these two units.

For example, if an incremental type encoder is mounted on the working table to measure the actual position of moving part. A servo motor is used to drive the moving part through a gear mechanism. The gear mechanism is used to convert the rotating motion of motor into linear motion. (See the following diagram). If the resolution of motor is 8000 pulses / round. The resolution of gear mechanism is 100 mm / round. (i.e., part moves 100 mm if motor turns one round). Then the resolution of command pulse will be 80 pulses / mm. If the resolution of encoder mounting on the table is 200 pulses / mm. Then you have to set the move ratio as  $200 / 80 = 2.5$  by the function:

```
_8443_set_move_ratio (axis, 2.5);
```



In case of absolute mode, pay attention to input mode (multiplication) setting of encoder feedback pulse.

If this ratio is not set before issuing the start moving command, it will cause problems when running in "Absolute Mode" since PPCle-8443 cannot recognize the actual absolute position during the operation.

#### Related functions:

**\_8443\_start\_ta\_move()**, **\_8443\_start\_tr\_move()**: See section 7.6

**\_8443\_motion\_done()**: See section 7.12

**\_8443\_set\_feedback\_src()**: See section 7.4

**\_8443\_set\_move\_ratio()**: See section 7.6

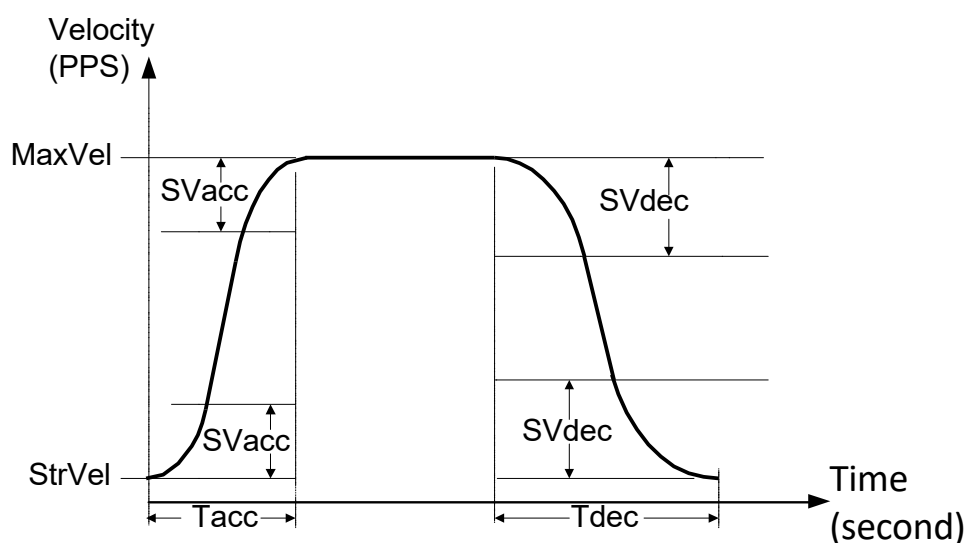
## 5.1.4 S-curve Profile Acceleration / Deceleration Operation

In S-curve acceleration / deceleration operations, vibrations are suppressed by smooth start of acceleration and smooth transition to the constant speed.

Smooth operations can reduce the load on the motor and the machine to extend the lives.

For S-curve acceleration / deceleration operation, setting of the following parameters is required.

- Pos:** target position in absolute mode (pulse)
- Dist:** moving distance in relative mode (pulse)
- StrVel:** start velocity, (PPS)
- MaxVel:** maximum velocity, (PPS)
- Tacc:** time for acceleration (StrVel → MaxVel) (s)
- Tdec:** time for deceleration (MaxVel → StrVel) (s)
- SVacc:** S-curve section during acceleration (pps)
- SVdec:** S-curve section during deceleration (pps)



In S-curve operation, acceleration / deceleration consists of one linear part and two S-curve sections.

In the first S-curve section, the velocity accelerates from StrVel to (StrVel + SVacc). Then, it is linear acceleration until the second S-curve section.

Finally, the velocity is accelerated from (MaxVel - SVacc) to MaxVel in the second S curve section. Deceleration will be the same rule as well.

### Note:

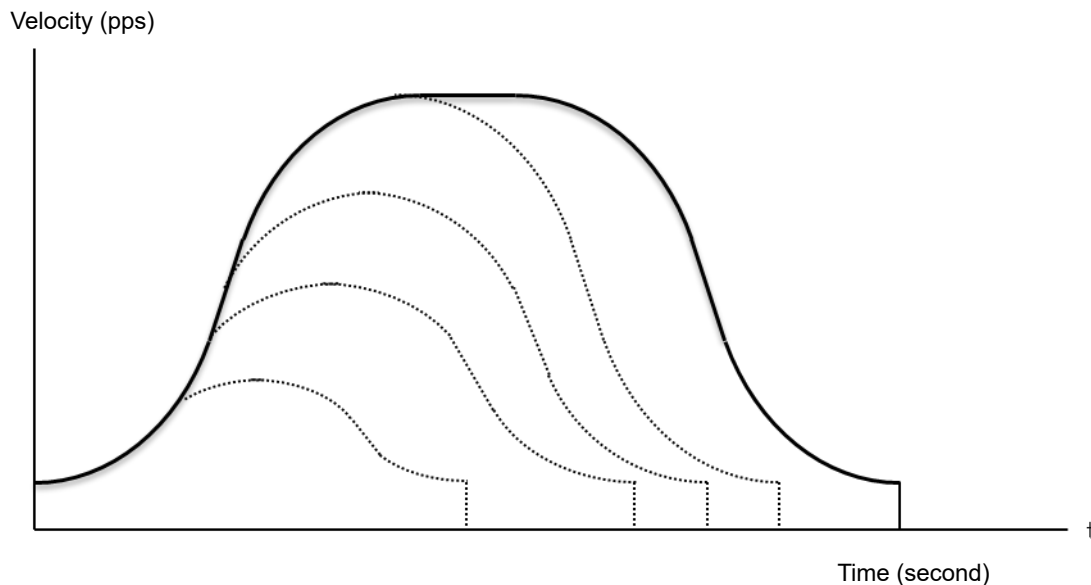
**If you want to remove the linear part, the SVacc/SVdec must be assigned "0".**

Remember that SVacc/SVdec is in unit of pps, and it should always keep in the range of  $[0 \sim (\text{MaxVel} - \text{StrVel})/2]$ , where "0" means no linear part. The S-curve profile motion functions are designed to always produce smooth motion.



If the time for linear / s-curve acceleration parameters combined with the final position does not allow the axis to reach the maximum velocity (i.e.: the moving distance is too small to reach MaxVel), the maximum velocity is automatically lowered (See the figure below).

The rule is to lower the value of MaxVel and Tacc, Tdec, SVacc, SVdec automatically, and keep StrVel, acceleration and jerk unchanged. It is also applicable to the trapezoidal profile motion.



#### Related functions:

\_8443\_start\_sr\_move(), \_8443\_start\_sa\_move(): See section 7.6

\_8443\_motion\_done(): See section 7.12

\_8443\_set\_feedback\_src(): See section 7.5

\_8443\_set\_move\_ratio(): See section 7.6

The following table shows the difference in single axis motion functions, including the Preset Mode (both trapezoidal and S-curve motions) and the constant velocity mode.

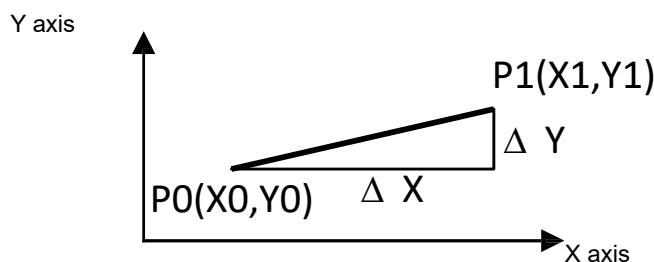
	Velocity Profile		Relative	Absolute
	Trapezoidal	S-curve		
<u>_8443_tv_move</u>	0	-	-	-
<u>_8443_sv_move</u>	-	0	-	-
<u>_8443_v_change</u>	0	0	-	-
<u>_8443_sd_stop</u>	0	0	-	-
<u>_8443_emg_stop()</u>	-	-	-	-
<u>_8443_start_ta_move</u>	0	-	-	0
<u>_8443_start_tr_move</u>	0	-	0	-
<u>_8443_start_sr_move</u>	-	0	0	-
<u>_8443_start_sa_move</u>	-	0	-	0

## 5.1.5 Linear Interpolation for Two to Four Axes

In this mode, any 2 of the 4 axes, 3 of the 4 axes, or all of the 4 axes may be chosen to perform a linear interpolation. "Interpolation between multi-axes" means these axes "start simultaneously, and reach their ending points at the same time". Linear means that the ratio of speed of every axis is a constant value. Notice that you cannot perform 2 groups of 2 axes linear interpolation in one board at the same time. But you can use one 2 axes linear and one 2 axes circular interpolation at the same time. If you want to stop one interpolation group, you can just use `_8443_sd_stop()` or `_8443_emg_stop()` for the first axis of the group as parameter to stop the all axes in this interpolation.

### 2 axes linear interpolation

As in the figure below, 2 axes linear interpolation means to move the XY (or any 2 of 4 axes) position from P0 to P1. The 2 axes start and stop simultaneously, and the path is a straight line.



The speed ratio along X-axis and Y-axis is ( $\Delta X : \Delta Y$ ), respectively, and the vector speed is:

$$\frac{\Delta P}{\Delta t} = \sqrt{\left(\frac{\Delta X}{\Delta t}\right)^2 + \left(\frac{\Delta Y}{\Delta t}\right)^2}$$

When calling the 2 axes linear interpolation functions, it is the vector speed to define the start velocity, **StrVel**, and the maximum velocity, **MaxVel** (Both trapezoidal and S-curve profile are available).

#### Example:

`_8443_start_tr_move_xy(0,30000.0,40000.0,1000.0,5000.0,0.1,0.2)`

It will execute X and Y axes (axes 0 & 1) of board 0 to perform a linear interpolation movement, in which:

$\Delta X = 30000$  pulse

$\Delta Y = 40000$  pulse

Start vector speed=1000 pps, X speed=600 pps, Y speed=800 pps

Max. vector speed=5000 pps, X speed=3000 pps, Y speed=4000 pps

Acceleration time = 0.1 sec

Deceleration time = 0.2 sec

There are two groups of functions that provide 2 axes linear interpolation.

1. The first group divides the 4 axes into XY (axis 0 & axis 1) and ZU (axis 2 & axis 3). By calling these functions, the target axes are already assigned.

```
_8443_start_tr_move_xy(), _8443_start_tr_move_zu(),
_8443_start_ta_move_xy(), _8443_start_ta_move_zu(),
_8443_start_sr_move_xy(), _8443_start_sr_move_zu(),
_8443_start_sa_move_xy(), _8443_start_sa_move_zu(),
```

: See section 7.7.

2. The second group allows you to assign the 2 target axes freely.

```
_8443_start_tr_line2(), _8443_start_sr_line2(),
_8443_start_ta_line2(), _8443_start_sa_line2(),
```

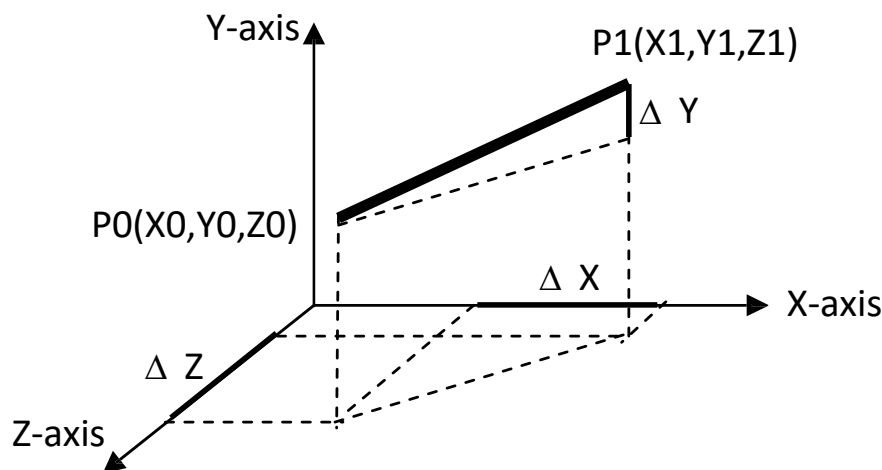
: See section 7.7.

The characters “t”, “s”, “r”, “a” after **\_8443\_start** means:

- t: Trapezoidal profile**
- s: S-curve profile**
- r: Relative motion**
- a: Absolute motion**

### 3 axes linear interpolation

Any 3 of the 4 axes of PPCle-8443 may perform 3 axes linear interpolation. As the figure below, 3 axes linear interpolation means to move the XYZ (if axes 0, 1, 2 are selected and assigned to be X, Y, Z respectively) position from P0 to P1 and start and stop simultaneously. The path is a straight line in space.



The speed ratio along X-axis, Y-axis and Z-axis is ( $\Delta X : \Delta Y : \Delta Z$ ) respectively, and the vector speed is:

$$\frac{\Delta P}{\Delta t} = \sqrt{\left(\frac{\Delta X}{\Delta t}\right)^2 + \left(\frac{\Delta Y}{\Delta t}\right)^2 + \left(\frac{\Delta Z}{\Delta t}\right)^2}$$

When calling the 3 axes linear interpolation functions, it is the vector speed to define the start velocity, **StrVel**, and the maximum velocity, **MaxVel**. Both trapezoidal and S-curve profiles are available.

For example:

`_8443_start_tr_line3(...,1000.0 /* ΔX*/, 2000.0 /*ΔY*/, 3000.0 /*ΔZ*/, 100.0 /*StrVel*/, 5000.0 /*MaxVel*/, 0.1 /*sec*/, 0.2 /*sec*/)`

ΔX = 1000 pulse

ΔY = 2000 pulse

ΔZ = 3000 pulse

Start vector speed=100 pps,      X speed =  $100 / \sqrt{14} = 26.7$  pps

Y speed =  $2 * 100 / \sqrt{14} = 53.3$  pps

Z speed =  $3 * 100 / \sqrt{14} = 80.1$  pps

Max vector speed=5000 pps,      X speed =  $5000 / \sqrt{14} = 1336$  pps

Y speed =  $2 * 5000 / \sqrt{14} = 2672$  pps

Z speed =  $3 * 5000 / \sqrt{14} = 4008$  pps

These functions related to 3 axes linear interpolation are listed as follows:

`_8443_start_tr_line3()`, `_8443_start_sr_line3()`

`_8443_start_ta_line3()`, `_8443_start_sa_line3()`

: See section 7.7.

The characters "t", "s", "r", "a" after `_8443_start` means:

t: Trapezoidal profile

s: S-curve profile

r: Relative motion

a: Absolute motion

#### 4 axes linear interpolation

In 4 axes linear interpolation, the speed ratio along X-axis, Y-axis, Z-axis and U-axis is (ΔX : ΔY : ΔZ : ΔU), respectively, and the vector speed is:

$$\frac{\Delta P}{\Delta t} = \sqrt{\left(\frac{\Delta X}{\Delta t}\right)^2 + \left(\frac{\Delta Y}{\Delta t}\right)^2 + \left(\frac{\Delta Z}{\Delta t}\right)^2 + \left(\frac{\Delta U}{\Delta t}\right)^2}$$

The functions related to 4 axes linear interpolation are listed below:

`_8443_start_tr_line4()`, `_8443_start_sr_line4()`

`_8443_start_ta_line4()`, `_8443_start_sa_line4()`

: See section 7.7.

The characters "t", "s", "r", "a" after **\_8443\_start** means:

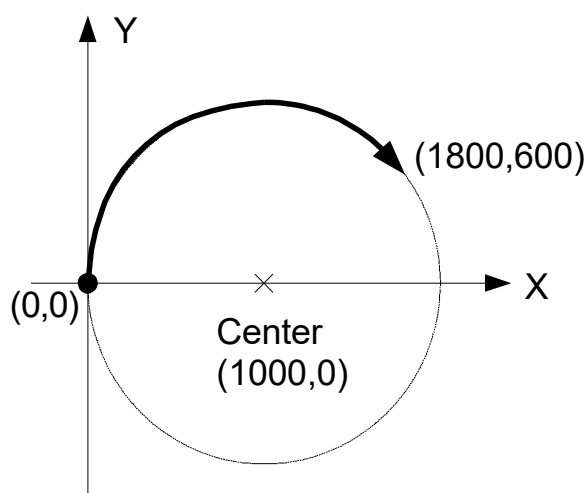
- t: Trapezoidal profile
- s: S-curve profile
- r: Relative motion
- a: Absolute motion

## 5.1.6 Circular Interpolation for Two Axes

Any 2 of the 4 axes of PPCle-8443 can perform a circular interpolation. As in the example below, the circular interpolation means X and Y (if axis 0 and 1 are selected, and are assigned to be X and Y respectively) axis simultaneous start from the initial point, (0, 0) and stop at the end point, (1800, 600). The path between them is an arc, and the max velocity, **MaxVel**, is the tangent speed.

Example:

```
_8443_start_a_arc_xy (0/*card No*/, 1000,0/*center X*/, 0/*center Y*/, 1800.0/* End X */, 600.0/*End Y  
*/ ,1000.0/* MaxVel */)
```

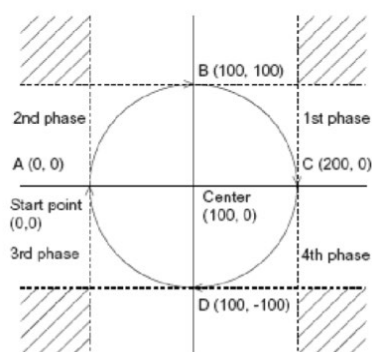


To specify a circular interpolation path, the following parameters must be clearly defined:

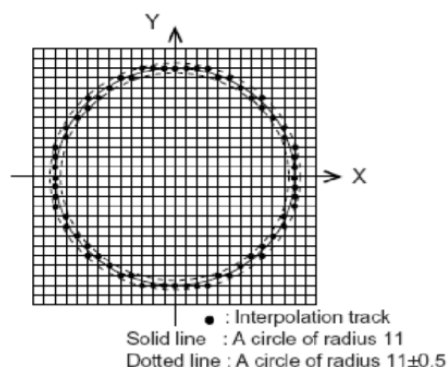
- Center point:** The coordinate of the center of arc (in absolute mode) or the off\_set distance to the center of arc (in relative mode)
- End point:** The coordinate of end point of the arc (in absolute mode) or the off\_set distance to center of the arc (in relative mode)
- Direction:** The moving direction, either CW or CCW.

It is not necessary to set the radius or the angle of arc since the information above provides enough constrains. The arc motion stopped when either of the 2 axes reaches the end point.

The final point can be set out of the path of arc; however, if the final point is in the location of the shadowed areas in the following graph, it will run circularly without stopping.



The command precision of circular interpolation is as shown below. The precision range is at radius  $\pm 1/2$  pulse.



There are two groups of functions that provide 2 axes circular interpolation.

1. The first group is to divide the 4 axes into XY (axis 0 & axis 1) and ZU (axis 2 & axis 3). By calling these functions, the target axes are already assigned.

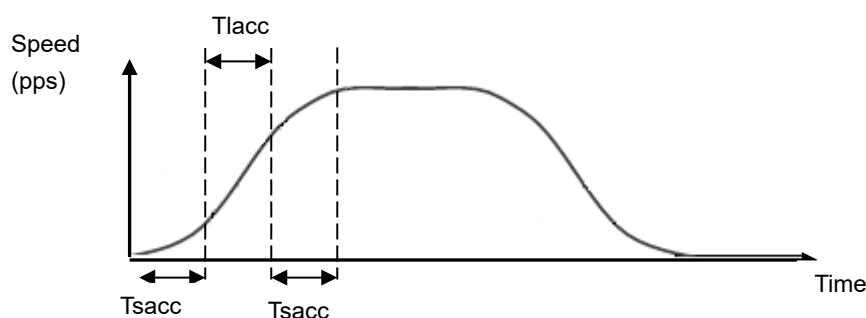
`_8443_start_r_arc_xy()`, `_8443_start_r_arc_zu()`,  
`_8443_start_a_arc_xy()`, `_8443_start_a_arc_zu()`,  
 : See section 7.8.

2. The second group allows you to freely assign any 2 target axes.

`_8443_start_r_arc2()`, `_8443_start_a_arc2()`,  
 : See section 7.8.

## 5.1.7 Circular Interpolation with Acceleration / Deceleration Time

In section 5.1.6, circular interpolations without acceleration and deceleration parameters are explained. You can perform neither Trapezoidal nor S-curve speed profile during the operations. However, sometimes you need this kind of speed profile to make a machine run smoothly even in a circular interpolation mode. PPCle-8443 has another group of circular interpolation functions to perform the speed profile. When you use, for example, axis 2 and axis 3 to perform a circular interpolation with Trapezoidal speed profile, you can use the function `_8443_start_tr_arc_zu()`. For the full list of these functions, see section 7.8.



Tlacc: Linear acceleration section

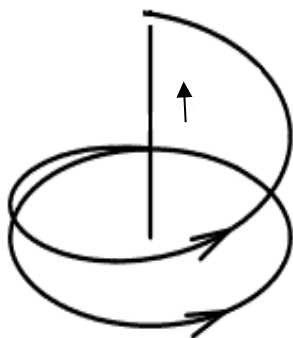
Tsacc: S-curve acceleration section

## 5.1.8 Helical Interpolation

In PPCle-8443, helical interpolation operation functions are equipped. The operation can be done by the “circular interpolation with U axis synchronization”, which is a feature of PCL6046 chip.

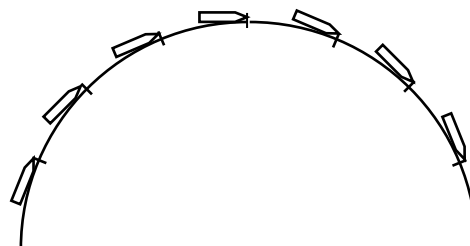
By using the functions, for example, a helical motion in which the Z axis moves upward in synchronization with arc motion, and a tangential interpolation operation for controlling the angle of the Z axis in synchronization with circular motions can be performed. See "7.9 Helical Interpolation Operation" for function format.

Example) when maintaining the cutter at a certain angle with respect to the circular arc tangent



Helical motion

Z axis moves upward in synchronization with arc motion.



Tangent interpolation operation

Circular interpolation and synchronization of Z axis angle.



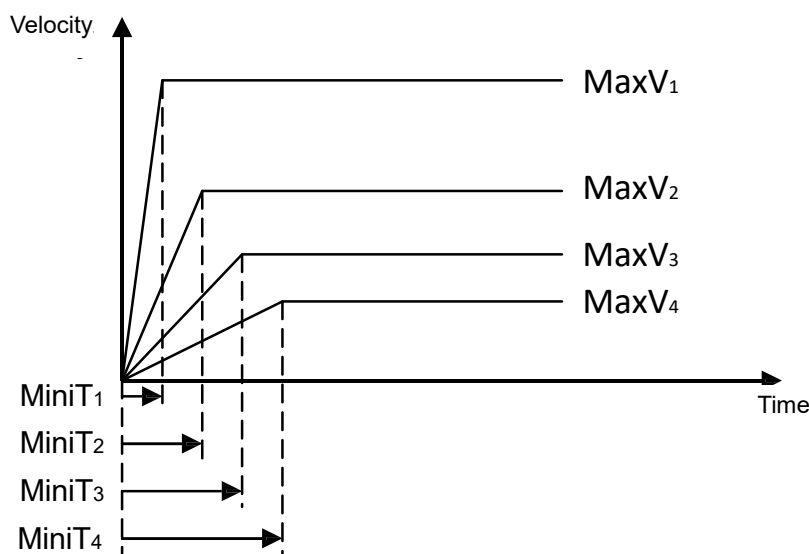
## 5.1.9 The Relationship between Velocity and Acceleration Time

The maximum velocity parameter of a motion has the minimum value of an acceleration time eventually. It means that there is a range to set in acceleration time to reach the required velocity value. If you want to set a small acceleration time, you must increase the maximum velocity value to match your requirement. We provide one function for doing that.

**`_8443_fix_speed_range()`**. This function can raise the maximum velocity value with a smaller acceleration time. However, it won't affect the actual motion velocity.

For example: the acceleration from 0 velocity to 5000 (pps) velocity in 1ms is not possible in the normal setting. But if you use the function with a higher velocity setting before the motion, the operation will be possible. The program will be like this:

```
_8443_fix_speed_range(AxisNo,OverVelocity);  
_8443_start_tr_move(AxisNo,5000,0,5000,0.001,0.001);
```



How to decide the optimized value of OverVelocity" in the **`_8443_fix_speed_range()`** function? We provide a function to calculate: **`_8443_verify_speed()`**. The input value of this function is the start velocity, the maximum velocity, and the over velocity in motion command. The output value will be the minimum and the maximum value of the acceleration time. For example, let's see the original acceleration range of this command:

```
_8443_start_tr_move(AxisNo,5000,0,5000,0.001,0.001);
```

You can try this function to obtain the limit values:

```
_8443_verify_speed(0,5000,&minAccT, &maxAccT,5000);
```

As the result, the value of minAccT will be 0.0267 sec and maxAccT will be 873.587 sec. This minimum acceleration time does not match your requirements, so you must use over speed value to do that.

If you use over speed as 20000,

```
_8443_verify_speed(0,5000,&minAccT, &maxAccT,20000);
```

The value of minAccT will be 0.00666 sec and maxAccT will be 218.387 sec. This minimum acceleration time still does not match the requirements. If we use over speed as 140000,

```
_8443_verify_speed(0,5000,&minAccT, &maxAccT,140000);
```

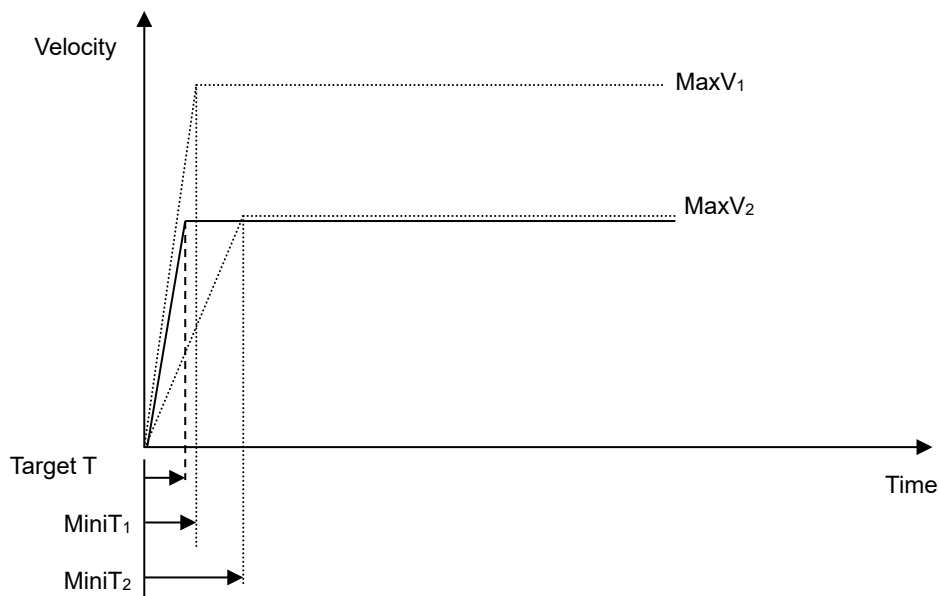
The value of minAccT will be 0.000948 sec and maxAccT will be 31.08 sec. This minimum acceleration time does match the requirements, and the motion command will be like:

```
_8443_fix_speed_range(AxisNo,140000);  
_8443_start_tr_move(AxisNo,5000,0,5000,0.001,0.001);
```

Note 1) The return value of `_8443_verify_speed()` function is the minimum velocity of your motion command, and it is not always equal to your start velocity setting. In the above example, the set value of the start velocity is 0 pps, but it is 3 pps in the actual operation.

Note 2) The velocity range setting fixed by `_8443_fix_speed_range()` function, can be disabled by `_8443_unfix_speed_range()` function.

Note 3) Do not to use the over speed unless you actually need it. Please be aware that increasing the velocity range setting will result in coarse resolution in velocity setting. .



Example:

Target profile: MaxV2

Target acceleration time: Target T

The minimum acceleration time to MaxV2 (MiniT2) is longer than the target value due to the (MaxV, MiniT) relationship

So we have to change the (MaxV and MiniT) relationship to a higher setting (MaxV1, MiniT1). Finally, the command for the target operation would be as follows:

```
_8443_fix_speed_range(AxisNo, MaxV1);
_8443_start_tr_move(AxisNo, Distance, 0, MaxV2, TargetT, Target T);
```

Related functions:

```
_8443_fix_speed_range(),
_8443_unfix_speed_range(), _8443_verify_speed()
```

: See section 7.5.

## 5.1.10 Continuous Operation

PPCLe-8443 allows you to perform continuous operations. Both single axis operation (section 5.1.3: Trapezoidal, section 5.1.4: S-curve) and multi-axis interpolation operation (5.1.5: linear interpolation, 5.1.6: circular interpolation) can be extended continuously.

For example, if you call the follow function to perform a single axis preset motion:

```
_8443_start_ta_move(0,50000.0,100.0,30000.0,0.1,0.0)
```

It will execute the axis "0" to move to position "50000.0". Before the axis arrives at the target position, you can call the second positioning operation:

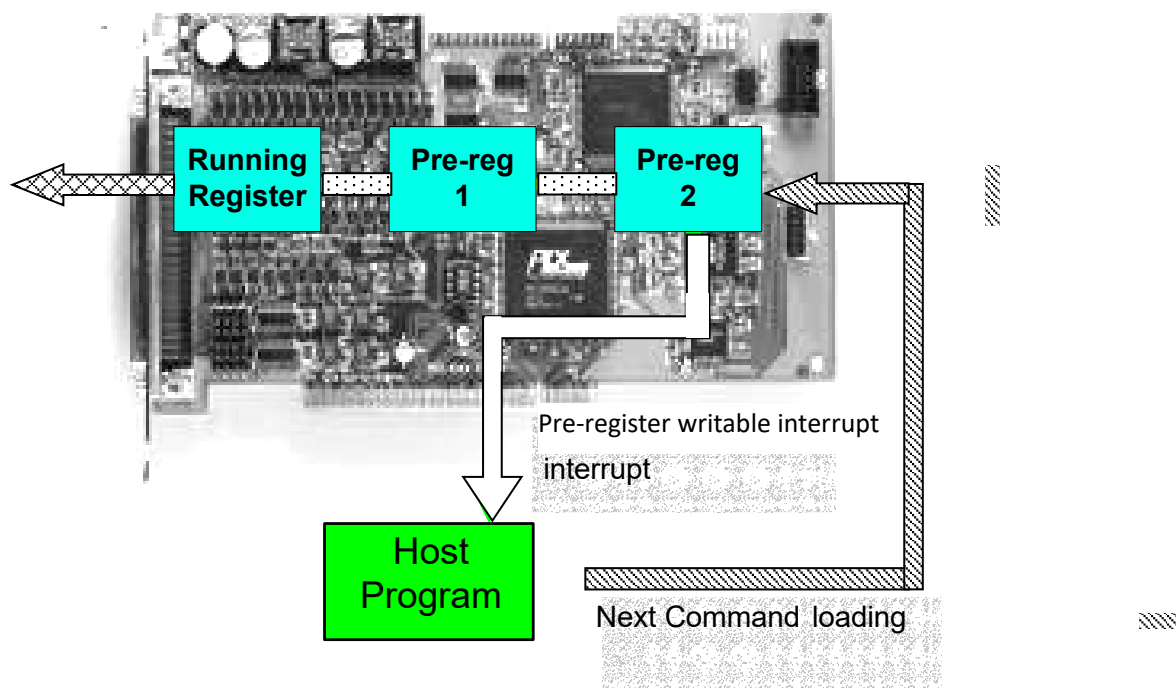
```
_8443_start_tr_move(0,20000.0,100.0,30000.0,0.0,0.2)
```

The second function call will not affect the first one. Actually, it will be executed and be written in the pre-register of PPCLe-8443. After the first operation is finished, PPCLe-8443 will continue the second operation per the pre-register value. No interval time exists between these two operations, and pulse trains will be continuously generated at the instant of position "50000.0".

The theory of continuous operations is described below:

### *Theory of continuous motion*

The following diagram shows the register data flow of PPCLe-8443.



Step 0: All Register and Pre-Register is empty.

Step 1: The first operation is executed and CPU writes corresponding values into the pre-register 2.

```
_8443_start_ta_move(0,50000.0,100.0,30000.0,0.1,0.0)
```

Step 2: Since the pre-register1 and the register are empty, the data in the pre-register 2 can be moved to the register automatically and executed instantly by PCL6046.

Step 3: The second function is called, and CPU writes the corresponding values into the

`_8443_start_tr_move(0,20000.0,100.0,30000.0,0.0,0.2)`

Step 4: Since the pre-register1 is empty, the data in the pre-register 2 is moved to the pre-register 1 automatically and is wait to be executed.

Step 5: Now you can execute the 3rd function, and it will be stored in the pre- register 2.

Step 6: When the first function is finished, the register becomes empty and the data in the pre-register 1 is moved to the register, and executed instantly by PCL6046.  
Then the data in pre-register2 is moved to the pre-register 1.

Step 7: PCL6046 will inform CPU by interrupt when the operations are completed. Then you can write the 4th operation into Pre-Register 2.

#### Procedures to perform a continuous operation:

The following shows the procedures for a continuous operation.

Step 1: Enable the interrupt operation by `_8443_int_contol()` and `_8443_int_enable()`.

Step 2: Set bit "2" of INT factor to "True" by `_8443_set_int_factor()`.

Step 3: Set the "conti\_logic" to "1" by: `_8443_set_continuous_move()`.

(Note: if the all operations are in relative mode, this function can be ignored.)

Step 4: Call the first three operation functions.

Step 5: Wait for EVENTof pre-register empty.

Step 6: Call the 4th operation function.

Step 7: Wait for EVENT of pre- register empty.

Step 8: Call the 5th operation function.

(Repeat Step 7 and 8...Continue...

Step n: Call the last operation function and wait for all operations to be completed.

(Note: Another way to detect completion of motion is by polling. You may constantly check the buffer status by `_8443_check_continuous_buffer()`).

#### Restrictions on continuous operation:

The following are restrictions and suggestions for continuous operations:

- 1) While the pre-register is not empty, you cannot execute further operations. Otherwise, the new operation will be overwritten over the previous operation in the pre-register 2.
- 2) To smoothly continue two operations, the end velocity of previous operation and the starting velocity of the following operation need to be set the same. The easiest way is to set the deceleration/acceleration time to be '0'.

Example:

First operation: `_8443_start_tr_move_XY(0,1000,0,0,5000,0.2,0.0)`

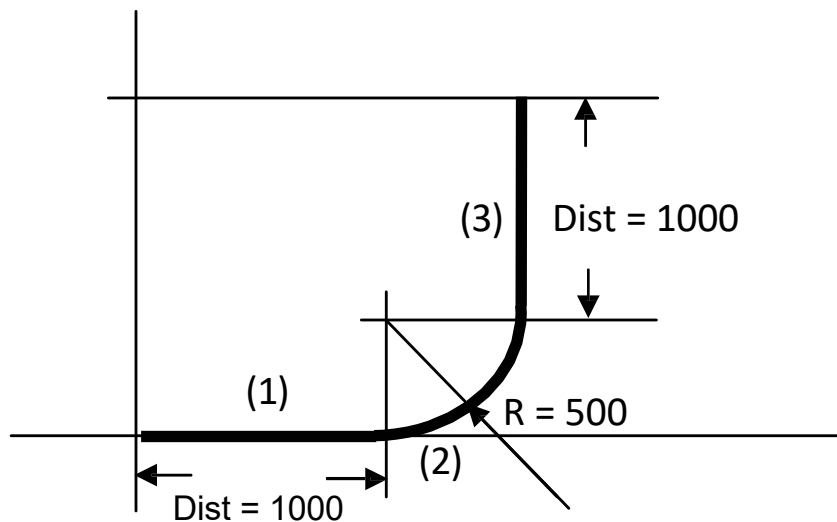
(Start a relative 2-axis linear interpolation, X distance=1000, Y distance= 0,  
Start vel = 0, Max vel = 5000, Tacc = 0.2, Tdec = 0)

Second operation: `_8443_start_r_arc_xy(0,0,500,500,500,1,5000);`

(Start a relative 2 axis circular interpolation, Center X distance=0,  
Center Y distance= 500, End X distance = 500, End Y distance = 500. Max vel = 5000.  
Quarter CCW circle, Velocity = 5000)

Third operation: `_8443_start_tr_move_XY(0,0,1000,0,5000,0.0, 0.2)`

(Start a relative 2-axis linear interpolation, X distance=0, Y distance= 1000 , Start vel = 0, Max vel = 5000,  
Tacc = 0.0,Tdec = 0.2)



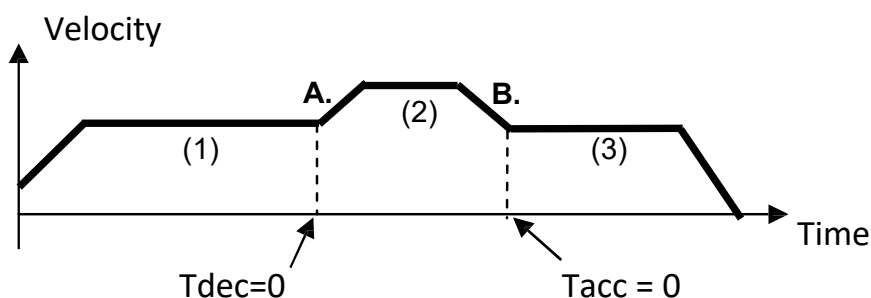
#### Explanation of example:

While these three operations are being executed sequentially without waiting, the 1st operation is written in the Register and is executed instantly; the 2nd operation is written in the Pre-Register 1 and is waiting for completion of the 1st operation; the 3rd operation is written in the Pre-Register 2 and is waiting for completion of the 2nd operation. Since the 1st operation has '0' deceleration time and the 2nd operation is on an arc at the constant velocity, which is the same as the max velocity of the 1st operation, the PPCle-8443 will output constant frequency at intersection between them.

1. Continuous operations between different axes cannot be performed since each axis has its own register and pre-register system.
2. Continuous motion between different numbers of axes is not allowed. For example: `_8443_start_tr_move()` cannot be followed by `_8443_start_ta_move_XY()`, or Vice versa, because these two functions belong to single axis and 2-axis mode individually.
3. It is possible to perform 3 axes or 4 axes continuous linear interpolation, but the speed continuity is impossible to achieve.
4. If any absolute mode is used during continuous operation, make sure that the `_8443_reset_target_pos()` is executed at least once after the home return operation (see 5.1.11: Home Return Operation (Origin Return))

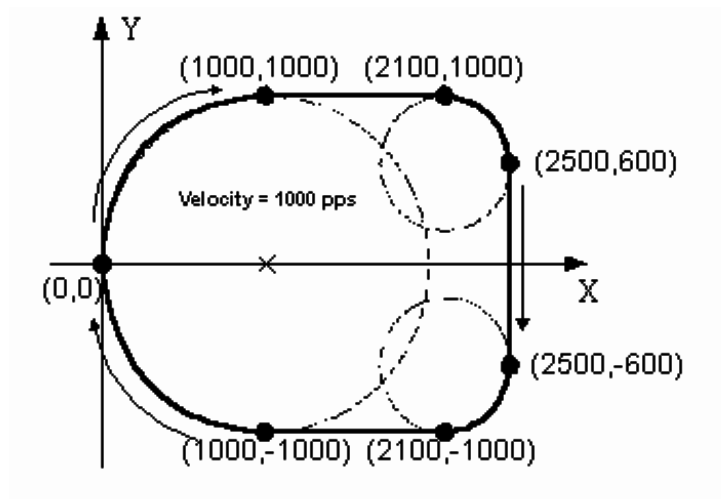
#### Example of continuous operation

1. Single axes continuous operation: Changing velocity at preset point.



This example demonstrates how to use continuous operation function to achieve the velocity changed at the pre-set point. The 1st motion (ta) moves an axis to point A, with  $T_{dec}=0$ , and then the 2nd continues instantly. The starting velocity of (2) is the same as the max velocity of (1), so that the velocity continuously exists at A. At point B. the  $T_{acc}$  of (3) is set to be 0, so the velocity continuity is also realized.

## 2. 2-axis continuous interpolation :



This example demonstrates how to use continuous operation function to achieve 2-axis continuous interpolation. In this application, the velocity continuity is the key concern. See the example in the previous page.

Related functions:

**`_8443_set_continuous_move()`**, **`_8443_check_continuous_buffer()`**

: See section 7.18.

### 5.1.11 Home Return Operation (Origin Return)

In this operation, you can control the home return sequence by writing the command `_8443_home_move()`. There are 13 Home modes provided by PPCle-8443. The “home\_mode” of function `_8443_set_home_config()` is used to select the mode.

After the home return operation is completed, all of the position related information will be reset to “0”. In PPCle-8443, there are 4 counters and 1 software-maintained position recorder.

Counter / Recorder	Description
Command position counter	To count the number of pulses output
Feedback position counter	To count the number of pulse input
Position error counter	To count the error between command and feedback pulse number.
General-Purpose counter:	General purpose (Select from below) Command output pulse, Feedback input pulse, Manual pulser, Reference CLK/2.
Target position recorder	To record the target position

(See section 5.4 for more detail explanation for the position counters)

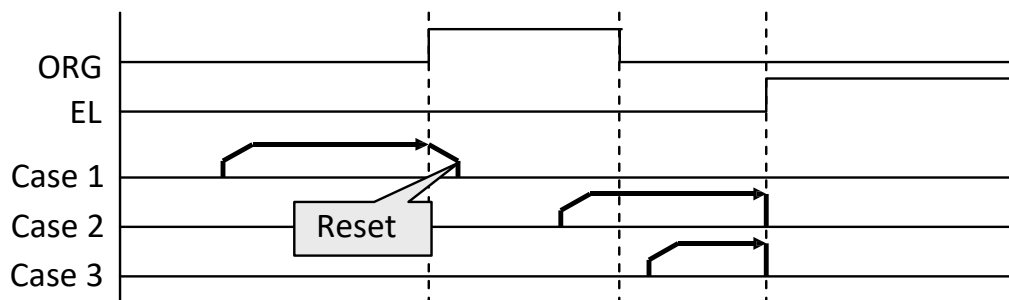
Once a Home return operation is completed, the first four counters will be cleared to “0” automatically. However the target position recorder will not cleared automatically since it is maintained by software. It is necessary to manually set the target position to “0” by calling the function: `_8443_reset_target_pos()`, so that, all of the positions information will be “0”.

The following figures show the various Home modes and the reset points, when the counter will be cleared to “0”.

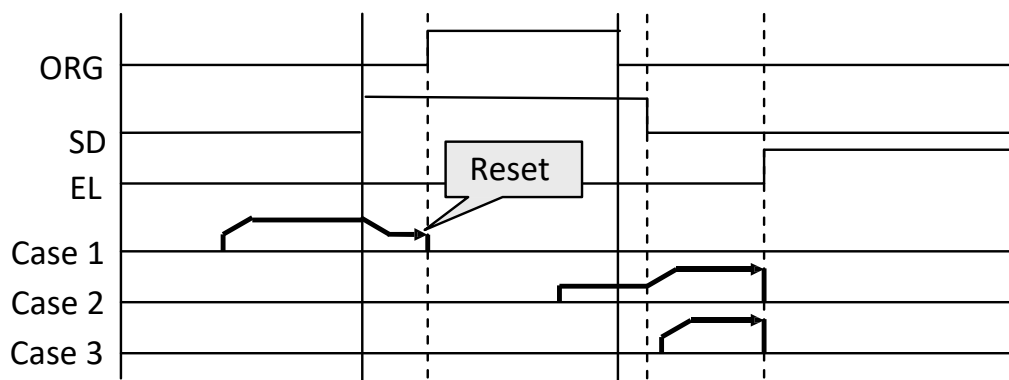
(See section 5.4 for more detail explanation about position counters)

**home\_mode = 0: ORG → Slow down → Stop**

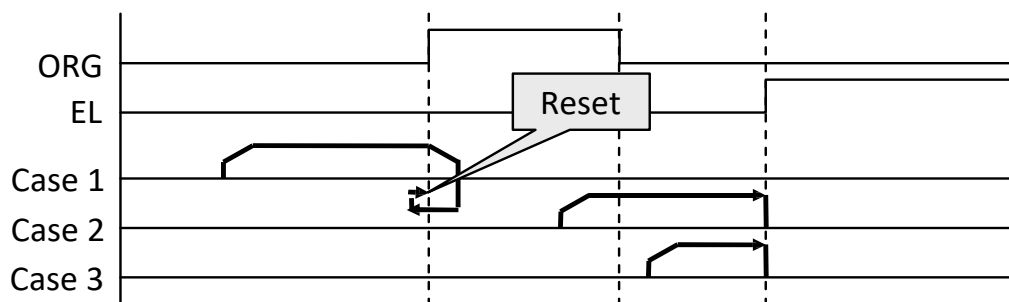
- SD(Ramp-down signal) is disabled



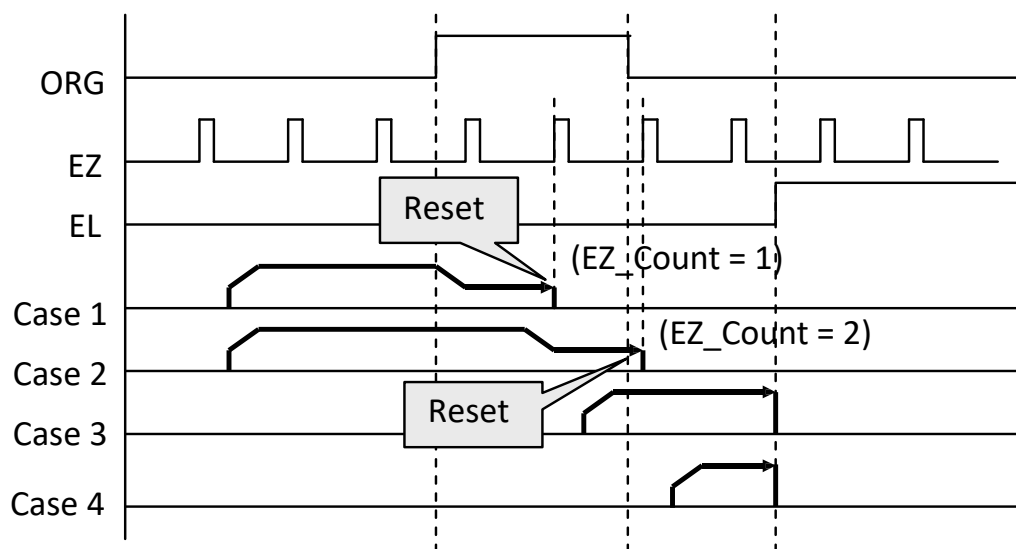
- SD(Ramp-down signal) is enabled



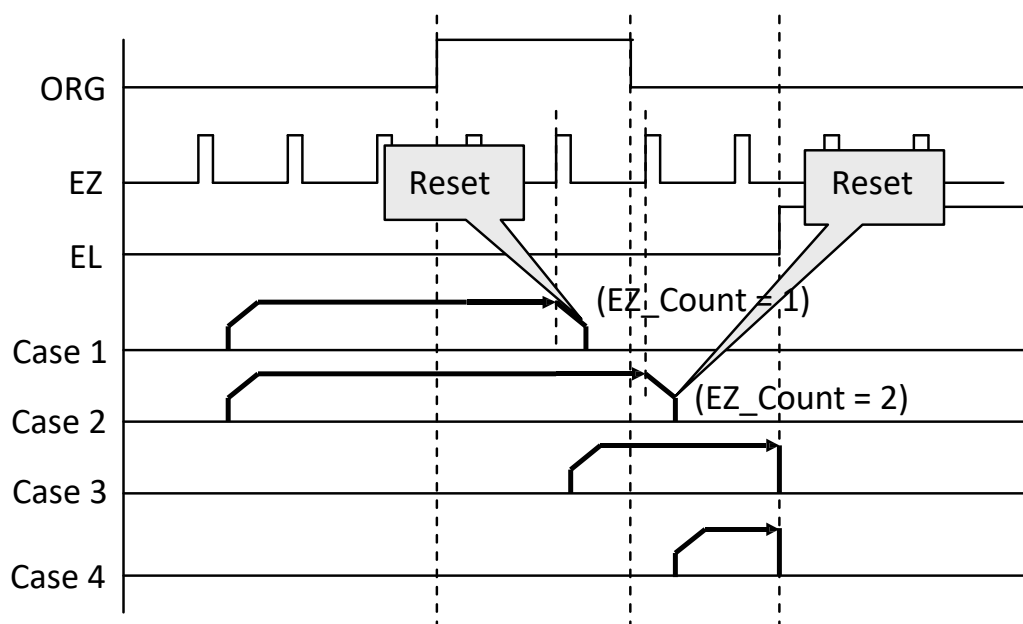
home\_mode = 1: ORG → Slow down → Stop at end of ORG



home\_mode = 2: ORG → Slow down → Stop on EZ signal

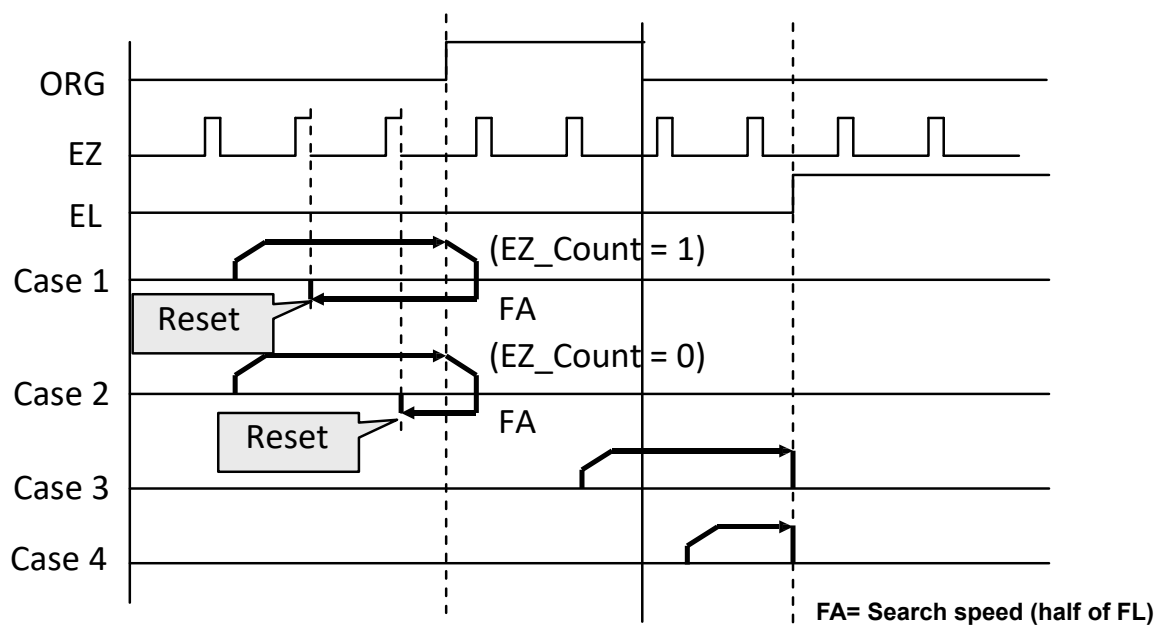


home\_mode = 3: ORG → EZ → Slow down → Stop

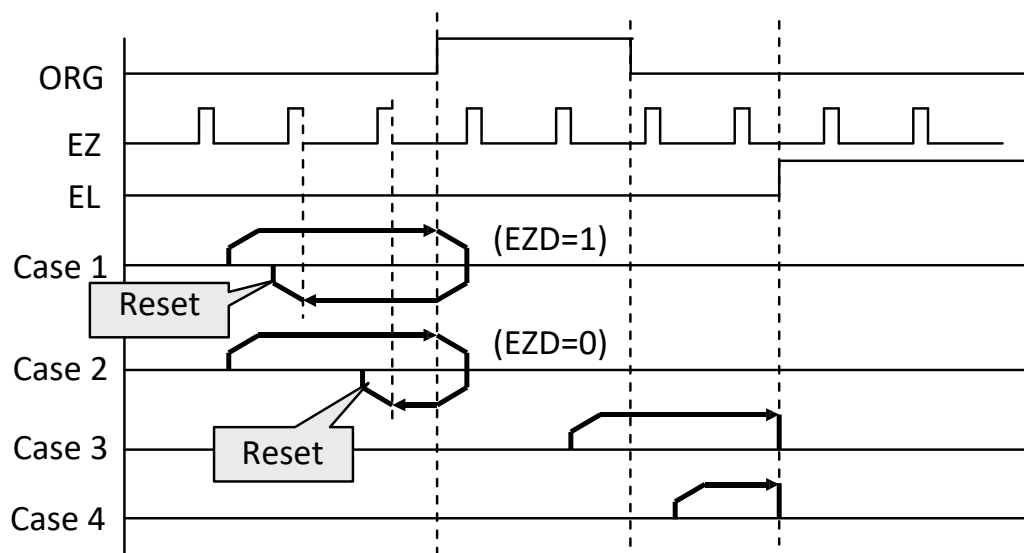




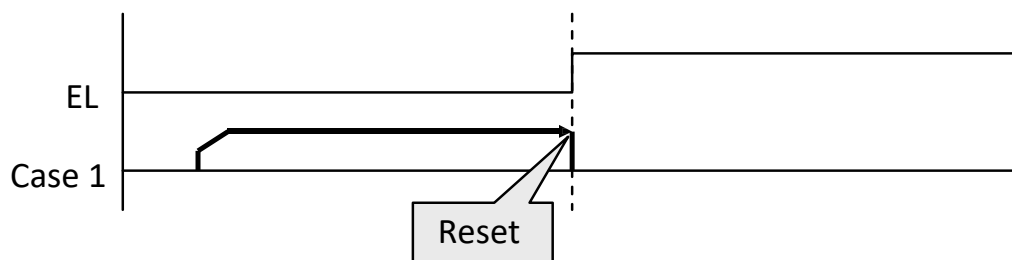
home\_mode = 4: ORG → Slow down → Go back at FA speed → EZ → Stop



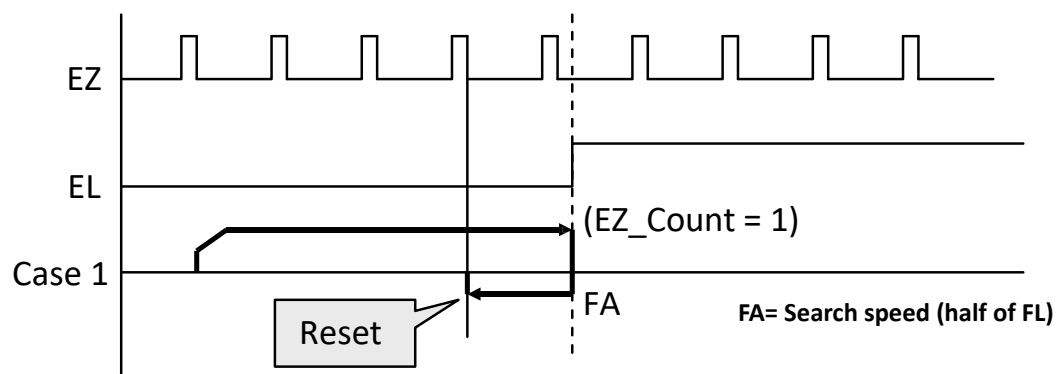
home\_mode = 5: ORG → Slow down → Go back → Accelerate to high speed → EZ → Slow down → Stop



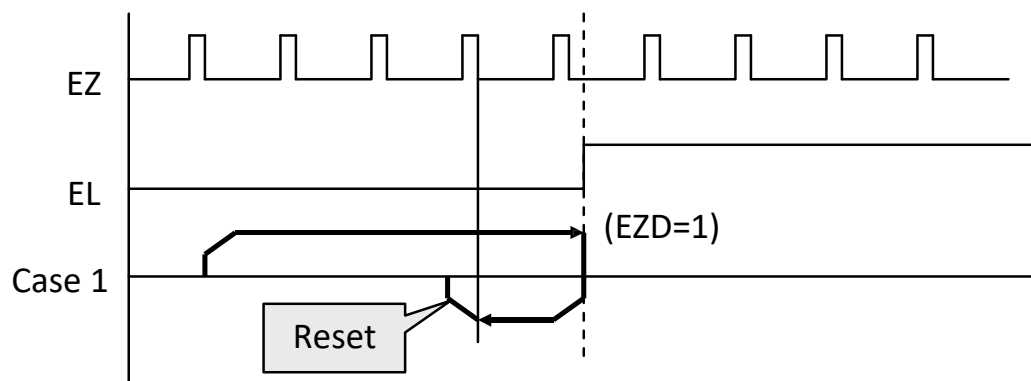
home\_mode = 6: EL only



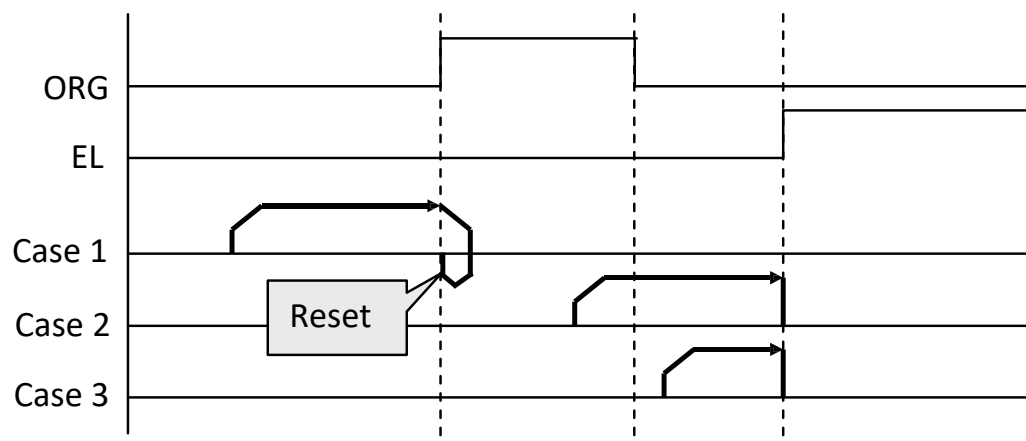
home\_mode = 7: EL → Go back → Stop on EZ signal



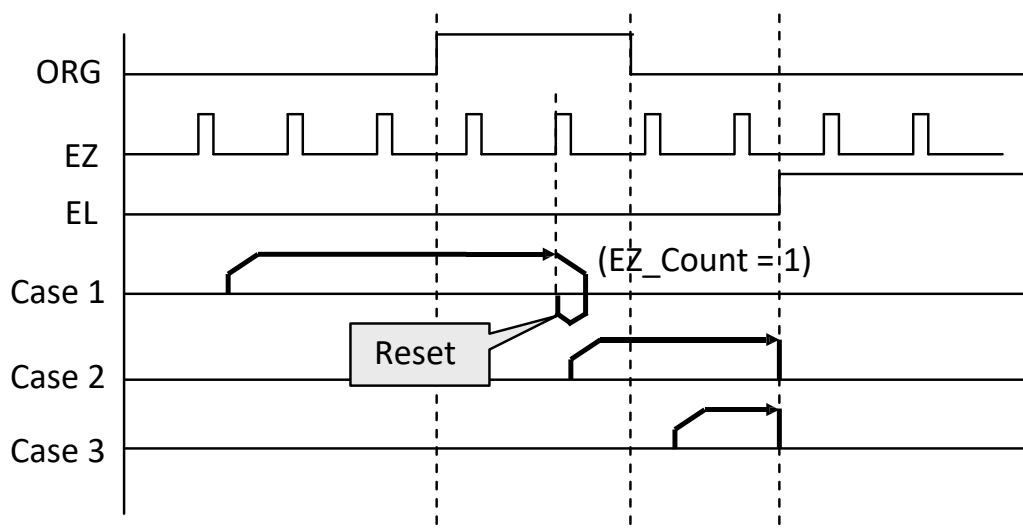
home\_mode = 8: EL → Go back → Accelerate to high speed → EZ → Slow down → Stop



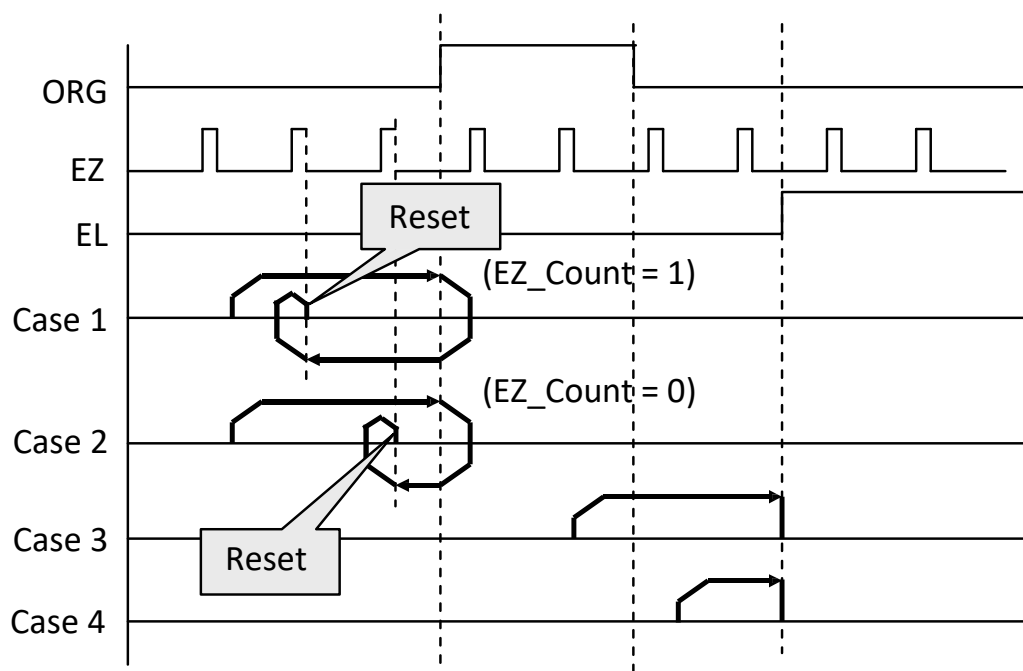
home\_mode = 9: ORG → Slow down → Go back → Stop at beginning edge of ORG



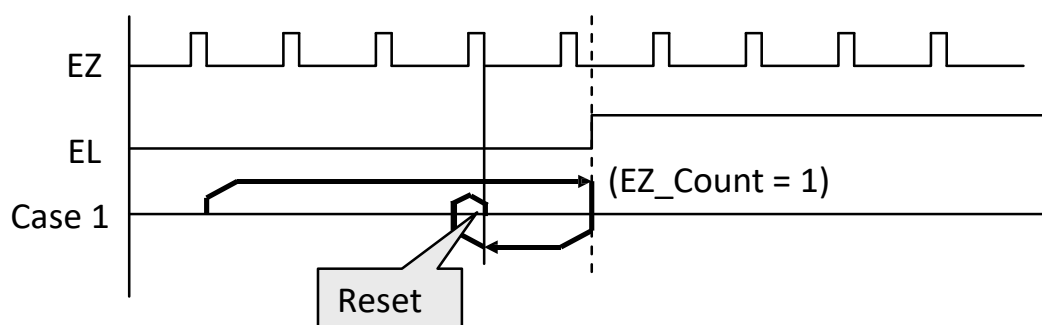
home\_mode = 10: ORG → EZ → Slow down → Go back → Stop at beginning edge of EZ



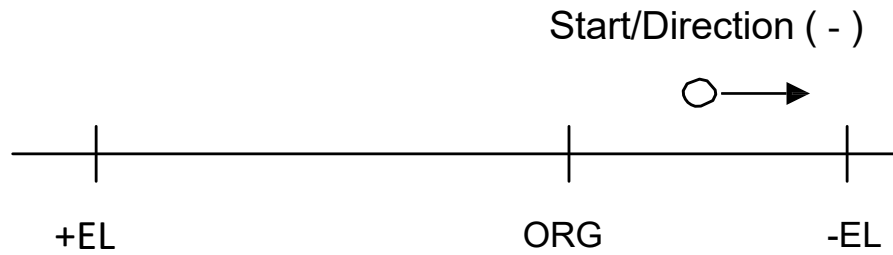
home\_mode = 11: ORG → Slow down → Go back (backward) → Accelerate to MaxVel → EZ → Slow down → Go back (forward) → Stop at beginning edge of EZ



home\_mode = 12: EL → Stop → Go back (backward) → Accelerate to MaxVel → EZ → Slow down → Go back again (forward) → Stop at beginning edge of EZ



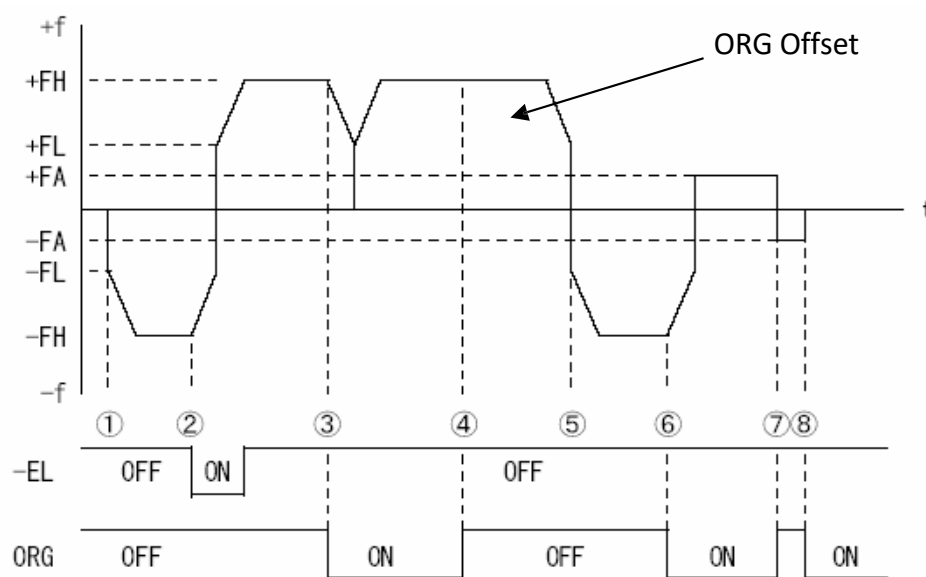
# Origin search function example (Home mode = 1)



FL = Starting velocity

FH = Max velocity (The sign shows the direction)

FA = Search speed (Half of the FL)



## Operation steps

1. Origin searching start (-)
2. Slow down at -EL and reverse moving (+)
3. Slow down at ORG
4. Escape from ORG operation (+) according to ORGOffset
5. Start searching again (-)
6. Slow down at ORG, escape from ORG (+) at searching speed.
7. After escape from ORG. search ORG at searching speed again (-).

Related functions:

`_8443_set_home_config()`, `_8443_home_move()`, `_8443_home_search()`, `_8443_auto_home_search()`

: See section 7.10.

## 5.1.12 Manual pulser operation

As for a manual operation device, you may use a manual pulser such as a rotary encoder. The PPCle-8443 can input signals from a pulser and output corresponding pulses from the OUT and DIR terminals. It allows you to simplify the external circuit and control the present position of axis.

This mode is effective when

`_8443_pulser_vmove()`, `_8443_pulser_pmove()`

or `_8443_pulser_home_move()` command has been called.

To stop, by `_8443_sd_stop()` or `_8443_emg_stop()` command or by the completion of the operation.

The PPCle-8443 receives positive direction and negative direction pulses (CW / CCW) or 90 degrees phase difference signals (A / B phase) from a pulser at PA and PB terminals. To set the input signal modes of a pulser, use

`_8443_set_pulser_iptmode()` function. The 90 degree phase difference signals can be input through multiplication by 1, 2 or 4. If the A / B phase input mode is selected, the PA and PB signals should be with 90 degree phase shifted, and the position counting will increase when the PA signal is leading the PB signal by 90 degree phase.

Related functions:

`_8443_pulser_vmove()`, `_8443_pulser_pmove()`, `_8443_pulser_home_move()`,  
`_8443_set_pulser_iptmode()`: See section 7.11.

## 5.1.13 Timer Mode

In this mode, you can delay the execution of program by a specified delay time (ms).

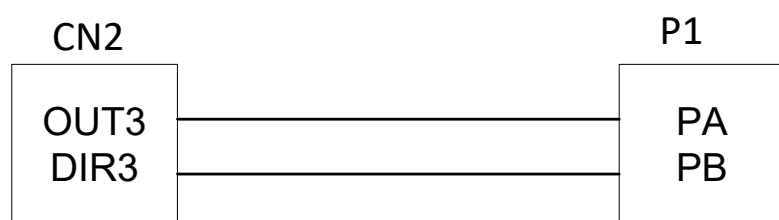
For example, `_8443_delay_time(0, 100)`; after executing this command, there will be 100 ms delay in executing the next command.

Relative Functions:

`_8443_delay_time()`: See section 7.1.

## 5.1.14 Pulser Interpolation

You can use a pulser for interpolation of motion (any of two axes on linear interpolation or any of two axes on circular interpolation). This mode can only work under incremental mode. See the following diagram. When one of the axes functions is used as a dummy axis, the axis cannot be used for interpolation. For example, when No. 3 axis is used as dummy axis, then any two axes from No.0 to No.2 axis can be used as interpolation axes.



### Any of two axes for linear interpolation

Referring to the above diagram, by executing `_8443_pulser_r_line2()` command, it is possible to execute linear interpolation motion. The axes used for interpolation can be set by AxisArray parameter. `_8443_pulser_r_line2()`: see section 7.11.

### Any of two axes for circular interpolation

Referring to the above diagram, by executing `_8443_pulser_r_arc2()` command, it is possible to execute circular interpolation motion. The axes used for interpolation can be set by AxisArray parameter. `_8443_pulser_r_line2()`: see section 7.11.

## 5.2. The Motor Driver Interface

PPCLe-8443 provides the INP, ALM, ERC, SVON, and RDY signals for the servomotor driver's control interface. The INP and ALM are used for feedback of a servo driver's status. The ERC is used to reset a servo driver's deviation counter under special conditions. The SVON is a general purpose output signal, and RDY is a general-purpose input signal. The meaning of "general purpose" is that the processing of the signals is not build-in procedure of hardware. The hardware can process INP, ALM, and ERC signals according to the pre-defined rule. For example, when receiving ALM signal, PPCLe-8443 will stop or decelerate to stop output pulses automatically. However, SVON and RDY are not the case; they actually act like common I/O.

### 5.2.1 INP

The processing of INP signal is a hardware build-in procedure, and it is designed to cooperate with the in-position signal of servomotor driver.

Usually, servomotor driver with pulse train input has a deviation (position error) counter to detect the deviation between the input pulse command and feedback counter. The driver controls the motion of servomotor to minimize the deviation until it becomes 0. Theoretically, the servomotor operates with some time delay from command pulses. Accordingly, when the pulse generator stops outputting pulses, the servomotor does not stop but keep running until the deviation counter become zero. Then, the servo driver sends out the in-position signal (INP) to the pulse generator to indicate the motor stops running.

Usually, PPCLe-8443 stops outputting pulses upon completion of outputting designated pulses. But by setting parameter **inp\_enable** in **\_8443\_set\_inp()** function, you can delay the completion of motion to the time when the INP signal is turned on, ie, the motor arrives the target position. Status of **\_8443\_motion\_done()** and INT signal are also delayed. That is, when performing under position control mode, the completion of **\_8443\_start\_ta\_move()**, **\_8443\_start\_sr\_move()**, etc., is delayed until INP signal is turned ON. The in-position function can be enabled or disable, and the input logic polarity is also programmable by parameter "inp\_logic" of **\_8443\_set\_inp()**. The INP signal status can be monitored by software function: **\_8443\_get\_io\_status()**.

Relative Functions:

**\_8443\_set\_inp()** : See section 7.13.

**\_8443\_get\_io\_status()** : See section 7.14.

**\_8443\_motion\_done()** : See section 7.12.

## 5.2.2 ALM

The processing of ALM signal is a hardware build-in procedure, and it is designed to cooperate with the alarm signal of a servomotor driver. The ALM signal is an output from a servomotor driver. Usually, it is designed to inform something wrong with the driver or the motor.

The ALM terminal receives the alarm signal output from a servo driver. The signal immediately stops PPCle-8443 generating pulses or stops after deceleration. If the ALM signal is in the ON status at the start, PPCle-8443 outputs the INT signal without generating any command pulses. The ALM signal may be a pulse signal, of which the shortest width is a time length of 5  $\mu$ s.

You can change the input logic of ALM by setting the parameter "*alm\_logic*" of `_8443_set_alm` function and the stop mode by "*alm\_mode*". Whether or not PPCle-8443 is generating pulses, the ALM signal lets it output the INT signal. The ALM status can be monitored by software function: `_8443_get_io_status()`. The ALM signal can generate IRQ if the interrupt operation is enabled (See section 5.10).

Relative Functions:

`_8443_set_alm()` : See section 7.13.

`_8443_get_io_status()` : See section 7.14.

### 5.2.3 ERC

The ERC signal is an output from PPCle-8443. The processing of ERC signal is a hardware build-in procedure, and it is designed to cooperate with the deviation counter clear signal of a servomotor driver.

The deviation counter clear signal is inserted in the following three situations:

- 1) Home return is completed.
- 2) Error type signals such as End-limit switch, Alarm signal, or Emergency signal is turned on during operations.
- 3) Emergency stop command is issued by software.

Since a servomotor operates with some delay from pulses generated from PPCle-8443, it keeps moving until the deviation counter of the driver down to zero even if PPCle-8443 stops outputting pulses because of the EL signal or the completion of home return. The ERC signal allows you to immediately stop the servomotor by resetting the deviation counter to zero. The ERC signal is output as a one-shot signal. The pulse width is a time length defined by function call `_8443_set_erc()`. The ERC signal will be automatically output when  $\pm$ EL signal or ALM signal is turned on to immediately stop the servomotor.

**Relative Functions:**

`_8443_set_erc()` : See section 7.13.

### 5.2.4 SVON and RDY

In PPCle-8443, each axis is equipped with SVON and RDY, which are general-purpose output and input channels, respectively. Usually, SVON is useful to cooperate with servomotor drivers as Servo ON command, and RDY to receive the Servo Ready signal from servomotor drivers. That is the reason why they are named as SVON and RDY. There is no build-in procedure for SVON and RDY.

The SVON signals are controlled by software function: `_8443_set_servo()`.

RDY terminals are dedicated for digital input use.

The status of the signal can be monitored by software function `_8443_get_io_status()`.

**Related functions:**

`_8443_set_servo()`: See section 7.13.

`_8443_get_io_status()`: See section 7.14.



## 5.3. Mechanical Input Interface and I/O Status

In this section, the following I/O signal operations are described:

- SD / PCS : Ramping down and Position change sensor
- $\pm$  EL : End-limit sensor
- ORG : Origin position
- EMG : Emergency stop

In any operation mode, if an  $\pm$  EL signal is active during a moving condition, it will cause PPCle-8443 to stop output pulses automatically. If an SD signal is active during a moving condition, it will cause PPCle-8443 to decelerate. If operating in the multi-axes mode, it automatically applies to all related axes.

### 5.3.1 SD / PCS

SD / PCS terminal for each axis is an input channel and is selectable to connect into SD (ramping down) or Position Change Signal (PCS). It can be configured by function call `_8443_set_sd_pin()`. (The default setting is SD.)

When SD / PCS terminal is directed to SD, PCS signal will be kept in the low level. While PCS is selected, SD signal will be kept in the low level. You need to pay attention to the logic and enable/disable attributes for unused signals.

The ramping-down signals are used to make the output pulse (OUT and DIR) decelerate and then keep it on the StrVel while it is active. The StrVel is usually smaller than MaxVel, so this signal is very useful to protect the machine when moving at high speed toward the mechanical limit. SD signal is effective for both positive and negative directions.

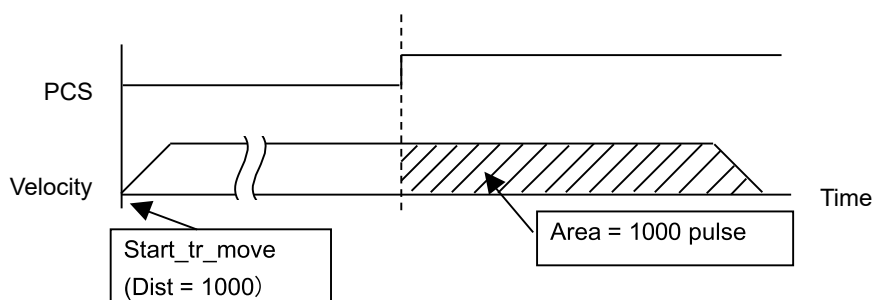
The ramping-down function can be enabled or disabled by the software function: `_8443_set_sd()`.

The input logic polarity, level operation mode, or latched input mode can also be set by this function. The signals status can be monitored by `_8443_get_io_status()`.

The PCS signal is used to define the starting point of a preset (tr / sr) motion.

The logic of PCS is configurable by `_8443_set_pcs_logic()`

See the following chart:



Related functions:

`_8443_set_sd_pin()`, `_8443_set_pcs_logic()`: See section 7.6.

`_8443_set_sd()`: See section 7.13.

`_8443_get_io_status()`: See section 7.14.

### 5.3.2 EL

The end-limit signals are used to stop the control output signals (OUT and DIR) when the end-limit is active. There are two possible stop modes; one is “stop immediately”, and the other is “decelerate to StrVel and stop”. To select the mode:

`_8443_set_el()`.

PEL signal indicates end-limit in positive (plus) direction while MEL signal indicates end-limit in negative (minus) direction. When the pulse signals (OUT and DIR) are output toward positive direction, the pulse train will be immediately stopped when the PEL signal is inserted. MEL signal is meaningless in this case, and vice versa. When the PEL is inserted, only the negative (minus) direction output pulse can be generated to move the motor in negative (minus) direction. EL signal can generate IRQ if the interrupt operation is enabled. (See section 5.10.)

You can use either ‘a’ contact switch or ‘b’ contact switch by setting the dip switch SW2. PPCle-8443 is delivered from the factory with all bits of SW2 set to ON.

The signal status can be monitored by software function: `_8443_get_io_status()`.

Related functions:

`_8443_set_el()`: See section 7.13.

`_8443_get_io_status()`: See section 7.14.

### 5.3.3 ORG

ORG signal is used when a motion controller is operated in the Home mode. There are 13 Home modes (see section 5.1.11), and you can select one of them by setting “**home\_mode**” argument in software function:

`_8443_set_home_config()`. The logic polarity of ORG signal is selectable by this software function.

After setting the configuration of Home mode by `_8443_set_home_config()`, the home return operation can be performed by `_8443_home_move()` function.

Relative Functions:

`_8443_set_home_config()`, `_8443_home_move()`: See section 7.10.

### 5.3.4 EMG

PPCLe-8443 provides a global digital input for emergency situation. Once the input is turned on, PPCLe-8443 will stop all axes’ operations immediately to prevent damages in the machine. Usually, you should connect it with the emergency stop button in the machine. You cannot use deceleration stop for emergency stop.

## 5.4. Counters

PPCLe-8443 provides 4 counters for each axis.

Counter	Description
Command position counter	To count the number of pulse output
Feedback position counter	To count the feedback input position counter
Position error counter	To count the error (deviation) between command pulse and feedback pulse number
General purpose counter	General purpose (Select from below) Pulse output, feedback pulse, manual pulser, or CLK / 2.

Also, the target position recorder (a software-maintained position recorder) will be discussed in this section. .

### 5.4.1 Command Position Counter

The command position counter is a 32-bits binary up / down counter, and the input source is an output pulse from PPCLe-8443, thus, it provide as exact information of the current command position.

Note: The command position is different from the target position. The command position increases or decreases per the pulse output while the target position changes only when a new motion command is executed. The target position is recorded by software, and need to be manually reset after the home return operation is completed.

The command position counter will be clear to “0” automatically after the home return operation is completed. Besides, the function call, `_8443_set_command()`, can be executed in any time to set a new command position value. To read the current command position, use the function: `_8443_get_command()`.

Relative Functions:

`_8443_set_command()`, `_8443_get_command()`: See section 7.16.

### 5.4.2 Feedback Position Counter

PPCLe-8443 has a 32-bits binary up / down counter for managing the present position feedback for each axis. The counter counts signals input from EA and EB terminals.

It can accept 2 kinds of pulse input: (1): plus and minus pulses input (CW/CCW mode); (2): 90° phase difference signals (A / B phase mode). 90° phase difference signals can be selected to be multiplied by a factor of 1, 2 or 4. The 4 x A / B phase mode is the most commonly used for incremental encoder input. For example, if a rotary encoder has 2000 pulses per phase (A or B phase), then the value read from the counter will be 8000 pulses per turn or – 8000 pulses per turn depending on its turning direction. These input modes can be selected by `_8443_set_pls_ipmode()` function.

In the case of applications without implementing an encoder (without any feedback), it is possible to set the feedback counter source to be the output pulse, just as the command counter. Thus, the feedback counter and the command counter will show the same value. To enable the counters counting pulses input from pulse output, set “Src” parameter of software function `_8443_set_feedback_src()` to “1”.

**Plus and minus pulses input mode (CW/CCW Pulse input mode)**

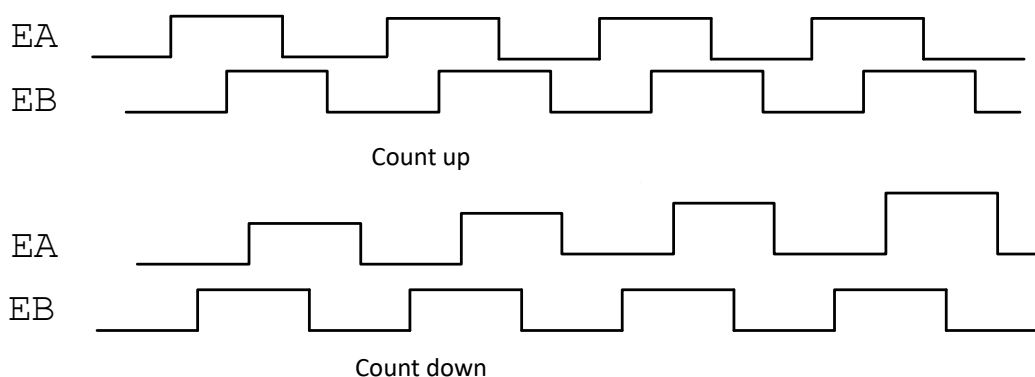
The pattern of pulses in this mode is the same as **Dual pulse output mode** in Pulse command output section, and the input terminals are EA and EB.

In this mode, a pulse from EA causes the counter to count up (+), whereas EB causes the counter to count down (-).

**90 degree phase difference signals input mode (A / B phase Mode)**

In this mode, EA signal is 90 degree phase leading or lagging in comparison with EB signal. "Lead" or "lag" of phase difference between two signals is determined by the turning direction of motors. The up / down counter counts up when the phase of EA signal leads the phase of EB signal.

The following diagram shows the waveform:



The index inputs (EZ) signals of encoders are used as the "ZERO" index. This signal is common in most of the rotational motors. EZ can be used to define the absolute position of the mechanism.

The input logic polarity of the EZ signals is programmable by software function `_8443_set_home_config()`. The EZ signals status of the four axis can be monitored by `_8443_get_io_status()`.

The feedback position counter will be cleared to "0" automatically after home return operation is completed.

Besides, the function call, `_8443_set_position()`, can be executed in any time to set a new command position value.

To read the current command position, use function call: `_8443_get_position()`.

Related functions:

`_8443_set_pls_ipmode()`, `_8443_set_feedback_src()`: See section 7.4.

`_8443_set_position()`, `_8443_get_position()`: See section 7.16.

`_8443_set_home_config()`: See section 7.10.

### 5.4.3 Position Error Counter

The position error counter is used to calculate the error between command position and feedback position. The working theory is that it adds one count when PPCle-8443 output one command pulse and subtracts one count when PPCle-8443 receives one feedback pulse (from EA, EB). It is very useful to detect the step-losing situation (out of step) of stepping motors when an encoder is applied.

Since the position error counter automatically calculate the difference between pulse output and pulse feedback, it is inevitable to get error if the motion ratio is not equal to "1".

To get the position error, use the function call: **`_8443_get_error_counter()`**.

To reset the position error counter, use the function call: **`_8443_reset_error_counter()`**.

(The position error counter will be automatically cleared to "0" after an home return operation is completed.)

Related functions:

**`_8443_get_error_counter()`**, **`_8443_reset_error_counter()`**: See section 7.16.

## 5.4.4 General Purpose Counter

See the table below for the counter input functions of the general purpose counter.

The default source of a general purpose counter is a manual pulser. (See section 5.1.12 for detail explanation of manual pulser.)

To set other source, use the function call: `_8443_set_general_counter()`.

To get the counter value, use the function call: `_8443_get_general_counter()`.

Relative Function:

`_8443_set_general_counter()`, `_8443_get_general_counter()`: See section 7.16.

Counter	Description	Counter Source	Function	Function description
Command position	To count the number of pulses output	Pulses output	<code>_8443_set_command</code>	Set a new value for command position
			<code>_8443_get_command</code>	Read current command position
Feedback position	To count the number of pulses input	EA/EB or Pulse output	<code>_8443_set_pls_iptmode</code>	Select the input modes of EA/EB
			<code>_8443_set_feedback_src</code>	Set the counters input source
			<code>_8443_set_position</code>	Set a new value for feedback position
			<code>_8443_get_position</code>	Read current feedback position
Position error	To count the error between command and feedback pulse	EA/EB and Pulse output	<code>_8443_get_error_counter</code>	To get the position error
			<code>_8443_reset_error_counter</code>	To reset the position error counter
General purpose	General purpose counter	Pulse output EA/EB Manual pulser CLK/2	<code>_8443_set_general_counter</code>	Set a new counter value
			<code>_8443_get_general_counter</code>	Read current counter value

## 5.4.5 Target Position Recorder

The target position recorder is very useful for providing target position information, which is managed by the software. For example, if PPCle-8443 is operating in a continuous operation with the absolute mode, the target position will let the next absolute operation knows the target position of the previous operation.

It is very important to know that the target position recorder is handled by the software. Please note the following:

- 1) Every time a new motion command is executed, the displacement is added automatically into the target position recorder.
- 2) To make sure the correctness of target position recorder, your need to manually maintain it in the following two situations by the function call `_8443_reset_target_pos()`:
  1. After a home return operation is completed
  2. After a new feedback position is set

Related functions:

`_8443_reset_target_pos()`: See section 7.16.

## 5.5. Multiple PPCle-8443 operation

The software function library supports up to 12 PPCle-8443 boards, which means up to 48 axes of motors can be controlled. Since PPCle-8443 has a characteristic of Plug-and-Play, you do not have to worry about setting the based address and IRQ level of boards. They are automatically assigned by the BIOS of system when booting up. You can utilize “PPCle8443 Utility” to check if the plugged PPCle-8443 boards are successfully installed and see the Base address and IRQ level assigned by BIOS.

One thing to be noticed is to identify the board number of PPCle-8443 when multiple boards are applied. There are two methods to identify the board number. One is automatic setting and the other is setting by the Card ID switch (SW1).

Using automatic setting, the number of a PPCle-8443 board depends on the locations on the PCIe slots. Generally, they are numbered in order from the board closer to the PC.

These card numbers will influence the corresponding axis number on the boards. And the axis number is the first argument for most functions called in the library. So it is important to identify the axis number before writing application programs.

For manual setting (setting by board ID setting switch (SW 1)), see section 3.11SW 1 board ID selection. Correspondence between board number and axis number is as shown in the below table as with automatic setting.

Axis No Board No	Axis 0	Axis 1	Axis 2	Axis 3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15
4	16	17	18	19
5	20	21	22	23
6	24	25	26	27
7	28	29	30	31
8	32	33	34	35
9	36	37	38	39
10	40	41	42	43
11	44	45	46	47

If you want to accelerate Axis 2 of board 1 from 0 to 10000 pps in 0.5 sec in the Constant Velocity Mode operation, the axis number should be 6, and the code on the program will be as follows:

```
_8443_start_tv_move(6,      // axis number
0,      // starting speed
10000,  // moving maximum speed
0.5);   // acceleration time
```

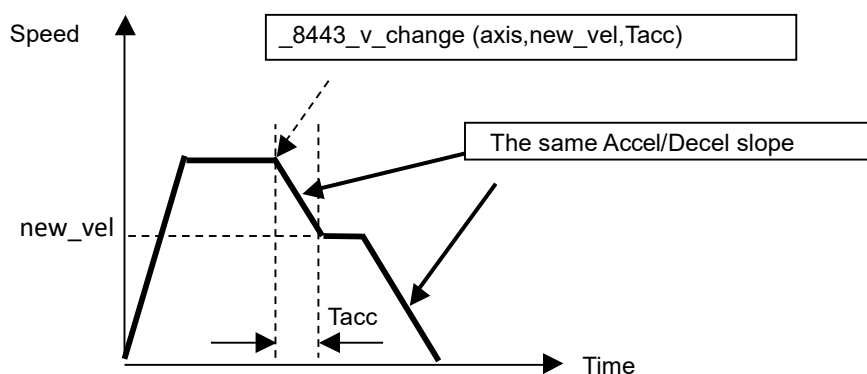
Please check the board number (axis number) using PPCle8443 Utility before application development.

For applications to move many axes simultaneously by using multiple PPCle-8443 boards, you should follow the connection diagrams in section 4.14 to make connections with K1/K2 connectors. Several functions described in section 7.19 may be useful when writing programs.

## 5.6. Change Position or Speed on-the-fly (Override Function)

The PPCle-8443 provides powerful position or speed changing function while an axis is moving. Changing speed / position on the fly means that the target speed/position can be altered after the operation has started. Yet, these functions are not unlimited, so please study carefully all constraints before implementing on-the-fly (override) functions.

### 5.6.1 Change Speed on-the-fly (Speed Override)



The change speed on the fly function is applicable in single axis motion only. Both velocity mode operation and position mode operation are applicable. The graph above shows the basic operating theory.

The following functions are related to change speed on the fly function:

<b>_8443_v_change():</b>	Change speed on the fly(Change MaxVel)
<b>_8443_cmp_v_change():</b>	Change velocity when general comparator is satisfied
<b>_8443_sd_stop():</b>	Ramp down to stop
<b>_8443_emg_stop():</b>	Emergency stop
<b>_8443_fix_speed_range():</b>	Define the speed range
<b>_8443_unfix_speed_range():</b>	Release the speed range constrain

All the first 4 functions enable speed changes during a single axis operation.

However,

**\_8443\_sd\_stop()** and **\_8443\_emg\_stop()** only change the axis speed to "0".

**\_8443\_fix\_speed\_range()** is necessary to be executed in prior to any **\_8443\_v\_change()** functions.

**\_8443\_unfix\_speed\_range()** release the speed range constrained by **\_8443\_fix\_speed\_range()**.

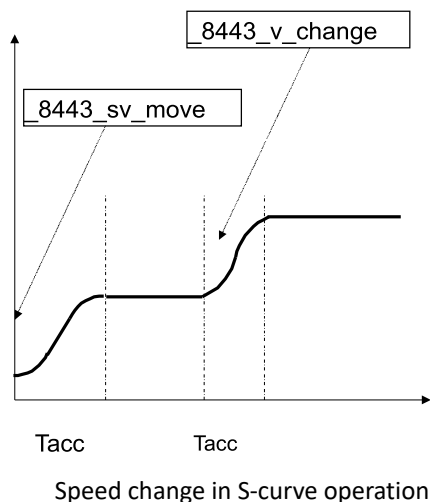
The **\_8443\_cmp\_v\_change()** has almost the same function as **\_8443\_v\_change()**, except that **\_8443\_cmp\_v\_change()** will act only when a general comparator is satisfied. See subsection 5.4.4 for more detail description about general comparators.



Next, we will focus on the `_8443_v_change()`.

#### Work theory of `_8443_v_change()`:

`_8443_v_change()` function is used to change the **MaxVel** on the fly. In a normal operation, the axis starts at **StrVel** speed, accelerates to **MaxVel**, and then keeps at **MaxVel** until entering deceleration section. If you change the **MaxVel**, it will force the axis to accelerate or decelerate to a new speed in a period of time defined by you. Both Trapezoidal and S-curve profiles are applicable.



#### Constraints of `_8443_v_change()`:

1. In single axis preset mode, there must be enough remaining pulses to reach new velocity. If not, the `_8443_v_change()` will return error and keep the velocity unchanged.

#### Example:

A trapezoidal relative motion is applied:

`_8443_start_tr_move(0,10000,0,1000,0.1,0.1).`

It will get axis 0 move for 10000 pulses, and the maximum velocity is 1000 pps.

At 5000 pulse, the `_8443_v_change(0,NewVel,Tacc)` is applied.

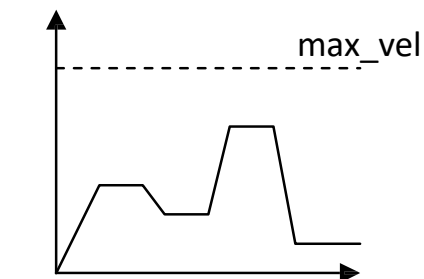
NewVel (pps)	Tacc (sec)	Necessary remaining pulse			OK/ Error
		Accel	Decel	Total	
5000	0.1	300	313	613	OK
5000	1	3000	3125	6125	Error
10000	0.1	550	556	1106	OK
50000	0.1	2550	2551	5101	Error

In the case that NewVel is 5000 and Tacc is 1 sec, there must be 6125 remaining pulses, and amount number of pulses will exceed the target position of 10000. Therefore, an error occurs.

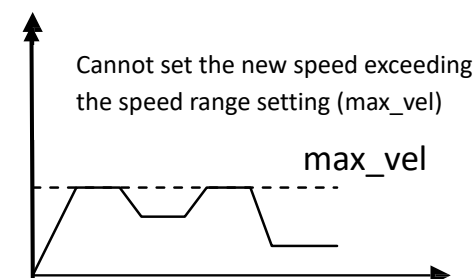
In the case that NewVel is 50000 and Tacc is 0.1 sec, 5101 remaining pulses are needed and amount number of pulses will exceed the target position 10000. Therefore, an error occurs again.

In other two combinations, speed changes can be executed because there are enough remaining pulses.

2. Please check the board number (axis number) using PPCle-8443 Utility before application development. You must set the maximum speed by `_8443_fix_speed_range()` so that the `_8443_v_change()` can work correctly. If not, the MaxVel set by `_8443_v_move()` or `_8443_start_ta_move()` becomes automatically lower the maximum speed since the maximum speed set by `_8443_v_change()` cannot be reached. .

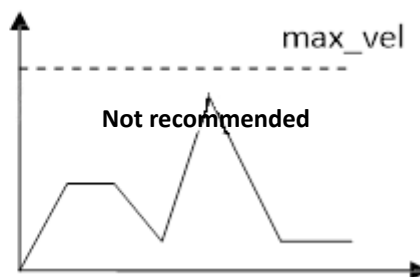
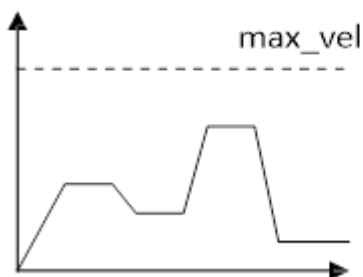


When setting the speed range including the target speed in advance with `fix_speed_range`



When `fix_speed_range setting` is not done in advance

3. `_8443_v_change()` function conducted during acceleration or deceleration period is not recommended. Although the function may work in most cases, the acceleration and deceleration times cannot be guaranteed.

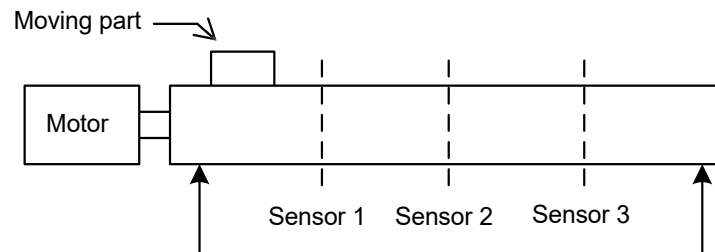


**Example:**

There are three speed change sensors during an absolute operation for 200,000 pulses.

The initial maximum speed is 10,000 pps. It will be changed to 25,000 pps if Sensor 1 is touched, it will be changed to 50,000 pps if Sensor 2 is touched, and it will be changed to 100,000pps if Sensor 3 is touched.

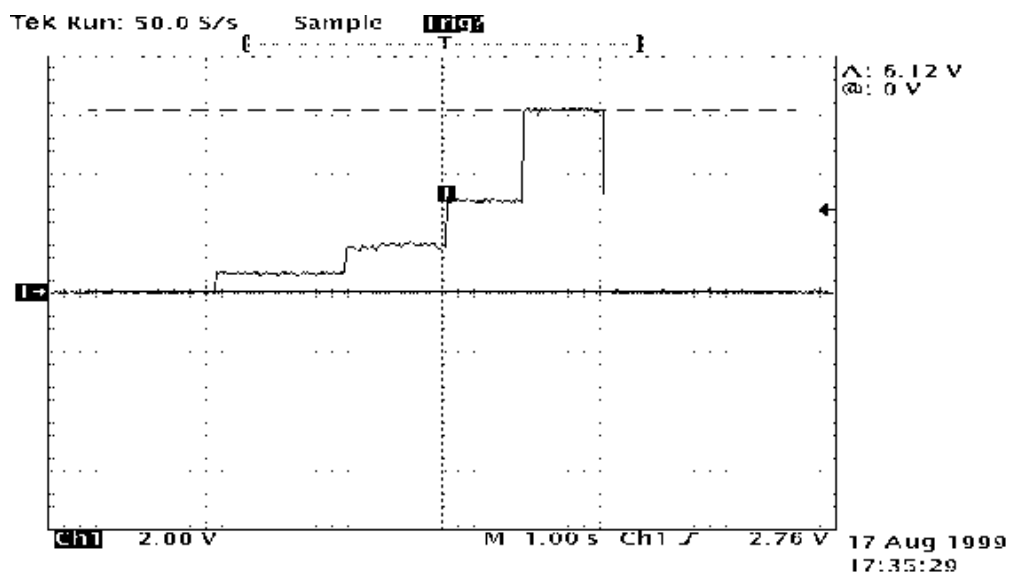
The code for this application and the resulting velocity profiles are shown as follows (Descriptions for sensor status acquisition are omitted here):



```
_8443_fix_speed_range(axis,100000.0);
_8443_start_ta_move(axis,200000.0,1000,10000,0.02,0.01);
while(!_8443_motion_done(axis))
{
    // Get sensor information from other I/O board

    if((Sensor1==High)&&(Sensor2==Low)&&(Sensor3==Low))
        _8443_v_change(axis,25000,0.02);
    else if((Sensor1==Low)&&(Sensor2==High)&&(Sensor3==Low))
        _8443_v_change(axis,50000,0.02);
    else if((Sensor1==Low)&&(Sensor2==Low)&&(Sensor3==High))
        _8443_v_change(axis,100000,0.02);
}
```

The information of three sensors is acquired from other I/O board. The resulting velocity profile from the experiment is shown below.



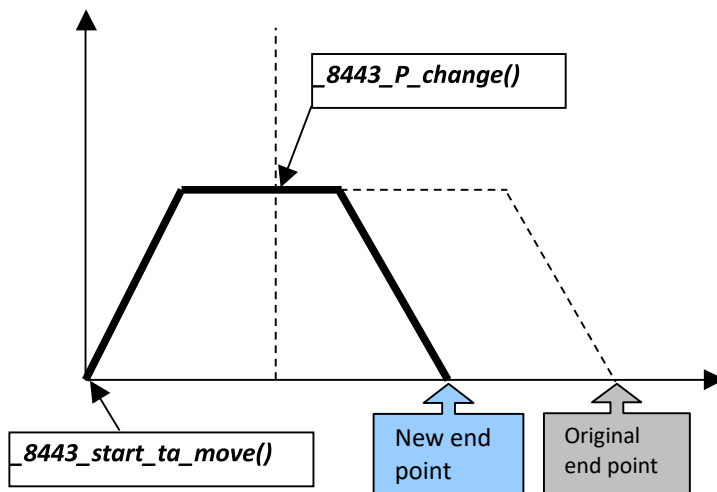
Related functions:

`_8443_v_change()`, `_8443_sd_stop()`, `_8443_emg_stop()`,  
`_8443_fix_speed_range()`, `_8443_unfix_speed_range()`,  
`_8443_get_currebt_speed()`

: See section 7.5.

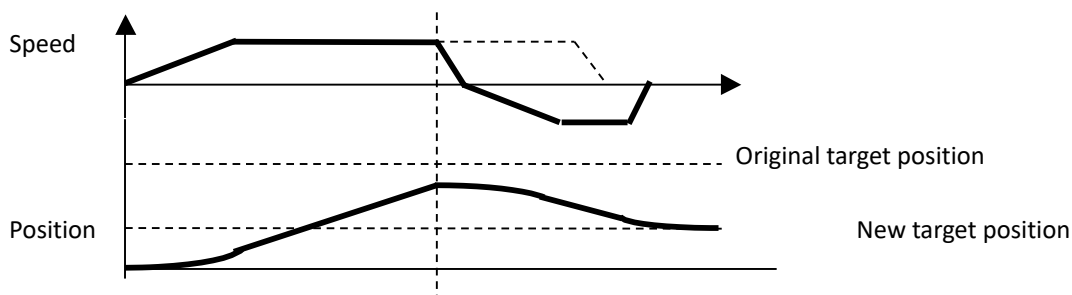
## 5.6.2 Change Position on-the-fly (Position Override)

When operating in the single-axis absolute pre-set motion, it is possible to change the target position during the operation by function call `_8443_p_change()`.



### Work theory of `_8443_p_change()` :

`_8443_p_change()` is applicable on `_8443_start_ta_move()`, and `_8443_start_sa_move()` only. It is used to change the target position that has been defined originally by these two functions. After changing the position, the axis will move to the new target position and totally disregard the original position. If the new position has been already passed, the axis will Decelerate to stop, and will reverse as shown in the below figure. The acceleration and deceleration rate, the StrVel and MaxVel, will be kept the same as the original settings.



### Constrains of `_8443_p_change()`:

1. `_8443_p_change()` is only applicable in single-axis absolute pre-set motion, ie, `_8443_start_ta_move()`, and `_8443_start_sa_move()` only.
2. Position change in the deceleration period is not allowed.
3. There must be enough distance between the new target position and the current position when `_8443_p_change()` is executed because PPCle-8443 needs enough pulses to finish the deceleration operation.

**Example:**

When a trapezoidal absolute positioning operation is applied:

**`_8443_start_ta_move(0,10000,0,1000,0.5,1):`**

Axis 0 will move to the pulse 10000 position, and the maximum velocity is 1000 pps.

The required number of pulses to decelerate is  $0.5 * 1000 * 1 = 500$ .

At the current position described below, using **`_8443_p_change(0, NewPos)`**, you will override the target position when New Position of 5000 with OK in the below table.

New Positon	Current Position	OK/ Error	Note
5000	4000	OK	
5000	4501	Error	
5000	5000	Error	
5000	5499	Error	
5000	6000	OK	Reverse operation
5000	9499	OK	Reverse operation
5000	9500	Error	
5000	9999	Error	

An error occurs if there are not enough pulses between the current position and the new position or the deceleration has already started when the function is executed.

**Related functions:**

**`_8443_p_change()`**: See section 7.6.

## 5.7. Comparator and Latch

PPCLe-8443 provides position comparator functions in axis 0 and 1, and position latch functions in axis 2 and 3. The comparator function is to output a trigger pulse when the counter reaches the value you set. CMP0 (Axis0) and CMP1 (Axis 1) are used for comparator triggers. The latch function is to capture values of all 4 counters (see section 5.4) at the instant when the latch signal is activate.

LTC2 (Axis 2) and LTC3 (Axis 3) in CN2 are used to receive latch pulses input.

### 5.7.1 Comparator of PPCLe-8443

There are five comparators in every axis of PPCLe-8443. Each comparator has a unique functionality. See the table below:

	Comparator source	Description	Related function
Comparator 1	Command position counter	Software limit (+) (See section 5.9.)	<i><code>_8443_set_soft_limit</code></i>
Comparator 2	Command position counter	Software limit (–) (See section 5.9.)	<i><code>_8443_enable_soft_limit</code> <code>_8443_disable_soft_limit</code></i>
Comparator 3	Position error counter	Out of step detection	<i><code>_8443_error_counter_check</code></i>
Comparator 4	Any counters	General-purpose	<i><code>_8443_set_general_comparator</code></i>
Comparator 5 (Axis 0 and 1 only)	Any counters	Position comparison function (Trigger)	<i><code>_8443_set_trigger_comparator</code> <code>_8443_build_compare_function</code> <code>_8443_build_compare_table</code> <code>_8443_set_auto_compare</code></i>

Note: Not all of the five comparators have the ability to output a trigger pulse via CMP. It is only the comparator 5.

The comparator 1 and 2 are for software limits (see section 5.9).

The comparator 3 is used to compare with position error counter. It is very useful for out-of-step detection of stepping motors. To enable or disable out-of-step detection and the allowable tolerance can be set by the function:

*`_8443_set_error_counter_check()`*.

PPCLe-8443 will generate an interrupt signal if the out-of-step detection is enabled and out-of-step is detected.

The comparator 4 is a general-purpose comparator, which will generate an interrupt (default setting) if the comparator condition is satisfied. The comparator source counter can be any counter. The comparator value, source counter, comparison method, and the reaction can be set by the software function call *`_8443_set_general_comparator()`*.

## 5.7.2 Position Comparator

The position comparator function is performed by the 5th comparator. Only the first two axes (0 and 1) can perform position comparator functions. The position comparator function is to trigger a pulse output via CMP, when the comparison condition is satisfied.

The comparing condition consists of two parts; the first is the value to be compared, and the other is the comparison modes. Comparison mode can be ">", "=", or "<". The easiest way to use position comparator function is to call the software function:

**\_8443\_set\_trigger\_comparator(AxisNo, CmpSrc, Method, Data)**

The third parameter "Method" indicates the comparison method, while the fourth "Data" is for value to be compared. In continuous comparison, this data will be ignored automatically since the comparison data will be created by other functions.

### Continuous comparison function

For users who want to compare multiple data continuously, functions to build comparison tables are provided as follows:

**\_8443\_build\_comp\_function(AxisNo, Start, End, Interval, device)**

**\_8443\_build\_comp\_table(AxisNo, tableArray, Size, device)**

**\_8443\_set\_auto\_compare(AxisNo, SelectSource)**

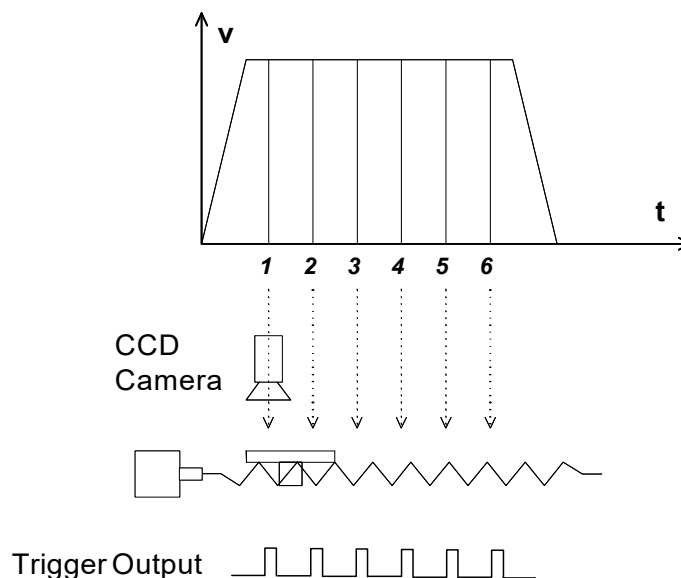
**Note: 1. Please turn off all interrupt functions when the above functions are running**

\_8443\_build\_comp\_function () is used to build a compare list by the start point, the end point and the constant interval.

\_8443\_build\_comp\_table () is used to build an arbitrary comparing table (data array).

\_8443\_set\_auto\_compare() is a comparison source selection function. Please put a value "1" in this parameter for using FIFO mode. Once it is set, the comparison mechanism will start. You can check the current value which is going to be compared by \_8443\_check\_compare\_data():

The following is an example of using continuous position comparison functions:



In this application, the table is controlled by the motion command and the CCD Camera is controlled by the position comparator output. The image of moving object can be obtained in this way easily.

**Working Specification:** 34000 triggering points per stroke and trigger speed is 6000 (pts / s)



Program will be set as follows:

- Table will start moving from counter position 0 to 36000.
- Comparing points are from 1001 to 35000 and the total will be 34000 pts. The points to points interval = 1 pulse.
- Moving speed is 6000 pps
- The comparison condition is “=”.

Trigger output per matching of the counter value and the data set by this function.

```
_8443_set_trigger_comparator(0, 1, 1, 1001);//set comparison method
```

```
_8443_build_compare_function(0, 1001, 35000, 1, 1); //build a constant interval comparison data table
```

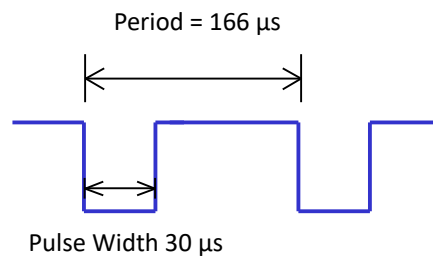
```
_8443_set_auto_compare(0, 1);//start continuously comparison
```

```
_8443_start_tr_move(0, 36000, 0, 6000, 0.01, 0.01);//axis operation starts
```

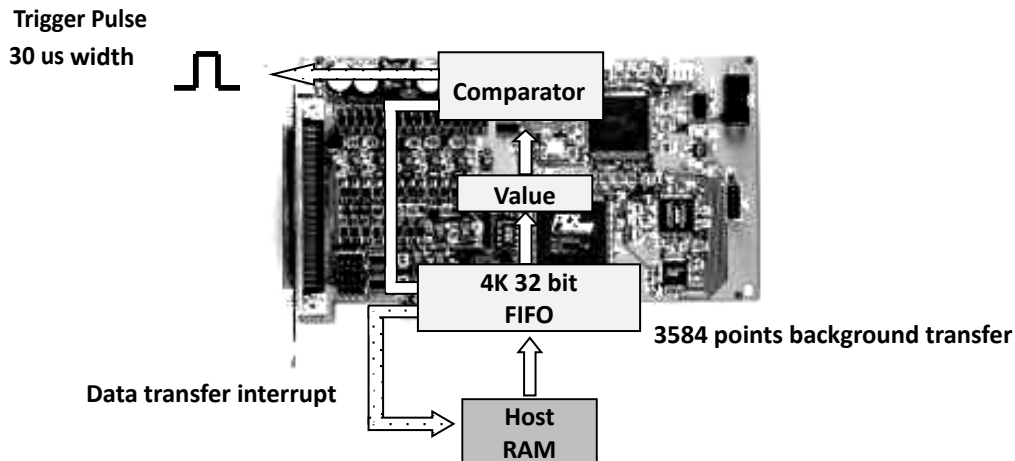
You can confirm the current comparison data and check whether the continuous comparator function is operating with the following function:

```
_8443_check_compare_data(0, 5, *CurrentData);
```

Operation results:



The mechanism of the continuous comparator function is as follows:



The “Value” block in the figure means the position which is going to be compared, and you can use **\_8443\_check\_compare\_data()** function to check it. Please note that at the final comparison point, it will still load an “after-final” point into the “Value” block. Please fill a dummy point into the comparison table array at the final position, and this value must be far away from the table’s stroke.

If using **\_8443\_build\_compare\_function()**, it will load a dummy “after-final” point automatically.

The value is 134217727.

Relative Function:

```
_8443_set_trigger_comparator(), _8443_build_compare_function(),
```

```
_8443_build_compare_table(), _8443_set_auto_compare(),
```

```
_8443_check_compare_data(), _8443_set_trigger_type ()
```

: See section 7.17.

### 5.7.3 Position Latch

The position latch function is to capture all counters' data instantly on receiving the LTC pulse input. The latency between occurring of latch signal and finishing of position capturing is extremely short, since the latching procedure is made by hardware. Only the last two axes (2 and 3) can do position latch function. LTC2 (Axis 2) and LTC3 (Axis 3) are used to receive the latch pulse.

To set the latch logic: `_8443_set_ltc_logic()`.

To get the latched values of counters: `_8443_get_latch_data`(AxisNo, CntNo, Pos).

The second parameter "CntNo" is used to indicate the counter of which the latched data will be read.

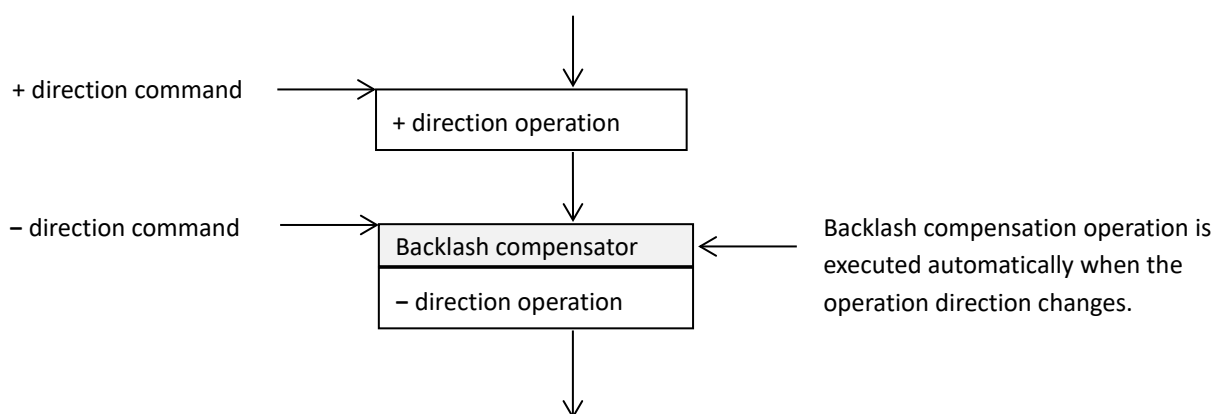
And, latch such as ORG input, besides the input by LTC terminal, can be set by the `_8443_set_enable_inp()` command.

Related functions:

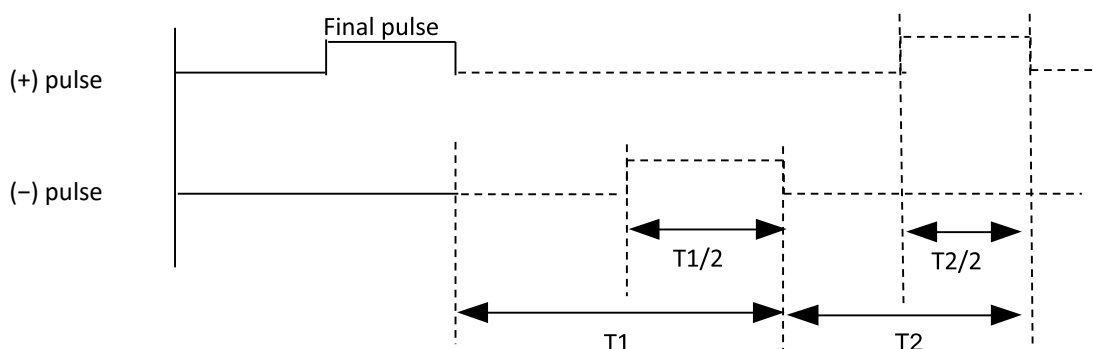
`_8443_set_ltc_logic()`, `_8443_get_latch_data()`: See section 7.17.

## 5.8. Backlash Compensator and Vibration Suppression

Whenever direction change is occurred, PPCle-8443 can output backlash corrective pulses before sending commands. The function `_8443_backlash_comp()` can set the number of pulses.



In order to minimize the vibration when a motor stops, PPCle-8443 can output a single pulse for negative direction and then a single pulse for positive direction right after completion of command movement. See the following figure: the function `_8443_suppress_vibration()` is used to set the T1 & T2.



Related functions:

`_8443_backlash_comp()`, `_8443_suppress_vibration()`: See section 7.6.

## 5.9. Software Limit Function

PPCLe-8443 provides two software limits for each axis. The soft limit is extremely useful to protect your mechanical system, and it works the same as a physical limit switch.

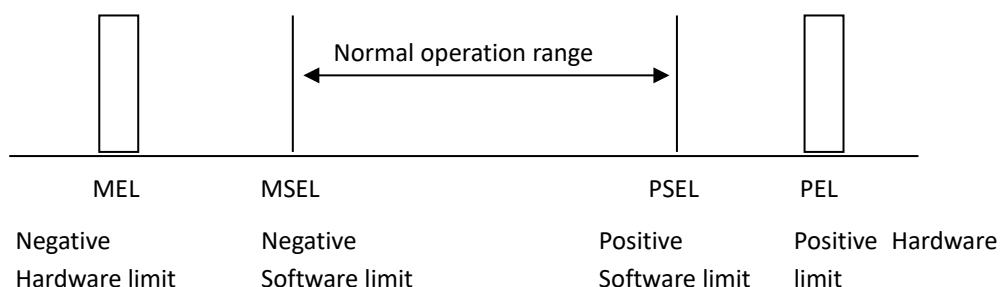
The software limit is built-in the comparator 1 and 2 (see subsection 5.7.1), and the compare source is the command position counter. The working theory is that if you pre-set limit values on comparator 1 and 2, when the command position counter reached the set limit values, PPCLe-8443 responses the same way as the physical limit switch is touched. Then, the motor stops immediately or decelerates to stop.

**\_8443\_set\_soft\_limit():** To set the software limit

**\_8443\_enable\_soft\_limit():** To enable the software limit

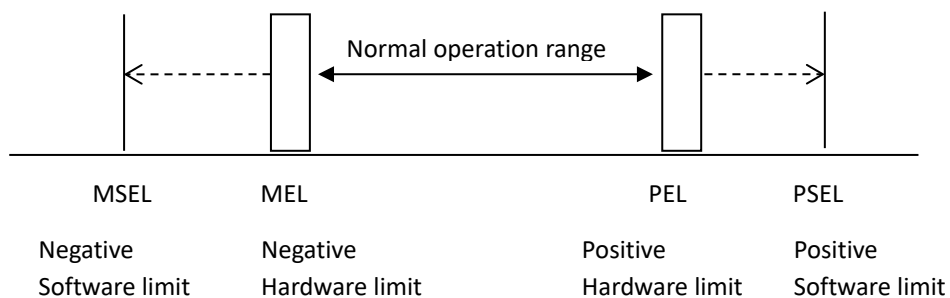
**\_8443\_diable\_soft\_limit():** To disable the software limit

[Setting example A]



By setting a software limit inside the hard limit, even if a large amount of movement is mistakenly set / commanded, it stops before the hardware limit.

[Setting example B]



By setting a software limit outside the hardware limit, even if the hard limit signal does not turn on due to sensor malfunction or the like, it is used as a failsafe to stop at a position slightly exceeding the hardware limit.

Note: The software limit will work based on the command position counter, not the feedback position counter (please refer to 5.4).

Unlike the physically placed hardware end limit sensor etc., please note that the software limit is set based on the position of the command counter 0.

Related functions:

**\_8443\_set\_soft\_limit(), \_8443\_enable\_soft\_limit(), \_8443\_diable\_soft\_limit()**

: Refer to section 7.17.

## 5.10. Interrupt Control

PPCLe-8443 can generate INT signal to the host PC. The parameter “intFlag” of software function call `_8443_int_control()`, can enable/disable the interrupt operations.

After an interrupt occurs, the function `_8443_get_int_status()` is used to receive the INT status, which contains information about INT signal.

The interrupt cause of PPCLe-8443 consists of an error interrupt factor (***error\_int\_status***) and an event interrupt factor (***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 `_8443_set_int_factor()`.

The ***error\_int\_status*** is for abnormal stop of PPCLe-8443. For example: EL, ALM ... etc., these kind of INT cannot be masked.

The following is the definitions of these two `int_status`:

<b><i>event_int_status</i></b> : Enable / disable of each bit can be set with function <code>_8443_int_factor()</code>	
Bit	Description
0	Normal stop
1	Next command starts
2	Command pre-register 2 is empty and can be written
3	(Reserved)
4	Acceleration Start
5	Acceleration End
6	Deceleration Start
7	Deceleration End
8	(Reserved)
9	(Reserved)
10	Out of step occur
11	General comparator compared(condition is satisfied)
12	Comparator triggered for axis 0 and axis1(satisfied)
13	(Reserved)
14	Counter Latched for axis 2 and axis 3
15	ORG input and latched
16	SD on
17	(Reserved)
18	CSTA, sync. start on
19	(Reserved)
20 ~ 31	(Reserved)

<b>error_int_status: cannot be masked if interrupt operation is activated</b>	
Bit	Description
0	+ Software limit on and stop
1	– Software limit on and stop
2	(Reserved)
3	General comparator on and stop
4	(Reserved)
5	+ End Limit on and stop
6	– End Limit on and stop
7	ALM happen and stop
8	CSTP, Sync. stop on and stop
9	CEMG, Emergency on and stop
10	SD on and slow down to stop
11	(Reserved)
12	Interpolation error and stop
13	Other axes 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)

#### Use events to deal with interrupt under Windows

In order to detect the interrupt signal from PPCle-8443 under Windows, you must create an event array first. Then use functions provided by PPCle-8443 to obtain the interrupt status. The sample program description is as follows:

1. Define the event array as a global value to deal with interrupt events. Please define the event array by the same element count as the maximum axis No. (+1) of PPCle-8443. Each event is linked to one axis

```
HANDLE hEvent[4]; // Define the event array for 4 axes
```

2. Enable an interrupt event operation, and set up the interrupt factors and enable the interrupt channel:

```
_8443_int_enable(0,&hEvent[0]); // Interrupt event setting
```

```
_8443_set_int_factor(0,0x01); // Interrupt factor setting(normal stop interrupt)
```

```
_8443_int_control(0,1); // Enable interrupt event
```

3. Start operation command

```
_8443_start_tr_move(0,12000,0,10000,0.1,0.1);
```

4. Wait axis 0 interrupt event

```
STS=WaitForSingleObject(hEvent[0],15000);
```

```
ResetEvent(hEvent[0]);
```

```
if(STS==WAIT_OBJECT_0)
```

```
{
```

```
    _8443_get_int_status(0,&error,&event); // Get interrupt event
```

```
    if(event==0x01) ..... ; // Successful
```

```
}
```

```
else if(STS==WAIT_TIME_OUT)
```

```
{
```

```
    // Time out, Failed
```

```
}
```

Now, you can receive interrupts from each axis with such a description.

Related functions:

`_8443_int_control()`, `_8443_set_int_factor()`, `_8443_int_enable()`,  
`_8443_int_disable()`, `_8443_get_int_status()`, `_8443_link_interrupt()`,

: See section 7.15.

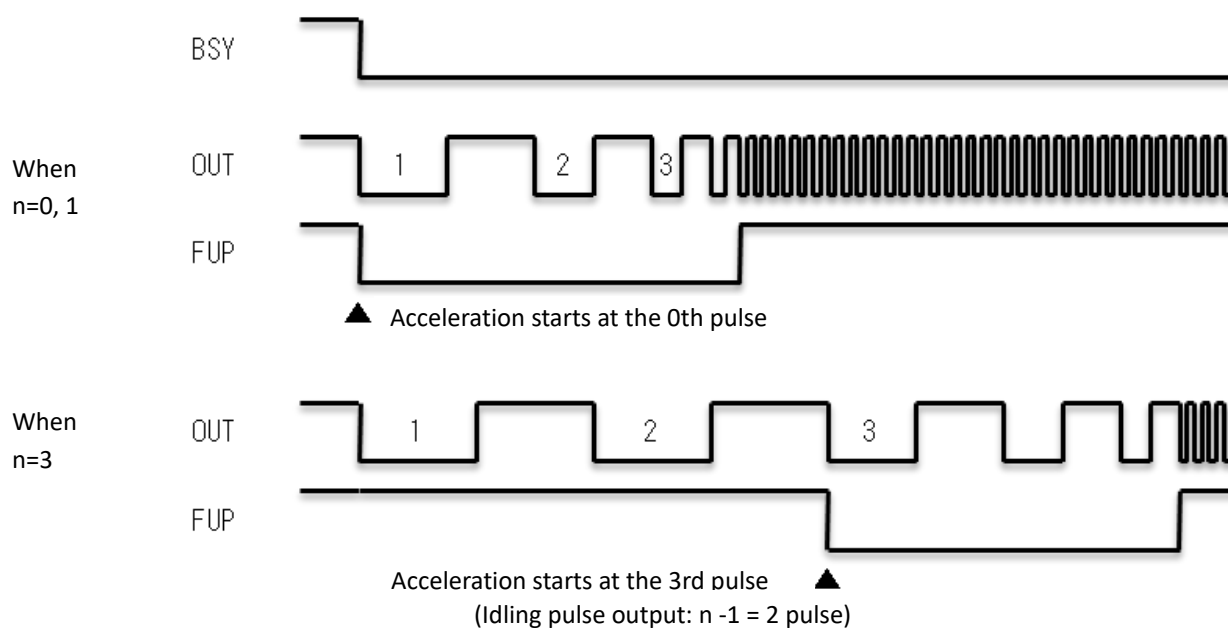
## 5.11. Idling Control

In this mode, acceleration can be started after outputting several idling pulses at the start speed (StrVer) in acceleration / deceleration operations.

The pulse number setting on *idl\_pulse* parameter of *\_8443\_set\_idle\_pulse()* command define the delay time of the acceleration. Even when this function is used on position mode, the total moving distance will remain unchanged.

The timing diagram of the idling pulse setting and how acceleration begins are shown as follows:

A value of "*idl\_pulse* parameter set value - 1" is applied to the idling pulse.



Related functions:

*\_8443\_set\_idle\_pulse()*: See section 7.6.

## 6. PPCle-8443 Utility

PPCLe-8443 Utility provides a simple, yet powerful means to setup, configure, test and debug a motion control system that uses PPCle-8443 boards.

After installing all the hardware properly according to Chapter 2 and 3, it is necessary to correctly configure boards and double check before running. This chapter gives guidelines for establishing a control system and manually exercising PPCle-8443 board to verify the correct operations.

### 6.1. Execute PPCle-8443 Utility

After installing the software driver of PPCle-8443 on Windows 7/8/10, PPCle-8443 Utility program can be found in <chosen path> \PPCLe-8443\Utility. To execute it, double click it or use desk top “Start”      Program files”      “PPCLe-8443”  
“PPCLe8443 Utility”.

### 6.2. About PPCle-8443 Utility

Before running PPCle-8443 Utility, note the following items:

- 1 PPCle-8443 Utility is a program written by VB 6.0, and is useable only for Windows with the screen resolution higher than 800 x 600 environments.
- 2 PPCle-8443 Utility allows you to save settings or configurations for PPCle-8443 boards and the saved settings and configurations will be loaded automatically when PPCle-8443 Utility is executed later again. The two files **8443.ini** and **8443MC.ini** in **windows root directory** are used to save all settings and configurations.
- 3 To duplicate configurations from one system to another system, just copy 8443.ini and 8443MC.ini into windows root directory.
- 4 If you want to use the configurations set by PPCle-8443 Utility, the DLL function call **\_8443\_config\_from\_file()** is helpful. After calling this function in your program, you can use those PPCle-8443 boards as the same configuration as set by PPCle-8443 Utility.



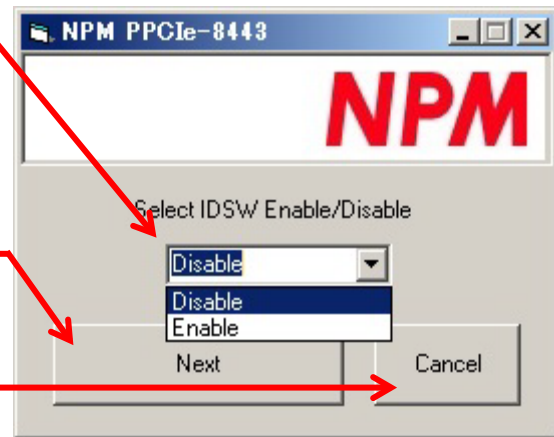
## 6.3. PPCle8443 Utility Screen Introduction

### 6.3.1 Board ID Switch Enable / Disable Screen

The Board ID switch Enable / Disable screen appears after starting PPCle8443 Utility.  
In this screen, select whether to enable or disable board ID switch.

- Select the Board ID switch Enable or Disable.  
When Enable is selected, board ID setting by SW1 is applied.  
See section 5.5 for details of board ID setting.

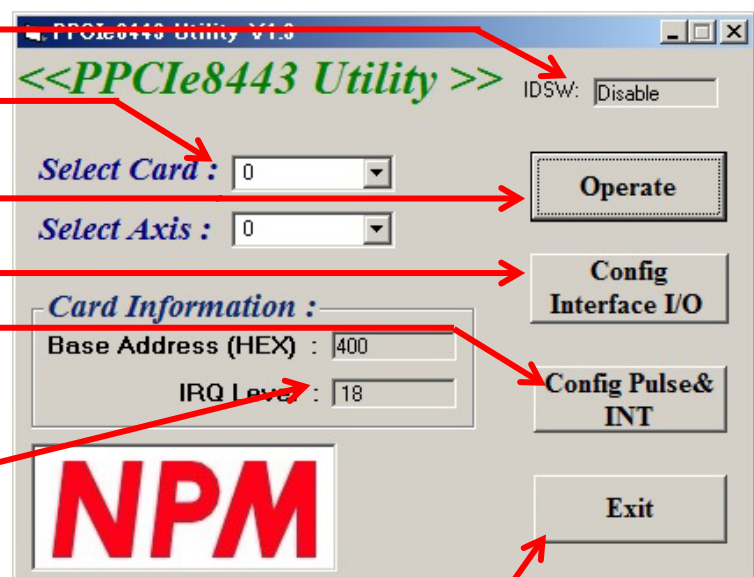
- Go to Main Screen
- Exit PPCle8443 Utility



### 6.3.2 Main Screen

The main screen is displayed as follows:

- Board ID switch enable / disable status indication
- Select operation board and axis
- Configuration screen (see section 6.3.4.)
- Go to Interface I/O Configuration Screen (see subsection 6.3.2.)
- Go to **Pulse & INT** configuration screen (see subsection 6.3.3.)
- Board information:  
Related functions  
`_8443_get_base_addr()`,  
`_8443_get_irq_channel()`
- Exit PPCle8443 Utility



### 6.3.3 Interface I/O Configuration Screen

In this screen, you can set configurations of EL, ORG, EZ, ERC, ALM, INP, SD, and LTC.

Interface I/O Configuration

PPC1e8443 Axis Number = 0

**Servo Motor Signal**

**ALM**

Logic: ☒ Active Low ☐ Active High

Response Mode: ☒ Stop immediately ☐ Dec. to Stop

**INP**

Logic: ☒ Active Low ☐ Active High

Enable/Disable: ☒ Disable INP ☐ Enable INP

**ERC**

Logic: ☒ Active Low ☐ Active High

Active Timing: ☒ 1.6 ms ☐ 13 ms ☐ 52 ms ☐ 104 ms

**Mech Signal**

**EL**

Response Mode: ☒ Stop immediately ☐ Dec. to Stop

**ORG**

Logic: ☒ Active Low ☐ Active High

**EZ**

Logic: ☒ Active Low ☐ Active High

**LTC**

Logic: ☒ Active Low ☐ Active High

**SD**

Enable/Disable: ☒ Disable SD ☐ Enable SD

Logic: ☒ Active Low ☐ Active High

Response Mode: ☒ Slow Down Only ☐ Slow Down Then Stop

SD Latch: ☐ Disable SD latch ☒ Enable SD latch

Buttons: Next Axis, Save Config, Operate, Config Pulse & INT, BACK

1. **ALM Logic and Response mode:** Select the logic and the response mode of ALM signal. The related function call is `_8443_set_alm()`.
2. **INP Logic and Enable / Disable selection:** Select the logic and Enable / Disable of INP signal. The related function call is `_8443_set_inp()`.
3. **ERC Logic and Active timing:** Select the logic and active timing (pulse signal width) of ERC signal. The related function call is `_8443_set_erc()`.
4. **EL Response mode:** Select the response mode of EL signal. The related function call is `_8443_set_el()`.
5. **ORG Logic:** Select the logic of ORG signal. The related function call is `_8443_set_home_config()`.
6. **EZ Logic:** Select the logic of EZ signal. The related function call is `_8443_set_home_config()`.
7. **SD Configuration:** Configuration of SD signal. The related function call is `_8443_set_sd()`.
8. **LTC Logic:** Select the logic of LTC signal. The related function call is `_8443_set_ltc_logic()`.
9. **Buttons:**
  - **Next Axis:** Click this button to change operating axis.
  - **Save Config:** Click this button to save current configuration to 8443.ini.
  - **Operate:** Go to operate form (see subsection 6.3.4.)
  - **Config Pulse & INT:** Go to Pulse IO & Interrupt Configuration Form (see subsection 6.3.3.)
  - **Back:** Click this button to go back main screen.

## 6.3.4 Pulse I/O and interrupt configuration screen

In this screen, you can set the configuration of pulse input / output, move ratio, and INT factor.

**Pulse IO & Interrupt Configuration**  
PPC1e8443 Axis Number = 0

**Pulse Output Mode** (1)

- ☒ OUT/DIR -- OUT is falling edge, DIR+ is high level
- ☐ OUT/DIR -- OUT is rising edge, DIR+ is high level
- ☐ OUT/DIR -- OUT is falling edge, DIR+ is low level
- ☐ OUT/DIR -- OUT is rising edge, DIR+ is low level
- ☐ CW/CCW Falling edge
- ☐ CW/CCW Rising edge
- ☐ A-B phase (OUT is leading 90 degrees phase to DIR)
- ☐ A-B phase (OUT is lagging 90 degrees phase to DIR)

**Pulse Input (Feedback Counter)** (2)

**Source**

- ☒ Encoder (External)
- ☐ Pulse Output (Internal)

Move Ratio (Feedback/Command) =

**Mode**

- ☒ 1X A/B Phase
- ☐ 2X A/B Phase
- ☐ 4X A/B Phase
- ☐ CW / CCW

**Logic**

- ☒ Inverse the Direction
- ☐ Do Not Inverse Direction

**INT Factor** (3)

- Bit 0 ☐ Normal Stop
- Bit 1 ☐ Next Command Continued
- Bit 2 ☐ Continuous Pre-register is Empty
- Bit 4 ☐ Acceleration Start
- Bit 5 ☐ Acceleration End
- Bit 6 ☐ Deceleration Start
- Bit 7 ☐ Deceleration End
- Bit 10 ☐ Position Error Occur
- Bit 11 ☐ General Comparator Compared
- Bit 12 ☐ Compare Trigger for Axis 0,1
- Bit 14 ☐ Latch for Axis 2,3
- Bit 15 ☐ ORG On
- Bit 16 ☐ SD On

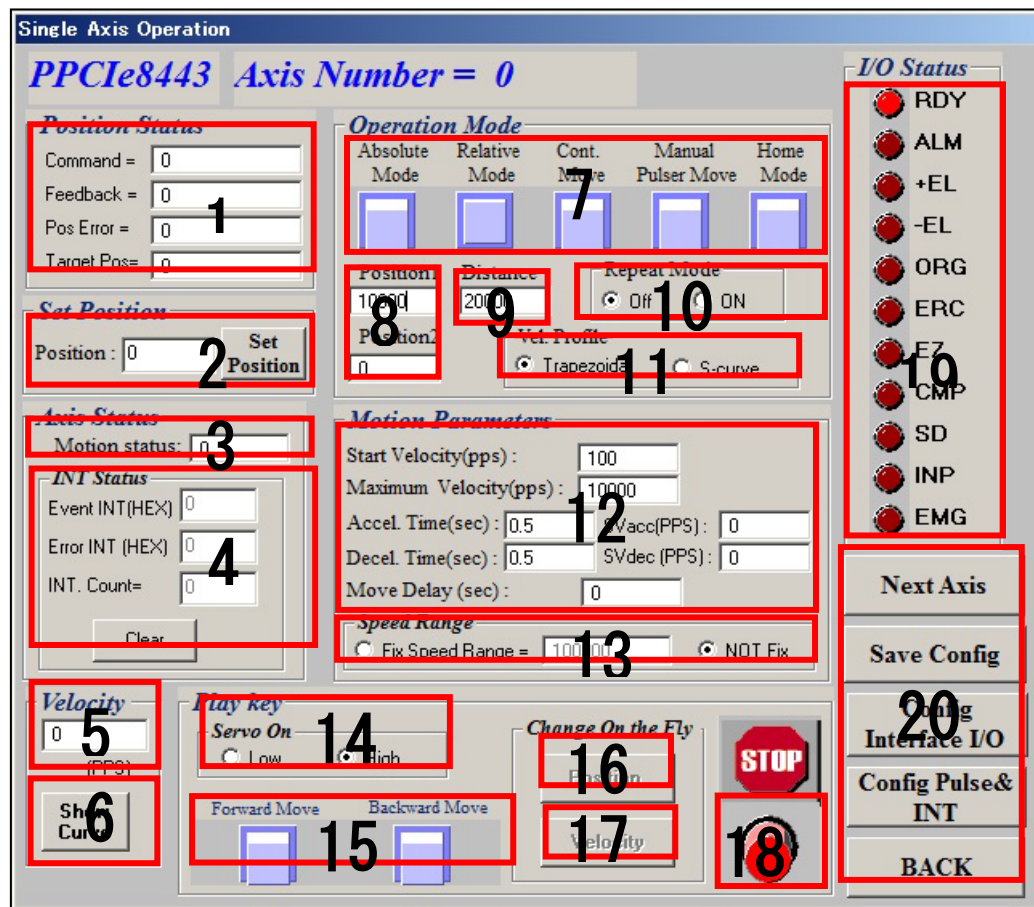
**Buttons** (4)

- Next Axis
- Save Config
- Operate
- Config Interface I/O
- BACK

- Pulse Output Mode:** Select the output mode of pulse signal (OUT / DIR). The related function call is `_8443_set_pls_outmode()`.
- Pulse Input:** Set the configurations of Pulse input signal (EA / EB). The related function call is `_8443_set_pls_iptmode()`, `_8443_set_feedback_src()`.  
**Move Ratio:** Set the resolution ratio (feedback / pulse command) for current target axis. The value should not be '0'. The related function call is `_8443_set_move_ratio()`.
- INT Factor:** Select factors to initiate the event INT. The related function call is `_8443_set_int_factor()`.
- Buttons:**
  - Next Axis:** Click this button to change the operating axis.
  - Save Config:** Click this button to save current configuration to 8443.ini.
  - Operate:** Go to operate form (see subsection 6.3.4.)
  - Config Interface I/O:** Go to Interface I/O Configuration Form (see subsection 6.3.2.)
  - Back:** Click this button to go back to Main screen.

## 6.3.5 Operation screen

This is the main screen for various operations for each axis (velocity mode, preset relative / absolute, manual pulser, and home return).



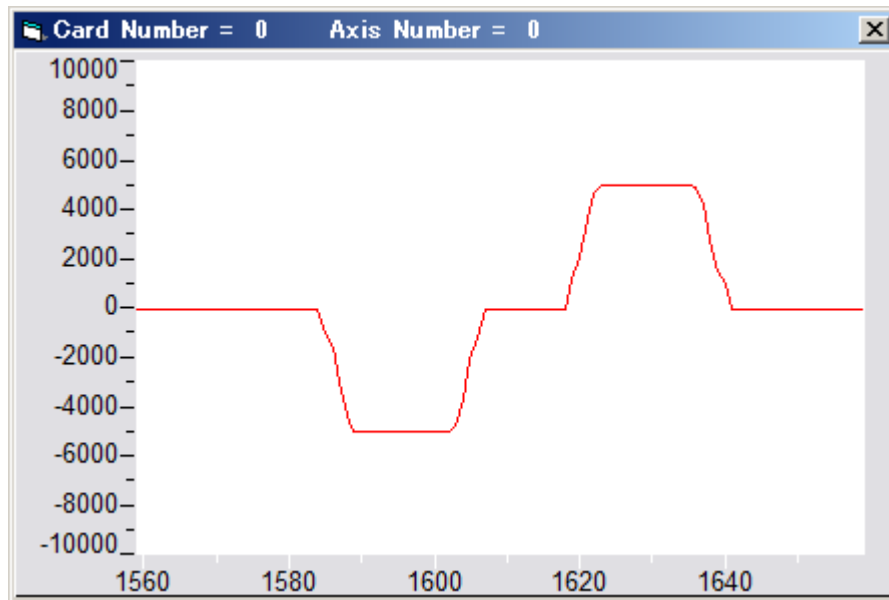
1. **Position:**
  - \*Command: display the value of command counter. The related function is `_8443_get_command()`.
  - \*Feedback: display the value of feedback position counter.  
The related function is `_8443_get_position()`.
  - \*Pos Error: display the value of position error counter.  
The related function is `_8443_get_error_counter()`.
  - \*Target Pos: display the value of target position recorder.  
The related function is `_8443_get_target_pos()`.
2. **Set Position:** Set all position counters to specified value. The related functions are:
  - `_8443_set_Posision()`
  - `_8443_set_command()`
  - `_8443_reset_error_counter()`
  - `_8443_reset_target_pos()`
3. **Motion Status:** display the return value of `_8443_motion_done` function. The related function is `_8443_motion_done()`.
4. **INT Status:**
  - Event: Display the `event_int_status` in Hex value. The related function is `_8443_get_int_status()`.
  - Error: Display the `error_int_status` in Hex value. The related function is `_8443_get_int_status()`.
  - Count: Count the total number of interrupts.

Clear Button: Click this button will clear all INT status and counter to "0".

5. **Velocity:**

Absolute value of the velocity (unit: pps). The related function is `_8443_get_current_speed()`.

6. **Show Velocity Curve Button:** Clicking this button will open the screen showing velocity vs. time curve. In this curve, a new velocity data is added every 100 ms. To close it, click this button again. To clear data, click on the curve in the graph.

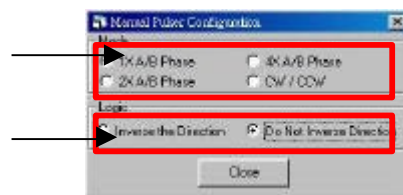


7. **Operation Mode:** Select operation mode.

- **Absolute Mode:** "Position1" and "position2" will be used as the absolute target position for an operation. The related function is `_8443_start_ta_move()` and `_8443_start_sa_move()`.
- **Relative Mode:** "Distance" will be used as relative displacement for an operation. The related function is `_8443_start_tr_move()` and `_8443_start_sr_move()`.
- **Cont. Move:** Velocity operation mode. The related function is `_8443_tv_move()`.
- **Manual Pulser Move:** Manual Pulser operation. Click this button will open the manual pulse configuration screen window to set input pulse mode and pulse logic.

Set the input pulse mode

Set the pulse logic



- **Home mode :**

Set the axis operation mode to the homing operation.

Home Move Configuration setting screen is displayed, and a home mode can be selected. The related function is `_8443_set_home_config()`.

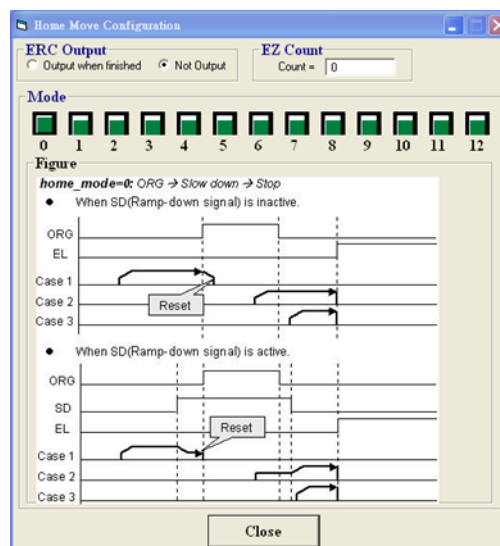
**ERC Output:** Select if the ERC signal will be sent when a homing operation is completed.

**EZ Count:** Set the EZ count number which is effective in a home return mode.

**Mode:** Select a home return mode. There are 13 modes available

**Figure:** The figure explains the actions of each homing mode.

**Close:** Click the button to close this screen.



8. **Position:**

Set the absolute position for "Absolute Mode". It is only effective when "Absolute Mode" is selected.

9. **Distance:**

Set the relative distance for "Relative Mode". It is only effective when "Relative Mode" is selected.

10. **Repeat Mode:**

When "ON" is selected, the motion will be repeated (reciprocating motion).

It is only effective when "Relative Mode" or "Absolute Mode" is selected.

In Absolute Mode: reciprocating motion between Position 1 and Position 2

In Relative Mode: reciprocating motion with the Distance value.

11. **Vel.Profile:**

Select the velocity profile. Either Trapezoidal or S-curve is available for "Absolute Mode", "Relative Mode", and "Cont. Move".

12. **Motion Parameter:**

Set the parameters for a single axis operation. These parameters are meaningless if "Manual Pulser Move" is selected since the velocity and moving distance is decided by the pulser input.

- **Start Velocity:** Set the start velocity of operation in the unit of pps. In "Absolute Mode" or "Relative Mode", only the value is effective (ie, -100.0 is the same as 100.0). In "Cont. Move", both the value and the sign are effective (-100.0 means 100.0 in the minus direction).
- **Maximum Velocity:** Set the maximum velocity of motion in unit of pps. In "Absolute Mode" or "Relative Mode", only the value is effective. (ie, -5000.0 is the same as 5000.0). In "Cont. Move", both the value and sign are effective (-5000.0 means 5000.0 in the minus direction).
- **Accel. Time:** Set the acceleration time (unit: s).
- **Decel. Time:** Set the deceleration time (unit: s).
- **SVacc:** Set the S-curve range during acceleration (unit: pps).
- **SVdec:** Set the S-curve range during deceleration (unit: pps).
- **Move delay:** This setting is effective only when repeat mode is set to "ON". The second operation is executed with a delay of the set time (seconds) after the first operation is completed.

- 

### 13. Speed Range:

Set the maximum speed of an operation. If “Not Fix” is selected, the “maximum speed” will automatically become the maximum speed range.

### 14. Servo On:

Set the SVON signal output status. The related function is `_8443_set_servo()`.

### 15. Play Key:

Left button: Click this button will cause PPCle-8443 start to output pulses according to the operation settings at the upper part of the screen.

In “Absolute Mode”, it causes the axis to move to position1.

In “Relative Mode”, it causes the axis to move in the positive direction.

In “Cont. Move”, it causes the axis to start moving according to the set velocity.

In “Manual Pulser Move”, it cause axis to be in the pulser move. The maximum speed can be set by “Maximum Velocity”

Right button: Click this button will cause PPCI8443 start to output pulses according to the operation settings at the upper part of the screen.

In “Absolute Mode”, it causes the axis to move to position 2.

In “Relative Mode”, it causes the axis to move in the negative direction.

In “Cont. Move”, it causes the axis to start moving according to the set velocity, but the other direction.

In “Manual Pulser Move”, it cause axis get into pulser move. The speed limit is the value set by “Maximum Velocity”

### 16. Change Position on-the-fly Button:

When this button is enabled, you can change the target position in the current motion. The new position must be defined in “Position 2”. The related function is `_8443_p_change()`.

### 17. Change Velocity on-the-fly Button:

When this button is enabled, you can change the velocity in the current motion. The new velocity must be defined in “Maximum Velocity”. The related function is `_8443_v_change()`.

### 18. Stop Button:

Click this button will cause PPCle-8443 to decelerate and stop. The deceleration time is defined in “Decel. Time”. The related function is `_8443_sd_stop()`.

### 19. I/O Status:

The status of motion I/Os, LED lighting indicates signal ON, and OFF indicates signal OFF. The related function is `_8443_get_io_status()`.

### 20. Buttons:

- **Next Axis:** Change the operating axis.
- **Save Config:** Save the current configuration in 8443.ini.
- **Config Pulse & INT:** Go to Pulse IO and Interrupt Configuration screen (see subsection 6.3.3.).
- **Config Interface I/O:** Go to Interface I/O Configuration screen (see subsection 6.3.2.).
- **Back:** Go back to Main screen.

## 7. Function Library

This chapter describes the supporting software for PPCle-8443. You can use these functions to develop application programs in C or Visual Basic or C++ language. If Delphi is used as the programming environment, it is necessary to transform the header file, **8443.h**, manually.

### 7.1. List of Functions

#### *Initialization (default)*

#### *Section 7.3*

Function Name	Description
<b>_8443_initial</b>	Software initialization
<b>_8443_close</b>	Software Close
<b>_8443_get_base_addr</b>	Get base address of PPCle-8443
<b>_8443_get_irq_channel</b>	Get PPCle-8443 board's IRQ number
<b>_8443_delay_time</b>	Delay execution of program for specified time (ms).
<b>_8443_config_from_file</b>	Configure PPCle-8443 board according to configuration file (ie. 8443.ini), which is created by PPCle8443 Utility.
<b>_8443_version_info</b>	Check the hardware and the software version
<b>_8443_enable_manual_id</b>	Enable the dip switch (SW1) on board to specify manual board ID

#### *Pulse Input/Output Configuration*

#### *Section 7.4*

Function Name	Description
<b>_8443_set_pls_outmode</b>	Set pulse command output mode
<b>_8443_set_pls_ipmode</b>	Set encoder input mode
<b>_8443_set_feedback_src</b>	Set counter input source

#### *Velocity mode operation*

#### *Section 7.5*

Function Name	Description
<b>_8443_tv_move</b>	Accelerate an axis to a constant velocity with trapezoidal profile
<b>_8443_sv_move</b>	Accelerate an axis to a constant velocity with S-curve profile
<b>_8443_v_change</b>	Change speed on the fly (Speed override)
<b>_8443_sd_stop</b>	Decelerate to stop
<b>_8443_emg_stop</b>	Immediately stop
<b>_8443_fix_speed_range</b>	Define the speed range
<b>_8443_unfix_speed_range</b>	Release the speed range constrain
<b>_8443_get_current_speed</b>	Get the current speed
<b>_8443_verify_speed</b>	Check the min/max acceleration time under the maximum speed



**Single Axis Positioning Operation****Section 7.6**

Function Name	Description
<b>_8443_start_tr_move</b>	Begin a relative trapezoidal profile move
<b>_8443_start_ta_move</b>	Begin an absolute trapezoidal profile move
<b>_8443_start_sr_move</b>	Begin a relative S-curve profile move
<b>_8443_start_sa_move</b>	Begin an absolute S-curve profile move
<b>_8443_set_move_ratio</b>	Set the ratio of command pulse and feedback pulse.
<b>_8443_p_change</b>	Change position on the fly (Position override)
<b>_8443_set_pcs_logic</b>	Set the logic of PCS (Position Change Signal)
<b>_8443_set_sd_pin</b>	Set SD/PCS terminal
<b>_8443_backlash_comp</b>	Set backlash corrective pulse for compensation
<b>_8443_suppress_vibration</b>	Set vibration suppressing timing
<b>_8443_set_idle_pulse</b>	Set suppress vibration idle pulse counts

**Linear Interpolation Operation****Section 7.7**

Function Name	Description
<b>_8443_start_tr_move_xy</b>	Begin a relative 2-axis linear interpolation for X & Y, with trapezoidal profile.
<b>_8443_start_ta_move_xy</b>	Begin an absolute 2-axis linear interpolation for X & Y, with trapezoidal. profile
<b>_8443_start_sr_move_xy</b>	Begin a relative 2-axis linear interpolation for X & Y, with S-curve profile.
<b>_8443_start_sa_move_xy</b>	Begin an absolute 2-axis linear interpolation for X & Y, with S-curve profile.
<b>_8443_start_tr_move_zu</b>	Begin a relative 2-axis linear interpolation for Z & U, with trapezoidal profile.
<b>_8443_start_ta_move_zu</b>	Begin an absolute 2-axis linear interpolation for Z & U, with trapezoidal profile.
<b>_8443_start_sr_move_zu</b>	Begin a relative 2-axis linear interpolation for Z & U, with S-curve profile.
<b>_8443_start_sa_move_zu</b>	Begin an absolute 2-axis linear interpolation for Z & U, with S-curve profile.
<b>_8443_start_tr_line2</b>	Begin a relative 2-axis linear interpolation for any 2 axes, with trapezoidalprofile.
<b>_8443_start_sr_line2</b>	Begin a relative 2-axis linear interpolation for any 2 axes, with S-curve profile.
<b>_8443_start_ta_line2</b>	Begin an absolute 2-axis linear interpolation for any 2 axes, with trapezoidal profile.
<b>_8443_start_sa_line2</b>	Begin an absolute 2-axis linear interpolation for any 2 axes, with S-curve profile.
<b>_8443_start_tr_line3</b>	Begin a relative 3-axis linear interpolation with trapezoidal.
<b>_8443_start_sr_line3</b>	Begin a relative 3-axis linear interpolation with S-curve profile.
<b>_8443_start_ta_line3</b>	Begin an absolute 3-axis linear interpolation with trapezoidal profile.
<b>_8443_start_sa_line3</b>	Begin an absolute 3-axis linear interpolation with S-curve profile.
<b>_8443_start_tr_line4</b>	Begin a relative 4-axis linear interpolation with trapezoidal profile.
<b>_8443_start_sr_line4</b>	Begin a relative 4-axis linear interpolation with S-curve profile.
<b>_8443_start_ta_line4</b>	Begin an absolute 4-axis linear interpolation with trapezoidal profile.
<b>_8443_start_sa_line4</b>	Begin an absolute 4-axis linear interpolation with S-curve profile.
<b>_8443_set_line_move_mode</b>	Set continuous line interpolation mode.
<b>_8443_set_axis_option</b>	Select the interpolation speed mode

**Circular Interpolation Operation****Section 7.8**

Function Name	Description
<b>_8443_start_r_arc_xy</b>	Begin a relative circular interpolation for X & Y.
<b>_8443_start_a_arc_xy</b>	Begin an absolute circular interpolation for X & Y.
<b>_8443_start_r_arc_zu</b>	Begin a relative circular interpolation for Z & U.
<b>_8443_start_a_arc_zu</b>	Begin an absolute circular interpolation for Z & U.
<b>_8443_start_r_arc2</b>	Begin a relative circular interpolation for any 2 of the 4 axes.
<b>_8443_start_a_arc2</b>	Begin an absolute circular interpolation for any 2 of the 4 axes.
<b>_8443_start_tr_arc_xy</b>	Begin a trapezoidal relative circular interpolation for X & Y.
<b>_8443_start_ta_arc_xy</b>	Begin a trapezoidal absolute circular interpolation for X & Y.
<b>_8443_start_sr_arc_xy</b>	Begin a s-curve relative circular interpolation for X & Y.
<b>_8443_start_sa_arc_xy</b>	Begin a s-curve absolute circular interpolation for X & Y.
<b>_8443_start_tr_arc_zu</b>	Begin a trapezoidal relative circular interpolation for Z & U.
<b>_8443_start_ta_arc_zu</b>	Begin a trapezoidal absolute circular interpolation for Z & U.
<b>_8443_start_sr_arc_zu</b>	Begin a s-curve relative circular interpolation for Z & U.
<b>_8443_start_sa_arc_zu</b>	Begin a s-curve absolute circular interpolation for Z & U.
<b>_8443_start_tr_arc2</b>	Begin a trapezoidal relative circular interpolation for any 2 of the 4 axes.
<b>_8443_start_ta_arc2</b>	Begin a trapezoidal absolute circular interpolation for any 2 of the 4 axes.
<b>_8443_start_sr_arc2</b>	Begin an s-curve relative circular interpolation for any 2 of the 4 axes.
<b>_8443_start_sa_arc2</b>	Begin an s-curve absolute circular interpolation for any 2 of the 4 axes.

**Helical Interpolation Operation****Section 7.9**

Function Name	Description
<b>_8443_set_tr_helical_xzy</b>	Begin a t-curve relative helical interpolation for X, Z and Y.
<b>_8443_set_ta_helical_xzy</b>	Begin a t-curve absolute helical interpolation for X, Z and Y.
<b>_8443_set_sr_helical_xzy</b>	Begin a s-curve relative helical interpolation for X, Z and Y.
<b>_8443_set_sa_helical_xzy</b>	Begin a s-curve absolute helical interpolation for X, Z and Y.
<b>_8443_set_tr_helical_xyz</b>	Begin a t-curve relative helical interpolation for X, Y and Z.
<b>_8443_set_ta_helical_xyz</b>	Begin a t-curve absolute helical interpolation for X, Y and Z.
<b>_8443_set_sr_helical_xyz</b>	Begin an s-curve relative helical interpolation for X, Y and Z.
<b>_8443_set_sa_helical_xyz</b>	Begin an s-curve absolute helical interpolation for X, Y and Z.

**Home Return Mode****Section 7.10**

Function Name	Description
<b>_8443_set_home_config</b>	Set the home / index logic configuration.
<b>_8443_home_move</b>	Begin a home return operation.
<b>8443_escape_home</b>	Begin escape home operation.
<b>_8443_home_search</b>	Begin Auto-Search Home Switch (Without ORG offset setting, the default ORG offset = 100).
<b>_8443_auto_home_search</b>	Begin Auto-Search Home Switch (With ORG offset).

**Manual Pulser Operation      Section 7.11**

Function Name	Description
<code>_8443_disable_pulser_input</code>	Disable the pulser input
<code>_8443_set_pulser_iptmode</code>	Set the pulser input mode.
<code>_8443_pulser_vmove</code>	Start the pulser v operation (velocity designation).
<code>_8443_pulser_pmove</code>	Start the pulser p operation (positioning).
<code>_8443_pulser_home_move</code>	Start the pulser home return operation.
<code>_8443_set_pulser_ratio</code>	Set the manual pulser ratio for actual output pulse rate.
<code>_8443_pulser_r_line2</code>	Pulser mode for 2-axis linear interpolation.
<code>_8443_pulser_r_arc2</code>	Pulser mode for 2-axis circular interpolation.

**Monitor axis operation status      Section 7.12**

Function Name	Description
<code>_8443_motion_done</code>	Return the operation status

**Motion Interface I/O      Section 7.13**

Function Name	Description
<code>_8443_set_alm</code>	Set the alarm logic and operating mode.
<code>_8443_set_el</code>	Set the EL operating mode.
<code>_8443_set_inp</code>	Set the INP logic and operating mode.
<code>_8443_set_erc</code>	Set the ERC logic and timing.
<code>_8443_set_servo</code>	Set the state of general-purpose output terminal.
<code>_8443_set_sd</code>	Set the SD logic and operating mode.

**Motion I/O Monitoring      Section 7.14**

Function Name	Description
<code>_8443_get_io_status</code>	Obtain all of the motion I/O status.

**Interrupt Control      Section 7.15**

Function Name	Description
<code>_8443_int_control</code>	Enable / Disable the INT service
<code>_8443_set_int_factor</code>	Set the INT factor
<code>_8443_int_enable</code>	Enable the event
<code>_8443_int_disable</code>	Disable the event
<code>_8443_get_int_status</code>	Get the INT Status
<code>_8443_link_interrupt</code>	Set the link to interrupt call back function
<code>_8443_set_axis_stop_int</code>	Enable the axis stop interrupt
<code>_8443_mask_axis_stop_int</code>	Mask the axis stop interrupt

**Position Control and Counters****Section 7.16**

Function Name	Description
<b>_8443_get_position</b>	Get the value of feedback position counter
<b>_8443_set_position</b>	Set the feedback position counter
<b>_8443_get_command</b>	Get the value of command position counter
<b>_8443_set_command</b>	Set the command position counter
<b>_8443_get_error_counter</b>	Get the value of position error counter
<b>_8443_reset_error_counter</b>	Reset the position error counter
<b>_8443_get_general_counter</b>	Get the value of general counter
<b>_8443_set_general_counter</b>	Set the general counter
<b>_8443_get_target_pos</b>	Get the value of target position recorder
<b>_8443_reset_target_pos</b>	Reset the target position recorder
<b>_8443_get_rest_command</b>	Get the remaining pulses until the end of an operation
<b>_8443_check_rdp</b>	Check the ramping down point data
<b>_8443_set_auto_rdp</b>	Enable the automatic setting ramping-down point

**Position Comparator and Latch****Section 7.17**

Function Name	Description
<b>_8443_set_ltc_logic</b>	Set the LTC logic
<b>_8443_get_latch_data</b>	Get the latched counter data
<b>_8443_set_soft_limit</b>	Set the software limit
<b>_8443_enable_soft_limit</b>	Enable the software limit function
<b>_8443_disable_soft_limit</b>	Disable the software limit function
<b>_8443_set_error_counter_check</b>	Set the out of step detection
<b>_8443_set_general_comparator</b>	Set the general-purpose comparator
<b>_8443_set_trigger_comparator</b>	Set the trigger comparator
<b>_8443_set_trigger_type</b>	Set the trigger output type
<b>_8443_check_compare_data</b>	Check the current comparator data
<b>_8443_check_compare_status</b>	Check the current comparator status
<b>_8443_set_auto_compare</b>	Set the comparing data source for auto loading
<b>_8443_build_compare_function</b>	Build the compare data via constant interval
<b>_8443_build_compare_table</b>	Build the compare data via compare table
<b>_8443_cmp_v_change</b>	Speed change by comparator
<b>_8443_set_latch_source</b>	Set the latch signal

**Continuous operation****Section 7.18**

Function Name	Description
<b>_8443_set_continuous_move</b>	Enable the continuous motion for absolute motion
<b>_8443_check_continuous_buffer</b>	Check if the pre-register for continuous motion is empty

**Multiple Axes Simultaneous Operation****Section 7.19**

Function Name	Description
<b>_8443_set_tr_move_all</b>	Relative trapezoidal multi-axis simultaneous operation setup.
<b>_8443_set_ta_move_all</b>	Absolute trapezoidal multi-axis simultaneous operation setup.
<b>_8443_set_sr_move_all</b>	Relative S-curve multi-axis simultaneous operation setup.
<b>_8443_set_sa_move_all</b>	Absolute S-curve multi-axis simultaneous operation setup.
<b>_8443_start_move_all</b>	Simultaneously begin the multi-axis motion
<b>_8443_stop_move_all</b>	Simultaneously stop the multi-axis motion
<b>_8443_set_sync_option</b>	Optional sync options
<b>_8443_set_sync_stop_mode</b>	Set the stop mode when CSTOP signal is ON

**Extended General-purpose Input/Output****Section 7.20**

Function Name	Description
<b>_8443_set_gpio_output</b>	Set digital outputs (whole port)
<b>_8443_get_gpio_output</b>	Get digital output status (whole port)
<b>_8443_get_gpio_input</b>	Get digital input status (whole port)
<b>_8443_set_gpio_output_CH</b>	Set digital output by channel (bit specified)
<b>_8443_get_gpio_output_CH</b>	Get digital output by channel (bit specified)
<b>_8443_get_gpio_input_CH</b>	Get digital input by channel (bit specified)

## 7.2. C/C++ Programming Library

This section gives the details of all the functions. The function prototypes and some common data types are decelerated in **PPCLe8443.H**. These data types are used by PPCLe-8443 library. We suggest you to use these data types in your application programs. The following table shows the data type names and the range.

Type	Description	Range
U8	8-bit ASCII character	0 ~ 255
I16	16-bit signed integer	- 32,768 ~ 32,767
U16	16-bit unsigned integer	0 ~ 65,535
I32	32-bit signed long integer	- 2,147,483,648 ~ 2,147,483,647
U32	32-bit unsigned long integer	0 ~ 4,294,967,295
F32	32-bit single-precision floating-point	- 3.402823E38 ~ 3.402823E38
F64	64-bit double-precision floating-point	- 1.797683134862315E309 ~ 1.797683134862315E308
Boolean	Boolean logic value	TRUE, FALSE

The functions of PPCLe-8443 software drivers use full names to represent the real meanings of the functions. The conventional rules for naming are:

In 'C' programming environment:

`_ {hardware_model} _ {action_name} .` e.g. `_8443_Initial()`.

In order to recognize the difference between C library and VB library, a capital "B" is put on the head of each function name

e.g. `B_8443_Initial()`.

## 7.3. Initialization

### @ Function Name

<code>_8443_initial</code>	- Software open and initialization process for PPCle-8443.
<code>_8443_close</code>	- Software release resources and close process of PPCle-8443.
<code>_8443_get_base_addr</code>	- Obtain the base address of PPCle-8443.
<code>_8443_get_irq_channel</code>	- Get the IRQ number for PPCle-8443 board.
<code>_8443_delay_time</code>	- Delay execution of program for specified time (unit: ms).
<code>_8443_config_from_file</code>	- Configure PPCle-8443 board according to the configuration file: "8443.ini".
<code>_8443_version_info</code>	- Check the hardware and the software version information.
<code>_8443_enable_manual_id</code>	- Enable the dip switch (SW1) on board to specify manual board ID.

### @Function Description

#### `_8443_initial:`

This function is used to initialize PPCle-8443 board. The all PPCle-8443 boards must be initialized by this function before calling other functions.

#### `_8443_close:`

This function is used to close PPCle-8443 and release the PPCle-8443 related resources, which should be called at the end of an application.

#### `_8443_get_base_addr:`

This function is used to get the base address for the PPCle-8443.

#### `_8443_get_irq_channel:`

This function is used to get IRQ number for the PPCle-8443.

#### `_8443_delay_time:`

This function is used to delay execution of program for specified time (unit: ms)

#### `_8443_config_from_file:`

This function is used to load the configuration of PPCle-8443 according to a specified file. By using PPCle8443 Utility, you can test and configure PPCle-8443 correctly. After pressing "save config" button, the "8443.ini" file in window directory is used to record the configurations. By specifying it in the parameter, the configuration will be automatically loaded.

When this function is executed, the all PPCle-8443 boards in the system will be configured as the following functions that were called according to the parameters recorded in **8443.ini**.

`_8443_set_pls_outmode`

`_8443_set_feedback_src`

`_8443_set_pls_iptmode`

`_8443_set_home_config`

`_8443_set_int_factor`

`_8443_set_el`

`_8443_set_ltc_logic`

`_8443_set_erc`

***\_8443\_set\_sd***

***\_8443\_set\_alm***

***\_8443\_set\_inp***

***\_8443\_set\_move\_ratio***

***\_8443\_version\_info:***

You can read back the hardware and software version information of the PPCle-8443.

***\_8443\_enable\_manual\_id:***

This function is used to enable the dip switch (SW1) on board to specify manual card ID. Please note that it would be used before the first calling ***\_8443\_Initial()***, otherwise it will be no effective.



**@ Syntax****C/C++ (Windows XP/7/8)**

```

I16 _8443_initial(I16 *existCards);
I16 _8443_close(void);
I16 _8443_get_base_addr(I16 cardNo, U16 *base_addr );
I16 _8443_get_irq_channel(I16 cardNo, U16 *irq_no );
I16 _8443_delay_time(I16 AxisNo, F64 MilliSec);
I16 _8443_config_from_file(char *file_name);
I16 _8443_version_info(I16 CardNo, U16 *HardwareInfo, U16 *SoftwareInfo, U16 *DriverInfo);
void _8443_enable_manual_id();

```

**VB.NET (Windows XP/7/8)**

```

B_8443_initial(ByRef existCards As Short) As Short
B_8443_close() As Short
B_8443_get_base_addr(ByVal cardNo As Short, ByRef base_addr As Short) As Short
B_8443_get_irq_channel(ByVal cardNo As Short, ByRef irq_no As Short) As Short
B_8443_delay_time(ByVal AxisNo As Short, ByVal MilliSec As Double) As Short
B_8443_config_from_file(ByVal filename As String) As Short
B_8443_version_info(ByVal CardNo As Short, ByRef HardwareInfo As Short, ByRef SoftwareInfo As Short, ByRef
DriverInfo As Short) As Short
B_8443_enable_manual_id()

```

**C# (Windows XP/7/8)**

```

Int16 _8443_initialx(UInt16 BaseAddress, UInt16 IRQNo);
Int16 _8443_close();
Int16 _8443_get_base_addr(Int16 cardNo, ref UInt16 base_addr);
Int16 _8443_get_irq_channel(Int16 cardNo, ref UInt16 irq_no);
Int16 _8443_delay_time(Int16 AxisNo, Double MilliSec);
Int16 _8443_config_from_file(string file_name);
Int16 _8443_version_info(Int16 CardNo, ref UInt16 HardwareInfo, ref UInt16 SoftwareInfo, ref UInt16 DriverInfo);
void _8443_enable_manual_id();

```

**@ Argument**

**\*existCards:** The number of existing PCIe-8443 boards

**cardNo:** The PCIe-8443 card index number (0 starts)

**\*irq\_no:** IRQ number of a specified PCIe-8443 board

**\*base\_addr:** Base address of a specified PCIe-8443 board

**\*file\_name:** A specified filename recording the configuration of PCIe-8443. This file must be created by PCIe8443 Utility.

**AxisNo:** Axis number designated to move or stop

**MilliSec:** Delay time (unit: ms)

**\*Hardwareinfo:** Hardware (Firmware) version read back

**\*Softwareinfo:** Software library version read back

**\*Driverinfo:** Device driver version read back

**@ Return Code**

ERR_NoError	:	0
ERR_NoCardFound	:	24
ERR_PCIBiosNotExist	:	8
ERR_ConigFileOpenError	:	39

## 7.4. Pulse Input/Output Configuration

### @ Function Name

<b>_8443_set_pls_outmode</b>	- Set the configuration for pulse command output.
<b>_8443_set_pls_iptmode</b>	- Set the configuration for feedback pulse input.
<b>_8443_set_feedback_src</b>	- Enable / Disable the external feedback pulse input

### @ Function Description

#### **\_8443\_set\_pls\_outmode:**

Configure the output modes of command pulse. There are 8 modes for command pulse output.

#### **\_8443\_set\_pls\_iptmode:**

Configure the input mode of external feedback pulse. There are four types for feedback pulse input. Note that this function is enabled only when Src parameter in **\_8443\_set\_feedback\_src()** function is enabled.

#### **\_8443\_set\_feedback\_src:**

If the external encoder feedback is available in the system, set the **Src** parameter in this function to enabled state. Then internal 32-bit up / down counter will count according to the configuration of **\_8443\_set\_pls\_iptmode()** function.

Otherwise, the counter will count the command pulse output. .

### @Syntax

#### **C/C++ (Windows XP/7/8)**

```
l16 _8443_set_pls_outmode(l16 AxisNo, l16 pls_outmode);
l16 _8443_set_pls_iptmode(l16 AxisNo, l16 pls_iptmode, l16 pls_logic);
l16 _8443_set_feedback_src(l16 AxisNo, l16 Src);
```

#### **VB.NET (Windows XP/7/8)**

```
B_8443_set_pls_outmode(ByVal AxisNo As Short, ByVal pls_outmode As Short) As Short
B_8443_set_pls_iptmode(ByVal AxisNo As Short, ByVal pls_iptmode As Short, ByVal pls_logic As Short) As Short
B_8443_set_feedback_src(ByVal AxisNo As Short, ByVal Src As Short) As Short
```

#### **C# (Windows XP/7/8)**

```
Int16 _8443_set_pls_outmode(Int16 AxisNo, Int16 pls_outmode);
Int16 _8443_set_pls_iptmode(Int16 AxisNo, Int16 pls_iptmode, Int16 pls_logic);
Int16 _8443_set_feedback_src(Int16 AxisNo, Int16 Src);
```

### @Argument

**AxisNo:** Axis number designated to configure pulse Input/Output (0 starts).

**pls\_outmode:** Setting of command pulse output mode

0: OUT/DIR	OUT: Falling edge, DIR+ is high level
1: OUT/DIR	OUT: OUT Rising edge, DIR+ is high level
2: OUT/DIR	OUT: OUT Falling edge, DIR+ is low level
3: OUT/DIR	OUT: OUT Rising edge, DIR+ is low level
4: CW /CCW	Falling edge
5: CW /CCW	Rising edge
6: AB phase	OUT is leading 90-degree phase to DIR
7: AB phase	OUT is lagging 90 degree phase to DIR

**pls\_iptmode:** setting of encoder feedback pulse input mode; EA, EB

0:	1X A/B
1:	2X A/B
2:	4X A/B
3:	CW/CCW Pulse input

**pls\_logic:** Logic of encoder feedback pulse

pls\_logic=0: Normal Low

pls\_logic=1: Normal High

**Src:** Feedback counter source

0: External Feedback pulse

1: Command pulse

**@Return Code**

ERR\_NoError : 0

## 7.5. Velocity Mode Operation

### @Function Name

<code>_8443_tv_move</code>	- Accelerate an axis to a constant velocity with trapezoidal profile
<code>_8443_sv_move</code>	- Accelerate an axis to a constant velocity with S-curve profile
<code>_8443_v_change</code>	- Change speed on the fly (speed override)
<code>_8443_sd_stop</code>	- Decelerate to stop
<code>_8443_emg_stop</code>	- Immediate stop
<code>_8443_fix_speed_range</code>	- Define the speed range
<code>_8443_unfix_speed_range</code>	- Release the speed range constrain
<code>_8443_get_current_speed</code>	- Obtain the current speed
<code>_8443_verify_speed</code>	- Obtain the minimum and maximum accel/decel time in a speed profile

### @Function Description

#### `_8443_tv_move:`

This function is to accelerate an axis to the specified constant velocity with trapezoidal profile. The axis will continue to run 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.

#### `_8443_sv_move:`

This function is to accelerate an axis to the specified constant velocity with S-curve profile. The axis will continue to run 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.

#### `_8443_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 `_8443_fix_speed_range`. `_8443_v_change` is also applicable in pre-set motion. Note: The velocity profile is decided by the original motion profile. When using in S-curve, please set the motion to be pure S-curve, which has no linear part. There are some limitations for this function: see subsection 5.6.1.

#### `_8443_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 operation (both trapezoidal and S-curve motion), manual operation, or home return function is performed.

Note: The velocity profile is decided by the original motion profile.

#### `_8443_emg_stop:`

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

#### `_8443_fix_speed_range:`

This function is used to define the speed range. It should be called before starting operation that contains a velocity change.

#### `_8443_unfix_speed_range:`

This function is used to release the speed range constrain.

#### `_8443_get_current_speed:`

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

#### `_8443_verify_speed:`

Obtain the minimum and maximum accel/decel time in a speed profile.

**@Syntax****C/C++ (Windows XP/7/8)**

```

I16 _8443_tv_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _8443_sv_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc, F64 SVacc);
I16 _8443_v_change(I16 AxisNo, F64 NewVel, F64 Tacc);
I16 _8443_sd_stop(I16 AxisNo, F64 Tdec);
I16 _8443_emg_stop(I16 AxisNo);
F64 _8443_fix_speed_range(I16 AxisNo, F64 MaxVel);
I16 _8443_unfix_speed_range(I16 AxisNo);
I16 _8443_get_current_speed(I16 AxisNo, F64 *speed);
F64 _8443_verify_speed(F64 StrVel, F64 MaxVel, F64 *minAccT, F64 *maxAccT, F64 MaxSpeed);

```

**VB.NET (Windows XP/7/8)**

```

B_8443_tv_move(ByVal AxisNo As Short, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Short
B_8443_sv_move(ByVal AxisNo As Short, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal SVacc As Double) As Short
B_8443_v_change(ByVal AxisNo As Short, ByVal NewVel As Double, ByVal Time As Double) As Short
B_8443_sd_stop(ByVal AxisNo As Short, ByVal Tdec As Double) As Short
B_8443_emg_stop(ByVal AxisNo As Short) As Short
B_8443_fix_speed_range(ByVal AxisNo As Short, ByVal MaxVel As Double) As Short
B_8443_unfix_speed_range(ByVal AxisNo As Short) As Short
B_8443_get_current_speed (ByVal AxisNo As Short, Speed As Double) As Short
B_8443_verify_speed(ByVal StrVel As Double, ByVal MaxVel As Double, ByRef minAccT As Double, ByRef maxAccT As Double, ByVal MaxSpeed As Double) As Double

```

**C# (Windows XP/7/8)**

```

Int16 _8443_tv_move(Int16 AxisNo, Double StrVel, Double MaxVel, Double Tacc);
Int16 _8443_sv_move(Int16 AxisNo, Double StrVel, Double MaxVel, Double Tacc, Double SVacc);
Int16 _8443_v_change(Int16 AxisNo, Double NewVel, Double Time);
Int16 _8443_sd_stop(Int16 AxisNo, Double Tdec);
Int16 _8443_emg_stop(Int16 AxisNo);
Int16 _8443_fix_speed_range(Int16 AxisNo, Double MaxVel);
Int16 _8443_unfix_speed_range(Int16 AxisNo);
Int16 _8443_get_current_speed(Int16 AxisNo, ref Double speed);
Double _8443_verify_speed(Double StrVel, Double MaxVel, ref Double minAccT, ref Double maxAccT, Double MaxSpeed);

```

**@Argument**

**AxisNo:** Axis number designated to move or stop (0 starts)

**StrVel:** Starting velocity (unit: pps)

**MaxVel:** Maximum velocity (unit: pps)

**Tacc:** Specified acceleration time (unit: sec)

**SVacc:** Specified velocity interval in which S-curve acceleration is performed.  
Note: SVacc = 0; S-curve without linear parts.

**NewVel:** New velocity (unit: pps)

**Time:**

**Tdec:** Specified deceleration time (unit: sec)

**\*Speed:** Variable to save the current speed (speed range: 0 ~ 6553500)

**minAccT:** Minimum acceleration time (sec)

**maxAccT:** Maximum acceleration time (sec)

**MaxSpeed:** The speed set by **\_8443\_verify\_speed** (pps)

**@Return Code**

ERR_NoError	:	0
ERR_SpeedError	:	11
ERR_SpeedChangeError	:	29
ERR_SlowDownPointError	:	16
ERR_AxisAlreadyStop	:	14

## 7.6. Single Axis Position Operation

### @Function Name

<code>_8443_start_tr_move</code>	Begin a relative trapezoidal profile operation.
<code>_8443_start_ta_move</code>	Begin an absolute trapezoidal profile operation.
<code>_8443_start_sr_move</code>	Begin a relative S-curve profile operation.
<code>_8443_start_sa_move</code>	Begin an absolute S-curve profile move operation.
<code>_8443_set_move_ratio</code>	Set the ratio of command pulse and feedback pulse.
<code>_8443_p_change</code>	Change position on the fly (target position override).
<code>_8443_set_pcs_logic</code>	Set the logic of PCS (position change signal) terminal.
<code>_8443_set_sd_pin</code>	Set the SD / PCS terminals.
<code>_8443_backlash_comp</code>	Set the backlash compensating pulse for a compensation operation.
<code>_8443_suppress_vibration</code>	Set the vibration suppress timing.
<code>_8443_set_idle_pulse</code>	Set the suppress vibration idle pulse counts.

### @Function Description

**Note:** The moving direction is determined by the sign of **Pos** or **Dist** parameter. If the moving distance is too short to reach a 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) are kept unchanged.

#### `_8443_start_tr_move:`

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

#### `_8443_start_ta_move:`

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

#### `_8443_start_sr_move:`

This function causes the axis to accelerate from 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 will not let the program wait for motion completion but immediately will return the control to the program.

#### `_8443_start_sa_move:`

This function causes the axis to accelerate from 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 will not let the program wait for motion completion but immediately will return control to the program.

#### `_8443_set_move_ratio:`

This function configures scale factors (ratio between command pulse and feedback pulse) at a 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**.

Please set **other than "0"**.

#### `_8443_p_change:`

This function is used to change a target position on the fly in motion. There are some limitations for this function. See subsection 5.6.2.

#### `_8443_set_pcs_logic:`

This function is used to set the logic of Position Change Signal (PCS). The PCS share the same terminal with SD signal. Only when the SD/PCS terminal was set to PCS by `_8443_set_sd_pin`, this `_8443_set_pcs_logic` function becomes effective.

**\_8443\_set\_sd\_pin:**

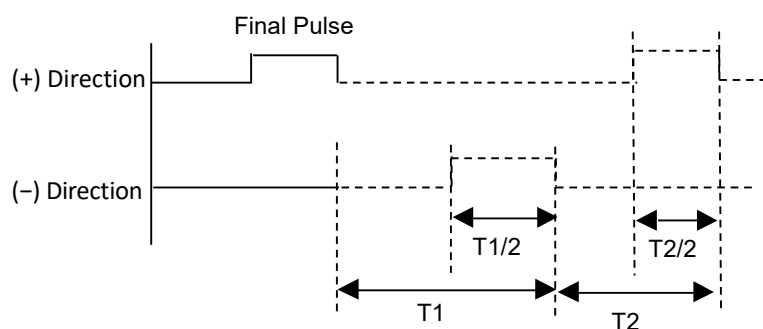
This function is used to set an operating mode of SD terminal. The SD terminal may be used either as Slow-Down signal input or as Position Change Signal (PCS) input. See subsection 5.3.1

**\_8443\_backlash\_comp:**

Whenever a direction change is occurred, PPCle-8443 will output backlash corrective pulses before sending commands. This function is used to set the compensation pulse numbers.

**\_8443\_suppress\_vibration:**

This function is used to suppress vibrations in a mechanical system by outputting a single pulse for negative direction and a single pulse for positive direction right after a completion of command operation.



T1: Reverse Time; T2: Forward Time

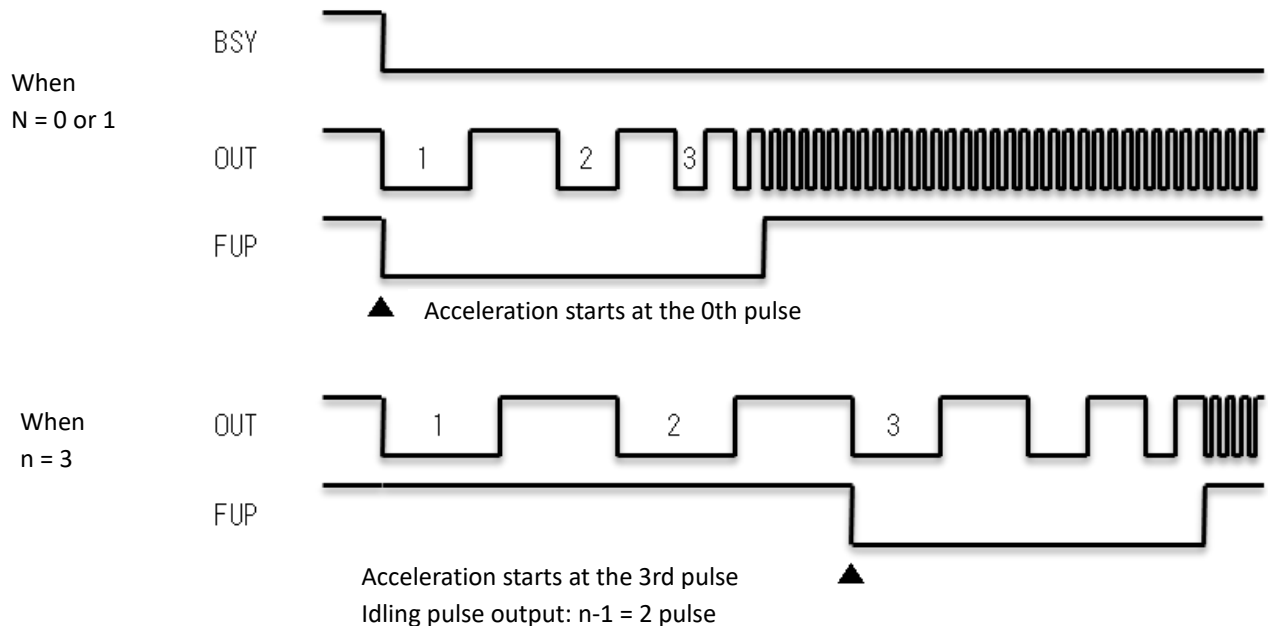
**\_8443\_set\_idle\_pulse:**

This idling pulse is to control the vibration when a machine is set up. Acceleration starts after several idling pulses are outputted at the start speed.

Attention:

Note: To use this function, set 2 - 7 as the set value. Please set 0 or 1 when not using.





**@Syntax****C/C++ (Windows/7/8)**

```

I16 _8443_start_tr_move(I16 AxisNo, F64 Dist, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8443_start_ta_move(I16 AxisNo, F64 Pos, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8443_start_sr_move(I16 AxisNo, F64 Dist, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _8443_start_sa_move(I16 AxisNo, F64 Pos, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _8443_set_move_ratio(I16 AxisNo, F64 move_ratio);
I16 _8443_p_change(I16 AxisNo, F64 NewPos);
I16 _8443_set_pcs_logic(I16 AxisNo, I16 pcs_logic);
I16 _8443_set_sd_pin(I16 AxisNo, I16 Type);
I16 _8443_backlash_comp(I16 AxisNo, I16 BCompPulse, I16 Mode);
I16 _8443_suppress_vibration(I16 AxisNo, U16 ReverseTime, U16 ForwardTime);
I16 _8443_set_idle_pulse(I16 AxisNo, I16 idl_pulse);

```

**VB.NET (Windows XP/7/8)**

```

B_8443_start_tr_move(ByVal AxisNo As Short, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double,
ByVal Tacc As Double, ByVal Tdec As Double) As Short
B_8443_start_ta_move(ByVal AxisNo As Short, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double,
ByVal Tacc As Double, ByVal Tdec As Double) As Short
B_8443_start_sr_move(ByVal AxisNo As Short, 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 Short
B_8443_start_sa_move(ByVal AxisNo As Short, 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 Short
B_8443_set_move_ratio(ByVal AxisNo As Short, ByVal move_ratio As Double) As Short
B_8443_p_change(ByVal AxisNo As Short, ByVal NewPos As Double) As Short
B_8443_set_pcs_logic(ByVal AxisNo As Short, ByVal pcs_logic As Short) As Short
B_8443_set_sd_pin(ByVal AxisNo As Short, ByVal Type As Short) As Short
B_8443_backlash_comp(ByVal AxisNo As Short, ByVal BCompPulse As Short, ByVal Mode As Short) As Short
B_8443_suppress_vibration(ByVal AxisNo As Short, ByVal ReverseTime As Short, ByVal ForwardTime As Short) As Short
B_8443_set_idle_pulse(ByVal AxisNo As Short, ByVal idl_pulse As Short) As Short

```

**C# (Windows XP/7/8)**

```

Int16 _8443_start_tr_move(Int16 AxisNo, Double Dist, Double StrVel, Double MaxVel, Double Tacc, Double Tdec);
Int16 _8443_start_ta_move(Int16 AxisNo, Double Pos, Double StrVel, Double MaxVel, Double Tacc, Double Tdec);
Int16 _8443_start_sr_move(Int16 AxisNo, Double Dist, Double StrVel, Double MaxVel, Double Tacc, Double Tdec,
Double SVacc, Double SVdec);
Int16 _8443_start_sa_move(Int16 AxisNo, Double Pos, Double StrVel, Double MaxVel, Double Tacc, Double Tdec,
Double SVacc, Double SVdec);
Int16 _8443_set_move_ratio(Int16 AxisNo, Double move_ratio);
Int16 _8443_p_change(Int16 AxisNo, Double NewPos);
Int16 _8443_set_pcs_logic(Int16 AxisNo, Int16 pcs_logic);
Int16 _8443_set_sd_pin(Int16 AxisNo, Int16 Type);
Int16 _8443_backlash_comp(Int16 AxisNo, Int16 BCompPulse, Int16 Mode);
Int16 _8443_suppress_vibration(Int16 AxisNo, UInt16 ReverseTime, UInt16 ForwardTime);
Int16 _8443_set_idle_pulse(Int16 AxisNo, Int16 idl_pulse);

```

**@Argument**

**AxisNo:** Axis number designated to move or change position (0 starts)

**Dist:** Specified relative distance to move

**Pos:** Specified absolute position to move

**StrVel:** Starting velocity of a velocity profile (pps)

**MaxVel:** Maximum velocity of a velocity profile (pps)

**Tacc:** Specified acceleration time (s)

**Tdec:** Specified acceleration time (s)

**SVacc:** Specified velocity interval when S-curve acceleration is performed.

Note: SVacc = 0; S-curve without linear part

**SVdec:** Specified velocity interval when S-curve deceleration is performed.

Note: SVacc = 0; S-curve without linear part

**move\_ratio:** Ratio of (feedback resolution): (command resolution)

Please set *other than "0"*.

**NewPos:** Specified new absolute position per the position on the fly (position override).

**pcs\_logic:** Specify the PCS logic setting:

0: Active Low

1: Active High

**Type:** Define the SD/PCS terminal usage:

0: SD

1: PCS

**BcompPulse:** Specified the number of corrective pulse (backlash compensation)

**Mode:** Backlash compensation setting

0: OFF

1: Backlash compensation enabled

2: Slip correction

**ReverseTime:** Specified Reverse Time

**ForwardTime:** Specified Forward Time

**idl\_pulse:** Idl\_pulse (Idling pulse) = 0 ~ 7 (Setting range)

#### @Return Code

ERR\_NoError : 0

ERR\_SpeedError : 11

ERR\_PChangeSlowDownPointError : 28

ERR\_MoveRatioError : 12

## 7.7. Linear Interpolation Operation

### @Function Name

<code>_8443_start_tr_move_xy</code>	- Relative 2-axis linear interpolation for X & Y, with trapezoidal profile
<code>_8443_start_ta_move_xy</code>	- Absolute 2-axis linear interpolation for X & Y, with trapezoidal profile
<code>_8443_start_sr_move_xy</code>	- Relative 2-axis linear interpolation for X & Y, with S-curve profile
<code>_8443_start_sa_move_xy</code>	- Absolute 2-axis linear interpolation for X & Y, with S-curve profile
<code>_8443_start_tr_move_zu</code>	- Relative 2-axis linear interpolation for Z & U, with trapezoidal profile
<code>_8443_start_ta_move_zu</code>	- Absolute 2-axis linear interpolation for Z & U, with trapezoidal profile
<code>_8443_start_sr_move_zu</code>	- Relative 2-axis linear interpolation for Z & U, with S-curve profile
<code>_8443_start_sa_move_zu</code>	- Absolute 2-axis linear interpolation for Z & U, with S-curve profile
<code>_8443_start_tr_line2</code>	- Relative 2-axis linear interpolation for any 2 axes, with trapezoidal profile
<code>_8443_start_sr_line2</code>	- Relative 2-axis linear interpolation for any 2 axes, with S-curve profile
<code>_8443_start_ta_line2</code>	- Absolute 2-axis linear interpolation for any 2 axes, with trapezoidal profile
<code>_8443_start_sa_line2</code>	- Absolute 2-axis linear interpolation for any 2 axes, with S-curve profile
<code>_8443_start_tr_line3</code>	- Relative 3-axis linear interpolation with trapezoidal profile
<code>_8443_start_sr_line3</code>	- Relative 3-axis linear interpolation with S-curve profile
<code>_8443_start_ta_line3</code>	- Absolute 3-axis linear interpolation with trapezoidal profile
<code>_8443_start_sa_line3</code>	- Absolute 3-axis linear interpolation with S-curve profile
<code>_8443_start_tr_line4</code>	- Relative 3-axis linear interpolation with trapezoidal profile
<code>_8443_start_sr_line4</code>	- Relative 4-axis linear interpolation with S-curve profile
<code>_8443_start_ta_line4</code>	- Absolute 4-axis linear interpolation with trapezoidal profile
<code>_8443_start_sa_line4</code>	- Absolute 4-axis linear interpolation with S-curve profile,
<code>_8443_set_line_move_mode</code>	- Set continuous line interpolation mode
<code>_8443_set_axis_option</code>	- Choose the interpolation speed mode

### @Function Description

Function	No. of axis	Speed profile	Relative position/ Absolute position	Target axis
<code>_8443_start_tr_move_xy</code>	2	T	R	Axis 0 and 1
<code>_8443_start_ta_move_xy</code>	2	T	A	Axis 0 and 1
<code>_8443_start_sr_move_xy</code>	2	S	R	Axis 0 and 1
<code>_8443_start_sa_move_xy</code>	2	S	A	Axis 0 and 1
<code>_8443_start_tr_move_zu</code>	2	T	R	Axis 2 and 3
<code>_8443_start_ta_move_zu</code>	2	T	A	Axis 2 and 3
<code>_8443_start_sr_move_zu</code>	2	S	R	Axis 2 and 3
<code>_8443_start_sa_move_zu</code>	2	S	A	Axis 2 and 3
<code>_8443_start_tr_line2</code>	2	T	R	Any two axes
<code>_8443_start_ta_line2</code>	2	T	A	Any two axes
<code>_8443_start_sr_line2</code>	2	S	R	Any two axes
<code>_8443_start_sa_line2</code>	2	S	A	Any two axes

<b>_8443_start_tr_line3</b>	3	T	R	Any three axes
<b>_8443_start_ta_line3</b>	3	T	A	Any three axes
<b>_8443_start_sr_line3</b>	3	S	R	Any three axes
<b>_8443_start_sa_line3</b>	3	S	A	Any three axes
<b>_8443_start_tr_line4</b>	4	T	R	Four axes
<b>_8443_start_ta_line4</b>	4	T	A	Four axes
<b>_8443_start_sr_line4</b>	4	S	R	Four axes
<b>_8443_start_sa_line4</b>	4	S	A	Four axes

T: Linear accel/ decel operation

S: S-curve accel / decel operation

R: Relative position specification mode

A: Absolute position specification mode

**\_8443\_set\_line\_move\_mode():**

This is the linear interpolation mode in 2 to 4 axis linear interpolation. If this mode is set to continuous ("1"), it will not stop until the continuous interpolation mode is entered and the stop function is written. When it is "0", it becomes positioning interpolation mode and stops when reaching the commanded target position / movement amount.

**@Syntax****C/C++(DOS, (Windows XP/7/8))**

```

I16 _8443_start_tr_move_xy(I16 CardNo, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8443_start_ta_move_xy(I16 CardNo, F64 PosX, F64 PosY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8443_start_sr_move_xy(I16 CardNo, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc,
F64 SVdec);
I16 _8443_start_sa_move_xy(I16 CardNo, F64 PosX, F64 PosY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc,
F64 SVdec);
I16 _8443_start_tr_move_zu(I16 CardNo, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8443_start_ta_move_zu(I16 CardNo, F64 PosX, F64 PosY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8443_start_sr_move_zu(I16 CardNo, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc,
F64 SVdec); I16 _8443_start_sa_move_zu(I16 CardNo, F64 PosX, F64 PosY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec,
F64 SVacc, F64 SVdec);
I16 _8443_start_tr_line2(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8443_start_ta_line2(I16 CardNo, I16 *AxisArray, F64 PosX, F64 PosY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8443_start_sr_line2(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec,
F64 SVacc, F64 SVdec);
I16 _8443_start_sa_line2(I16 CardNo, I16 *AxisArray, F64 PosX, F64 PosY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec,
F64 SVacc, F64 SVdec);
I16 _8443_start_tr_line3(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY, F64 DistZ, F64 StrVel, F64 MaxVel, F64 Tacc,
F64 Tdec);
I16 _8443_start_ta_line3(I16 CardNo, I16 *AxisArray, F64 PosX, F64 PosY, F64 PosZ, F64 StrVel, F64 MaxVel, F64 Tacc,
F64 Tdec);
I16 _8443_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 _8443_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 _8443_start_tr_line4(I16 CardNo, F64 DistX, F64 DistY, F64 DistZ, F64 DistU, F64 StrVel, F64 MaxVel, F64 Tacc, F64
Tdec);
I16 _8443_start_ta_line4(I16 CardNo, F64 PosX, F64 PosY, F64 PosZ, F64 PosU, F64 StrVel, F64 MaxVel, F64 Tacc, F64
Tdec);
I16 _8443_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 _8443_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 _8443_set_line_move_mode(I16 AxisNo, I16 Mode);
I16 _8443_set_axis_option(I16 AxisNo, I16 option);

```

**VB.NET (Windows XP/7/8)**



B\_8443\_set\_axis\_option(ByVal AxisNo As Short, ByVal OptionItem As Short) As Short

#### C# (Windows XP/7/8)

```

Int16 _8443_start_tr_move_xy(Int16 CardNo, Double DistX, Double DistY, Double StrVel, Double MaxVel, Double
Tacc, Double Tdec);
Int16 _8443_start_ta_move_xy(Int16 CardNo, Double PosX, Double PosY, Double StrVel, Double MaxVel, Double Tacc,
Double Tdec);
Int16 _8443_start_sr_move_xy(Int16 CardNo, Double DistX, Double DistY, Double StrVel, Double MaxVel, Double
Tacc, Double Tdec, Double SVacc, Double SVdec);
Int16 _8443_start_sa_move_xy(Int16 CardNo, Double PosX, Double PosY, Double StrVel, Double MaxVel, Double
Tacc, Double Tdec, Double SVacc, Double SVdec);
Int16 _8443_start_tr_move_zu(Int16 CardNo, Double DistX, Double DistY, Double StrVel, Double MaxVel, Double
Tacc, Double Tdec);
Int16 _8443_start_ta_move_zu(Int16 CardNo, Double PosX, Double PosY, Double StrVel, Double MaxVel, Double
Tacc, Double Tdec);
Int16 _8443_start_sr_move_zu(Int16 CardNo, Double DistX, Double DistY, Double StrVel, Double MaxVel, Double
Tacc, Double Tdec, Double SVacc, Double SVdec);
Int16 _8443_start_sa_move_zu(Int16 CardNo, Double PosX, Double PosY, Double StrVel, Double MaxVel, Double
Tacc, Double Tdec, Double SVacc, Double SVdec);
Int16 _8443_start_tr_line2(Int16 CardNo, ref Int16 AxisArray, Double DistX, Double DistY, Double StrVel, Double
MaxVel, Double Tacc, Double Tdec);
Int16 _8443_start_sr_line2(Int16 CardNo, ref Int16 AxisArray, Double DistX, Double DistY, Double StrVel, Double
MaxVel, Double Tacc, Double Tdec, Double SVacc, Double SVdec);
Int16 _8443_start_ta_line2(Int16 CardNo, ref Int16 AxisArray, Double PosX, Double PosY, Double StrVel, Double
MaxVel, Double Tacc, Double Tdec);
Int16 _8443_start_sa_line2(Int16 CardNo, ref Int16 AxisArray, Double PosX, Double PosY, Double StrVel, Double
MaxVel, Double Tacc, Double Tdec, Double SVacc, Double SVdec);
Int16 _8443_start_tr_line3(Int16 CardNo, ref Int16 AxisArray, Double DistX, Double DistY, Double DistZ, Double StrVel,
Double MaxVel, Double Tacc, Double Tdec);
Int16 _8443_start_sr_line3(Int16 CardNo, ref Int16 AxisArray, Double DistX, Double DistY, Double DistZ, Double StrVel,
Double MaxVel, Double Tacc, Double Tdec, Double SVacc, Double SVdec);
Int16 _8443_start_ta_line3(Int16 CardNo, ref Int16 AxisArray, Double PosX, Double PosY, Double PosZ, Double StrVel,
Double MaxVel, Double Tacc, Double Tdec);
Int16 _8443_start_sa_line3(Int16 CardNo, ref Int16 AxisArray, Double PosX, Double PosY, Double PosZ, Double StrVel,
Double MaxVel, Double Tacc, Double Tdec, Double SVacc, Double SVdec);
Int16 _8443_start_tr_line4(Int16 CardNo, Double DistX, Double DistY, Double DistZ, Double DistU, Double StrVel,
Double MaxVel, Double Tacc, Double Tdec);
Int16 _8443_start_sr_line4(Int16 CardNo, Double DistX, Double DistY, Double DistZ, Double DistU, Double StrVel,
Double MaxVel, Double Tacc, Double Tdec, Double SVacc, Double SVdec);
Int16 _8443_start_ta_line4(Int16 CardNo, Double PosX, Double PosY, Double PosZ, Double PosU, Double StrVel,
Double MaxVel, Double Tacc, Double Tdec);
Int16 _8443_start_sa_line4(Int16 CardNo, Double PosX, Double PosY, Double PosZ, Double PosU, Double StrVel,
Double MaxVel, Double Tacc, Double Tdec, Double SVacc, Double SVdec);
Int16 _8443_set_line_move_mode(Int16 AxisNo, Int16 Mode);
Int16 _8443_set_axis_option(Int16 AxisNo, Int16 option);

```

#### @Argument

**CardNo:** Card number designated to perform linear interpolation (0 starts)

**DistX:** Specified relative distance of axis 0 operation

**DistY:** Specified relative distance of axis 1 operation

**DistZ:** Specified relative distance of axis 2 operation

**DistU:** Specified relative distance of axis 3 operation)

**PosX:** Specified absolute position of axis 0 operation)

**PosY:** Specified absolute position of axis 1 operation)

**PosZ:** Specified absolute position of axis 2 operation)

**PosU:** Specified absolute position of axis 3 operation)

**StrVel:** Starting velocity in a velocity profile (pps)

**MaxVel:** Maximum velocity in a velocity profile (pps)

**Tacc:** Specified acceleration time (sec)

**Tdec:** Specified deceleration time (sec)

**SVacc:** Specified velocity interval in which S-curve acceleration is performed.

Note: SVacc = 0, for pure S-curve

**SVdec:** Specified velocity interval when S-curve deceleration is performed.

Note: SVacc = 0; S-curve without linear part for pure S-curve

**AxisArray:** Array of the axis numbers to perform interpolation.

Example: Int AxisArray[2] = {0, 2}; // axis 0 and 2

Int AxisArray[3] = {0, 1, 3}; // axis 0, 1 and 3

Note: Be sure to store axis numbers in the array in ascending order.

**Mode:** Interpolation mode

0: Positioning line interpolation mode

1: Continuous line interpolation mode

**Option:**

0: Default line move mode

1: Composite speed constant mode

#### @Return Code

ERR\_NoError : 0

ERR\_SpeedError : 11

ERR\_AxisArrayError : 15



## 7.8. Circular Interpolation Operation

### @Function Name

<code>_8443_start_r_arc_xy</code>	- Begin a relative circular interpolation for X & Y
<code>_8443_start_a_arc_xy</code>	- Begin an absolute circular interpolation for X & Y
<code>_8443_start_r_arc_zu</code>	- Begin a relative circular interpolation for Z & U
<code>_8443_start_a_arc_zu</code>	- Begin an absolute circular interpolation for Z & U
<code>_8443_start_r_arc2</code>	- Begin a relative circular interpolation for any 2 axes
<code>_8443_start_a_arc2</code>	- Begin an absolute circular interpolation for any 2 axes
<code>_8443_start_tr_arc_xy</code>	- Begin a Trapezoidal relative circular interpolation for X & Y
<code>_8443_start_ta_arc_xy</code>	- Begin a Trapezoidal absolute circular interpolation for X & Y
<code>_8443_start_sr_arc_xy</code>	- Begin an S-curve relative circular interpolation for X & Y
<code>_8443_start_sa_arc_xy</code>	- Begin an S-curve absolute circular interpolation for X & Y
<code>_8443_start_tr_arc_zu</code>	- Begin a Trapezoidal relative circular interpolation for Z & U
<code>_8443_start_ta_arc_zu</code>	- Begin a Trapezoidal absolute circular interpolation for Z & U
<code>_8443_start_sr_arc_zu</code>	- Begin an S-curve relative circular interpolation for Z & U
<code>_8443_start_sa_arc_zu</code>	- Begin an S-curve absolute circular interpolation for Z & U
<code>_8443_start_tr_arc2</code>	- Begin a Trapezoidal relative circular interpolation for any 2 axes
<code>_8443_start_ta_arc2</code>	- Begin a Trapezoidal absolute circular interpolation for any 2 axes
<code>_8443_start_sr_arc2</code>	- Begin an S-curve relative circular interpolation for any 2 axes
<code>_8443_start_sa_arc2</code>	- Begin an S-curve absolute circular interpolation for any 2 axes

### @Function Description

Function	No. of axis	Speed profile	Relative position/ Absolute position
<code>_8443_start_r_arc_xy</code>	R	Flat	Axis 0 and 1
<code>_8443_start_a_arc_xy</code>	A	Flat	Axis 0 and 1
<code>_8443_start_r_arc_zu</code>	R	Flat	Axis 2 and 3
<code>_8443_start_a_arc_zu</code>	A	Flat	Axis 2 and 3
<code>_8443_start_r_arc2</code>	R	Flat	Any two axes
<code>_8443_start_a_arc2</code>	A	Flat	Any two axes
<code>_8443_start_tr_arc_xy</code>	R	Trapezoidal	Axis 0 and 1
<code>_8443_start_ta_arc_xy</code>	A	Trapezoidal	Axis 0 and 1
<code>_8443_start_sr_arc_xy</code>	R	S-curve	Axis 0 and 1
<code>_8443_start_sa_arc_xy</code>	A	S-curve	Axis 0 and 1

_8443_start_tr_arc_zu	R	Trapezoidal	Axis 2 and 3
_8443_start_ta_arc_zu	A	Trapezoidal	Axis 2 and 3
_8443_start_sr_arc_zu	R	S-curve	Axis 2 and 3
_8443_start_sa_arc_zu	A	S-curve	Axis 2 and 3
_8443_start_tr_arc2	R	Trapezoidal	Any two axes
_8443_start_ta_arc2	A	Trapezoidal	Any two axes
_8443_start_sr_arc2	R	S-curve	Any two axes
_8443_start_sa_arc2	A	S-curve	Any two axes

T: Linear accel/ decel operation

S: S-curve accel / decel operation

R: Relative position specification mode

A: Absolute position specification mode

**@Syntax****C/C++ (DOS. Windows XP/7/8)**

```

I16 _8443_start_r_arc_xy(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
I16 _8443_start_a_arc_xy(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 MaxVel);
I16 _8443_start_r_arc_zu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
I16 _8443_start_a_arc_zu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 MaxVel);
I16 _8443_start_r_arc2(I16 CardNo, I16 *AxisArray, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64
MaxVel);
I16 _8443_start_a_arc2(I16 CardNo, I16 *AxisArray, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 MaxVel);
I16 _8443_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 _8443_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 _8443_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 _8443_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 _8443_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 _8443_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 _8443_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 _8443_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 _8443_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 _8443_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 _8443_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 _8443_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);

```

**VB.NET (Windows XP/7/8)**

```

B_8443_start_r_arc_xy(ByVal CardNo As Short, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As
Double, ByVal OffsetEy As Double, ByVal DIR As Short, ByVal MaxVel As Double) As Short
B_8443_start_a_arc_xy(ByVal CardNo As Short, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey
As Double, ByVal DIR As Short, ByVal MaxVel As Double) As Short
B_8443_start_r_arc_zu(ByVal CardNo As Short, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As
Double, ByVal OffsetEy As Double, ByVal DIR As Short, ByVal MaxVel As Double) As Short
B_8443_start_a_arc_zu(ByVal CardNo As Short, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey
As Double, ByVal DIR As Short, ByVal MaxVel As Double) As Short

B_8443_start_r_arc2(ByVal CardNo As Short, ByVal AxisArray() As Short, ByVal OffsetCx As Double, ByVal OffsetCy As
Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Short, ByVal MaxVel As Double) As Short
B_8443_start_a_arc2(ByVal CardNo As Short, ByVal AxisArray() As Short, ByVal Cx As Double, ByVal Cy As Double, ByVal
Ex As Double, ByVal Ey As Double, ByVal DIR As Short, ByVal MaxVel As Double) As Short

B_8443_start_tr_arc_xy(ByVal CardNo As Short, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As
Double, ByVal OffsetEy As Double, ByVal DIR As Short, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As
Double) As Short
B_8443_start_ta_arc_xy(ByVal CardNo As Short, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey
As Double, ByVal DIR As Short, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Short

```

B\_8443\_start\_sa\_arc\_zu(ByVal CardNo As Short, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Short, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal SVacc As Double) As Short

B\_8443\_start\_sa\_arc2(ByVal CardNo As Short, ByVal AxisArray() As Short, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Short, 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 Short

```

Int16 _8443_start_r_arc_xy(Int16 CardNo, Double OffsetCx, Double OffsetCy, Double OffsetEx, Double OffsetEy, Int16 DIR, Double MaxVel);
Int16 _8443_start_a_arc_xy(Int16 CardNo, Double Cx, Double Cy, Double Ex, Double Ey, Int16 DIR, Double MaxVel);
Int16 _8443_start_r_arc_zu(Int16 CardNo, Double OffsetCx, Double OffsetCy, Double OffsetEx, Double OffsetEy, Int16 DIR, Double MaxVel);
Int16 _8443_start_a_arc_zu(Int16 CardNo, Double Cx, Double Cy, Double Ex, Double Ey, Int16 DIR, Double MaxVel);
Int16 _8443_start_r_arc2(Int16 CardNo, ref Int16 AxisArray, Double OffsetCx, Double OffsetCy, Double OffsetEx, Double OffsetEy, Int16 DIR, Double MaxVel);
Int16 _8443_start_a_arc2(Int16 CardNo, ref Int16 AxisArray, Double Cx, Double Cy, Double Ex, Double Ey, Int16 DIR, Double MaxVel);
Int16 _8443_start_tr_arc_xy(Int16 CardNo, Double OffsetCx, Double OffsetCy, Double OffsetEx, Double OffsetEy, Int16 DIR, Double StrVel, Double MaxVel, Double Tacc, Double Tdec);
Int16 _8443_start_ta_arc_xy(Int16 CardNo, Double Cx, Double Cy, Double Ex, Double Ey, Int16 DIR, Double StrVel, Double MaxVel, Double Tacc, Double Tdec);
Int16 _8443_start_sr_arc_xy(Int16 CardNo, Double OffsetCx, Double OffsetCy, Double OffsetEx, Double OffsetEy, Int16 DIR, Double StrVel, Double MaxVel, Double Tacc, Double Tdec, Double SVacc, Double SVdec);
Int16 _8443_start_sa_arc_xy(Int16 CardNo, Double Cx, Double Cy, Double Ex, Double Ey, Int16 DIR, Double StrVel, Double MaxVel, Double Tacc, Double Tdec, Double SVacc, Double SVdec);
Int16 _8443_start_tr_arc_zu(Int16 CardNo, Double OffsetCx, Double OffsetCy, Double OffsetEx, Double OffsetEy, Int16 DIR, Double StrVel, Double MaxVel, Double Tacc, Double Tdec);
Int16 _8443_start_ta_arc_zu(Int16 CardNo, Double Cx, Double Cy, Double Ex, Double Ey, Int16 DIR, Double StrVel, Double MaxVel, Double Tacc, Double Tdec);
Int16 _8443_start_sr_arc_zu(Int16 CardNo, Double OffsetCx, Double OffsetCy, Double OffsetEx, Double OffsetEy, Int16 DIR, Double StrVel, Double MaxVel, Double Tacc, Double Tdec, Double SVacc, Double SVdec);
Int16 _8443_start_sa_arc_zu(Int16 CardNo, Double Cx, Double Cy, Double Ex, Double Ey, Int16 DIR, Double StrVel, Double MaxVel, Double Tacc, Double Tdec, Double SVacc, Double SVdec);

```

```

Int16 _8443_start_tr_arc2(Int16 CardNo,ref Int16 AxisArray, Double OffsetCx, Double OffsetCy, Double OffsetEx,
Double OffsetEy, Int16 DIR, Double StrVel,Double MaxVel, Double Tacc,Double Tdec);
Int16 _8443_start_ta_arc2(Int16 CardNo,ref Int16 AxisArray, Double Cx, Double Cy, Double Ex, Double Ey, Int16
DIR, Double StrVel, Double MaxVel, Double Tacc,Double Tdec);
Int16 _8443_start_sr_arc2(Int16 CardNo,ref Int16 AxisArray, Double OffsetCx, Double OffsetCy, Double OffsetEx,
Double OffsetEy, Int16 DIR, Double StrVel,Double MaxVel, Double Tacc,Double Tdec,Double SVacc,Double SVdec);
Int16 _8443_start_sa_arc2(Int16 CardNo,ref Int16 AxisArray, Double Cx, Double Cy, Double Ex, Double Ey, Int16
DIR, Double StrVel, Double MaxVel, Double Tacc,Double Tdec,Double SVacc,Double SVdec);

```

#### @Argument

**CardNo:** Board number designated to perform a 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 when S-curve deceleration is performed.  
Note: SVacc = 0; S-curve without linear parts

**AxisArray:** Array of axis number to perform an interpolation.  
Example: Int AxisArray[2] = {0,2}; // axis 0 and 2  
Int AxisArray[2] = {1,3}; // axis 1 and 3  
Note: Be sure to store axis numbers in the array in ascending order.

#### @Return Code

```

ERR_NoError          : 0
ERR_SpeedError       : 11
ERR_AxisArrayError   : 15

```

## 7.9. Helical Interpolation Operation

### @Function Name

<code>_8443_start_tr_helical_xzy</code>	- Begin a T-curve relative helical interpolation for X, Z and Y axis
<code>_8443_start_ta_helical_xzy</code>	- Begin a T-curve absolute helical interpolation for X, Z and Y axis
<code>_8443_start_sr_helical_xzy</code>	- Begins S-curve relative helical interpolation for X, Z and Y axis
<code>_8443_start_sa_helical_xzy</code>	- Begins S-curve absolute helical interpolation for X, Z and Y axis
<code>_8443_start_tr_helical_xyz</code>	- Begin a T-curve relative helical interpolation for X, Y and Z axis
<code>_8443_start_ta_helical_xyz</code>	- Begin a T-curve absolute helical interpolation for X, Y and Z axis
<code>_8443_start_sr_helical_xyz</code>	- Begins S-curve relative helical interpolation for X, Y and Z axis
<code>_8443_start_sa_helical_xyz</code>	- Begins S-curve absolute helical interpolation for X, Y and Z axis

### @Function Description

These functions perform helical interpolation operations with different profiles. Detail comparisons of these functions are described in the below table. These functions can be used for a circular interpolation with axis X and axis Z and to adjust the angle of a jig toward an arc tangent point with Y axis.

In this operation, U axis operation will be a “dummy motion” so it cannot be used for any other purpose.

The speed specified by this function is a speed of circular interpolation.

Function	No. of axis	Speed profile	Relative position/ Absolute position	Target axes
<code>_8443_start_tr_helical_xzy</code>	4	T	R	Axis 0, 2, 1
<code>_8443_start_ta_helical_xzy</code>	4	T	A	Axis 0, 2, 1
<code>_8443_start_sr_helical_xzy</code>	4	S	R	Axis 0, 2, 1
<code>_8443_start_sa_helical_xzy</code>	4	S	A	Axis 0, 2, 1
<code>_8443_start_tr_helical_xyz</code>	4	T	R	Axis 0, 1, 2
<code>_8443_start_ta_helical_xyz</code>	4	T	A	Axis 0, 1, 2
<code>_8443_start_sr_helical_xyz</code>	4	S	R	Axis 0, 1, 2
<code>_8443_start_sa_helical_xyz</code>	4	S	A	Axis 0, 1, 2

T: Linear accel/ decel operation

S: S-curve accel / decel operation

R: Relative position specification mode

A: Absolute position specification mode

### @Syntax

#### C/C++ (Windows XP/7/8)

```

I16 _8443_start_tr_helical_xzy(I16 CardNo, F64 OffsetCx, F64 OffsetCz, F64 OffsetEx, F64 OffsetEz, F64 PitchDist, I16
CW_CCW, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8443_start_ta_helical_xzy(I16 CardNo, F64 Cx, F64 Cz, F64 Ex, F64 Ez, F64 PitchPos, I16 CW_CCW, F64 StrVel, F64
MaxVel, F64 Tacc, F64 Tdec);
I16 _8443_start_sr_helical_xzy(I16 CardNo, F64 OffsetCx, F64 OffsetCz, F64 OffsetEx, F64 OffsetEz, F64 PitchDist, I16
CW_CCW, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _8443_start_sa_helical_xzy(I16 CardNo, F64 Cx, F64 Cz, F64 Ex, F64 Ez, F64 PitchPos, I16 CW_CCW, F64 StrVel, F64
MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _8443_start_tr_helical_xyz(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, F64 PitchDist, I16
CW_CCW, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8443_start_ta_helical_xyz(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, F64 PitchPos, I16 CW_CCW, F64 StrVel, F64
MaxVel, F64 Tacc, F64 Tdec);
I16 _8443_start_sr_helical_xyz(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, F64 PitchDist, I16
CW_CCW, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

```

I16 \_8443\_start\_sa\_helical\_xyz(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, F64 PitchPos, I16 CW\_CCW, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

#### VB.NET (Windows XP/7/8)

B\_8443\_start\_tr\_helical\_xyz(ByVal CardNo As Short, ByVal OffsetCx As Double, ByVal OffsetCz As Double, ByVal OffsetEx As Double, ByVal OffsetEz As Double, ByVal PitchDist As Double, ByVal CW\_CCW As Short, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Short

B\_8443\_start\_ta\_helical\_xyz(ByVal CardNo As Short, ByVal Cx As Double, ByVal Cz As Double, ByVal Ex As Double, ByVal Ez As Double, ByVal PitchPos As Double, ByVal CW\_CCW As Short, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Short

B\_8443\_start\_sr\_helical\_xyz(ByVal CardNo As Short, ByVal OffsetCx As Double, ByVal OffsetCz As Double, ByVal OffsetEx As Double, ByVal OffsetEz As Double, ByVal PitchDist As Double, ByVal CW\_CCW As Short, 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 Short

B\_8443\_start\_sa\_helical\_xyz(ByVal CardNo As Short, ByVal Cx As Double, ByVal Cz As Double, ByVal Ex As Double, ByVal Ez As Double, ByVal PitchPos As Double, ByVal CW\_CCW As Short, 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 Short

B\_8443\_start\_tr\_helical\_xyz(ByVal CardNo As Short, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal PitchDist As Double, ByVal CW\_CCW As Short, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Short

B\_8443\_start\_ta\_helical\_xyz(ByVal CardNo As Short, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal PitchPos As Double, ByVal CW\_CCW As Short, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Short

B\_8443\_start\_sr\_helical\_xyz(ByVal CardNo As Short, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal PitchDist As Double, ByVal CW\_CCW As Short, 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 Short

B\_8443\_start\_sa\_helical\_xyz(ByVal CardNo As Short, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal PitchPos As Double, ByVal CW\_CCW As Short, 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 Short

#### C# (Windows XP/7/8)

Int16 \_8443\_start\_tr\_helical\_xyz(Int16 CardNo, Double OffsetCx, Double OffsetCz, Double OffsetEx, Double OffsetEz, Double PitchDist, Int16 CW\_CCW, Double StrVel, Double MaxVel, Double Tacc, Double Tdec);

Int16 \_8443\_start\_ta\_helical\_xyz(Int16 CardNo, Double Cx, Double Cz, Double Ex, Double Ez, Double PitchDist, Int16 CW\_CCW, Double StrVel, Double MaxVel, Double Tacc, Double Tdec);

Int16 \_8443\_start\_sr\_helical\_xyz(Int16 CardNo, Double OffsetCx, Double OffsetCz, Double OffsetEx, Double OffsetEz, Double PitchDist, Int16 CW\_CCW, Double StrVel, Double MaxVel, Double Tacc, Double Tdec, Double SVacc, Double SVdec);

Int16 \_8443\_start\_sa\_helical\_xyz(Int16 CardNo, Double Cx, Double Cz, Double Ex, Double Ez, Double PitchDist, Int16 CW\_CCW, Double StrVel, Double MaxVel, Double Tacc, Double Tdec, Double SVacc, Double SVdec);

Int16 \_8443\_start\_tr\_helical\_xyz(Int16 CardNo, Double OffsetCx, Double OffsetCy, Double OffsetEx, Double OffsetEy, Double PitchDist, Int16 CW\_CCW, Double StrVel, Double MaxVel, Double Tacc, Double Tdec);

Int16 \_8443\_start\_ta\_helical\_xyz(Int16 CardNo, Double Cx, Double Cy, Double Ex, Double Ey, Double PitchDist, Int16 CW\_CCW, Double StrVel, Double MaxVel, Double Tacc, Double Tdec);

Int16 \_8443\_start\_sr\_helical\_xyz(Int16 CardNo, Double OffsetCx, Double OffsetCy, Double OffsetEx, Double OffsetEy, Double PitchDist, Int16 CW\_CCW, Double StrVel, Double MaxVel, Double Tacc, Double Tdec, Double SVacc, Double SVdec);

Int16 \_8443\_start\_sa\_helical\_xyz(Int16 CardNo, Double Cx, Double Cy, Double Ex, Double Ey, Double PitchDist, Int16 CW\_CCW, Double StrVel, Double MaxVel, Double Tacc, Double Tdec, Double SVacc, Double SVdec);

#### @Argument

**CardNo:** Board number designated to perform a helical interpolation

**AxisNo:** Axis number designated to move or stop.

**OffsetCx:** X-axis (first axis of target axes) offset to center

**OffsetCz:** Z-axis (second axis of target axes) offset to center

**OffsetEx:** X-axis (first axis of target axes) offset to end of arc

**OffsetEz:** Z-axis offset to end of arc

**PitchDist:** Y-axis specified relative distance to move

**Cx:** X-axis (first axis of target axes) absolute position of center of arc  
**Cz:** Z-axis (second axis of target axes) absolute position of center of arc  
**Ex:** X-axis (first axis of target axes) absolute position of end of arc  
**Ez:** Z-axis (second axis of target axes) absolute position of end of arc  
**PitchPos:** Y-axis specified absolute position to move  
**CW\_CCW:** Specified direction of arc. 0: CW, 1: CCW  
**StrVel:** Starting velocity of a velocity profile in units of pulse per second.  
**MaxVel:** Maximum velocity in units of pulse per second.  
**Tacc:** Specified acceleration time in units of seconds.  
**Tdec:** Specified deceleration time in units of seconds.  
**SVacc:** Specified velocity interval in which S-curve acceleration is performed.  
    Note: SVacc = 0, S-curve without linear parts.  
**SVdec:** specified velocity interval when S-curve deceleration is performed.  
    Note: SVacc = 0; S-curve without linear parts.

**@Return Code**

ERR_CardNoError	: 30
ERR_AxisRangeError	: 26
ERR_CanNotPitchCompensation	: 77
ERR_MotionBusy	: 70
ERR_SpeedError	: 11
ERR_PosOutOfRange	: 13
ERR_NoError	: 0



## 7.10. Home Return Mode (Origin Return)

### @Function Name

- \_8443\_set\_home\_config** - Set the configuration for home return operation
- \_8443\_home\_move** - Perform a home return operation
- \_8443\_escape\_home** - Escape the home function
- \_8443\_home\_search** - Auto-search home switch (without ORGOffset setting, Default ORGOffset = 100)
- \_8443\_auto\_home\_search** - Auto-search home switch (with ORGOffset)

### @Function Description

#### **\_8443\_set\_home\_config:**

Configure the home return mode, origin and Index signal (EZ) Logic, EZ count and ERC output options for **home\_move()** function. See subsection 5.1.11 for the setting of home mode control.

#### **\_8443\_home\_move:**

This function will cause the axis to perform a home return operation according to the setting of **\_8443\_set\_home\_config()** function. The direction of moving is determined by the sign of velocity parameter (StaVel, MaxVel). Since the stopping condition of this function is determined by **home\_mode** setting, you should be careful to select the initial moving direction. Or you should be careful to handle the condition when a limit switch is touched or other conditions that is possible causing the axis to stop. Executing **v\_stop()** function during **home\_move()** can stop the axis.

#### **\_8443\_escape\_home:**

After homing, use this function to leave home.

#### **\_8443\_home\_search:**

#### **\_8443\_auto\_home\_search:**

Both of the above two functions have the same behavior for Auto-Search of the home switch. The two functions are used to perform a home-search move according to the settings of **\_8443\_set\_home\_config()** function. The direction of movement is set by the sign of the velocity parameter (MaxVel). If MaxVel is positive value, the moving direction is positive and the vice versa. User would also select a specified setting of home mode, moving direction, starting velocity, maximum velocity, and acceleration for a specified home application. The only one difference between them is ORG Offset parameter. **\_8443\_home\_search()** with default ORGOffset value is 100 pulse. However, you can use **\_8443\_auto\_home\_search()** to specify a new ORGOffset value, and modify the default ORGOffset value. For the details, see **Home Search Example** in chapter 5.1.11.

### @Syntax

#### **C/C++ (DOS, Windows XP/7/8)**

```

I16 _8443_set_home_config(I16 AxisNo, I16 home_mode, I16 org_logic, I16 ez_logic, I16 ez_count, I16 erc_out);
I16 _8443_home_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _8443_escape_home(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _8443_home_search(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _8443_auto_home_search(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc, F64 ORGOffset);

```

#### **VB.NET (Windows XP/7/8)**

```

B _8443_set_home_config(ByVal AxisNo As Short, ByVal home_mode As Short, ByVal org_logic As Short, ByVal ez_logic
As Short, ByVal ez_count As Short, ByVal erc_out As Short) As Short
B _8443_home_move(ByVal AxisNo As Short, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As
Short
B _8443_escape_home(ByVal AxisNo As Short, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double)
As Short
B _8443_home_search(ByVal AxisNo As Short, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double)
As Short
B _8443_auto_home_search(ByVal AxisNo As Short, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As
Double, ByVal ORGOffset As Double) As Short

```

**C# (Windows XP/7/8)**

```

Int16 _8443_set_home_config(Int16 AxisNo, Int16 home_mode, Int16 org_logic, Int16 ez_logic, Int16 ez_count, Int16
erc_out);
Int16 _8443_home_move(Int16 AxisNo, Double StrVel, Double MaxVel, Double Tacc);
Int16 _8443_escape_home(Int16 AxisNo, Double SrVel, Double MaxVel, Double Tacc);
Int16 _8443_home_search(Int16 AxisNo, Double StrVel, Double MaxVel, Double Tacc, Double ORGOffset);
Int16 _8443_auto_home_search(Int16 AxisNo, Double StrVel, Double MaxVel, Double Tacc, Double ORGOffset);

```

**@Argument**

**AxisNo:** Designated axis number (0 starts))

**home\_mode:** Stopping modes for home return operation, 0 ~ 12 (See subsection 5.1.11)

**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 (See subsection 5.1.11)

**erc\_out:** ERC output option

erc\_out=0            No erc output

erc\_out=1            erc is output when homing is completed

**StrVel:** Starting velocity in a velocity profile (unit: pps)

**MaxVel:** Maximum velocity in a velocity profile (unit: pps)

**Tacc:** Specified acceleration time (unit: sec)

**ORGOffset:** The escape pulse amounts when the home search touches the home signal (=when the home signal is ON)

**@Return Code**

ERR\_NoError : 0

## 7.11. Manual Pulser Operation

### @Function Name

<code>_8443_disable_pulser_input</code>	- Disable the pulser input
<code>_8443_set_pulser_iptmode</code>	- Set the input signal modes of pulser
<code>_8443_pulser_vmove</code>	- Start the manual pulser v_move
<code>_8443_pulser_pmove</code>	- Start the manual pulser p_move
<code>_8443_pulser_home_move</code>	- Start the manual pulser home operation
<code>_8443_set_pulser_ratio</code>	- Set the manual pulser ratio for actual output pulse rate
<code>_8443_pulser_r_line2</code>	- Start the pulser mode for 2-axis linear interpolation
<code>_8443_pulser_r_arc2</code>	- Start the pulser mode for 2-axis circular interpolation

### @Function Description

`_8443_disable_pulser_input`: This function is used to set the pulser input disable or enable.

`_8443_set_pulser_iptmode`: This function is used to configure the input mode of manual pulser.

`_8443_pulser_vmove`:

As this command is written, the axis starts to move according to the manual pulser input. The axis will move one step when receiving one pulse from the pulser until the **`sd_stop`** or **`emg_stop`** command is written.

`_8443_pulser_pmove`:

With this command, the axis starts to move according to the manual pulser input. The axis will move one step when receiving one pulse from the pulser until the **`sd_stop`** or **`emg_stop`** command is written or the output pulse number reach the Dist.

`_8443_pulser_home_move`:

As this command is written, the axis starts to move according to the manual pulser input. The axis will move one step when receiving one pulse from pulser until the **`sd_stop`** or **`emg_stop`** command is written or the home move is completed.

`_8443_set_pulser_ratio`:

With this command, you can perform multiplication setting when outputting command pulses from pulser input.

`_8443_pulser_r_line2`:

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

`_8443_pulser_r_arc2`:

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

### @Syntax

#### C/C++ (DOS, Windows XP/7/8)

```
I16 _8443_disable_pulser_input(I16 AxisNo, U16 Disable);
I16 _8443_set_pulser_iptmode(I16 AxisNo, I16 InputMode, I16 Inverse);
I16 _8443_pulser_vmove(I16 AxisNo, F64 SpeedLimit);
I16 _8443_pulser_pmove(I16 AxisNo, F64 Dist, F64 SpeedLimit);
I16 _8443_pulser_home_move(I16 AxisNo, I16 HomeType, F64 SpeedLimit);
I16 _8443_set_pulser_ratio(I16 AxisNo, I16 PDV, I16 PMG);
I16 _8443_pulser_r_line2(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY, F64 SpeedLimit);
I16 _8443_pulser_r_arc2(I16 CardNo, I16 *AxisArray, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
```

**VB.NET (Windows XP/7/8)**

```

B_8443_disable_pulser_input(ByVal AxisNo As Short, ByVal Disable As Integer) As Short
B_8443_set_pulser_iptmode(ByVal AxisNo As Short, ByVal InputMode As Short, ByVal Inverse As Short) As Short
B_8443_pulser_vmove(ByVal AxisNo As Short, ByVal SpeedLimit As Double) As Short
B_8443_pulser_pmove(ByVal AxisNo As Short, ByVal Dist As Double, ByVal SpeedLimit As Double) As Short
B_8443_pulser_home_move(ByVal AxisNo As Short, ByVal HomeType As Short, ByVal SpeedLimit As Double) As Short
B_8443_set_pulser_ratio(ByVal AxisNo As Short, ByVal PDV As Short, ByVal PMG As Short) As Short
B_8443_pulser_r_line2(ByVal CardNo As Short, ByVal AxisArray() As Short, ByVal DistX As Double, ByVal DistY As Double, ByVal SpeedLimit As Double) As Short
B_8443_pulser_r_arc2(ByVal CardNo As Short, ByVal AxisArray() As Short, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Short, ByVal MaxVel As Double) As Short

```

**C# (Windows XP/7/8)**

```

Int16 _8443_disable_pulser_input(Int16 AxisNo, UInt16 Disable);
Int16 _8443_set_pulser_iptmode(Int16 AxisNo, Int16 InputMode, Int16 Inverse);
Int16 _8443_pulser_vmove(Int16 AxisNo, Double SpeedLimit);
Int16 _8443_pulser_pmove(Int16 AxisNo, Double Dist, Double SpeedLimit);
Int16 _8443_pulser_home_move(Int16 AxisNo, Int16 HomeType, Double SpeedLimit);
Int16 _8443_set_pulser_ratio(Int16 AxisNo, Int16 PDV, Int16 PMG);
Int16 _8443_pulser_r_line2(Int16 CardNo, ref Int16 AxisArray, Double DistX, Double DistY, Double SpeedLimit);
Int16 _8443_pulser_r_arc2(Int16 CardNo, ref Int16 AxisArray, Double OffsetCx, Double OffsetCy, Double OffsetEx, Double OffsetEy, Int16 DIR, Double MaxVel);

```

**@Argument**

**AxisNo:** Designated axis number (0 starts)

**Disable:** Disable pulser input

Disable=1: Disable pulser

Disable=0: Enable pulser

**InputMode:** Setting of manual pulser input mode from PA and PB

InputMode = 0: 1 x A/B phase type pulse input

InputMode = 1: 2 x A/B phase type pulse input

InputMode = 2: 4 x A/B phase type pulse input

InputMode = 3: 4X A/B phase type pulse input

**Inverse:** Reverse the moving direction from the pulse direction

Inverse= 0: No reverse

Inverse= 1: Reverse

**SpeedLimit:** The maximum speed in pulser operation.

For example, if SpeedLimit is set to 100 pps, the axis can move at 100 pps at the fastest, even the input pulser signal rate is more than 100 pps.

**Dist:** Specified the relative distance to move

**HomeType:** Specified home move type

HomeType = 0: Command Origin.(The axis stops when the command counter becomes '0')

HomeType = 1: 1, Feedback Origin.(The axis stops when the feedback counter becomes '0')

**PDV, PMG:** Division and Multiple factor settings:

The settable range is: PDV: 0 ~ 2047, PMG: 0 ~31.

The formula for calculating the number of output pulses is as follows:

When PDV = 1 ~ 2047

Output Pulse Count = Input Pulser Count x (PMG + 1) x PDV / 2048

When PDV = 0

Output Pulse Count = Input Pulser Count x (PMG + 1)

DistX: Specified relative distance of axis X to move

DistY: Specified relative distance of axis Y to move

OffsetCx: X-axis offset to the center

OffsetCy: Y-axis offset to the center

OffsetEx: X-axis offset to the end of arc

OffsetEy: Y-axis offset to the end of arc

DIR: Specified direction of the arc; 0: CW, 1: CCW

MaxVel: Maximum tangential velocity (unit: pps)

**@Return Code**

ERR\_NoError : 0

ERR\_PulserHomeTypeError : 35

## 7.12. Motion Status

### @Function Name

**\_8443\_motion\_done** - Return the axis motion status

### @Function Description

**\_8443\_motion\_done:**

Return the motion status

### @Syntax

**C/C++ (DOS, Windows XP/7/8)**

```
l16 _8443_motion_done(l16 AxisNo);
```

**VB.NET (Windows XP/7/8)**

```
B_8443_motion_done(ByVal AxisNo As Short) As Short
```

**C# (Windows XP/7/8)**

```
Int16 _8443_motion_done(Int16 AxisNo);
```

### @Argument

**AxisNo:** Number of the axis (0 starts)

### @Return Code

- 0 Stop
- 1 Reserved
- 2 Reserved
- 3 Reserved
- 4 Wait for other axes to stop
- 5 Wait for ERC output to finish
- 6 Wait for DIR Change
- 7 Backlash is being compensated
- 8 Wait for PA/PB input
- 9 In home special speed motion
- 10 In start velocity operation
- 11 In accelerating
- 12 In max velocity operation
- 13 In decelerating
- 14 Wait for INP
- 15 Other (Other axes are still moving)

## 7.13. Motion Interface I/O

### @Function Name

<b>_8443_set_alm</b>	- Set alarm logic and alarm operating mode
<b>_8443_set_el</b>	- Set EL operating mode
<b>_8443_set_inp</b>	- Set INP logic and operating mode (INP enabled/disabled)
<b>_8443_set_erc</b>	- ERC logic and output timing
<b>_8443_set_servo</b>	- SVON signal terminal (General purpose output) ON/OFF
<b>_8443_set_sd</b>	- Set SD logic and operation mode (SD enabled/disabled)

### @Function Description

#### **\_8443\_set\_alm:**

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

#### **\_8443\_set\_el:**

Set the reacting modes for EL signal.

#### **\_8443\_set\_inp:**

Set the active logic of In-Position signal input from a servo driver. You can select either enable or disable of this function. The default state is disabled.

#### **\_8443\_set\_erc:**

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

#### **\_8443\_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 status to GND.

#### **\_8443\_set\_sd:**

Set the active logic, the latch control and the operating mode of SD signal input from a mechanical system. Users can select whether or not to enable this function. The default state is disabled.

### @Syntax

#### **C/C++ (DOS, Windows XP/7/8)**

```

I16 _8443_set_alm(I16 AxisNo, I16 alm_logic, I16 alm_mode);
I16 _8443_set_el(I16 AxisNo, I16 el_mode);
I16 _8443_set_inp(I16 AxisNo, I16 inp_enable, I16 inp_logic);
I16 _8443_set_erc(I16 AxisNo, I16 erc_logic, I16 erc_on_time);
I16 _8443_set_servo(I16 AxisNo, I16 on_off);
I16 _8443_set_sd(I16 AxisNo, I16 enable, I16 sd_logic, I16 sd_latch, I16 sd_mode);

```

#### **VB.NET (Windows XP/7/8)**

```

B_8443_set_alm(ByVal AxisNo As Short, ByVal alm_logic As Short, ByVal alm_mode As Short) As Short
B_8443_set_el(ByVal AxisNo As Short, ByVal el_mode As Short) As Short
B_8443_set_inp(ByVal AxisNo As Short, ByVal inp_enable As Short, ByVal inp_logic As Short) As Short
B_8443_set_erc(ByVal AxisNo As Short, ByVal erc_logic As Short, ByVal erc_on_time As Short) As Short
B_8443_set_servo(ByVal AxisNo As Short, ByVal on_off As Short) As Short
B_8443_set_sd(ByVal AxisNo As Short, ByVal enable As Short, ByVal sd_logic As Short, ByVal sd_latch As Short, ByVal sd_mode As Short) As Short

```

**C# (Windows XP/7/8)**

```

Int16 _8443_set_alm(Int16 AxisNo, Int16 alm_logic, Int16 alm_mode);
Int16 _8443_set_el(Int16 AxisNo, Int16 el_mode);
Int16 _8443_set_inp(Int16 AxisNo, Int16 inp_enable, Int16 inp_logic);
Int16 _8443_set_erc(Int16 AxisNo, Int16 erc_logic, Int16 erc_on_time);
Int16 _8443_set_servo(Int16 AxisNo, Int16 on_off);
Int16 _8443_set_sd(Int16 AxisNo, Int16 enable, Int16 sd_logic, Int16 sd_latch, Int16 sd_mode)

```

**@Argument**

**AxisNo:** Number of the axis (0 starts)

**alm\_logic:** Set the logic of 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 and stops.

**el\_mode:** Reacting modes when receiving EL signal.

el\_mode = 0, Motor immediately stops. (default)

el\_mode = 1, Motor decelerates and 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 = 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 of 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, No 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 and stop



**@Return Code**

ERR\_NoError : 0

## 7.14. Motion I/O Monitoring

### @Function Name

**\_8443\_get\_io\_status** - Obtain all motion I/O status

### @Function Description

**\_8443\_get\_io\_status:**

Obtain all of the motion I/O status. The definition of each bit is as follows:

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

### @Syntax

**C/C++ (DOS, Windows XP/7/8)**

```
l16 _8443_get_io_status(l16 AxisNo, U16 *io_sts);
```

**VB.NET (Windows XP/7/8)**

```
B_8443_get_io_status(ByteVal AxisNo As Short, ByRef io_sts As Short) As Short
```

**C# (Windows XP/7/8)**

```
Int16 _8443_get_io_status(Int16 AxisNo, ref UInt16 io_sts);
```

### @Argument

**AxisNo:** Number of the axis (0 starts)

**\*io\_sts:** I/O status

Where "1" is ON and "0" is OFF. ON/OFF state is read based on the corresponding logic to set.

### @Return Code

ERR\_NoError : 0

## 7.15. Interrupt Operation

### @Function Name

<b><code>_8443_int_control</code></b>	- Enable/Disable of INT operation
<b><code>_8443_set_int_factor</code></b>	- Set INT factor
<b><code>_8443_int_enable</code></b>	- Enable event
<b><code>_8443_int_disable</code></b>	- Disable event
<b><code>_8443_get_int_status</code></b>	- Obtain INT Status
<b><code>_8443_link_interrupt</code></b>	- Set link to the interrupt call back function
<b><code>_8443_set_axis_stop_int</code></b>	- Enable axis stop INT
<b><code>_8443_mask_axis_stop_int</code></b>	- Mask axis stop INT

### @Function Description

#### **`_8443_int_control`:**

This function is used to enable an interrupt generation to the host PC

#### **`_8443_set_int_factor`:**

This function allows you to select the factor to initiate an event INT. INT status of PPCle-8443 is composed of two independent parts: **`error_int_status`** and **`event_int_status`**.

The **`event_int_status`** recodes motions and comparator events under normal operations, and this kind of INT status can be masked (=disabled) by **`_8443_set_int_factor()`**.

The **`error_int_status`** is for an abnormal stop of PPCle-8443 such as EL, ALM ...etc. This kind of INT status cannot be masked.

The following is the definition of these two **`int_status`**. By setting the relative bit to "1", PPCle-8443 can generate INT signal to the host PC.

Bit	Description
0	Normal stop
1	Next command continued
2	Command when 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 is satisfied
12	Compared triggered for axis 0 and 1
13	(Reserved)
14	Latched for axis 2 and 3
15	When latching the count value by ORG input
16	SD input ON
17	(Reserved)

18	CSTA, Sync. start ON
19	(Reserved)
20~30	(Reserved)
31	Axis stop interrupt occurred. Only this bit cannot be masked by <b><i>8443_int_factor()</i></b> . See the function descriptions of <b><i>_8443_set_axis_stop_int()</i></b> and <b><i>_8443_mask_axis_stop_int()</i></b> for the details.

#### ***\_8443\_int\_enable:***

This function is used to assign a Windows INT event.

#### ***\_8443\_int\_disable:***

This function is used to disable the Windows INT event.

#### ***\_8443\_get\_int\_status:***

This function allows you to identify what causes the interrupt signal. After you get 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 cannot be masked by ***\_8443\_set\_int\_factor()***. The definitions for bits of ***error\_int\_status*** are as follows:

<b><i>Error interrupt factors: error_int_status</i></b>	
<b>Bit</b>	<b>Description</b>
0	+SL(Software Limit) stop
1	−SL(Software Limit) stop
2	(Reserved)
3	General Comparator stop
4	(Reserved)
5	+EL(End Limit)
6	−EL
7	ALM(Alarm)
8	(Reserved)
9	(Reserved)
10	SD(Ramp down) is ON and stop
11	(Reserved)
12	Interpolation error and stop
13	Other axes stop during interpolations
14	Pulser input buffer overflow and stop
15	Interpolation counter overflow
16	Encoder input signal error
17	Pulser input signal error
18~31	(Reserved)

Event interrupt factors: The second one is for `event_int_status`, which can be masked by `_8443_set_int_factor()`. The definition for the bit of `event_int_status` is as follows:

<b>Event interrupt factors: <code>event_int_status</code></b>	
<b>Bit</b>	<b>Description</b>
0	Normal stop
1	Next command continued
2	Continuous pre-register is empty and a new command can be written
3	(Reserved)
4	Acceleration start
5	Acceleration end
6	Deceleration start
7	Deceleration end
8	(Reserved)
9	(Reserved)
10	Out of step(Step-losing) occurs
11	General comparator is satisfied
12	Compared triggered for axis 0 and 1
13	(Reserved)
14	Latched for axis 2 and 3
15	ORG is ON
16	SD is ON
17	(Reserved)
18	(Reserved)
19	CSTA, Sync. start is ON
20~31	(Reserved)

#### **\_8443\_link\_interrupt:**

This function is used to link interrupt call back functions.

#### **\_8443\_set\_axis\_stop\_int:**

This function enables an axis stop interrupt factor. When enabled, an interrupt will occur regardless of a normal stop or an error stop. This interrupt condition can be turned on or off accompanied every motion command by setting `_8443_mask_axis_stop_int()`. See the function descriptions. This kind of interrupt condition is different from `_8443_set_int_factor()`. It can be controlled in each motion command, and it is possible to set an interrupt only when the final command is completed in continuous motion.

Note 1: Enable the interrupt function `_8443_int_enable()` and `_8443_int_control()` before using the axis stop interrupt.

Note 2: Execution of `_8443_get_int_status()` is also required for checking the interrupt status factor. Bit 31 of the event interrupt parameter "`event_int_status`" indicates that an axis stop interrupt has occurred.

#### **\_8443\_mask\_axis\_stop\_int:**

This function will affect an axis stop interrupt factor which is set by `_8443_set_axis_stop_int()`. This function is usually used in a continuous motion. When setting parameter "`int_enable`" to 1, only the final motion command will issue an axis stop interrupt in the progression of continuous operation. When setting parameter "`int_enable`" to 0, each motion command issues an axis stop interrupt in the progression of continuous motion.

**@Syntax****C/C++ (Windows XP/7/8)**

```

I16 _8443_int_control(U16 cardNo, U16 intFlag );
I16 _8443_set_int_factor(I16 AxisNo, U32 int_factor );
I16 _8443_int_enable(I16 CardNo, HANDLE *phEvent);
I16 _8443_int_disable(I16 CardNo);
I16 _8443_get_int_status(I16 AxisNo, U32 *error_int_status, U32 *event_int_status );
I16 _8443_link_interrupt(I16 CardNo, void ( __stdcall *callbackAddr)(I16 IntAxisNoInCard));
I16 _8443_set_axis_stop_int(I16 AxisNo, I16 axis_stop_int);
I16 _8443_mask_axis_stop_int(I16 AxisNo, I16 int_disable);

```

**VB.NET (Windows XP/7/8)**

```

B_8443_int_control(ByVal cardNo As Short, ByVal intFlag As Short) As Short
B_8443_set_int_factor(ByVal AxisNo As Short, ByVal int_factor As Integer) As Short
B_8443_int_enable(ByVal CardNo As Short, ByRef phEvent As Int32) As Short
B_8443_int_disable(ByVal CardNo As Short) As Short
B_8443_get_int_status(ByVal AxisNo As Short, ByRef error_int_status As Integer, ByRef event_int_status As Integer) As Short
B_8443_link_interrupt(ByVal CardNo As Short, ByVal lpCallbackProc As Integer) As Short
B_8443_set_axis_stop_int(ByVal AxisNo As Short, ByVal axis_stop_int As Short) As Short
B_8443_mask_axis_stop_int(ByVal AxisNo As Short, ByVal int_disable As Short) As Short

```

**C# (Windows XP/7/8)**

```

Int16 _8443_int_control(UInt16 CardNo, UInt16 intFlag );
Int16 _8443_set_int_factor(Int16 AxisNo, UInt32 int_factor );
Int16 _8443_int_enable( Int16 CardNo, ref IntPtr phEvent);
Int16 _8443_int_disable(Int16 CardNo);
Int16 _8443_get_int_status(Int16 AxisNo,ref UInt32 error_int_status,ref UInt32 event_int_status );
Int16 _8443_link_interrupt(Int16 CardNo, Callback mycallback);
Int16 _8443_set_axis_stop_int(Int16 AxisNo, Int16 axis_stop_int);
Int16 _8443_mask_axis_stop_int(Int16 AxisNo, Int16 int_disable);

```

**@Argument****CardNo:** Board number (0 starts)**AxisNo:** Axis number (0 starts)**intFlag:** int flag, 0 or 1 (0: Disable, 1: Enable)**int\_factor:** interrupt factor (see previous table)**\*phEvent:** Interrupt event handler (Windows)**\*error\_int\_status:** Error interrupt cause (see the previous table).**\*event\_int\_status:** Event interrupt factor (see the previous table).**int\_disable:** 0: Each motion command issues an axis stop interrupt.

1: Only the last motion command issues an axis stop interrupt in the progression of continuous motion.

**axis\_stop\_int:** Axis stop interrupt (0: Disable 1: Enable )**@Return Code**

ERR_NoError	: 0
ERR_EventNotEnableYet	: 37
ERR_LinkIntError	: 31
ERR_CardNoError	: 30

## 7.16. Position Controls and Counters

### @Function Name

<code>_8443_get_position</code>	- Get the value of feedback position counter
<code>_8443_set_position</code>	- Set the value of feedback position counter
<code>_8443_get_command</code>	- Get the value of command position counter
<code>_8443_set_command</code>	- Set the value of command position counter
<code>_8443_get_error_counter</code>	- Get the value of position error counter
<code>_8443_reset_error_counter</code>	- Reset the position error counter
<code>_8443_get_general_counter</code>	- Get the value of general-purpose counter
<code>_8443_set_general_counter</code>	- Set the general-purpose counter
<code>_8443_get_target_pos</code>	- Get the value of target position recorder
<code>_8443_reset_target_pos</code>	- Reset target position recorder
<code>_8443_get_rest_command</code>	- Get remaining pulse number until the end of operation
<code>_8443_check_rdp</code>	- Get the ramping down point value data
<code>_8443_set_auto_rdp</code>	- Enable the automatic setting ramping-down point

### @Function Description

#### `_8443_get_position()`:

This function is used to read the value in the feedback position counter.

**Note:** This is a value converted by move ratio set by `_8443_set_move_ratio` function. If the move ratio is 0.5, the value read will be twice of the counter value of PCL6046. The source of feedback counter is selectable by either function `_8443_set_feedback_src()` to be external EA/EB or command pulse output of PPCle-8443.

#### `_8443_set_position()`:

This function is used to change the feedback position counter to a specified value.

Note: This value to be set here will be calculated by move ratio function: `_8443_set_move_ratio`. If the move ratio is 0.5, the set value in the PCL6046 counter will be 1/2 of the given value.

#### `_8443_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 PPCle-8443.

#### `_8443_set_command()`:

This function is used to set the specified value in command position counter.

#### `_8443_get_error_counter()`:

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

#### `_8443_reset_error_counter()`:

This function is used to clear the position error counter.

#### `_8443_get_general_counter()`:

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

#### `_8443_set_general_counter()`:

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

#### `_8443_get_target_pos()`:

This function is used to read the value of target position recorder. The target position recorder is maintained by PPCle-8443 software driver. It records the position for the current running motion to settle down.

#### `_8443_reset_target_pos()`:

This function is used to set a new value for the target position recorder. It is necessary to call this function when a home return operation is completed or when a new feedback counter value is set by function `_8443_set_position()`.

**\_8443\_get\_rest\_command():**

This function is used to read remaining pulse counts until the end of current operation.

**\_8443\_check\_rdp():**

This function is used to read the ramping down point data. The ramping down point is a position where a deceleration starts. The data is stored as the number of pulse count, and the axis will start to decelerate when the remaining pulse count reaches the set data.

**\_8443\_set\_auto\_rdp():**

Enable the automatic ramping-down point setting. The default setting is manual setting (off) and the RPD value is calculated by DLL. If you select the automatic setting (On), the RPD value is calculated by motion ASIC PCL6046, but there are some limitations. We recommend you to use the default setting (manual setting).

**@Syntax**

C/C++ (DOS, Windows XP/7/8)

```

I16 _8443_get_position(I16 AxisNo, F64 *pos);
I16 _8443_set_position(I16 AxisNo, F64 pos);
I16 _8443_get_command(I16 AxisNo, I32 *cmd);
I16 _8443_set_command(I16 AxisNo, I32 cmd);
I16 _8443_get_error_counter(I16 AxisNo, I16 *error_counter);
I16 _8443_reset_error_counter(I16 AxisNo);
I16 _8443_get_general_counter(I16 AxisNo, F64 *CntValue);
I16 _8443_set_general_counter(I16 AxisNo, I16 CntSrc, F64 CntValue);
I16 _8443_get_target_pos(I16 AxisNo, F64 *T_pos);
I16 _8443_reset_target_pos(I16 AxisNo, F64 T_pos);
I16 _8443_get_rest_command(I16 AxisNo, I32 *rest_command);
I16 _8443_check_rdp(I16 AxisNo, I32 *rdp_command);
I16 _8443_set_auto_rdp(I16 CardNo, I16 on_off);

```

**VB.NET (Windows XP/7/8)**

```

B _8443_get_position(ByVal AxisNo As Short, ByRef pos As Double) As Short
B _8443_set_position(ByVal AxisNo As Short, ByVal pos As Double) As Short
B _8443_get_command(ByVal AxisNo As Short, ByRef cmd As Integer) As Short
B _8443_set_command(ByVal AxisNo As Short, ByVal cmd As Integer) As Short
B _8443_get_error_counter(ByVal AxisNo As Short, ByRef Err As Short) As Short
B _8443_reset_error_counter(ByVal AxisNo As Short) As Short
B _8443_get_general_counter(ByVal AxisNo As Short, ByRef CntValue As Double) As Short
B _8443_set_general_counter(ByVal AxisNo As Short, ByVal CntSrc As Short, ByVal CntValue As Double) As Short
B _8443_get_target_pos(ByVal AxisNo As Short, ByRef pos As Double) As Short
B _8443_reset_target_pos(ByVal AxisNo As Short, ByVal pos As Double) As Short
B _8443_get_rest_command(ByVal AxisNo As Short, ByRef rest_command As Integer) As Short
B _8443_check_rdp(ByVal AxisNo As Short, ByRef rdp_command As Integer) As Short
B _8443_set_auto_rdp(ByVal CardNo As Short, ByVal on_off As Short) As Short

```

**C# (Windows XP/7/8)**

```

Int16 _8443_get_position(Int16 AxisNo, ref Double pos);Int16 _8443_get_position(Int16 AxisNo, ref Double pos);Int16
_8443_get_position(Int16 AxisNo, ref Double pos);Int16 _8443_get_position(Int16 AxisNo, ref Double pos);Int16
_8443_get_position(Int16 AxisNo, ref Double pos);Int16 _8443_get_position(Int16 AxisNo, ref Double pos);Int16
_8443_get_position(Int16 AxisNo, ref Double pos);Int16 _8443_get_position(Int16 AxisNo, ref Double pos);Int16
_8443_get_position(Int16 AxisNo, ref Double pos);Int16 _8443_get_position(Int16 AxisNo, ref Double pos);Int16
_8443_get_position(Int16 AxisNo, ref Double pos);
Int16 _8443_set_position(Int16 AxisNo, Double pos);
Int16 _8443_get_command(Int16 AxisNo, ref Int32 cmd);
Int16 _8443_set_command(Int16 AxisNo, Int32 cmd);
Int16 _8443_get_error_counter(Int16 AxisNo, ref Int16 error);
Int16 _8443_reset_error_counter(Int16 AxisNo);
Int16 _8443_get_general_counter(Int16 AxisNo, ref Double CntValue);
Int16 _8443_set_general_counter(Int16 AxisNo, Int16 CntSrc, Double CntValue);

```



```

Int16 _8443_get_target_pos(Int16 AxisNo, ref Double pos);
Int16 _8443_reset_target_pos(Int16 AxisNo, Double pos);
Int16 _8443_get_rest_command(Int16 AxisNo, ref Int32 rest_command);
Int16 _8443_check_rdp(Int16 AxisNo, ref Int32 rdp_command);
Int16 _8443_set_auto_rdp(Int16 CardNo, Int16 on_off);

```

#### @Argument

**AxisNo:** Axis number (0 starts)

**Pos, \*Pos:** Feedback position counter value (Range: -2147483648 ~ 2147483647)

**cmd, \*cmd:** Command position counter value (Range: -2147483648 ~ 2147483647)

**error\_counter, \*error\_counter:** Position error counter value (Range: -32768 ~ 32767)

**T\_pos, \*T\_pos:** Target position recorder value (Range: -2147483648 ~ 2147483647)

**CntValue, \*CntValue:** General purpose counter value (Range: -2147483648 ~ 2147483647)

**rest\_command, \*rest\_command:** Remaining pulse count until the end of operation (Range: -2147483648 ~ 2147483647)

**rdp\_command, \*rdp\_command:** Ramping down point value data (range: 0 ~ 167777215)

**CntSrc:** Source of general purpose counter input data selection:

CntSrc=0: Command pulse

CntSrc=1: EA / EB

CntSrc=2: PA / PB Default setting)

CntSrc=3: CLK / 2

**on\_off:** Automatic ramping-down point setting (ON/OFF)

0 (off): Manual RDP: Ramp down point calculation (RDP calculated by DLL)

1 (on): Automatic RDP Ramp down point calculation (RDP calculated by PCL6046)

#### @Return Code

ERR\_NoError : 0

ERR\_PosOutOfRange : 13

## 7.17. Position comparator and Latch

### @Function Name

<code>_8443_set_ltc_logic</code>	- Set the LTC logic
<code>_8443_get_latch_data</code>	- Get the Latch counter data
<code>_8443_set_soft_limit</code>	- Set the Software limit
<code>_8443_enable_soft_limit</code>	- Enable the Software limit function
<code>_8443_disable_soft_limit-</code>	- Disable the Software limit function
<code>_8443_set_error_counter_check</code>	- Set the out of step (Step losing) detection
<code>_8443_set_general_comparator</code>	- Set the general purpose comparator
<code>_8443_set_trigger_comparator</code>	- Set the trigger comparator
<code>_8443_set_trigger_type</code>	- Set the trigger output type
<code>_8443_check_compare_data</code>	- Check the current comparator data
<code>_8443_check_compare_status</code>	- Check the current comparator status
<code>_8443_set_auto_compare</code>	- Set the comparator data source for auto loading
<code>_8443_build_compare_function</code>	- Build the comparator data via constant interval
<code>_8443_build_compare_table</code>	- Build the comparator data via compare table
<code>_8443_cmp_v_change</code>	- Speed change by the comparator
<code>_8443_set_latch_source</code>	- Set the latch signal for counter

### @Function Description

#### `_8443_set_ltc_logic()`:

This function is used to set the logic of a latch input. This function is applicable only for the last two axes in each PPCle-8443 board.

#### `_8443_get_latch_data()`:

This function is used to read latched value of a counter when a latch signal is input.

#### `_8443_set_soft_limit()`:

This function is used to set a value of software limit.

#### `_8443_enable_soft_limit(),_8443_disable_soft_limit()`:

These two functions are used to enable/disable the software limit function. When enabled, the action of software limit will be exactly the same as a physical hardware limit.

#### `_8443_set_error_counter_check()`:

This function is used to enable out-of-step (step losing) checking function. By giving a tolerance value, the PPCle-8443 will generate an interrupt (***event\_int\_status, bit 10***) when a position error counter exceeds the tolerance value.

#### `_8443_set_general_comparator()`:

This function is used to set a source and a comparing value for the general purpose comparator. When the source counter value reaches the comparing value, the PPCle-8443 will generate an interrupt (***event\_int\_status, bit 11***).

#### `_8443_set_trigger_comparator()`:

This function is used to set a comparing method and a value for trigger comparator. When the feedback position counter value reaches the comparing value, the PPCle-8443 will generate a trigger pulse output via CMP and an interrupt (***event\_int\_status, bit 12***) will be also sent to the host PC. If ***\_8443\_set\_auto\_compare()*** function is used, the comparing value set by this function will be ignored automatically.

**Note:** it is applicable only for the first two axes (axis 0 and axis 1) in each PPCle-8443 board.

#### `_8443_set_trigger_type()`:

This function is used to set a trigger output mode.

**\_8443\_check\_compare\_data():**

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

**\_8443\_check\_compare\_status():**

This function is used to get the status of all comparators. When some comparators are satisfied, the relative bit of cmp\_sts will become '1'. When not satisfied, it will become '0'.

**Note 1:** This function cannot be used when **\_8443\_build\_compare\_function()** and **\_8443\_build\_compare\_table()** are used.

**\_8443\_set\_auto\_compare():**

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

This function is used to set comparison target data of the trigger comparator, and enable the automatic continuous comparison function.

The comparison data target data can be generated by the **\_8444\_build\_compare\_function**, or a table composed of arbitrary data can be set.

**\_8443\_build\_compare\_function():**

This function is used to build a comparing function by defining the start / end point and the interval. There is no limitation on the max number of comparing data. It will automatically load the final point after your end point. That is  $(\text{end point} + \text{Interval} \times \text{total points}) \times \text{move ratio}$

*Note: Please turn off all interrupt functions while triggering is running.*

**\_8443\_build\_compare\_table():**

This function is used to build a comparing table by defining data array. The size of array is limited to 1024.

*Note: Please turn off all interrupt functions while triggering is running.*

**\_8443\_cmp\_v\_change():**

This function is used to set up the comparator velocity change function. It is a **v\_change** function but acts when general purpose comparator is satisfied. When this function is used, the parameter "CmpAction" of **\_8443\_set\_general\_comparator()** must be set to '3'.

The compare data is also set by **\_8443\_set\_general\_comparator()**. The remaining distance, the velocity of the comparing point, the new velocity, and the acceleration time are set by the function **\_8443\_cmp\_v\_change()**.

**\_8443\_set\_latch\_source():**

This function is used to set latched signals.

**@Syntax****C/C++ (Windows XP/7/8)**

```

I16 _8443_set_ltc_logic(I16 AxisNo_2or3, I16 ltc_logic);
I16 _8443_get_latch_data(I16 AxisNo, I16 LatchNo, F64 *Pos);
I16 _8443_set_soft_limit(I16 AxisNo, I32 PLimit, I32 NLimit);
I16 _8443_disable_soft_limit(I16 AxisNo);
I16 _8443_enable_soft_limit(I16 AxisNo, I16 Action);
I16 _8443_set_error_counter_check(I16 AxisNo, I16 Tolerance, I16 On_Off);
I16 _8443_set_general_comparator(I16 AxisNo, I16 CmpSrc, I16 CmpMethod, I16 CmpAction, F64 Data);
I16 _8443_set_trigger_comparator(I16 AxisNo, I16 CmpSrc, I16 CmpMethod, F64 Data);
I16 _8443_set_trigger_type(I16 AxisNo, I16 TriggerType);
I16 _8443_check_compare_data(I16 AxisNo, I16 CompType, F64 *Pos);
I16 _8443_check_compare_status(I16 AxisNo, U16 *cmp_sts);
I16 _8443_set_auto_compare(I16 AxisNo, I16 SelectSrc);
I16 _8443_cmp_v_change(I16 AxisNo, F64 Res_dist, F64 oldvel, F64 newvel, F64 AccTime)

```

```

I16 _8443_set_latch_source(I16 AxisNo, I16 ltc_src);
I16 _8443_build_compare_function(I16 AxisNo, F64 Start, F64 End, F64 Interval, I16 Device);
I16 _8443_build_compare_table(I16 AxisNo, F64 *TableArray, I32 Size, I16 Device);VB.NET (Windows XP/7/8)
B_8443_set_ltc_logic(ByVal AxisNo As Short, ByVal ltc_logic As Short) As Short
B_8443_get_latch_data(ByVal AxisNo As Short, ByVal LatchNo As Short, ByRef Pos As Double) As Short
B_8443_set_soft_limit(ByVal AxisNo As Short, ByVal PLimit As Integer, ByVal NLimit As Integer) As Short
B_8443_disable_soft_limit(ByVal AxisNo As Short) As Short
B_8443_enable_soft_limit(ByVal AxisNo As Short, ByVal Action As Short) As Short
B_8443_set_error_counter_check(ByVal AxisNo As Short, ByVal Tolerance As Short, ByVal On_Off As Short) As Short
B_8443_set_general_comparator(ByVal AxisNo As Short, ByVal CmpSrc As Short, ByVal CmpMethod As Short, ByVal
CmpAction As Short, ByVal Data As Double) As Short
B_8443_set_trigger_comparator(ByVal AxisNo As Short, ByVal CmpSrc As Short, ByVal CmpMethod As Short, ByVal
Data As Double) As Short
B_8443_set_trigger_type(ByVal AxisNo As Short, ByVal TriggerType As Short) As Short
B_8443_check_compare_data(ByVal AxisNo As Short, ByVal CompType As Short, ByRef Pos As Double) As Short
B_8443_check_compare_status(ByVal AxisNo As Short, ByRef cmp_sts As Short) As Short
B_8443_set_auto_compare(ByVal AxisNo As Short, ByVal SelectSrc As Short) As Short
B_8443_cmp_v_change(ByVal AxisNo As Short, ByVal Res_Dist As Double, ByVal OldVel As Double, ByVal NewVel As
Double, ByVal Time As Double) As Short
B_8443_set_latch_source(ByVal AxisNo As Short, ByVal ltc_src As Short) As Short
B_8443_build_compare_function(ByVal AxisNo As Short, ByVal Start As Double, ByVal EndPoint As Double, ByVal
Interval As Double, ByVal Device As Short) As Short
B_8443_build_compare_table(ByVal AxisNo As Short, ByVal TableArray() As Double, ByVal Size As Integer, ByVal Device
As Short) As Short

```

#### **C# (Windows XP/7/8)**

```

Int16 _8443_set_ltc_logic(Int16 AxisNo, Int16 ltc_logic);
Int16 _8443_get_latch_data(Int16 AxisNo, Int16 LatchNo, ref Double Pos);
Int16 _8443_set_soft_limit(Int16 AxisNo, Int32 PLimit, Int32 NLimit);
Int16 _8443_enable_soft_limit(Int16 AxisNo, Int16 Action);
Int16 _8443_disable_soft_limit(Int16 AxisNo);
Int16 _8443_set_error_counter_check(Int16 AxisNo, Int16 Tolerance, Int16 On_Off);
Int16 _8443_set_general_comparator(Int16 AxisNo, Int16 CmpSrc, Int16 CmpMethod, Int16 CmpAction,
Double Data);
Int16 _8443_set_trigger_comparator(Int16 AxisNo, Int16 CmpSrc, Int16 CmpMethod, Double Data);
Int16 _8443_set_trigger_type(Int16 AxisNo, Int16 TriggerType);

```

#### **@Argument**

**AxisNo:** Axis number (Axis 2 and 3 only)

**ltc\_logic:** LTC signal logic 0: Rising edge 1: Falling edge

**AxisNo:** Axis number (0 starts)

**Counter:** Specify the counter to latch

Counter = 1: Command counter

Counter = 2: Feedback counter

Counter = 3: Error counter

Counter = 4: General purpose counter

**Pos:** Latched counter value

**PLimit:** Software limit value in positive direction

**NLimit:** Software limit value in negative direction

**Action:** Response for software limit ON

Action = 0: INT only

Action = 1: Immediate stop

Action = 2: Decelerate and stop

Action = 3: Reserved

**Tolerance:** Tolerance of out-of-step (step losing) detection

**On\_Off:** Enable / Disable of out-of-step (step-losing) detection

On\_Off = 0: Disable

On\_Off = 1: Enable

**CmpSrc:** Set comparing source counter

CmpSrc = 0: Command counter

CmpSrc = 1: Feedback position counter

CmpSrc = 2: Position error counter

CmpSrc = 3: General purpose counter

**CmpMethod:** Comparing method

CmpMethod = 0: Comparing function off

CmpMethod = 1: Cmp value = Counter (No direction designation)

CmpMethod = 2: Cmp value = Counter (Positive direction)

CmpMethod = 3: Cmp value = Counter (Negative direction)

CmpMethod = 4: Cmp value > Counter

CmpMethod = 5: Cmp value < Counter

**CmpAction:** Reaction when the comparison is satisfied

CmpAction = 0: INT only

CmpAction = 1: Immediate stop

CmpAction = 2: Ramp down and stop

CmpAction = 3: Speed change

**Data:** Comparing value

**TriggerType:** Selection of type of trigger output mode

TriggerType = 0: Normal high level (default value)

TriggerType = 1: Normal low level

**CompType:** Comparator selection

CompType = 1 Positive side software limit

CompType = 2 Negative side software limit

CompType = 3 Error counter comparator value

CompType = 4 General purpose comparator value

CompType = 5 Trigger output comparator value

**cmp\_sts:** Comparator status bit data

Bit: 0 Positive side software limit ON

Bit: 1 Minus side software limit ON

Bit: 2 Error counter comparator ON

Bit: 3 General purpose comparator ON

Bit: 4 Trigger comparator ON (Axis 0 and 1 only)

**SelectSrc:** Automatic continuous comparing function setting

SelectSrc=0: Disable automatic comparing function

SelectSrc=1: Use FIFO (Enable automatic comparing function)

**Start:** Start point setting (for generating a constant interval table)

**End:** End point setting (for generating a constant interval table)

**Interval:** Interval setting (for generating a constant interval table)

**TableArray:** Comparing data table

**Size:** Size of table array (Element count)

**Device:** Selection of reload device for comparator data:

Device=0: RAM and Interrupt

Device=1: FIFO

**Res\_dist:** The remaining distance from the compare point to the target position.

**oldvel:** Velocity at compare point.

**newvel:** New velocity

**AccTime:** Acceleration time

**ltc\_src:** Latch source (signal/condition)

0: LTC input

1: Comparator 4 is satisfied

2: Comparator 5 is satisfied

3: ORG signal

#### @Return Code

ERR_NoError	: 0
ERR_CompareNoError	: 18
ERR_CompareMethodError	: 17
ERR_CompareAxisError	: 19
ERR_CompareTableSizeError	: 20
ERR_CompareFunctionError	: 21
ERR_CompareTableNotReady	: 22
ERR_CompareLineNotReady	: 23
ERR_HardwareCompareAxisWrong	: 32
ERR_AutoCompareSourceWrong	: 33
ERR_CompareDeviceTypeError	: 34

## 7.18. Continuous Operation

### @Function Name

- `_8443_set_continuous_move`** - Set enable/disable of a continuous operation.
- `_8443_check_continuous_buffer`** - Check if the buffer is empty.

### @Function Description

#### **`_8443_set_continuous_move()`:**

This function is necessary to be executed before or after a continuous motion.

#### **`_8443_check_continuous_buffer()`:**

This function is used to detect if the command pre-register is empty. When the command pre-register is empty, you can write a next operation command. If the command pre-register is NOT empty, you cannot write the command.

### @Syntax

#### **C/C++ (DOS, Windows XP/7/8)**

```
l16 _8443_set_continuous_move(l16 AxisNo, l16 conti_flag);
l16 _8443_check_continuous_buffer(l16 AxisNo);
```

#### **VB.NET (Windows XP/7/8)**

```
B_8443_set_continuous_move(ByVal AxisNo As Short, ByVal conti_logic As Short) As Short
B_8443_check_continuous_buffer(ByVal AxisNo As Short) As Short
```

#### **C# (Windows XP/7/8)**

```
Int16 _8443_set_continuous_move(Int16 AxisNo, Int16 conti_logic);
Int16 _8443_check_continuous_buffer(Int16 AxisNo);
```

### @Argument

**AxisNo:** Axis number (0 starts)

**conti\_flag:** Continuous operation enable/disable setting

conti\_flag=0: Continuous operation disabled

conti\_flag=1: Continuous operation enabled

### @Return Code

ERR\_NoError

Return value of **`_8443_check_continuous_buffer()`** :

- 0: All command registers are empty
- 1: Count register is in-use (2 remaining)
- 2: Pre-register 1 is in-use. (1 remaining)
- 3: Pre-register 2 is in-use (0 remaining)

## 7.19. Multiple Axes Simultaneous Operation

### @Function Name

<code>_8443_set_tr_move_all</code>	- Multi-axis simultaneous relative trapezoidal positioning operation setup
<code>_8443_set_ta_move_all</code>	- Multi-axis simultaneous absolute trapezoidal positioning operation setup
<code>_8443_set_sr_move_all</code>	- Multi-axis simultaneous relative S-curve positioning operation setup
<code>_8443_set_sa_move_all</code>	- Multi-axis simultaneous absolute S-curve positioning operation setup
<code>_8443_start_move_all</code>	- Start a simultaneous multi-axis operation
<code>_8443_stop_move_all</code>	- Stop a simultaneous multi-axis operation
<code>_8443_set_sync_option</code>	- Sync operation setting with other axes
<code>_8443_set_sync_stop_mode</code>	- Setting of the stop mode of CSTOP signal.

### @Function Description

These functions are related to simultaneous operations of multi-axis even between different boards. The simultaneous multi-axis operation is to start or to stop moving specified axes at the same time. The move axes are specified by parameter “**AxisArray**” and the number of axes are defined by the parameter “**TotalAxes**” in `_8443_set_tr_move_all()`.

The function `_8443_set_xx_move_all()` is used for the operation setting (velocity, acceleration/deceleration velocities, moving amount) of the axis performs the same operation and places it in the start input wait status.

`_8443_start_move_all()` will send a simultaneous start (STA) signal to the axes in the wait status.

The axes will simultaneously start by the (STA) signal.

`_8443_stop_move_all()` will send a simultaneous stop signal (STP) to the axes for which simultaneous operation has been set.

**Note:** It is necessary to make connections according to section 4.14 “Simultaneous Start/Stop Signals: STA and STP”, so that you can use the above two functions to perform a simultaneous operation between different boards. .

The following code shows how to utilize these functions. This code can perform a relative positioning move between axis 0 (axis 0 in board 0) and axis 4 (axis 0 in board 1) by the moving amounts of 8000.0 and 12000.0 respectively. It is not an interpolation operation, but if you set the velocities and the acceleration times proportional to the ratio of distances, the axes will arrive at the end points at the same time (simultaneous motion).

```
int main()
{
    I16  axes[2] = {0, 4};
    F64  dist[2] = {8000, 12000},
    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};

    _8443_set_tr_move_all(2, axes, dist, str_vel, max_vel, Tacc, Tdec);
    _8443_start_move_all(axes[0]);

    return  ERR_NoError;
}
```



**\_8443\_set\_sync\_option():**

This will set up a simultaneous start of two or more different command operation groups. For example, if you want to start one 2-axis linear interpolation and one 1-axis single operation simultaneously, you can turn on this option and set up before sending a start command to each axis.

In this function, it is also possible to start by waiting for completion of the operation of the other axis. For example, axis1 can start when the operation of axis 2 is completed.

**\_8443\_set\_sync\_stop\_mode():**

This function is used to set up the mode of simultaneous stop. There are two types of stop mode: immediate stop or deceleration stop. When the execution of **\_8443\_stop\_move\_all()** or STP signal input, the axes will stop according to this setting.

**@ Syntax****C/C++ (Windows XP/7/8)**

```

I16 _8443_set_tr_move_all(I16 TotalAxes, I16 *AxisArray, F64 *DistA, F64 *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA);
I16 _8443_set_sa_move_all(I16 TotalAx, I16 *AxisArray, F64 *PosA, F64 *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA, F64 *SVaccA, F64 *SVdecA);
I16 _8443_set_ta_move_all(I16 TotalAx, I16 *AxisArray, F64 *PosA, F64 *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA);
I16 _8443_set_sr_move_all(I16 TotalAx, I16 *AxisArray, F64 *DistA, F64 *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA, F64 *SVaccA, F64 *SVdecA);
I16 _8443_start_move_all(I16 FirstAxisNo);
I16 _8443_stop_move_all(I16 FirstAxisNo);
I16 _8443_set_sync_option(I16 AxisNo, I16 sync_stop_on, I16 cstop_output_on, I16 sync_option1, I16 sync_option2);
I16 _8443_set_sync_stop_mode(I16 AxisNo, I16 stop_mode);

```

**VB.NET (Windows XP/7/8)**

```

B_8443_set_tr_move_all(ByVal TotalAxes As Short, ByVal AxisArray() As Short, ByVal DistA() As Double, ByVal StrVelA() As Double, ByVal MaxVelA() As Double, ByVal TaccA() As Double, ByVal TdecA() As Double) As Short
B_8443_set_sa_move_all(ByVal TotalAx As Short, ByVal AxisArray() As Short, ByVal PosA() As Double, ByVal StrVelA() As Double, ByVal MaxVelA() As Double, ByVal TaccA() As Double, ByVal TdecA() As Double, ByVal SVaccA() As Double, ByVal SVdecA() As Double) As Short
B_8443_set_ta_move_all(ByVal TotalAx As Short, ByVal AxisArray() As Short, ByVal PosA() As Double, ByVal StrVelA() As Double, ByVal MaxVelA() As Double, ByVal TaccA() As Double, ByVal TdecA() As Double) As Short
B_8443_set_sr_move_all(ByVal TotalAx As Short, ByVal AxisArray() As Short, ByVal DistA() As Double, ByVal StrVelA() As Double, ByVal MaxVelA() As Double, ByVal TaccA() As Double, ByVal TdecA() As Double, ByVal SVaccA() As Double, ByVal SVdecA() As Double) As Short
B_8443_start_move_all(ByVal FirstAxisNo As Short) As Short
B_8443_stop_move_all(ByVal FirstAxisNo As Short) As Short
B_8443_set_sync_option(ByVal AxisNo As Short, ByVal sync_stop_on As Short, ByVal cstop_output_on As Short, ByVal sync_option1 As Short, ByVal sync_option2 As Short) As Short
B_8443_set_sync_stop_mode(ByVal AxisNo As Short, ByVal stop_mode As Short) As Short

```

**C# (Windows XP/7/8)**

```

Int16 _8443_set_tr_move_all(Int16 TotalAxes, ref Int16 AxisArray, ref Double DistA, ref Double StrVelA, ref Double MaxVelA, ref Double TaccA, ref Double TdecA);
Int16 _8443_set_ta_move_all(Int16 TotalAx, ref Int16 AxisArray, ref Double PosA, ref Double StrVelA, ref Double MaxVelA, ref Double TaccA, ref Double TdecA);
Int16 _8443_set_sr_move_all(Int16 TotalAx, ref Int16 AxisArray, ref Double DistA, ref Double StrVelA, ref Double MaxVelA, ref Double TaccA, ref Double TdecA, ref Double SVaccA, ref Double SVdecA);
Int16 _8443_set_sa_move_all(Int16 TotalAx, ref Int16 AxisArray, ref Double PosA, ref Double StrVelA, ref Double MaxVelA, ref Double TaccA, ref Double TdecA, ref Double SVaccA, ref Double SVdecA);
Int16 _8443_start_move_all(Int16 FirstAxisNo);
Int16 _8443_stop_move_all(Int16 FirstAxisNo);
Int16 _8443_set_sync_option(Int16 AxisNo, Int16 sync_stop_on, Int16 cstop_output_on, Int16 sync_option1, Int16 sync_option2);
Int16 _8443_set_sync_stop_mode(Int16 AxisNo, Int16 stop_mode);

```

**@Argument**

**TotalAx:** number of axes for simultaneous operation (1 ~ 48)

**AxisArray:** specified axes number array designated to simultaneous operation

Note: Be sure to store axis numbers in the array in ascending order.

**DistA:** Array containing movement amount / target position of each axis (number of pulse)

**StrVelA:** Array containing initial moving speed of each axis (pps)

**MaxVelA:** Array containing the maximum moving speed of each axis (pps)

**TaccA:** Array containing acceleration time of each axis (pps)

**TdecA:** Array containing deceleration time of each axis (pps)

**SVaccA:** Array containing S-curve part in the acceleration of each axis (pps)

**SVdecA:** Array containing S-curve part in the deceleration of each axis (pps)

**FirstAxisNo:** The first axis number in the target Axis Array

**sync\_stop\_on:** Simultaneous stop (STP) signal setting; Enabled/Disabled.

0: Disabled, 1: Enabled

**cstop\_output\_on:** Setting of automatic output of STP signal at abnormal stop by ALM, EL, etc.

0: Disable, 1: Enable

**sync\_option1:** Select command start type:

0: Default (immediately start by operation commands)

1: Wait for simultaneous start input **\_8443\_start\_move\_all()** or start with STA signal input.

2: Reserved

3: Check Sync\_option 2 condition and start

**sync\_option2:** Example; Bit is designated, and multi-axis setting is available.

0: Default value (No stand by)

1: Start when axis 0 stops

2: Start when axis 1 stops

4: Start when axis 2 stops

8: Start when axis 3 stops

5: Start when both axis 0 and axis 2 stop

15: Start when all axis 0 to 3 stop

**stop\_mode:** Set simultaneous stop mode

0: Immediately stop

1: Decelerate to stop

**@Return Code**

ERR\_NoError : 0

ERR\_SpeedError : 11

## 7.20. Extended General-Purpose Input/Output

### @Function Name

**\_8443\_set\_gpio\_output** - Set general purpose output (whole port)  
**\_8443\_get\_gpio\_output** - Obtain the status of general purpose output (whole port)  
**\_8443\_get\_gpio\_input** - Obtain the status of general-purpose input (whole port)  
**\_8443\_set\_gpio\_output\_CH** - Set general purpose output  
**\_8443\_get\_gpio\_output\_CH** - Obtain the status of general purpose output  
**\_8443\_get\_gpio\_input\_CH** - Obtain the status of general purpose input

### @Function Description

**\_8443\_set\_gpio\_output:**  
 Set ON/OFF status for general purpose output signal (whole port)  
**\_8443\_get\_gpio\_output:**  
 Obtain ON/OFF status for general purpose output signal (whole port)  
**\_8443\_get\_gpio\_input:**  
 Obtain ON/OFF status for general purpose input signal (whole port)  
**\_8443\_set\_gpio\_output\_CH:**  
 Set ON/OFF status for general purpose output signal (bit specified)  
**\_8443\_get\_gpio\_output\_CH:**  
 Obtain ON/OFF status for general purpose output signal (bit specified)  
**\_8443\_get\_gpio\_input\_CH:**  
 Obtain ON/OFF status for general purpose input signal (bit specified)

### @Syntax

#### C/C++ (Windows XP/7/8)

```
I16 FNTYPE _8443_set_gpio_output(I16 CardNo, U16 DoValue);
I16 FNTYPE _8443_get_gpio_output(I16 CardNo, U16 *DoValue);
I16 FNTYPE _8443_get_gpio_input(I16 CardNo, U16 *DiValue);
I16 FNTYPE _8443_set_gpio_output_CH(I16 CardNo, U16 Channel, U16 Value);
I16 FNTYPE _8443_get_gpio_output_CH(I16 CardNo, U16 Channel, U16 *Value);
I16 FNTYPE _8443_get_gpio_input_CH(I16 CardNo, U16 Channel, U16 *Value);
```

#### VB.NET (Windows XP/7/8)

```
B _8443_set_gpio_output(ByVal CardNo As Short, ByVal DoValue As Integer) As Short
B _8443_get_gpio_output(ByVal CardNo As Short, ByRef DoValue As Integer) As Short
B _8443_get_gpio_input(ByVal CardNo As Short, ByRef DiValue As Integer) As Short
B _8443_set_gpio_output_CH(ByVal CardNo As Short, ByVal Channel As Short, ByVal Value As Integer) As Short
B _8443_get_gpio_output_CH(ByVal CardNo As Short, ByVal Channel As Short, ByRef Value As Integer) As Short
B _8443_get_gpio_input_CH(ByVal CardNo As Short, ByVal Channel As Short, ByRef Value As Integer) As Short
```

#### C# (Windows XP/7/8)

```
Int16 _8443_set_gpio_output(Int16 CardNo, UInt16 DoValue);
Int16 _8443_get_gpio_output(Int16 CardNo, ref UInt16 DoValue);
Int16 _8443_get_gpio_input(Int16 CardNo, ref UInt16 DiValue);
Int16 _8443_set_gpio_output_CH(Int16 CardNo, Int16 Channel, UInt16 Value);
Int16 _8443_get_gpio_output_CH(Int16 CardNo, Int16 Channel, ref UInt16 Value);
Int16 _8443_get_gpio_input_CH(Int16 CardNo, Int16 Channel, ref UInt16 Value);
```

**@Argument**

**CardNo:** Board number (0 starts)

**Channel:** Bit number (0 ~ 15)

**DoValue:** All output value (whole port)

**DiValue:** All input value (whole port)

**Value:** ON/OFF value for specified bit (0 or 1)

**@Return Code**

ERR\_NoError : 0

## 7.21. Error Code List

Return Code	Identifier	Description
0	ERR_NoError	No error
1	ERR_BoardNoInit	Initialization incomplete
2	ERR_InvalidBoardNumber	Reserved
3	ERR_InitializedBoardNumber	Reserved
4	ERR_BaseAddressError	Reserved
5	ERR_BaseAddressConflict	Reserved
6	ERR_DuplicateBoardSetting	Reserved
7	ERR_DuplicateIrqSetting	Reserved
8	ERR_PCIBiosNotExist	Board not found
9	ERR_PCIIrqNotExist	Reserved
10	ERR_PCICardNotExist	Reserved
11	ERR_SpeedError	Speed specification error (MaxVel: 0, etc.)
12	ERR_MoveRatioError	move_ratio setting error
13	ERR_PosOutOfRange	Feedback counter setting value error (set with -134217728 ~ 134217727 )
14	ERR_AxisAlreadyStop	The axis is already stopped.
15	ERR_AxisArrayError	Reserved
16	ERR_SlowDownPointError	Ramping down point value is invalid
17	ERR_CompareMethodError	Comparator method specification error
18	ERR_CompareNoError	Comparator value error
19	ERR_CompareAxisError	Comparator axis specification error
20	ERR_CompareTableSizeError	Comparator table size is out of range
21	ERR_CompareFunctionError	Comparator function data setting error
22	ERR_CompareTableNotReady	Comparator table RAM error
23	ERR_CompareLineNotReady	Comparator function RAM error
24	ERR_NoCardFound	PPCLe-8443 board not detected
25	ERR_LatchNoError	Latch counter specification error
26	ERR_AxisRangeError	Axis number specification error
27	ERR_DioNoError	General purpose output port number specification error
28	ERR_PChangeSlowDownPointError	Position override error (The specified position is closer than the ramp down point.)
29	ERR_SpeedChangeError	Speed change specification value error
30	ERR_CardNoError	Board number specification error
31	ERR_LinkIntError	LinkINT error (thread for interrupt is not created yet)
32	ERR_HardwareCompareAxisWrong	Comparator function axis number specification error
33	ERR_AutoCompareSourceWrong	Comparator function comparison counter specification error
34	ERR_CompareDeviceTypeError	Comparator function device specification error
35	ERR_PulserHomeTypeError	Pulser home return type selection error

Return Code	Identifier	Description
36	ERR_EventAlreadyEnable	int_enable error (event is already created)
37	ERR_EventNotEnableYet	Event setting error (event disabled)
38	ERR_LineArcParameterError	Reserved
39	ERR_ConfigFileOpenError	Configuration file cannot be opened
40	ERR_CompareFIFONotReady	Comparator FIFO initialization incomplete error
41	ERR_EventInitError	Thread initialization error
42	ERR_MemAllocError	Memory allocation error
43	ERR_FIFOSourceERROR	Comparator function FIFO source specification error
44	ERR_OtherProcessExist	Error; other process is being executed
45	ERR_DelayTimeError	Delay time setting error
46	ERR_DelayDistError	Delay distance setting error
47	ERR_FIFOModeOn	FIFO mode error
48	ERR_FIFOBusy	Reserved
49	ERR_OpenDriverFailed	Driver open error
50	ERR_OSVersionError	OS version error
51	ERR_OwnerSet	Reserved
52	ERR_SignalHandle	Reserved
53	ERR_SignalNotify	Reserved
54	ERR_AllocateMemory	Reserved
55	ERR_VChangeTimeError	Velocity override time setting error
56	ERR_EventInvalid	Reserved
57	ERR_ErrorIntCome	Reserved
58	ERR_Unknown	Reserved
59	ERR_WaitAbandoned	Reserved
60	ERR_WaitDelayTimeOut	Reserved
61	ERR_NoSeqAttached	Reserved
62	ERR_CardTypeWrong	Board type error
63	ERR_RotarySourceWrong	Reserved
64	ERR_PXISourceWrong	Reserved
65	ERR_PXIChannelWrong	Reserved
66	ERR_PulseModeError	Reserved
67	ERR_EventMapRangeError	Reserved
68	ERR_EventTypeError	Reserved
69	ERR_AAModeWrong	Reserved
70	ERR_MotionBusy	Still operating
71	ERR_ArraySizeTooBig	Reserved
72	ERR_UserCodeWrite	Reserved
73	ERR_SecurityCode	Security code error
74	ERR_CompareDataNotReady	Compared data is not ready

Return Code	Identifier	Description
75	ERR_ParameterError	Invalid parameter
76	ERR_PitchCompensationWrong	Reserved
77	ERR_CanNotPitchCompensation	Reserved
78	ERR_CanNotUseInPitchCompMode	Reserved
79	ERR_PitchCompNotEnable	Reserved
80	ERR_DriverVersionError	Wrong driver version

(Note) Errors in “Reserved” are for manufacturer maintenance purpose and basically do not occur.

## Revision

Revision	Date	Contents
6 th	Oct 27, 2017	New document.
7 th	June 5, 2018	<p>1-1.Features  1-3.Software supporting  5. PPCle-8443 utility  - Changed supported OS  Added Windows10  Deleted WindowsXP  2.3.5 Precautions (Sleep function)  Added the section</p>
8 th	September 14, 2020	<p>5.1.1 Output pulse mod  Corrected Figure  Form changed. According to form changed, Chapter No. shifted from 1 to 2, etc.  5.7.1 Comparator of PPCle-8443  - Comparator source of Comparator 5  Corrected from "Feedback position counter" → "Any counters"  5.7.2 Position Comparator  - Deleted "whose comparidon source is the feedback position counter"  - Corrected from "<b>_8443_set_trigger_comparator(AxisNo, Method, Data)</b>"  The second parameter "Method" indicates the comparison method, while the third "Data" is for value to be compared. In continuous comparison, this data will be ignored automatically since the comparison data will be created by other functions."  → "<b>_8443_set_trigger_comparator(AxisNo, CmpSrc, Method, Data)</b>"  The third parameter "Method" indicates the comparison method, while the fourth "Data" is for value to be compared. In continuous comparison, this data will be ignored automatically since the comparison data will be created by other functions."</p>
9 th	November 9, 2020	<p>7.3 Initialization  Corrected the C# description example  from "Int16_8443_config_from_file(ref Char file_name)"  to "Int16_8443_config_from_file(string file_name)".  7.9 Helical Interpolation Operation  Added the followings to the function description.  The speed specified by this function is a speed of circular interpolation.</p> <p style="text-align: right;">TA600061-EN0/1</p>
10th	-	-
		TA600061-EN0/2



Revision	Date	Contents
11 th	March 3, 2023	<p>1.2.3.1 Precautions for transportation and installation Additional notes for installation to a PC</p> <p>1.3 Warranty Description is revised</p> <p>3.3.5 Precautions for use (sleep function and fast startup function) Sleep function → Sleep function and fast startup function</p> <p>7.18 Continuous Operation  <b>“_8443_check_continuous_buffer():”</b>  This function is used to detect if the command pre-register is empty. When the command pre-register is empty, you can write a next operation command. The next command will be overwritten in the 2nd pre-register.  →  This function is used to detect if the command pre-register is empty. When the command pre-register is empty, you can write a next operation command. If the command pre-register is NOT empty, you cannot write the command.</p> <p style="text-align: right;">TA600061-EN0/3</p>



[www.pulsemotor.com/global](http://www.pulsemotor.com/global)

Information

[www.pulsemotor.com/global/contact](http://www.pulsemotor.com/global/contact)

Eleventh edition issued in March 2023  
Copyright 2017 Nippon Pulse Motor Co., Ltd.

---