APS_FunctionLibrary_V2.0

Build Date: 2.3.2021

Support Products:

DPAC-1000 DPAC-3000

PCI-8392(H)

PCI-8253/56

PCI-8144

AMP-104C

PCI(e)-7856

MNET-4XMO MNET-4XMO-(C)

MNET-1XMO

HSL-4XMO HSL-DIO

PCI-8102/PCI-C154(+)

PCI-8154/8158 PCIe-8154/8158

EMX-100

PCI-8254/58 / AMP-204/8C

PCIe-833x

ECAT-4XMO ECAT-TRG4

Contents

Introduction

APS means “Automation Product Software”. APS library provides users a uniform interface to access all of ADLINK products which support it. It can cover many automation fields especially in machine automation. The most important component in machine automation is motion control. APS library was first born with motion control which co-working components such as system platform management, field bus communication function, general digital input/output, general analog input/output and various counter/timer supports are all built-in components in APS. The APS library will be an all-in-one solution in automation field of ADLINK products.

The benefits of using this library are

A. Hardware independent
B. OS independent
C. Programming style consistent

The first benefit is hardware independent. In the past, each product has its own software function set. Every time users want to add or remove different kinds of product even for the same purpose, they must re-program their software to fit it. Most of time, they must re-study new function usage. That’s a big effort to users in development and maintenance. It’s also not easy to achieve on time development. Now, if users use APS library, they can take APS library as their middle layer of software. It is easy to re-use their own software component which is interfacing with APS without taking care different kinds of same purpose product. That’s the meaning of hardware independent.

The second benefit is OS independent. We will continuously research and develop new operating system supports. The standard package of APS

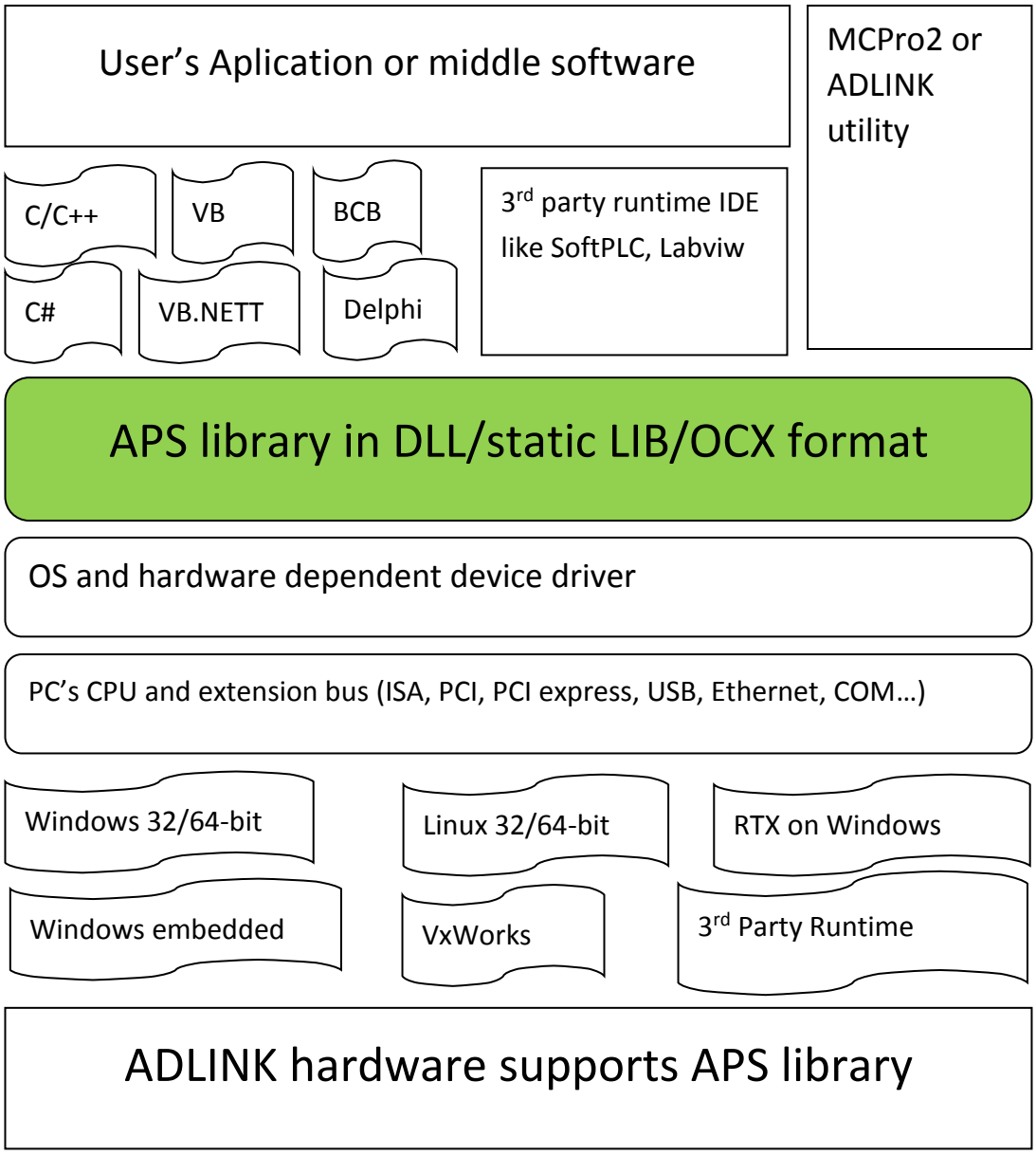
supports Microsoft Windows series like Windows XP/2000/Vista and coming new Windows OS. No matter it is 32-bit or 64-bit and no matter platform is single core or multi-cores (SMP), it guarantees all functions running in every OS identically so users don't need to worry about it. It saves much time for users to focus on their machine design. For non-Windows OS, APS also has plan to support it. It will support not only general OS like Linux, and DOS but also real-time OS like RTX, VxWorks and so on. This benefit can help users on product positioning from low-end to high-end machine.

The third benefit is programming style consistant. APS library makes different type of applications like motion control, I/O control and communication to have the same programming style. No matter the motor is stepper or servo, no matter it is distributed or centralized topology, APS library has the same style in programming and also in parameters definitions. APS library also provides various programming language interface and examples for users like ANSI C/C++, Microsoft Visual C/C++, Visual Basic, C#, Visual Basic.NET and Borland Delphi, C/C++ builder and so on. It satisfies different users and purposes on machine development. APS library also provides a visual user interface under Windows system to test all functions of product. This software is based on APS library. In other words, any product supports APS library, the utility also supports them. The utility is called "MotionCreatorPro2" or newer version. It is good to software programmer and system setup people because users don't even need to write any code before verifying the control results and hardware function. It is a good way from product testing to system development and debug.

APS library is not only a library. It is a total package ADLINK wants to provide. It includes various kinds of OS device drivers, dynamic or static link library, many kinds of programming language interface, visualization utility, version control information, rich document, long time support and one-step installation software. It supports most of ADLINK automation products

especially in machine control field. By using this library, users can reduce development time and no worry about PC's CPU and operating system changes.

The following diagram is about APS library's position.



1. Programming Library

APS supports many kinds of programming language. The header file of APS library contents function declarations, type definitions and constant varaiable definitions. The following is the example of C/C++ library. Others please refer to installed header file of corresponding languages.

The function prototype and some common data type are declared in **APS168.h**. We suggest you to use these data types in your application programs for compatibility. The following table shows the data type's name and the numeric range.

Type Name	C/C++ Data types	Description	Range
U8	unsigned char	8-bit ASCII character	0 to 255
I16	Short	16-bit signed integer	-32768 to 32767
U16	unsigned short	16-bit unsigned integer	0 to 65535
I32	long	32-bit signed long integer	-2147483648 to 2147483647
U32	unsigned long	32-bit unsigned long integer	0 to 4294967295
F32	Float	32-bit single-precision floating-point	-3.402823E38 to 3.402823E38
F64	double	64-bit double-precision floating-point	-1.797683134862315E308 to 1.797683134862315E309
Boolean	Char	Boolean logic value	TRUE, FALSE

The naming rule of APS library is full-name of purpose.

In a 'C' programming environment:

APS_{purpose_name}.

e.g. **APS_initial()**, **APS_get_position()**, **APS_relative_move()**

2. List of all functions

All functions List

Sec.	Function name	Descriptions
3	**System & Initialization**	
	APS_initial	Device initialization
	APS_close	Device close
	APS_version	Get the version of the library
	APS_device_driver_version	Get the driver's version of devices
	APS_get_axis_info	Get the information of the specified axis
	APS_get_card_name	Get card index
	APS_disable_device	Disable cards
	APS_set_board_param	Set board parameter
	APS_get_board_param	Get board parameter
	APS_set_axis_param	Set axis parameter
	APS_get_axis_param	Get axis parameter
	APS_set_axis_param_f	Set axis parameter by double
	APS_get_axis_param_f	Get axis parameter by double
	APS_get_system_timer	Get system timer counter
	APS_get_device_info	Get device information
	APS_save_parameter_to_flash	Save system & axes parameters to flash
	APS_load_parameter_from_flash	Load system & axes parameters from flash
	APS_load_parameter_from_default	Load system & axes parameters by default value.
	APS_set_security_key	Set security password
	APS_check_security_key	Varify security password
	APS_reset_security_key	Reset security password

	APS_get_first_axisId	Get first axis id of the card
	APS_save_param_to_file	Save parameters to file
	APS_load_param_from_file	Load parameters from file
	APS_register_emx	Register EMX in library
	APS_get_deviceIP	Get Device IP
	APS_reset_emx_alarm	Reset the alarm signal of device
	APS_get_curr_sys_ctrl_mode	Get current system control mode
4	**SSCNET function**	
	APS_start_sscnet	Start the network of SSCNET
	APS_stop_sscnet	Stop the network of SSCNET
	APS_get_sscnet_servo_param	Read current servo parameter value
	APS_set_sscnet_servo_param	Set servo parameter
	APS_get_sscnet_servo_alarm	Get current servo alarm information
	APS_reset_sscnet_servo_alarm	Servo alarm reset
	APS_save_sscnet_servo_param	Save servo parameter to flash ROM
	APS_get_sscnet_servo_abs_position	Get absolute reference position from servo driver
	APS_save_sscnet_servo_abs_position	Save absolute reference position to flash ROM
	APS_load_sscnet_servo_abs_position	Load absolute reference position from flash ROM
	APS_get_sscnet_link_status	Get SSCNET link status
	APS_set_sscnet_servo_monitor_src	Set servo monitor data source
	APS_get_sscnet_servo_monitor_src	Get servo monitor data source
	APS_get_sscnet_servo_monitor_data	Get servo monitor data
5	**Motion IO and motion status**	
	APS_motion_status	Return motion status
	APS_motion_io_status	Return motion IO status
	APS_set_servo_on	Set servo ON/OFF

	APS_get_position	Get feedback position
	APS_set_position	Set feedback position
	APS_get_command	Get command position
	APS_set_command	Set command position
	APS_get_command_velocity	Get command velocity
	APS_get_feedback_velocity	Get feedback velocity
	APS_get_error_position	Get error position
	APS_get_target_position	Get target position
	APS_get_position_f	Get feedback position by double
	APS_set_position_f	Set feedback position by double
	APS_get_command_f	Get command position by double
	APS_set_command_f	Set command position by double
	APS_get_error_position_f	Get error position by double
	APS_get_target_position_f	Get target position by double
	APS_get_command_velocity_f	Get command velocity by double
	APS_get_feedback_velocity_f	Get feedback velocity by double
	APS_get_mq_free_space	Get free space of motion queue
	APS_get_mq_usage	Get usage of motion queue
	APS_get_stop_code	Get stop code
	APS_get_encoder	Get raw feedback counter
	APS_get_command_counter	Get raw command counter
	APS_reset_command_counter	Reset raw command counter
	APS_get_actual_torque	Get actual torque value
	APS_get_axis_latch_data	Get ORG/EZ latch data
6	**Single axis motion**	
	APS_relative_move	Begin a relative distance move
	APS_absolute_move	Begin a absolute position move
	APS_velocity_move	Begin a velocity move

	APS_home_move	Begin a home move
	APS_stop_move	Stop move
	APS_emg_stop	Emergency stop
	APS_relative_move2	Begin a relative distance move with speed profile
	APS_absolute_move2	Begin a absolute position move with speed profile
	APS_home_move2	Begin a home move with speed profile
	APS_speed_override	Change speed on the fly
	APS_relative_move_ovrd	Begin a relative distance move or override it with new distance and speed
	APS_absolute_move_ovrd	Begin an absolute position move or override it with new position and speed
	APS_home_escape	Leave home switch
7	**Multi-axes move trigger & stop**	
	APS_move_trigger	Send a trigger to sync all waiting moves
	APS_stop_move_multi	Multi-axes stop move
	APS_emg_stop_multi	Multi-axes emg stop move
8	**Jog move**	
	APS_set_jog_param	Set Jog parameters
	APS_get_jog_param	Get Jog parameters
	APS_jog_mode_switch	Enable / Disable jog move
	APS_jog_start	Start / stop jog move
9	**Interpolation**	
	APS_absolute_linear_move	Begin an absolute position linear interpolation
	APS_relative_linear_move	Begin a relative distance linear interpolation
	APS_absolute_arc_move	Begin an absolute position circular interpolation
	APS_relative_arc_move	Begin a relative distance circular

		interpolation
	APS_absolute_arc_move_3pe	Begin a absolute position circular interpolation by pass and end point method
	APS_relative_arc_move_3pe	Begin a relative distance circular interpolation by pass and end point method
	APS_absolute_helix_move	Begin an absolute position helical interpolation
	APS_relative_helix_move	Begin a relative distance helical intepolation
	APS_absolute_helical_move	Begin an absolute position helical interpolation
	APS_relative_helical_move	Begin a relative distance helical interpolation
10	**Advanced single move & interpolation**	
	APS_ptp	Begin a single move
	APS_ptp_v	Begin a single move with Vm profile
	APS_ptp_all	Begin a single move with all profile
	APS_vel	Begin a velocity move
	APS_vel_all	Begin a velocity move with all profile
	APS_line	Begin a line move
	APS_line_v	Begin a line move with Vm profile
	APS_line_all	Begin a line move with all profile
	APS_arc2_ca	Begin an Arc2 move of angle type
	APS_arc2_ca_v	Begin an Arc2 move of angle type with Vm profile
	APS_arc2_ca_all	Begin an Arc2 move of angle type with all profile
	APS_arc2_ca	Begin an Arc2 move of end position
	APS_arc2_ca_v	Begin an Arc2 move of end position with Vm profile
	APS_arc2_ca_all	Begin an Arc2 move of end position with all profile

	APS_arc2_ce	Begin an Arc2 move of end position
	APS_arc2_ce_v	Begin an Arc2 move of end position with Vm profile
	APS_arc2_ce_all	Begin an Arc2 move of end position with all profile
	APS_arc3_ca	Begin an Arc3 move of angle type
	APS_arc3_ca_v	Begin an Arc3 move of angle type with Vm profile
	APS_arc3_ca_all	Begin an Arc3 move of angle type with all profile
	APS_arc3_ce	Begin an Arc3 move of end position
	APS_arc3_ce_v	Begin an Arc3 move of end position with Vm profile
	APS_arc3_ce_all	Begin an Arc3 move of end position with all profile
	APS_spiral_ca	Begin a 3D spiral-helix move of angle type
	APS_spiral_ca_v	Begin a 3D spiral-helix move of angle type with Vm profile
	APS_spiral_ca_all	Begin a 3D spiral-helix move of angle type with all profile
	APS_spiral_ce	Begin a 3D spiral-helix move of end position
	APS_spiral_ce_v	Begin a 3D spiral-helix move of end position with Vm profile
	APS_spiral_ce_all	Begin a 3D spiral-helix move of end position with all profile
11	**Interrupt**	
	APS_int_enable	Interrupt main switch
	APS_set_int_factor	Enable/Disable interrupt factor and get interrupt handle.
	APS_get_int_factor	Get interrupt factor enable or disable
	APS_wait_single_int	Wait single interrupt event

	APS_wait_multiple_int	Wait multiple interrupt events
	APS_wait_error_int	Wait error interrupts(Non-mask)
	APS_reset_int	Reset interrupt event to non-signaled state.
	APS_set_int	Set interrupt event to signaled state.
	APS_set_int_factorH	Enable/Disable interrupt factor and get interrupt handle.(Win32)
	APS_int_no_to_handle	Convert interrupt event number to interrupt handle.(Win32)
	APS_set_field_bus_int_factor_motion	Enable/Disable motion interrupt factor and get interrupt handle for MotionNet series.
	APS_get_field_bus_int_factor_motion	Get motion interrupt factor enable or disable for MotionNet series.
	APS_set_field_bus_int_factor_error	Enable/Disable error interrupt factor and get interrupt handle for MotionNet series.
	APS_get_field_bus_int_factor_error	Get error interrupt factor status for MotionNet series.
	APS_reset_field_bus_int_motion	Reset interrupt status of axes for MotionNet series.
	APS_wait_field_bus_error_int_motion	Wait error interrupt event for MotionNet series.
	APS_set_field_bus_int_factor_di	Assign DI interrupt bits and get interrupt handle for HSL series.
	APS_get_field_bus_int_factor_di	Get DI interrupt bits assigned for HSL series.
12	**Sampling**	
	APS_set_sampling_param	Set sampling parameter.
	APS_get_sampling_param	Get sampling parameter.
	APS_wait_trigger_sampling	Waiting for sample data.
	APS_wait_trigger_sampling_async	Waiting for sample data asynchronously
	APS_get_sampling_count	Get sampled data count.
	APS_stop_wait_sampling	Force stop wait sampling

	APS_auto_sampling	Start/Stop auto sampling
	APS_get_sampling_data	Get sampling data in auto sampling mode by 4 Channels.
	APS_set_sampling_param_ex	Set sampling parameter by structure. It is an extension to 8 channels.
	APS_get_sampling_param_ex	Get sampling parameter by structure. It is an extension to 8 channels.
	APS_wait_trigger_sampling_ex	Waiting for sample data. It is an extension to 8 channels.
	APS_wait_trigger_sampling_async_ex	Waiting for sample data asynchronously. It is an extension to 8 channels.
	APS_get_sampling_data_ex	Get sampling data in auto sampling mode. It is an extension to 8 channels.
13	DIO & AIO	
	APS_set_field_bus_d_channel_output	Set field bus digital output by channel
	APS_get_field_bus_d_channel_output	Get field bus digital output by channel
	APS_get_field_bus_d_channel_input	Get field bus digital input by channel
	APS_set_field_bus_d_port_output	Set field bus digital output by port
	APS_get_field_bus_d_port_input	Get field bus digital input by port
	APS_get_field_bus_d_port_output	Get field bus digital output by port
	APS_write_d_output	Set digital output value
	APS_read_d_output	Read digital output value
	APS_read_d_input	Read digital input value
	APS_write_d_channel_output	Set digital output value by channel
	APS_read_d_channel_output	Read digital output value by channel
	APS_read_d_channel_input	Read digital input value by channel
	APS_read_a_input_value	Read back analog input value by volt
	APS_read_a_input_data	Read back analog input raw data
	APS_write_a_output_value	Set analog output value by volt
	APS_write_a_output_data	Set analog output value by raw data

14	Point table motion	
	APS_set_point_table	Set point table move parameters
	APS_get_point_table	Get point table move parameters
	APS_point_table_move	Start a point table move
	APS_get_running_point_index	Get current point move index when axis is perform a point move
	APS_get_start_point_index	Get the first point move index when axis is perform a point move
	APS_get_end_point_index	Get the last point move index when axis is perform a point move
	APS_set_table_move_pause	Pause point table move
	APS_set_table_move_ex_pause	Decelerate to stop move and control I/O
	APS_set_table_move_ex_rollback	Rollback to starting position of current point index
	APS_set_table_move_ex_resume	Re-start point table move and keep I/O status
	APS_set_table_move_repeat	Set point table move repeat
	APS_set_point_table_mode2	Set point table mode
	APS_set_point_table2	Set point table
	APS_point_table_continuous_move2	Start a point table continuous move
	APS_point_table_single_move2	Start a point table single move
	APS_get_running_point_index2	Get current point move index when axis is perform a point move
	APS_point_table_status2	Get point table stauts
	APS_set_point_table3	Set point table
	APS_point_table_move3	Start a point table single move
	APS_set_point_table_param3	Set speed parameter
	APS_set_feeder_group	Set axes into a feeder group
	APS_get_feeder_group	Return the configuration in one feeder group
	APS_free_feeder_group	Free a feeder group and it’s resources

	APS_reset_feeder_buffer	Reset the feeder's point buffer
	APS_set_feeder_point_2D	Add a point into feeder's buffer
	APS_set_feeder_point_2D_ex	Add a point into feeder's buffer
	APS_start_feeder_move	Start point table move and feed points.
	APS_get_feeder_status	Get feeder status
	APS_get_feeder_running_index	Get which point is in operation.
	APS_get_feeder_feed_index	Get which point is set into point table.
	APS_set_feeder_ex_pause	Motion paused(stopped) and feeder paused
	APS_set_feeder_ex_rollback	Move back to the starting position of paused index
	APS_set_feeder_ex_resume	Resume the point-table move.
	APS_set_point_table_ex	Set point table move parameters with entend option
	APS_get_point_table_ex	Get point table move parameters with entend option
15	**Advanced Point table**	
	APS_pt_enable	Enable point table.
	APS_pt_disable	Disable point table.
	APS_get_pt_info	Get information of point table.
	APS_pt_set_vs	Set configuration of Vs to point table
	APS_pt_get_vs	Get configuration of Vs in the point table
	APS_pt_start	Set control command to point table
	APS_pt_stop	Stop point table
	APS_get_pt_status	Get status of point table
	APS_reset_pt_buffer	Reset buffer of point table
	APS_pt_roll_back	Rollback to previous point
	APS_get_pt_error	Get error code of point table
	APS_pt_dwell	Push a dwell move into point buffer of point table.
	APS_pt_line	Push a line move into point buffer of point

		table.
	APS_pt_arc2_ca	Push a 2d arc move into point buffer of point table.
	APS_pt_arc2_ce	Push a 2d arc move into point buffer of point table.
	APS_pt_arc3_ca	Push a 3d arc move into point buffer of point table.
	APS_pt_arc3_ce	Push a 3d arc move into point buffer of point table.
	APS_pt_spiral_ca	Push a helical move into point buffer of point table.
	APS_pt_spiral_ce	Push a helical move into point buffer of point table.
	APS_pt_ext_set_do_ch	Set Do extension command into command buffer. Command buffer is active when pushing a move into point table.
	APS_pt_set_absolute	Set absolute profile into profile buffer.
	APS_pt_set_relative	Set relative profile into profile buffer.
	APS_pt_set_trans_buffered	Set transition to buffer mode in profile buffer.
	APS_pt_set_trans_inp	Set transition to in-position mode in profile buffer.
	APS_pt_set_trans_blend_dec	Set transition to blending mode with deceleration in profile buffer.
	APS_pt_set_trans_blend_dist	Set transition to blending mode with residue distant in profile buffer.
	APS_pt_set_trans_blend_pcnt	Set transition to blending mode with residue distant percetange in profile buffer.
	APS_pt_set_acc	Set accerlation profile into profile buffer.
	APS_pt_set_dec	Set deceleration profile into profile buffer.
	APS_pt_set_acc_dec	Set accerlation / deceleration profile into profile buffer

	APS_pt_set_s	Set S-factor profile into profile buffer.
	APS_pt_set_vm	Set maximum velocity profile into profile buffer.
	APS_pt_set_ve	Set end velocity profile into profile buffer.
16	**Field bus functions**	
	APS_set_field_bus_param	Set field bus related parameters
	APS_get_field_bus_param	Get field bus related parameters
	APS_scan_field_bus	Scan field bus and generate ENI file
	APS_start_field_bus	Start the network of specified field bus
	APS_stop_field_bus	Stop the network of specified field bus
	APS_field_bus_d_set_output	Set field bus digital output
	APS_field_bus_d_get_output	Get field bus digital output
	APS_field_bus_d_get_input	Get field bus digital input
	APS_field_bus_d_set_output_ex	Set field bus digital output for 64 bit DIO operation
	APS_field_bus_d_get_output_ex	Get field bus digital output for 64 bit DIO operation
	APS_field_bus_d_get_input_ex	Get field bus digital input for 64 bit DIO operation
	APS_set_field_bus_slave_param	Set parameter to field bus slave module
	APS_get_field_bus_slave_param	Get parameter from field bus slave module
	APS_set_field_bus_a_output	Set field bus analog output
	APS_get_field_bus_a_output	Get field bus analog output
	APS_get_field_bus_a_input	Get field bus analog input
	APS_get_slave_connect_quality	Get the connected quality of slave
	APS_get_slave_online_status	Get the online status of slave
	APS_get_field_bus_master_status	Get field bus master status
	APS_get_field_bus_last_scan_info	Get fieldbus info after system scanning.
	APS_get_field_bus_master_type	Get master type of the fieldbus

	APS_get_field_bus_slave_type	Get slave type on the fieldbus
	APS_get_field_bus_slave_name	Get slave name on the fieldbus
	APS_get_field_bus_slave_first_axisno	Get first axis of the slave module
	APS_get_field_bus_device_info	Get device information on a specified field bus
	APS_get_field_bus_module_info	Get slave information
	APS_reset_field_bus_alarm	Reset the alarm signal of slave
	APS_get_field_bus_alarm	Get alarm code of slave
	APS_get_field_bus_pdo	Get value from PDO memory
	APS_set_field_bus_pdo	Set value to PDO memory
	APS_get_field_bus_pdo_offset	Get PDO information
	APS_get_field_bus_sdo	Get SDO data from slave
	APS_set_field_bus_sdo	Set SDO data to slave
	APS_set_field_bus_od_data	Set EtherCAT OD raw data
	APS_get_field_bus_od_data	Get EtherCAT OD raw data
	APS_get_field_bus_od_module_info	Get EtherCAT slave information
	APS_get_field_bus_module_map	Get mapped slave ID in manual ID mode
	APS_set_field_bus_module_map	Set mapped slave ID in manual ID mode
	APS_get_field_bus_slave_state	Get the status of slave's state machine
	APS_set_field_bus_slave_state	Set the status of slave's state machine
	APS_get_field_bus_ESC_register	Get EtherCAT Slave Controller register
	APS_set_field_bus_ESC_register	Set EtherCAT Slave Controller register
	APS_get_system_loading	Get system loop loading
	APS_get_field_bus_analysis_topology	Get current and past topology then analysis
	APS_get_field_bus_loss_package	Get the loss of EtherCAT frame count on receive bus direction.
17	**Gear / Gantry functions**	
	APS_set_gantry_param	Set gantry function related parameter

	APS_get_gantry_param	Get gantry function related parameter
	APS_set_gantry_axis	Set two axes in a gantry group
	APS_get_gantry_axis	Get which axes in a gantry group
	APS_get_gantry_error	Get gantry axes deviation error
	APS_get_encoder	Get encoder(Be used for compensation of gantry home return)
	APS_ get_latch_event	Get latch event by axis(Be used for compensation of gantry home return)
	APS_get_latch_counter	Get latch counter by axis(Be used for compensation of gantry home return)
	APS_start_gear	Enable/Disable a specified gear mode
	APS_get_gear_status	Get gear status
	APS_get_gantry_number	Get number of this master's corresponding slaves
	APS_get_gantry_info	Get slave axis ID array
	APS_get_gantry_deviation	Get position deviation between master and slaves
18	**Compare trigger**	
	APS_set_trigger_param	Set compare trigger related parameter
	APS_get_trigger_param	Get compare trigger related parameter
	APS_set_trigger_linear	Set linear comparing function
	APS_set_trigger_table	Set table comparing function
	APS_set_trigger_manual	Manual output trigger
	APS_set_trigger_manual_s	Manual output trigger synchronously
	APS_get_trigger_table_cmp	Get current table comparing value
	APS_get_trigger_linear_cmp	Get current linear comparing value
	APS_get_trigger_count	Get triggered count.
	APS_reset_trigger_count	Reset triggered count.
	APS_enable_trigger_fifo_cmp	Enable trigger fifo comparator
	APS_get_trigger_fifo_cmp	Get trigger fifo comparator

	APS_get_trigger_fifo_status	Get trigger fifo status
	APS_set_trigger_fifo_data	Set trigger fifo status
	APS_start_timer	Start timer
	APS_get_timer_counter	Get timer count value
	APS_set_timer_counter	Set timer count value
	APS_start_trigger_timer	Start timer
	APS_get_trigger_timer_counter	Get timer count value
	APS_set_multi_trigger_table	Set table comparing function
	APS_get_multi_trigger_table_cmp	Get current table comparing value
	APS_set_trigger_table_data	Set table compator data (Fast table compare trigger function)
	APS_get_trigger_table_status	Get table comparator status (Fast table compare trigger function)
	APS_get_trigger_cmp_value	Get table comparator value (Fast table compare trigger function)
	APS_enable_trigger_table	Enable table comparator (Fast table compare trigger function)
	APS_reset_trigger_table	Reset table comparator (Fast table compare trigger function)
19	**Program download**(*1)	
	APS_load_vmc_program	Load VMC file to task memory
	APS_save_vmc_program	Save to VMC file from task memory
	APS_set_task_mode	Set task run mode
	APS_get_task_mode	Get task run mode
	APS_start_task	Start task control command
	APS_get_task_info	Get task information
	APS_get_task_msg	Get message of all tasks
20	**Manual Pulse Generator functions**	
	APS_manual_pulser_start	Enable/Disable PA/PB input
	APS_manual_pulser_velocity_move	Begin a pulser velocity move
	APS_manual_pulser_relative_move	Begin a pulser relative distance move
	APS_manual_pulser_home_move	Begin a pulser home move

	APS_get_pulser_counter	Get pluse input counter
	APS_set_pulser_counter	Set pluse input counter
21	Pitch error compensation functions	
	APS_set_pitch_table	Set configurations and data of pitch error compensation table
	APS_get_pitch_table	Get configurations and data of pitch error compensation table
	APS_start_pitch_comp	Start pitch error compensation
22	DPAC System Functions	
	APS_rescan_CF	Reset DPAC Slave CF slot
	APS_get_battery_status	Get DPAC SRAM Battery status
	APS_get_display_data	Get 7-Segment Data
	APS_set_display_data	Set 7-Segment Data
	APS_get_button_status	Get the Push Button Input Status
23	NV RAM funciton	
	APS_set_nv_ram	Set RAM data
	APS_get_nv_ram	Get RAM data
	APS_clear_nv_ram	Clear RAM data
24	Field bus Compare trigger	
	APS_set_field_bus_trigger_param	Set compare trigger related parameter
	APS_get_field_bus_trigger_param	Get compare trigger related parameter
	APS_set_field_bus_trigger_linear	Set linear comparing function
	APS_set_field_bus_trigger_table	Set table comparing function
	APS_set_field_bus_trigger_manual	Manual output trigger
	APS_set_field_bus_trigger_manual_s	Manual output trigger synchronously
	APS_get_field_bus_trigger_table_cmp	Get current table comparing value
	APS_get_field_bus_trigger_linear_cmp	Get current linear comparing value
	APS_get_field_bus_trigger_count	Get triggered count.
	APS_reset_field_bus_trigger_count	Reset triggered count.

	APS_get_field_bus_linear_cmp_remain_count	Get remaining counter of linear comparator
	APS_get_field_bus_table_cmp_remain_count	Get remaining counter of table comparator
	APS_get_field_bus_encoder	Get encoder counter
	APS_set_field_bus_encoder	Set encoder counter
26	**Watch dog timer**	
	APS_wdt_start	Start / Stop watch dog timer
	APS_wdt_get_timeout_period	Get a timeout period of watch dog timer
	APS_wdt_reset_counter	Reset counter of watch dog timer
	APS_wdt_get_counter	Get counter of watch dog timer
	APS_wdt_set_action_event	Set action event of watch dog timer
	APS_wdt_get_action_event	Get action event of watch dog timer
27	**VAO/PWM functions(Laser function)**	
	APS_set_vao_param	Set parameter to VAO table
	APS_get_vao_param	Get parameter of VAO table
	APS_set_vao_table	Set VAO table
	APS_switch_vao_table	Switch to specified VAO table
	APS_start_vao	Enable VAO output channel
	APS_get_vao_status	Get VAO status
	APS_check_vao_param	Check parameters setting of specified VAO table
	APS_set_vao_param_ex	Set table parameters via VAO structure.
	APS_get_vao_param_ex	Get table parameters via VAO structure.
	APS_set_pwm_on	Start to output PWM signal
	APS_set_pwm_width	Set pulse width to a PWM channel
	APS_set_pwm_frequency	Set pulse frequency to a PWM channel
	APS_get_pwm_width	Get pulse width from a PWM channel
	APS_get_pwm_frequency	Get pulse frequency from a PWM channel
28	**Circular limit functions**	
	APS_set_circular_limit	Set circular limit configurations

	APS_get_circular_limit	Get circular limit configurations
29	**Simultaneous move functions**	
	APS_set_relative_simultaneous_move	Setup a relative simultaneous move
	APS_set_absolute_simultaneous_move	Setup a absolute simultaneous move
	APS_start_simultaneous_move	Begin a simultaneous move
	APS_stop_simultaneous_move	Stop a simultaneous move
30	**Single-latch functions**	
	APS_manual_latch2	Manual latch for a axis
	APS_get_latch_data2	Get latch data for a axis
31	**Multi-latch functions**	
	APS_set_ltc_counter	Set latch counter
	APS_get_ltc_counter	Get latch data for a axis
	APS_set_ltc_fifo_param	Set latch parameter
	APS_get_ltc_fifo_param	Get latch parameter
	APS_manual_latch	Latch data manually and synchronously.
	APS_enable_ltc_fifo	Enable/Disable ltc fifo
	APS_reset_ltc_fifo	Reset ltc fifo
	APS_get_ltc_fifo_data	Get one latch data from fifo
	APS_get_ltc_fifo_usage	Get usage of latch fifo
	APS_get_ltc_fifo_free_space	Get free space of latch fifo
	APS_get_ltc_fifo_status	Get fifo status
	APS_get_ltc_fifo_point	Get latch point array
32	**Ring counter functions**	
	APS_set_ring_counter	Enable ring counter function
	APS_get_ring_counter	Get limitation value of ring counter
33	**Speed Profile Calculation**	
	APS_relative_move_profile	Get relative speed profile(PCI-C154)
	APS_absolute_move_profile	Get absolute speed profile(PCI-C154)
	APS_check_motion_profile_emx	Get relative speed profile(EMX-100)
34	**Backlash functions**	
	APS_set_backlash_en	Enable/Disable backlash
	APS_get_backlash_en	Check backlash is enabled / disabled
35	**2-D compensation**	

	APS_set_2d_compensation_table	Create 2D compensation table
	APS_get_2d_compensation_table	Get 2D compensation table configuration
	APS_start_2d_compensation	Start or stop 2D compensation table
	APS_absolute_linear_move_2d_compensation	2D absolute linear interpolation
	APS_get_2d_compensation_command_position	Get command and feedback position
36	**Table definition**	
	Board parameter table	
	Axis parameter table	
	Sampling parameter table	
	Sampling source table	
	Motion IO status and motion status definitions	
	Motion status definition table	
	Interrupt factor table	
	Field bus parameter table	
	Gantry parameters table	
	Trigger parameter table	
	Latch parameter table	
	Device information table	
	Field bus slave parameter table	
	DPAC display index table	
	DPAC button status table	
	SSCNET servo monitor source table	
	VAO parameter table	
	APS functions return code	

List of all functions for DPAC-1000

Sec.	Function name	Descriptions
3	**System & Initialization**	
	APS_initial	Device initialization
	APS_close	Device close
	APS_version	Get the version of the library
	APS_device_driver_version	Get the driver's version of devices
	APS_get_card_name	Get card index
	APS_set_board_param	Set board parameter
	APS_get_board_param	Get board parameter
	APS_get_device_info	Get device information
	APS_load_param_from_file	Load parameters from file
11	**Interrupt**	
	APS_int_enable	Interrupt main switch
	APS_set_int_factor	Enable/Disable interrupt factor and get interrupt handle.
	APS_get_int_factor	Get interrupt factor enable or disable
	APS_wait_single_int	Wait single interrupt event
	APS_wait_multiple_int	Wait multiple interrupt events
	APS_reset_int	Reset interrupt event to non-signaled state.
	APS_set_int	Set interrupt event to signaled state.
	APS_set_int_factorH	Enable/Disable interrupt factor and get interrupt handle.(Win32)
13	**DIO & AIO**	
	APS_write_d_output	Set digital output value
	APS_read_d_output	Read digital output value
	APS_read_d_input	Read digital input value

20	**Manual Pulse Generator functions**	
	APS_get_pulser_counter	Get pluse input counter
	APS_set_pulser_counter	Set pluse input counter
22	**DPAC System Function**	
	APS_rescan_CF	Rescan DPAC Slave CF slot
	APS_get_battery_status	Get DPAC SRAM Battery status
	APS_get_display_data	Get 7-Segment LED Data
	APS_set_display_data	Set 7-Segment LED Data
	APS_get_button_status	Get the Push Button Input Status
23	**NV RAM funciton**	
	APS_set_nv_ram	Set RAM data
	APS_get_nv_ram	Get RAM data
	APS_clear_nv_ram	Clear RAM data
36	**Table definition**	
	Board parameter table	
	Interrupt factor table	
	Device information table	
	DPAC display index table	
	DPAC button status table	
	APS functions return code	

List of all functions for DPAC-3000

Sec.	Function name	Descriptions
3	**System & Initialization**	
	APS_initial	Device initialization
	APS_close	Device close
	APS_version	Get the version of the library
	APS_device_driver_version	Get the driver's version of devices
	APS_get_card_name	Get card index
	APS_get_axis_info	Get the information of the specified axis
	APS_set_board_param	Set board parameter
	APS_get_board_param	Get board parameter
	APS_get_device_info	Get device information
	APS_load_param_from_file	Load parameters from file
11	**Interrupt**	
	APS_int_enable	Interrupt main switch
	APS_set_int_factor	Enable/Disable interrupt factor and get interrupt handle.
	APS_get_int_factor	Get interrupt factor enable or disable
	APS_wait_single_int	Wait single interrupt event
	APS_wait_multiple_int	Wait multiple interrupt events
	APS_reset_int	Reset interrupt event to non-signaled state.
	APS_set_int	Set interrupt event to signaled state.
	APS_set_int_factorH	Enable/Disable interrupt factor and get interrupt handle.(Win32)
13	**DIO & AIO**	
	APS_write_d_output	Set digital output value

	APS_read_d_output	Read digital output value
	APS_read_d_input	Read digital input value
16	**Field bus functions**	
	APS_set_field_bus_param	Set field bus related parameters
	APS_get_field_bus_param	Get field bus related parameters
	APS_start_field_bus	Start the network of specified field bus
	APS_stop_field_bus	Stop the network of specified field bus
	APS_field_bus_d_set_output	Set field bus digital output
	APS_field_bus_d_get_output	Get field bus digital output
	APS_field_bus_d_get_input	Get field bus digital input
	APS_set_field_bus_slave_param	Set parameter to field bus slave module
	APS_get_field_bus_slave_param	Get parameter from field bus slave module
	APS_set_field_bus_a_output	Set field bus analog output
	APS_get_field_bus_a_output	Get field bus analog output
	APS_get_field_bus_a_input	Get field bus analog input
	APS_get_slave_connect_quality	Get the connected quality of slave
	APS_get_slave_online_status	Get the online status of slave
	APS_get_field_bus_last_scan_info	Get fieldbus info after system scanning.
	APS_get_field_bus_master_type	Get master type of the fieldbus
	APS_get_field_bus_slave_type	Get slave type on the fieldbus
	APS_get_field_bus_slave_name	Get slave name on the fieldbus
	APS_get_field_bus_slave_first_axisno	Get first axis of the slave module
	APS_get_field_bus_device_info	Get device information on a specified field bus
20	**Manual Pulse Generator functions**	
	APS_get_pulser_counter	Get pluse input counter
	APS_set_pulser_counter	Set pluse input counter
22	**DPAC System Function**	
	APS_rescan_CF	Rescan DPAC Slave CF slot

	APS_get_battery_status	Get DPAC SRAM Battery status
	APS_get_display_data	Get 7-Segment LED Data
	APS_set_display_data	Set 7-Segment LED Data
	APS_get_button_status	Get the Push Button Input Status
23	**NV RAM funciton**	
	APS_set_nv_ram	Set RAM data
	APS_get_nv_ram	Get RAM data
	APS_clear_nv_ram	Clear RAM data
36	**Table definition**	
	Board parameter table	
	Field bus parameter table	
	Interrupt factor table	
	Device information table	
	DPAC display index table	
	DPAC button status table	
	APS functions return code	

List of all functions for PCI-8392(H)

Sec.	Function name	Descriptions
3	**System & Initialization**	
	APS_initial	Device initialization
	APS_close	Device close
	APS_version	Get the version of the library
	APS_device_driver_version	Get the driver's version of devices
	APS_get_axis_info	Get the information of the specified axis
	APS_get_card_name	Get card index
	APS_set_board_param	Set board parameter
	APS_get_board_param	Get board parameter
	APS_set_axis_param	Set axis parameter
	APS_get_axis_param	Get axis parameter
	APS_get_system_timer	Get system timer counter
	APS_get_device_info	Get device information
	APS_save_parameter_to_flash	Save system & axes parameters to flash
	APS_load_parameter_from_flash	Load system & axes parameters from flash
	APS_load_parameter_from_default	Load system & axes parameters by default value.
	APS_load_param_from_file	Load parameters from file
4	**SSCNET function**	
	APS_start_sscnet	Start the network of SSCNET
	APS_stop_sscnet	Stop the network of SSCNET
	APS_get_sscnet_servo_param	Read current servo parameter value
	APS_set_sscnet_servo_param	Set servo parameter
	APS_get_sscnet_servo_alarm	Get current servo alarm information

	APS_reset_sscnet_servo_alarm	Servo alarm reset
	APS_save_sscnet_servo_param	Save servo parameter to flash ROM
	APS_get_sscnet_servo_abs_position	Get absolute reference position from servo driver
	APS_save_sscnet_servo_abs_position	Save absolute reference position to flash ROM
	APS_load_sscnet_servo_abs_position	Load absolute reference position from flash ROM
	APS_get_sscnet_link_status	Get SSCNET link status
	APS_set_sscnet_servo_monitor_src	Set servo monitor data source
	APS_get_sscnet_servo_monitor_src	Get servo monitor data source
	APS_get_sscnet_servo_monitor_data	Get servo monitor data
5	**Motion IO and motion status**	
	APS_motion_status	Return motion status
	APS_motion_io_status	Return motion IO status
	APS_set_servo_on	Set servo ON/OFF
	APS_get_position	Get feedback position
	APS_set_position	Set feedback position
	APS_get_command	Get command position
	APS_set_command	Set command position
	APS_get_command_velocity	Get command velocity
	APS_get_feedback_velocity	Get feedback velocity
	APS_get_error_position	Get error position
	APS_get_target_position	Get target position
6	**Single axis motion**	
	APS_relative_move	Begin a relative distance move
	APS_absolute_move	Begin a absolute position move
	APS_velocity_move	Begin a velocity move
	APS_home_move	Begin a home move

	APS_stop_move	Stop move
	APS_emg_stop	Emergency stop
	APS_relative_move2	Begin a relative distance move with speed profile
	APS_absolute_move2	Begin a absolute position move with speed profile
	APS_home_move2	Begin a home move with speed profile
8	**Jog move**	
	APS_set_jog_param	Set Jog parameters
	APS_get_jog_param	Get Jog parameters
	APS_jog_mode_switch	Enable / Disable jog move
	APS_jog_start	Start / stop jog move
9	**Interpolation**	
	APS_absolute_linear_move	Begin an absolute position linear interpolation
	APS_relative_linear_move	Begin a relative distance linear interpolation
	APS_absolute_arc_move	Begin an absolute position circular interpolation
	APS_relative_arc_move	Begin a relative distance circular interpolation
11	**Interrupt**	
	APS_int_enable	Interrupt main switch
	APS_set_int_factor	Enable/Disable interrupt factor and get interrupt handle.
	APS_get_int_factor	Get interrupt factor enable or disable
	APS_wait_single_int	Wait single interrupt event
	APS_wait_multiple_int	Wait multiple interrupt events
	APS_reset_int	Reset interrupt event to non-signaled state.
	APS_set_int	Set interrupt event to signaled state.
	APS_set_int_factorH	Enable/Disable interrupt factor and get

		interrupt handle.(Win32)
12	Sampling	
	APS_set_sampling_param	Set sampling parameter.
	APS_get_sampling_param	Get sampling parameter.
	APS_wait_trigger_sampling	Waiting for sample data.
	APS_wait_trigger_sampling_async	Waiting for sample data asynchronously
	APS_get_sampling_count	Get sampled data count.
	APS_stop_wait_sampling	Force stop wait sampling
14	Point table motion	
	APS_set_point_table	Set point table move parameters
	APS_get_point_table	Get point table move parameters
	APS_set_point_table_ex	Set point table move parameters with entend option
	APS_get_point_table_ex	Get point table move parameters with entend option
	APS_point_table_move	Start a point table move
	APS_get_running_point_index	Get current point move index when axis is perform a point move
	APS_get_start_point_index	Get the first point move index when axis is perform a point move
	APS_get_end_point_index	Get the last point move index when axis is perform a point move
	APS_set_table_move_pause	Pause point table move
	APS_set_table_move_repeat	Set point table move repeat
16	Field bus functions	
	APS_set_field_bus_param	Set field bus related parameters
	APS_get_field_bus_param	Get field bus related parameters
	APS_start_field_bus	Start the network of specified field bus
	APS_stop_field_bus	Stop the network of specified field bus
	APS_field_bus_d_set_output	Set field bus digital output

	APS_field_bus_d_get_output	Get field bus digital output
	APS_field_bus_d_get_input	Get field bus digital input
	APS_set_field_bus_slave_param	Set parameter to field bus slave module
	APS_get_field_bus_slave_param	Get parameter from field bus slave module
	APS_set_field_bus_a_output	Set field bus analog output
	APS_get_field_bus_a_output	Get field bus analog output
	APS_get_field_bus_a_input	Get field bus analog input
	APS_get_slave_connect_quality	Get the connected quality of slave
	APS_get_slave_online_status	Get the online status of slave
	APS_get_field_bus_last_scan_info	Get fieldbus info after system scanning.
	APS_get_field_bus_master_type	Get master type of the fieldbus
	APS_get_field_bus_slave_type	Get slave type on the fieldbus
	APS_get_field_bus_slave_name	Get slave name on the fieldbus
	APS_get_field_bus_slave_first_axisno	Get first axis of the slave module
	APS_get_field_bus_device_info	Get device information on a specified field bus
17	**Gear / Gantry functions**	
	APS_set_gantry_param	Set gantry function related parameter
	APS_get_gantry_param	Get gantry function related parameter
	APS_set_gantry_axis	Set two axes in a gantry group
	APS_get_gantry_axis	Get which axes in a gantry group
	APS_get_gantry_error	Get gantry axes deviation error
36	**Table definition**	
	Board parameter table	
	Axis parameter table	
	Sampling parameter table	
	Sampling source table	
	Motion IO status and motion status definitions	
	Motion status definition table	

	Interrupt factor table
	Field bus parameter table
	Gantry parameters table
	Device information table
	Field bus slave parameter table
	SSCNET servo monitor source table
	APS functions return code

List of all functions for PCI-8253/56

Sec.	Function name	Descriptions
3	**System & Initialization**	
	APS_initial	Device initialization
	APS_close	Device close
	APS_version	Get the version of the library
	APS_device_driver_version	Get the driver's version of devices
	APS_get_axis_info	Get the information of the specified axis
	APS_get_card_name	Get card index
	APS_set_board_param	Set board parameter
	APS_get_board_param	Get board parameter
	APS_set_axis_param	Set axis parameter
	APS_get_axis_param	Get axis parameter
	APS_get_system_timer	Get system timer counter
	APS_get_device_info	Get device information
	APS_save_parameter_to_flash	Save system & axes parameters to flash
	APS_load_parameter_from_flash	Load system & axes parameters from flash
	APS_load_parameter_from_default	Load system & axes parameters by default value.
	APS_load_param_from_file	Load parameters from file
5	**Motion IO and motion status**	
	APS_motion_status	Return motion status
	APS_motion_io_status	Return motion IO status
	APS_set_servo_on	Set servo ON/OFF
	APS_get_position	Get feedback position
	APS_set_position	Set feedback position

	APS_get_command	Get command position
	APS_set_command	Set command position
	APS_get_command_velocity	Get command velocity
	APS_get_feedback_velocity	Get feedback velocity
	APS_get_error_position	Get error position
	APS_get_target_position	Get target position
6	**Single axis motion**	
	APS_relative_move	Begin a relative distance move
	APS_absolute_move	Begin a absolute position move
	APS_velocity_move	Begin a velocity move
	APS_home_move	Begin a home move
	APS_stop_move	Stop move
	APS_emg_stop	Emergency stop
	APS_relative_move2	Begin a relative distance move with speed profile
	APS_absolute_move2	Begin a absolute position move with speed profile
	APS_home_move2	Begin a home move with speed profile
8	**Jog move**	
	APS_set_jog_param	Set Jog parameters
	APS_get_jog_param	Get Jog parameters
	APS_jog_mode_switch	Enable / Disable jog move
	APS_jog_start	Start / stop jog move
9	**Interpolation**	
	APS_absolute_linear_move	Begin an absolute position linear interpolation
	APS_relative_linear_move	Begin a relative distance linear interpolation
	APS_absolute_arc_move	Begin an absolute position circular interpolation

	APS_relative_arc_move	Begin a relative distance circular interpolation
	APS_absolute_arc_move_3pe	Begin a absolute position circular interpolation by pass and end point method
	APS_relative_arc_move_3pe	Begin a relative distance circular interpolation by pass and end point method
	APS_absolute_helix_move	Begin a absolute position helical interpolation
	APS_relative_helix_move	Begin a relative distance helical intepolation
11	**Interrupt**	
	APS_int_enable	Interrupt main switch
	APS_set_int_factor	Enable/Disable interrupt factor and get interrupt handle.
	APS_get_int_factor	Get interrupt factor enable or disable
	APS_wait_single_int	Wait single interrupt event
	APS_wait_multiple_int	Wait multiple interrupt events
	APS_reset_int	Reset interrupt event to non-signaled state.
	APS_set_int	Set interrupt event to signaled state.
	APS_set_int_factorH	Enable/Disable interrupt factor and get interrupt handle.(Win32)
12	**Sampling**	
	APS_set_sampling_param	Set sampling parameter.
	APS_get_sampling_param	Get sampling parameter.
	APS_wait_trigger_sampling	Waiting for sample data.
	APS_wait_trigger_sampling_async	Waiting for sample data asynchronously
	APS_get_sampling_count	Get sampled data count.
	APS_stop_wait_sampling	Force stop wait sampling
13	**DIO & AIO**	
	APS_write_d_output	Set digital output value
	APS_read_d_output	Read digital output value

	APS_read_d_input	Read digital input value
	APS_read_a_input_value	Read back analog input value by volt
	APS_read_a_input_data	Read back analog input raw data
	APS_write_a_output_value	Set analog output value by volt
	APS_write_a_output_data	Set analog output value by raw data
14	**Point table motion**	
	APS_set_point_table	Set point table move parameters
	APS_get_point_table	Get point table move parameters
	APS_point_table_move	Start a point table move
	APS_get_running_point_index	Get current point move index when axis is perform a point move
	APS_get_start_point_index	Get the first point move index when axis is perform a point move
	APS_get_end_point_index	Get the last point move index when axis is perform a point move
	APS_set_table_move_pause	Pause point table move
	APS_set_table_move_ex_pause	Decelerate to stop move and control I/O
	APS_set_table_move_ex_rollback	Rollback to starting position of current point index
	APS_set_table_move_ex_resume	Re-start point table move and keep I/O status
	APS_set_table_move_repeat	Set point table move repeat
17	**Gear / Gantry functions**	
	APS_set_gantry_param	Set gantry function related parameter
	APS_get_gantry_param	Get gantry function related parameter
	APS_set_gantry_axis	Set two axes in a gantry group
	APS_get_gantry_axis	Get which axes in a gantry group
	APS_get_gantry_error	Get gantry axes deviation error
	APS_get_encoder	Get encoder(Be used for compensation of gantry home return)

	APS_get_latch_event	Get latch event by axis(Be used for compensation of gantry home return)
	APS_get_latch_counter	Get latch counter by axis(Be used for compensation of gantry home return)
18	**Compare trigger**	
	APS_set_trigger_param	Set compare trigger related parameter
	APS_get_trigger_param	Get compare trigger related parameter
	APS_set_trigger_linear	Set linear comparing function
	APS_set_trigger_table	Set table comparing function
	APS_set_trigger_manual	Manual output trigger
	APS_set_trigger_manual_s	Manual output trigger synchronously
	APS_get_trigger_table_cmp	Get current table comparing value
	APS_get_trigger_linear_cmp	Get current linear comparing value
	APS_get_trigger_count	Get triggered count.
	APS_reset_trigger_count	Reset triggered count.
20	**Manual Pulse Generator functions**	
	APS_get_pulser_counter	Get pluse input counter
27	**VAO/PWM functions(Laser function)**	
	APS_set_vao_param	Set parameter to VAO table
	APS_get_vao_param	Get parameter of VAO table
	APS_set_vao_table	Set VAO table
	APS_switch_vao_table	Switch to specified VAO table
	APS_start_vao	Enable VAO output channel
	APS_get_vao_status	Get VAO status
	APS_check_vao_param	Check parameters setting of specified VAO table
	APS_set_vao_param_ex	Set table parameters via VAO structure
	APS_get_vao_param_ex	Get table parameters via VAO structure
	APS_set_pwm_on	Start to output PWM signal
	APS_set_pwm_width	Set pulse width to a PWM channel

	APS_set_pwm_frequency	Set pulse frequency to a PWM channel
	APS_get_pwm_width	Get pulse width from a PWM channel
	APS_get_pwm_frequency	Get pulse frequency from a PWM channel
36	**Table definition**	
	Board parameter table	
	Axis parameter table	
	Sampling parameter table	
	Sampling source table	
	Motion IO status and motion status definitions	
	Motion status definition table	
	Interrupt factor table	
	Gantry parameters table	
	Trigger parameter table	
	Device information table	
	VAO parameter table	
	APS functions return code	

List of all functions for PCI-8144

Sec.	Function name	Descriptions
3	**System & Initialization**	
	APS_initial	Device initialization
	APS_close	Device close
	APS_version	Get the version of the library
	APS_device_driver_version	Get the driver's version of devices
	APS_get_axis_info	Get the information of the specified axis
	APS_get_card_name	Get card index
	APS_set_axis_param	Set axis parameter
	APS_get_axis_param	Get axis parameter
	APS_get_device_info	Get device information
	APS_set_security_key	Set security password
	APS_check_security_key	Varify security password
	APS_reset_security_key	Reset security password
	APS_load_param_from_file	Load parameters from file
5	**Motion IO and motion status**	
	APS_motion_status	Return motion status
	APS_motion_io_status	Return motion IO status
	APS_get_command	Get command position
	APS_set_command	Set command position
	APS_get_command_velocity	Get command velocity
6	**Single axis motion**	
	APS_relative_move	Begin a relative distance move
	APS_velocity_move	Begin a velocity move
	APS_home_move	Begin a home move

	APS_stop_move	Stop move
	APS_emg_stop	Emergency stop
11	**Interrupt**	
	APS_int_enable	Interrupt main switch
	APS_set_int_factor	Enable/Disable interrupt factor and get interrupt handle.
	APS_get_int_factor	Get interrupt factor enable or disable
	APS_wait_single_int	Wait single interrupt event
	APS_wait_multiple_int	Wait multiple interrupt events
	APS_reset_int	Reset interrupt event to non-signaled state.
	APS_set_int	Set interrupt event to signaled state.
	APS_set_int_factorH	Enable/Disable interrupt factor and get interrupt handle.(Win32)
13	**DIO & AIO**	
	APS_write_d_output	Set digital output value
	APS_read_d_output	Read digital output value
	APS_read_d_input	Read digital input value
23	**NV RAM funciton**	
	APS_set_nv_ram	Set RAM data
	APS_get_nv_ram	Get RAM data
	APS_clear_nv_ram	Clear RAM data
36	**Table definition**	
	Axis parameter table	
	Motion IO status and motion status definitions	
	Motion status definition table	
	Interrupt factor table	
	Device information table	
	APS functions return code	

List of all functions for AMP-104C

Sec.	Function name	Descriptions
3	**System & Initialization**	
	APS_initial	Device initialization
	APS_close	Device close
	APS_version	Get the version of the library
	APS_device_driver_version	Get the driver's version of devices
	APS_get_axis_info	Get the information of the specified axis
	APS_get_card_name	Get card index
	APS_set_axis_param	Set axis parameter
	APS_get_axis_param	Get axis parameter
	APS_get_device_info	Get device information
	APS_set_security_key	Set security password
	APS_check_security_key	Varify security password
	APS_reset_security_key	Reset security password
	APS_load_param_from_file	Load parameters from file
5	**Motion IO and motion status**	
	APS_motion_status	Return motion status
	APS_motion_io_status	Return motion IO status
	APS_get_command	Get command position
	APS_set_command	Set command position
	APS_get_command_velocity	Get command velocity
	APS_get_encoder	Get raw feedback counter
6	**Single axis motion**	
	APS_relative_move	Begin a relative distance move
	APS_absolute_move	Begin a absolute position move

	APS_velocity_move	Begin a velocity move
	APS_home_move	Begin a home move
	APS_stop_move	Stop move
	APS_emg_stop	Emergency stop
9	**Interpolation**	
	APS_absolute_linear_move	Begin an absolute position linear interpolation
	APS_relative_linear_move	Begin a relative distance linear interpolation
11	**Interrupt**	
	APS_int_enable	Interrupt main switch
	APS_set_int_factor	Enable/Disable interrupt factor and get interrupt handle.
	APS_get_int_factor	Get interrupt factor enable or disable
	APS_wait_single_int	Wait single interrupt event
	APS_wait_multiple_int	Wait multiple interrupt events
	APS_reset_int	Reset interrupt event to non-signaled state.
	APS_set_int	Set interrupt event to signaled state.
	APS_set_int_factorH	Enable/Disable interrupt factor and get interrupt handle.(Win32)
13	**DIO & AIO**	
	APS_write_d_output	Set digital output value
	APS_read_d_output	Read digital output value
	APS_read_d_input	Read digital input value
31	**Multi-latch functions**	
	APS_get_ltc_fifo_point	Get latch point array
	APS_set_ltc_fifo_param	Set latch parameter value
	APS_get_ltc_fifo_param	Get latch parameter value
	APS_reset_ltc_fifo	Reset latch queue and fifo
	APS_get_ltc_fifo_usage	Get latch queue used space
	APS_get_ltc_fifo_free_space	Get latch queue free space
	APS_get_ltc_fifo_status	Get latch queue and fifo status

36	Table definition
	Axis parameter table
	Motion IO status and motion status definitions
	Motion status definition table
	Interrupt factor table
	Latch parameter table
	Device information table
	APS functions return code

List of all functions for PCI(e)-7856

Sec.	Function name	Descriptions
3	**System & Initialization**	
	APS_initial	Device initialization
	APS_close	Device close
	APS_version	Get the version of the library
	APS_device_driver_version	Get the driver's version of devices
	APS_get_axis_info	Get the information of the specified axis
	APS_get_card_name	Get card index
	APS_set_board_param	Set board parameter
	APS_get_board_param	Get board parameter
	APS_get_device_info	Get device information
	APS_save_param_to_file	Save parameters to file
	APS_load_param_from_file	Load parameters from file
11	**Interrupt**	
	APS_int_enable	Interrupt main switch
	APS_set_int_factor	Enable/Disable interrupt factor and get interrupt handle.
	APS_get_int_factor	Get interrupt factor enable or disable
	APS_wait_single_int	Wait single interrupt event
	APS_wait_multiple_int	Wait multiple interrupt events
	APS_reset_int	Reset interrupt event to non-signaled state.
	APS_set_int	Set interrupt event to signaled state.
	APS_set_int_factorH	Enable/Disable interrupt factor and get interrupt handle.(Win32)
16	**Field bus functions**	
	APS_set_field_bus_param	Set field bus related parameters

	APS_get_field_bus_param	Get field bus related parameters
	APS_start_field_bus	Start the network of specified field bus
	APS_stop_field_bus	Stop the network of specified field bus
	APS_field_bus_d_set_output	Set field bus digital output
	APS_field_bus_d_get_output	Get field bus digital output
	APS_field_bus_d_get_input	Get field bus digital input
	APS_set_field_bus_slave_param	Set parameter to field bus slave module
	APS_get_field_bus_slave_param	Get parameter from field bus slave module
	APS_set_field_bus_a_output	Set field bus analog output
	APS_get_field_bus_a_output	Get field bus analog output
	APS_get_field_bus_a_input	Get field bus analog input
	APS_get_slave_connect_quality	Get the connected quality of slave
	APS_get_slave_online_status	Get the online status of slave
	APS_get_field_bus_last_scan_info	Get fieldbus info after system scanning.
	APS_get_field_bus_master_type	Get master type of the fieldbus
	APS_get_field_bus_slave_type	Get slave type on the fieldbus
	APS_get_field_bus_slave_name	Get slave name on the fieldbus
	APS_get_field_bus_slave_first_axisno	Get first axis of the slave module
	APS_get_field_bus_device_info	Get device information on a specified field bus
	APS_field_bus_d_set_output_ex	Set field bus digital output for 64 bit DIO operation
	APS_field_bus_d_get_output_ex	Get field bus digital output for 64 bit DIO operation
	APS_field_bus_d_get_input_ex	Get field bus digital input for 64 bit DIO operation
23	NV RAM funciton	
	APS_set_nv_ram	Set RAM data
	APS_get_nv_ram	Get RAM data

	APS_clear_nv_ram	Clear RAM data
36	**Table definition**	
	Board parameter table	
	Field bus parameter table	
	Interrupt factor table	
	Device information table	
	APS functions return code	

List of all functions for MNET-4XMO

Sec.	Function name	Descriptions
3	**System & Initialization**	
	APS_get_axis_info	Get the information of the specified axis
	APS_set_axis_param	Set axis parameter
	APS_get_axis_param	Get axis parameter
	APS_save_param_to_file	Save parameters to file
	APS_load_param_from_file	Load parameters from file
5	**Motion IO and motion status**	
	APS_motion_status	Return motion status
	APS_motion_io_status	Return motion IO status
	APS_set_servo_on	Set servo ON/OFF
	APS_get_position	Get feedback position
	APS_set_position	Set feedback position
	APS_get_command	Get command position
	APS_set_command	Set command position
	APS_get_command_velocity	Get command velocity
	APS_get_error_position	Get error position
	APS_get_target_position	Get target position
	APS_get_position_f	Get feedback position by double
	APS_set_position_f	Set feedback position by double
	APS_get_command_f	Get command position by double
	APS_set_command_f	Set command position by double
	APS_get_command_velocity_f	Get command velocity by double
	APS_get_error_position_f	Get error position by double
	APS_get_target_position_f	Get target position by double

6	**Single axis motion**	
	APS_relative_move	Begin a relative distance move
	APS_absolute_move	Begin a absolute position move
	APS_velocity_move	Begin a velocity move
	APS_home_move	Begin a home move
	APS_stop_move	Stop move
	APS_emg_stop	Emergency stop
	APS_home_escape	Leave home switch
9	**Interpolation**	
	APS_absolute_linear_move	Begin an absolute position linear interpolation
	APS_relative_linear_move	Begin a relative distance linear interpolation
	APS_absolute_arc_move	Begin an absolute position circular interpolation
	APS_relative_arc_move	Begin a relative distance circular interpolation
11	**Interrupt**	
	APS_int_enable	Interrupt main switch
	APS_reset_field_bus_int_motion	Reset interrupt status of axes for MotionNet series.
	APS_set_field_bus_int_factor_motion	Enable/Disable motion interrupt factor and get interrupt handle for MotionNet series.
	APS_get_field_bus_int_factor_motion	Get motion interrupt factor enable or disable for MotionNet series.
	APS_set_field_bus_int_factor_error	Enable/Disable error interrupt factor and get interrupt handle for MotionNet series.
	APS_get_field_bus_int_factor_error	Get error interrupt factor status for MotionNet series.
	APS_wait_single_int	Wait single interrupt event
	APS_wait_multiple_int	Wait multiple interrupt events
	APS_reset_int	Reset interrupt event to non-signaled state.

	APS_set_int	Set interrupt event to signaled state.
	APS_int_no_to_handle	Convert interrupt event number to interrupt handle.(Win32)
	APS_wait_field_bus_error_int_motion	Wait error interrupt event for MotionNet series.
12	**Sampling**	
	APS_set_sampling_param	Set sampling parameter.
	APS_get_sampling_param	Get sampling parameter.
	APS_wait_trigger_sampling	Waiting for sample data.
	APS_wait_trigger_sampling_async	Waiting for sample data asynchronously
	APS_get_sampling_count	Get sampled data count.
	APS_stop_wait_sampling	Force stop wait sampling
16	**Field bus functions**	
	APS_set_field_bus_param	Set field bus related parameters
	APS_get_field_bus_param	Get field bus related parameters
	APS_start_field_bus	Start the network of specified field bus
	APS_stop_field_bus	Stop the network of specified field bus
	APS_field_bus_d_set_output	Set field bus digital output
	APS_field_bus_d_get_output	Get field bus digital output
	APS_field_bus_d_get_input	Get field bus digital input
	APS_get_slave_online_status	Get the online status of slave
	APS_get_field_bus_last_scan_info	Get fieldbus info after system scanning.
	APS_get_field_bus_master_type	Get master type of the fieldbus
	APS_get_field_bus_slave_type	Get slave type on the fieldbus
	APS_get_field_bus_slave_name	Get slave name on the fieldbus
	APS_get_field_bus_slave_first_axisno	Get first axis of the slave module
	APS_get_field_bus_device_info	Get device information on a specified field bus
29	**Simultaneous move functions**	

	APS_set_relative_simultaneous_move	Setup a relative simultaneous move
	APS_set_absolute_simultaneous_move	Setup a absolute simultaneous move
	APS_start_simultaneous_move	Begin a simultaneous move
	APS_stop_simultaneous_move	Stop a simultaneous move
30	**Single-latch functions**	
	APS_manual_latch2	Manual latch for a axis
	APS_get_latch_data2	Get latch data for a axis
36	**Table definition**	
	Axis parameter table	
	Motion IO status and motion status definitions	
	Motion status definition table	
	Interrupt factor table	
	Field bus parameter table	
	Device information table	
	APS functions return code	

List of all functions for MNET-4XMO-C

Sec.	Function name	Descriptions
3	**System & Initialization**	
	APS_get_axis_info	Get the information of the specified axis
	APS_set_axis_param	Set axis parameter
	APS_get_axis_param	Get axis parameter
	APS_save_param_to_file	Save parameters to file
	APS_load_param_from_file	Load parameters from file
5	**Motion IO and motion status**	
	APS_motion_status	Return motion status
	APS_motion_io_status	Return motion IO status
	APS_set_servo_on	Set servo ON/OFF
	APS_get_position	Get feedback position
	APS_set_position	Set feedback position
	APS_get_command	Get command position
	APS_set_command	Set command position
	APS_get_command_velocity	Get command velocity
	APS_get_error_position	Get error position
	APS_get_target_position	Get target position
	APS_get_position_f	Get feedback position by double
	APS_set_position_f	Set feedback position by double
	APS_get_command_f	Get command position by double
	APS_set_command_f	Set command position by double
	APS_get_command_velocity_f	Get command velocity by double
	APS_get_error_position_f	Get error position by double
6	**Single axis motion**	

	APS_relative_move	Begin a relative distance move
	APS_absolute_move	Begin a absolute position move
	APS_velocity_move	Begin a velocity move
	APS_home_move	Begin a home move
	APS_stop_move	Stop move
	APS_emg_stop	Emergency stop
	APS_home_escape	Leave home switch
9	**Interpolation**	
	APS_absolute_linear_move	Begin an absolute position linear interpolation
	APS_relative_linear_move	Begin a relative distance linear interpolation
	APS_absolute_arc_move	Begin an absolute position circular interpolation
	APS_relative_arc_move	Begin a relative distance circular interpolation
11	**Interrupt**	
	APS_int_enable	Interrupt main switch
	APS_reset_field_bus_int_motion	Reset interrupt status of axes for MotionNet series.
	APS_set_field_bus_int_factor_motion	Enable/Disable motion interrupt factor and get interrupt handle for MotionNet series.
	APS_get_field_bus_int_factor_motion	Get motion interrupt factor enable or disable for MotionNet series.
	APS_set_field_bus_int_factor_error	Enable/Disable error interrupt factor and get interrupt handle for MotionNet series.
	APS_get_field_bus_int_factor_error	Get error interrupt factor status for MotionNet series.
	APS_wait_single_int	Wait single interrupt event
	APS_wait_multiple_int	Wait multiple interrupt events
	APS_reset_int	Reset interrupt event to non-signaled state.
	APS_set_int	Set interrupt event to signaled state.

	APS_int_no_to_handle	Convert interrupt event number to interrupt handle.(Win32)
	APS_wait_field_bus_error_int_motion	Wait error interrupt event for MotionNet series.
12	**Sampling**	
	APS_set_sampling_param	Set sampling parameter.
	APS_get_sampling_param	Get sampling parameter.
	APS_wait_trigger_sampling	Waiting for sample data.
	APS_wait_trigger_sampling_async	Waiting for sample data asynchronously
	APS_get_sampling_count	Get sampled data count.
	APS_stop_wait_sampling	Force stop wait sampling
14	**Point table motion**	
	APS_set_point_table_mode2	Set point table mode
	APS_set_point_table2	Set point table
	APS_point_table_continuous_move2	Start a point table continuous move
	APS_point_table_single_move2	Start a point table single move
	APS_get_running_point_index2	Get current point move index when axis is perform a point move
	APS_point_table_status2	Get point table stauts
16	**Field bus functions**	
	APS_set_field_bus_param	Set field bus related parameters
	APS_get_field_bus_param	Get field bus related parameters
	APS_start_field_bus	Start the network of specified field bus
	APS_stop_field_bus	Stop the network of specified field bus
	APS_field_bus_d_set_output	Set field bus digital output
	APS_field_bus_d_get_output	Get field bus digital output
	APS_field_bus_d_get_input	Get field bus digital input
	APS_get_slave_online_status	Get the online status of slave
	APS_get_field_bus_last_scan_info	Get fieldbus info after system scanning.

	APS_get_field_bus_master_type	Get master type of the fieldbus
	APS_get_field_bus_slave_type	Get slave type on the fieldbus
	APS_get_field_bus_slave_name	Get slave name on the fieldbus
	APS_get_field_bus_slave_first_axisno	Get first axis of the slave module
	APS_get_field_bus_device_info	Get device information on a specified field bus
24	**Field bus Compare trigger**	
	APS_set_field_bus_trigger_param	Set compare trigger related parameter
	APS_get_field_bus_trigger_param	Get compare trigger related parameter
	APS_set_field_bus_trigger_linear	Set linear comparing function
	APS_set_field_bus_trigger_table	Set table comparing function
	APS_set_field_bus_trigger_manual	Manual output trigger
	APS_set_field_bus_trigger_manual_s	Manual output trigger synchronously
	APS_get_field_bus_trigger_table_cmp	Get current table comparing value
	APS_get_field_bus_trigger_linear_cmp	Get current linear comparing value
	APS_get_field_bus_trigger_count	Get triggered count.
	APS_reset_field_bus_trigger_count	Reset triggered count.
	APS_get_field_bus_linear_cmp_remain_count	Get remaining counter of linear comparator
	APS_get_field_bus_table_cmp_remain_count	Get remaining counter of table comparator
	APS_get_field_bus_encoder	Get encoder counter
	APS_set_field_bus_encoder	Set encoder counter
29	**Simultaneous move functions**	
	APS_set_relative_simultaneous_move	Setup a relative simultaneous move
	APS_set_absolute_simultaneous_move	Setup a absolute simultaneous move
	APS_start_simultaneous_move	Begin a simultaneous move
	APS_stop_simultaneous_move	Stop a simultaneous move
30	**Single-latch functions**	
	APS_manual_latch2	Manual latch for a axis

	APS_get_latch_data2	Get latch data for a axis
36	**Table definition**	
	Axis parameter table	
	Motion IO status and motion status definitions	
	Motion status definition table	
	Interrupt factor table	
	Field bus parameter table	
	Trigger parameter table	
	Device information table	
	APS functions return code	

List of all functions for MNET-1XMO

Sec.	Function name	Descriptions
3	**System & Initialization**	
	APS_get_axis_info	Get the information of the specified axis
	APS_set_axis_param	Set axis parameter
	APS_get_axis_param	Get axis parameter
	APS_save_param_to_file	Save parameters to file
	APS_load_param_from_file	Load parameters from file
5	**Motion IO and motion status**	
	APS_motion_status	Return motion status
	APS_motion_io_status	Return motion IO status
	APS_set_servo_on	Set servo ON/OFF
	APS_get_position	Get feedback position
	APS_set_position	Set feedback position
	APS_get_command	Get command position
	APS_set_command	Set command position
	APS_get_command_velocity	Get command velocity
	APS_get_error_position	Get error position
	APS_get_target_position	Get target position
	APS_get_position_f	Get feedback position by double
	APS_set_position_f	Set feedback position by double
	APS_get_command_f	Get command position by double
	APS_set_command_f	Set command position by double
	APS_get_command_velocity_f	Get command velocity by double
	APS_get_error_position_f	Get error position by double
	APS_get_target_position_f	Get target position by double

6	**Single axis motion**	
	APS_relative_move	Begin a relative distance move
	APS_absolute_move	Begin a absolute position move
	APS_velocity_move	Begin a velocity move
	APS_home_move	Begin a home move
	APS_stop_move	Stop move
	APS_emg_stop	Emergency stop
	APS_speed_override	Change speed on the fly
	APS_relative_move_ovrd	Begin a relative distance move or override it with new distance and speed
	APS_absolute_move_ovrd	Begin an absolute position move or override it with new position and speed
	APS_home_escape	Leave home switch
11	**Interrupt**	
	APS_int_enable	Interrupt main switch
	APS_reset_field_bus_int_motion	Reset interrupt status of axes for MotionNet series.
	APS_set_field_bus_int_factor_motion	Enable/Disable motion interrupt factor and get interrupt handle for MotionNet series.
	APS_get_field_bus_int_factor_motion	Get motion interrupt factor enable or disable for MotionNet series.
	APS_set_field_bus_int_factor_error	Enable/Disable error interrupt factor and get interrupt handle for MotionNet series.
	APS_get_field_bus_int_factor_error	Get error interrupt factor status for MotionNet series.
	APS_wait_single_int	Wait single interrupt event
	APS_wait_multiple_int	Wait multiple interrupt events
	APS_reset_int	Reset interrupt event to non-signaled state.
	APS_set_int	Set interrupt event to signaled state.
	APS_int_no_to_handle	Convert interrupt event number to interrupt handle.(Win32)

	APS_wait_field_bus_error_int_motion	Wait error interrupt event for MotionNet series.
16	**Field bus functions**	
	APS_set_field_bus_param	Set field bus related parameters
	APS_get_field_bus_param	Get field bus related parameters
	APS_start_field_bus	Start the network of specified field bus
	APS_stop_field_bus	Stop the network of specified field bus
	APS_field_bus_d_set_output	Set field bus digital output
	APS_field_bus_d_get_output	Get field bus digital output
	APS_get_slave_online_status	Get the online status of slave
	APS_get_field_bus_last_scan_info	Get fieldbus info after system scanning.
	APS_get_field_bus_master_type	Get master type of the fieldbus
	APS_get_field_bus_slave_type	Get slave type on the fieldbus
	APS_get_field_bus_slave_name	Get slave name on the fieldbus
	APS_get_field_bus_slave_first_axisno	Get first axis of the slave module
30	**Single-latch functions**	
	APS_manual_latch2	Manual latch for a axis
	APS_get_latch_data2	Get latch data for a axis
36	**Table definition**	
	Axis parameter table	
	Motion IO status and motion status definitions	
	Motion status definition table	
	Interrupt factor table	
	Field bus parameter table	
	APS functions return code	

List of all functions for HSL-4XMO

Sec.	Function name	Descriptions
3	**System & Initialization**	
	APS_get_axis_info	Get the information of the specified axis
	APS_set_axis_param	Set axis parameter
	APS_get_axis_param	Get axis parameter
	APS_load_param_from_file	Load parameters from file
5	**Motion IO and motion status**	
	APS_motion_status	Return motion status
	APS_motion_io_status	Return motion IO status
	APS_set_servo_on	Set servo ON/OFF
	APS_get_position	Get feedback position
	APS_set_position	Set feedback position
	APS_get_command	Get command position
	APS_set_command	Set command position
	APS_get_command_velocity	Get command velocity
	APS_get_error_position	Get error position
	APS_get_target_position	Get target position
6	**Single axis motion**	
	APS_relative_move	Begin a relative distance move
	APS_absolute_move	Begin a absolute position move
	APS_velocity_move	Begin a velocity move
	APS_home_move	Begin a home move
	APS_stop_move	Stop move
	APS_emg_stop	Emergency stop
9	**Interpolation**	

	APS_absolute_linear_move	Begin an absolute position linear interpolation
	APS_relative_linear_move	Begin a relative distance linear interpolation
	APS_absolute_arc_move	Begin an absolute position circular interpolation
	APS_relative_arc_move	Begin a relative distance circular interpolation
14	**Point table motion**	
	APS_set_point_table3	Set point table
	APS_point_table_move3	Start a point table single move
	APS_set_point_table_param3	Set speed parameter
16	**Field bus functions**	
	APS_set_field_bus_param	Set field bus related parameters
	APS_get_field_bus_param	Get field bus related parameters
	APS_start_field_bus	Start the network of specified field bus
	APS_stop_field_bus	Stop the network of specified field bus
	APS_field_bus_d_set_output	Set field bus digital output
	APS_field_bus_d_get_output	Get field bus digital output
	APS_field_bus_d_get_input	Get field bus digital input
	APS_get_slave_online_status	Get the online status of slave
	APS_get_field_bus_last_scan_info	Get fieldbus info after system scanning.
	APS_get_field_bus_master_type	Get master type of the fieldbus
	APS_get_field_bus_slave_type	Get slave type on the fieldbus
	APS_get_field_bus_slave_name	Get slave name on the fieldbus
	APS_get_field_bus_slave_first_axisno	Get first axis of the slave module
	APS_get_field_bus_device_info	Get device information on a specified field bus
24	**Field bus Compare trigger**	
	APS_set_field_bus_trigger_param	Set compare trigger related parameter

	APS_get_field_bus_trigger_param	Get compare trigger related parameter
	APS_set_field_bus_trigger_linear	Set linear comparing function
	APS_set_field_bus_trigger_table	Set table comparing function
	APS_get_field_bus_trigger_table_cmp	Get current table comparing value
	APS_get_field_bus_trigger_linear_cmp	Get current linear comparing value
36	**Table definition**	
	Axis parameter table	
	Motion IO status and motion status definitions	
	Motion status definition table	
	Field bus parameter table	
	Trigger parameter table	
	Device information table	
	APS functions return code	

List of all functions for HSL-DIO

Sec.	Function name	Descriptions
11	**Interrupt**	
	APS_int_enable	Interrupt main switch
	APS_set_field_bus_int_factor_di	Assign DI interrupt bits and get interrupt handle for HSL series.
	APS_get_field_bus_int_factor_di	Get DI interrupt bits assigned
	APS_wait_single_int	Wait single interrupt event
	APS_wait_multiple_int	Wait multiple interrupt events
	APS_reset_int	Reset interrupt event to non-signaled state.
	APS_set_int	Set interrupt event to signaled state.
16	**Field bus functions**	
	APS_set_field_bus_param	Set field bus related parameters
	APS_get_field_bus_param	Get field bus related parameters
	APS_start_field_bus	Start the network of specified field bus
	APS_stop_field_bus	Stop the network of specified field bus
	APS_field_bus_d_set_output	Set field bus digital output
	APS_field_bus_d_get_output	Get field bus digital output
	APS_field_bus_d_get_input	Get field bus digital input
	APS_get_slave_online_status	Get the online status of slave
	APS_get_field_bus_last_scan_info	Get fieldbus info after system scanning.
	APS_get_field_bus_master_type	Get master type of the fieldbus
	APS_get_field_bus_slave_type	Get slave type on the fieldbus
	APS_get_field_bus_slave_name	Get slave name on the fieldbus
36	**Table definition**	
	Field bus parameter table	
	APS functions return code	

List of all functions for PCI-8102 / PCI-C154(+)

Sec.	Function name	Descriptions
3	**System & Initialization**	
	APS_initial	Device initialization
	APS_close	Device close
	APS_version	Get the version of the library
	APS_device_driver_version	Get the driver's version of devices
	APS_get_axis_info	Get the information of the specified axis
	APS_get_card_name	Get card index
	APS_set_axis_param	Set axis parameter
	APS_get_axis_param	Get axis parameter
	APS_get_device_info	Get device information
	APS_load_param_from_file	Load parameters from file
5	**Motion IO and motion status**	
	APS_motion_status	Return motion status
	APS_motion_io_status	Return motion IO status
	APS_set_servo_on	Set servo ON/OFF
	APS_get_position	Get feedback position
	APS_set_position	Set feedback position
	APS_get_command	Get command position
	APS_set_command	Set command position
	APS_get_command_velocity	Get command velocity
	APS_get_error_position	Get error position
	APS_get_target_position	Get target position
	APS_get_position_f	Get feedback position by double
	APS_set_position_f	Set feedback position by double

	APS_get_command_f	Get command position by double
	APS_set_command_f	Set command position by double
	APS_get_command_velocity_f	Get command velocity by double
	APS_get_error_position_f	Get error position by double
	APS_get_target_position_f	Get target position by double
6	**Single axis motion**	
	APS_relative_move	Begin a relative distance move
	APS_absolute_move	Begin a absolute position move
	APS_velocity_move	Begin a velocity move
	APS_home_move	Begin a home move
	APS_stop_move	Stop move
	APS_emg_stop	Emergency stop
	APS_speed_override	Change speed on the fly (PCI-C154+ only)
	APS_relative_move_ovrd	Begin a relative distance move or override it with new distance and speed (PCI-C154+ only)
	APS_absolute_move_ovrd	Begin an absolute position move or override it with new position and speed (PCI-C154+ only)
	APS_home_escape	Leave home switch
9	**Interpolation**	
	APS_absolute_linear_move	Begin an absolute position linear interpolation
	APS_relative_linear_move	Begin a relative distance linear interpolation
	APS_absolute_arc_move	Begin an absolute position circular interpolation
	APS_relative_arc_move	Begin a relative distance circular interpolation
11	**Interrupt**	
	APS_int_enable	Interrupt main switch

	APS_set_int_factor	Enable/Disable interrupt factor and get interrupt handle.
	APS_get_int_factor	Get interrupt factor enable or disable
	APS_wait_single_int	Wait single interrupt event
	APS_wait_multiple_int	Wait multiple interrupt events
	APS_wait_error_int	Wait error interrupts(Non-mask)
	APS_reset_int	Reset interrupt event to non-signaled state.
	APS_set_int	Set interrupt event to signaled state.
	APS_set_int_factorH	Enable/Disable interrupt factor and get interrupt handle.(Win32)
	APS_int_no_to_handle	Convert interrupt event number to interrupt handle.(Win32)
13	DIO & AIO	
	APS_write_d_output	Set digital output value
	APS_read_d_output	Read digital output value
	APS_read_d_input	Read digital input value
18	Compare trigger	
	APS_set_trigger_param	Set trigger parameters
	APS_get_trigger_param	Get compare trigger related parameter
	APS_reset_trigger_count	Reset trigger counter
	APS_get_trigger_count	Get trigger counter
	APS_enable_trigger_fifo_cmp	Enable trigger fifo comparator
	APS_get_trigger_fifo_cmp	Get trigger fifo comparator
	APS_get_trigger_fifo_status	Get trigger fifo status
	APS_set_trigger_fifo_data	Set trigger fifo data
	APS_set_trigger_linear	Set trigger linear comparator
	APS_get_trigger_linear_cmp	Get trigger linear comparator
	APS_set_trigger_manual	Set trigger manual
	APS_set_trigger_manual_s	Manual output trigger synchronously
	APS_start_timer	Start timer (simulate for encoder)

	APS_get_timer_counter	Get timer count value (simulate for encoder)
	APS_set_timer_counter	Set timer count value (simulate for encoder)
	APS_start_trigger_timer	Start timer (generate trigger signal)
	APS_get_trigger_timer_counter	Get timer count value (generate trigger signal)
20	**Manual Pulse Generator functions**	
	APS_manual_pulser_start	Enable/Disable PA/PB input
	APS_manual_pulser_velocity_move	Begin a pulser velocity move
	APS_manual_pulser_relative_move	Begin a pulser relative distance move
	APS_manual_pulser_home_move	Begin a pulser home move
29	**Simultaneous move functions(PCI-C154+ only)**	
	APS_set_relative_simultaneous_move	Setup a relative simultaneous move
	APS_set_absolute_simultaneous_move	Setup a absolute simultaneous move
	APS_start_simultaneous_move	Begin a simultaneous move
	APS_stop_simultaneous_move	Stop a simultaneous move
30	**Single-latch functions**	
	APS_manual_latch2	Manual latch for a axis
	APS_get_latch_data2	Get latch data for a axis
31	**Multi-latch functions**	
	APS_set_ltc_counter	Set latch counter
	APS_get_ltc_counter	Get latch data for a axis
	APS_set_ltc_fifo_param	Set latch parameter
	APS_get_ltc_fifo_param	Get latch parameter
	APS_manual_latch	Latch data manually and synchronously.
	APS_enable_ltc_fifo	Enable/Disable ltc fifo
	APS_reset_ltc_fifo	Reset ltc fifo
	APS_get_ltc_fifo_data	Get one latch data from fifo
	APS_get_ltc_fifo_usage	Get usage of latch fifo
	APS_get_ltc_fifo_free_space	Get free space of latch fifo

	APS_get_ltc_fifo_status	Get fifo status
33	**Speed Profile Calculation**	
	APS_relative_move_profile	Get relative speed profile
	APS_absolute_move_profile	Get absolute speed profile
36	**Table definition**	
	Axis parameter table	
	Motion IO status and motion status definitions	
	Motion status definition table	
	Interrupt factor table(PCI-8102) / Interrupt factor table(PCI-C154+)	
	Trigger parameter table(PCI-C154+ only)	
	Latch parameter table(PCI-C154+ only)	
	Device information table	
	APS functions return code	

List of all functions for PCI-8154/8158

Sec.	Function name	Descriptions
3	**System & Initialization**	
	APS_initial	Device initialization
	APS_close	Device close
	APS_version	Get the version of the library
	APS_device_driver_version	Get the driver's version of devices
	APS_get_axis_info	Get the information of the specified axis
	APS_get_card_name	Get card index
	APS_set_axis_param	Set axis parameter
	APS_get_axis_param	Get axis parameter
	APS_get_device_info	Get device information
	APS_set_security_key	Set security password
	APS_check_security_key	Varify security password
	APS_reset_security_key	Reset security password
	APS_load_param_from_file	Load parameters from file
5	**Motion IO and motion status**	
	APS_motion_status	Return motion status
	APS_motion_io_status	Return motion IO status
	APS_set_servo_on	Set servo ON/OFF
	APS_get_position	Get feedback position
	APS_set_position	Set feedback position
	APS_get_command	Get command position
	APS_set_command	Set command position
	APS_get_command_velocity	Get command velocity
	APS_get_error_position	Get error position

	APS_get_target_position	Get target position
	APS_get_position_f	Get feedback position by double
	APS_set_position_f	Set feedback position by double
	APS_get_command_f	Get command position by double
	APS_set_command_f	Set command position by double
	APS_get_command_velocity_f	Get command velocity by double
	APS_get_error_position_f	Get error position by double
	APS_get_target_position_f	Get target position by double
6	**Single axis motion**	
	APS_relative_move	Begin a relative distance move
	APS_absolute_move	Begin a absolute position move
	APS_velocity_move	Begin a velocity move
	APS_home_move	Begin a home move
	APS_stop_move	Stop move
	APS_emg_stop	Emergency stop
	APS_speed_override	Change speed on the fly
	APS_relative_move_ovrd	Begin a relative distance move or override it with new distance and speed
	APS_absolute_move_ovrd	Begin an absolute position move or override it with new position and speed
	APS_home_escape	Leave home switch
9	**Interpolation**	
	APS_absolute_linear_move	Begin an absolute position linear interpolation
	APS_relative_linear_move	Begin a relative distance linear interpolation
	APS_absolute_arc_move	Begin an absolute position circular interpolation
	APS_relative_arc_move	Begin a relative distance circular interpolation
	APS_absolute_helical_move	Begin an absolute position helical

		interpolation
	APS_relative_helical_move	Begin a relative distance helical interpolation
10	**Advanced single move & interpolation**	
	APS_ptp_v	Begin a single move with Vm profile
	APS_ptp_all	Begin a single move with all profile
	APS_vel	Begin a velocity move
	APS_vel_all	Begin a velocity move with all profile
	APS_line	Begin a line move
	APS_line_v	Begin a line move with Vm profile
	APS_line_all	Begin a line move with all profile
	APS_arc2_ca	Begin an Arc2 move of angle type
	APS_arc2_ca_v	Begin an Arc2 move of angle type with Vm profile
	APS_arc2_ca_all	Begin an Arc2 move of angle type with all profile
	APS_arc2_ce	Begin an Arc2 move of end position
	APS_arc2_ce_v	Begin an Arc2 move of end position with Vm profile
	APS_arc2_ce_all	Begin an Arc2 move of end position with all profile
	APS_spiral_ce	Begin a 3D spiral-helix move of end position
	APS_spiral_ce_v	Begin a 3D spiral-helix move of end position with Vm profile
	APS_spiral_ce_all	Begin a 3D spiral-helix move of end position with all profile
11	**Interrupt**	
	APS_int_enable	Interrupt main switch
	APS_set_int_factor	Enable/Disable interrupt factor and get

		interrupt handle.
	APS_get_int_factor	Get interrupt factor enable or disable
	APS_wait_single_int	Wait single interrupt event
	APS_wait_multiple_int	Wait multiple interrupt events
	APS_wait_error_int	Wait error interrupts(Non-mask)
	APS_reset_int	Reset interrupt event to non-signaled state.
	APS_set_int	Set interrupt event to signaled state.
	APS_set_int_factorH	Enable/Disable interrupt factor and get interrupt handle.(Win32)
	APS_int_no_to_handle	Convert interrupt event number to interrupt handle.(Win32)
13	DIO & AIO	
	APS_write_d_output	Set digital output value
	APS_read_d_output	Read digital output value
	APS_read_d_input	Read digital input value
18	Compare trigger (only DB-8150)	
	APS_set_trigger_param	Set trigger parameters
	APS_reset_trigger_count	Reset trigger counter
	APS_get_trigger_count	Get trigger counter
	APS_enable_trigger_fifo_cmp	Enable trigger fifo comparator
	APS_get_trigger_fifo_cmp	Get trigger fifo comparator
	APS_get_trigger_fifo_status	Get trigger fifo status
	APS_set_trigger_fifo_data	Set trigger fifo data
	APS_get_trigger_param	Get compare trigger related parameter
	APS_set_trigger_linear	Set trigger linear comparator
	APS_get_trigger_linear_cmp	Get trigger linear comparator
	APS_set_trigger_manual	Set trigger manual
	APS_start_timer	Start timer(simulate for encoder)
29	Simultaneous move functions	
	APS_set_relative_simultaneous_move	Setup a relative simultaneous move

	APS_set_absolute_simultaneous_move	Setup a absolute simultaneous move
	APS_start_simultaneous_move	Begin a simultaneous move
	APS_stop_simultaneous_move	Stop a simultaneous move
30	**Single-latch functions**	
	APS_manual_latch2	Manual latch for a axis
	APS_get_latch_data2	Get latch data for a axis
32	**Ring counter functions**	
	APS_set_ring_counter	Enable ring counter function
	APS_get_ring_counter	Get limitation value of ring counter
36	**Table definition**	
	Axis parameter table	
	Motion IO status and motion status definitions	
	Motion status definition table	
	Interrupt factor table	
	Trigger parameter table(only DB-8150)	
	Device information table	
	APS functions return code	

List of all functions for PCIe-8154/8158

Sec.	Function name	Descriptions
3	**System & Initialization**	
	APS_initial	Device initialization
	APS_close	Device close
	APS_version	Get the version of the library
	APS_device_driver_version	Get the driver's version of devices
	APS_get_axis_info	Get the information of the specified axis
	APS_get_card_name	Get card index
	APS_set_axis_param	Set axis parameter
	APS_get_axis_param	Get axis parameter
	APS_set_axis_param_f	Set axis parameter by double
	APS_get_axis_param_f	Get axis parameter by double
	APS_get_device_info	Get device information
	APS_load_param_from_file	Load parameters from file
5	**Motion IO and motion status**	
	APS_motion_status	Return motion status
	APS_motion_io_status	Return motion IO status
	APS_set_servo_on	Set servo ON/OFF
	APS_get_position	Get feedback position
	APS_set_position	Set feedback position
	APS_get_command	Get command position
	APS_set_command	Set command position
	APS_get_command_velocity	Get command velocity
	APS_get_error_position	Get error position
	APS_get_target_position	Get target position

	APS_get_position_f	Get feedback position by double
	APS_set_position_f	Set feedback position by double
	APS_get_command_f	Get command position by double
	APS_set_command_f	Set command position by double
	APS_get_command_velocity_f	Get command velocity by double
	APS_get_error_position_f	Get error position by double
	APS_get_target_position_f	Get target position by double
6	**Single axis motion**	
	APS_relative_move	Begin a relative distance move
	APS_absolute_move	Begin a absolute position move
	APS_velocity_move	Begin a velocity move
	APS_home_move	Begin a home move
	APS_stop_move	Stop move
	APS_emg_stop	Emergency stop
	APS_speed_override	Change speed on the fly
	APS_relative_move_ovrd	Begin a relative distance move or override it with new distance and speed
	APS_absolute_move_ovrd	Begin an absolute position move or override it with new position and speed
	APS_home_escape	Leave home switch
9	**Interpolation**	
	APS_absolute_linear_move	Begin an absolute position linear interpolation
	APS_relative_linear_move	Begin a relative distance linear interpolation
	APS_absolute_arc_move	Begin an absolute position circular interpolation
	APS_relative_arc_move	Begin a relative distance circular interpolation
	APS_absolute_helical_move	Begin an absolute position helical interpolation

	APS_relative_helical_move	Begin a relative distance helical interpolation
10	**Advanced single move & interpolation**	
	APS_ptp_v	Begin a single move with Vm profile
	APS_ptp_all	Begin a single move with all profile
	APS_vel	Begin a velocity move
	APS_vel_all	Begin a velocity move with all profile
	APS_line	Begin a line move
	APS_line_v	Begin a line move with Vm profile
	APS_line_all	Begin a line move with all profile
	APS_arc2_ca	Begin an Arc2 move of angle type
	APS_arc2_ca_v	Begin an Arc2 move of angle type with Vm profile
	APS_arc2_ca_all	Begin an Arc2 move of angle type with all profile
	APS_arc2_ce	Begin an Arc2 move of end position
	APS_arc2_ce_v	Begin an Arc2 move of end position with Vm profile
	APS_arc2_ce_all	Begin an Arc2 move of end position with all profile
	APS_spiral_ce	Begin a 3D spiral-helix move of end position
	APS_spiral_ce_v	Begin a 3D spiral-helix move of end position with Vm profile
	APS_spiral_ce_all	Begin a 3D spiral-helix move of end position with all profile
11	**Interrupt**	
	APS_int_enable	Interrupt main switch
	APS_set_int_factor	Enable/Disable interrupt factor and get interrupt handle.

	APS_get_int_factor	Get interrupt factor enable or disable
	APS_wait_single_int	Wait single interrupt event
	APS_wait_multiple_int	Wait multiple interrupt events
	APS_wait_error_int	Wait error interrupts(Non-mask)
	APS_reset_int	Reset interrupt event to non-signaled state.
	APS_set_int	Set interrupt event to signaled state.
	APS_set_int_factorH	Enable/Disable interrupt factor and get interrupt handle.(Win32)
	APS_int_no_to_handle	Convert interrupt event number to interrupt handle.(Win32)
13	**DIO & AIO**	
	APS_write_d_output	Set digital output value
	APS_read_d_output	Read digital output value
	APS_read_d_input	Read digital input value
	APS_write_d_channel_output	Set digital output value by channel
	APS_read_d_channel_output	Read digital output value by channel
	APS_read_d_channel_input	Read digital input value by channel
20	**Manual Pulse Generator functions**	
	APS_manual_pulser_start	Enable/Disable PA/PB input
	APS_manual_pulser_velocity_move	Begin a pulser velocity move
	APS_manual_pulser_relative_move	Begin a pulser relative distance move
	APS_manual_pulser_home_move	Begin a pulser home move
29	**Simultaneous move functions**	
	APS_set_relative_simultaneous_move	Setup a relative simultaneous move
	APS_set_absolute_simultaneous_move	Setup a absolute simultaneous move
	APS_start_simultaneous_move	Begin a simultaneous move
	APS_stop_simultaneous_move	Stop a simultaneous move
30	**Single-latch functions**	
	APS_manual_latch2	Manual latch for a axis

	APS_get_latch_data2	Get latch data for a axis
36	**Table definition**	
	Axis parameter table	
	Motion IO status and motion status definitions	
	Motion status definition table	
	Interrupt factor table	
	Trigger parameter table	
	Device information table	
	APS functions return code	

List of all functions for EMX-100

Sec.	Function name	Descriptions
3	**System & Initialization**	
	APS_initial	Device initialization
	APS_close	Device close
	APS_version	Get the version of the library
	APS_get_axis_info	Get the information of the specified axis
	APS_get_card_name	Get card index
	APS_set_board_param	Set board parameter
	APS_get_board_param	Get board parameter
	APS_set_axis_param	Set axis parameter
	APS_get_axis_param	Get axis parameter
	APS_get_device_info	Get device information
	APS_get_first_axisId	Get first axis id of the card
	APS_load_parameter_from_default	Load system & axes parameters by default value
	APS_load_param_from_file	Load parameters from file
	APS_register_emx	Register EMX in library
	APS_get_deviceIP	Get Device IP
	APS_reset_emx_alarm	Reset the alarm signal of device
5	**Motion IO and motion status**	
	APS_motion_status	Return motion status
	APS_motion_io_status	Return motion IO status
	APS_set_servo_on	Set servo ON/OFF
	APS_get_position	Get feedback position
	APS_set_position	Set feedback position

	APS_get_command	Get command position
	APS_set_command	Set command position
	APS_get_error_position	Get error position
	APS_get_target_position	Get target position
	APS_get_command_velocity	Get command velocity
	APS_get_feedback_velocity	Get feedback velocity
6	**Single axis motion**	
	APS_relative_move	Begin a relative distance move
	APS_absolute_move	Begin a absolute position move
	APS_velocity_move	Begin a velocity move
	APS_home_move	Begin a home move
	APS_stop_move	Stop move
	APS_emg_stop	Emergency stop
8	**Jog move**	
	APS_jog_start	Start / stop jog move
9	**Interpolation**	
	APS_absolute_linear_move	Begin an absolute position linear interpolation
	APS_relative_linear_move	Begin a relative distance linear interpolation
13	**DIO & AIO**	
	APS_write_d_output	Set digital output value
	APS_read_d_output	Read digital output value
	APS_read_d_input	Read digital input value
	APS_write_d_channel_output	Set digital output value by channel
	APS_read_d_channel_output	Read digital output value by channel
18	**Compare trigger**	
	APS_set_trigger_param	Set compare trigger related parameter
	APS_get_trigger_param	Get compare trigger related parameter
	APS_get_trigger_count	Get triggered count

	APS_reset_trigger_count	Reset triggered count
33	**Speed Profile Calculation**	
	APS_check_motion_profile_emx	Get relative speed profile
36	**Table definition**	
	Board parameter table	
	Axis parameters definition table	
	Motion IO status and motion status definitions	
	Motion status definition table	
	Trigger parameter table	
	Device information table	
	APS functions return code	

List of all functions for PCI-8254/58 / AMP-204/8C

Sec.	Function name	Descriptions
3	**System & Initialization**	
	APS_initial	Device initialization
	APS_close	Device close
	APS_version	Get the version of the library
	APS_device_driver_version	Get the driver's version of devices
	APS_get_axis_info	Get the information of the specified axis
	APS_get_card_name	Get card index
	APS_disable_device	Disable cards
	APS_set_board_param	Set board parameter
	APS_get_board_param	Get board parameter
	APS_set_axis_param	Set axis parameter
	APS_get_axis_param	Get axis parameter
	APS_set_axis_param_f	Set axis parameter by double
	APS_get_axis_param_f	Get axis parameter by double
	APS_get_system_timer	Get system timer counter
	APS_get_device_info	Get device information
	APS_get_first_axisId	Get first axis id of the card
	APS_save_parameter_to_flash	Save system & axes parameters to flash
	APS_load_parameter_from_flash	Load system & axes parameters from flash
	APS_load_parameter_from_default	Load system & axes parameters by default value
	APS_load_param_from_file	Load parameters from file
	APS_get_curr_sys_ctrl_mode	Get current system control mode
5	**Motion IO and motion status**	
	APS_motion_status	Return motion status

	APS_motion_io_status	Return motion IO status
	APS_set_servo_on	Set servo ON/OFF
	APS_get_position	Get feedback position
	APS_set_position	Set feedback position
	APS_get_command	Get command position
	APS_set_command	Set command position
	APS_get_command_velocity	Get command velocity
	APS_get_feedback_velocity	Get feedback velocity
	APS_get_error_position	Get error position
	APS_get_target_position	Get target position
	APS_get_position_f	Get feedback position by double
	APS_set_position_f	Set feedback position by double
	APS_get_command_f	Get command position by double
	APS_set_command_f	Set command position by double
	APS_get_error_position_f	Get error position by double
	APS_get_target_position_f	Get target position by double
	APS_get_command_velocity_f	Get command velocity by double
	APS_get_feedback_velocity_f	Get feedback velocity by double
	APS_get_mq_free_space	Get free space of motion queue
	APS_get_mq_usage	Get usage of motion queue
	APS_get_stop_code	Get stop code
	APS_get_encoder	Get raw feedback counter
	APS_get_command_counter	Get raw command counter
	APS_get_axis_latch_data	Get ORG/EZ latch data
6	**Single axis motion**	
	APS_relative_move	Begin a relative distance move
	APS_absolute_move	Begin a absolute position move
	APS_velocity_move	Begin a velocity move

	APS_home_move	Begin a home move
	APS_stop_move	Stop move
	APS_emg_stop	Emergency stop
7	**Multi-axes move trigger & stop**	
	APS_move_trigger	Send a trigger to sync all waiting moves
	APS_stop_move_multi	Multi-axes stop move
	APS_emg_stop_multi	Multi-axes emg stop move
8	**Jog move**	
	APS_jog_start	Start / stop jog move
9	**Interpolation**	
	APS_absolute_linear_move	Begin an absolute position linear interpolation
	APS_relative_linear_move	Begin a relative distance linear interpolation
	APS_absolute_arc_move	Begin an absolute position circular interpolation
	APS_relative_arc_move	Begin a relative distance circular interpolation
10	**Advanced single move & interpolation**	
	APS_ptp	Begin a single move
	APS_ptp_v	Begin a single move with Vm profile
	APS_ptp_all	Begin a single move with all profile
	APS_vel	Begin a velocity move
	APS_vel_all	Begin a velocity move with all profile
	APS_line	Begin a line move
	APS_line_v	Begin a line move with Vm profile
	APS_line_all	Begin a line move with all profile
	APS_arc2_ca	Begin an Arc2 move of angle type
	APS_arc2_ca_v	Begin an Arc2 move of angle type with Vm profile

	APS_arc2_ca_all	Begin an Arc2 move of angle type with all profile
	APS_arc2_ce	Begin an Arc2 move of end position
	APS_arc2_ce_v	Begin an Arc2 move of end position with Vm profile
	APS_arc2_ce_all	Begin an Arc2 move of end position with all profile
	APS_arc3_ca	Begin an Arc3 move of angle type
	APS_arc3_ca_v	Begin an Arc3 move of angle type with Vm profile
	APS_arc3_ca_all	Begin an Arc3 move of angle type with all profile
	APS_arc3_ce	Begin an Arc3 move of end position
	APS_arc3_ce_v	Begin an Arc3 move of end position with Vm profile
	APS_arc3_ce_all	Begin an Arc3 move of end position with all profile
	APS_spiral_ca	Begin a 3D spiral-helix move of angle type
	APS_spiral_ca_v	Begin a 3D spiral-helix move of angle type with Vm profile
	APS_spiral_ca_all	Begin a 3D spiral-helix move of angle type with all profile
	APS_spiral_ce	Begin a 3D spiral-helix move of end position
	APS_spiral_ce_v	Begin a 3D spiral-helix move of end position with Vm profile
	APS_spiral_ce_all	Begin a 3D spiral-helix move of end position with all profile
11	**Interrupt**	
	APS_int_enable	Interrupt main switch
	APS_set_int_factor	Enable/Disable interrupt factor and get interrupt handle.
	APS_get_int_factor	Get interrupt factor enable or disable

	APS_wait_single_int	Wait single interrupt event
	APS_wait_multiple_int	Wait multiple interrupt events
	APS_reset_int	Reset interrupt event to non-signaled state.
	APS_set_int	Set interrupt event to signaled state.
	APS_set_int_factorH	Enable/Disable interrupt factor and get interrupt handle.(Win32)
	APS_int_no_to_handle	Convert interrupt event number to interrupt handle.(Win32)
12	**Sampling**	
	APS_set_sampling_param	Set sampling parameter.
	APS_get_sampling_param	Get sampling parameter.
	APS_wait_trigger_sampling	Waiting for sample data.
	APS_wait_trigger_sampling_async	Waiting for sample data asynchronously
	APS_get_sampling_count	Get sampled data count.
	APS_stop_wait_sampling	Force stop wait sampling
	APS_auto_sampling	Start/Stop auto sampling
	APS_get_sampling_data	Get sampling data in auto sampling mode by 4 Channels.
	APS_set_sampling_param_ex	Set sampling parameter by structure. It is an extension to 8 channels.
	APS_get_sampling_param_ex	Get sampling parameter by structure. It is an extension to 8 channels.
	APS_wait_trigger_sampling_ex	Waiting for sample data. It is an extension to 8 channels.
	APS_wait_trigger_sampling_async_ex	Waiting for sample data asynchronously. It is an extension to 8 channels.
	APS_get_sampling_data_ex	Get sampling data in auto sampling mode. It is an extension to 8 channels.
13	**DIO & AIO**	
	APS_write_d_output	Set digital output value
	APS_read_d_output	Read digital output value

	APS_read_d_input	Read digital input value
	APS_write_d_channel_output	Set digital output value by channel
	APS_read_d_channel_output	Read digital output value by channel
	APS_read_a_input_value (*1)	Read back analog input value by volt
	APS_write_a_output_value (*1)	Set analog output value by volt
14	**Point table motion**	
	APS_set_feeder_group	Set axes into a feeder group
	APS_get_feeder_group	Return the configuration in one feeder group
	APS_free_feeder_group	Free a feeder group and it's resources
	APS_reset_feeder_buffer	Reset the feeder's point buffer
	APS_set_feeder_point_2D	Add a point into feeder's buffer
	APS_set_feeder_point_2D_ex	Add a point into feeder's buffer
	APS_start_feeder_move	Start point table move and feed points.
	APS_get_feeder_status	Get feeder status
	APS_get_feeder_running_index	Get which point is in operation.
	APS_get_feeder_feed_index	Get which point is set into point table.
	APS_set_feeder_ex_pause	Motion paused(stopped) and feeder paused
	APS_set_feeder_ex_rollback	Move back to the starting position of paused index
	APS_set_feeder_ex_resume	Resume the point-table move
15	**Advanced Point table**	
	APS_pt_enable	Enable point table.
	APS_pt_disable	Disable point table.
	APS_get_pt_info	Get information of point table.
	APS_pt_set_vs	Set configuration of Vs to point table
	APS_pt_get_vs	Get configuration of Vs in the point table
	APS_pt_start	Set control command to point table
	APS_pt_stop	Stop point table
	APS_get_pt_status	Get status of point table

	APS_reset_pt_buffer	Reset buffer of point table
	APS_pt_roll_back	Rollback to previous point
	APS_pt_get_error	Get error code of point table
	APS_pt_dwell	Push a dwell move into point buffer of point table.
	APS_pt_line	Push a line move into point buffer of point table.
	APS_pt_arc2_ca	Push a 2d arc move into point buffer of point table.
	APS_pt_arc2_ce	Push a 2d arc move into point buffer of point table.
	APS_pt_arc3_ca	Push a 3d arc move into point buffer of point table.
	APS_pt_arc3_ce	Push a 3d arc move into point buffer of point table.
	APS_pt_spiral_ca	Push a helical move into point buffer of point table.
	APS_pt_spiral_ce	Push a helical move into point buffer of point table.
	APS_pt_ext_set_do_ch	Set Do extension command into command buffer. Command buffer is active when pushing a move into point table.
	APS_pt_set_absolute	Set absolute profile into profile buffer.
	APS_pt_set_relative	Set relative profile into profile buffer.
	APS_pt_set_trans_buffered	Set transition to buffer mode in profile buffer.
	APS_pt_set_trans_inp	Set transition to in-position mode in profile buffer.
	APS_pt_set_trans_blend_dec	Set transition to blending mode with deceleration in profile buffer.
	APS_pt_set_trans_blend_dist	Set transition to blending mode with residue distant in profile buffer.

	APS_pt_set_trans_blend_pcnt	Set transition to blending mode with residue distant percetange in profile buffer.
	APS_pt_set_acc	Set accerlation profile into profile buffer.
	APS_pt_set_dec	Set deceleration profile into profile buffer.
	APS_pt_set_acc_dec	Set accerlation / deceleration profile into profile buffer
	APS_pt_set_s	Set S-factor profile into profile buffer.
	APS_pt_set_vm	Set maximum velocity profile into profile buffer.
	APS_pt_set_ve	Set end velocity profile into profile buffer.
17	**Gear / Gantry functions**	
	APS_start_gear	Enable/Disable a specified gear mode
	APS_get_gear_status	Get gear status
18	**Compare trigger**	
	APS_set_trigger_param	Set compare trigger related parameter
	APS_get_trigger_param	Get compare trigger related parameter
	APS_set_trigger_linear	Set linear comparing function
	APS_set_trigger_table	Set table comparing function
	APS_set_trigger_manual	Manual output trigger
	APS_set_trigger_manual_s	Manual output trigger synchronously
	APS_get_trigger_table_cmp	Get current table comparing value
	APS_get_trigger_linear_cmp	Get current linear comparing value
	APS_get_trigger_count	Get triggered count
	APS_reset_trigger_count	Reset triggered count
	APS_get_timer_counter	Get timer count
	APS_set_timer_counter	Set timer count
	APS_set_multi_trigger_table	Set table comparing function
	APS_get_multi_trigger_table_cmp	Get current table comparing value
	APS_set_trigger_table_data	Set table compator data (Fast table compare trigger function)

	APS_get_trigger_table_status	Get table comparator status (Fast table compare trigger function)
	APS_get_trigger_cmp_value	Get table comparator value (Fast table compare trigger function)
	APS_enable_trigger_table	Enable table comparator (Fast table compare trigger function)
	APS_reset_trigger_table	Reset table comparator (Fast table compare trigger function)
19	**Program download**(*1)	
	APS_load_vmc_program	Load VMC file to task memory
	APS_save_vmc_program	Save to VMC file from task memory
	APS_set_task_mode	Set task run mode
	APS_get_task_mode	Get task run mode
	APS_start_task	Start task control command
	APS_get_task_info	Get task information
	APS_get_task_msg	Get message of all tasks
20	**Manual Pulse Generator functions**	
	APS_manual_pulser_start	Start manual pulser operation
	APS_manual_pulser_velocity_move	Start velocity move in manual pulser operation
21	**Pitch error compensation functions**	
	APS_set_pitch_table	Set configurations and data of pitch error compensation table
	APS_get_pitch_table	Get configurations and data of pitch error compensation table
	APS_start_pitch_comp	Start pitch error compensation
26	**Watch dog timer**	
	APS_wdt_start	Start / Stop watch dog timer
	APS_wdt_get_timeout_period	Get a timeout period of watch dog timer
	APS_wdt_reset_counter	Reset counter of watch dog timer
	APS_wdt_get_counter	Get counter of watch dog timer
	APS_wdt_set_action_event	Set action event of watch dog timer
	APS_wdt_get_action_event	Get action event of watch dog timer
27	**VAO/PWM functions(Laser function)**	

	APS_set_vao_param	Set parameter to VAO table
	APS_get_vao_param	Get parameter of VAO table
	APS_set_vao_table	Set VAO table
	APS_switch_vao_table	Switch to specified VAO table
	APS_start_vao	Enable VAO output channel
	APS_get_vao_status	Get VAO status
	APS_check_vao_param	Check parameters setting of specified VAO table
	APS_set_vao_param_ex	Set table parameters via VAO structure
	APS_get_vao_param_ex	Get table parameters via VAO structure
	APS_set_pwm_on	Start to output PWM signal
	APS_set_pwm_width	Set pulse width to a PWM channel
	APS_set_pwm_frequency	Set pulse frequency to a PWM channel
	APS_get_pwm_width	Get pulse width from a PWM channel
	APS_get_pwm_frequency	Get pulse frequency from a PWM channel
28	**Circular limit functions**	
	APS_set_circular_limit	Set circular limit configurations
	APS_get_circular_limit	Get circular limit configurations
31	**Multi-latch functions**	
	APS_enable_ltc_fifo	Enable position latch process
	APS_get_ltc_fifo_point	Get latch point array
	APS_set_ltc_fifo_param	Set latch parameter value
	APS_get_ltc_fifo_param	Get latch parameter value
	APS_reset_ltc_fifo	Reset latch queue and fifo
	APS_get_ltc_fifo_usage	Get latch queue used space
	APS_get_ltc_fifo_free_space	Get latch queue free space
	APS_get_ltc_fifo_status	Get latch queue and fifo status
34	**Backlash functions**	
	APS_set_backlash_en	Enable/Disable backlash

	APS_get_backlash_en	Check backlash is enabled / disabled
35	**2-D compensation**	
	APS_set_2d_compensation_table	Create 2D compensation table
	APS_get_2d_compensation_table	Get 2D compensation table configuration
	APS_start_2d_compensation	Start or stop 2D compensation table
	APS_absolute_linear_move_2d_compensation	2D absolute linear interpolation
	APS_get_2d_compensation_command_position	Get command and feedback position
36	**Table definition**	
	Board parameter table	
	Axis parameter table	
	Sampling parameter table	
	Sampling source table	
	Motion IO status and motion status definitions	
	Motion status definition table	
	Interrupt factor table	
	Trigger parameter table	
	Latch parameter table	
	Device information table	
	VAO parameter table	
	APS functions return code	

***1. AMP series don't support this feature.**

List of all functions for PCIe-833x

Sec.	Function name	Descriptions
3	**System & Initialization**	
	APS_initial	Device initialization
	APS_close	Device close
	APS_version	Get the version of the library
	APS_device_driver_version	Get the driver's version of devices
	APS_get_axis_info	Get the information of the specified axis
	APS_get_card_name	Get card index
	APS_disable_device	Disable cards
	APS_set_board_param	Set board parameter
	APS_get_board_param	Get board parameter
	APS_set_axis_param	Set axis parameter
	APS_get_axis_param	Get axis parameter
	APS_set_axis_param_f	Set axis parameter by double
	APS_get_axis_param_f	Get axis parameter by double
	APS_get_system_timer	Get system timer counter
	APS_get_device_info	Get device information
	APS_save_parameter_to_flash	Save system & axes parameters to flash
	APS_load_parameter_from_flash	Load system & axes parameters from flash
	APS_load_parameter_from_default	Load system & axes parameters by default value.
5	**Motion IO and motion status**	
	APS_motion_status	Return motion status
	APS_motion_io_status	Return motion IO status
	APS_set_servo_on	Set servo ON/OFF
	APS_get_position_f	Get feedback position by double

	APS_set_position_f	Set feedback position by double
	APS_get_command_f	Get command position by double
	APS_set_command_f	Set command position by double
	APS_get_error_position_f	Get error position by double
	APS_get_target_position_f	Get target position by double
	APS_get_command_velocity_f	Get command velocity by double
	APS_get_feedback_velocity_f	Get feedback velocity by double
	APS_get_mq_free_space	Get free space of motion queue
	APS_get_mq_usage	Get usage of motion queue
	APS_get_stop_code	Get stop code
	APS_get_encoder	Get raw feedback counter
	APS_get_command_counter	Get raw command counter
	APS_reset_command_counter	Reset raw command counter
	APS_get_actual_torque	Get actual torque value
6	**Single axis motion**	
	APS_home_move	Begin a home move
	APS_stop_move	Stop move
	APS_emg_stop	Emergency stop
7	**Multi-axes move trigger & stop**	
	APS_move_trigger	Send a trigger to sync all waiting moves
	APS_stop_move_multi	Multi-axes stop move
	APS_emg_stop_multi	Multi-axes emg stop move
8	**Jog move**	
	APS_jog_start	Start / stop jog move
10	**Advanced single move & interpolation**	
	APS_ptp	Begin a single move
	APS_ptp_v	Begin a single move with Vm profile
	APS_ptp_all	Begin a single move with all profile

	APS_vel	Begin a velocity move
	APS_vel_all	Begin a velocity move with all profile
	APS_line	Begin a line move
	APS_line_v	Begin a line move with Vm profile
	APS_line_all	Begin a line move with all profile
	APS_arc2_ca	Begin an Arc2 move of angle type
	APS_arc2_ca_v	Begin an Arc2 move of angle type with Vm profile
	APS_arc2_ca_all	Begin an Arc2 move of angle type with all profile
	APS_arc2_ce	Begin an Arc2 move of end position
	APS_arc2_ce_v	Begin an Arc2 move of end position with Vm profile
	APS_arc2_ce_all	Begin an Arc2 move of end position with all profile
	APS_arc3_ca	Begin an Arc3 move of angle type
	APS_arc3_ca_v	Begin an Arc3 move of angle type with Vm profile
	APS_arc3_ca_all	Begin an Arc3 move of angle type with all profile
	APS_arc3_ce	Begin an Arc3 move of end position
	APS_arc3_ce_v	Begin an Arc3 move of end position with Vm profile
	APS_arc3_ce_all	Begin an Arc3 move of end position with all profile
	APS_spiral_ca	Begin a 3D spiral-helix move of angle type
	APS_spiral_ca_v	Begin a 3D spiral-helix move of angle type with Vm profile
	APS_spiral_ca_all	Begin a 3D spiral-helix move of angle type with all profile
	APS_spiral_ce	Begin a 3D spiral-helix move of end position

	APS_spiral_ce_v	Begin a 3D spiral-helix move of end position with Vm profile
	APS_spiral_ce_all	Begin a 3D spiral-helix move of end position with all profile
11	**Interrupt**	
	APS_int_enable	Interrupt main switch
	APS_set_int_factor	Enable/Disable interrupt factor and get interrupt handle.
	APS_get_int_factor	Get interrupt factor enable or disable
	APS_wait_single_int	Wait single interrupt event
	APS_wait_multiple_int	Wait multiple interrupt events
	APS_reset_int	Reset interrupt event to non-signaled state.
	APS_set_int	Set interrupt event to signaled state.
	APS_set_int_factorH	Enable/Disable interrupt factor and get interrupt handle.(Win32)
	APS_int_no_to_handle	Convert interrupt event number to interrupt handle.(Win32)
12	**Sampling**	
	APS_set_sampling_param	Set sampling parameter.
	APS_get_sampling_param	Get sampling parameter.
	APS_wait_trigger_sampling	Waiting for sample data.
	APS_wait_trigger_sampling_async	Waiting for sample data asynchronously
	APS_get_sampling_count	Get sampled data count.
	APS_stop_wait_sampling	Force stop wait sampling
	APS_auto_sampling	Start/Stop auto sampling
	APS_get_sampling_data	Get sampling data in auto sampling mode by 4 Channels.
	APS_set_sampling_param_ex	Set sampling parameter by structure. It is an extension to 8 channels.
	APS_get_sampling_param_ex	Get sampling parameter by structure. It is an extension to 8 channels.

	APS_wait_trigger_sampling_ex	Waiting for sample data. It is an extension to 8 channels.
	APS_wait_trigger_sampling_async_ex	Waiting for sample data asynchronously. It is an extension to 8 channels.
	APS_get_sampling_data_ex	Get sampling data in auto sampling mode. It is an extension to 8 channels.
13	**DIO & AIO**	
	APS_set_field_bus_d_channel_output	Set field bus digital output by channel
	APS_get_field_bus_d_channel_output	Get field bus digital output by channel
	APS_get_field_bus_d_channel_input	Get field bus digital input by channel
	APS_set_field_bus_d_port_output	Set field bus digital output by port
	APS_get_field_bus_d_port_input	Get field bus digital input by port
	APS_get_field_bus_d_port_output	Get field bus digital output by port
	APS_write_d_output	Set digital output value
	APS_read_d_output	Read digital output value
	APS_read_d_input	Read digital input value
	APS_write_d_channel_output	Set digital output value by channel
	APS_read_d_channel_output	Read digital output value by channel
	APS_read_d_channel_input	Read digital input value by channel
	APS_read_a_input_value	Read back analog input value by volt
	APS_write_a_output_value	Set analog output value by volt
15	**Advanced Point table**	
	APS_pt_enable	Enable point table.
	APS_pt_disable	Disable point table.
	APS_get_pt_info	Get information of point table.
	APS_pt_set_vs	Set configuration of Vs to point table
	APS_pt_get_vs	Get configuration of Vs in the point table
	APS_pt_start	Set control command to point table
	APS_pt_stop	Stop point table

	APS_get_pt_status	Get status of point table
	APS_reset_pt_buffer	Reset buffer of point table
	APS_pt_roll_back	Rollback to previous point
	APS_pt_dwell	Push a dwell move into point buffer of point table.
	APS_pt_line	Push a line move into point buffer of point table.
	APS_pt_arc2_ca	Push a 2d arc move into point buffer of point table.
	APS_pt_arc2_ce	Push a 2d arc move into point buffer of point table.
	APS_pt_arc3_ca	Push a 3d arc move into point buffer of point table.
	APS_pt_arc3_ce	Push a 3d arc move into point buffer of point table.
	APS_pt_spiral_ca	Push a helical move into point buffer of point table.
	APS_pt_spiral_ce	Push a helical move into point buffer of point table.
	APS_pt_ext_set_do_ch	Control digital output within advanced point-table in EPS-6000 slave
	APS_pt_set_absolute	Set absolute profile into profile buffer.
	APS_pt_set_relative	Set relative profile into profile buffer.
	APS_pt_set_trans_buffered	Set transition to buffer mode in profile buffer.
	APS_pt_set_trans_inp	Set transition to in-position mode in profile buffer.
	APS_pt_set_trans_blend_dec	Set transition to blending mode with deceleration in profile buffer.
	APS_pt_set_trans_blend_dist	Set transition to blending mode with residue distant in profile buffer.
	APS_pt_set_trans_blend_pcnt	Set transition to blending mode with residue

		distant percetange in profile buffer.
	APS_pt_set_acc	Set accerlation profile into profile buffer.
	APS_pt_set_dec	Set deceleration profile into profile buffer.
	APS_pt_set_acc_dec	Set accerlation / deceleration profile into profile buffer
	APS_pt_set_s	Set S-factor profile into profile buffer.
	APS_pt_set_vm	Set maximum velocity profile into profile buffer.
	APS_pt_set_ve	Set end velocity profile into profile buffer.
16	**Field bus functions**	
	APS_scan_field_bus	Scan field bus and generate ENI file
	APS_start_field_bus	Start the network of specified field bus
	APS_stop_field_bus	Stop the network of specified field bus
	APS_set_field_bus_a_output	Set field bus analog output
	APS_get_field_bus_a_output	Get field bus analog output
	APS_get_field_bus_a_input	Get field bus analog input
	APS_get_field_bus_master_status	Get field bus master status
	APS_get_slave_online_status	Get the status of slave
	APS_get_field_bus_last_scan_info	Get fieldbus info after system scanning.
	APS_get_field_bus_module_info	Get slave information
	APS_reset_field_bus_alarm	Reset the alarm signal of slave
	APS_get_field_bus_alarm	Get alarm code of slave
	APS_get_field_bus_pdo	Get value from PDO memory
	APS_set_field_bus_pdo	Set value to PDO memory
	APS_get_field_bus_pdo_offset	Get PDO information
	APS_get_field_bus_sdo	Get SDO data from slave
	APS_set_field_bus_sdo	Set SDO data to slave
	APS_set_field_bus_od_data	Set EtherCAT OD raw data
	APS_get_field_bus_od_data	Get EtherCAT OD raw data

	APS_get_field_bus_od_module_info	Get EtherCAT slave information
	APS_get_field_bus_module_map	Get mapped slave ID in manual ID mode
	APS_set_field_bus_module_map	Set mapped slave ID in manual ID mode
	APS_get_field_bus_slave_state	Get the status of slave's state machine
	APS_set_field_bus_slave_state	Set the status of slave's state machine
	APS_get_field_bus_ESC_register	Get EtherCAT Slave Controller register
	APS_set_field_bus_ESC_register	Set EtherCAT Slave Controller register
	APS_get_system_loading	Get system loop loading
	APS_get_field_bus_analysis_topology	Get current and past topology then analysis
	APS_get_field_bus_loss_package	Get the loss of EtherCAT frame count on receive bus direction.
17	**Gear / Gantry functions**	
	APS_start_gear	Enable/Disable a specified gear mode
	APS_get_gear_status	Get gear status
	APS_get_gantry_number	Get number of this master's corresponding slaves
	APS_get_gantry_info	Get slave axis ID array
	APS_get_gantry_deviation	Get position deviation between master and slaves
20	**Manual Pulse Generator functions**	
	APS_manual_pulser_start	Start manual pulser operation
	APS_manual_pulser_velocity_move	Start velocity move in manual pulser operation
26	**Watch dog timer**	
	APS_wdt_start	Start / Stop watch dog timer
	APS_wdt_get_timeout_period	Get a timeout period of watch dog timer
	APS_wdt_reset_counter	Reset counter of watch dog timer
	APS_wdt_get_counter	Get counter of watch dog timer
	APS_wdt_set_action_event	Set action event of watch dog timer
	APS_wdt_get_action_event	Get action event of watch dog timer
28	**Circular limit functions**	
	APS_set_circular_limit	Set circular limit configurations

	APS_get_circular_limit	Get circular limit configurations
34	**Backlash functions**	
	APS_set_backlash_en	Enable/Disable backlash
	APS_get_backlash_en	Check backlash is enabled / disabled
35	**2-D compensation**	
	APS_set_2d_compensation_table	Create 2D compensation table
	APS_get_2d_compensation_table	Get 2D compensation table configuration
	APS_start_2d_compensation	Start or stop 2D compensation table
	APS_absolute_linear_move_2d_compensation	2D absolute linear interpolation
	APS_get_2d_compensation_command_position	Get command and feedback position
36	**Table definition**	
	Board parameter table	
	Axis parameter table	
	Sampling parameters table	
	Sampling source table	
	Motion IO status and motion status definitions	
	Motion status definition table	
	Interrupt factor table	
	Device information table	
	APS functions return code	

List of all functions for ECAT-4XMO

Sec.	Function name	Descriptions
3	System & Initialization	
	Use with PCIe-833x	
5	Motion IO and motion status	
	Use with PCIe-833x	
6	Single axis motion	
	Use with PCIe-833x	
7	Multi-axes move trigger & stop	
	Use with PCIe-833x	
8	Jog move	
	Use with PCIe-833x	
10	Advanced single move & interpolation	
	Use with PCIe-833x	
12	Sampling	
	Use with PCIe-833x	
13	DIO & AIO	
	Use with PCIe-833x	
15	Advanced Point table	
	Use with PCIe-833x	
16	Field bus functions	
	Use with PCIe-833x	
17	Gear / Gantry functions	
	Use with PCIe-833x	
20	Manual Pulse Generator functions	
	Use with PCIe-833x	

24	**Field bus compare trigger**	
	APS_set_field_bus_trigger_param	Set compare trigger related parameter
	APS_get_field_bus_trigger_param	Get compare trigger related parameter
	APS_set_field_bus_trigger_linear	Set linear comparing function
	APS_set_field_bus_trigger_table	Set table comparing function
	APS_set_field_bus_trigger_manual	Manual output trigger
	APS_set_field_bus_trigger_manual_s	Manual output trigger synchronously
	APS_get_field_bus_trigger_table_cmp	Get current table comparing value
	APS_get_field_bus_trigger_linear_cmp	Get current linear comparing value
	APS_get_field_bus_trigger_count	Get triggered count.
	APS_reset_field_bus_trigger_count	Reset triggered count.
	APS_get_field_bus_linear_cmp_remain_count	Get remaining counter of linear comparator
	APS_get_field_bus_table_cmp_remain_count	Get remaining counter of table comparator
	APS_get_field_bus_encoder	Get encoder counter
	APS_get_field_bus_timer_counter	Get timer count.
	APS_set_field_bus_timer_counter	Set timer count.
25	**Field bus position latch functions**	
	APS_get_field_bus_ltc_fifo_point	Get latch point array.
	APS_set_field_bus_ltc_fifo_param	Set latch parameter value.
	APS_get_field_bus_ltc_fifo_param	Get latch parameter value.
	APS_reset_field_bus_ltc_fifo	Reset latch queue and fifo.
	APS_get_field_bus_ltc_fifo_usage	Get latch queue used space.
	APS_get_field_bus_ltc_fifo_free_space	Get latch queue free space.
	APS_get_field_bus_ltc_fifo_status	Get latch queue and fifo status.
28	**Circular limit functions**	
	Use with PCIe-833x	
35	**2-D compensation**	

	Use with PCIe-833x
36	**Table definition**
	Trigger parameter table
	Latch parameter table
	APS functions return code

List of all functions for ECAT-TRG4

Sec.	Function name	Descriptions
13	**DIO & AIO**	
	Use with PCIe-833x	
16	**Field bus functions**	
	Use with PCIe-833x	
24	**Field bus compare trigger**	
	APS_set_field_bus_trigger_param	Set compare trigger related parameter
	APS_get_field_bus_trigger_param	Get compare trigger related parameter
	APS_set_field_bus_trigger_linear	Set linear comparing function
	APS_set_field_bus_trigger_table	Set table comparing function
	APS_set_field_bus_trigger_manual	Manual output trigger
	APS_set_field_bus_trigger_manual_s	Manual output trigger synchronously
	APS_get_field_bus_trigger_table_cmp	Get current table comparing value
	APS_get_field_bus_trigger_linear_cmp	Get current linear comparing value
	APS_get_field_bus_trigger_count	Get triggered count.
	APS_reset_field_bus_trigger_count	Reset triggered count.
	APS_get_field_bus_linear_cmp_remain_count	Get remaining counter of linear comparator
	APS_get_field_bus_table_cmp_remain_count	Get remaining counter of table comparator
	APS_get_field_bus_encoder	Get encoder counter
	APS_get_field_bus_timer_counter	Get timer count.
	APS_set_field_bus_timer_counter	Set timer count.
25	**Field bus position latch functions**	
	APS_get_field_bus_ltc_fifo_point	Get latch point array.
	APS_set_field_bus_ltc_fifo_param	Set latch parameter value.

	APS_get_field_bus_ltc_fifo_param	Get latch parameter value.
	APS_reset_field_bus_ltc_fifo	Reset latch queue and fifo.
	APS_get_field_bus_ltc_fifo_usage	Get latch queue used space.
	APS_get_field_bus_ltc_fifo_free_space	Get latch queue free space.
	APS_get_field_bus_ltc_fifo_status	Get latch queue and fifo status.
36	**Table definition**	
	Trigger parameter table	
	Latch parameter table	
	APS functions return code	

3. System and Initialization

APS_initial	Device initialization

Support Products: PCI-8253/56, PCI-8392 (H), DPAC-1000, DPAC-3000, PCI-8144,PCI(e)-7856, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100, PCI-8254/58 / AMP-204/8C, PCIe-833x, AMP-104C

Descriptions:

This function is used to initialize all products on local controller supported by APS function library. It allocates system hardware resources for each board including I/O address, memory address, IRQ and DMA if needed. It retrieves a board ID for each board which is assigned by on-board switch or operating system. The board ID is a unique number for the board in the system. It is used for any other APS functions to access corresponding hardware.
If users choose on-board switch (Mode = manual ID) initial mode and there are some boards don't support this feature in the system, the board ID of these boards will be arranged after the boards having on-board switch automatically. The card ID (dip-switch) cannot be set the same when you used "manual-ID" or the function will return error.

For EMX-100：
NOTE:
(1) After EMX series product's power on or reconnection is completed, it is suggested to execute APS_initial() 20 seconds later.
(2) Before using APS_initial(), user has to register EMX series product in APS library using APS_register_emx().

Syntax:

C/C++:
I32 FNTYPE APS_initial(I32 *BoardID_InBits, I32 Mode);
Visual Basic:
APS_initial (BoardID_InBits As Long, ByVal Mode As Long) As Long

Parameters:

I32 * BoardID_InBits: Card ID information in bit format.
Example: If the value of BoardID_InBits is 0x11 which means that there are 2 cards in your system and those card's ID are 0 and 4.

I32 Mode:

Bit 0	Enable the On board dip switch (SW1) to decide the Card ID. [0:By system assigned, 1:By dip switch]
Bit 1	Parallel type axis indexing mode 0 : auto mode (default) 1 : fixed mode
Bit 2	Serial type axis indexing mode For PCIe-833x: [Note 1] 0: auto mode (default) 1: fixed mode
Bit 4 Bit 5	Option of load system & axes parameters method. For PCI-8253/6 and PCI-8392(H) (00B) 0: load according to boot mode setting in each board parameter. (01B) 1: load from default for all boards (02B) 2: load from flash for all boards For PCI-8254/58 / AMP-204/8C and PCIe-833x (00B) 0: Do nothing, parameters keep current value. (01B) 1: Load from default (02B) 2: Load from flash(*1)
Bit 6	Option to select system mode. (PCI(e)-7856 Only) (0) – Polling mode. (Not support motion interrupt) (1) – Interrupt mode. (Support motion interrupt)
Bit 9	Option to select behavior of MDN bit of motion status. (PCI-8254/58 / AMP-204/8C and PCIe-833x) only 0: MDN turns on when CSTP or ASTP occurs. (default) 1: MDN turns on when CSTP occurs.
Bit 10	EtherCAT Slave ID number setting.(PCIe-833x Only) [Note 1] 0: The slave ID number be auto assigned by system. (start from zero) 1: The slave ID number be manual assigned by user setting.
Others	Reserved(set it to 0)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret; // return value
I32 BoardID_InBits;
```

I32 Mode = 0; //By system assigned

For EMX-100 : ret = APS_register_emx(1, 0); // Register EMX series products in APS library

ret = APS_initial(&BoardID_InBits, Mode);

…// Do something

ret = APS_close(); //Close all cards in the system

See also:

For EMX-100 : APS_register_emx()

APS_close();APS_get_axis_info()

Only for PCIe-833X:

[Note 1]

As follows is the usage of manual slave ID number setting:

[Step 1]

Use MotionCreatorPro 2 (MCP2) utility program to set the manual slave ID number into each online slave. After that, close MCP2 and turn off the power of all online slaves.

[Step 2]

Turn on the power of all online slaves.

According to the [TABLE 1], user can select relationship mapping between slave ID and axis ID by the parameter "**I32 Mode**" in **APS_initial()**.

[Step 3]

Execute **APS_start_field_bus()** to start communication. Then the relationship mapping between slave ID and axis ID will be indicated by [TABLE 1]. Now, user can indicate slave ID or axis ID to control each online slave.

[TABLE 1]

	Slave ID	Axis ID
[Selection 1] I32 Mode = 0x0 (bit 10 = 0, bit 2 = 0)	Assigned by system (start from zero) Example: Slave ID: 0,1,2,3…	Assigned by system (start from zero) Example: Axis ID: 0,1,2,3…
[Selection 2]	Assigned by user's setting	Assigned by parameter "I32

I32 Mode = 0x400 (bit 10 = 1, bit 2 = 0)	(by MCP2 utility) Example: Slave ID: 100,200,300,400…	Starting_Axis_ID" in APS_start_field_bus() Example: I32 Starting_Axis_ID = 1000; Axis ID: 1000,1001,1002,1003…
[Selection 3] I32 Mode = 0x404 (bit 10 = 1, bit 2 = 1)	Assigned by user's setting (by MCP2 utility) Example: Slave ID: 100,200,300,400…	Refer to slave ID. For detail, please refer to example [TABLE 2].

[TABLE 2]

Slave ID (Assigned by user's setting)	Numbers of axis	Axis ID
100	1	100
200	2	200,201
300	3	300,301,302
400	1	400

APS_close	Devices close

Support Products: PCI-8253/56, PCI-8392 (H), DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-7856, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100, PCI-8254/58 / AMP-204/8C, PCIe-833x, AMP-104C

Descriptions:

This function is used to close all resources allocated by APS library. The resources include system hardware resource like I/O address, memory address, IRQ and DMA. It also deletes some objects, handles or memory allocated by APS library.

Syntax:

C/C++:

I32 FNTYPE APS_close()

Visual Basic:

APS_close() As Long

Parameters:

No parameter.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret; // return value
I32 BoardID_InBits;
I32 Mode = 0; //By system assigned
For EMX-100 : ret = APS_register_emx( 1, 0); // Register EMX series products in APS library
ret = APS_initial( &BoardID_InBits, Mode);

…// Do something
ret = APS_close(); //Close all cards in the system
```

See also:

APS_initial()

APS_version	Get the version of the library

Support Products : PCI-8253/56, PCI-8392 (H), DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-7856, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100, PCI-8254/58 / AMP-204/8C , PCIe-833x, AMP-104C

Descriptions :

This function is used to get APS library (DLL) version information.

Syntax:

C/C++:

I32 FNTYPE APS_version();

Visual Basic:

APS_version() As Long

Parameters:

No Parameters

Return Values:

Return library (DLL) version.

Example:

I32 version;

version = APS_version();

See also:

APS_device_driver_version	Get the driver's version of devices

Support Products : PCI-8253/56, PCI-8392 (H), DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-7856, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), PCI-8254/58 / AMP-204/8C , PCIe-833x, AMP-104C

Descriptions :

This function is used to get device driver version information of one board. The version information is the same for one type of board in system.

Syntax:

C/C++

I32 FNTYPE APS_device_driver_version(I32 Board_ID)

Visual Basic:

APS_device_driver_version(ByVal Board_ID As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

Return Values:

Positive value: The device driver version number,

Negative value: Error code: Please refer to error code table.

Example:

```
I32 version;

//Get device driver version of board 0
version = APS_device_driver_version( 0 );
```

See also:

APS_get_axis_info	Get the information of the specified axis

Support Products: PCI-8253/56, PCI-8392 (H) , DPAC-3000 , PCI-8144, PCI(e)-7856, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100, PCI-8254/58, PCI-8254/58 / AMP-204/8C, PCIe-833x, AMP-104C, ECAT-4XMO

Descriptions:

This function is used to get information of one axis. The information includes attached board ID, serial port ID, serial module ID and module type. There are two categories for axis ID index: parallel and serial types. PCI-8254/58 and PCI-8392 SSCNET 3 are parallel type axis. MNET-4XMO-(C) and HSL-4XMO are serial type axis.

The parallel type axis ID indexing rule is according to the board ID. The formula is:
Axis ID = Board ID x Maximum number of axis within one board + Axis No
The Axis No parameter is the axis number within the board. The Maximum number of axis within one board parameter is an inside system variable of APS library. If APS system is running under auto mode, the value depends on board type. If APS system is running under fixed mode, the default value is 32. If the system has some boards without axes, it still counts the formula when indexing under fixed mode. Fixed mode is useful for users to remove/add some boards from system without rearranging axis index.
For example, a user has two boards: PCI-8392 and PCI-8253 and PCI-8258 .

	PCI-8392 (ID=0), 8-axis	PCI-8253 (ID=1), 3-axis	PCI-8258 (ID=0), 8-axis
Auto Mode	Axis ID ranges 0~7	Axis ID ranges 8~10	Axis ID ranges 0~7
Fixed Mode	Axis ID ranges 0~7	Axis ID ranges 32~34	Axis ID ranges 0~15

If the board ID is not continuous,

	PCI-8392 (ID=0), 8-axis	PCI-8253 (ID=2), 3-axis	PCI-8258 (ID=0), 8-axis
Auto Mode	Axis ID ranges 0~7	Axis ID ranges 8~10	Axis ID ranges 0~7
Fixed Mode	Axis ID ranges 0~7	Axis ID ranges 64~66	Axis ID ranges 0~15

The serial type axis ID indexing rule is according to the module ID and assigned with a starting axis ID first. The formula of serial port axis would be:
Axis ID = Module ID x Maximum number of axis within one module + Starting Axis ID of port + Axis No
The Axis No parameter is the axis number within the module. The Maximum number of axis within this module parameter is an inside port variable of APS library. If APS field bus system is running under auto mode, the value depends on module type. If APS field bus system is running under fixed mode, the default value is 4. Starting Axis ID of port parameter is the

starting axis ID of one port assigned by users when field bus starts. The default value is 0. In fixed mode, if the port has some modules without axis, it still counts the formula when indexing. Fixed mode is useful for users to remove/add some modules from system without rearranging axis index of other modules

For example, a user has 2 MNET modules on PCI(e)-7856 with board ID=0

	MNET-J3 (ID=0)	MNET-4XMO (ID=1), 4-axis
Auto Mode	Axis ID ranges 0 only	Axis ID ranges 1~4
Fixed Mode	Axis ID ranges 0~3	Axis ID ranges 4~7

If the module ID is not continuous,

	MNET-J3 (ID=0)	MNET-4XMO (ID=2), 4-axis
Auto Mode	Axis ID ranges 0 only	Axis ID ranges 1~4
Fixed Mode	Axis ID ranges 0~3	Axis ID ranges 8~11

For EMX-100 and PCIe-833x, this function is used to get information of specified axis. The information includes corresponding board ID, number of axis on board, number of bus and auto slave ID.

Syntax:

C/C++:

I32 FNTYPE APS_get_axis_info(I32 Axis_ID, I32 *Board_ID, I32 *Axis_No, I32 *Port_ID, I32 *Module_ID);

Visual Basic:

APS_get_axis_info(ByVal Axis_ID As Long, Board_ID As Long, Axis_No As Long, Port_ID As Long, Module_ID As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535

I32 *Board_ID: The returned board ID for the Axis ID. Range is from 0 to 31.

I32 *Axis_No: The axis number within the board. Range is from 0 to maximum number of axis within this module.

I32 *Port_ID: The returned field bus port ID of board for the axis. Range is from 0 to 15.

*Port_ID=-1 means no serial port exists.

For PCI(e)-7856, HSL field bus is Port ID 0 and MNET field bus is Port ID 1.

I32 *Module_ID: The returned module ID of port for the axis. Range is from 0~65535.

*Module_ID=-1 means no serial port exists.

For High Speed Link(HSL) type field bus, the range of module number is 1 to 63. Note: In HSL, the Module_ID is the first id occupied by the module.

For MNET field bus, Range is from 0~63.

For EMX-100 and PCIe-833x :

I32 Axis_ID: The axis ID. That support auto axis ID and manual axis ID simultaneously.

In manual axis ID, the input range is from 1000 to 655357.

I32 *Board_ID: Return the corresponding board ID.

I32 *Axis_No: Return the corresponding number of axis on board.

I32 *Port_ID: Return the corresponding number of bus.

I32 *Module_ID: Return the corresponding auto slave ID.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Axis_ID = 0;
I32 Board_ID, Axis_No, Port_ID, Module_ID;

//According to axis id, get related information
APS_get_axis_info( Axis_ID, & Board_ID, &Axis_No, &Port_ID, &Module_ID );
```

See also:

APS_start_field_bus();APS_initial()

APS_get_card_name	Get card index

Support Products: PCI-8253/56, PCI-8392 (H) , DPAC-1000,DPAC-3000 , PCI-8144, PCI(e)-7856, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100 , PCI-8254/58 / AMP-204/8C , PCIe-833x, AMP-104C

Descriptions:

This function is used to get card name. After executing APS_initial(), user could get each board's name by passing specified board id.

Syntax:

C/C++:

I32 FNTYPE APS_get_card_name (I32 Board_ID, I32 *CardName);

Visual Basic:

APS_get_card_name (ByVal Board_ID As Long, CardName As Long) As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

I32 * CardName: Board's name of specified board id.

0: PCI_8392, 1: PCI_825x, 2: PCI_8154, 3: PCI_785X
4: PCI_8158, 5: PCI_7856, 6: ISA_DPAC1000, 7: ISA_DPAC3000
8: PCI_8144, 9: PCI_8258, 10: PCI_8102, 11: PCI_V8258
12: PCI_V8254, 13: PCI_8158A, 14: PCI_20408C, 15: PCI_8353
16: PCI_8392F, 17: PCI_C154, 18: PCI_C154_PLUS, 19: PCI_8353_RTX
20: PCIe_8338, 21: PCIe_8154, 22: PCIE_8158, 23: ENET_EMX100,
24: PCIe_8334, 25:PCIe_8332, 26:PCIe-8331, 27: PCIE_7856, 28: AMP-104C

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 CardName;
//Board id is 0. Get board's name
APS_get_card_name ( 0, & CardName);
```

See also:

APS_disable_device	Disable specified device. It is used to ignore the disabling device during initialization.

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x

Descriptions :

This function could disable specified board via its name. It is used to ignore the disabling device during initialization.

Syntax:

C/C++:

I32 FNTYPE APS_disable_device(I32 DeviceName);

Visual Basic:

APS_disable_device(ByVal DeviceName As Long) As Long

Parameters:

I32 DeviceName : Specify a device name.

0: PCI_8392, 1: PCI_825x, 2: PCI_8154, 3: PCI_785X
4: PCI_8158, 5: PCI_7856, 6: ISA_DPAC1000, 7: ISA_DPAC3000
8: PCI_8144, 9: PCI_825458, 10: PCI_8102, 11: PCI_V8258
12: PCI_V8254, 13: PCI_8158A, 14: PCI_20408C, 15: PCI_8353
16: PCI_8392F, 17: PCI_C154, 18: PCI_C154_PLUS, 19: PCI_8353_RTX
20: PCIe_8338, 21: PCIe_8154, 22: PCIE_8158, 23: ENET_EMX100,
24: PCIe_8334 25:PCIe_8332, 26:PCIE_8331, 27:PCIE_7856

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 BoardID_InBits;

//Disable PCI-8258
APS_disable_device( 9 );

//Initial all card, but PCI_8258
APS_initial( &BoardID_InBits, 0 );
```

See also:

APS_set_board_param	Set board parameter

Support Products: PCI-8253/56, PCI-8392 (H), DPAC-1000, DPAC-3000, PCI(e)-7856, EMX-100 , PCI-8254/58 / AMP-204/8C , PCIe-833x

Descriptions:

This function is used to set all kinds of parameter which has relationship with a board. Please refer to the board parameter table for the definition and detail descriptions.

Syntax:

C/C++

I32 FNTYPE APS_set_board_param(I32 Board_ID, I32 BOD_Param_No, I32 BOD_Param);

Visual Basic:

APS_set_board_param (ByVal Board_ID As Long, ByVal BOD_Param_No As Long, ByVal BOD_Param As Long) As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

I32 BOD_Param_No: Board parameter number. Please refer the board parameter table for definition.

I32 BOD_Param: Board parameter value. Refer to the board parameter table for detail.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
//Board id is 0. Set EMG logic to 1.
APS_set_board_param( 0, 0x00, 1 );
```

See also:

APS_get_board_param()

APS_get_board_param	Get board parameter

Support Products: PCI-8253/56, PCI-8392 (H), DPAC-1000, DPAC-3000, PCI(e)-7856, EMX-100 , PCIe-833x

Descriptions:

This function is used to get all kinds of parameter which has relationship with a board. Please refer to the board parameter table for the definition and detail descriptions.

Syntax:

C/C++:

I32 FNTYPE APS_get_board_param(I32 Board_ID, I32 BOD_Param_No, I32 *BOD_Param);

Visual Basic:

APS_get_board_param (ByVal Board_ID As Long, ByVal BOD_Param_No As Long, BOD_Param As Long) As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

I32 BOD_Param_No: Board parameter number. Please refer the board parameter table for definition.

I32 *BOD_Param: The returned board parameter value. Refer to board parameter table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 paramVal;

//Board id is 0. Get EMG logic.
APS_get_board_param( 0, 0x00, & paramVal );
```

See also:

APS_set_board_param()

APS_set_axis_param	Set axis parameter

Support Products: PCI-8253/56, PCI-8392 (H), PCI-8144, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100 , PCI-8254/58 / AMP-204/8C, PCIe-833x, AMP-104C, ECAT-4XMO

Descriptions:

This function is used to set all kinds of parameter of one axis. The parameters include run mode, acceleration rate, deceleration rate, Jerk, motion I/O logic and so on. Please refer to the axis parameter table for the definition and detail descriptions.

Syntax:

C/C++

I32 FNTYPE APS_set_axis_param(I32 Axis_ID, I32 AXS_Param_No, I32 AXS_Param);

Visual Basic:

APS_set_axis_param(ByVal Axis_ID As Long, ByVal AXS_Param_No As Long, ByVal AXS_Param As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 AXS_Param_No: Axis parameter number from 0 to 65535. Each parameter is defined by a unique symbol in 3~6 characters .Refer to axis parameter table.

I32 AXS_Param: Axis parameter value. Refer to axis parameter table

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
//Axis id is 0. Set EL logic to 1
APS_set_axis_param ( 0, 0x00, 1 );
```

See also:

APS_get_axis_param()

APS_get_axis_param	Get axis parameter

Support Products: PCI-8253/56, PCI-8392 (H), PCI-8144, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100 , PCI-8254/58 / AMP-204/8C, PCIe-833x, AMP-104C, ECAT-4XMO

Descriptions:

This function is used to get all kinds of parameter of one axis. The parameters include run mode, acceleration rate, deceleration rate, Jerk, motion I/O logic and so on. Please refer to the axis parameter table for the definition and detail descriptions.

Syntax:

C/C++:

I32 FNTYPE APS_get_axis_param(I32 Axis_ID, I32 AXS_Param_No, I32 *AXS_Param);

Visual Basic:

APS_get_axis_param (ByVal Axis_ID As Long, ByVal AXS_Param_No As Long, AXS_Param As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 AXS_Param_No: Axis parameter number from 0 to 65535. Each parameter is defined by a unique symbol in 3~6 characters .Refer to axis parameter table.

I32 *AXS_Param: Axis parameter value. Refer to axis parameter table

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 paramVal;
//Axis id is 0. Get EL logic
APS_get_axis_param ( 0, 0x00, &paramVal );
```

See also:

APS_set_axis_param()

APS_set_axis_param_f	Set axis parameter by double

Support Products : PCIe-8154/8158, PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set parameters by double. Those double parameters include acceleration rate, deceleration rate, Jerk and so on. Please refer to the axis parameter table for the definition and detailed descriptions.

Syntax:

C/C++

I32 FNTYPE APS_set_axis_param_f(I32 Axis_ID, I32 AXS_Param_No, F64 AXS_Param);

Visual Basic:

APS_set_axis_param_f(ByVal Axis_ID As Long, ByVal AXS_Param_No As Long, ByVal AXS_Param As Double) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 AXS_Param_No: Axis parameter number from 0 to 65535. Each parameter is defined by a unique symbol in 3~6 characters .Refer to axis parameter table.

F64 AXS_Param: Axis parameter value. (F64 type) Refer to axis parameter table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
//Axis id is 0. Set acceleration to 100000.0
APS_set_axis_param_f ( 0, 0x13, 100000.0 );
```

See also:

APS_get_axis_param_f();APS_set_axis_param();APS_get_axis_param()

APS_get_axis_param_f	Get axis parameter by double

Support Products : PCIe-8154/8158, PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get axis parameters by flaot. The double parameters include acceleration rate, deceleration rate, Jerk and so on. Please refer to the axis parameter table for the definition and detailed descriptions.

Syntax:

C/C++:

I32 FNTYPE APS_get_axis_param_f(I32 Axis_ID, I32 AXS_Param_No, F64 *AXS_Param);

Visual Basic:

APS_get_axis_param_f(ByVal Axis_ID As Long, ByVal AXS_Param_No As Long, AXS_Param As Double) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 AXS_Param_No: Axis parameter number from 0 to 65535. Each parameter is defined by a unique symbol in 3~6 characters .Refer to axis parameter table.

F64 *AXS_Param: Axis parameter value. (F64 type) Refer to axis parameter table

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
F64 paramVal;

//Axis id is 0. Get acceleration
APS_get_axis_param_f ( 0, 0x13, &paramVal );
```

See also:

APS_set_axis_param_f();APS_set_axis_param();APS_get_axis_param()

APS_get_system_timer	Get system timer counter

Support Products: PCI-8253/56, PCI-8392 (H), PCI-8254/58 / AMP-204/8C , PCIe-833x

Descriptions:

This function is used to get system timer counter.The counter will count up every cycle time after system is ready. Users can use this function to check if the system is under control or not.

Syntax:

C/C++:

I32 FNTYPE APS_get_system_timer(I32 Board_ID, I32 *Timer);

Visual Basic:

APS_get_system_timer(ByVal Board_ID As Long, Timer As Long) As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

I32 *Timer: return system timer.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 timer;

//Get system timer of board id 0
APS_get_system_timer( 0, &timer );
```

See also:

APS_get_device_info	Get device information

Support Products: PCI-8253/56, PCI-8392 (H), DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-7856, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100 , PCI-8254/58 / AMP-204/8C , PCIe-833x, AMP-104C

Descriptions:

This function is used to get specified device (board) information. The information includes driver version, firmware version, PCB version and so on. Refer to device iformation table.

Syntax:

C/C++

I32 FNTYPE APS_get_device_info(I32 Board_ID, I32 Info_No, I32 *Info);

Visual Basic:

APS_get_device_info(ByVal Board_ID As Long, ByVal Info_No As Long, Info As Long) As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

I32 Info_No: Reference to device iformation table.

I32 *Info: Reference to device iformation table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Info;
ret = APS_get_device_info( 0, 1, &Info );
if( ret != ERR_NoError )
{
	//Show device information.
}
```

See also:

APS_get_first_axisId	Get first axis id of specified board

Support Products: EMX-100, PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get first axis id of specified board.

Syntax:

C/C++:

I32 FNTYPE APS_get_first_axisId(I32 Board_ID, I32 *StartAxisID, I32 *TotalAxisNum);

Visual Basic:

APS_get_first_axisId (ByVal Board_ID As Long, StartAxisID As Long, TotalAxisNum As Long) As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

I32 * StartAxisID: First aixs id of specified board id.

I32 * TotalAxisNum: Total axes on specidied board id

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 StartAxisID;
I32 TotalAxisNum;

//Board id is 0. Get axis info
APS_get_first_axisId ( 0, & StartAxisID, & TotalAxisNum );
```

See also:

APS_save_parameter_to_flash	Save system parameters & axes parameters to flash

Support Products: PCI-8253/56, PCI-8392 (H) , PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to save system parameters and axes parameters to flash. User must set parameters to board then use this function to save all parameters of board to flash.

Syntax:

C/C++:

I32 FNTYPE APS_save_parameter_to_flash(I32 Board_ID);

Visual Basic:

APS_save_parameter_to_flash(ByVal Board_ID As Long)As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
ret = APS_set_board_param ( 0 ,0x0,1); //set EMG LOGIC inverse
ret = APS_save_parameter_to_flash ( 0 );//save board and axis parameter to flash
if( ret == ERR_NoError )
    // Save parameters success.
```

See also:

APS_load_parameter_from_flash(); APS_load_parameter_from_default()

APS_load_parameter_from_flash	Load system parameters & axes parameters from flash

Support Products: PCI-8253/56, PCI-8392 (H) , PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

Load system parameters and axes parameters from flash.

Syntax:

C/C++:

I32 FNTYPE APS_load_parameter_from_flash(I32 Board_ID);

Visual Basic:

APS_load_parameter_from_flash(ByVal Board_ID As Long) As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
ret = APS_load_parameter_from_flash ( 0 );
if( ret == ERR_NoError )
    // Load parameters success.
```

See also:

APS_save_parameter_to_flash(); APS_load_parameter_from_default()

APS_load_parameter_from_default	Load system parameters & axes parameters by default value.

Support Products: PCI-8253/56, PCI-8392(H), EMX-100, PCIe-833x, ECAT-4XMO

Descriptions:

Load default setting to system parameters & axes parameters.

Syntax:

C/C++:

I32 FNTYPE APS_load_parameter_from_default(I32 Board_ID);

Visual Basic:

APS_load_parameter_from_default(ByVal Board_ID As Long)As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
ret = APS_load_parameter_from_default( 0 );
if( ret == ERR_NoError )
    // Load parameters success.
```

See also:

APS_save_parameter_to_flash(); APS_load_parameter_from_flash()

APS_set_security_key	Set security password

Support Products: PCI-8144, PCI-8154/58, AMP-104C

Descriptions:

This function is used to set a security code (16 bits) to EEPROM on controller. Therefore, the security code will never be clear when power is turned off.

Do not use this function frequently. EEPROM ganrentee access 1,000,000 times

Syntax:

C/C++:

I32 FNTYPE APS_set_security_key(I32 Board_ID, I32 OldPassword, I32 NewPassword);

Visual Basic:

APS_set_security_key(ByVal Board_ID As Long, ByVal OldPassword As Long, ByVal NewPassword As Long)As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

I32 OldPassword: Current (Old) password stored in EEPROM. (16 bits)

I32 NewPassword: New password to replace old password. (16 bits)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Ret;
I32 OldPassword = 0x1234;
I32 NewPassword = 0x5678;

Ret = APS_set_security_key(0, OldPassword, NewPassword );
// Check Ret…
```

See also:

APS_check_security_key(); APS_reset_security_key()

APS_check_security_key	Varify security password

Support Products: PCI-8144, PCI-8154/58, AMP-104C

Descriptions:

This function is used to verify the security code which users stored in EEPROM by "APS_set_security_key()".

Syntax:

C/C++:

I32 FNTYPE APS_check_security_key(I32 Board_ID, I32 Password);

Visual Basic:

APS_check_security_key(ByVal Board_ID As Long, ByVal Password As Long) As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

I32 Password: 16 bits password.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Ret;
I32 OldPassword = 0x1234;
I32 NewPassword = 0x5678;

Ret = APS_set_security_key(0, OldPassword, NewPassword );
// Check Ret…

Ret = APS_check_security_key(0, NewPassword );
If( Ret == ERR_NoError )
{
    // Password checking pass.
}else
{
    // Password checking failed.
}
```

See also:

APS_set_security_key(); APS_reset_security_key()

APS_reset_security_key	Reset security password

Support Products: PCI-8144, PCI-8154/58, AMP-104C

Descriptions:

This function is used to reset the security code which stored in EEPROM to default value. The default security code is 0x0000.

Syntax:

C/C++:

I32 FNTYPE APS_reset_security_key(I32 Board_ID);

Visual Basic:

APS_reset_security_key(ByVal Board_ID As Long) As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Ret;
Ret = APS_reset_security_key( 0 );
If( Ret == ERR_NoError ) // Security key reset success.
```

See also:

APS_set_security_key(); APS_check_security_key()

APS_save_param_to_file	Save parameters to file

Support Products: PCI(e)-7856, MNET-4XMO-(C), MNET-1XMO

Descriptions:

This function is used to save axis parameters and board parameters to XML file. When user specifies an existing XML file, those parameters overwrite the specified file. When user inputs a NULL file, the system automatically creates a new XML file and save those parameters to it.

For fieldbus motion series, all axes parameters of different slaves on this fieldbus are saved to xml file. If the quality of communication is unstable, it returns ERR_TimeOut.

Note: Another dynamic dll named "ApsXmlParser.dll" is called when using this function. The dll will be installed into system document after installing SDK.

Note: If user inputs a NULL file, the default name of created XML file is "MotionNetParam.xml" for MotionNet series.

Syntax:

C/C++:

I32 FNTYPE APS_save_param_to_file(I32 Board_ID, const char *pXMLFile);

Visual Basic:

APS_save_param_to_file(ByVal Board_ID As Long , pXMLFile As String) As Long

Parameters:

const char *pXMLFile: Specified an existing XML file which created by MCPro2.exe. Otherwise, input a null file to create automatically a new XML file.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Ret;
I32 BoardID_InBits;
I32 Mode = 0; //By system assigned
I32 BoardID = 0;

APS_initial( &BoardID_InBits, Mode);
//Input an existing file, then overwrite it.
```

```
Ret =
APS_save_param_to_file( BoardID ,“C:\\WINDOWS\\system32\\ApsParameters.xml”);
//Otherwise, Input a NULL file to create a new XML file.
Ret = APS_save_param_to_file( BoardID, NULL );
If( Ret != ERR_NoError )
{ //Error – save parameters to file.}
```

See also:

APS_get_axis_param(); APS_get_board_param()

APS_load_param_from_file	Load parameters from file

Support Products: All products.

Descriptions:

This function is used to load all parameters which are recoded in the input file (XML file). You can use Motion Creator Pro2 utility to create or modify a XML files.

This function will process the XML file with following functions.

APS_set_axis_param()

APS_set_board_param()

APS_set_axis_param_f() (PCIe-8154/8158 and PCI-8254/58 / AMP-204/8C)

When it process an unrecognized parameter or a wrong parameter, the load process will be stopped immediately and return an error. So that the other parameters which after the unrecognized parameter will not be set into the devices. Therefore you must check the file validly before you load into your system.

Syntax:

C/C++:

I32 FNTYPE APS_load_param_from_file(const char *pXMLFile);

Visual Basic:

APS_load_param_from_file(pXMLFile As String) As Long

Parameters:

const char *pXMLFile: Specified a XML file which created by MCPro2.exe.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Ret;
I32 BoardID_InBits;
I32 Mode = 0; //By system assigned

APS_initial( &BoardID_InBits, Mode);
Ret = APS_load_param_from_file( “C:\\WINDOWS\\system32\\ApsParameters.xml” );
If( Ret != ERR_NoError )
```

```
{ //Error load parameters from file.}
```

See also:

APS_set_axis_param(); APS_set_board_param()

APS_register_emx	Register EMX in library

Support Products : EMX-100

Descriptions:

This function is used for user to register EMX series product in APS library before starting the initialization process using APS_initial().

Syntax:

C/C++:

I32 FNTYPE APS_register_emx(I32 emx_online, I32 option);

Visual Basic:

APS_register_emx (emx_online As Long, option As Long) As Long

Parameters:

I32 emx_online: 0: Don't use EMX product; 1: Use EMX product

I32 option: reserved

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
ret = APS_register_emx( 1, 0); // Register EMX series products in APS library
ret = APS_initial( &BoardID_InBits, Mode);
…// Do something
ret = APS_close(); //Close all cards in the system
```

See also:

APS_initial()

APS_get_deviceIP	Ge Device IP

Support Products: EMX-100

Descriptions:

This function is used to get device ip. After executing APS_initial(), user could get each board's ip by passing specified board id.

Syntax:

C/C++:

I32 FNTYPE APS_get_deviceIP(I32 Board_ID, char** ipAddress);

Visual Basic:

APS_get_deviceIP (ByVal Board_ID As Integer, ByRef options As String) As Integer

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

Char** ipAddress: To get EMX ip

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Ret;
char** ip = (char**)malloc(sizeof(char*));
ret = APS_get_deviceIP(Board_ID, ip);
```

See also:

APS_reset_emx_alarm	Reset the alarm signal of device

Support Products: EMX-100

Descriptions:

When servo drives occured alarm, and alarm severity is not critical you can reset the alarm signal by this function..

Syntax:

C/C++:

I32 FNTYPE APS_reset_emx_alarm(I32 Axis_ID);

Visual Basic:

APS_reset_emx_alarm (ByVal Axis_ID As Integer) As Integer

Parameters:

I32 Axis_ID: Number of axis.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Axis_ID = 0;
ret = APS_reset_emx_alarm( Axis_ID);
if( ret != ERR_NoError )
{
   printf(“Reset alarm successful.\n”);
}
```

See also:

APS_get_curr_sys_ctrl_mode	Get current system control mode in FPGA

Support Products: , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get current control mode in FPGA. There are three kinds of control mode. One is pulse mode, another is analog mode and the other is step mode. User could get control mode of specified axis.

Syntax:

C/C++:

I32 FNTYPE APS_get_curr_sys_ctrl_mode(I32 Axis_ID, I32 * Mode);

Visual Basic:

APS_get_curr_sys_ctrl_mode(ByVal Axis_ID As Long, Mode As Long) As Long

Parameters:

I32 Axis_ID:The Axis ID from 0 to 65535.

I32 * Mode: Control mode:

0: pulse mode

1: analog mode

2: step mode

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Mode = 0;

//Get control mode of each axix
APS_get_curr_sys_ctrl_mode( Axis_ID, &Mode );
```

See also:

4. SSCNET function

APS_start_sscnet	Start the network of SSCNET

Support Products: PCI-8392(H)

Descriptions:

This function is used to start SSCNET networking. Once it is started, the SSCNET will start to search the servo drivers connected to the network. It returns axis connecting status inside the bit of the 32-bit value. This function will hold until SSCNET communication established when users issue the function.

Some SSCNET parameter should be set before start the network such as SSCNET cycle time and so on. Please refer to the SSCNET parameter table for the detail description.

Syntax:

C/C++:

I32 FNTYPE APS_start_sscnet(I32 Board_ID, I32 *AxisFound_InBits);

Visual Basic:

APS_start_sscnet (ByVal Board_ID As Long, AxisFound_InBits As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 *AxisFound_InBits: The returned connected axis in bit.

Eg. AxisFound_InBits = 0x111 means Axis switch index: 0, 4 and 8 are connected on line.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “APS168.h”

I32 AxisFound_InBits;
I32 ret;

// Set SSCNET relative parameter befor start sscnet.
```

```
// Start sscnet.
Ret    = APS_start_sscnet( 0, &AxisFound_InBits );
if( ret == ERR_NoError )
{
    // Servo control…
}

// Stop sscnet.
Ret = APS_stop_sscnet( 0 );
```

See also:

APS_stop_sscnet ();APS_set_board_param(); APS_get_board_param()

APS_stop_sscnet	Stop the network of SSCNET

Support Products: PCI-8392(H)

Descriptions:

This function is used to stop SSCNET networking. Once it is stopped, the SSCNET will stop communicating the servo drivers and all servo drivers will be free running after that.

Syntax:

C/C++:

I32 FNTYPE APS_stop_sscnet(I32 Board_ID);

Visual Basic:

APS_stop_sscnet (ByVal Board_ID As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include "APS168.h"

I32 AxisFound_InBits;
I32 ret;

// Set SSCNET relative parameter befor start sscnet.

// Start sscnet.
Ret    = APS_start_sscnet( 0, &AxisFound_InBits );
if( ret == ERR_NoError )
{
    // Servo control…
}

// Stop sscnet.
Ret = APS_stop_sscnet( 0 );
```

See also:

APS_start_sscnet()

APS_get_sscnet_servo_param	Read current servo parameter value

Support Products: PCI-8392(H)

Descriptions:

This function is used to get servo parameters from servo driver. User can read two servo parameters at once. It also can read only one parameter using Para_No1. If users set Para_No2 = 0, Para_dat2 can be set to null.

This function is valid only after SSCNET network is started.

Never try to change parameters which is manufacturer setting.

The definition of servo parameter, please refer to Mitsubishi J3B manual.

Syntax:

C/C++:

I32 FNTYPE APS_get_sscnet_servo_param(I32 Axis_ID, I32 Para_No1, I32 *Para_Dat1, I32 Para_No2, I32 *Para_Dat2);

Visual Basic:

APS_get_sscnet_servo_param(ByVal Axis_ID As Long, ByVal Para_No1 As Long, Para_Dat1 As Long, ByVal Para_No2 As Long, Para_Dat2 As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Para_No1: Servo parameter Number. The parameter meaning, please refer to the manual of servo driver.

Format : 0 x 0 N XX

N : PA : 0, PB : 1, PC : 2, PD : 3

XX: parameter number.

Eg. 0x0107: PB07, 0x000A: PA10, 0x020F

I32 *Para_Dat1:

I32 Para_No2: Servo parameter Number. The parameter meaning, please refer to the manual of servo driver.

Format : 0 x 0 N XX

N : PA : 0, PB : 1, PC : 2, PD : 3

XX: parameter number.

Eg. 0x0107: PB07, 0x000A: PA10, 0x020F

I32 *Para_Dat2: Pointer of I32 variable. When Para_No2 is set to 0, The Para_Dat2 could be set to null (0).

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “APS168.h”
I32 AxisFound_InBits;
I32 ret;
I32 Para_Dat1, Para_Dat2;

// Set SSCNET relative parameter befor start sscnet.

// Start sscnet.
Ret    = APS_start_sscnet( 0, &AxisFound_InBits );
if( ret == ERR_NoError )
{
    // This function is used only when network is established.
    Ret = APS_get_sscnet_servo_param( 0, 0x0107, &Para_Dat1, 0x0108, &Para_Dat2 );
}
```

…

See also:

APS_set_sscnet_servo_param()

APS_set_sscnet_servo_param	Set servo parameter

Support Products: PCI-8392(H)

Descriptions:

This function is used to set servo parameters to servo driver. User can write two servo parameters at once. It also can write only one parameter using Para_No1. If users set Para_No2 = 0, Para_dat2 is meaningless.

This function is valid only after SSCNET network is started.

Some servo parameters change is not allowed after network is started. User should restart the network to make it active.

The definition of servo parameter, please refer to Mitsubishi J3B manual.

Syntax:

C/C++:

I32 FNTYPE APS_set_sscnet_servo_param(I32 Axis_ID, I32 Para_No1, I32 Para_Dat1, I32 Para_No2, I32 Para_Dat2);

Visual Basic:

APS_set_sscnet_servo_param(ByVal Axis_ID As Long, ByVal Para_No1 As Long, ByVal Para_Dat1 As Long, ByVal Para_No2 As Long, ByVal Para_Dat2 As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Para_No1: Servo parameter Number. The parameter meaning, please refer to the manual of servo driver.

Format : 0 x 0 N XX

N : PA : 0, PB : 1, PC : 2, PD : 3

XX: parameter number.

Eg. 0x0107: PB07, 0x000A: PA10, 0x020F

I32 Para_Dat1:

I32 Para_No2: Servo parameter Number. The parameter meaning, please refer to the manual of servo driver.

Format : 0 x 0 N XX

N : PA : 0, PB : 1, PC : 2, PD : 3

XX: parameter number.

Eg. 0x0107: PB07, 0x000A: PA10, 0x020F

I32 Para_Dat2: Servo parameter data. When Para_No2 is set to 0, The Para_Dat2 could be set to null (0).

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “APS168.h”
I32 AxisFound_InBits;
I32 ret;

// Set SSCNET relative parameter befor start sscnet.

// Start sscnet.
Ret   = APS_start_sscnet( 0, &AxisFound_InBits );
if( ret == ERR_NoError )
{
    // This function is used only when network is established.
    Ret = APS_set_sscnet_servo_param( 0, 0x0009, 13, 0, 0 );
    // Check ret for function return success…
}
```

See also:

APS_get_sscnet_servo_param()

APS_get_sscnet_servo_alarm	Get current servo alarm information

Support Products: PCI-8392(H)

Descriptions:

This function is used to get alarm number when servo alarm occurs. The alarm information includes alarm number and alarm detail. Please refer to servo driver manual for the detail description.

When servo alarm occurred, user should use this function before reset alarm otherwise the alarm information will be reset.

Syntax:

C/C++:

I32 FNTYPE APS_get_sscnet_servo_alarm(I32 Axis_ID, I32 *Alarm_No, I32 *Alarm_Detail);

Visual Basic:

APS_get_sscnet_servo_alarm(ByVal Axis_ID As Long, Alarm_No As Long, Alarm_Detail As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Alarm_No: Alarm number. Please refer to servo driver manual.

I32 *Alarm_Detail: Alarm detail. Please refer to servo driver manual.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Alarm_No;
I32 Alarm_Detail;

…//Alarm occurred!
APS_get_sscnet_servo_alarm(Axis_ID, &Alarm_No, &Alarm_Detail ); //Get alarm operation159i
…//Remove the alarm cause
APS_reset_sscnet_servo_alarm(Axis_ID ); //Reset servo alarm
...
```

See also:

APS_reset_sscnet_servo_alarm()

APS_reset_sscnet_servo_alarm	Servo alarm reset

Support Products: PCI-8392(H)

Descriptions:

When servo alarm occurs, servo motor will stop moving. After the alarm condition passed, this function can help to clear alarm and reset servo.

Syntax:

C/C++:

I32 FNTYPE APS_reset_sscnet_servo_alarm(I32 Axis_ID);

Visual Basic:

APS_reset_sscnet_servo_alarm(ByVal Axis_ID As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Alarm_No;
I32 Alarm_Detail;

…//Alarm occurred!
APS_get_sscnet_servo_alarm(Axis_ID, &Alarm_No, &Alarm_Detail ); //Get alarm
operation161i
…//Remove the alarm cause
APS_reset_sscnet_servo_alarm(Axis_ID ); //Reset servo alarm...
```

See also:

APS_get_sscnet_servo_alarm()

APS_save_sscnet_servo_param	Save servo parameter to flash

Support Products: PCI-8392(H)

Descriptions:

This function is used to save servo parameters from SDRAM to flash memory on the controller card.

When system (Controller) is power on, it copies servo parameters from flash or from default table to SDRAM. The servo parameters will be transferred to servo drivers when SSCNET network is established. Users can choose the other mode from axis parameters which servo drivers remain its settings when network is established. The parameter is remained default if the Axis is null (The axis ID doesn't be used).

Servo parameters of all axes (16 axes) will be saved at once when you issue this function. You cannot save every servo driver's parameter separately.

Syntax:

C/C++:

I32 FNTYPE APS_save_sscnet_servo_param(I32 Board_ID);

Visual Basic:

APS_save_sscnet_servo_param(ByVal Board_ID as Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
// Config servo parameter
// APS_set_sscnet_servo_param …
APS_save_sscnet_servo_param( Board_ID ); //Save servo parameter to flash.
```

See also:

APS_set_sscnet_servo_param(); APS_get_sscnet_servo_param()

APS_get_sscnet_servo_abs_position	Get absolute reference position from servo driver

Support Products: PCI-8392(H)

Descriptions:

This function is used to get absolute position from SSCNET servo driver. This function can be issued only when SSCNET network is started. Normally, in order to establish ABS position system, users must perform a home return operation first then users must issue this function to get absolute position from servo driver. In the meantime, controller will copy the absolute position of servo drive to axis parameters. Finally, users can use APS_save_sscnet_servo_abs_position() to save all axes' ABS information on flash memory for next time use.

Axis parameter define
PRA_SSC_SERVO_ABS_CYC_CNT
PRA_SSC_SERVO_ABS_RES_CNT

The details of axis parameter please refer to axis parameter table.

Syntax:

C/C++:

I32 FNTYPE APS_get_sscnet_servo_abs_position(I32 Axis_ID, I32 *Cyc_Cnt, I32 *Res_Cnt);

Visual Basic:

APS_get_sscnet_servo_abs_position(ByVal Axis_ID As Long, Cyc_Cnt As Long, Res_Cnt As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Cyc_Cnt: Cycle counter of servo driver

I32 *Res_Cnt: Resolution counter of servo driver.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

//1. Initial card and start SSCNET network

//2. Perform a home return operation

Ret = APS_get_sscnet_servo_abs_position(Axis_ID, Cyc_Cnt, Res_Cnt);

// Record the abs. position data for next homing operation.

See also:

APS_save_sscnet_servo_abs_position();APS_load_sscnet_servo_abs_position();
APS_set_axis_param();APS_get_axis_param()

APS_save_sscnet_servo_abs_position	Save absolute reference position to flash ROM

Support Products: PCI-8392(H)

Descriptions:

This function is used to save absolute position from axis parameter to flash memory. Normally, in order to establish absolute position system, users must do home procedure first. Then use ”APS_get_sscnet_servo_abs_position” function to get the absolute position from driver. Finally, users must call this function to save all absolute position of axes to flash memory for next time use.

Notice that servo parameters of all axes (16 axes) will be saved at once when users issue this function. You cannot save each servo driver separately.

Axis parameter define
PRA_SSC_SERVO_ABS_CYC_CNT
PRA_SSC_SERVO_ABS_RES_CNT

Syntax:

C/C++:

I32 FNTYPE APS_save_sscnet_servo_abs_position(I32 Board_ID);

Visual Basic:

APS_save_sscnet_servo_abs_position(ByVal Board_ID As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
//1. Initial card and start SSCNET network
//2. Perform home return operations.
//3. Get abs position for servo drivers.
For(Axis_ID = 0; Axis_ID < 16; Axis_ID ++ )
{
      Ret = APS_get_sscnet_servo_abs_position(   Axis_ID, Cyc_Cnt, Res_Cnt );
}
```

Ret = APS_save_sscnet_servo_abs_position(Board_ID); //Save all abs. position to flash memory.

…

See also:

APS_get_sscnet_servo_abs_position();APS_load_sscnet_servo_abs_position();
APS_set_axis_param(); APS_get_axis_param()

APS_load_sscnet_servo_abs_position	Load absolute reference position from flash ROM

Support Products: PCI-8392(H)

Descriptions:

This function is used to load servo absolute position from flash memory to axis parameter. If user has never saved servo absolute position, calling this function will return error.

User can load all ABS position at once by specified function perameter “Option” for convenient purpose. Refer to parameter description.

Normally, if users want to use ABS position system, they will use this function to load ABS information from flash to axis parameters before SSCNET network is established. Also need to set ABS position system enable in axis parameter before SSCNET network is established.

Syntax:

C/C++:

I32 FNTYPE APS_load_sscnet_servo_abs_position(I32 Axis_ID, I32 Option, I32 *Cyc_Cnt, I32 *Res_Cnt);

Visual Basic:

APS_load_sscnet_servo_abs_position(ByVal Axis_ID As Long, ByVal Option As Long, Cyc_Cnt As Long, Res_Cnt As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535

I32 Option: Load option.

0: Load one axis’ ABS position to axis parameter

1: Load all axes’ ABS positions to axes parameters.

I32 *Cyc_Cnt: Get cycle counter from flash memory. Set this parameter 0 to ignore.

I32 *Res_Cnt: Get resolution counter from flash memory. Set this parameter 0 to ingnor.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
//1. Initial card
//2. load abs. position from flash memory.
Ret = APS_load_sscnet_servo_abs_position(Axis_ID, 1, 0 ,0); //Option = 1 load all axes
```

APS_set_axis_param(Axis_ID, PRA_SSC_SERVO_ABS_POS_OPT, 1); //Enable abs. position system.
APS_start_sscnet(Board_ID, &AxisFound_InBits); //Start SSCNET network.
// Go to home position by absolute move function.

See also:

APS_get_sscnet_servo_abs_position();APS_save_sscnet_servo_abs_position();
APS_set_axis_param();APS_get_axis_param()

APS_get_sscnet_link_status	Get SSCNET link status

Support Products: PCI-8392(H)

Descriptions:

This function is used to get SSCNET link staus. You can easily use this function to check SSCNET connection is linked or not.

Syntax:

C/C++:

I32 FNTYPE APS_get_sscnet_link_status(I32 Board_ID, I32 *Link_Status);

Visual Basic:

APS_get_sscnet_link_status(ByVal Board_ID As Long, Link_Status As Long) As Long

Parameters:

I32 Board_ID: Board ID, zero base parameter.

I32 *Link_Status: Link status.

Return 1 : SSCNET is linked

Return 0 : SSCNET is not linked.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “APS168.h”
#include “ErrorCodeDef.h”
I32 link; //Get SSCNET link status.
I32 err;
// Start SSCNET.
//Check SSCNET link status.
Do{
    err   = APS_get_sscnet_link_status( 0, &link );
    if( link == 0 )
    {
        // Connection is broken.
        Break;
    }
```

```
}while( err == ERR_NoError )
```

See also:

APS_set_sscnet_servo_monitor_src	Set servo monitor data source

Support Products: PCI-8392(H)

Descriptions:

This function is used to set the source of each servo monitor channel.

In SSCNETIII controller, each axis has 4 channels which can be used to monitor SSCNET servo driver status. You could change monitor source by this function. The monitor sources please refer SSCNET servo monitor source table. In addition, you can get monitor data by "*APS_get_sscnet_servo_monitor_data*()".

This function is valid when SSCNET communication is connected.

Syntax:

C/C++:

I32 FNTYPE APS_set_sscnet_servo_monitor_src(I32 Axis_ID, I32 Mon_No, I32 Mon_Src);

Visual Basic:

APS_set_sscnet_servo_monitor_src(ByVal Axis_ID As Long, ByVal Mon_No As Long, ByVal Mon_Src As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Mon_No: Monitor channel number. 0~3 refer to channel 0 ~ channel 3.

I32 Mon_Src: Monitor source number. Please refer to SSCNET servo monitor source table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include "APS168.h"
#include "ErrorCodeDef.h"

// Initial APSLibrary and start SSCNET first.
{
    I32 ret;
    I32 Axis_ID = 0;
```

```
    ret = APS_set_sscnet_servo_monitor_src( Axis_ID, 0, 1 ); //Set channel 0, source = 1.
    //Check ret.
    Ret = APS_set_sscnet_servo_monitor_src( Axis_ID, 1, 2 ); //Set channel 1, source = 2.
    //Check ret.
}
```

See also:

APS_get_sscnet_servo_monitor_src(); APS_get_sscnet_servo_monitor_data()

APS_get_sscnet_servo_monitor_src	Get servo monitor data source

Support Products: PCI-8392(H)

Descriptions:

This function is used to get the source of each servo monitor channel.

In SSCNETIII controller, each axis has 4 channels which can be used to monitor SSCNET servo driver status. You could get monitor source by this function. The monitor sources please refer SSCNET servo monitor source table.

This function is valid when SSCNET communication is connected.

Syntax:

C/C++:

I32 FNTYPE APS_get_sscnet_servo_monitor_src(I32 Axis_ID, I32 Mon_No, I32 *Mon_Src);

Visual Basic:

APS_get_sscnet_servo_monitor_src(ByVal Axis_ID As Long, Mon_No As Long, ByVal Mon_Src As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Mon_No: Monitor channel number. 0~3 refer to channel 0 ~ channel 3.

I32 *Mon_Src: Return monitor source number. Please refer to SSCNET servo monitor source table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “APS168.h”
#include “ErrorCodeDef.h”

// Initial APSLibrary and start SSCNET first.
{
    I32 ret;
    I32 Axis_ID = 0;
    I32 Mon_Src;
```

```
    ret = APS_get_sscnet_servo_monitor_src( Axis_ID, 0, &Mon_Src );
    //Check ret.
    Ret = APS_get_sscnet_servo_monitor_src( Axis_ID, 1, &Mon_Src );
    //Check ret.
}
```

See also:

APS_set_sscnet_servo_monitor_src();APS_get_sscnet_servo_monitor_data()

APS_get_sscnet_servo_monitor_data	Get servo monitor data

Support Products: PCI-8392(H)

Descriptions:

This function is used to get sscnet servo monitor data. This function can be used only when SSCNET is connected.

In SSCNETIII controller, each axis has 4 channels which can be used to monitor SSCNET servo driver status. You can use this function to get all (4 channels) monitor data at once. In addition, you could change monitor source by the function

" *APS_set_sscnet_servo_monitor_src*()". Monitor sources please refer SSCNET servo monitor source table.

Syntax:

C/C++:

I32 FNTYPE APS_get_sscnet_servo_monitor_data(I32 Axis_ID, I32 Arr_Size, I32 *Data_Arr);

Visual Basic:

APS_get_sscnet_servo_monitor_data(ByVal Axis_ID As Long, ByVal Arr_Size As Long, Data_Arr As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Arr_Size: Specifiy data array size. Min:1 ~ Max:4.

I32 *Data_Arr: Get monitor data array. The array size is according to "Arr_Size".

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include "APS168.h"
#include "ErrorCodeDef.h"

// Initial APSLibrary and start SSCNET first.
{
    I32 Axis_ID = 0; //Axis ID
    I32 Data_Arr[4]; //Total 4 channels
    I32 ret; //Return code.
```

```
    // Get SSCNET monitor data.
    Ret = APS_get_sscnet_servo_monitor_data(Axis_ID, 4, Data_Arr );
    if( ret == ERR_NoError )
    {    //Show Data_Arr[];
    }
}
```

See also:

APS_set_sscnet_servo_monitor_src();APS_get_sscnet_servo_monitor_src();

5. Motion IO and motion status

APS_motion_status	Return motion status

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8144, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100 , PCI-8254/58 / AMP-204/8C , PCIe-833x, AMP-104C, ECAT-4XMO

Descriptions:

This function is used to get one axis' motion status. The status includes running, normal stop, abnormal stop by reasons, in waiting other axis, follow status, in some modes, in accelerating or decelerating and so on. Status can be more than two such like mode and running. Users need to use this function to check whether the 'Fire-and-forget' function is done in polling system. In even driven system, users can use interrupt event functions.

Please refer to the motion status table for detail description.

Syntax:

C/C++:

I32 FNTYPE APS_motion_status(I32 Axis_ID);

Visual Basic:

APS_motion_status (ByVal Axis_ID As Long) As Long

Parameters:

I32 Axis_ID:The Axis ID from 0 to 65535

Return Values:

Positive value:

The value of motion status. Please refer to motion status bit number definition table for the value meaning

Negative value:

Error Code: Please refer to error code table.

Example:

```
I32 MotionStatus;
MotionStatus = APS_motion_status( Axis_ID ); //Get Motion status
…
```

See also:

APS_motion_io_status();

APS_motion_io_status	Return motion IO status

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8144, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100 , PCI-8254/58 / AMP-204/8C , PCIe-833x, AMP-104C, ECAT-4XMO

Descriptions:

This function is used to get one axis' motion I/O information like ORG, PEL, MEL, SVON, INP and so on. These statuses are connected to external switched or servo drivers.

Please refer to the motion IO status table for detail description.

Syntax:

C/C++:

I32 FNTYPE APS_motion_io_status(I32 Axis_ID);

Visual Basic:

APS_motion_io_status (ByVal Axis_ID As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535

Return Values:

Positive value:

The value of motion IO status, please refer to motion IO status bit number definition table for the value meaning

Negative value:

Error Code: Please refers to error code table.

Example:

```
I32 MotionIO;
MotionIO = APS_motion_io_status(Axis_ID ); //Get Motion IO status
…
```

See also:

APS_motion_status ();

APS_set_servo_on	Set servo ON/OFF

Support Products: PCI-8253/56, PCI-8392(H) , MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100 , PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to command servo driver of specified axis to starts controlling its servomotor. Then motion function could be applied on this axis.

Syntax:

C/C++:

I32 FNTYPE APS_set_servo_on(I32 Axis_ID, I32 Servo_on);

Visual Basic:

APS_set_servo_on (ByVal Axis_ID As Long, ByVal ServoOn As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535

I32 Servo_on:

0: Servo OFF, 1: Servo ON

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

…//Initialization

APS_set_servo_on(Axis_ID, 1); // Set servo ON

… //Motion action

APS_set_servo_on(Axis_ID, 0); //Set servo OFF

…//Release

See also:

APS_get_position	Get feedback position

Support Products: PCI-8253/56, PCI-8392(H) , MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get the position counter of one axis. The counter is in unit of pulse.

Syntax:

C/C++:

I32 FNTYPE APS_get_position(I32 Axis_ID, I32 *Position);

Visual Basic:

APS_get_position (ByVal Axis_ID As Long, Position As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Position: Feedback position. Unit in pulse

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Position;
APS_get_position(Axis_ID, &Position ); //Get feedback position
…
```

See also:

APS_get_command(); APS_set_position(); APS_set_command()

APS_set_position	Set feedback position

Support Products: PCI-8253/56, PCI-8392(H) , MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to set the position counter of one axis. The counter is in unit of pulse. It assigns a new position at instance but the motor will not move due to this function.

Syntax:

C/C++:

I32 FNTYPE APS_set_position(I32 Axis_ID, I32 Position);

Visual Basic:

APS_set_position (ByVal Axis_ID As Long, ByVal Position As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Position: Set feedback position. Unit in pulse.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

…

APS_set_position(Axis_ID, 0); // Set feedback position to zero

See also:

APS_get_position(); APS_get_command(); APS_set_command()

APS_get_command	Get command position

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8144, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100 , PCI-8254/58 / AMP-204/8C, AMP-104C

Descriptions:

This function is used to get the command counter of one axis. The counter is in unit of pulse.

Syntax:

C/C++:

I32 FNTYPE APS_get_command(I32 Axis_ID, I32 *Command);

Visual Basic:

APS_get_command (ByVal Axis_ID As Long, Command As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Command: Command position. Unit in pulse.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 Command ;

APS_get_command(Axis_ID, &Command); //Get command position.

…//

See also:

APS_get_position(); APS_set_position(); APS_set_command()

APS_set_command	Set command position

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8144, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100 , PCI-8254/58 / AMP-204/8C, AMP-104C

Descriptions:

This function is used to set the command counter of one axis. The counter is in unit of pulse. It assigns a new command counter at instance but the motor will not move due to this function.

Syntax:

C/C++:

I32 FNTYPE APS_set_command(I32 Axis_ID, I32 Command);

Visual Basic:

APS_set_command (ByVal Axis_ID As Long, ByVal Command As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Command: Position command. Unit in pulse.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

…//

APS_set_command(Axis_ID, 0); //Set command position to zero.

See also:

APS_get_position(); APS_get_command(); APS_set_position();

APS_get_command_velocity	Get command velocity

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8144, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100 , PCI-8254/58 / AMP-204/8C, AMP-104C

Descriptions:

This function is used to get command velocity. The minimum value depends on speed calculation resolution of system.

Syntax:

C/C++:

I32 FNTYPE APS_get_command_velocity(I32 Axis_ID, I32 *Velocity);

Visual Basic:

APS_get_command_velocity(ByVal Axis_ID As Long, Velocity As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Velocity: Return command velocity. Unit: pps

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Axis_ID = 0;
I32 Velocity;
ret = APS_get_command_velocity ( Axis_ID, &Velocity);
if( ret == ERR_NoError )
{
	//Velocity
}
```

See also:

APS_get_position(); APS_get_command();APS_get_feedback_velocity()

APS_get_feedback_velocity	Get feedback velocity

Support Products: PCI-8253/56, PCI-8392(H) , EMX-100 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get feedback velocity. The minimum value depends on speed calculation resolution of system.

Syntax:

C/C++:

I32 FNTYPE APS_get_feedback_velocity(I32 Axis_ID, I32 *Velocity);

Visual Basic:

APS_get_feedback_velocity(ByVal Axis_ID As Long, Velocity As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Velocity: Return feedback velocity. Unit: pps

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Axis_ID = 0;
I32 Velocity;
ret = APS_get_feedback_velocity( Axis_ID, &Velocity);
if( ret == ERR_NoError )
{
	//Velocity
}
```

See also:

APS_get_position(); APS_get_command(); APS_get_command_velocity ();

APS_get_error_position	Get error position

Support Products: PCI-8253/56, PCI-8392(H) , MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get error position value. This value is defined as command minus feedback position

Syntax:

C/C++:

I32 FNTYPE APS_get_error_position(I32 Axis_ID, I32 *Err_Pos);

Visual Basic:

APS_get_error_position(ByVal Axis_ID As Long, Err_Pos As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Err_Pos: Return error position.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Axis_ID = 0;
I32 Err_Pos;
ret = APS_get_error_position(Axis_ID, &Err_Pos );
if( ret == ERR_NoError )
    //Show error position.
```

See also:

APS_get_position();APS_get_command();APS_get_command_velocity ();APS_get_feedback_velocity()

APS_get_target_position	Get target position

Support Products: PCI-8253/56, PCI-8392(H) , MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get target position record. In linear positioning mode, the value is target position. In circular positioning mode, the value is the same as command. In velocity and jog mode, the value is the same as command.

For EMX-100:

This function is used to get single axis target position of point-to-point move and linear move. And target position will not be updated when using Jog move, Velocity move, and Home move.

Syntax:

C/C++:

I32 FNTYPE APS_get_target_position(I32 Axis_ID, I32 *Targ_Pos);

Visual Basic:

APS_get_target_position(ByVal Axis_ID As Long, Targ_Pos As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Targ_Pos: Return target position.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Axis_ID = 0;
I32 Targ_Pos;
ret = APS_get_target_position(Axis_ID, &Targ_Pos );
if( ret == ERR_NoError )
    //Show target position.
```

See also:

APS_get_position();APS_get_command();APS_get_command_velocity ();APS_get_feedback_velocity()

APS_get_position_f	Get feedback position by double

Support Products: MNET-4XMO-(C), MNET-1XMO, PCI(e)-8154/8158, PCI-8102/58A , PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get the position counter of one axis by double. The counter is in unit of pulse.

Syntax:

C/C++:

I32 FNTYPE APS_get_position_f(I32 Axis_ID, F64 *Position);

Visual Basic:

APS_get_position_f(ByVal Axis_ID As Long, Position As Double) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

F64 *Position: Feedback position. Unit in pulse

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
F64 Position;

APS_get_position_f(Axis_ID, &Position ); //Get feedback position
…
```

See also:

APS_get_command_f(); APS_set_position_f(); APS_set_command_f()

APS_set_position_f	Set feedback position by double

Support Products: MNET-4XMO-(C), MNET-1XMO, PCI(e)-8154/8158, PCI-8102/58A , PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set the position counter of one axis by double. The counter is in unit of pulse. It assigns a new position at instance but the motor will not move due to this function.

Syntax:

C/C++:

I32 FNTYPE APS_set_position_f(I32 Axis_ID, F64 Position);

Visual Basic:

APS_set_position_f(ByVal Axis_ID As Long, ByVal Position As Double) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

F64 Position: Set feedback position. Unit in pulse.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

…

APS_set_position_f(Axis_ID, 0.0); // Set feedback position to zero

See also:

APS_get_position_f(); APS_get_command_f(); APS_set_command_f()

APS_get_command_f	Get command position by double

Support Products: MNET-4XMO-(C), MNET-1XMO, PCI(e)-8154/8158, PCI-8102/58A , PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get the command counter of one axis by double. The counter is in unit of pulse.

Syntax:

C/C++:

I32 FNTYPE APS_get_command_f(I32 Axis_ID, F64 *Command);

Visual Basic:

APS_get_command_f(ByVal Axis_ID As Long, Command As Double) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

F64 *Command: Command position. Unit in pulse.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

F64 Command ;

APS_get_command_f(Axis_ID, &Command); //Get command position by double

…//

See also:

APS_get_position_f(); APS_set_position_f(); APS_set_command_f()

APS_set_command_f	Set command position by double

Support Products: MNET-4XMO-(C), MNET-1XMO, PCI(e)-8154/8158, PCI-8102/58A, PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set the command counter of one axis by double. The counter is in unit of pulse. It assigns a new command counter at instance but the motor will not move due to this function.

Syntax:

C/C++:

I32 FNTYPE APS_set_command_f(I32 Axis_ID, F64 Command);

Visual Basic:

APS_set_command_f(ByVal Axis_ID As Long, ByVal Command Double) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

F64 Command: Position command. Unit in pulse.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

…//

APS_set_command_f(Axis_ID, 0.0); //Set command position to zero.

See also:

APS_get_position_f(); APS_get_command_f(); APS_set_position_f();

APS_get_target_position_f	Get target position by double

Support Products: MNET-4XMO-(C), MNET-1XMO, PCI(e)-8154/8158, PCI-8102/58A, PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get target position record by double. In linear positioning mode, the value is target position. In circular positioning mode, the value is the same as command. In velocity and jog mode, the value is the same as command.

Syntax:

C/C++:

I32 FNTYPE APS_get_target_position_f(I32 Axis_ID, F64 *Targ_Pos);

Visual Basic:

APS_get_target_position_f(ByVal Axis_ID As Long, Targ_Pos As Double) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

F64 *Targ_Pos: Return target position.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Axis_ID = 0;
F64 Targ_Pos;
ret = APS_get_target_position_f(Axis_ID, &Targ_Pos );
if( ret == ERR_NoError )
    //Show target position.
```

See also:

APS_get_position_f();APS_get_command_f();APS_get_command_velocity_f ();APS_get_feedback_velocityf()

APS_get_error_position_f	Get error position by double

Support Products: MNET-4XMO-(C), MNET-1XMO, PCI(e)-8154/8158, PCI-8102/58A, PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get error position record by double. This value is defined as command minus feedback position.

Syntax:

C/C++:

I32 FNTYPE APS_get_error_position_f(I32 Axis_ID, F64 *Err_Pos);

Visual Basic:

APS_get_error_position_f(ByVal Axis_ID As Long, Err_Pos As Double) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

F64 *Err_Pos: Return error position.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Axis_ID = 0;
F64 Err_Pos;
ret = APS_get_error_position_f(Axis_ID, &Err_Pos );
if( ret == ERR_NoError )
    //Show error position.
```

See also:

APS_get_position_f();APS_get_command_f();APS_get_command_velocity_f();APS_get_feedback_velocityf();APS_get_target_position_f

APS_get_command_velocity_f	Get command velocity by double

Support Products: MNET-4XMO-(C), MNET-1XMO, PCI(e)-8154/8158, PCI-8102/58A, PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get command velocity by double. The minimum value depends on speed calculation resolution of system.

Syntax:

C/C++:

I32 FNTYPE APS_get_command_velocity_f(I32 Axis_ID, F64 *Velocity);

Visual Basic:

APS_get_command_velocity_f(ByVal Axis_ID As Long, Velocity As Double) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

F64 *Velocity: Return command velocity. Unit: pps

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Axis_ID = 0;
F64 Velocity;
ret = APS_get_command_velocity_f ( Axis_ID, &Velocity);
if( ret == ERR_NoError )
{
    //Velocity
}
```

See also:

APS_get_position_f(); APS_get_command_f();APS_get_feedback_velocityf()

APS_get_feedback_velocity_f	Get feedback velocity by double

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get feedback velocity by double. The minimum value depends on speed calculation resolution of system.

Syntax:

C/C++:

I32 FNTYPE APS_get_feedback_velocity_f(I32 Axis_ID, F64 *Velocity);

Visual Basic:

APS_get_feedback_velocity_f(ByVal Axis_ID As Long, Velocity As Double) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

F64 *Velocity: Return feedback velocity. Unit: pps

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Axis_ID = 0;
F64 Velocity;
ret = APS_get_feedback_velocity_f ( Axis_ID, &Velocity);
if( ret == ERR_NoError )
{}
```

See also:

APS_get_position_f(); APS_get_command_f();APS_get_command_velocityf()

APS_get_mq_free_space	Get current free space of motion queue

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get current free space of motion queue.

Each axis has own motion queue (FIFO) to buffer the motion commands.

Syntax:

C/C++:

I32 FNTYPE APS_get_mq_free_space(I32 Axis_ID, I32 *Sapce);

Visual Basic:

APS_get_mq_free_sapce (ByVal Axis_ID As Long, Sapce As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Sapce: Free space of motion queue of specified axis.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Space;
APS_get_mq_free_space(Axis_ID, &Space ); //Get free space of motion queue
…//
```

See also:

APS_get_mq_usage();

APS_get_mq_usage	Get current usage from motion queue

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get current usage from motion queue.

Each axis has own motion queue (FIFO) to buffer the motion commands.

Syntax:

C/C++:

I32 FNTYPE APS_get_mq_usage(I32 Axis_ID, I32 *Usage);

Visual Basic:

APS_get_mq_usage(ByVal Axis_ID As Long, Usage As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Usage: The usage of motion queue of specified axis.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Usage;
APS_get_mq_free_space(Axis_ID, &Usage ); //Get usage of motion queue
…//
```

See also:

APS_get_mq_free_space();

APS_get_stop_code	Get stop code by axis

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get stop code. The stop code is a stop reason for an axis when a move is stopping.

The possible stop code is shown in following table. If axis performs a PTP move of single axis and stops normally, you will get a stop code belong to STOP_NORMAL.

Symbol	Code	Description
STOP_NORMAL	0	Stop normally
STOP_EMG	1	Stop when EMG is turn ON
STOP_ALM	2	Stop when ALM is turn ON
STOP_SVNO	3	Stop when servo is turn-OFF
STOP_PEL	4	Stop by PEL signal turn ON
STOP_MEL	5	Stop by MEL signal turn ON
STOP_SPEL	6	Stop by soft-limit condition – plus end limit
STOP_SMEL	7	Stop by soft-limit condition – minus end limit
STOP_USER_EMG	8	EMG stop by user
STOP_USER	9	Stop by user
STOP_GAN_L1	10	Stop by E-Gear gantry protect level 1 condition is met.
STOP_GAN_L2	11	Stop by E-Gear gantry protect level 2 condition is met
STOP_GEAR_SLAVE	12	Stop because gear slave axis
STOP_ERROR_LEVEL	13	Error position check level
STOP_DI	14	DI
STOP_ERROR_ECAT_HOME	15	Stop by EtherCAT home error.

Note: If motion is multi-axes motion (Interpolation), the stop code is only updated on the reference axis. Other axes keep previous stop code.

Syntax:

C/C++:

I32 FNTYPE APS_get_stop_code(I32 Axis_ID, I32 *Code);

Visual Basic:

APS_get_stop_code(ByVal Axis_ID As Long, Code As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 * Code: Stop code (stop reason).

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Code;
I32 Axis_ID;

APS_get_stop_code(Axis_ID, &Code ); //Get stop code
…//
```

See also:

APS_get_encoder	Get raw encoder counter

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO, AMP-104C

Descriptions:

This function is used to get raw encoder counter of one axis. It is read only and applied to debug or monitor raw counter.

For PCIe-833x series products, this value is mapping to ECAT PDO actual position value (OD 0x6064).

Syntax:

C/C++:

I32 FNTYPE APS_get_encoder(I32 Axis_ID, I32 *Encoder);

Visual Basic:

APS_get_encoder(ByVal Axis_ID As Long, Encoder As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Encoder: Raw encoder counter. Unit in pulse.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Encoder;
APS_get_encoder(Axis_ID, &Encoder ); //Get raw encoder counter.
…//
```

See also:

APS_get_command_counter	Get raw command counter

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get raw command counter of one axis. It is read only and applied to debug or monitor raw counter.

For PCIe-833x series products, this value is mapping to ECAT PDO target position (OD 0x607A).

Syntax:

C/C++:

I32 FNTYPE APS_get_command_counter(I32 Axis_ID, I32 *Counter);

Visual Basic:

APS_get_command_counter(ByVal Axis_ID As Long, Counter As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Counter: Raw command counter. Unit in pulse.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 Counter;

APS_get_command_counter(Axis_ID, & Counter); //Get raw command counter

…//

See also:

APS_reset_command_counter	Reset raw command counter

Support Products: PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to reset raw command counter (copy raw encoder counter to raw command counter) of one axis.

Syntax:

C/C++:

I32 APS_ reset _command_counter(I32 Axis_ID);

Visual Basic:

APS_ reset _command_counter(ByVal Axis_ID As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 Counter;

APS_ reset _command_counter(Axis_ID, & Counter); //Reset raw command counter

…//

See also:

APS_get_actual_torque	Get actual torque value

Support Products: PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get actual torque value from device.

Syntax:

C/C++:

I32 FNTYPE APS_get_actual_torque(I32 Axis_ID, I32 *Torque)

Visual Basic:

APS_get_actual_torque(ByVal Axis_ID As Integer, ByRef Torque As Integer) As Integer

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Torque: The actual torque value from device.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_get_axis_latch_data	Get ORG/EZ latch data

Support Products: PCI-8254/58 / AMP-204/8C,

Descriptions:

Users can use this function to get latch data from the input signal belongs to an axis, such has ORG and EZ. This function's behavior for getting latch data is read clear which means when users call this function, the latched data will be retrieved and latch counter will be clear to zero for next latch. When users know there is a data latched, users can call this function to get the most update latch data from motor encoder.

Syntax:

C/C++:

I32 FNTYPE APS_get_axis_latch_data(I32 Axis_ID, I32 latch_channel, I32 *latch_data)

Visual Basic:

APS_get_axis_latch_data(ByVal Axis_ID As Long, ByVal latch_data As Long, latch_data As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to maximum axis number in one system.

I32 latch_channel: for channel selection: 0 for ORG and 1 for EZ

I32 *latch_data: a data pointer to get motor encoder latch data

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

6. Single axis motion

APS_relative_move	Begin a relative distance move

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8144, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100, PCI-8254/58 / AMP-204/8C, AMP-104C

Descriptions:

This function is used to start a single axis relative motion. Although there is maximum speed setting in function parameter, the traveling distance and accelerating rate may not be enough due to user's setting to reach the maximum speed. The speed profile's acceleration and deceleration rate and curve are set by axis parameter function.

This function is 'fire-and-forget' type. That means user's program or procedure will not be pended during axis traveling. Users must use motion status checking function or interrupt event waiting function to wait it done.

For PCI-8253/56, PCI-8392(H), PCI-8254/58 / AMP-204/8C , users can start a new move command including stop command to override the previous one during the axis traveling. The axis will be switched to new command immediately according to new setting of target position, new speed.

This command can't be overridden by other motion modes like Jog, home, manual pulse generation, contour motion. Users must stop axis motion before switching to those modes mentioned above.

For EMX-100 , this function is used for single axis point-to-point motion using relative position. Two speed profile parameters distance and maximum velocity are given by user, and other parameters like start velocity, acceleration rate, deceleration rate and s-factor are configurable by axis parameter table. The actual command velocity may not reach maximum velocity due to small traveling distance or accelerating rate are given.

This function uses 'fire-and-forget' mode to avoid blocking user's program or procedure during axis traveling. Users can read motion status MDN to check the motion is completed (MDN = 1) or not (MDN = 0). Except stop command, users CAN NOT start any new move command before previous motion is completed.

Syntax:

C/C++:

I32 FNTYPE APS_relative_move(I32 Axis_ID, I32 Distance, I32 Max_Speed);

Visual Basic:

APS_relative_move (ByVal Axis_ID As Long, ByVal Distance As Long, ByVal Max_Speed As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Distance: Relative distance. Unit is pulse.

I32 Max_Speed: The maximum speed of this move profile. Unit: pulse/sec.

For EMX-100: I32 Max_Speed: The maximum speed of this move profile, its range is 1 ~ 8,000,000 (Unit: pulse/sec).

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
APS_set_axis_param(Axis_ID, PRA_ACC, 1000000 ); //Set acceleration rate
APS_set_axis_param(Axis_ID, PRA_DEC, 1000000 ); //Set deceleration rate
//Execute a relative move.
APS_relative_move( Axis_ID, 10000, 10000 );
```

See also:

APS_relative_move();APS_absolute_move();APS_velocity_move();APS_home_move(); APS_stop_move(); APS_emg_stop()

APS_absolute_move	Begin a absolute position move

Support Products: PCI-8253/56, PCI-8392(H) , MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100, PCI-8254/58 / AMP-204/8C, AMP-104C

Descriptions:

This function is used to start a single axis absolute positioning motion. Although there is maximum speed setting in function parameter, the traveling distance and accelerating rate may not be enough due to user's setting to reach the maximum speed. The speed profile's acceleration and deceleration rate and curve are set by axis parameter function.
This function is 'fire-and-forget' type. That means user's program or procedure will not be pended during axis traveling. Users must use motion status checking function or interrupt event waiting function to wait it done.

For PCI-8253/56, PCI-8392(H) , PCI-8254/58 / AMP-204/8C , users can start a new move command including stop command to override the previous one during the axis traveling. The axis will be switched to new command immediately according to new setting of target position, new speed.
This command can't be overridden by other motion modes like Jog, home, manual pulse generation, contour motion. Users must stop axis motion before switching to those modes mentioned above.

For EMX-100 , this function is used for single axis point-to-point motion using absolute position. Two speed profile parameters distance and maximum velocity are given by user, and other parameters like start velocity, acceleration rate, deceleration rate and s-factor are configurable by axis parameter table. The actual command velocity may not reach maximum velocity due to small traveling distance or accelerating rate are given.
This function uses 'fire-and-forget' mode to avoid blocking user's program or procedure during axis traveling. Users can read motion status MDN to check the motion is completed (MDN = 1) or not (MDN = 0). Except stop command, users CAN NOT start any new move command before previous motion is completed.

Syntax:

C/C++:
I32 FNTYPE APS_absolute_move(I32 Axis_ID, I32 Position, I32 Max_Speed);
Visual Basic:

APS_absolute_move (ByVal Axis_ID As Long, ByVal Position As Long, ByVal Max_Speed As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Position: Absolute command position. Unit is pulse.

I32 Max_Speed: The maximum speed of this move profile. Unit: pulse/sec

For EMX-100: I32 Max_Speed: The maximum speed of this move profile, and its range is 1 ~ 8,000,000 (Unit: pulse/sec)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
APS_set_axis_param(Axis_ID, PRA_ACC, 1000000 ); //Set acceleration rate
APS_set_axis_param(Axis_ID, PRA_DEC, 1000000 ); //Set deceleration rate
//Execute an absolute move
APS_absolute_move( Axis_ID, 10000, 10000 );
```

See also:

APS_relative_move();APS_absolute_move();APS_home_move();APS_stop_move();
APS_emg_stop()

APS_velocity_move	Begin a velocity move

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8144, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/ PCI-C154(+), EMX-100, PCI-8254/58 / AMP-204/8C, AMP-104C

Descriptions:

This function is used to start a velocity move. The axis will stop when users issue stop move command. The speed profile's acceleration and deceleration rate and curve are set by axis parameter function.

This function is 'fire-and-forget' type. That means user's program or procedure will not be pended during axis traveling. Users must use motion status checking function or interrupt event waiting function to wait it done after axis is stopped by command or abnormal situation.

For PCI-8253/56, PCI-8392(H), PCI-8254/58 / AMP-204/8C, users can start a new move command including stop command to override the previous one during the axis traveling. The axis will be switched to new command immediately according to new setting of target position, new speed.

This command can't be overridden by other motion modes like Jog, home, manual pulse generation, contour motion. Users must stop axis motion before switching to those modes mentioned above.

The velocity move is one kind of positioning control. The controller will try to make feedback position to catch up command position. That means if the axis is stopped, the controller will control axis's position to command because it is in position closed loop mode.

For EMX-100 , this function is used for single axis velocity move. The speed profile parameter maximum velocity is given by user, and other parameters like start velocity, acceleration rate, deceleration rate and s-factor are configurable by axis parameter table. The axis will reach the maximum velocity first and move continuously (MDN = 0) until the stop move command is issued (MDN = 1).

Syntax:

C/C++:

I32 FNTYPE APS_velocity_move(I32 Axis_ID, I32 Max_Speed);

Visual Basic:

APS_velocity_move (ByVal Axis_ID As Long, ByVal Max_Speed As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Max_Speed: The maximum speed of this move profile. Unit: pulse/sec

For EMX-100: I32 Max_Speed: The maximum speed of this move profile, and its range is 1 ~ 8,000,000 (Unit: pulse/sec)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
APS_set_axis_param(Axis_ID, PRA_ACC, 1000000 ); //Set acceleration rate
APS_set_axis_param(Axis_ID, PRA_DEC, 1000000 ); //Set deceleration rate
APS_velocity_move(Axis_ID, Max_Speed ); //Start velocity move
…
APS_stop_move(Axis_ID); //Stop velocity move
```

See also:

APS_relative_move(); APS_absolute_move(); APS_velocity_move(); APS_home_move(); APS_stop_move(); APS_emg_stop()

APS_home_move	Begin a home move

Support Products: PCI-8253/56, PCI-8392(H), PCI-8144, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/PCI-C154(+), EMX-100, PCI-8254/58 / AMP-204/8C, PCIe-833x, AMP-104C, ECAT-4XMO

Descriptions:

This function is used to start a HOME (ORG or DOG) position of the axis. There are several modes which can be selected by axis parameter setting functions. After it is done, the position of the axis will be renew base on the physical location of HOME.

This function is 'fire-and-forget' type. That means user's program or procedure will not be pended during axis traveling. Users must use motion status checking function or interrupt event waiting function to wait it done.

Users needn't to write a home sequence to accomplish homing. All the sequences are controlled inside the board without CPU resource.

Note:

1. Home parameters are depended on the type of procucts; please refer to "axis parameter table" below.
2. Some products haven't "Home ACC", "Home VS" and "Home Curve" parameters; they are decided by "PRA_ACC", "PRA_VS" and "PRA_CURVE" respectively. Please refer to "axis parameter table" below.

Syntax:

C/C++:

I32 FNTYPE APS_home_move(I32 Axis_ID);

Visual Basic:

APS_home_move (ByVal Axis_ID As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example1:

Below example is for PCI-8253/6

//Set homing parameters
APS_set_axis_param(Axis_ID, PRA_HOME_MODE, 0); //Set home mode
APS_set_axis_param(Axis_ID, PRA_HOME_DIR, 1); //Set home direction
APS_set_axis_param(Axis_ID, PRA_HOME_CURVE, 0); //Set acceleration paten (T-curve)
APS_set_axis_param(Axis_ID, PRA_HOME_ACC, 1000000); //Set homing acceleration rate
APS_set_axis_param(Axis_ID, PRA_HOME_VS, 0); //Set homing start velocity
APS_set_axis_param(Axis_ID, PRA_HOME_VM, 2000000); //Set homing maximum velocity.
APS_set_axis_param(Axis_ID, PRA_HOME_VO, 200000); //Set homing

APS_home_move(Axis_ID); //Start homing
…//Check homing done(Motion done)

Example2:
Below example is for MNET-4XMO, MNET-4XMO-C and PCI(e)-8154/8
//Set homing parameters
APS_set_axis_param(Axis_ID, PRA_HOME_MODE, 0); //Set home mode
APS_set_axis_param(Axis_ID, PRA_HOME_DIR, 1); //Set home direction
APS_set_axis_param(Axis_ID, PRA_CURVE, 0);// Set acceleration paten (T-curve)
APS_set_axis_param(Axis_ID, PRA_ACC, 1000000); //Set homing acceleration rate
APS_set_axis_param(Axis_ID, PRA_VS , 0);//Set homing start velocity. *1
APS_set_axis_param(Axis_ID, PRA_HOME_VM, 2000000); //Set homing maximum velocity.
APS_set_axis_param(Axis_ID, PRA_HOME_VO, 200000); //Set homing FA velocity. *1

APS_home_move(Axis_ID); //Start homing
…//Check homing done(Motion done)

Example3:
Below example is for PCI-8254/58 / AMP-204/8C and PCIe-833x
//Set homing parameters
APS_set_axis_param(Axis_ID, PRA_HOME_MODE, 0); //Set home mode
APS_set_axis_param(Axis_ID, PRA_HOME_DIR, 1); //Set home direction
APS_set_axis_param_f(Axis_ID, PRA_HOME_CURVE, 0.5); //Set s-factor to 0.5
APS_set_axis_param_f(Axis_ID, PRA_HOME_ACC, 100000.0); //Set homing acceleration rate
APS_set_axis_param_f(Axis_ID, PRA_HOME_VM, 2000000.0); //Set homing maximum velocity.
APS_set_axis_param_f(Axis_ID, PRA_HOME_VO, 200000.0); //Set homing leave home velocity

APS_home_move(Axis_ID); //Start homing
…//Check homing done(Motion done)

Example4:

Below example is for PCI-8144 / AMP-104C
//Set homing parameters
APS_set_axis_param(Axis_ID, PRA_HOME_MODE, 0); //Set home mode
APS_set_axis_param(Axis_ID, PRA_HOME_DIR, 1); //Set home direction
APS_set_axis_param(Axis_ID, PRA_SD_EN , 1); // Enable slow down when SD input is turned ON.
APS_set_axis_param(Axis_ID, PRA_ACC, 1000); //Set homing acceleration rate
APS_set_axis_param(Axis_ID, PRA_HOME_VM, 1000.0); //Set homing maximum velocity.
APS_set_axis_param(Axis_ID, PRA_HOME_DOWN_COUNTER, 10.0); //Set Homing down counter.

APS_home_move(Axis_ID); //Start homing
…//Check homing done(Motion done)

See also:

APS_set_axis_param(); APS_get_axis_param(); APS_stop_move(); APS_emg_stop()

*1: This value must be smaller than PRA_HOME_VM

PCIe-833x home mode scheme:

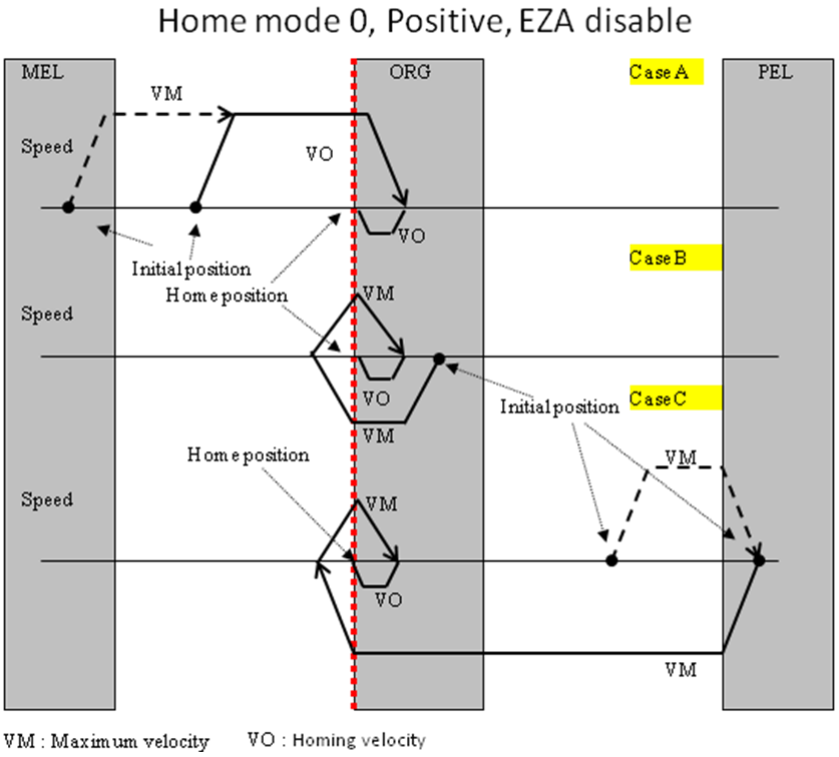


Figure 2 Home mode 0(ORG), positive direction, EZA disable

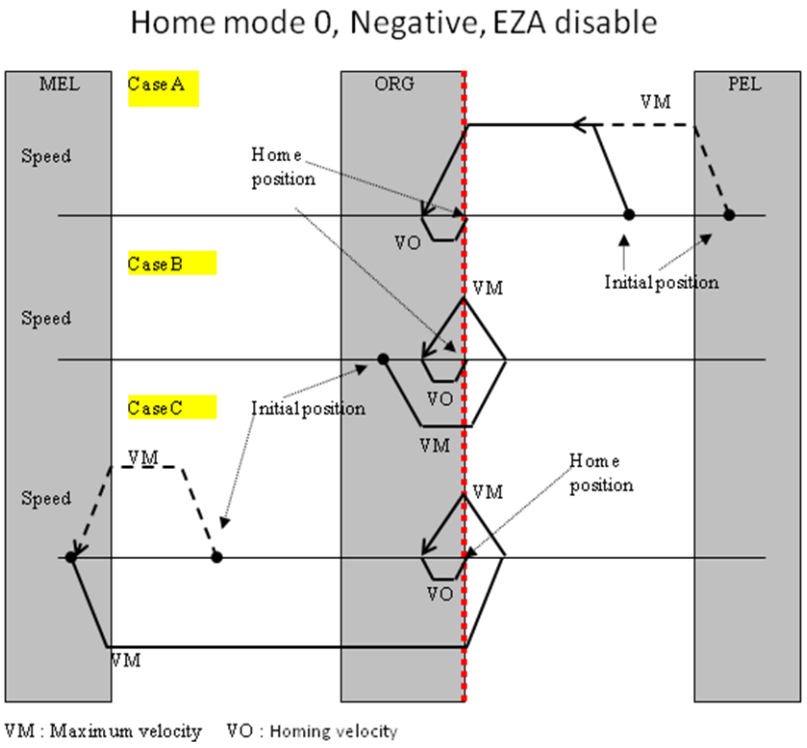


Figure 3 Home mode 0(ORG), negative direction, EZA disable

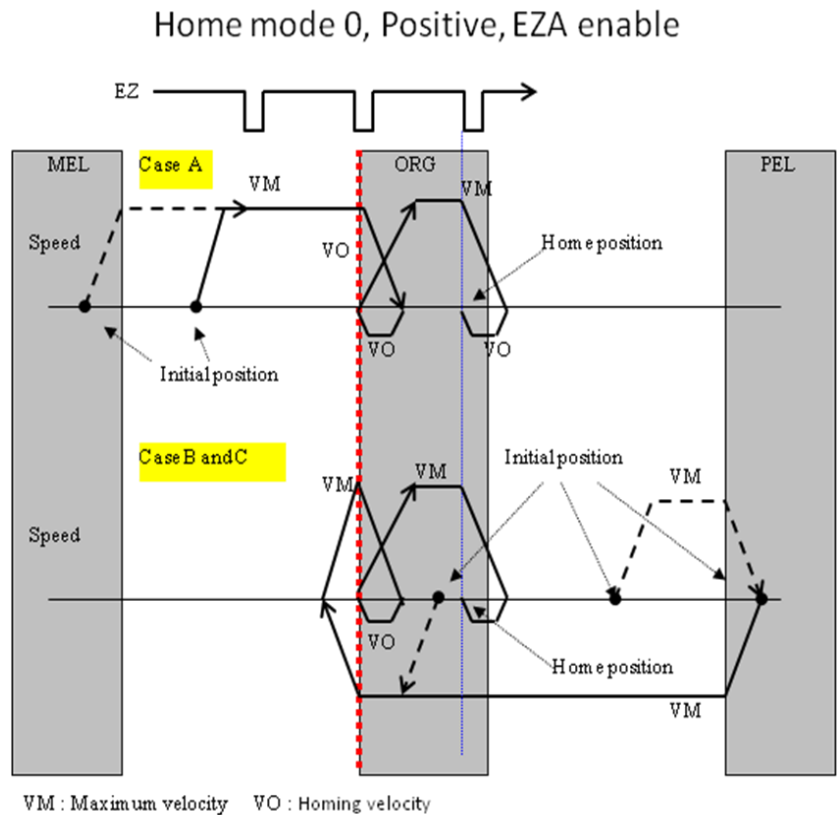


Figure 4 Home mode 0(ORG), positive direction, EZA enable, EZ_DIR disable

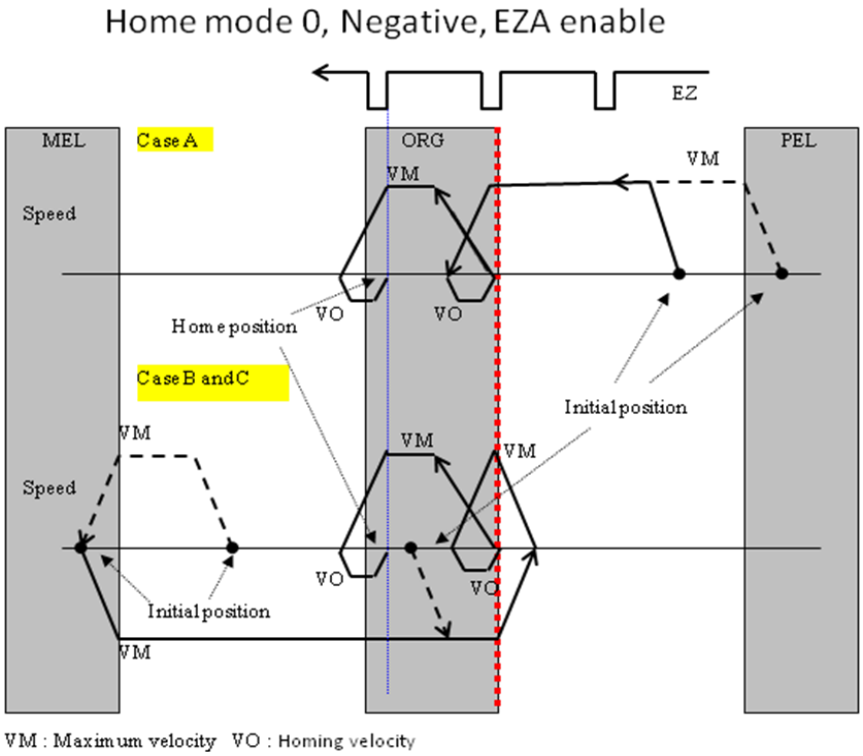


Figure 5 Home mode 0(ORG), negative direction, EZA enable, EZ_DIR disable

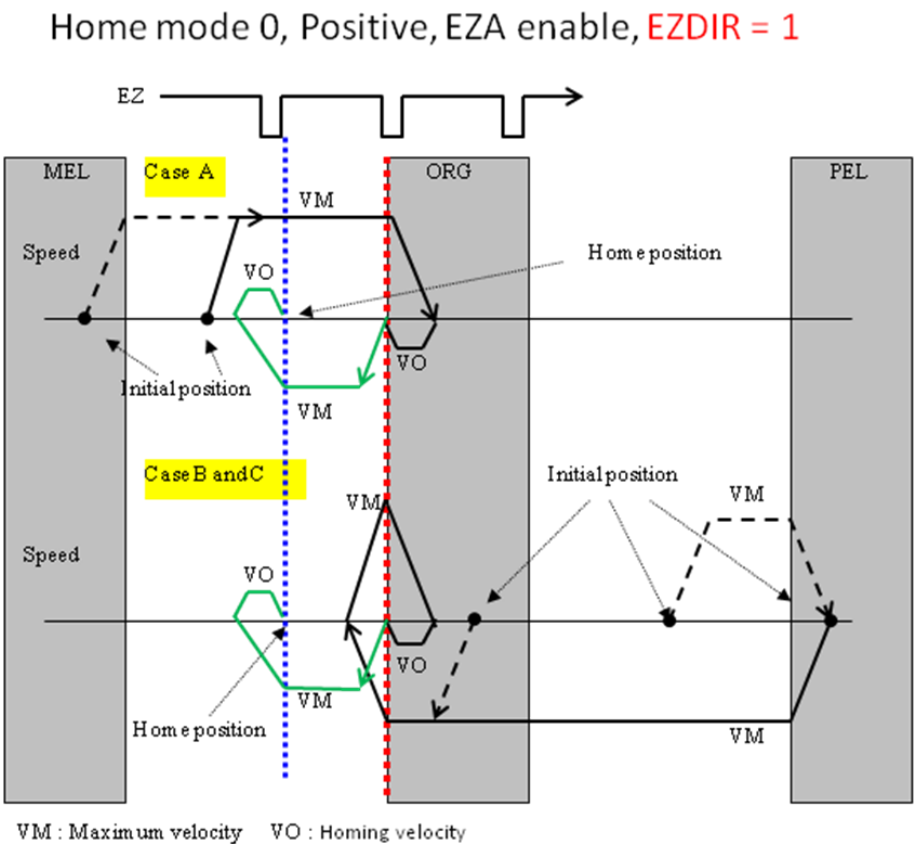


Figure 6 Home mode 0(ORG), positive direction, EZA enable, EZ_DIR enable

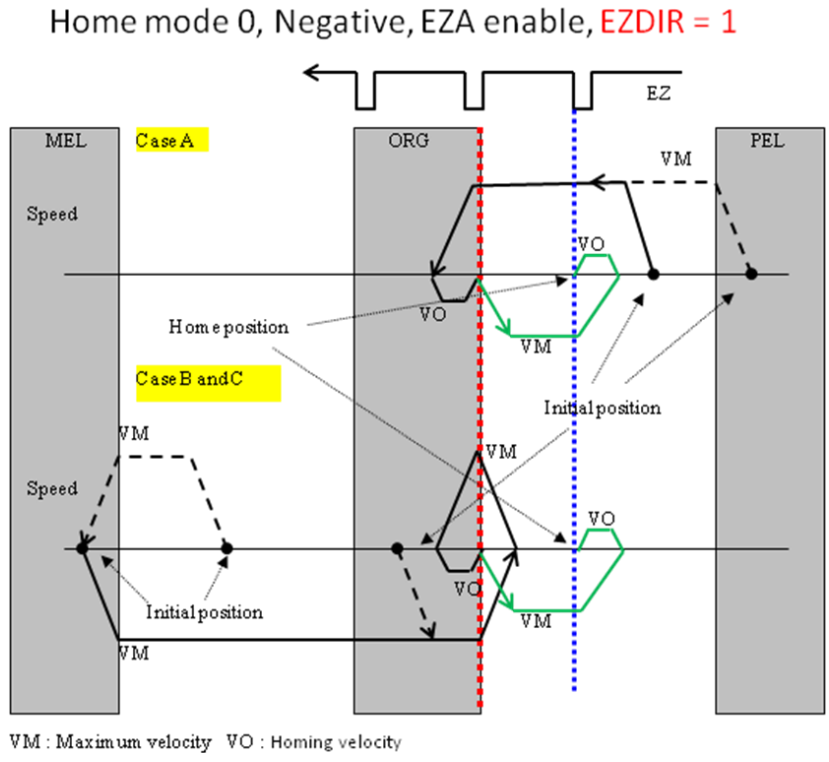


Figure 7 Home mode 0(ORG), negative direction, EZA enable, EZ_DIR enable

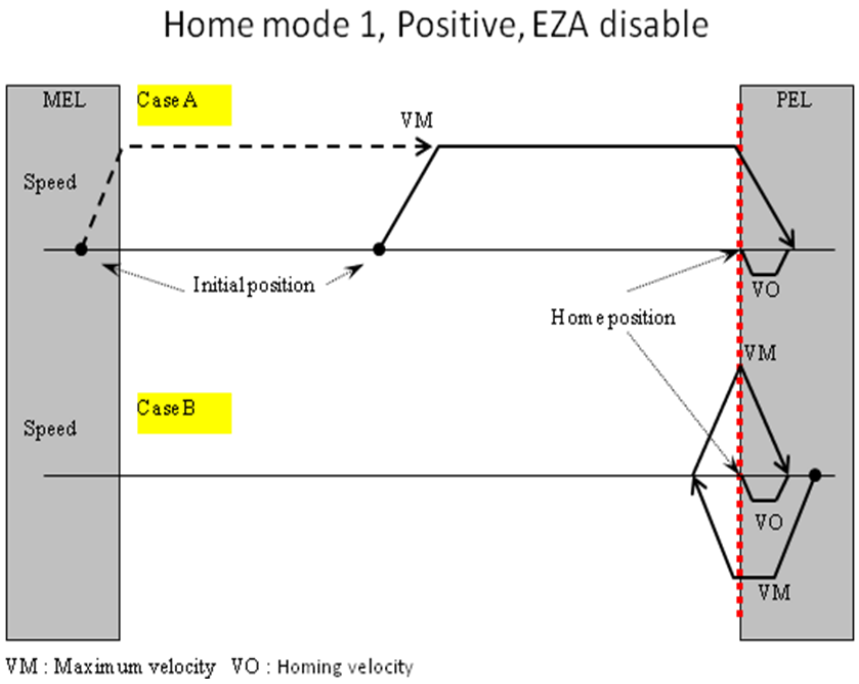


Figure 8 Home mode 1(EL), positive direction, EZA disable

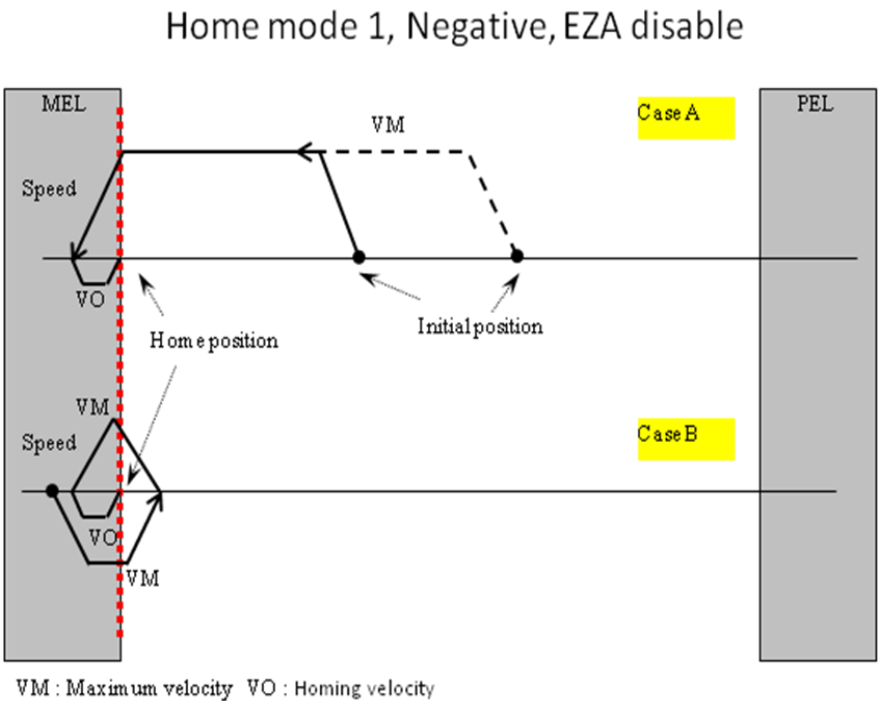


Figure 9 Home mode 1(EL), negative direction, EZA disable

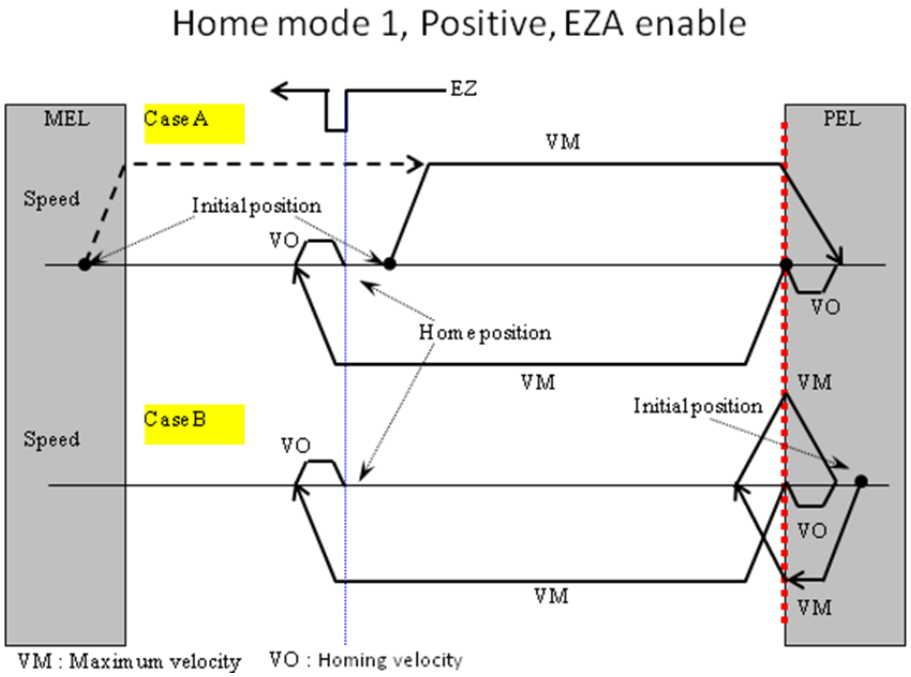


Figure 10 Home mode 1(EL), positive direction, EZA enable

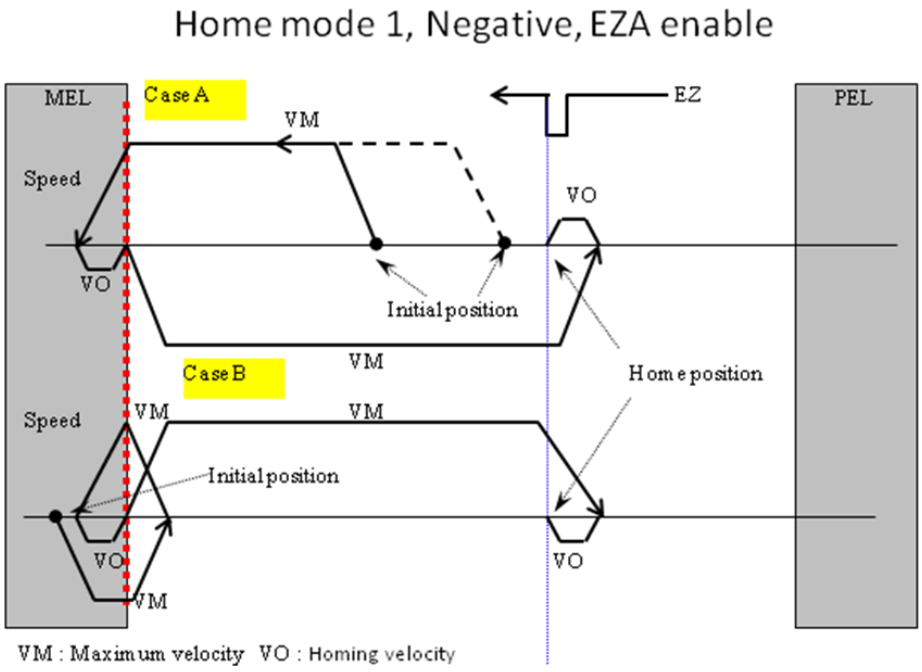


Figure 11 Home mode 1(EL), negative direction, EZA enable

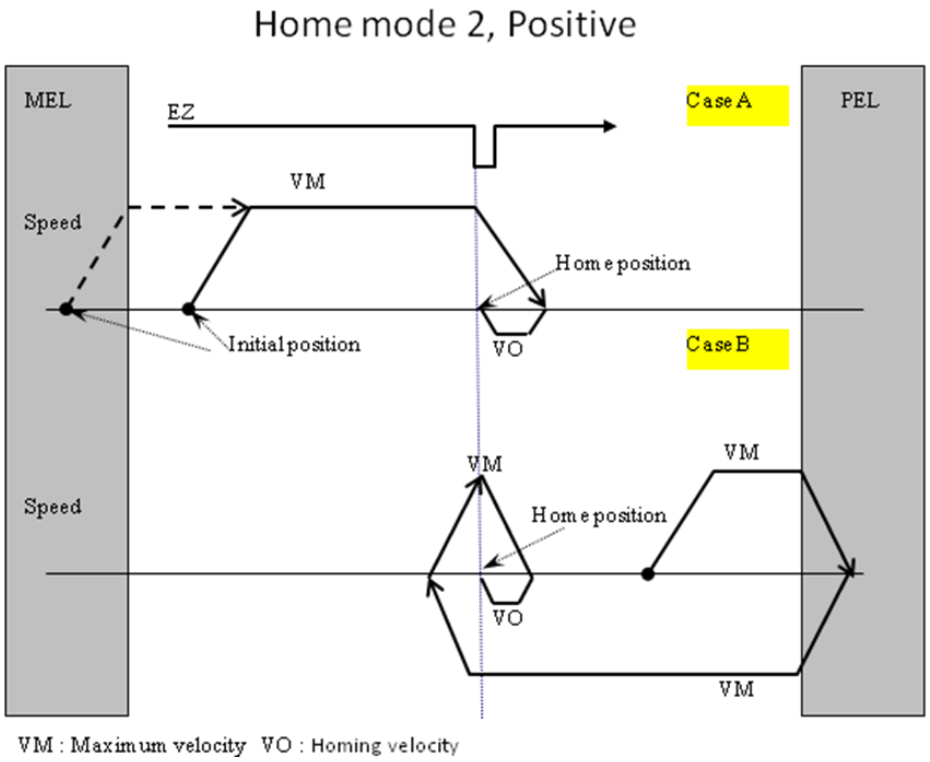


Figure 12 Home mode 2(EZ), position direction

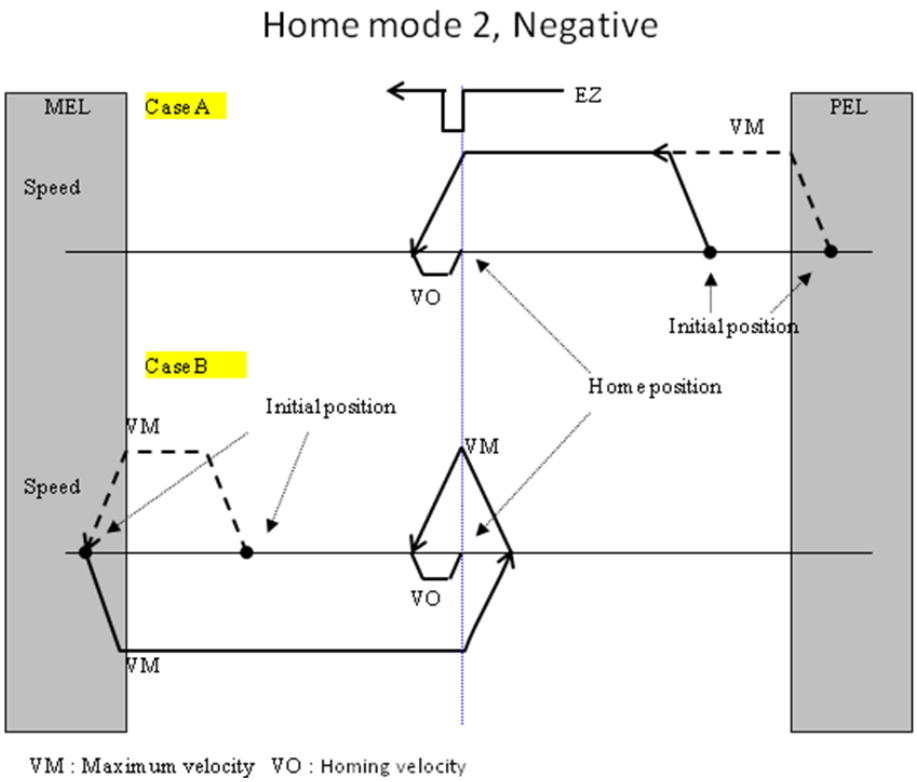


Figure 13 Home mode 2(EZ), negative direction

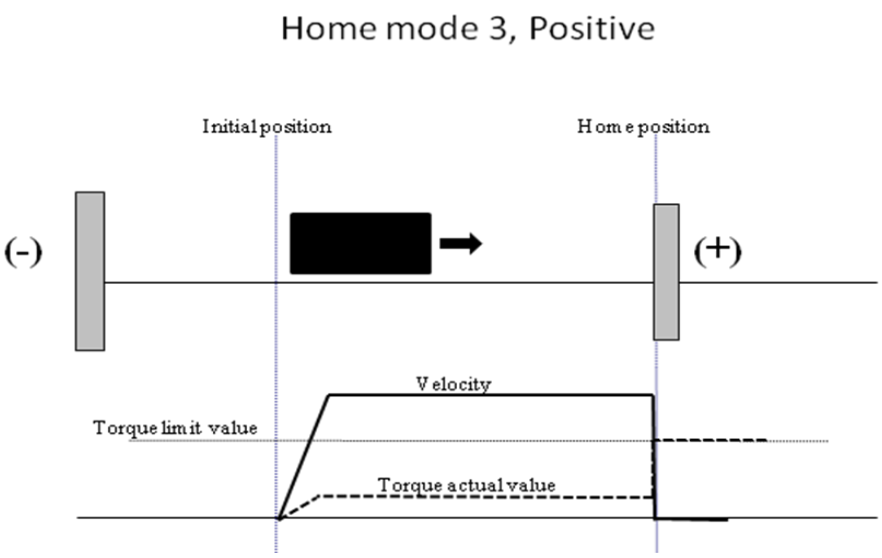


Figure 14 Home mode 3(torque), positive direction

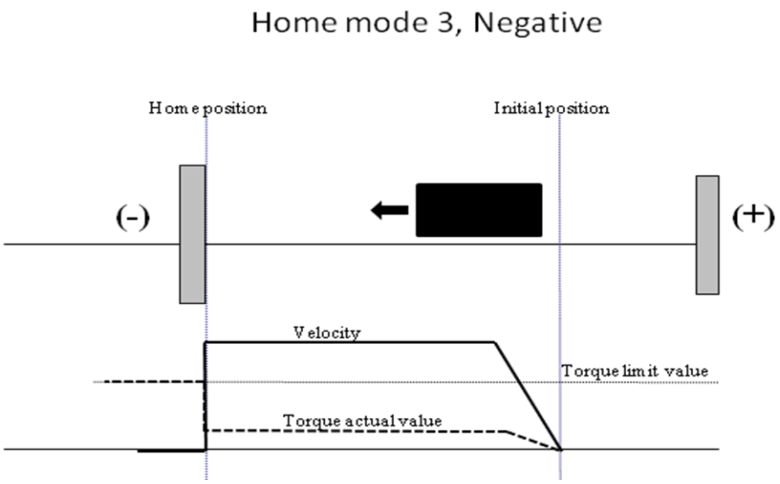
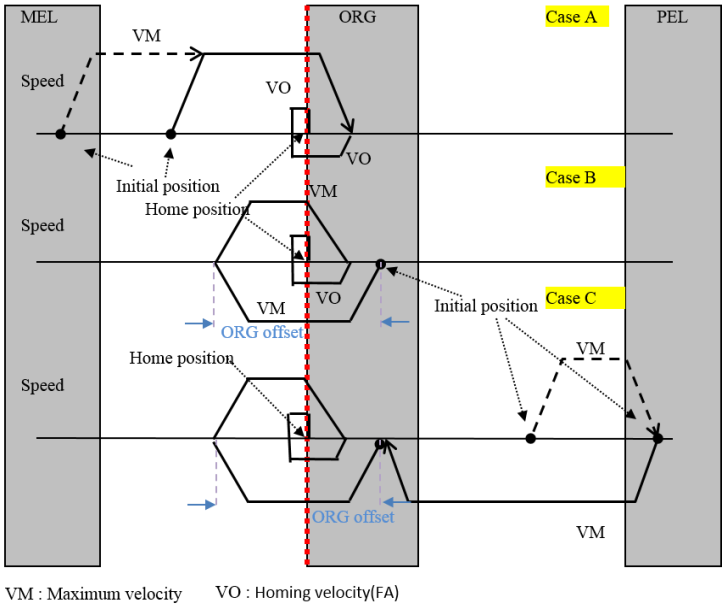


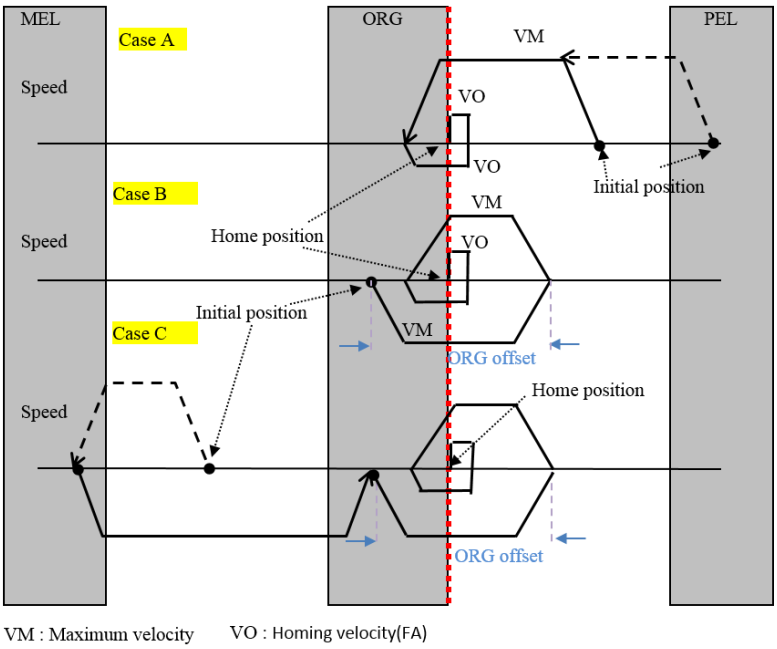
Figure 15 Home mode 3(torque), negative direction

Home mode 4, Positive
ORG-> immediately stop

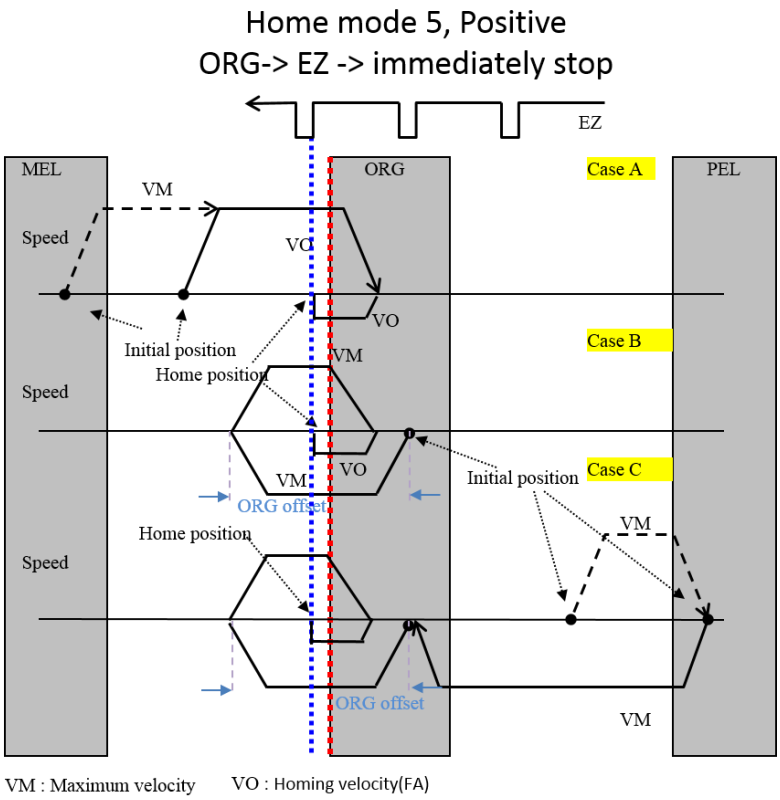


Home mode 4(ORG, immediately stop), positive direction

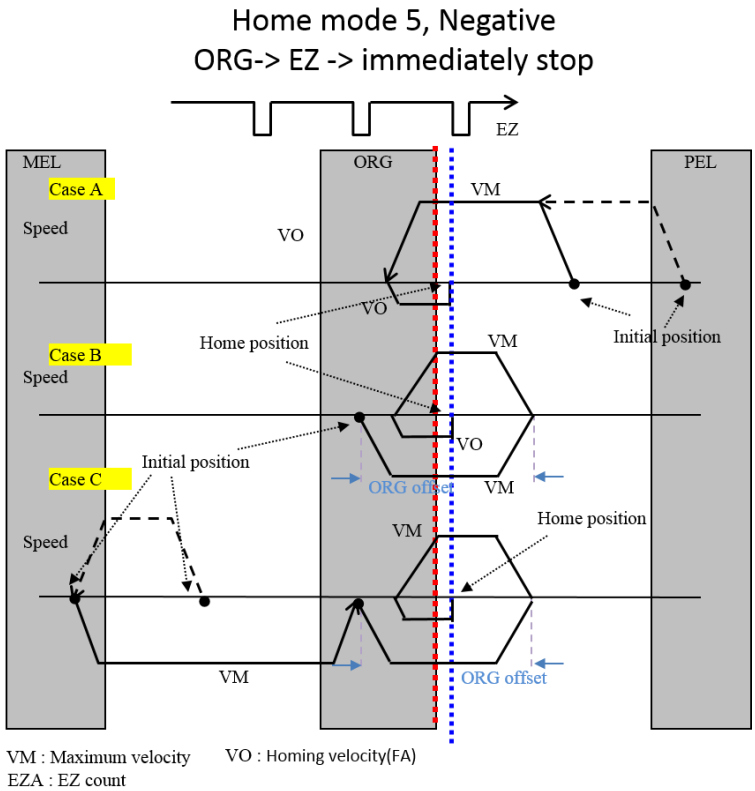
Home mode 4, Negative
ORG-> immediately stop



Home mode 4(ORG, immediately stop), negative direction

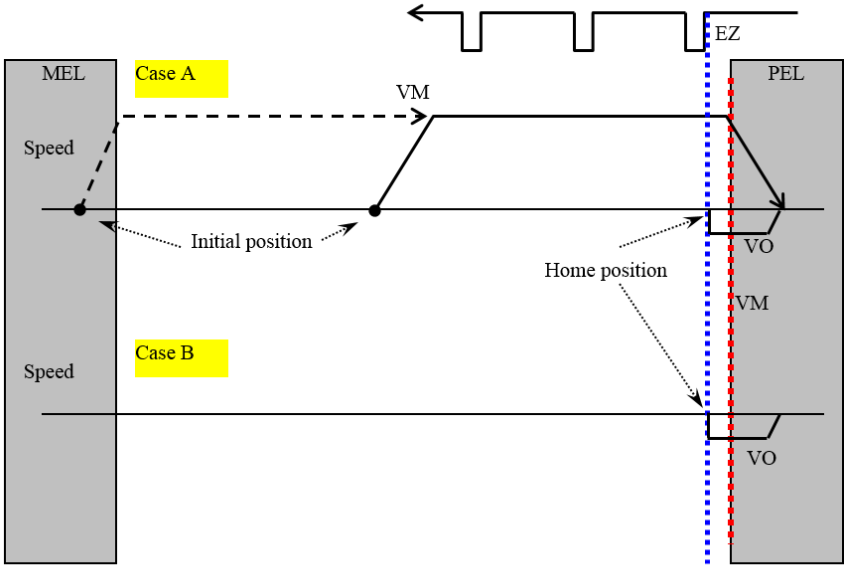


Home mode 5(ORG+EZ, immediately stop), positive direction



Home mode 5(ORG+EZ, immediately stop), negative direction

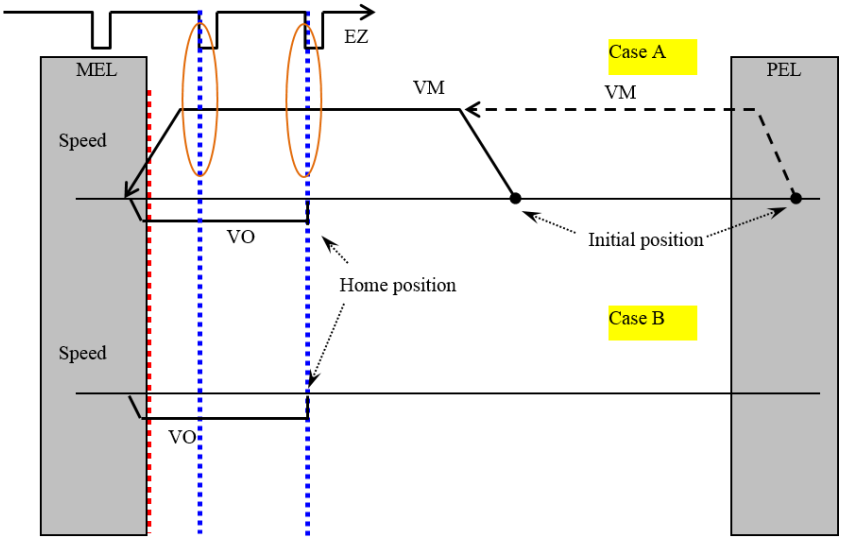
Home mode 6, Positive
EL-> Reverse -> EZ-> immediately stop



VM : Maximum velocity VO : Homing velocity

Home mode 6(EL+EZ, immediately stop), positive direction

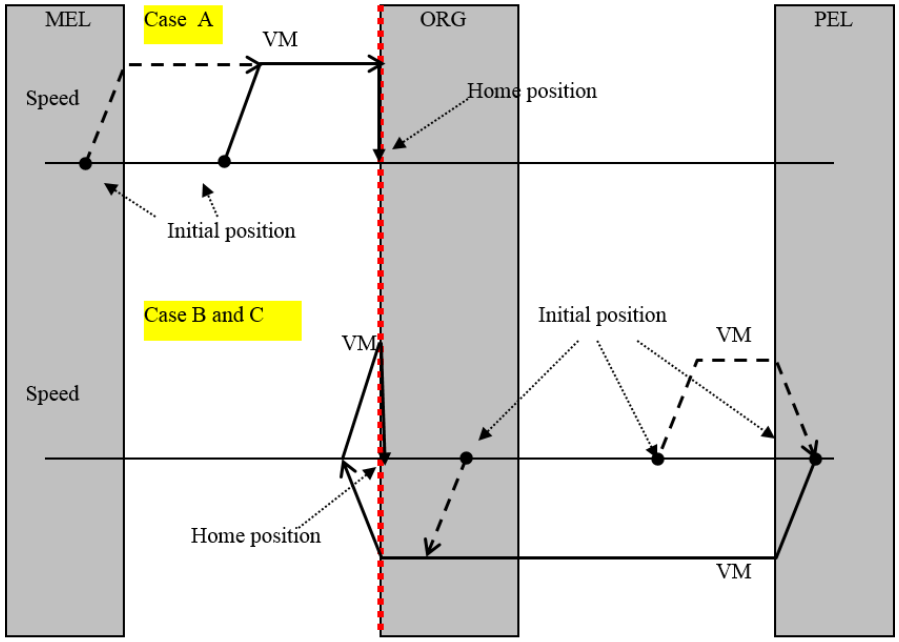
Home mode 6, Negative
EL-> Reverse -> EZ-> immediately stop



VM : Maximum velocity VO : Homing velocity
EZA : EZ count set 1

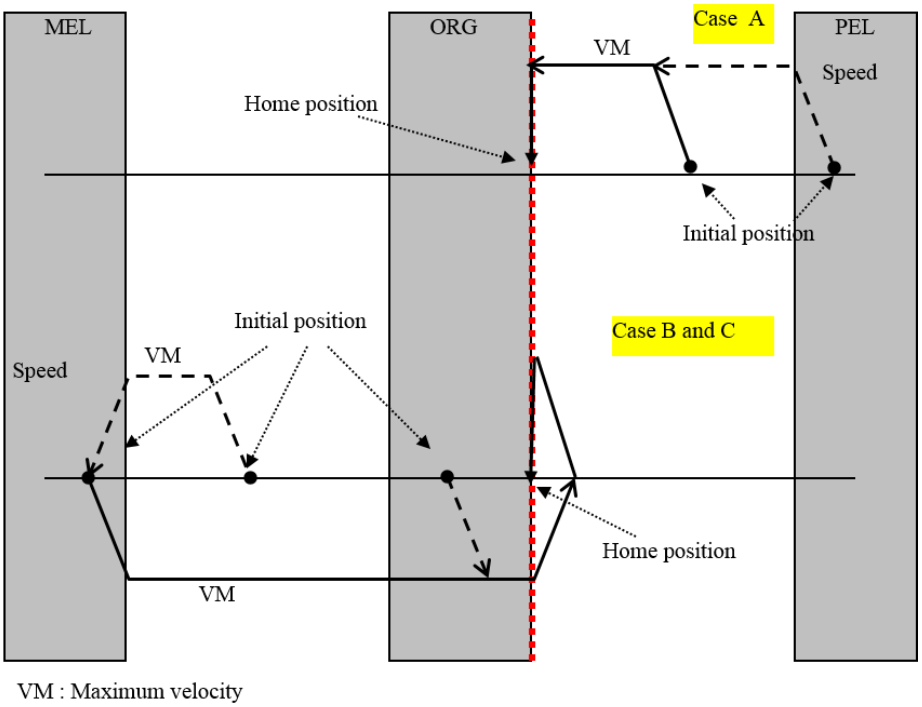
Home mode 6(EL+EZ, immediately stop), negative direction

Home mode 7, positive

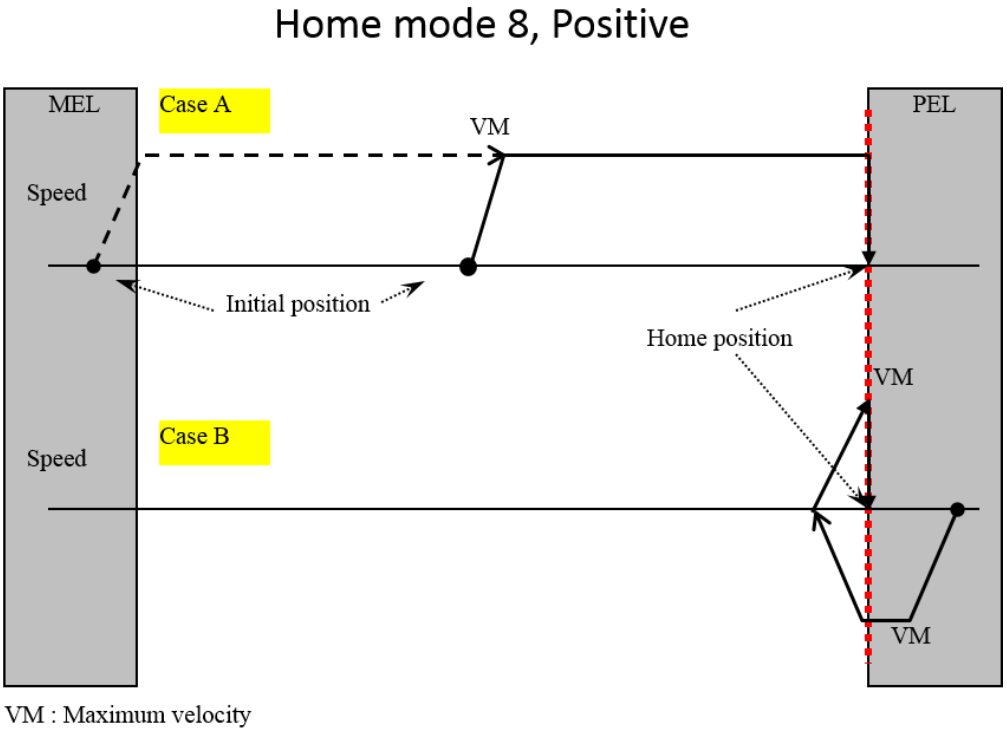


Home mode 7(ORG, immediately stop), positive direction

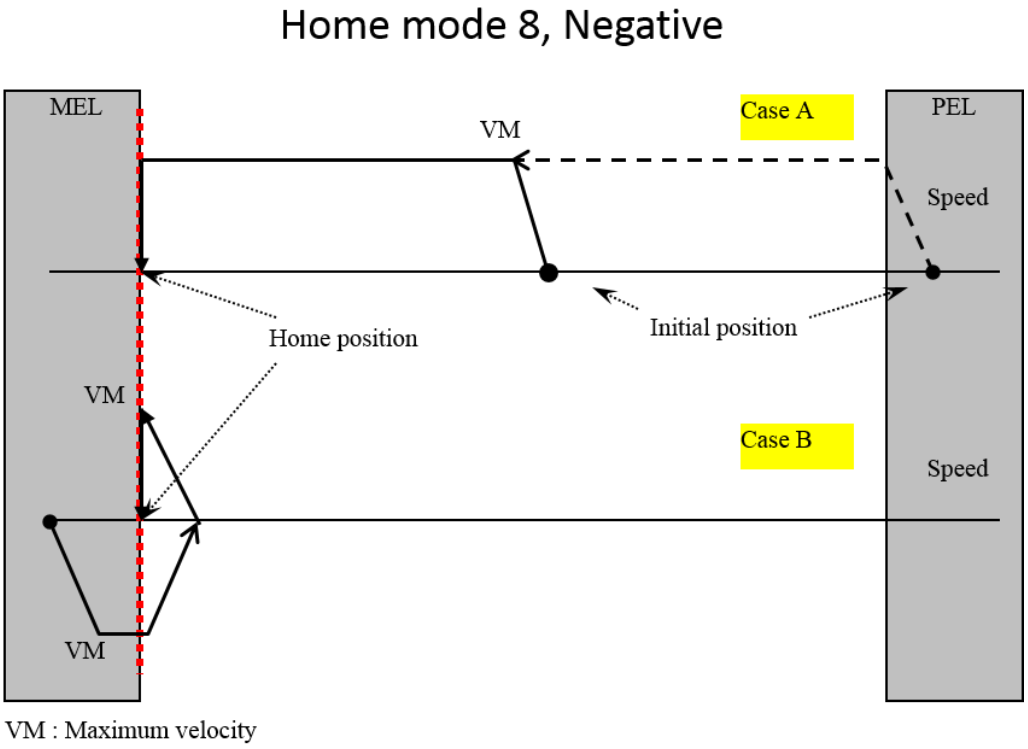
Home mode 7, negative



Home mode 7(ORG, immediately stop), negative direction



Home mode 8(EL, immediately stop), positive direction



Home mode 8(EL, immediately stop), negative direction

APS_stop_move	Stop move

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8144, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/PCI-C154(+), EMX-100, PCI-8254/58 / AMP-204/8C , PCIe-833x, AMP-104C, ECAT-4XMO

Descriptions:

This function is used to stop single or multiple axes motion at once. It can stop single axis homing, positioning and speed moving. It also can stop multiple axes interpolation motion when users place one of axis ID which is relative to interpolation moving. The deceleration profile is set by axis parameter function which is different from normal deceleration setting. The deceleration parameter is different from normal move profile. It can be set individually.
The stop function can't be overridden by other functions.

Syntax:

C/C++:
I32 FNTYPE APS_stop_move(I32 Axis_ID);
Visual Basic:
APS_stop_move (ByVal Axis_ID As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
// APS_absolute_move(Axis_ID, Position, Max_Speed );
// APS_home_move(Axis_ID ); //Home move
…
APS_stop_move(Axis_ID); //Stop move
```

See also:

APS_emg_stop()

APS_emg_stop	Emergency stop

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8144, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, PCI(e)-8154/8158, PCI-8102/PCI-C154(+), EMX-100, PCI-8254/58 / AMP-204/8C , PCIe-833x, AMP-104C, ECAT-4XMO

Descriptions:

This function is used to stop single or multiple axes motion immediately. It can stop single axis homing, positioning and speed moving. It also can stop multiple axes interpolation motion when users place one of axis ID which is relative to interpolation moving. Because the stop function will stop axis accidentally, it will generate an abnormal stop interrupt event rather than normal stop event if interrupt factor is set. The motion status will also be set to an abnormal stop status. The abnormal stop status or event will be clear by next motion command. This function has no deceleration profile.

Syntax:

C/C++:

I32 FNTYPE APS_emg_stop(I32 Axis_ID);

Visual Basic:

APS_emg_stop (ByVal Axis_ID As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

// APS_absolute_move(Axis_ID, Position, Max_Speed);

// APS_home_move(Axis_ID); //Home move

…

APS_emg_stop (Axis_ID); //EMG stop

See also:

APS_stop_move()

APS_relative_move2	Begin a relative distance move with speed profile

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to start a relative distance move. The ability of this function is similar with “APS_relative_move()” function. The different between these two functions is that this function is issued with speed profile within one system cycle. The system cycle means on handshage time with controller from Host PC.

Syntax:

C/C++:

I32 FNTYPE APS_relative_move2(I32 Axis_ID, I32 Distance, I32 Start_Speed, I32 Max_Speed, I32 End_Speed, I32 Acc_Rate, I32 Dec_Rate);

Visual Basic:

APS_relative_move2(ByVal Axis_ID As Long, ByVal Distance As Long, ByVal Start_Speed As Long, ByVal Max_Speed As Long, ByVal End_Speed As Long, ByVal Acc_Rate As Long, ByVal Dec_Rate As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Distance: Relative distance. Unit is pulse.

I32 Start_Speed: The starting speed of this move profile. Unit: pulse/sec

I32 Max_Speed: The maximum speed of this move profile. Unit: pulse/sec.

I32 End_Speed: The end speed of this move profile. Unit: pulse/sec

I32 Acc_Rate: Acceleration rate. Pulse/(sec^2)

I32 Dec_Rate: Deceleration rate. Pulse/(sec^2)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_relative_move()

APS_absolute_move2	Begin a absolute position move with speed profile

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to start an absolute position move. The ability of this function is similar with “APS_absolute_move()” function. The different between these two functions is that this function is called with speed profile parameters and this function only take one system cycle to pass parameters. The system cycle means on handshage time with controller from Host PC.

Syntax:

C/C++:

I32 FNTYPE APS_absolute_move2(I32 Axis_ID, I32 Position, I32 Start_Speed, I32 Max_Speed, I32 End_Speed, I32 Acc_Rate, I32 Dec_Rate);

Visual Basic:

APS_absolute_move2(ByVal Axis_ID As Long, ByVal Position As Long, ByVal Start_Speed As Long, ByVal Max_Speed As Long, ByVal End_Speed As Long, ByVal Acc_Rate As Long, I32 Dec_Rate As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Position: The absolute position. Unit: pulse

I32 Start_Speed: The starting speed of this move profile. Unit: pulse/sec

I32 Max_Speed: The maximum speed of this move profile. Unit: pulse/sec.

I32 End_Speed: The end speed of this move profile. Unit: pulse/sec

I32 Acc_Rate: Acceleration rate. Unit: pulse/sec^2

I32 Dec_Rate: Deceleration rate. Unit: pulse/sec^2

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_absolute_move()

APS_home_move2	Begin a home move with speed profile

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to start a home move operation. The ability of this function is similar with "APS_home_move()" function. The different between these two functions is that this function is called with speed profile parameters and this function only take one system cycle to pass parameters.

The system cycle means on handshage time with controller from Host PC.

Syntax:

C/C++:

I32 FNTYPE APS_home_move2(I32 Axis_ID, I32 Dir, I32 Acc, I32 Start_Speed, I32 Max_Speed, I32 ORG_Speed);

Visual Basic:

APS_home_move2(ByVal Axis_ID As Long, ByVal Dir As Long, ByVal Acc As Long, ByVal Start_Speed As Long, ByVal Max_Speed As Long, ByVal ORG_Speed As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Dir: Homing direction.

0: positive direction (default)

1: negative direction

I32 Acc: Home move acceleration/Deceleration rate. Unit: pulse/sec^2

I32 Start_Speed: Homing start velocity. Unit pulse/sec

I32 Max_Speed: Homing maximum velocity. Unit: pulse/sec.

I32 ORG_Speed: Homing leave home velocity. Unit: pulse/sec.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_home_move()

APS_speed_override	Change speed on the fly

Support Products : MNET-1XMO, MNET-4XMO, MNET-4XMO-C, PCI(e)-8154/58, PCI-C154(+)

Descriptions :

During the axis traveling, users can change a new move speed to override the previous motion. The axis will be switched to new speed immediately.

Note: If original distant is not enough to override to new speed, it will return ERR_DistantNotEnough.

Note: If new speed is the same as current moving speed, it will return ERR_ParametersInvalid.

Syntax:

C/C++:

I32 FNTYPE APS_speed_override(I32 Axis_ID, I32 Max_Speed);

Visual Basic:

APS_speed_override (ByVal Axis_ID As Long, ByVal Max_Speed As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Max_Speed: The maximum speed to override previous motion.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Distance;
I32 Max_Speed;
I32 New_Speed;
I32 ret;

APS_relative_move(Axis_ID, Distance, Max_Speed ); //Start relative move
//Speed override
ret = APS_speed_override(Axis_ID, New_Speed ); //Change to new speed
…
```

See also:

APS_relative_move_ovrd	Begin a relative distance move. Or override it with new distance and speed.

Support Products : MNET-1XMO, MNET-4XMO, MNET-4XMO-C, PCI(e)-8154/58, PCI-C154(+)

Descriptions :

Begin a relative distance:

This function is used to start a single axis relative motion. Although there is maximum speed setting in function parameter, the traveling distance and accelerating rate may not be enough due to user's setting to reach the maximum speed. The speed profile's acceleration and deceleration rate and curve are set by axis parameter function.

Override during the axis traveling:

During the axis traveling, users can start a new move command to override the previous one. The axis will be switched to new command immediately according to new setting of new distance, new speed.

Notice that if new distance is not enough to override to new speed, it will return ERR_DistantNotEnough.

Notice that, new distance was reference to command counter when overriding regardless of the setting of the axis parameter PRA_FEEDBACK_SRC.

Syntax:

C/C++:

I32 FNTYPE APS_ relative_move_ovrd (I32 Axis_ID, I32 Distance, I32 Max_Speed);

Visual Basic:

APS_ relative_move_ovrd (ByVal Axis_ID As Long, ByVal Distance As Long , ByVal Max_Speed As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Distance: Relative distance. Unit is pulse.

I32 Max_Speed: The maximum speed of this move profile. Unit: pulse/sec.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Distance;
I32 Max_Speed;
I32 New_Distance:
I32 New_Speed;
I32 ret;

// Begin a relative distance
Ret = APS_relative_move_ovrd(Axis_ID, Distance, Max_Speed );
// Override during the axis traveling
ret = APS_relative_move_ovrd(Axis_ID, New_Distance , New_Speed );
…
```

See also:

APS_absolute_move_ovrd	Begin an absolute position move. Or override it with new position and speed.

Support Products : MNET-1XMO, MNET-4XMO, MNET-4XMO-C, PCI(e)-8154/58, PCI-C154(+)

Descriptions :

Begin an absolute position move:

This function is used to start a single axis absolute positioning motion. Although there is maximum speed setting in function parameter, the traveling distance and accelerating rate may not be enough due to user's setting to reach the maximum speed. The speed profile's acceleration and deceleration rate and curve are set by axis parameter function.

Override during the axis traveling:

During the axis traveling, users can start a new move command to override the previous one. The axis will be switched to new command immediately according to new setting of absolute position, new speed.

Notice that if new position is not enough to override to new speed, it will return ERR_DistantNotEnough.

Notice that, new position was reference to command counter when overriding regardless of the setting of the axis parameter PRA_FEEDBACK_SRC.

Syntax:

C/C++:

I32 FNTYPE APS_absolute_move_ovrd (I32 Axis_ID, I32 Position, I32 Max_Speed);

Visual Basic:

APS_absolute_move_ovrd (ByVal Axis_ID As Long, ByVal Position As Long , ByVal Max_Speed As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Position: Absolute position. Unit is pulse.

I32 Max_Speed: The maximum speed of this move profile. Unit: pulse/sec.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Position;
I32 Max_Speed;
I32 New_Position:
I32 New_Speed;
I32 ret;

// Begin an absolute position move
Ret = APS_absolute_move_ovrd (Axis_ID, Position, Max_Speed );
// Override during the axis traveling
ret = APS_absolute_move_ovrd (Axis_ID, New_Position, New_Speed );
…
```

See also:

APS_home_escape	Leave home switch

Support Products:MNET-4XMO-(C), MNET-1XMO, PCI(e)-8154/8158, PCI-8102/PCI-C154(+)

Descriptions:

This function is used to leave HOME (ORG) position.

Note:

1. Home parameters are depended on the type of procucts; please refer to "axis parameter table" below.
2. Some products haven't "Home ACC", "Home VS" and "Home Curve" parameters; they are decided by "PRA_ACC", "PRA_VS" and "PRA_CURVE" respectively. Please refer to "axis parameter table" below.

Syntax:

C/C++:

I32 FNTYPE APS_home_escape(I32 Axis_ID);

Visual Basic:

APS_home_escape (ByVal Axis_ID As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

//Set homing parameters

APS_set_axis_param(Axis_ID, PRA_HOME_DIR, 1); //Set home direction

APS_set_axis_param(Axis_ID, PRA_HOME_CURVE, 0); //Set acceleration paten (T-curve)

APS_set_axis_param(Axis_ID, PRA_HOME_ACC, 10000); //Set homing acceleration rate

APS_set_axis_param(Axis_ID, PRA_HOME_VS, 0); //Set homing start velocity

APS_set_axis_param(Axis_ID, PRA_HOME_VM, 10000); //Set homing maximum velocity.

APS_home_escape(Axis_ID); //Escape home

…//Check homing done(Motion done)

See also:

APS_set_axis_param(); APS_get_axis_param(); APS_stop_move(); APS_emg_stop()

7. Multi-axes move trigger & stop

APS_move_trigger	Send a trigger to sync all waiting moves

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

The function is used to send a trigger to sync all waiting moves. Refer to chapter advanced single move & interpolation. User could set bit 8 of Option parameter by invoking advanced motion functions, so this move will set to waiting state.

Syntax:

C/C++

I32 FNTYPE APS_move_trigger(I32 Dimension, I32 *Axis_ID_Array);

Visual Basic:

APS_move_trigger(ByVal Dimension As Long, Axis_ID_Array As Long) As Long

Parameters:

I32 Dimension: The dimension of simultaneous axes.

I32 *Axis_ID_Array: The axis ID array from 0 to 65535.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Axis_ID_Array[2] = { axis_id0, axis_id1 };
//Bit 8 set to 1. Be a waiting state.
I32 opt = 0x0100; //absolute, wait trigger, Aborting mode
ASYNCALL *wait = NULL;

//An absolute move to position 10000 in wating state
APS_ptp( axis_id0, opt, 10000, wait );
APS_ptp( axis_id1, opt, 10000, wait );

// send a trigger to sync all waiting moves
APS_move_trigger( 2, Axis_ID_Array );
...
```

```
// Stop a simultaneous move
APS_stop_move_multi ( 2, Axis_ID_Array );
```

See also:

APS_stop_move_multi()

APS_stop_move_multi	Multi-axes stop move

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to stop multiple axes motion at the same time. Generally speaking, it is used to stop synchronized move. The deceleration profile, defined to be PRA_SD_DEC, is set by invoking APS_set_axis_param_f(). User could refer to axis parameter table for the details.

Syntax:

C/C++:

I32 FNTYPE APS_stop_move_multi (I32 Dimension, I32 *Axis_ID_Array);

Visual Basic:

APS_stop_move_multi (ByVal Dimension As Long, Axis_ID_Array As Long) As Long

Parameters:

I32 Dimension: The dimension of stopped axes.

I32 *Axis_ID_Array: The axis ID array from 0 to 65535.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Axis_ID_Array[2] = { axis_id0, axis_id1 };
//Bit 8 set to 1. Be a waiting state.
I32 opt = 0x0100; //absolute, wait trigger, Aborting mode
ASYNCALL *wait = NULL;

//An absolute move to position 10000 in wating state
APS_ptp( axis_id0, opt, 10000, wait );
APS_ptp( axis_id1, opt, 10000, wait );

// send a trigger to sync all waiting moves
APS_move_trigger( 2, Axis_ID_Array );
...
// Stop a simultaneous move
APS_stop_move_multi ( 2, Axis_ID_Array );
```

See also:

APS_move_trigger()

APS_emg_stop_multi	Multi-axes emg stop move

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to stop immediately multiple axes motion at the same time. Because the stop function will stop axis accidentally, it will generate an abnormal stop interrupt event rather than normal stop event if interrupt factor is set. The motion status will also be set to an abnormal stop status. The abnormal stop status or event will be clear by next motion command. This function has no deceleration profile.

Syntax:

C/C++:

I32 FNTYPE APS_emg_stop_multi (I32 Dimension, I32 *Axis_ID_Array);

Visual Basic:

APS_emg_stop_multi (ByVal Dimension As Long, Axis_ID_Array As Long) As Long

Parameters:

I32 Dimension: The dimension of stopped axes.

I32 *Axis_ID_Array: The axis ID array from 0 to 65535.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Axis_ID_Array[2] = { axis_id0, axis_id1 };
//Bit 8 set to 1. Be a waiting state.
I32 opt = 0x0100; //absolute, wait trigger, Aborting mode
ASYNCALL *wait = NULL;

//An absolute move to position 10000 in wating state
APS_ptp( axis_id0, opt, 10000, wait );
APS_ptp( axis_id1, opt, 10000, wait );

// send a trigger to sync all waiting moves
APS_move_trigger( 2, Axis_ID_Array );
...
```

```
// Emg stop a simultaneous move
APS_emg _stop_multi ( 2, Axis_ID_Array );
```

See also:

APS_move_trigger()

8. Jog move

APS_set_jog_param	Set Jog parameters

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to set jog move relative parameters. The parameters are also available in axis parameter table.

Syntax:

C/C++:

I32 FNTYPE APS_set_jog_param(I32 Axis_ID, JOG_DATA *pStr_Jog, I32 Mask);

Visual Basic:

APS_set_jog_param(ByVal Axis_ID As Long, pStr_Jog As JOG_DATA, ByVal Mask As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

JOG_DATA *pStr_Jog: Structure of jog move parameters. Define in “type_def.h”

```
typedef struct
{
    I16 i16_jogMode; // Jog mode. 0:Free running mode, 1:Step mode
    I16 i16_dir;              // Jog direction. 0:positive, 1:negative direction
    I16 i16_accType;      // Acceleration and Deceleration pattern 0: T-curve, 1: S-curve
    I32 i32_acc;      // Acceleration rate ( pulse / sec2 )
    I32 i32_dec;      // Deceleration rate ( pulse / sec2 )
    I32 i32_maxSpeed; // A Positive value, maximum velocity. ( pulse / s )
    I32 i32_offset;       // A Positive value, step offset. For step jog mode. (pulse)
    I32 i32_delayTime; // Delay time, For step jog mode. ( range: 0 ~ 65535 millisecond, align by cycle time)
} JOG_DATA;
```

I32 Mask: Mask parameter setting. Bit format, set 0 will be masked.

Mask item	Mask bit number
Acceleration rate (i32_acc)	0
Deceleration rate(i32_dec)	1

Maximum velocity (i32_maxSpeed)	2
Step offset (i32_offset)	3
Delay time (i32_delayTime)	4
Jog mode (i16_jog mode)	5
Jog direction (i16_direction)	6
Jog acceleration/deceleration pattern	7

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS168.h”
// Initial cards first…

I32 ret;
JOG_DATA jog;

jog.i16_jogMode = 1; //Mask = 0x20
jog.i16_dir = 0; //Mask = 0x40

ret = APS_set_jog_param( Axis_ID, &jog, 0x20 | 0x40 );
if( ret != 0 ) //Error
```

See also:

APS_set_axis_param(),APS_get_axis_param(),APS_get_jog_param()

APS_get_jog_param	Get Jog parameters

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to get jog move relative parameters.

Syntax:

C/C++:

I32 FNTYPE APS_get_jog_param(I32 Axis_ID, JOG_DATA *pStr_Jog);

Visual Basic:

APS_get_jog_param(ByVal Axis_ID As Long, pStr_Jog As JOG_DATA) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

JOG_DATA *pStr_Jog: Structure of jog move parameters. Define in “type_def.h”

```
typedef struct
{
    I16 i16_jogMode; // Jog mode. 0:Free running mode, 1:Step mode
    I16 i16_dir;              // Jog direction. 0:positive, 1:negative direction
    I16 i16_accType;      // Acceleration and Deceleration pattern 0: T-curve, 1: S-curve
    I32 i32_acc;      // Acceleration rate ( pulse / sec2 )
    I32 i32_dec;      // Deceleration rate ( pulse / sec2 )
    I32 i32_maxSpeed; // A Positive value, maximum velocity. ( pulse / s )
    I32 i32_offset;       // A Positive value, step offset. For step jog mode. (pulse)
    I32 i32_delayTime; // Delay time, For step jog mode. ( range: 0 ~ 65535 millisecond, align
by cycle time)
} JOG_DATA;
```

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS168.h”
// Initial cards first…
```

```
I32 ret;
JOG_DATA jog;

ret = APS_get_jog_param( Axis_ID, &jog );
if( ret != 0 ) //Error
```

See also:

APS_set_axis_param();APS_get_axis_param();APS_set_jog_param()

APS_jog_mode_switch	Enable / Disable jog move

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to switch specified axis to jog mode. When the axis is in jog mode, it cannot accept other move command except stop command.

Users must enable jog move mode before perform jog move.

Syntax:

C/C++:

I32 FNTYPE APS_jog_mode_switch(I32 Axis_ID, I32 Turn_No);

Visual Basic:

APS_jog_mode_switch(ByVal Axis_ID As Long, ByVal Turn_No As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Turn_No: 0:Disable jog mode, 1:Enable jog mode.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
// Configure jog move parameter.

Ret = APS_jog_mode_switch(Axis_ID, 1 ); //Turn on jog move mode.
// perform jog move …(APS_jog_start)
…
ret = APS_jog_mode_switch(Axis_ID, 0 ); //Turn off jog move mode.
// perform other move commands
```

See also:

APS_set_jog_param(); APS_get_jog_param();APS_jog_start()

APS_jog_start	Start / stop jog move

Support Products: PCI-8253/56, PCI-8392(H), EMX-100 , PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to start / stop a jog move. Before start a jog move, you must enable the axis to jog mode.

For EMX-100 , this function is used to start / stop a jog move. The axis parameters shown below are used to configure speed profile of Jog motion.It only support s-factor = 0 (T-Curve).

Symbol define	ParamNo	Description
PRA_JG_DIR	0x41	(I32) Jog move direction [0: Positive direction, 1: Negative direction]
PRA_JG_ACC	0x43	Jog move acceleration
PRA_JG_VM	0x45	Jog move max velocity
PRA_JG_STOP	0x4C	Jog stop mode

For PCI-8254/58 / AMP-204/8C and PCIe-833x, ECAT-4XMO,following parameters are shown to configure Jog parameter: The details refer to axis parameter table.

Symbol define	ParamNo	Description
PRA_JG_MODE	0x40	(I32) Jog mode [0:Continuous mode, 1:Step mode]
PRA_JG_DIR	0x41	(I32) Jog move direction [0: Positive direction, 1: Negative direction]
PRA_JG_SF	0x42	(F64) Jog S factor [0 ~ 1]
PRA_JG_ACC	0x43	(F64) Jog move acceleration [value > 0]
PRA_JG_DEC	0x44	(F64) Jog move deceleration [value > 0]
PRA_JG_VM	0x45	(F64) Jog move max velocity [value > 0]
PRA_JG_OFFSET	0x46	(F64) Jog offset, for step mode [value >= 0]
PRA_JG_DELAY	0x47	(I32) Jog delay, for step mode, microsecond [0 ~ 10,000,000]
PRA_JG_MAP_DI_EN	0x48	(I32) Enable Digital input map to jog command signal
PRA_JG_P_JOG_DI	0x49	(I32) Mapping configuration for positive jog and digital input.
PRA_JG_N_JOG_DI	0x4A	(I32) Mapping configuration for negative jog and digital input.
PRA_JG_JOG_DI	0x4B	(I32) Mapping configuration for jog and digital input.

Syntax:

C/C++:

I32 FNTYPE APS_jog_start(I32 Axis_ID, I32 STA_On);

Visual Basic:

APS_jog_start(ByVal Axis_ID As Long, ByVal STA_On As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 STA_On: 1:STA signal on, 0:STA signal off.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example1:

Below example is for PCI-8253/56, PCI-8392(H)

```
// Configure jog move parameter.

Ret = APS_jog_mode_switch(Axis_ID, 1 ); //Turn on jog move mode.

// perform jog move …(APS_jog_start)
APS_jog_start( Axis_ID,1 ); //STA signal ON
…
APS_jog_start(Axis_ID, 0); //STA signal OFF

ret = APS_jog_mode_switch(Axis_ID, 0 ); //Turn off jog move mode.
// perform other move commands
```

Example2:

Below example is for EMX-100

```
// Configure jog move parameter.
APS_set_axis_param( Axis_ID, PRA_JG_DIR, 1 ); //Set jog to negative direction
APS_set_axis_param( Axis_ID, PRA_JG_ACC, 100000 ); //Set jog move acceleration

// perform jog move …(APS_jog_start)
APS_jog_start( Axis_ID,1 ); //STA signal ON
…
APS_jog_start(Axis_ID, 0); //STA signal OFF
```

Example3:

Below example is for PCI-8254/58 / AMP-204/8C or PCIe-833x, ECAT-4XMO

```
// Configure jog move parameter.
APS_set_axis_param( Axis_ID, PRA_JG_MODE, 0 ); //Set to continuous mode
APS_set_axis_param( Axis_ID, PRA_JG_DIR, 1 ); //Set jog to negative direction
APS_set_axis_param_f( Axis_ID, PRA_JG_ACC, 100000.0 ); //Set jog move acceleration

// perform jog move …(APS_jog_start)
APS_jog_start( Axis_ID,1 ); //STA signal ON
…
APS_jog_start(Axis_ID, 0); //STA signal OFF
```

See also:

APS_set_jog_param(); APS_get_jog_param(); APS_jog_mode_switch();

9. Interpolation

APS_absolute_linear_move	Begin a absolute position linear interpolation

Support Products: PCI-8253/56,PCI-8392(H) ,
MNET-4XMO-(C),HSL-4XMO,PCI(e)-8154/8158, PCI-8102/PCI-C154(+), EMX-100,
PCI-8254/58 / AMP-204/8C, AMP-104C

Descriptions:

This function is used to start an absolute linear interpolation positioning motion. Although there is maximum speed setting in function parameter, the traveling distance and accelerating rate may not be enough due to user's setting to reach the maximum speed. The speed profile's acceleration and deceleration rate and curve are set by axis parameter function. Because the speed parameter is in vector direction, this function will take the master axis's acceleration and deceleration time constant to calculate. The master axis is the minimum axis number that user perform an interpolation.

This function is 'fire-and-forget' type. That means user's program or procedure will not be pended during axis traveling. Users must use motion status checking function or interrupt event waiting function to wait it done.

During the axis traveling, users can start a new move command including stop command to override the previous one. The axis will be switched to new command immediately according to new setting of target position, new speed.

The overridden command must have the same dimension and axis ID of previous one. These two commands can't be overridden by other motion modes like home operation. Users must stop axis motion before switching to those modes mentioned above.

Note: The axes specified in Axis_ID_Array must be of the same card.

For EMX-100 , this function is used for linear interpolation positioning motion using absolute postion. It only supports two axes motion at the same time with the same device. The master axis is defined as the first axis ID that user gives for interpolation. Two speed profile parameters travel distance and maximum velocity are given by user, and other parameters like start velocity, acceleration rate, deceleration rate and s-factor are configurable by master axis's parameter table. The actual command velocity may not reach maximum velocity due to small traveling distance or accelerating rate are given.

This function uses 'fire-and-forget' mode to avoid blocking user's program or procedure during axis traveling. Users can read motion status MDN to check the motion is completed (MDN = 1)

or not (MDN = 0). Except stop command, users CAN NOT start any new move command before previous motion is completed.

Note: The axes specified in Axis_ID_Array must be of the same card.

Syntax:

C/C++:

I32 FNTYPE APS_absolute_linear_move(I32 Dimension, I32 *Axis_ID_Array, I32 *Position_Array, I32 Max_Linear_Speed);

Visual Basic:

APS_absolute_linear_move(ByVal Dimension As Long, Axis_ID_Array As Long, Position_Array As Long, ByVal Max_Linear_Speed As Long) As Long

Parameters:

I32 Dimension: The dimension of interpolation axes. (2~4 axes)

I32 *Axis_ID_Array: The axis ID array from 0 to 65535.

For EMX-100: I32 *Axis_ID_Array: The Axis ID array from 0 to 65535. First element of array is master and puts it in ascending order. For instance Axis_ID_Array[2] = {1,3} or Axis_ID_Array[2] = {0,3} or Axis_ID_Array[2] = {0,1} or Axis_ID_Array[2] = {2,3}

I32 *Position_Array: Absolute position array. (unit: pulse)

I32 Max_Linear_Speed: Maximum linear interpolation speed (unit: pulse/sec)

For EMX-100: I32 Max_Linear_Speed: Maximum linear interpolation speed; Its range is 1 ~ 8,000,000 (Unit: pulse/sec)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

//…Initial card

I32 Dimension = 4;

I32 Master_Axis_ID = 1; //Master axis

I32 Axis_ID_Array[4] = { 1, 2, 3, 4}; //Axis ID 1 is master axis.

I32 Position_Array [4] = {10000, 20000, 30000, 40000 };

I32 Max_Linear_Speed = 10000;

I32 Ret;

APS_set_axis_param(Master_Axis_ID, PRA_CURVE, 0); //Set T-curve

APS_set_axis_param(Master_Axis_ID, PRA_ACC, 100000); //Set acceleration

```
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 100000 ); //Set deceleration

Ret = APS_absolute_linear_move ( Dimension, Axis_ID_Array, Position_Array,
Max_Linear_Speed );
…
```

See also:

APS_relative_linear_move()

APS_relative_linear_move	Begin a relative distance linear interpolation

Support Products: PCI-8253/56, PCI-8392(H), MNET-4XMO-(C),HSL-4XMO, PCI(e)-8154/8158, PCI-8102/PCI-C154(+), EMX-100 , PCI-8254/58 / AMP-204/8C, AMP-104C

Descriptions:

This function is used to start a relative linear interpolation positioning motion. Although there is maximum speed setting in function parameter, the traveling distance and accelerating rate may not be enough due to user's setting to reach the maximum speed. The speed profile's acceleration and deceleration rate and curve are set by axis parameter function. Because the speed parameter is in vector direction, this function will take the master axis's acceleration and deceleration time constant to calculate. The master axis is the minimum axis number that user perform an interpolation.

This function is 'fire-and-forget' type. That means user's program or procedure will not be pended during axis traveling. Users must use motion status checking function or interrupt event waiting function to wait it done.

During the axis traveling, users can start a new move command including stop command to override the previous one. The axis will be switched to new command immediately according to new setting of target position, new speed.

The overridden command must have the same dimension and axis ID of previous one. These two commands can't be overridden by other motion modes like home operation. Users must stop axis motion before switching to those modes mentioned above.

Note: The axes specified in Axis_ID_Array must be of the same card.

For EMX-100 , this function is used for linear interpolation positioning motion using relative postion.It only supports two axes motion at the same time with the same device. The master axis is defined as the first axis ID that user gives for interpolation. Two speed profile parameters travel distance and maximum velocity are given by user, and other parameters like start velocity, acceleration rate, deceleration rate and s-factor are configurable by master axis's parameter table. The actual command velocity may not reach maximum velocity due to small traveling distance or accelerating rate are given.

This function uses 'fire-and-forget' mode to avoid blocking user's program or procedure during axis traveling. Users can read motion status MDN to check the motion is completed (MDN = 1) or not (MDN = 0). Except stop command, users CAN NOT start any new move command before previous motion is completed.

Note: The axes specified in Axis_ID_Array must be of the same card.

Syntax:

C/C++:

I32 FNTYPE APS_relative_linear_move(I32 Dimension, I32 *Axis_ID_Array, I32 *Distance_Array, I32 Max_Linear_Speed);

Visual Basic:

APS_relative_linear_move(ByVal Dimension As Long, Axis_ID_Array As Long, Distance_Array As Long, ByVal Max_Linear_Speed As Long) As Long

Parameters:

I32 Dimension: The dimension of interpolation axes. (2~4 axes)

I32 *Axis_ID_Array: The Axis ID array from 0 to 65535.

For EMX-100: The Axis ID array from 0 to 65535. First element of array is master and puts it in ascending order. For instance Axis_ID_Array[2] = {1,3} or Axis_ID_Array[2] = {2,3} or Axis_ID_Array[2] = {0,1}

I32 *Distance_Array: Relative distance array. (unit: pulse)

I32 Max_Linear_Speed: Maximum linear interpolation speed (unit: pulse/sec)

For EMX-100: I32 Max_Linear_Speed: Maximum linear interpolation speed; Its range is 1 ~ 8,000,000 (Unit: pulse/sec)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
//…Initial card
I32 Dimension = 4;
I32 Master_Axis_ID = 0;
I32 Axis_ID_Array[4] = {0, 1, 2, 3}; //Axis ID 0 is master axis.
I32 Distance_Array[4] = {10000, 20000, 30000, 40000 };
I32 Max_Linear_Speed = 10000;
I32 Ret;
APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 100000 ); //Set acceleration
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 100000 ); //Set deceleration

Ret = APS_relative_linear_move( Dimension, Axis_ID_Array, Distance_Array, Max_Linear_Speed );
```

…

See also:

APS_relative_linear_move();APS_set_axis_param();

APS_absolute_arc_move	Begin an absolute position circular interpolation

Support Products: PCI-8253/56, PCI-8392(H) , MNET-4XMO-(C) , HSL-4XMO, PCI(e)-8154/8158, PCI-8102/PCI-C154(+), PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to start an absolute circular interpolation positioning motion. User must specify absolute center position and traveling angle for circular interpolation. The speed profile's acceleration and deceleration rate are set by axis parameter function. The following axis parameter should be setting before you calling this function.

PRA_CURVE
PRA_ACC
PRA_DEC
PRA_VS
PRA_VE

The details of parameters please refer the axis parameter table.

Symbol define	ParamNo	Description
PRA_SF	0x20	Move S-factor
PRA_ACC	0x21	Acceleration rate
PRA_DEC	0x22	Deceleration rate
PRA_VS	0x23	Start velocity
PRA_VM	0x24	Maximum velocity
PRA_VE	0x25	End velocity

Although there is maximum speed setting in function parameter, the traveling distance and accelerating rate may not be enough due to user's setting to reach the maximum speed. Because the speed parameter is in vector direction (Tangent to the circular), this function will take the master axis's acceleration and deceleration time constant to calculate. The master axis is the minimum axis number that user perform an interpolation. For example, Axis ID 2 and 3 are performing a circular interpolation. The Axis ID 2 is the master axis.

This function is 'fire-and-forget' type. That means user's program or procedure will not be pended during axis traveling. Users must use motion status checking function or interrupt event waiting function to wait it done. The motion status "circular interpolation signal (CIP)" of each axis performing a circular interpolation will be turn on when command is started and will be turned off at command is finished. If circular interpolation is stop normally, the normal stop signal (NSTP) will be turned on. On the contrary, if circular interpolation is stopped abnormally (such as ALM, EMG, SEMG and so on is turned on), abnormal stop signal (ASTP) will be turned on.

During the axis traveling, users can start a new move command including stop command to override the previous one (The dimension and Axis_ID_Array must be the same). The axis will be switched to new command immediately according to new setting of target center position, new speed profile.
This command can't be overridden by other motion modes like home operation. Users must stop axis motion before switching to those modes mentioned above.

Note: The 2 axes specified in Axis_ID_Array must be of the same card.

Syntax:
C/C++:
I32 FNTYPE APS_absolute_arc_move(I32 Dimension, I32 *Axis_ID_Array, I32 *Center_Pos_Array, I32 Max_Arc_Speed, I32 Angle);
Visual Basic:
APS_absolute_arc_move(ByVal Dimension As Long, Axis_ID_Array As Long, Center_Pos_Array As Long, ByVal Max_Arc_Speed As Long, ByVal Angle As Long)As Long

Parameters:
I32 Dimension: The dimension of interpolation axes. (The maximum dimensions refer to product specification)
I32 *Axis_ID_Array: The Axis ID array from 0 to 65535.
I32 *Center_Pos_Array: Absolute circular center position. Unit: pulse.
I32 Max_Arc_Speed: Maximum circular interpolation speed (Circular tangent speed). Unit: pulse/sec
I32 Angle: Travel angle. Value range: -360 ~360 degree. Positive for counterclockwise.

Return Values:
I32 Error code: Please refer to APS Functions Return Code.

Example:
```
I32 Dimension = 2;
I32 Axis_ID_Array[2] = { 2, 4 }; //Axis_ID 2 is the master axis.
I32 Master_Axis_ID = 2; //Axis_ID 2 is the master axis.
I32 Center_Pos_Array[2] = {100000, 0};
I32 Max_Arc_Speed = 10000; // pulse/sec
I32 Angle = -180; // clockwise 180 degree.
I32 Ret; //Return code.
```

```
//…
APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 100000 ); //Set acceleration
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 100000 ); //Set deceleration
Ret = APS_absolute_arc_move( Dimension, Axis_ID_Array, Center_Pos_Array,
Max_Arc_Speed, Angle ); //Perform a circular interpolation
```

See also:

APS_relative_arc_move();APS_set_axis_param();APS_get_axis_param ();
APS_motion_status();APS_stop_move();APS_emg_stop()

APS_relative_arc_move	Begin a relative distance circular interpolation

Support Products: PCI-8253/56, PCI-8392(H) , MNET-4XMO-(C) , HSL-4XMO, PCI(e)-8154/8158, PCI-8102/PCI-C154(+), PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to start an relative circular interpolation positioning motion. User must specified a center position relative current commend position and traveling angle for circular interpolation. The speed profile's acceleration and deceleration rate and curve are set by axis parameter function. The following axis parameter should be setting before you calling this function.

PRA_CURVE
PRA_ACC
PRA_DEC
PRA_VS
PRA_VE

The details of parameters please refer the axis parameter table.

Symbol define	ParamNo	Description
PRA_SF	0x20	Move S-factor
PRA_ACC	0x21	Acceleration rate
PRA_DEC	0x22	Deceleration rate
PRA_VS	0x23	Start velocity
PRA_VM	0x24	Maximum velocity
PRA_VE	0x25	End velocity

Although there is maximum speed setting in function parameter, the traveling distance and accelerating rate may not be enough due to user's setting to reach the maximum speed. Because the speed parameter is in vector direction(tangent to the circular), this function will take the master axis's acceleration and deceleration time constant to calculate. The master axis is the minimum axis number that user perform an interpolation. For example, Axis ID 2 and 3 are performing a circular interpolation. Therefore, the Axis ID 2 is the master axis.

This function is 'fire-and-forget' type. That means user's program or procedure will not be pended during axis traveling. Users must use motion status checking function or interrupt event waiting function to wait it done. Motion status: in circular interpolation signal (CIP) will be turn on when it start and will be turn off at command is finished. If circular interpolation is stop normally, the normal stop signal (NSTP) will be turn on. On the contrary, circular interpolation is stopped abnormally (ALM, EMG, SEMG, and so on), abnormal stop signal (ASTP) will be turn on.

During the axis traveling, users can start a new move command including stop command to override the previous one (The dimension and Axis_ID_Array must be the same). The axis will be switched to new command immediately according to new setting of target center position, new speed profile.
This command can't be overridden by other motion modes like home operation. Users must stop axis motion before switching to those modes mentioned above.

Note: The 2 axes specified in Axis_ID_Array must be of the same card.

Syntax:
C/C++:
I32 FNTYPE APS_relative_arc_move(I32 Dimension, I32 *Axis_ID_Array, I32 *Center_Offset_Array, I32 Max_Arc_Speed, I32 Angle);
Visual Basic:
APS_relative_arc_move(ByVal Dimension As Long, Axis_ID_Array As Long, Center_Offset_Array As Long, ByVal Max_Arc_Speed As Long, ByVal Angle As Long) As Long

Parameters:
I32 Dimension: The dimension of interpolation axes. (The maximum dimensions refer to product specification)
I32 *Axis_ID_Array: The Axis ID array from 0 to 65535.
I32 *Center_Offset_Array: circular center position relative to current command position. Unit: pulse
I32 Max_Arc_Speed: Maximum circular interpolation speed (Circular tangent speed). Unit: pulse/sec
I32 Angle: Travel angle. Value range: -360 ~360 degree. Positive for counterclockwise.

Return Values:
I32 Error code: Please refer to APS Functions Return Code.

Example:
```
I32 Dimension = 2;
I32 Axis_ID_Array[2] = { 1, 3}; //Axis_ID 1 is the master axis.
I32 Master_Axis_ID = 1; //Axis_ID 1 is the master axis.
I32 Center_Offset_Array [2] = {300000, 0};
I32 Max_Arc_Speed = 20000; // pulse/sec
```

```
I32 Angle = 90; // counterclockwise 90 degree.
I32 Ret; //Return code.

//…
APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 100000 ); //Set acceleration
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 100000 ); //Set deceleration
Ret = APS_relative_arc_move( Dimension, Axis_ID_Array, Center_Offset_Array,
Max_Arc_Speed, Angle ); //Perform a circular interpolation
```

See also:

APS_absolute_arc_move ();APS_set_axis_param ();APS_get_axis_param ();
APS_motion_status(); APS_stop_move();APS_emg_stop()

APS_absolute_arc_move_3pe	Begin an absolute position circular interpolation by pass and end point mode

Support Products: PCI-8253/56

Descriptions:

This function is used to start an absolute circular interpolation positioning motion. User must specify absolute pass position and end position for circular interpolation. The speed profile's acceleration and deceleration rate are set by axis parameter function. The following axis parameter should be setting before you calling this function.

PRA_CURVE
PRA_ACC
PRA_DEC
PRA_VS
PRA_VE

The details of parameters please refer the axis parameter table.

Although there is maximum speed setting in function parameter, the traveling distance and accelerating rate may not be enough due to user's setting to reach the maximum speed. Because the speed parameter is in vector direction (Tangent to the circular), this function will take the master axis's acceleration and deceleration time constant to calculate. The master axis is the minimum axis number that user perform an interpolation. For example, Axis ID 2 and 3 are performing a circular interpolation. The Axis ID 2 is the master axis.

This function is 'fire-and-forget' type. That means user's program or procedure will not be pended during axis traveling. Users must use motion status checking function or interrupt event waiting function to wait it done. The motion status "circular interpolation signal (CIP)" of each axis performing a circular interpolation will be turn on when command is started and will be turned off at command is finished. If circular interpolation is stop normally, the normal stop signal (NSTP) will be turned on. On the contrary, if circular interpolation is stopped abnormally (such as ALM, EMG, SEMG and so on is turned on), abnormal stop signal (ASTP) will be turned on.

During the axis traveling, users can start a new move command including stop command to override the previous one (The dimension and Axis_ID_Array must be the same). The axis will be switched to new command immediately according to new setting of target center position, new speed profile.

This command can't be overridden by other motion modes like home operation. Users must stop axis motion before switching to those modes mentioned above.

Note:

1. This mode support 2D and 3D circular interpolation motion.

2. The 2 or 3 axes specified in Axis_ID_Array must be of the same card.

3. Circular interpolation by pass and end point mode do not support full circle.

Syntax:

C/C++:

I32 FNTYPE APS_absolute_arc_move_3pe(I32 Dimension, I32 *Axis_ID_Array, I32 *Pass_Pos_Array, I32 *End_Pos_Array, I32 Max_Arc_Speed);

Visual Basic:

APS_absolute_arc_move_3pe(ByVal Dimension As Long, Axis_ID_Array As Long, Pass_Pos_Array As Long, End_Pos_Array As Long, ByVal Max_Arc_Speed As Long)As Long

Parameters:

I32 Dimension: The dimension of interpolation axes. (The maximum dimension is support to 3D)

I32 *Axis_ID_Array: The Axis ID array from 0 to 65535.

I32 *Pass_Pos_Array: Absolute pass position. Unit: pulse.

I32 *End_Pos_Array: Absolute end position. Unit: pulse.

I32 Max_Arc_Speed: Maximum circular interpolation speed (Circular tangent speed). Unit: pulse/sec

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Dimension = 3;
I32 Axis_ID_Array[3] = { 2, 3, 4 }; //Axis_ID 2 is the master axis.
I32 Master_Axis_ID = 2; //Axis_ID 2 is the master axis.
I32 Pass_Pos_Array[3] = {50000, 50000, 50000};
I32 End_Pos_Array[3] = {100000, 100000, 0}
I32 Max_Arc_Speed = 400000; // pulse/sec
I32 Ret; //Return code.

//…
APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 1000000 ); //Set acceleration
```

```
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 1000000 ); //Set deceleration
Ret = APS_absolute_arc_move_3pe( Dimension, Axis_ID_Array, Pass_Pos_Array,
End_Pos_Array, Max_Arc_Speed ); //Perform a circular interpolation
```

See also:

APS_absolute_arc_move();APS_relative_arc_move();APS_relative_arc_move_3pe();
APS_set_axis_param();APS_get_axis_param();APS_motion_status();APS_stop_move();
APS_emg_stop()

APS_relative_arc_move_3pe	Begin a relative distance circular interpolation by pass and end mode

Support Products: PCI-8253/56

Descriptions:

This function is used to start a relative circular interpolation positioning motion. User must specify pass position and end position relative current commend position for circular interpolation. The speed profile's acceleration and deceleration rate and curve are set by axis parameter function. The following axis parameter should be setting before you calling this function.
PRA_CURVE
PRA_ACC
PRA_DEC
PRA_VS
PRA_VE
The details of parameters please refer the axis parameter table.

Although there is maximum speed setting in function parameter, the traveling distance and accelerating rate may not be enough due to user's setting to reach the maximum speed. Because the speed parameter is in vector direction (tangent to the circular), this function will take the master axis's acceleration and deceleration time constant to calculate. The master axis is the minimum axis number that user perform an interpolation. For example, Axis ID 2 and 3 are performing a circular interpolation. Therefore, the Axis ID 2 is the master axis.

This function is 'fire-and-forget' type. That means user's program or procedure will not be pended during axis traveling. Users must use motion status checking function or interrupt event waiting function to wait it done. Motion status: in circular interpolation signal (CIP) will be turn on when it start and will be turn off at command is finished. If circular interpolation is stop normally, the normal stop signal (NSTP) will be turn on. On the contrary, circular interpolation is stopped abnormally (ALM, EMG, SEMG, and so on), abnormal stop signal (ASTP) will be turn on.

During the axis traveling, users can start a new move command including stop command to override the previous one (The dimension and Axis_ID_Array must be the same). The axis will be switched to new command immediately according to new setting of target center position, new speed profile.
This command can't be overridden by other motion modes like home operation. Users must stop axis motion before switching to those modes mentioned above.

Note:

1. This mode support 2D and 3D circular interpolation motion.

2. The 2 or 3 axes specified in Axis_ID_Array must be of the same card.

3. Circular interpolation by pass and end point mode do not support full circle.

Syntax:

C/C++:

I32 FNTYPE APS_relative_arc_move_3pe(I32 Dimension, I32 *Axis_ID_Array, I32 *Pass_PosOffset_Array, I32 *End_PosOffset_Array, I32 Max_Arc_Speed);

Visual Basic:

APS_relative_arc_move_3pe(ByVal Dimension As Long, Axis_ID_Array As Long, Pass_PosOffset_Array As Long, End_PosOffset_Array As Long, ByVal Max_Arc_Speed As Long) As Long

Parameters:

I32 Dimension: The dimension of interpolation axes. (The maximum dimension is support to 3D).

I32 *Axis_ID_Array: The Axis ID array from 0 to 65535.

I32 *Pass_PosOffset_Array: circular pass position relative to current command position. Unit: pulse

I32 *End_PosOffset_Array: circular end position relative to current command position. Unit: pulse

I32 Max_Arc_Speed: Maximum circular interpolation speed (Circular tangent speed). Unit: pulse/sec

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Dimension = 3;
I32 Axis_ID_Array[3] = { 0, 1, 2}; //Axis_ID 0 is the master axis.
I32 Master_Axis_ID = 0; //Axis_ID 0 is the master axis.
I32 Pass_PosOffset_Array [3] = {50000, 50000, 50000};
I32 End_PosOffset_Array[3] = {50000, 50000, -50000};
I32 Max_Arc_Speed = 200000; // pulse/sec
I32 Ret; //Return code.
```

```
//…
APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 1000000 ); //Set acceleration
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 1000000 ); //Set deceleration
Ret = APS_relative_arc_move_3pe( Dimension, Axis_ID_Array, Pass_PosOffset_Array,
End_PosOffset_Array, Max_Arc_Speed ); //Perform a circular interpolation
```

See also:

APS_relative_arc_move();APS_absolute_arc_move();APS_absolute_arc_move_3pe();
APS_set_axis_param();APS_get_axis_param();APS_motion_status();APS_stop_move();APS_
emg_stop()

APS_absolute_helix_move	Begin an absolute position helical interpolation

Support Products: PCI-8253/56

Descriptions:

This function is used to start an absolute helical interpolation positioning motion. User must specify absolute circle center position (2D), pitch length, total screw height, and move direction for helical interpolation. The speed profile's acceleration and deceleration rate are set by axis parameter function. The following axis parameter should be setting before you calling this function.

PRA_CURVE
PRA_ACC
PRA_DEC
PRA_VS
PRA_VE

The details of parameters please refer the axis parameter table.

Although there is maximum speed setting in function parameter, the traveling distance and accelerating rate may not be enough due to user's setting to reach the maximum speed. Because the speed parameter is in vector direction (Tangent to the circular), this function will take the master axis's acceleration and deceleration time constant to calculate. The master axis is the minimum axis number that user perform an interpolation. For example, Axis ID 2 and 3 are performing a circular interpolation and synchronized linear travel in axis ID 4. The Axis ID 2 is the master axis.

This function is 'fire-and-forget' type. That means user's program or procedure will not be pended during axis traveling. If helical interpolation is stop normally, the normal stop signal (NSTP) will be turned on. On the contrary, if helical interpolation is stopped abnormally (such as ALM, EMG, SEMG and so on is turned on), abnormal stop signal (ASTP) will be turned on.

During the axis traveling, users can start a new move command including stop command to override the previous one (The dimension and Axis_ID_Array must be the same). The axis will be switched to new command immediately according to new setting of target center position, new speed profile.

This command can't be overridden by other motion modes like home operation. Users must stop axis motion before switching to those modes mentioned above.

Note:

1. Helical interpolation just supports 3D coordinate space.

2. The last axis number in Axis ID array must be linear axis.

3. Circle center position just support 2D

Syntax:

C/C++:

I32 FNTYPE APS_absolute_helix_move(I32 Dimension, I32 *Axis_ID_Array, I32 *Center_Pos_Array, I32 Max_Arc_Speed, I32 Pitch, I32 TotalHeight, I32 CwOrCcw);

Visual Basic:

APS_absolute_helix_move(ByVal Dimension As Long, Axis_ID_Array As Long, Center_Pos_Array As Long, ByVal Max_Arc_Speed As Long, ByVal Pitch As Long, ByVal TotalHeight As Long, ByVal CwOrCcw As Long)As Long

Parameters:

I32 Dimension: The dimension of interpolation axes. (Just support 3D)

I32 *Axis_ID_Array: The Axis ID array from 0 to 65535.

I32 *Center_Pos_Array: Absolute pass position. Unit: pulse.

I32 Max_Arc_Speed: Maximum circular interpolation speed (Circular tangent speed). Unit: pulse/sec

I32 Pitch: The pitch of helix. Unit: pulse

I32 TotalHeight: The depth of helix. Unit: pulse

I32 CwOrCcw: Move direction

CwOrCcw = 0 ---→ Clockwise

CwOrCcw = 1----→ Counterclockwise

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Dimension = 3;
I32 Axis_ID_Array[3] = { 2, 3, 4 }; //Axis_ID 2 is the master axis.
I32 Master_Axis_ID = 2; //Axis_ID 2 is the master axis.
I32 Center_Pos_Array[2] = {50000, 0};
I32 Max_Arc_Speed = 400000; // pulse/sec
I32 Pitch = 2500;
I32 TotalHeight = 5000;
I32 CwOrCcw = 1; // Counterclockwise
I32 Ret; //Return code.

//…
```

```
APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 1000000 ); //Set acceleration
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 1000000 ); //Set deceleration
Ret = APS_absolute_helix_move ( Dimension, Axis_ID_Array, Center_Pos_Array,
Max_Arc_Speed , Pitch, TotalHeight, CwOrCcw ); //Perform a helical interpolation
```

See also:

APS_relative_helix_move();APS_set_axis_param ();APS_get_axis_param ();
APS_motion_status();APS_stop_move(), APS_emg_stop()

APS_relative_helix_move	Begin a relative distance helical interpolation

Support Products: PCI-8253/56

Descriptions:

This function is used to start a relative helical interpolation positioning motion. User must specify circle center position (2D) relative current command position, pitch length, total screw height, and move direction for helical interpolation. The speed profile's acceleration and deceleration rate and curve are set by axis parameter function. The following axis parameter should be setting before you calling this function.

PRA_CURVE
PRA_ACC
PRA_DEC
PRA_VS
PRA_VE

The details of parameters please refer the axis parameter table.

Although there is maximum speed setting in function parameter, the traveling distance and accelerating rate may not be enough due to user's setting to reach the maximum speed. Because the speed parameter is in vector direction (Tangent to the circular), this function will take the master axis's acceleration and deceleration time constant to calculate. The master axis is the minimum axis number that user perform an interpolation. For example, Axis ID 2 and 3 are performing a circular interpolation and synchronized linear travel in axis ID 4. The Axis ID 2 is the master axis.

This function is 'fire-and-forget' type. That means user's program or procedure will not be pended during axis traveling. If helical interpolation is stop normally, the normal stop signal (NSTP) will be turned on. On the contrary, if helical interpolation is stopped abnormally (such as ALM, EMG, SEMG and so on is turned on), abnormal stop signal (ASTP) will be turned on.

During the axis traveling, users can start a new move command including stop command to override the previous one (The dimension and Axis_ID_Array must be the same). The axis will be switched to new command immediately according to new setting of target center position, new speed profile.

This command can't be overridden by other motion modes like home operation. Users must stop axis motion before switching to those modes mentioned above.

Note:

1. Helical interpolation just supports 3D coordinate space.

2. The last axis number in Axis ID array must be linear axis.

3. Circle center position just support 2D

Syntax:

C/C++:

I32 FNTYPE APS_relative_helix_move(I32 Dimension, I32 *Axis_ID_Array, I32 *Center_PosOffset_Array, I32 Max_Arc_Speed, I32 Pitch, I32 TotalHeight, I32 CwOrCcw);

Visual Basic:

APS_relative_helix_move(ByVal Dimension As Long, Axis_ID_Array As Long, Center_PosOffset_Array As Long, ByVal Max_Arc_Speed As Long, ByVal Pitch As Long, ByVal TotalHeight As Long, ByVal CwOrCcw As Long)As Long

Parameters:

I32 Dimension: The dimension of interpolation axes. (Just support 3D)

I32 *Axis_ID_Array: The Axis ID array from 0 to 65535.

I32 *Center_PosOffset_Array: Circular center position relative to current command position. Unit:pulse

I32 Max_Arc_Speed: Maximum circular interpolation speed (Circular tangent speed). Unit: pulse/sec

I32 Pitch: The pitch of helix. Unit: pulse

I32 TotalHeight: The depth of helix. Unit: pulse

I32 CwOrCcw: Move direction

CwOrCcw = 0 ---→ Clockwise

CwOrCcw = 1----→ Counterclockwise

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Dimension = 3;
I32 Axis_ID_Array[3] = { 2, 3, 4 }; //Axis_ID 2 is the master axis.
I32 Master_Axis_ID = 2; //Axis_ID 2 is the master axis.
I32 Center_PosOffset_Array[2] = {50000, 0};
I32 Max_Arc_Speed = 400000; // pulse/sec
I32 Pitch = 2500;
```

```
I32 TotalHeight = 5000;
I32 CwOrCcw = 1; // Counterclockwise
I32 Ret; //Return code.

//…
APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 1000000 ); //Set acceleration
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 1000000 ); //Set deceleration
Ret = APS_absolute_helix_move ( Dimension, Axis_ID_Array, Center_PosOffset_Array,
Max_Arc_Speed , Pitch, TotalHeight, CwOrCcw ); //Perform a helical interpolation
```

See also:

APS_absolute_helix_move();APS_set_axis_param();
APS_get_axis_param();APS_motion_status();APS_stop_move();APS_emg_stop()

APS_absolute_helical_move	Begin an absolute position helical interpolation

Support Products: PCI(e)-8154/8158

Descriptions:

This function is used to start an absolute helical interpolation positioning motion. User must specify absolute circle center position (2D), absolute end position, pitch, and move direction for helical interpolation. The speed profile's acceleration and deceleration rate are set by axis parameter function. The following axis parameter should be setting before you calling this function.

PRA_CURVE

PRA_ACC

PRA_DEC

PRA_VS

PRA_VE

The details of parameters please refer the axis parameter table.

Although there is maximum speed setting in function parameter, the traveling distance and accelerating rate may not be enough due to user's setting to reach the maximum speed. Because the speed parameter is in vector direction (Tangent to the circular), this function will take the master axis's acceleration and deceleration time constant to calculate. The master axis is the minimum axis number that user perform an interpolation. For example, Axis ID 0 and 1 are performing a circular interpolation and synchronized linear travel in axis ID 2. The Axis ID 0 is the master axis.

This function is 'fire-and-forget' type. That means user's program or procedure will not be pended during axis traveling. If helical interpolation is stop normally, the normal stop signal (NSTP) will be turned on. On the contrary, if helical interpolation is stopped abnormally (such as ALM, EMG, SEMG and so on is turned on), abnormal stop signal (ASTP) will be turned on.

Syntax:

C/C++:

I16 APS_absolute_helical_move(I32 *Axis_ID_Array, I32 *Center_Pos_Array, I32 *End_Pos_Array, I32 Pitch, I32 Dir, I32 Max_Speed);

Visual Basic:

APS_absolute_helical_move(Axis_ID_Array As Long, Center_Pos_Array As Long, End_Pos_Array As Long, ByVal Pitch As Long, ByVal Dir As Long, ByVal Max_Speed As Long)As Long

Parameters:

I32 *Axis_ID_Array: The Axis ID array. Each helical set needs 4 axes, for example, PCIe-8154 needs axis 0~3, and PCIe-8158 need 0~3 or 4~7.

I32 *Center_Pos_Array: Absolute center position. Unit: pulse.

I32 *End_Pos_Array: Absolute end position. Unit: pulse.

I32 Max_Speed: Maximum circular interpolation speed (Circular tangent speed). Unit: pulse/sec

I32 Pitch: The pitch of helical. Unit: pulse

I32 Dir: Move direction

Dir = 0 --- Clockwise

Dir = 1---- Counterclockwise

Example:

I32 AxisArray[4] = {0, 1, 2, 3}; //Set axis ID. In this example, axis 0&1 run circular interpolation, axis 2 runs vertical direction, and axis 3 is virtual axis for internal calculation.

I32 Center_Pos_Array[2] = {1000, 2000}; //Set center position to move (unit: pulse)

I32 End_Pos_Array[2] = {2000, 4000}; // Set end position to move (unit: pulse)

I32 MaxVel = 5000; //Set maximum velocity in units of pulse per second

I32 Pitch = 500; //Set pitch length

I32 Dir = 0; //Set direction clockwise

```
for( int i = 0; i < total_axis; i++)
{
    APS_set_axis_param(i, PRA_CURVE, 0 ); //Set T-curve
    APS_set_axis_param(i, PRA_ACC, 50000 ); //Set acceleration
    APS_set_axis_param(i, PRA_DEC, 50000 ); //Set deceleration
    APS_set_axis_param(i, PRA_VS, 0 ); //Set start velocity
}
```

APS_absolute_helical_move (AxisArray, Center_Pos_Array, End_Pos_Array, Pitch, Dir, MaxVel);

See also:

APS_relative_helical_move();APS_set_axis_param();
APS_get_axis_param();APS_motion_status();APS_stop_move();APS_emg_stop()

APS_relative_helical_move	Begin a relative distance helical interpolation

Support Products: PCI(e)-8154/8158

Descriptions:

This function is used to start a relative helical interpolation positioning motion. User must specify circle center offset position (2D), circle end offset position, pitch, and move direction for helical interpolation. The speed profile's acceleration and deceleration rate are set by axis parameter function. The following axis parameter should be setting before you calling this function.

PRA_CURVE

PRA_ACC

PRA_DEC

PRA_VS

PRA_VE

The details of parameters please refer the axis parameter table.

Although there is maximum speed setting in function parameter, the traveling distance and accelerating rate may not be enough due to user's setting to reach the maximum speed. Because the speed parameter is in vector direction (Tangent to the circular), this function will take the master axis's acceleration and deceleration time constant to calculate. The master axis is the minimum axis number that user perform an interpolation. For example, Axis ID 0 and 1 are performing a circular interpolation and synchronized linear travel in axis ID 2. The Axis ID 0 is the master axis.

This function is 'fire-and-forget' type. That means user's program or procedure will not be pended during axis traveling. If helical interpolation is stop normally, the normal stop signal (NSTP) will be turned on. On the contrary, if helical interpolation is stopped abnormally (such as ALM, EMG, SEMG and so on is turned on), abnormal stop signal (ASTP) will be turned on.

Syntax:

C/C++:

I16 APS_relative_helical_move(I32 *Axis_ID_Array, I32 *Center_Offset_Array, I32 *End_Offset_Array, I32 Pitch, I32 Dir, I32 Max_Speed);

Visual Basic:

APS_relative_helical_move(Axis_ID_Array As Long, Center_Offset_Array As Long, End_Offset_Array As Long, ByVal Pitch As Long, ByVal Dir As Long, ByVal Max_Speed As Long)As Long

Parameters:

I32 *Axis_ID_Array: The Axis ID array. Each helical set needs 4 axes, for example, PCIe-8154 needs axis 0~3, and PCIe-8158 need 0~3 or 4~7.

I32 *Center_Offset_Array: Relative center offset position. Unit: pulse.

I32 *End_Offset_Array: Relative end offset position. Unit: pulse.

I32 Max_Speed: Maximum circular interpolation speed (Circular tangent speed). Unit: pulse/sec

I32 Pitch: The pitch of helical. Unit: pulse

I32 Dir: Move direction

Dir = 0 --- Clockwise

Dir = 1---- Counterclockwise

Example:

```
I32 AxisArray[4] = {0, 1, 2, 3}; //Set axis ID. In this example, axis 0&1 run circular interpolation, axis 2 runs vertical direction, and axis 3 is virtual axis for internal calculation.
I32 Center_Offset_Array[2] = {1000, 2000}; //Set center position to move (unit: pulse)
I32 End_Offset_Array[2] = {2000, 4000}; // Set end position to move (unit: pulse)
I32 MaxVel = 5000; //Set maximum velocity in units of pulse per second
I32 Pitch = 500; //Set pitch length
I32 Dir = 0; //Set direction clockwise
for( int i = 0; i < total_axis; i++)
{
    APS_set_axis_param(i, PRA_CURVE, 0 ); //Set T-curve
    APS_set_axis_param(i, PRA_ACC, 50000 ); //Set acceleration
    APS_set_axis_param(i, PRA_DEC, 50000 ); //Set deceleration
    APS_set_axis_param(i, PRA_VS, 0 ); //Set start velocity
}
APS_relative_helical_move (AxisArray, Center_Offset_Array, End_Offset_Array, Pitch, Dir, MaxVel);
```

See also:

APS_absolute_helical_move();APS_set_axis_param ();APS_get_axis_param ();
APS_motion_status();APS_stop_move();APS_emg_stop()

10. Advanced single move & interpolation

APS_ptp	Begin a single move

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a single move by axis. No any suffix represents that no any motion profile is necessary to perform a single move. Other motion profiles are set in axis parameters. User could refer to axis parameter table for the details.

Syntax:

C/C++:

I32 FNTYPE APS_ptp(I32 Axis_ID, I32 Option, F64 Position, ASYNCALL *Wait);

Visual Basic:

APS_ptp(ByVal Axis_ID As Long, ByVal Option As Long, ByVal Position As Double, Wait As ASYNCALL) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

For PCI-8254/58 / AMP-204/8C:

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode						Force Abort	Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9: This bit is used to assign profile abort behavior when different direction or insufficient decelerating distance happened.

0: Profile will change smoothly. (Default value)

1: Profile will change immediately.

Bit 10 ~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting

0001b(1): Buffered

0010b(2): Blending low

0011b(3): Blending previous

0100b(4): Blending next

0101b(5): Blending high

Bit 16~: Reserved for future, set to 0.

F64 Position: A value specifies how many position/distance to move.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

For PCIe-833x, ECAT-4XMO:

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting

0001b(1): Buffered

0010b(2): Blending low

0011b(3): Blending previous

0100b(4): Blending next

0101b(5): Blending high

Bit 16~: Reserved for future, set to 0.

F64 Position: A value specifies how many position/distance to move.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 opt = 0x1000; //absolute, not wait trigger, Buffered mode
ASYNCALL *wait = NULL; //A waiting call

//An absolute move to position 10000
APS_ptp( Axis_ID, opt, 10000, wait );
```

See also:

APS_ptp_v	Begin a single move with Vm profile

Support Products: PCI(e)-8154/58, PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a single move by axis. The _v suffix represents that only one motion profile, that is Vm, is necessary to perform a single move. Other motion profiles are set in axis parameters. User could refer to axis parameter table for the details.

Syntax:

C/C++:

I32 FNTYPE APS_ptp_v(I32 Axis_ID, I32 Option, F64 Position, F64 Vm, ASYNCALL *Wait);

Visual Basic:

APS_ptp_v(ByVal Axis_ID As Long, ByVal Option As Long, ByVal Position As Double, ByVal Vm As Double, Wait As ASYNCALL) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

For PCI(e)-8154/58:

7	6	5	4	3	2	1	0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~11: Reserved for future, set to 0.

Bit12~15: Buffer mode:

0000b(0): Aborting, would start position move, or override old movement with new position and speed while in motion.

00001b(): Buffered. **Note: It is reserved for future.**

F64 Position: A value specifies how many position/distance to move.

F64 Vm: A value specifies the maximum velocity.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

For PCI-8254/58 / AMP-204/8C:

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode						Force Abort	Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9: This bit is used to assign profile abort behavior when different direction or insufficient decelerating distance happened.

0: Profile will change smoothly. (Default value)

1: Profile will change immediately.

Bit 10~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting

0001b(1): Buffered

0010b(2): Blending low

0011b(3): Blending previous

0100b(4): Blending next

0101b(5): Blending high

Bit 16~: Reserved for future, set to 0.

F64 Position: A value specifies how many position/distance to move.

F64 Vm: A value specifies the maximum velocity.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

For PCIe-833x, ECAT-4XMO :

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting

0001b(1): Buffered

0010b(2): Blending low

0011b(3): Blending previous

0100b(4): Blending next

0101b(5): Blending high

Bit 16~: Reserved for future, set to 0.

F64 Position: A value specifies how many position/distance to move.

F64 Vm: A value specifies the maximum velocity.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 opt = 0x1000;    //absolute, aborting mode
ASYNCALL *wait = NULL; //A waiting call

//An absolute move to position(10000) with Vm(100000)
APS_ptp_v ( Axis_ID, opt, 10000, 100000, wait );
```

See also:

APS_relative_move();APS_absolute_move();APS_relative_move_ovrd();APS_absolute_move_ovrd()

APS_ptp_all	Begin a single move with all motion profile

Support Products: PCI(e)-8154/58, PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a single move. The _all suffix represents that all motion profiles, including Vs, Vm, Ve, Acc, Dec and SFac, are necessary to perform a single move.

Syntax:

C/C++:

I32 FNTYPE APS_ptp_all(I32 Axis_ID, I32 Option, F64 Position, F64 Vs, F64 Vm, F64 Ve, F64 Acc, F64 Dec, F64 Sfac, ASYNCALL *Wait);

Visual Basic:

APS_ptp_all(ByVal Axis_ID As Long, ByVal Option As Long, ByVal Position As Double, ByVal Vs As Double, ByVal Vm As Double, ByVal Ve As Double, ByVal Acc As Double, ByVal Dec As Double, ByVal Sfac As Double, Wait As ASYNCALL) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

For PCI(e)-8154/58:

7	6	5	4	3	2	1	0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~11: Reserved for future, set to 0.

Bit12~15: Buffer mode:

0000b(0): Aborting, would start position move, or override old movement with new position and speed while in motion.

00001b(): Buffered. **Note: It is reserved for future.**

F64 Position: A value specifies how many position/distance to move.

F64 Vs: A value specifies the starting velocity.

F64 Vm: A value specifies the maximum velocity.

F64 Ve: A value specifies the ending velocity.

F64 Acc: A value specifies the acceleration.

F64 Dec: A value specifies the deceleration.

F64 Sfac: A value specifies the s factor.

ASYNCALL *Wait: A pointer to ASYNCALL structure. Note: It is reserved for future. Passing it with NULL defines a waiting call. If it is a valid pointer, the call is non-waiting and the functions returns immediately.

For PCI-8254/58 / AMP-204/8C:

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode						Force Abort	Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9: This bit is used to assign profile abort behavior when different direction or insufficient decelerating distance happened.

0: Profile will change smoothly. (Default value)

1: Profile will change immediately.

Bit 10~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting

0001b(1): Buffered

0010b(2): Blending low

0011b(3): Blending previous

0100b(4): Blending next

0101b(5): Blending high

Bit 16~: Reserved for future, set to 0.

F64 Position: A value specifies how many position/distance to move.

F64 Vs: A value specifies the starting velocity.

F64 Vm: A value specifies the maximum velocity.

F64 Ve: A value specifies the ending velocity.

F64 Acc: A value specifies the acceleration.

F64 Dec: A value specifies the deceleration.

F64 SFac: A value specifies the s factor.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

For PCIe-833x, ECAT-4XMO :

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting

0001b(1): Buffered

0010b(2): Blending low

0011b(3): Blending previous

0100b(4): Blending next

0101b(5): Blending high

Bit 16~: Reserved for future, set to 0.

F64 Position: A value specifies how many position/distance to move.

F64 Vs: A value specifies the starting velocity.

F64 Vm: A value specifies the maximum velocity.

F64 Ve: A value specifies the ending velocity.

F64 Acc: A value specifies the acceleration.

F64 Dec: A value specifies the deceleration.

F64 Sfac: A value specifies the s factor.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 opt = 0x1000;    //absolute, aborting mode
ASYNCALL *wait = NULL; //A waiting call

//An absolute move to position(10000) with Vs(10), Vm(100000), Ve(20), Acc/Dec(200000),
Sfac(0.5)
APS_ptp_all( Axis_ID, opt, 10000, 10, 100000, 20, 200000, 200000, 0.5, wait );
```

See also:

APS_relative_move();APS_absolute_move();APS_relative_move_ovrd();APS_absolute_move_ovrd(); APS_ptp_v()

APS_vel	Begin a velocity move

Support Products: PCI(e)-8154/58, PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a velocity move with Vm. No any suffix represents that no any motion profile is necessary to perform a velocity move. Other motion profiles are set in axis parameters. User could refer to axis parameter table for the details.

Syntax:

C/C++:

I32 FNTYPE APS_vel(I32 Axis_ID, I32 Option, F64 Vm, ASYNCALL *Wait);

Visual Basic:

APS_vel(ByVal Axis_ID As Long, ByVal Option As Long, ByVal Vm As Double, Wait As ASYNCALL) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

For PCI(e)-8154/58:

7	6	5	4	3	2	1	0
							0: Positive / 1: Negative
15	14	13	12	11	10	9	8

Bit 0: 0: Positive direction, 1: Negative direction

Bit 1~15: Reserved for future, set to 0.

F64 Vm: A value specifies the maximum velocity.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**Passing it with NULL defines a waiting call. If it is a valid pointer, the call is non-waiting and the functions returns immediately.

For PCI-8254/58 / AMP-204/8C:

7	6	5	4	3	2	1	Bit : 0

							0: Positive / 1: Negative
15	14	13	12	11	10	9	8
						Force Abort	Wait trigger

Bit 0: 0: Positive direction, 1: Negative direction

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9: This bit is used to assign profile abort behavior when different direction happened.

0: Profile will change smoothly. (Default value)

1: Profile will change immediately.

Bit 10~: Reserved for future, set to 0.

F64 Vm: A value specifies the maximum velocity.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

For PCIe-833x, ECAT-4XMO :

7	6	5	4	3	2	1	Bit : 0
							0: Positive / 1: Negative
15	14	13	12	11	10	9	8
							Wait trigger

Bit 0: 0: Positive direction, 1: Negative direction

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~: Reserved for future, set to 0.

F64 Vm: A value specifies the maximum velocity.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 opt = 0; // Positive direction, not wait trigger
ASYNCALL *wait = NULL; //A waiting call
//Execute a velocity move with Vm(10000).
APS_vel( Axis_ID, opt, 10000, wait );
```

See also:

APS_velocity_move()

APS_vel_all	Begin a velocity move with all profile

Support Products: PCI(e)-8154/58, PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a velocity move. The _all suffix represents that all motion profiles, including Position, Vs, Vm, Acc and Sfac, are necessary to perform a velocity move

Syntax:

C/C++:

I32 FNTYPE APS_vel_all(I32 Axis_ID, I32 Option, F64 Vs, F64 Vm, F64 Ve, F64 Acc, F64 Dec, F64 Sfac, ASYNCALL *Wait);

Visual Basic:

APS_vel_all(ByVal Axis_ID As Long, ByVal Option As Long, ByVal Vs As Double, ByVal Vm As Double, ByVal Ve As Double, ByVal Acc As Double, ByVal Dec As Double, ByVal Sfac As Double, Wait As ASYNCALL) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

For PCI(e)-8154/58:

7	6	5	4	3	2	1	0
							0: Positive / 1: Negative
15	14	13	12	11	10	9	8

Bit 0: 0: Positive direction, 1: Negative direction

Bit 1~15: Reserved for future, set to 0.

F64 Vs: A value specifies the starting velocity.

F64 Vm: A value specifies the maximum velocity.

F64 Ve: A value specifies the ending velocity.

F64 Acc: A value specifies the acceleration.

F64 Dec: A value specifies the deceleration.

F64 Sfac: A value specifies the s factor.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.** Passing it with NULL defines a waiting call. If it is a valid pointer, the call is non-waiting and the functions returns immediately.

For PCI-8254/58 / AMP-204/8C:

7	6	5	4	3	2	1	Bit : 0
							0: Positive / 1: Negative
15	14	13	12	11	10	9	8
						Force Abort	Wait trigger

Bit 0: 0: Positive direction, 1: Negative direction

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9: This bit is used to assign profile abort behavior when different direction happened.

0: Profile will change smoothly. (Default value)

1: Profile will change immediately.

Bit 10~: Reserved for future, set to 0.

F64 Vs: A value specifies the starting velocity.

F64 Vm: A value specifies the maximum velocity.

F64 Ve: A value specifies the ending velocity.

F64 Acc: A value specifies the acceleration.

F64 Dec: A value specifies the deceleration.

F64 SFac: A value specifies the s factor.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

For PCIe-833x, ECAT-4XMO :

7	6	5	4	3	2	1	Bit : 0
							0: Positive / 1: Negative
15	14	13	12	11	10	9	8
							Wait trigger

Bit 0: 0: Positive direction, 1: Negative direction

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~: Reserved for future, set to 0.

F64 Vs: A value specifies the starting velocity.

F64 Vm: A value specifies the maximum velocity.

F64 Ve: A value specifies the ending velocity.

F64 Acc: A value specifies the acceleration.

F64 Dec: A value specifies the deceleration.

F64 Sfac: A value specifies the s factor.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 opt = 0; // Positive direction, not wait trigger
ASYNCALL *wait = NULL; //A waiting call

//Execute a velocity move with Vs(10), Vm(100000), Ve(20), Acc/Dec(200000), Sfac(0.5)
APS_vel_all( Axis_ID, opt, 10, 100000, 20, 200000, 200000, 0.5, wait );
```

See also:

APS_velocity_move(); APS_vel()

APS_line	Begin a line interpolation

Support Products: PCI(e)-8154/58 , PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a line interpolation. No any suffix represents that no any motion profile is necessary to perform a line interpolation. Other motion profiles are set in axis parameters. User could refer to axis parameter table for the details.

Syntax:

C/C++:

I32 FNTYPE APS_line(I32 Dimension, I32 *Axis_ID_Array, I32 Option, F64 *PositionArray, F64 *TransPara, ASYNCALL *Wait);

Visual Basic:

APS_line (ByVal Dimension As Long, Axis_ID_Array As Long, ByVal Option As Long, PositionArray As Double, TransPara As Double, Wait As ASYNCALL) As Long

Parameters:

I32 Dimension: A value specifies axes dimension. Range is from 2 to 4.

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

Note: The axes specified in Axis_ID_Array must be of the same card.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

For PCI(e)-8154/58:

7	6	5	4	3	2	1	0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~15: Reserved for future, set to 0.

F64 *PositionArray: A pointer indicates the starting address of position array.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

Note: It is reserved for future.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

For PCI-8254/58 / AMP-204/8C and PCIe-833x, ECAT-4XMO:

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Aborting – stop then go (TransPara_0 as deceleration. [dec >0] , if dec <= 0, kernel takes new coming deceleration.)

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance >= 0.0)

0110b(6): Blending – Residue distance in travel distance's percentage

(TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *PositionArray: A pointer indicates the starting address of position array.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

ASYNCALL *Wait: A pointer to ASYNCALL structure.

Note: It is reserved for future.

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example1:

Below example is for PCI(e)-8154/58

```
I32 opt = 0; //absolute
I32 Dimension = 4;
I32 Master_Axis_ID = 0;
```

```
I32 Axis_ID_Array[4] = {0, 1, 2, 3}; //Axis ID 0 is master axis.
I32 Distance_Array[4] = {10000, 20000, 30000, 40000 };
I32 Max_Linear_Speed = 10000;
I32 Ret;
F64 TransPara = 0;
ASYNCALL *wait = NULL;

APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 100000 ); //Set acceleration
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 100000 ); //Set deceleration

//Execute a line move
Ret = APS_line ( Dimension, Axis_ID_Array, opt, PositionArray, &TransPara, wait );
```

Example2:

Below example is for PCI-8254/58 / AMP-204/8C or PCIe-833x

```
I32 opt = 0x3000; //absolute, not wait trigger, Buffered mode
F64 TransPara = 0; //don't care in buffered mode
ASYNCALL *wait = NULL; //A waiting call

//Execute a line move
APS_line ( Dimension, Axis_ID_Array, opt, PositionArray, &TransPara, wait );
```

See also:

APS_relative_linear_move();APS_absolute_linear_move()

APS_line_v	Begin a line interpolation with Vm profile

Support Products: PCI(e)-8154/58 , PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a line interpolation. The _v suffix represents that only one motion profile, that is Vm, is necessary to perform a line interpolation. Other motion profiles are set in axis parameters. User could refer to axis parameter table for the details.

Syntax:

C/C++:

I32 FNTYPE APS_line_v(I32 Dimension, I32 *Axis_ID_Array, I32 Option, F64 *PositionArray, F64 *TransPara, F64 Vm, ASYNCALL *Wait);

Visual Basic:

APS_line (ByVal Dimension As Long, Axis_ID_Array As Long, ByVal Option As Long, PositionArray As Double, TransPara As Double, ByVal Vm As Double, Wait As ASYNCALL) As Long

Parameters:

For PCI(e)-8154/58:

I32 Dimension: A value specifies axes dimension. Range is from 2 to 4.

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

Note: The axes specified in Axis_ID_Array must be of the same card.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~15: Reserved for future, set to 0.

F64 *PositionArray: A pointer indicates the starting address of position array.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

Note: It is reserved for future.

F64 Vm: A value specifies the maximum velocity.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

For PCI-8254/58 / AMP-204/8C and PCIe-833x, ECAT-4XMO:

I32 Dimension: A value specifies axes dimension. Range is from 2 to 6.

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Aborting – stop then go (TransPara_0 as deceleration. [dec >0] , if dec <= 0, kernel takes new coming deceleration.)

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance >= 0.0)

0110b(6): Blending – Residue distance in travel distance's percentage (TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *PositionArray: A pointer indicates the starting address of position array.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

F64 Vm: A value specifies the maximum velocity.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example1:

Below example is for PCI(e)-8154/58

```
I32 opt = 0; //absolute
I32 Dimension = 4;
I32 Master_Axis_ID = 0;
I32 Axis_ID_Array[4] = {0, 1, 2, 3}; //Axis ID 0 is master axis.
F64 Distance_Array[4] = {10000, 20000, 30000, 40000 };
F64 Max_Linear_Speed = 10000;
I32 Ret;
F64 TransPara = 0;
ASYNCALL *wait = NULL;

APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 100000 ); //Set acceleration
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 100000 ); //Set deceleration

//Execute a line move
Ret = APS_line_v ( Dimension, Axis_ID_Array, opt, PositionArray, &TransPara,
   Max_Linear_Speed, wait );
```

Example2:

Below example is for PCI-8254/58 / AMP-204/8C or PCIe-833x, ECAT-4XMO

```
I32 opt = 0x3000; //absolute, not wait trigger, Buffered mode
F64 TransPara = 0; //don't care in buffered mode
ASYNCALL *wait = NULL; //A waiting call

//Execute a line move with Vm(10000)
APS_line_v( Dimension, Axis_ID_Array, opt, PositionArray, &TransPara, 10000, wait );
```

See also:

APS_relative_linear_move();APS_absolute_linear_move();APS_line()

APS_line_all	Begin a line interpolation with all profile

Support Products: PCI(e)-8154/58, PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a line interpolation. The _all suffix represents that all motion profiles, including Vs, Vm, Ve, Acc, Dec and SFac, are necessary to perform a line interpolation.

Syntax:

C/C++:

I32 FNTYPE APS_line_all(I32 Dimension, I32 *Axis_ID_Array, I32 Option, F64 *PositionArray, F64 *TransPara, F64 Vs, F64 Vm, F64 Ve, F64 Acc,F64 Dec, F64 SFac, ASYNCALL *Wait);

Visual Basic:

APS_line_all(ByVal Dimension As Long, Axis_ID_Array As Long, ByVal Option As Long, PositionArray As Double, TransPara As Double, ByVal Vs As Double, ByVal Vm As Double, ByVal Ve As Double, ByVal Acc As Double, ByVal Dec As Double, ByVal SFac As Double, Wait As ASYNCALL) As Long

Parameters:

For PCI(e)-8154/58:

I32 Dimension: A value specifies axes dimension. Range is from 2 to 4.

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array. **Note: The axes specified in Axis_ID_Array must be of the same card**.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~15: Reserved for future, set to 0.

F64 *PositionArray: A pointer indicates the starting address of position array.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

Note: It is reserved for future.

F64 Vs: A value specifies the starting velocity.

F64 Vm: A value specifies the maximum velocity.

F64 Ve: A value specifies the ending velocity.

F64 Acc: A value specifies the acceleration.

F64 Dec: A value specifies the deceleration.

F64 SFac: A value specifies the s factor.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

For PCI-8254/58 / AMP-204/8C and PCIe-833x, ECAT-4XMO:

I32 Dimension: A value specifies axes dimension. Range is from 2 to 6.

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Aborting – stop then go (TransPara_0 as deceleration. [dec >0] , if dec <= 0, kernel takes new coming deceleration.)

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance >= 0.0)

0110b(6): Blending – Residue distance in travel distance’s percentage (TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *PositionArray: A pointer indicates the starting address of position array.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

F64 Vs: A value specifies the starting velocity.

F64 Vm: A value specifies the maximum velocity.

F64 Ve: A value specifies the ending velocity.

F64 Acc: A value specifies the acceleration.

F64 Dec: A value specifies the deceleration.

F64 SFac: A value specifies the s factor.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example1:

Below example is for PCI(e)-8154/58

```
I32 opt = 0; //absolute
I32 Dimension = 4;
I32 Master_Axis_ID = 0;
I32 Axis_ID_Array[4] = {0, 1, 2, 3}; //Axis ID 0 is master axis.
F64 Distance_Array[4] = {10000, 20000, 30000, 40000 };
I32 Ret;
F64 TransPara = 0;
ASYNCALL *wait = NULL;

APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 100000 ); //Set acceleration
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 100000 ); //Set deceleration

//Execute a line move
//Execute a line move with Vs(10), Vm(100000), Ve(20), Acc/Dec(200000), SFac(0.5)
  APS_line_all( Dimension, Axis_ID_Array, opt, PositionArray, &TransPara, 10, 100000, 20,
  200000, 200000, 0.5, wait );
```

Example2:

Below example is for PCI-8254/58 / AMP-204/8C or PCIe-833x, ECAT-4XMO

```
I32 opt = 0x3000; //absolute, not wait trigger, Buffered mode
F64 TransPara = 0; //don't care in buffered mode
ASYNCALL *wait = NULL; //A waiting call

//Execute a line move with Vs(10), Vm(100000), Ve(20), Acc/Dec(200000), SFac(0.5)
```

APS_line_all(Dimension, Axis_ID_Array, opt, PositionArray, &TransPara, 10, 100000, 20, 200000, 200000, 0.5, wait);

See also:

APS_relative_linear_move(); APS_absolute_linear_move();APS_line()

APS_arc2_ca	Begin an Arc2 move of angle type

Support Products: PCI(e)-8154/58 , PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a 2D arc interpolation of angle type, named _arc2_ca. It follows with center position and angle. Current position and center position arguments would decide radius of arc.No any suffix represents that no any motion profile is necessary to perform a 2D arc interpolation. Other motion profiles are set in axis parameters. User could refer to axis parameter table for the details.

Syntax:

C/C++:

I32 FNTYPE APS_arc2_ca(I32 *Axis_ID_Array, I32 Option, F64 *CenterArray, F64 Angle, F64 *TransPara, ASYNCALL *Wait);

Visual Basic:

APS_arc2_ca(Axis_ID_Array As Long, ByVal Option As Long, CenterArray As Double, ByVal Angle As Double, TransPara As Double, Wait As ASYNCALL) As Long

Parameters:

For PCI(e)-8154/58:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

Note: The axes specified in Axis_ID_Array must be of the same card.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~15: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 Angle: A value specifies the angle. Unit in radian. Range is -2*PI ~ 2*PI. Positive value is counterclockwise, negative value is clockwise.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

Note: It is reserved for future.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

For PCI-8254/58 / AMP-204/8C and PCIe-833x, ECAT-4XMO:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Reserved. **Note: if setting to mode 2, it will return error code.**

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance **>= 0.0**)

0110b(6): Blending – Residue distance in travel distance's percentage (TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 Angle: A value specifies the angle. Unit in radian.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 opt = 0; //absolute
I32 Master_Axis_ID = 0;
I32 Axis_ID_Array[2] = {0, 1}; //Axis ID 0 is master axis.
F64 Center_Pos_Array[2] = {100000, 0};
F64 Angle = -180 * (2PI / 360);    // clockwise 180 degree.
I32 Ret;
F64 TransPara = 0;
ASYNCALL *wait = NULL;

APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 100000 ); //Set acceleration
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 100000 ); //Set deceleration

//Execute a arc move
APS_arc2_ca (Axis_ID_Array, opt, Center_Pos_Array, Angle, &TransPara, wait );
```

See also:

APS_relative_arc_move(); APS_absolute_arc_move()

APS_arc2_ca_v	Begin an Arc2 move of angle type with Vm profile

Support Products: PCI(e)-8154/58 , PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a 2D arc interpolation of angle type, named _arc2_ca. It follows with center position and angle. Current position and center position arguments would decide radius of arc.The _v suffix represents that only one motion profile, that is Vm, is necessary to perform a 2D arc interpolation.Other motion profiles are set in axis parameters. User could refer to axis parameter table for the details.

Syntax:

C/C++:

I32 FNTYPE APS_arc2_ca_v(I32 *Axis_ID_Array, I32 Option, F64 *CenterArray, F64 Angle, F64 *TransPara, F64 Vm, ASYNCALL *Wait);

Visual Basic:

APS_arc2_ca_v(Axis_ID_Array As Long, ByVal Option As Long, CenterArray As Double, ByVal Angle As Double, TransPara As Double, ByVal Vm As Double, Wait As ASYNCALL) As Long

Parameters:

For PCI(e)-8154/58:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

Note: The axes specified in Axis_ID_Array must be of the same card.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~15: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 Angle: A value specifies the angle. Unit in radian. Range is -2*PI ~ 2*PI. Positive value is counterclockwise, negative value is clockwise.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

Note: It is reserved for future.

F64 Vm: A value specifies the maximum velocity.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

For PCI-8254/58 / AMP-204/8C and PCIe-833x, ECAT-4XMO:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Reserved. **Note: if setting to mode 2, it will return error code.**

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance >= 0.0)

0110b(6): Blending – Residue distance in travel distance's percentage (TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 Angle: A value specifies the angle. Unit in radian.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

F64 Vm: A value specifies the maximum velocity.

ASYNCALL *Wait: A pointer to ASYNCALL structure. Note: It is reserved for future.

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 opt = 0; //absolute
I32 Master_Axis_ID = 0;
I32 Axis_ID_Array[2] = {0, 1}; //Axis ID 0 is master axis.
F64 Center_Pos_Array[2] = {100000, 0};
F64 Angle = -180 * (2*PI / 360);    // clockwise 180 degree.
F64 Speed = 10000.0;
I32 Ret;
F64 TransPara = 0;
ASYNCALL *wait = NULL;

APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 100000 ); //Set acceleration
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 100000 ); //Set deceleration

//Execute a arc move
APS_arc2_ca_v (Axis_ID_Array, opt, Center_Pos_Array, Angle, &TransPara, Speed ,wait );
```

See also:

APS_relative_arc_move();APS_absolute_arc_move();APS_arc2_ca()

APS_arc2_ca_all	Begin an Arc2 move of angle type with all profile

Support Products: PCI(e)-8154/58 , PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a 2D arc interpolation of angle type, named _arc2_ca. It follows with center position and angle. Current position and center position arguments would decide radius of arc.The _all suffix represents that all motion profiles, including Vs, Vm, Ve, Acc, Dec and SFac, are necessary to perform a 2D arc interpolation.

Syntax:

C/C++:

I32 FNTYPE APS_arc2_ca_all(I32 *Axis_ID_Array, I32 Option, F64 *CenterArray, F64 Angle, F64 *TransPara, F64 Vs, F64 Vm, F64 Ve, F64 Acc,F64 Dec, F64 SFac, ASYNCALL *Wait);

Visual Basic:

APS_arc2_ca_all(Axis_ID_Array As Long, ByVal Option As Long, CenterArray As Double, ByVal Angle As Double, TransPara As Double, ByVal Vs As Double, ByVal Vm As Double, ByVal Ve As Double, ByVal Acc As Double, ByVal Dec As Double, ByVal SFac As Double, Wait As ASYNCALL) As Long

Parameters:

For PCI(e)-8154/58:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

Note: The axes specified in Axis_ID_Array must be of the same card.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~15: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 Angle: A value specifies the angle. Unit in radian. Range is -2*PI ~ 2*PI. Positive value is counterclockwise, negative value is clockwise.

F64 *TransPara: A pointer indicates the starting address of transfer parameters. **Note: It is reserved for future**.

F64 Vs: A value specifies the starting velocity.

F64 Vm: A value specifies the maximum velocity.

F64 Ve: A value specifies the ending velocity.

F64 Acc: A value specifies the acceleration.

F64 Dec: A value specifies the deceleration.

F64 SFac: A value specifies the s factor.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

For PCI-8254/58 / AMP-204/8C and PCIe-833x, ECAT-4XMO:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Reserved. **Note: if setting to mode 2, it will return error code.**

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance **>= 0.0**)

0110b(6): Blending – Residue distance in travel distance's percentage (TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 Angle: A value specifies the angle. Unit in radian.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

F64 Vs: A value specifies the starting velocity.

F64 Vm: A value specifies the maximum velocity.

F64 Ve: A value specifies the ending velocity.

F64 Acc: A value specifies the acceleration.

F64 Dec: A value specifies the deceleration.

F64 SFac: A value specifies the s factor.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 opt = 0; //absolute
I32 Master_Axis_ID = 0;
I32 Axis_ID_Array[2] = {0, 1}; //Axis ID 0 is master axis.
F64 Center_Pos_Array[2] = {100000, 0};
F64 Angle = -180 * (2*PI / 360);    // clockwise 180 degree.
I32 Ret;
F64 TransPara = 0;
ASYNCALL *wait = NULL;

APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 100000 ); //Set acceleration
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 100000 ); //Set deceleration

//Execute a arc move with Vs(10), Vm(100000), Ve(20), Acc/Dec(200000), SFac(0.5)
APS_arc2_ca_all (Axis_ID_Array, opt, Center_Pos_Array, Angle, &TransPara, 10, 100000, 20,
200000,
200000, 0.5, wait );
```

See also:

APS_relative_arc_move();APS_absolute_arc_move();APS_arc2_ca()

APS_arc2_ce	Begin an Arc2 move of end position

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO, PCI(e)-8154/58

Descriptions:

This function is used to execute a 2D arc interpolation of end position type, named _arc2_ce. It follows with center position, end position and Dir. No any suffix represents that no any motion profile is necessary to perform a 2D arc interpolation. Other motion profiles are set in axis parameters. User could refer to axis parameter table for the details.

Syntax:

C/C++:

I32 FNTYPE APS_arc2_ce(I32 *Axis_ID_Array, I32 Option, F64 *CenterArray, F64 *EndArray, I16 Dir, F64 *TransPara, ASYNCALL *Wait);

Visual Basic:

I32 FNTYPE APS_arc2_ce(Axis_ID_Array As Long, ByVal Option As Long, CenterArray As Double, EndArray As Double, ByVal Dir As Short, TransPara As Double, Wait As ASYNCALL) As Long

Parameters:

For PCI(e)-8154/58:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

Note: The axes specified in Axis_ID_Array must be of the same card.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~15: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *EndArray: A pointer indicates the starting address of end array.

I16 Dir: A value specifies the rotate direction. If dir set 1 means ***Dir***=1 and rotate in positive direction,when dir set <=0 means ***Dir***=-1 and rotate in negative direction. The total rotate *angle = theta + **Dir** x 2PI*, where *theta* is the angle of two vectors: center to start and center to end.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

Note: It is reserved for future.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

For PCI-8254/58 / AMP-204/8C and PCIe-833x, ECAT-4XMO:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Reserved. **Note: if setting to mode 2, it will return error code.**

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance **>= 0.0**)

0110b(6): Blending – Residue distance in travel distance's percentage(TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *EndArray: A pointer indicates the starting address of end array.

I16 Dir: A value specifies the rotate direction. If dir set 0 means rotate in positive direction, dir = -1 rotate in negative direction. The total rotate *angle = theta + **Dir** x 2PI*, where *theta* is the angle of two vectors: center to start and center to end.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 opt = 0; //absolute
I32 Dir  = 1; // Counter clockwise
I32 Master_Axis_ID = 0;
I32 Axis_ID_Array[2] = {0, 1}; //Axis ID 0 is master axis.
F64 Center_Pos_Array[2] = {0, 0};
//Start point set(100000,0)
F64 End_Pos_Array [2] = {0, 100000};
I32 Ret;
F64 TransPara = 0;
ASYNCALL *wait = NULL;

APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 100000 ); //Set acceleration
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 100000 ); //Set deceleration
//Execute a arc move
APS_arc2_ce (Axis_ID_Array, opt, Center_Pos_Array, End_Pos_Array, Dir  , &TransPara,
   wait );
```

See also:

APS_arc2_ce_v	Begin an Arc2 move of end position with Vm profile

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO PCI(e)-8154/58

Descriptions:

This function is used to execute a 2D arc interpolation of end position type, named _arc2_ce. It follows with center position, end position and Dir. The _v suffix represents that only one motion profile, that is Vm, is necessary to perform a 2D arc interpolation. Other motion profiles are set in axis parameters. User could refer to axis parameter table for the details.

Syntax:

C/C++:

I32 FNTYPE APS_arc2_ce_v(I32 *Axis_ID_Array, I32 Option, F64 *CenterArray, F64 *EndArray, I16 Dir, F64 *TransPara, F64 Vm, ASYNCALL *Wait);

Visual Basic:

APS_arc2_ce_v(Axis_ID_Array As Long, ByVal Option As Long, CenterArray As Double, EndArray As Double, ByVal Dir As Short, TransPara As Double, ByVal Vm As Double, Wait As ASYNCALL) As Long

Parameters:

For PCI(e)-8154/58:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

Note: The axes specified in Axis_ID_Array must be of the same card.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~15: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *EndArray: A pointer indicates the starting address of end array.

I16 Dir: A value specifies the rotate direction. If dir set 1 means ***Dir***=1 and rotate in positive direction,when dir set <=0 means ***Dir***=-1 and rotate in negative direction. The total rotate *angle = theta + **Dir** x 2PI*, where *theta* is the angle of two vectors: center to start and center to end.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

Note: It is reserved for future.

F64 Vm: A value specifies the maximum velocity.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

For PCI-8254/58 / AMP-204/8C and PCIe-833x, ECAT-4XMO:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Reserved. **Note: if setting to mode 2, it will return error code.**

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance **>= 0.0**)

0110b(6): Blending – Residue distance in travel distance's percentage(TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *EndArray: A pointer indicates the starting address of end array.

I16 Dir: A value specifies the rotate direction. If dir set 0 means rotate in positive direction, dir = -1 rotate in negative direction. The total rotate *angle = theta + **Dir** x 2PI*, where *theta* is the angle of two vectors: center to start and center to end.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

F64 Vm: A value specifies the maximum velocity.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 opt = 0; //absolute
I32 Dir  = 1; // Counter clockwise
I32 Master_Axis_ID = 0;
I32 Axis_ID_Array[2] = {0, 1}; //Axis ID 0 is master axis.
F64 Center_Pos_Array[2] = {0, 0};
//Start point set(100000,0)
F64 End_Pos_Array [2] = {0, 100000};
F64 Speed = 10000.0;
I32 Ret;
F64 TransPara = 0;
ASYNCALL *wait = NULL;

APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 100000 ); //Set acceleration
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 100000 ); //Set deceleration
//Execute a arc move
APS_arc2_ce_v (Axis_ID_Array, opt, Center_Pos_Array, Angle, &TransPara, Speed ,wait );
```

See also:

APS_arc2_ce_all	Begin an Arc2 move of end position with all profile

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO PCI(e)-8154/58

Descriptions:

This function is used to execute a 2D arc interpolation of end position type, named _arc2_ce. It follows with center position, end position and Dir. The _all suffix represents that all motion profiles, including Vs, Vm, Ve, Acc, Dec and SFac, are necessary to perform a 2D arc interpolation.

Syntax:

C/C++:

I32 FNTYPE APS_arc2_ce_all(I32 *Axis_ID_Array, I32 Option, F64 *CenterArray, F64 *EndArray, I16 Dir, F64 *TransPara, F64 Vs, F64 Vm, F64 Ve, F64 Acc,F64 Dec, F64 SFac, ASYNCALL *Wait);

Visual Basic:

APS_arc2_ce_all(Axis_ID_Array As Long, ByVal Option As Long, CenterArray As Double, EndArray As Double, ByVal Dir As Short, TransPara As Double, ByVal Vs As Double, ByVal Vm As Double, ByVal Ve As Double, ByVal Acc As Double, ByVal Dec As Double, ByVal SFac As Double, Wait As ASYNCALL) As Long

Parameters:

For PCI(e)-8154/58:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

Note: The axes specified in Axis_ID_Array must be of the same card.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~15: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *EndArray: A pointer indicates the starting address of end array.

I16 Dir: A value specifies the rotate direction. If dir set 1 means ***Dir***=1 and rotate in positive direction,when dir set <=0 means ***Dir***=-1 and rotate in negative direction. The total rotate *angle = theta + **Dir** x 2PI*, where *theta* is the angle of two vectors: center to start and center to end.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

Note: It is reserved for future.

F64 Vs: A value specifies the starting velocity.

F64 Vm: A value specifies the maximum velocity.

F64 Ve: A value specifies the ending velocity.

F64 Acc: A value specifies the acceleration.

F64 Dec: A value specifies the deceleration.

F64 SFac: A value specifies the s factor.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

For PCI-8254/58 / AMP-204/8C and PCIe-833x, ECAT-4XMO:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Reserved. **Note: if setting to mode 2, it will return error code.**

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance **>= 0.0**)

0110b(6): Blending – Residue distance in travel distance's percentage(TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *EndArray: A pointer indicates the starting address of end array.

I16 Dir: A value specifies the rotate direction. If dir set 0 means rotate in positive direction, dir = -1 rotate in negative direction. The total rotate *angle = theta + **Dir** x 2PI*, where *theta* is the angle of two vectors: center to start and center to end.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

F64 Vs: A value specifies the starting velocity.

F64 Vm: A value specifies the maximum velocity.

F64 Ve: A value specifies the ending velocity.

F64 Acc: A value specifies the acceleration.

F64 Dec: A value specifies the deceleration.

F64 SFac: A value specifies the s factor.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 opt = 0; //absolute
I32 Dir  = 1; // Counter clockwise
I32 Master_Axis_ID = 0;
I32 Axis_ID_Array[2] = {0, 1}; //Axis ID 0 is master axis.
F64 Center_Pos_Array[2] = {0, 0};
//Start point set(100000,0)
F64 End_Pos_Array [2] = {0, 100000};
I32 Ret;
F64 TransPara = 0;
ASYNCALL *wait = NULL;

APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 100000 ); //Set acceleration
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 100000 ); //Set deceleration
//Execute a arc move with Vs(10), Vm(100000), Ve(20), Acc/Dec(200000), SFac(0.5)
APS_arc2_ce_all (Axis_ID_Array, opt, Center_Pos_Array, Angle, &TransPara, 10, 100000, 20,
200000, 200000, 0.5, wait );
```

See also:

APS_arc3_ca	Begin an Arc3 move of angle type

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a 3D arc interpolation of angle type, named _arc3_ca. It follows with angle, center position and normal vector. No any suffix represents that no any motion profile is necessary to perform a 3D arc interpolation. Other motion profiles are set in axis parameters. User could refer to axis parameter table for the details.

Syntax:

C/C++:

I32 FNTYPE APS_arc3_ca(I32 *Axis_ID_Array, I32 Option, F64 *CenterArray, F64 *NormalArray, F64 Angle, F64 *TransPara, ASYNCALL *Wait);

Visual Basic:

APS_arc3_ca(Axis_ID_Array As Long, ByVal Option As Long, CenterArray As Double, NormalArray As Double, ByVal Angle As Double, TransPara As Double, Wait As ASYNCALL) As Long

Parameters:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Reserved. **Note: if setting to mode 2, it will return error code.**

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance >= 0.0)

0110b(6): Blending – Residue distance in travel distance's percentage

(TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 * NormalArray: A pointer indicates the starting address of normal vector array.

F64 Angle: A value specifies the angle. Unit in radian.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_arc3_ca_v	Begin an Arc3 move of angle type with Vm profile

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a 3D arc interpolation of angle type, named _arc3_ca. It follows with angle, center position and normal vector. The _v suffix represents that only one motion profile, that is Vm, is necessary to perform a 3D arc interpolation. Other motion profiles are set in axis parameters. User could refer to axis parameter table for the details.

Syntax:

C/C++:

I32 FNTYPE APS_arc3_ca_v(I32 *Axis_ID_Array, I32 Option, F64 *CenterArray, F64 *NormalArray, F64 Angle, F64 *TransPara, F64 Vm, ASYNCALL *Wait);

Visual Basic:

APS_arc3_ca_v(Axis_ID_Array As Long, ByVal Option As Long, CenterArray As Double, NormalArray As Double, ByVal Angle As Double, TransPara As Double, ByVal Vm As Double, Wait As ASYNCALL) As Long

Parameters:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Reserved. **Note: if setting to mode 2, it will return error code.**

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance >= 0.0)

0110b(6): Blending – Residue distance in travel distance's percentage

(TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 * NormalArray: A pointer indicates the starting address of normal vector array.

F64 Angle: A value specifies the angle. Unit in radian.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

F64 Vm: A value specifies the maximum velocity.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_arc3_ca_all	Begin an Arc3 move of angle type with all profile

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a 3D arc interpolation of angle type, named _arc3_ca. It follows with angle, center position and normal vector. The _all suffix represents that all motion profiles, including Vs, Vm, Ve, Acc, Dec and SFac, are necessary to perform a 3D arc interpolation.

Syntax:

C/C++:

I32 FNTYPE APS_arc3_ca_all(I32 *Axis_ID_Array, I32 Option, F64 *CenterArray, F64 *NormalArray, F64 Angle, F64 *TransPara, F64 Vs, F64 Vm, F64 Ve, F64 Acc,F64 Dec, F64 SFac, ASYNCALL *Wait);

Visual Basic:

APS_arc3_ca_all(Axis_ID_Array As Long, ByVal Option As Long, CenterArray As Double, NormalArray As Double, ByVal Angle As Double, TransPara As Double, ByVal Vs As Double, ByVal Vm As Double, ByVal Ve As Double, ByVal Acc As Double, ByVal Dec As Double, ByVal SFac As Double, Wait As ASYNCALL) As Long

Parameters:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Reserved. **Note: if setting to mode 2, it will return error code.**

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance **>= 0.0**)

0110b(6): Blending – Residue distance in travel distance's percentage

(TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *NormalArray: A pointer indicates the starting address of normal vector array.

F64 Angle: A value specifies the angle. Unit in radian.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

F64 Vs: A value specifies the starting velocity.

F64 Vm: A value specifies the maximum velocity.

F64 Ve: A value specifies the ending velocity.

F64 Acc: A value specifies the acceleration.

F64 Dec: A value specifies the deceleration.

F64 SFac: A value specifies the s factor.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_arc3_ce	Begin an Arc3 move of end position

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a 3D arc interpolation of end position type, named _arc3_ce. It follows with center position, end position and Dir. No any suffix represents that no any motion profile is necessary to perform a 3D arc interpolation. Other motion profiles are set in axis parameters. User could refer to axis parameter table for the details.

Syntax:

C/C++:

I32 FNTYPE APS_arc3_ce(I32 *Axis_ID_Array, I32 Option, F64 *CenterArray, F64 *EndArray, I16 Dir, F64 *TransPara, ASYNCALL *Wait);

Visual Basic:

I32 FNTYPE APS_arc3_ce(Axis_ID_Array As Long, ByVal Option As Long, CenterArray As Double, EndArray As Double, ByVal Dir As Short, TransPara As Double, Wait As ASYNCALL) As Long

Parameters:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Reserved. **Note: if setting to mode 2, it will return error code.**

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance **>= 0.0**)

0110b(6): Blending – Residue distance in travel distance's

percentage(TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *EndArray: A pointer indicates the starting address of end array.

I16 Dir: A value specifies the rotate direction. If dir set 0 means rotate in positive direction, dir = -1 rotate in negative direction. The total rotate *angle = theta + **Dir** x 2PI*, where *theta* is the angle of two vectors: center to start and center to end.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_arc3_ce_v	Begin an Arc3 move of end position with Vm profile

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a 3D arc interpolation of end position type, named _arc3_ce. It follows with center position, end position and Dir. The _v suffix represents that only one motion profile, that is Vm, is necessary to perform a 3D arc interpolation. Other motion profiles are set in axis parameters. User could refer to axis parameter table for the details.

Syntax:

C/C++:

I32 FNTYPE APS_arc3_ce_v(I32 *Axis_ID_Array, I32 Option, F64 *CenterArray, F64 *EndArray, I16 Dir, F64 *TransPara, F64 Vm, ASYNCALL *Wait);

Visual Basic:

APS_arc3_ce_v(Axis_ID_Array As Long, ByVal Option As Long, CenterArray As Double, EndArray As Double, ByVal Dir As Short, TransPara As Double, ByVal Vm As Double, Wait As ASYNCALL) As Long

Parameters:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Reserved. **Note: if setting to mode 2, it will return error code.**

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance **>= 0.0**)

0110b(6): Blending – Residue distance in travel distance's

percentage(TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *EndArray: A pointer indicates the starting address of end array.

I16 Dir: A value specifies the rotate direction. If dir set 0 means rotate in positive direction, dir = -1 rotate in negative direction. The total rotate *angle = theta + **Dir** x 2PI*, where *theta* is the angle of two vectors: center to start and center to end.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

F64 Vm: A value specifies the maximum velocity.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_arc3_ce_all	Begin an Arc3 move of end position with all profile

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a 3D arc interpolation of end position type, named _arc3_ce. It follows with center position, end position and Dir. The _all suffix represents that all motion profiles, including Vs, Vm, Ve, Acc, Dec and SFac, are necessary to perform a 3D arc interpolation.

Syntax:

C/C++:

I32 FNTYPE APS_arc3_ce_all(I32 *Axis_ID_Array, I32 Option, F64 *CenterArray, F64 *EndArray, I16 Dir, F64 *TransPara, F64 Vs, F64 Vm, F64 Ve, F64 Acc,F64 Dec, F64 SFac, ASYNCALL *Wait);

Visual Basic:

APS_arc3_ce_all(Axis_ID_Array As Long, ByVal Option As Long, CenterArray As Double, EndArray As Double, ByVal Dir As Short, TransPara As Double, ByVal Vs As Double, ByVal Vm As Double, ByVal Ve As Double, ByVal Acc As Double, ByVal Dec As Double, ByVal SFac As Double, Wait As ASYNCALL) As Long

Parameters:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Reserved. **Note: if setting to mode 2, it will return error code.**

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance **>= 0.0**)

0110b(6): Blending – Residue distance in travel distance's

percentage(TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *EndArray: A pointer indicates the starting address of end array.

I16 Dir: A value specifies the rotate direction. If dir set 0 means rotate in positive direction, dir = -1 rotate in negative direction. The total rotate *angle = theta + **Dir** x 2PI*, where *theta* is the angle of two vectors: center to start and center to end.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

F64 Vs: A value specifies the starting velocity.

F64 Vm: A value specifies the maximum velocity.

F64 Ve: A value specifies the ending velocity.

F64 Acc: A value specifies the acceleration.

F64 Dec: A value specifies the deceleration.

F64 SFac: A value specifies the s factor.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_spiral_ca	Begin a 3D spiral-helix move of angle type

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a 3D spiral-helix interpolation of angle type, named _spiral_ca. It follows with angle, center position, normal vector, DeltaH and FinalR. No any suffix represents that no any motion profile is necessary to perform a 3D spiral-helix interpolation. Other motion profiles are set in axis parameters. User could refer to axis parameter table for the details.

Syntax:

C/C++:

I32 FNTYPE APS_spiral_ca(I32 *Axis_ID_Array, I32 Option, F64 *CenterArray, F64 *NormalArray, F64 Angle, F64 DeltaH, F64 FinalR, F64 *TransPara, ASYNCALL *Wait);

Visual Basic:

APS_ spiral_ca(Axis_ID_Array As Long, ByVal Option As Long, CenterArray As Double, NormalArray As Double, ByVal Angle As Double, ByVal DeltaH As Double, ByVal FinalR As Double, TransPara As Double, Wait As ASYNCALL) As Long

Parameters:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Reserved. **Note: if setting to mode 2, it will return error code.**

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance >= 0.0)

0110b(6): Blending – Residue distance in travel distance's percentage

(TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *NormalArray: A pointer indicates the starting address of normal vector array.

F64 Angle: A value specifies the angle. Unit in radian.

F64 DeltaH: A value specifies the height.

F64 FinalR: A value specifies the distant from end position to normal vector.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_spiral_ca_v	Begin a 3D spiral-helix move of angle type with Vm profile

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x

Descriptions:

This function is used to execute a 3D spiral-helix interpolation of angle type, named _spiral_ca. It follows with angle, center position, normal vector, DeltaH and FinalR. The _v suffix represents that only one motion profile, that is Vm, is necessary to perform a 3D spiral-helix interpolation. Other motion profiles are set in axis parameters. User could refer to axis parameter table for the details.

Syntax:

C/C++:

I32 FNTYPE APS_spiral_ca_v(I32 *Axis_ID_Array, I32 Option, F64 *CenterArray, F64 *NormalArray, F64 Angle, F64 DeltaH, F64 FinalR, F64 *TransPara, F64 Vm, ASYNCALL *Wait);

Visual Basic:

APS_spiral_ca_v(Axis_ID_Array As Long, ByVal Option As Long, CenterArray As Double, NormalArray As Double, ByVal Angle As Double, ByVal DeltaH As Double, ByVal FinalR As Double, TransPara As Double, ByVal Vm As Double, Wait As ASYNCALL) As Long

Parameters:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Reserved. **Note: if setting to mode 2, it will return error code.**

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance >= 0.0)

0110b(6): Blending – Residue distance in travel distance's percentage (TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *NormalArray: A pointer indicates the starting address of normal vector array.

F64 Angle: A value specifies the angle. Unit in radian.

F64 DeltaH: A value specifies the height.

F64 FinalR: A value specifies the distant from end position to normal vector.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

F64 Vm: A value specifies the maximum velocity.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_spiral_ca_all	Begin a 3D spiral-helix move of angle type with all profile

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a 3D spiral-helix interpolation of angle type, named _spiral_ca. It follows with angle, center position, normal vector, DeltaH and FinalR. The _all suffix represents that all motion profiles, including Vs, Vm, Ve, Acc, Dec and SFac, are necessary to perform a 3D spiral-helix interpolation.

Syntax:

C/C++:

I32 FNTYPE APS_spiral_ca_all(I32 *Axis_ID_Array, I32 Option, F64 *CenterArray, F64 *NormalArray, F64 Angle, F64 DeltaH, F64 FinalR, F64 *TransPara, F64 Vs, F64 Vm, F64 Ve, F64 Acc,F64 Dec, F64 SFac, ASYNCALL *Wait);

Visual Basic:

APS_spiral_ca_all(Axis_ID_Array As Long, ByVal Option As Long, CenterArray As Double, NormalArray As Double, ByVal Angle As Double, F64 DeltaH, F64 FinalR, TransPara As Double, ByVal Vs As Double, ByVal Vm As Double, ByVal Ve As Double, ByVal Acc As Double, ByVal Dec As Double, ByVal SFac As Double, Wait As ASYNCALL) As Long

Parameters:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Reserved. **Note: if setting to mode 2, it will return error code.**

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance **>= 0.0**)

0110b(6): Blending – Residue distance in travel distance's percentage (TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *NormalArray: A pointer indicates the starting address of normal vector array.

F64 Angle: A value specifies the angle. Unit in radian.

F64 DeltaH: A value specifies the height.

F64 FinalR: A value specifies the distant from end position to normal vector.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

F64 Vs: A value specifies the starting velocity.

F64 Vm: A value specifies the maximum velocity.

F64 Ve: A value specifies the ending velocity.

F64 Acc: A value specifies the acceleration.

F64 Dec: A value specifies the deceleration.

F64 SFac: A value specifies the s factor.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_spiral_ce	Begin a 3D spiral-helix move of end position

Support Products: PCI(e)-8154/58 , PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a 3D spiral-helix interpolation of end position type, named _spiral_ce. It follows with center position, normal vector, end position and Dir. No any suffix represents that no any motion profile is necessary to perform a 3D spiral-helix interpolation. Other motion profiles are set in axis parameters. User could refer to axis parameter table for the details.

Base on PCI(e)-8154/58, this function support 1 fixed radius circular interpolation and synchronized linear travel in normal axis. The rotate angle is the theta of two vectors: center to start and center to end. Unlike PCIe–8338 and PCI-8254/58 with soft motion base, the circular interpolation angle range only has -360◦to 360◦. That means spiral curve only has 1 pitch, just like helical curve.

Note : Due to 8154/58 limit, 4th / 8th axis operation will be a dummy motion and it can't be used for any other puspose. This axis need to be set servo-off. If not, it will return ERR_InServoOnState(-48)

e.g.

PCI(e)-8154, choose {0,1,2} be APS_spiral_ce_v(), axis 3 operation is always a dummy motion and it cannot be added into *Axis_ID_Array.

PCI(e)-8158, choose {4,5,6} be APS_spiral_ce_v(), axis 7 operation is always a dummy motion and it cannot be added into *Axis_ID_Array.

Syntax:

C/C++:

I32 FNTYPE APS_spiral_ce(I32 *Axis_ID_Array, I32 Option, F64 *CenterArray, F64 *NormalArray, F64 *EndArray, I16 Dir, F64 *TransPara, ASYNCALL *Wait);

Visual Basic:

I32 FNTYPE APS_spiral_ce(Axis_ID_Array As Long, ByVal Option As Long, CenterArray As Double, NormalArray As Double, EndArray As Double, ByVal Dir As Short, TransPara As Double, Wait As ASYNCALL) As Long

Parameters:

For PCI(e)-8154/58:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

Note: The axes specified in Axis_ID_Array must be of the same card.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~15: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *NormalArray: A pointer indicates the starting address of the normal vector array.

F64 *EndArray: A pointer indicates the starting address of end array.

I16 Dir: A value specifies the rotate direction. If Dir >= 0 means rotate in positive direction(counterclockwise), Dir <= -1 (clockwise) rotate in negative direction.

F64 *TransPara: A pointer indicates the starting address of transfer parameters. **Note: It is reserved for future**.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

For PCI-8254/58 / AMP-204/8C and PCIe-833x, ECAT-4XMO:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Reserved. **Note: if setting to mode 2, it will return error code.**

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance **>= 0.0**)

0110b(6): Blending – Residue distance in travel distance's

percentage(TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *NormalArray: A pointer indicates the starting address of the normal vector array.

F64 *EndArray: A pointer indicates the starting address of end array.

I16 Dir: A value specifies the rotate direction. If dir set 0 means rotate in positive direction, dir = -1 rotate in negative direction. The total rotate *angle = theta + **Dir** x 2PI*, where *theta* is the angle of two vectors: center to start and center to end.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for PCI(e)-8154/58

```
I32 opt = 1; //relative mode
I32 Axis_ID_Array[3] = {0, 1, 2};
F64 NormalArray[3] = { 0, 1, 0 }; //      choose axis 1 is vertical axis
F64 Center_Pos_Array[2] = {10000, 0, 20000};
F64 End_Pos_array[2] = {20000, 20000, 40000};
// height = End_Pos_array[1] - Center_Pos_Array[1]
I16 Dir = 1;            //  rotate counterclockwise
F64 TransPara = 0;
ASYNCALL *wait = NULL;

APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 100000 ); //Set acceleration
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 100000 ); //Set deceleration

APS_spiral_ce (
    Axis_ID_Array                // I32 *Axis_ID_Array
```

```
        , opt                   // I32 Option
        , Center_Pos_Array      // F64 * CenterArray
       , NormalArray            // F64 * NormalArray
       , End_Pos_array          // F64 * Enday
       , Dir                    // I16 Dir
       , &TransPara             // reserved
       , wait );                // reserved
```

Description:

Initial:

Start point: 0, 0, 0

Input:

Option: 1(Relative)

Center point (relative to start point): 10000, 0, 20000

Normal vector: 0, 1, 0 (to positive y-axis)

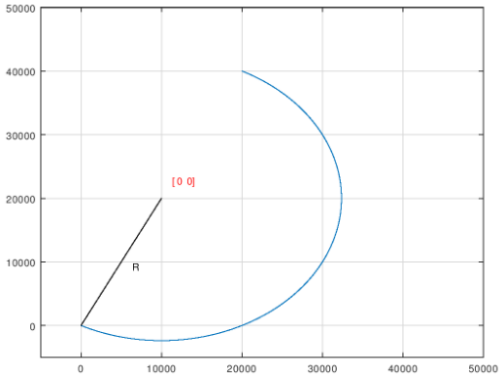
End point: 20000, 20000, 40000

Output:

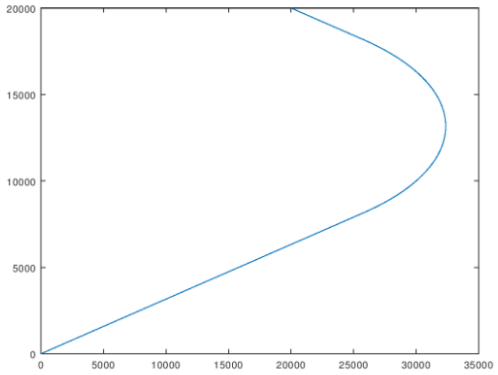
Theta: 180◦, counterclockwise

R: sqrt(10000*10000 + 20000*20000)

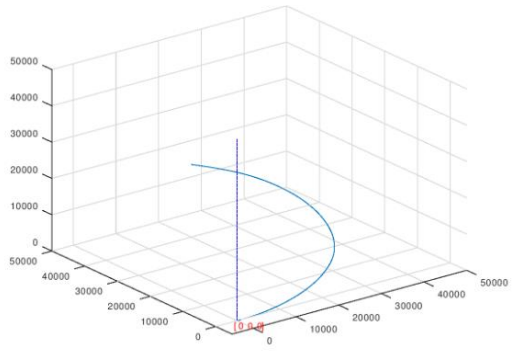
Height = 20000



XZ plane



XY plane



3D view

See also:

APS_absolute_helical_move();APS_relative_helical_move()

APS_spiral_ce_v	Begin a 3D spiral-helix move of end position with Vm profile

Support Products: PCI(e)-8154/58 , PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a 3D spiral-helix interpolation of end position type, named _spiral_ce. It follows with center position, normal vector, end position and Dir. The _v suffix represents that only one motion profile, that is Vm, is necessary to perform a 3D spiral-helix interpolation. Other motion profiles are set in axis parameters. User could refer to axis parameter table for the details.

Base on PCI(e)-8154/58, this function support 1 fixed radius circular interpolation and synchronized linear travel in normal axis. The rotate angle is the theta of two vectors: center to start and center to end. Unlike PCIe–8338 and PCI-8254/58 with soft motion base, the circular interpolation angle range only has -360◦to 360◦. That means spiral curve only has 1 pitch, just like helical curve.

Note : Due to 8154/58 limit, 4th / 8th axis operation will be a dummy motion and it can't be used for any other puspose. This axis need to be set servo-off. If not, it will return ERR_InServoOnState(-48)

e.g.

PCI(e)-8154, choose {0,1,2} be APS_spiral_ce_v(), axis 3 operation is always a dummy motion and it cannot be added into *Axis_ID_Array.

PCI(e)-8158, choose {4,5,6} be APS_spiral_ce_v(), axis 7 operation is always a dummy motion and it cannot be added into *Axis_ID_Array.

Syntax:

C/C++:

I32 FNTYPE APS_spiral_ce_v(I32 *Axis_ID_Array, I32 Option, F64 *CenterArray, F64 *NormalArray, F64 *EndArray, I16 Dir, F64 *TransPara, F64 Vm, ASYNCALL *Wait);

Visual Basic:

APS_spiral_ce_v(Axis_ID_Array As Long, ByVal Option As Long, CenterArray As Double, NormalArray As Double, EndArray As Double, ByVal Dir As Short, TransPara As Double, ByVal Vm As Double, Wait As ASYNCALL) As Long

Parameters:

For PCI(e)-8154/58:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

Note: The axes specified in Axis_ID_Array must be of the same card.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~15: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *NormalArray: A pointer indicates the starting address of the normal vector array.

F64 *EndArray: A pointer indicates the starting address of end array.

I16 Dir: A value specifies the rotate direction. If Dir >= 0 means rotate in positive direction(counterclockwise), Dir <= -1 (clockwise) rotate in negative direction.

F64 *TransPara: A pointer indicates the starting address of transfer parameters. **Note: It is reserved for future**.

F64 Vm: A value specifies the maximum velocity.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

For PCI-8254/58 / AMP-204/8C and PCIe-833x, ECAT-4XMO:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Reserved. **Note: if setting to mode 2, it will return error code.**

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance **>= 0.0**)

0110b(6): Blending – Residue distance in travel distance's

percentage(TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *NormalArray: A pointer indicates the starting address of the normal vector array.

F64 *EndArray: A pointer indicates the starting address of end array.

I16 Dir: A value specifies the rotate direction. If dir set 0 means rotate in positive direction, dir = -1 rotate in negative direction. The total rotate *angle = theta + **Dir** x 2PI*, where *theta* is the angle of two vectors: center to start and center to end.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

F64 Vm: A value specifies the maximum velocity.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for PCI(e)-8154/58

```
I32 opt = 1; //relative mode
I32 Axis_ID_Array[3] = {0, 1, 2};
F64 NormalArray[3] = { 0, 1, 0 }; //      choose axis 1 is vertical axis
F64 Center_Pos_Array[2] = {10000, 0, 20000};
F64 End_Pos_array[2] = {20000, 20000, 40000};
I16 Dir = 1;            //  rotate counterclockwise
F64 TransPara = 0;
F64 Vm = 10000;
ASYNCALL *wait = NULL;

APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 100000 ); //Set acceleration
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 100000 ); //Set deceleration

APS_spiral_ce_v (
```

```
    Axis_ID_Array           // I32 *Axis_ID_Array
    , opt                   // I32 Option
    , Center_Pos_Array      // F64 * CenterArray
  , NormalArray             // F64 * NormalArray
  , End_Pos_array           // F64 * Enday
  , Dir                     // I16 Dir
  , &TransPara              // reserved
  , Vm                      // Vm
  , wait );                 // reserved
```

See also:

APS_absolute_helical_move();APS_relative_helical_move();APS_spiral_ce()

APS_spiral_ce_all	Begin a 3D spiral-helix move of end position with all profile

Support Products: PCI(e)-8154/58 , PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to execute a 3D spiral-helix interpolation of end position type, named _spiral_ce. It follows with center position, normal vector, end position and Dir. The _all suffix represents that all motion profiles, including Vs, Vm, Ve, Acc, Dec and SFac, is necessary to perform a 3D spiral-helix interpolation. Other motion profiles are set in axis parameters. User could refer to axis parameter table for the details.

Base on PCI(e)-8154/58, this function support 1 fixed radius circular interpolation and synchronized linear travel in normal axis. The rotate angle is the theta of two vectors: center to start and center to end. Unlike PCIe–8338 and PCI-8254/58 with soft motion base, the circular interpolation angle range only has -360◦to 360◦. That means spiral curve only has 1 pitch, just like helical curve.

Note : Due to 8154/58 limit, 4th / 8th axis operation will be a dummy motion and it can't be used for any other puspose. This axis need to be set servo-off. If not, it will return ERR_InServoOnState(-48)

e.g.

PCI(e)-8154, choose {0,1,2} be APS_spiral_ce_v(), axis 3 operation is always a dummy motion and it cannot be added into *Axis_ID_Array.

PCI(e)-8158, choose {4,5,6} be APS_spiral_ce_v(), axis 7 operation is always a dummy motion and it cannot be added into *Axis_ID_Array.

Syntax:

C/C++:

I32 FNTYPE APS_spiral_ce_all(I32 *Axis_ID_Array, I32 Option, F64 *CenterArray, F64 *NormalArray, F64 *EndArray, I16 Dir, F64 *TransPara, F64 Vs, F64 Vm, F64 Ve, F64 Acc,F64 Dec, F64 SFac, ASYNCALL *Wait);

Visual Basic:

APS_spiral_ce_all(Axis_ID_Array As Long, ByVal Option As Long, CenterArray As Double, NormalArray As Double, EndArray As Double, ByVal Dir As Short, TransPara As Double, ByVal Vs As Double, ByVal Vm As Double, ByVal Ve As Double, ByVal Acc As Double, ByVal Dec As Double, ByVal SFac As Double, Wait As ASYNCALL) As Long

Parameters:

For PCI(e)-8154/58:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

Note: The axes specified in Axis_ID_Array must be of the same card.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~15: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *NormalArray: A pointer indicates the starting address of the normal vector array.

F64 *EndArray: A pointer indicates the starting address of end array.

I16 Dir: A value specifies the rotate direction. If Dir >= 0 means rotate in positive direction(counterclockwise), Dir <= -1 (clockwise) rotate in negative direction.

F64 *TransPara: A pointer indicates the starting address of transfer parameters. **Note: It is reserved for future**.

F64 Vs: A value specifies the starting velocity.

F64 Vm: A value specifies the maximum velocity.

F64 Ve: A value specifies the ending velocity.

F64 Acc: A value specifies the acceleration.

F64 Dec: A value specifies the deceleration.

F64 SFac: A value specifies the s factor.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

For PCI-8254/58 / AMP-204/8C and PCIe-833x, ECAT-4XMO:

I32 *Axis_ID_Array: A pointer indicates the starting address of axes array.

I32 Option: A bit set specifies the option, which could enable specified parameters and functions.

7	6	5	4	3	2	1	Bit : 0
							Absolute(0) / Relative(1)
15	14	13	12	11	10	9	8
Buffer mode							Wait trigger

Bit 0: 1:Relative move, 0:Absolute move

Bit 1~7: Reserved for future, set to 0.

Bit 8: Set a move to waiting state. This axis will not move until triggered.

Bit 9~11: Reserved for future, set to 0.

Bit 12~15: Buffer mode:

0000b(0): Aborting – stop and blend (TransPara_0 as deceleration. [dec >0], if dec <= 0, kernel takes new coming deceleration.)

0001b(1): Aborting – force abort

0010b(2): Reserved. **Note: if setting to mode 2, it will return error code.**

0011b(3): Buffered

0100b(4): Blending – Deceleration event

0101b(5): Blending – Residue distance (TransPara_0 as residue distance **>= 0.0**)

0110b(6): Blending – Residue distance in travel distance's percentage(TransPara_0 as residue distance % value range: 0.0 ~ 1.0)

Bit 16~: Reserved for future, set to 0.

F64 *CenterArray: A pointer indicates the starting address of center array.

F64 *NormalArray: A pointer indicates the starting address of the normal vector array.

F64 *EndArray: A pointer indicates the starting address of end array.

I16 Dir: A value specifies the rotate direction. If dir set 0 means rotate in positive direction, dir = -1 rotate in negative direction. The total rotate *angle = theta + **Dir** x 2PI*, where *theta* is the angle of two vectors: center to start and center to end.

F64 *TransPara: A pointer indicates the starting address of transfer parameters.

F64 Vs: A value specifies the starting velocity.

F64 Vm: A value specifies the maximum velocity.

F64 Ve: A value specifies the ending velocity.

F64 Acc: A value specifies the acceleration.

F64 Dec: A value specifies the deceleration.

F64 SFac: A value specifies the s factor.

ASYNCALL *Wait: A pointer to ASYNCALL structure. **Note: It is reserved for future.**

Passing it with NULL defines a waiting call.

If it is a valid pointer, the call is non-waiting and the functions returns immediately.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for PCI(e)-8154/58

```
I32 opt = 1; //relative mode
I32 Axis_ID_Array[3] = {0, 1, 2};
F64 NormalArray[3] = { 0, 1, 0 }; //      choose axis 1 is vertical axis
F64 Center_Pos_Array[2] = {10000, 0, 20000};
```

```
F64 End_Pos_array[2] = {20000, 20000, 40000};
I16 Dir = 1;            //  rotate counterclockwise
F64 TransPara = 0;
ASYNCALL *wait = NULL;

APS_set_axis_param( Master_Axis_ID, PRA_CURVE, 1 ); //Set S-curve
APS_set_axis_param( Master_Axis_ID, PRA_ACC, 100000 ); //Set acceleration
APS_set_axis_param( Master_Axis_ID, PRA_DEC, 100000 ); //Set deceleration

APS_spiral_ce_all (
        Axis_ID_Array           // I32 *Axis_ID_Array
        , opt                   // I32 Option
        , Center_Pos_Array      // F64 * CenterArray
      , NormalArray             // F64 * NormalArray
      , End_Pos_array           // F64 * Enday
      , Dir                     // I16 Dir
      , &TransPara              // reserved
      , 0                       // Vs
      , 10000                   // Vm
      , 0                       // Ve
      , 11111                   // Acc
      , 11111                   // Dec
      , Sfac                    // 1, S-curve
      , wait );                 // reserved
```

See also:

APS_absolute_helical_move();APS_relative_helical_move(); APS_spiral_ce()

11. Interrupt

APS_int_enable	Interrupt main switch

Support Products: PCI-8253/56, PCI-8392(H), DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-7856, PCI(e)-8154/8158, PCI-8102/PCI-C154(+), PCI-8254/58 / AMP-204/8C , PCIe-833x, AMP-104C

Descriptions:

This function is used to enable/disable interrupt of one board to host computer. It is a hardware main switch of this board. Once it is disabled, host computer will not received any hardware interrupt even the interrupt factor is enabled. Users must enable this function before using any interrupt relative functions and disable this function when users do not use interrupt anymore.

Syntax:

C/C++:

I32 FNTYPE APS_int_enable(I32 Board_ID, I32 Enable);

Visual Basic:

APS_int_enable (ByVal Board_ID As Long, ByVal Enable As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Enable: Enable/Disable interrupt.

0: Disable. 1: Enable

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Int_No; //Interrupt number
I32 returnCode; // function return code

Int_No = APS_set_int_factor( Board_ID, Item_No, Factor_No, 1 ); //Enable the interrupt factor
APS_int_enable( Board_ID, 1 ); //Enable the interrupt main switch
returnCode = APS_wait_single_int( Int_No, Time_Out ); //Wait interrupt
if( returnCode == ERR_NoError )
{ //Interrupt occurred
```

```
        APS_reset_int( Int_No );
        …//Do something
}

APS_set_int_factor( Board_ID, Item_No, Factor_No, 0 ); //Disable the interrupt factor
APS_int_enable( Board_ID, 0 ); //Disable the interrupt main switch
```

See also:

APS_set_int_factor(); APS_get_int_factor();APS_wait_single_int();APS_wait_multiple_int();
APS_reset_int(); APS_set_int();

APS_set_int_factor	Enable/Disable interrupt factor and get interrupt handle.

Support Products : PCI-8253/56, PCI-8392(H), DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-7856, PCI(e)-8154/8158, PCI-8102/PCI-C154(+), PCI-8254/58 / AMP-204/8C , PCIe-833x, AMP-104C

Descriptions :

This function is used to turn on/off the interrupt factor bit. If it is turned on, the function will return a notification event for this bit and return an I32 type event number. Users can wait this event by assigning corresponding event number into a wait function. The event number is unique in one system but it is not a event handler. It is just a virtual number of event APS converts.

The interrupt factor definition, please refer to the interrupt factor table.

Syntax:

C/C++:

I32 FNTYPE APS_set_int_factor(I32 Board_ID, I32 Item_No, I32 Factor_No, I32 Enable);

Visual Basic:

APS_set_int_factor (ByVal Board_ID As Long, ByVal Item_No As Long, ByVal Factor_No As Long, ByVal Enable As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Item_No: Interrupt factor table item number. Refer to interrupt factor table.

I32 Factor_No: Factor number of one item. Refer to interrupt factor table.

I32 Enable: Enable interrupt factor. 0: Disable; 1:Enable

Return Values:

When:

[Enable = 1] : Enable the interrupt factor

Return positive value: I32 Interrupt event number.

Return negative value: I32 Error code: Please refer to APS Functions Return Code.

[Enable = 0] : Disable the interrupt factor

Return I32 Error code: Please refer to APS Functions Return Code.

Example1:

```
<Set axis 2 NSTP interrupt of PCI-8392 or PCI-8253/56>
I32 Int_No; //Interrupt number
I32 returnCode; // function return code

Int_No = APS_set_int_factor( Board_ID, Item_No=2, Factor_No=BIT12, 1 ); //Enable the
interrupt factor
APS_int_enable( Board_ID, 1 ); //Enable the interrupt main switch
returnCode = APS_wait_single_int( Int_No, Time_Out ); //Wait interrupt
if( returnCode == ERR_NoError )
{ //Interrupt occurred
        APS_reset_int( Int_No );
    …//Do something
}

APS_set_int_factor( Board_ID, Item_No, Factor_No, 0 ); //Disable the interrupt factor
APS_int_enable( Board_ID, 0 ); //Disable the interrupt main switch
```

Example2:

```
<Set axis 2 IMDN interrupt of PCI-8254/58
I32 Int_No; //Interrupt number
I32 returnCode; // function return code

Int_No = APS_set_int_factor( Board_ID, Item_No=2, Factor_No=BIT12, 1 ); //Enable the
interrupt factor
APS_int_enable( Board_ID, 1 ); //Enable the interrupt main switch
returnCode = APS_wait_single_int( Int_No, Time_Out ); //Wait interrupt
if( returnCode == ERR_NoError )
{ //Interrupt occurred
        APS_reset_int( Int_No );
    …//Do something
}

APS_set_int_factor( Board_ID, Item_No, Factor_No, 0 ); //Disable the interrupt factor
APS_int_enable( Board_ID, 0 ); //Disable the interrupt main switch
```

See also:

APS_int_enable();APS_get_int_factor();APS_wait_single_int();APS_wait_multiple_int();
APS_reset_int(); APS_set_int()

APS_get_int_factor	Get interrupt factor enable or disable

Support Products : PCI-8253/56, PCI-8392(H), DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-7856, PCI(e)-8154/8158, PCI-8102/PCI-C154(+), PCI-8254/58 / AMP-204/8C, PCIe-833x, AMP-104C

Descriptions :

This function is used to get the setting of interrupt factor.

Syntax:

C/C++:

I32 FNTYPE APS_get_int_factor(I32 Board_ID, I32 Item_No, I32 Factor_No, I32 *Enable);

Visual Basic:

APS_get_int_factor (ByVal Board_ID As Long, ByVal Item_No As Long, ByVal Factor_No As Long, Enable As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Item_No: Interrupt factor table item number. Refer to interrupt factor table.

I32 Factor_No: Factor number of one item. Refer to interrupt factor table.

I32 *Enable: Return enable or disable. 0: Disable, 1:Enable.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ReturnCode;
I32 Enable;
ReturnCode = APS_get_int_factor( Board_ID, Item_No, Factor_No, &Enable );
…
```

See also:

APS_int_enable();APS_set_int_factor();APS_wait_single_int();APS_wait_multiple_int();

APS_reset_int(); APS_set_int()

APS_wait_single_int	Wait single interrupt event

Support Products: PCI-8253/56, PCI-8392(H), DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-7856, PCI(e)-8154/8158, PCI-8102/PCI-C154(+), PCI-8254/58 / AMP-204/8C, PCIe-833x, AMP-104C

Descriptions:

When the user enabled the interrupt function for specified factors by “APS_set_int_factor”, it could use this function to wait a specific interrupt. When this function was running, the process would never stop until the event was be triggered or the function was time out. This function returns when one of the following occurs:

1. The specified interrupt factor is in the signaled state.
2. The time-out interval elapses.

This function checks the current state of the specified interrupt factor. If the state is non-signaled, the calling thread enters the wait state. It uses no processor time while waiting for the INT state to become signaled or the time-out interval to elapse.

When the interrupt is occurred and the wait function is return. User should use **APS_reset_int ()** to reset the interrupt by themselves. If user does not reset the interrupt, the wait function will pass immediately next time.

Syntax:

C/C++:

I32 FNTYPE APS_wait_single_int(I32 Int_No, I32 Time_Out);

Visual Basic:

APS_wait_single_int (ByVal Int_No As Long, ByVal Time_Out As Long) As Long

Parameters:

I32 Int_No: Interrupt event number. Get from APS_set_int_factor() function.

I32 Time_Out: Wait timeout time. Unit is milli-second. If value is set -1, the function’s time-out interval never elapses (infinite). If *Time_Out* is zero, the function tests the interrupt’s state and returns immediately.

Return Values:

ERR_NoError(0): The event is wait success.

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 Int_No; //Interrupt number

```
I32 returnCode; // function return code

Int_No = APS_set_int_factor( Board_ID, Item_No, Factor_No, 1 ); //Enable the interrupt factor
APS_int_enable( Board_ID, 1 ); //Enable the interrupt main switch
returnCode = APS_wait_single_int( Int_No, Time_Out ); //Wait interrupt
if( returnCode == ERR_NoError )
{ //Interrupt occurred
        APS_reset_int( Int_No );
        …//Do something
}

APS_set_int_factor( Board_ID, Item_No, Factor_No, 0 ); //Disable the interrupt factor
APS_int_enable( Board_ID, 0 ); //Disable the interrupt main switch
```

See also:

APS_int_enable();APS_set_int_factor();APS_get_int_factor();APS_wait_multiple_int();
APS_reset_int(); APS_set_int()

APS_wait_multiple_int	Wait multiple interrupt events

Support Products: PCI-8253/56, PCI-8392(H), DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-7856, PCI(e)-8154/8158, PCI-8102/PCI-C154(+), PCI-8254/58 / AMP-204/8C, PCIe-833x, AMP-104C

Descriptions:

When the user enabled the interrupt function for specified factors by "**APS_set_int_factor()**", users could use this function to wait specific interrupts. When this function was running, the process would never stop until the event was be triggered or the function was time out. This function returns when one of the following occurs:

1. Either any one or all of the interrupt factors are in the signaled state.
2. The time-out interval elapses.

This function checks the current state of the specified interrupt factor. If the state is non-signaled, the calling thread enters the wait state. It uses no processor time while waiting for the INT state to become signaled or the time-out interval to elapse.

Users must use **APS_reset_int()** to reset the events themselves before wait the events next time.

Syntax:

C/C++:

I32 FNTYPE APS_wait_multiple_int(I32 Int_Count, I32 *Int_No_Array, I32 Wait_All, I32 Time_Out);

Visual Basic:

APS_wait_multiple_int (ByVal Int_Count As Long, Int_No_Array As Long, ByVal Wait_All As Long, ByVal Time_Out As Long) As Long

Parameters:

I32 Int_Count: Specifies the number of Interrupt. The maximum number of factors is 64.

I32 *Int_No_Array: Interrupt event number array. Get from APS_set_int_factor() function.

I32 Wait_All: Wait option.

FALSE: (0) The function returns when the state of any one of the events in the array is signaled.

TRUE: (1) The function returns when the state of all events in the array is signaled.

I32 Time_Out: Wait timeout time. Unit is milli-second. If value is set -1, the function's time-out interval never elapses (infinite).

Return Values:

Postive value: (Int_Count – 1): The events are wait success.

If Wait_All is FALSE (0), the return value indicates that the state of all specified objects is signaled.

If Wait_All is FALSE(0), the return value indicates the array index of the object that satisfied the wait. If more than one event became operation during the call, this is the array index of the operation object with the smallest index value of all the operation objects.

Negative value: I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Int_No[2]; //Interrupt number
I32 returnCode; // function return code

Int_No[0] = APS_set_int_factor( Board_ID, Item_No1, Factor_No1, 1 ); //Enable the interrupt factor
Int_No[1] = APS_set_int_factor( Board_ID, Item_No2, Factor_No2, 1 ); //Enable the interrupt factor

APS_int_enable( Board_ID, 1 ); //Enable the interrupt main switch
returnCode = APS_wait_multiple_int( 2, Int_No, 1, Time_Out ); //Wait multiple interrupts, (wait all)
if( returnCode >= ERR_NoError )
{ //Interrupts occurred
    APS_reset_int( Int_No[0] );
    APS_reset_int( Int_No[1] );
    …//Do something
}

APS_set_int_factor( Board_ID, Item_No1, Factor_No1, 0 ); //Disable the interrupt factor
APS_set_int_factor( Board_ID, Item_No2, Factor_No2, 0 ); //Disable the interrupt factor
APS_int_enable( Board_ID, 0 ); //Disable the interrupt main switch
```

See also:

APS_int_enable(); APS_set_int_factor();APS_get_int_factor();APS_wait_single_int(); APS_reset_int(); APS_set_int()

APS_wait_error_int	Wait error interrupts (non-mask)

Support Products: PCI(e)-8154/8158, PCI-8102/PCI-C154(+)

Descriptions:

Users could use this function to wait error interrupts. When this function was running, the process would never stop until the event was be triggered or the function was time out. This function returns when one of the following occurs:

1. Either any one or all of the error interrupts are in the signaled state.
2. The time-out interval elapses.

This function checks the current state of the error interrupts. If the state is non-signaled, the calling thread enters the wait state. It uses no processor time while waiting for the INT state to become signaled or the time-out interval to elapse.

When the error interrupt is occurred and the wait function is return.

Syntax:

I32 FNTYPE APS_wait_error_int(I32 Board_ID, I32 Item_No, I32 Time_Out);

APS_wait_single_int (ByVal Board_ID As Long, ByVal Item_No As Long, ByVal Time_Out As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Item_No: Interrupt factor table item number. Refer to interrupt factor table.

I32 Time_Out: Wait timeout time. Unit in mini-second. If value is set -1, the function's time-out interval never elapses (infinite). If *Time_Out* is zero, the function tests the interrupt's state and returns immediately.

Return Values:

When:

[Enable = 1] : Enable the interrupt

Return positive value: I32 Error interrupt event number or Time_Out.

Return negative value: I32 Error code: Please refer to APS Functions Return Code.

[Enable = 0] : Disable the interrupt

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 returnCode; // function return code

```
APS_int_enable( Board_ID, 1 ); //Enable the interrupt main switch
returnCode = APS_wait_ error_int(Board_ID , Item_No, Time_Out ); //Wait error interrupt
if( returnCode >= 0 )
{
	//Interrupts occurred or Time_Out
	//Do something
}
APS_int_enable( Board_ID, 0 ); //Disable the interrupt main switch
```

See also:

APS_int_enable();APS_set_int_factor();APS_get_int_factor();APS_wait_single_int();
APS_wait_multiple_int(); APS_reset_int(); APS_set_int()

APS_reset_int	Reset interrupt event to non-signaled state.

Support Products : PCI-8253/56, PCI-8392(H), DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-7856, PCI(e)-8154/8158, PCI-8102/PCI-C154(+), PCI-8254/58 / AMP-204/8C , PCIe-833x, AMP-104C

Descriptions :

This function is used to reset singled event to non-singled state.

Syntax:

C/C++:

I32 FNTYPE APS_reset_int(I32 Int_No);

Visual Basic:

APS_reset_int (ByVal Int_No As Long) As Long

Parameters:

I32 Int_No: Interrupt event number. Get from APS_set_int_factor() function.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Int_No; //Interrupt number
I32 returnCode; // function return code

Int_No = APS_set_int_factor( Board_ID, Item_No, Factor_No, 1 ); //Enable the interrupt factor
APS_int_enable( Board_ID, 1 ); //Enable the interrupt main switch
returnCode = APS_wait_single_int( Int_No, Time_Out ); //Wait interrupt
if( returnCode == ERR_NoError )
{ //Interrupt occurred
        APS_reset_int( Int_No );
        …//Do something
}

APS_set_int_factor( Board_ID, Item_No, Factor_No, 0 ); //Disable the interrupt factor
APS_int_enable( Board_ID, 0 ); //Disable the interrupt main switch
```

See also:

APS_int_enable();APS_set_int_factor();APS_get_int_factor();APS_wait_single_int();
APS_wait_multiple_int(); APS_set_int()

APS_set_int	Set interrupt event to signaled state.

Support Products : PCI-8253/56, PCI-8392(H), DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-7856, PCI(e)-8154/8158, PCI-8102/PCI-C154(+), PCI-8254/58 / AMP-204/8C , PCIe-833x, AMP-104C

Descriptions :

This function is used to signal the specified event interrupt. The wait function will return (pass) when this function is set.

Syntax:

C/C++:

I32 FNTYPE APS_set_int(I32 Int_No);

Visual Basic:

APS_set_int (ByVal Int_No As Long) As Long

Parameters:

I32 Int_No: Interrupt event number. Get from APS_set_int_factor() function.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Int_No; //Interrupt number
I32 returnCode; // function return code

Int_No = APS_set_int_factor( Board_ID, Item_No, Factor_No, 1 ); //Enable the interrupt factor
APS_int_enable( Board_ID, 1 ); //Enable the interrupt main switch
APS_set_int(Int_No ); //Signaled interrupt event.
returnCode = APS_wait_single_int( Int_No, Time_Out ); //Wait function will pass immediately
if( returnCode == ERR_NoError )
{ //Interrupt occurred
        APS_reset_int( Int_No );
        …//Do something
}

APS_set_int_factor( Board_ID, Item_No, Factor_No, 0 ); //Disable the interrupt factor
APS_int_enable( Board_ID, 0 ); //Disable the interrupt main switch
```

See also:

APS_int_enable();APS_set_int_factor();APS_get_int_factor();APS_wait_single_int();
APS_wait_multiple_int(); APS_reset_int()

APS_set_int_factorH	Enable/Disable interrupt factor and get interrupt handle.(Win32)

Support Products : PCI-8253/56, PCI-8392(H), DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-7856, PCI(e)-8154/8158, PCI-8102/PCI-C154(+), PCI-8254/58 / AMP-204/8C , PCIe-833x, AMP-104C

Descriptions :

This function is used to turn on/off the interrupt factor bit. If it is turned on, the function will return a notification event for this bit and return a HANDLE type (define in windows.h) event handle. Users can use this handle directly with win32 API functions. The event number is unique in one system.

The interrupt factor definition, please refer to the interrupt factor table.

Syntax:

C/C++:

HANDLE APS_set_int_factorH(I32 Board_ID, I32 Item_No, I32 Factor_No, I32 Enable);

Visual Basic:

APS_set_int_factorH (ByVal Board_ID As Long, ByVal Item_No As Long, ByVal Factor_No As Long, ByVal Enable As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Item_No: Interrupt factor table item number. Refer to interrupt factor table.

I32 Factor_No: Factor number of one item. Refer to interrupt factor table.

I32 Enable: Enable interrupt factor. 0: Disable; 1:Enable

Return Values:

When:

[Enable = 1] : Enable the interrupt factor

Return win32 event handle if function success, or return null(0) for failed.

[Enable = 0] : Disable the interrupt factor

Return null(0).

Example:

```
#include <windows.h>
HANDLE hInt; //Interrupt handle
DWORD returnCode; // function return code
```

```
hInt = APS_set_int_factorH( Board_ID, Item_No, Factor_No, 1 ); //Enable the interrupt factor
APS_int_enable( Board_ID, 1 ); //Enable the interrupt main switch
returnCode = WaitForSingleObject( hInt, 1000 );

if( returnCode == WAIT_OBJECT_0 )
{ //Interrupt occurred
        ResetEvent (hInt ); //Win32 SDK function
    …//Do something
}

APS_set_int_factor( Board_ID, Item_No, Factor_No, 0 ); //Disable the interrupt factor
APS_int_enable( Board_ID, 0 ); //Disable the interrupt main switch
```

See also:

APS_int_enable();APS_get_int_factor();APS_wait_single_int();APS_wait_multiple_int();
APS_reset_int(); APS_set_int()

APS_int_no_to_handle	Convert interrupt event number to interrupt handle.(Win32)

Support Products : PCI-8253/56, PCI-8392(H), DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-7856, PCI(e)-8154/8158, PCI-8102/PCI-C154(+), PCI-8254/58 / AMP-204/8C , PCIe-833x

Descriptions :

This function is used to convert interrupt number to a HANDLE type (define in windows.h) event handle. User could get an I32 type event number by APS_set_factor(), then convert this number to a HANDLE.

Syntax:

C/C++:

HANDLE APS_int_no_to_handle(I32 Int_No);

Visual Basic:

APS_int_no_to_handle(ByVal Int_No As Long) As Long

Parameters:

I32 Int_No: Interrupt event number. Get from APS_set_int_factor() function.

Return Values:

Return win32 event handle.

Example:

```
#include <windows.h>
HANDLE hInt; //Interrupt handle
I32 Int_No;
DWORD returnCode; // function return code

Int_No = APS_set_int_factor( Board_ID, Item_No, Factor_No, 1 ); //Enable the interrupt factor
hInt = APS_int_no_to_handle( Int_No ); //Convert to a handle.
APS_int_enable( Board_ID, 1 ); //Enable the interrupt main switch

returnCode = WaitForSingleObject( hInt, 1000 );

if( returnCode == WAIT_OBJECT_0 )
{ //Interrupt occurred
```

```
            ResetEvent (hInt ); //Win32 SDK function
        …//Do something
}

APS_set_int_factor( Board_ID, Item_No, Factor_No, 0 ); //Disable the interrupt factor
APS_int_enable( Board_ID, 0 ); //Disable the interrupt main switch
```

See also:

APS_int_enable(); APS_set_int_factor(); APS_set_field_bus_int_factor_motion ()

APS_set_field_bus_int_factor_motion	Enable/Disable motion interrupt factor and get interrupt handle for MotionNet series on PCI(e)-7856.

Support Products : PCI(e)-7856, MNET-4XMO-(C), MNET-1XMO

Descriptions :

This function is used to turn on/off the interrupt factor bit on MNET products. If it is turned on, the function will return a notification event for this bit and return an I32 type event number. Users can wait this event by assigning corresponding event number into a wait function. The event number is unique in one system but it is not an event handler. It is just a virtual number of event APS converts.

The MotionNet motion interrupt factor definition, please refer to the interrupt factor table.

Note that be sure to set to interrupt mode, bit 6 set to 1, by calling APS_initial().

Note that you should call this function after starting field bus. Be sure all axes of the MENT field bus were built by calling APS_start_field_bus(). Then, user can set interrupt factor to specialized axis by using this function. Otherwise, error code returns.

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_int_factor_motion(I32 Axis_ID, I32 Factor_No, I32 Enable);

Visual Basic:

APS_set_field_bus_int_factor_motion (ByVal Axis_ID As Long, ByVal Factor_No As Long, ByVal Enable As Long) As Long

Parameters:

I32 Axis_ID: Specialized axis of MNET system.

I32 Factor_No: Factor number of axes. Refer to interrupt factor table.

I32 Enable: Enable interrupt factor. 0: Disable; 1:Enable

Return Values:

When:

[Enable = 1] : Enable the interrupt factor

Return positive value: I32 Interrupt event number.

Return negative value: I32 Error code: Please refer to APS Functions Return Code.

[Enable = 0] : Disable the interrupt factor

Return I32 Error code: Please refer to APS Functions Return Code.

Example:

```
<Set axis 1000 INSTP (BIT 0) interrupt ON MotionNet field bus on PCI(e)-7856 >
I32 Axis_ID = 1000; //MNET's axis
I32 returnCode; // function return code

//Enable the interrupt factor
Int_No = APS_set_field_bus_int_factor_motion ( Board_ID, Axis_ID, Factor_No=0, 1 );
APS_int_enable( Board_ID, 1 ); //Enable the interrupt main switch
//Reset interrupt status of the axis
APS_reset_field_bus_int_motion ( Axis_ID );
returnCode = APS_wait_single_int( Int_No, Time_Out ); //Wait interrupt
if( returnCode == ERR_NoError )
{ //Interrupt occurred
        APS_reset_int( Int_No );
    …//Do something
}

APS_set_field_bus_int_factor_motion ( Axis_ID, Factor_No, 0 ); //Disable the interrupt factor
APS_int_enable( Board_ID, 0 ); //Disable the interrupt main switch
```

See also:

APS_int_enable();APS_get_field_bus_int_factor_motion();APS_wait_single_int();
APS_wait_multiple_int(); APS_reset_int(); APS_set_int()

APS_get_field_bus_int_factor_motion	Get motion interrupt factor enable or disable for MotionNet series on PCI(e)-7856

Support Products : PCI(e)-7856, MNET-4XMO-(C), MNET-1XMO

Descriptions :

This function is used to get the setting of interrupt factor.

Note that you should call this function after starting field bus. Be sure all axes of the port were built by calling APS_start_field_bus(). Then, user can get interrupt factor from specialized axis by using this function. Otherwise, error code returns.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_int_factor_motion(I32 Axis_ID, I32 Factor_No, I32 *Enable);

Visual Basic:

APS_get_field_bus_int_factor_motion (ByVal Axis_ID As Long, ByVal Factor_No As Long, Enable As Long) As Long

Parameters:

I32 Axis_ID: Specialized axis of MNET system.

I32 Factor_No: Factor number of axes. Refer to interrupt factor table.

I32 *Enable: Return enable or disable. 0: Disable, 1:Enable.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ReturnCode;
I32 Enable;
ReturnCode = APS_get_field_bus_int_factor_motion ( Axis_ID, Factor_No, &Enable );
…
```

See also:

APS_int_enable();APS_set_field_bus_int_factor_motion();APS_wait_single_int();
APS_wait_multiple_int(); APS_reset_int(); APS_set_int()

APS_set_field_bus_int_factor_error	Enable/Disable error interrupt factor and get error interrupt handle for MotionNet series on PCI(e)-7856.

Support Products : PCI(e)-7856, MNET-4XMO-(C), MNET-1XMO

Descriptions :

This function is used to turn on/off the error interrupt factor bit on MNET products. If it is turned on, the function will return a notification event for this bit and return an I32 type event number. Users can wait this event by assigning corresponding event number into a wait function. The event number is unique in one system but it is not an event handler. It is just a virtual number of event APS converts.

The MotionNet error interrupt factor definition, please refer to the interrupt factor table.

Note that all default error factors are turned on.

Note that be sure to set to interrupt mode, bit 6 set to 1, by calling APS_initial().

Note that you should call this function after starting field bus. Be sure all axes of the MENT field bus were built by calling APS_start_field_bus(). Then, user can set interrupt factor to specialized axis by using this function. Otherwise, error code returns.

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_int_factor_error(I32 Axis_ID, I32 Factor_No, I32 Enable);

Visual Basic:

APS_set_field_bus_int_factor_error(ByVal Axis_ID As Long, ByVal Factor_No As Long, ByVal Enable As Long) As Long

Parameters:

I32 Axis_ID: Specialized axis of MNET system.

I32 Factor_No: Factor number of axes. Refer to error interrupt factor table.

I32 Enable: Enable interrupt factor. 0: Disable; 1:Enable

Return Values:

When:

[Enable = 1] : Enable the error interrupt factor

Return positive value: I32 Interrupt event number.

Return negative value: I32 Error code: Please refer to APS Functions Return Code.

[Enable = 0] : Disable the error interrupt factor

Return I32 Error code: Please refer to APS Functions Return Code.

Example:

```
<Set axis 1000 EPEL (BIT 5) error interrupt ON MotionNet field bus on PCI(e)-7856 >
I32 Axis_ID = 1000; //MNET's axis
I32 returnCode; // function return code

//Enable the interrupt factor
Int_No = APS_set_field_bus_int_factor_error ( Board_ID, Axis_ID, Factor_No=5, 1 );
APS_int_enable( Board_ID, 1 ); //Enable the interrupt main switch
//Reset interrupt status of the axis
APS_reset_field_bus_int_motion ( Axis_ID );
returnCode = APS_wait_single_int( Int_No, Time_Out ); //Wait interrupt
if( returnCode == ERR_NoError )
{ //Interrupt occurred
        APS_reset_int( Int_No );
    …//Do something
}

APS_set_field_bus_int_factor_error ( Axis_ID, Factor_No, 0 ); //Disable the error interrupt
factor
APS_int_enable( Board_ID, 0 ); //Disable the interrupt main switch
```

See also:

APS_int_enable();APS_set_field_bus_int_factor_motion();APS_get_field_bus_int_factor_motio n(); APS_wait_single_int();APS_wait_multiple_int();APS_reset_int();APS_set_int(); APS_get_field_bus_int_factor_error(); APS_wait_field_bus_error_int_motion()

APS_get_field_bus_int_factor_error	Get error interrupt factor status for MotionNet series on PCI(e)-7856

Support Products : PCI(e)-7856, MNET-4XMO-(C), MNET-1XMO

Descriptions :

This function is used to get the setting of error interrupt factor.

Note that all default error factors are turned on.

Note that you should call this function after starting field bus. Be sure all axes of the port were built by calling APS_start_field_bus(). Then, user can get error interrupt factor from specialized axis by using this function. Otherwise, error code returns.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_int_factor_error (I32 Axis_ID, I32 Factor_No, I32 *Enable);

Visual Basic:

APS_get_field_bus_int_factor_error (ByVal Axis_ID As Long, ByVal Factor_No As Long, Enable As Long) As Long

Parameters:

I32 Axis_ID: Specialized axis of MNET system.

I32 Factor_No: Factor number of axes. Refer to error interrupt factor table.

I32 *Enable: Return enable or disable. 0: Disable, 1:Enable.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ReturnCode;
I32 Enable;
ReturnCode = APS_get_field_bus_int_factor_error ( Axis_ID, Factor_No, &Enable );
…
```

See also:

APS_int_enable();APS_set_field_bus_int_factor_motion();APS_get_field_bus_int_factor_motion(); APS_wait_single_int();APS_wait_multiple_int();APS_reset_int();APS_set_int(); APS_get_field_bus_int_factor_error (); APS_wait_field_bus_error_int_motion()

APS_reset_field_bus_int_motion	Reset interrupt status of axes for MotionNet series on PCI(e)-7856.

Support Products : PCI(e)-7856, MNET-4XMO-(C), MNET-1XMO

Descriptions :

This function is used to reset interrupt status of axes.

After the user enabled the interrupt function by “APS_int_enable()”, users should use this function to reset interrupt status which remain in slave modules.
Residual interrupt status in slave modules will cause unexpected procedure such as breaking the interrupt mechanism. Be sure to reset those interrupt status of axes after user enabled the interrupt function.

Syntax:

C/C++:

I32 FNTYPE APS_reset_field_bus_int_motion (I32 Axis_ID);

Visual Basic:

APS_reset_field_bus_int_motion (ByVal Axis_ID As Long) As Long

Parameters:

I32 Axis_ID: Specialized axis of MNET system.

I32 Time_Out: Wait timeout time. Unit is milli-second. If value is set -1, the function’s time-out interval never elapses (infinite).

Return Values:

Postive value: (Int_Count – 1): The events are wait success.

The return value indicates the index of the error events that satisfied the wait. If more than one event became operation during the call, this is the array index of the signaled events with the smallest index value of all the signaled events.

Negative value: I32 Error code: Please refer to APS Functions Return Code.

Example:

.. set factor by axis

```
APS_int_enable( Board_ID, 1 ); //Enable the interrupt main switch
//Reset interrupt status of the axis
```

APS_reset_field_bus_int_motion (Axis_ID);

.. Wait event

See also:

APS_int_enable();APS_set_field_bus_int_factor_motion();APS_get_field_bus_int_factor_motion()

APS_wait_field_bus_error_int_motion	Wait error interrupt event for MotionNet series on PCI(e)-7856.

Support Products : PCI(e)-7856, MNET-4XMO-(C), MNET-1XMO

Descriptions :

This function is used to wait error interrupt event.

When the user enabled the interrupt function by “APS_int_enable()”, users could use this function to wait error interrupts. When this function was running, the process would never stop until the event was be triggered or the function was time out. This function returns when one of the following occurs:

1. Any one of the error interrupt factors is in the signaled state.
2. The time-out interval elapses.

This function checks the current state of the error interrupt factors. If the state is non-signaled, the calling thread enters the wait state. It uses no processor time while waiting for the INT state to become signaled or the time-out interval to elapse.

If any one of the error interrupts is triggered, the event will be automatically reset by system.

The MotionNet error interrupt factor definition, please refer to the interrupt factor table.

Note that all default error factors are turned on.

Note that “APS_set_field_bus_int_factor_error( )” could turn off the error interrupt factor bit.

Syntax:

C/C++:

I32 FNTYPE APS_wait_field_bus_error_int_motion(I32 Axis_ID, I32 Time_Out);

Visual Basic:

APS_wait_field_bus_error_int_motion (ByVal Axis_ID As Long, ByVal Time_Out As Long) As Long

Parameters:

I32 Axis_ID: Specialized axis of MNET system.

I32 Time_Out: Wait timeout time. Unit is milli-second. If value is set -1, the function’s time-out interval never elapses (infinite).

Return Values:

Postive value: The events are wait success.

The return value indicates the index of the error events that satisfied the wait. If more than one event became operation during the call, this is the array index of the signaled events with the smallest index value of all the signaled events.

Negative value: I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 ReturnCode;

I32 Time_Out = 1000;(means 1000 ms)

ReturnCode = APS_wait_field_bus_error_int_motion (Axis_ID, Time_Out);

…

See also:

APS_int_enable();APS_set_field_bus_int_factor_error();APS_get_field_bus_int_factor_error();

APS_reset_field_bus_int_motion()

APS_set_field_bus_int_factor_di	Assign DI interrupt bits and get interrupt handle.

Support Products: PCI(e)-7856

Descriptions:

This function is used to assign the HSL DI interrupt bits and return an I32 type event number for a HSL DI module. When the states of bits assigned are changed(no matter 1 to 0, or 0 to 1), you can wait the interrupt event via the event number. The event number is unique in one system but it is not an event handler. It is just a virtual number of event APS converts. Please note that one DIO module has only one event number.

Syntax:

C/C++:

APS_set_field_bus_int_factor_di (I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 bitsOfCheck);

Visual Basic:

APS_set_field_bus_int_factor_di (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal bitsOfCheck As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number. Value: 0~1, In PCI(e)-7856, this value must be 0.

I32 MOD_No: The first id occupied by HSL slave module. It can't be 0.

I32 bitsOfCheck: This parameter is used with bit-formated. This operation assigns the bits which can cause di-interrupt in a slave module. If slave module has more than 16 bits input, the high word is for bit16~31 and low word is for bit0~15.

Please note that the next bitsOfCheck override the previous bitsOfCheck.

Return Values:

Return positive value: I32 Interrupt event number.

Return negative value: I32 Error code: Please refer to APS Functions Return Code.

Negative value: I32 Error code: Please refer to APS Functions Return Code.

Example:

In following case, thedi interrupt occurs when any of bits on the DI32 slave module are changed. The module occupies id 1.

```
I32 Module_No = 1;
I32 BUS_No = 0;
I32 IntNo; //int number
I32 returnCode; // function return code
```

```
I32 bitsOfCheck = 0xffffffff;
//1. Enable int
APS_int_enable( Board_ID, Enable );

//2. Interrupt factor setting
IntNo = APS_set_field_bus_int_factor_di ( Board_ID, BUS_No, MOD_No, bitsOfCheck );

//3. Wait int
returnCode = APS_wait_single_int( IntNo, 10000 ); //Wait for 10 sec.
If( ret == 0 ) //receive interrupt
{
        ….// do something
}
//clear int
APS_reset_int( IntNo );
```

See also:

APS_int_enable();APS_get_field_bus_int_factor_di();APS_wait_single_int();
APS_wait_multiple_int(); APS_reset_int(); APS_set_int()

APS_get_field_bus_int_factor_di	Get DI interrupt bits assigned

Support Products: PCI(e)-7856

Descriptions:

This function is used to get the setting of DI interrupt bits.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_int_factor_di(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 *bitsOfCheck);

Visual Basic:

APS_get_field_bus_int_factor_di (ByVal Board_ID As Long, ByVal BUS_No, ByVal MOD_No As Long, ByRef bitsOfCheck As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number. Value: 0~1, In PCI(e)-7856, this value must be 0.

I32 MOD_No: The first id occupied by HSL slave module. It can't be 0.

I32 *bitsOfCheck: Return di interrupt bits.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ReturnCode;
I32 bitsOfCheck;
ReturnCode = APS_get_field_bus_int_factor_di ( Board_ID,    BUS_No, MOD_No, &bitsOfCheck);
…
```

See also:

APS_int_enable();APS_set_field_bus_int_factor_di();APS_wait_single_int(); APS_wait_multiple_int(); APS_reset_int(); APS_set_int()

12. Sampling

APS_set_sampling_param	Set sampling parameter.

Support Products: PCI-8253/56, PCI-8392(H), MNET-4XMO , PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set sampling parameters such as sampling rate, sampling channel source and so on. Please refer to the sampling parameters table for the definition and detail descriptions.

On PCI-8253/56 and PCI-8392(H) and PCI-8254/58 / AMP-204/8C and PCIe-833x, sampling function is only for the boards have DSP or CPU inside. It is for real-time issue. The sampling functions garantees each sampled point are record under hard realtime environment.

On MNET-4XMO, sampling function is based on the system timer. So, the system state would affect the accuracy of sampling data. According to our test, the higher sampling rate you set the worse accuracy you get.

Syntax:

C/C++:

I32 FNTYPE APS_set_sampling_param(I32 Board_ID, I32 Param_No, I32 Param_Dat);

Visual Basic:

APS_set_sampling_param(ByVal Board_ID As Long, ByVal ParaNum As Long, ByVal ParaDat As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Param_No: Specified sampling parameter number, refer to sampling parameters table for definition.

I32 Param_Dat: The corresponding parameter value of sampling number. Refer to the sampling table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

//... initial card.

```
I32 Ret = APS_set_sampling_param( Board_ID, SAMP_PA_RATE, 2 ); //Set sampling rate
…
```

See also:

APS_get_sampling_param();APS_wait_trigger_sampling()

APS_get_sampling_param	Get sampling parameter.

Support Products: PCI-8253/56, PCI-8392(H), MNET-4XMO, PCI-8254/58 / AMP-204/8C, PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get sampling parameters such as sampling rate, sampling channel source and so on. Please refer to the sampling parameters table for the definition and detail descriptions.

Syntax:

C/C++:

I32 FNTYPE APS_get_sampling_param(I32 Board_ID, I32 ParaNum, I32 *ParaDat);

Visual Basic:

APS_get_sampling_param(ByVal Board_ID As Long, ByVal ParaNum As Long, ParaDat As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 ParaNum: Sampling parameter number. Refer to the sampling parameters table.

I32 *ParaDat: Return sampling parameter value. Refer to the sampling parameters table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 ParaDat:

Ret = APS_set_sampling_param(Board_ID, SAMP_PA_EDGE, & ParaDat); //Get trigger edge

…

See also:

APS_set_sampling_param();APS_wait_trigger_sampling()

APS_wait_trigger_sampling	Waiting for sample data.

Support Products: PCI-8253/56, PCI-8392(H), MNET-4XMO, PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to sample data from controller. When the function is issued, the program stating to sample the information and put the data to the internal buffer. Until the trigger signal is turned on, program fetched a mass of data which size is pre-trigger length from internal buffer to the user's data buffer and continuous sample the data until reach the length that users designated. In other hand, if the timeout time is reached and the trigger signal does not raised, this function will be timeout and return an error message.

Use **APS_stop_wait_sampling** to forced stop the wait sampling

.

Caution:

APS_wait_trigger_sampling and **APS_wait_trigger_sampling_async** and **APS_auto_sampling** functions cannot be used at the same time.

Syntax:

C/C++:

I32 FNTYPE APS_wait_trigger_sampling(I32 Board_ID, I32 Length, I32 PreTrgLen, I32 TimeOutMs, STR_SAMP_DATA_4CH *DataArr);

Visual Basic:

APS_wait_trigger_sampling(ByValBoard_ID As Long, ByVal Length As Long, ByVal PreTrgLen As Long, ByVal TimeOutMs As Long, DataArr As STR_SAMP_DATA_4CH) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Length: The number of sampling data. (array size)

I32 PreTrgLen: Pre-trigger length.

I32 TimeOutMs: Timeout time. Unit is millisecond.

STR_SAMP_DATA_4CH *DataArr: Get sampling data structure array. Array size must be larger than the parameter "Length".

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
//... initial card.
APS_set_sampling_param( Board_ID, SAMP_PA_RATE, 2    ); //Set sampling rate
APS_set_sampling_param( Board_ID, SAMP_PA_EDGE, 0    ); //Set trigger edge (rising edge)
APS_set_sampling_param( Board_ID, SAMP_PA_LEVEL, 1    ); //Set trigger level ( 1)
APS_set_sampling_param( Board_ID, SAMP_PA_TRIGCH, 0    ); //Set trigger channel
(channel 0)
APS_set_sampling_param( Board_ID, SAMP_PA_SRC_CH0, SAMP_CMD_VEL ); //Set
channel_0 sampling source.
APS_set_sampling_param( Board_ID, SAMP_PA_SRC_CH1, SAMP_MIO_INP    ); //Set
channel_1 sampling source.
I32 Length = 1024; //Total sampling data array size.
I32 PreTrgLen = 100; //The number of pre-trigger points
STR_SAMP_DATA_4CH DataArr[1024];
I32 TimeOutMs = 10000; //10 second timeout
Ret =APS_wait_trigger_sampling( Board_ID, Length, PreTrgLen, TimeOutMs, DataArr );
If( Ret == ERR_NoError )
{ //Sampling successed
      // DataArr are ready to used.
}
```

See also:

APS_set_sampling_param(); APS_get_sampling_param(); APS_stop_wait_sampling()

APS_wait_trigger_sampling_async	Waiting for sample data asynchronously

Support Products: PCI-8253/56, PCI-8392(H), MNET-4XMO, PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to sample data from controller. This function will return immediately. And create a background thread to sampling the data.

Use **APS_get_sampling_count** function to get the count of data be sampled. When the sampled count reachs data **length**, it means sampling finish. If sample count = -1, it means wait failed.

Use **APS_stop_wait_sampling** to forced stop the asynchronous wait sampling. The sampling count than will become -1.

Caution:

APS_wait_trigger_sampling and **APS_wait_trigger_sampling_async** and **APS_auto_sampling** functions cannot be used at the same time.

Syntax:

C/C++:

I32 FNTYPE APS_wait_trigger_sampling_async(I32 Board_ID, I32 Length, I32 PreTrgLen, I32 TimeOutMs, STR_SAMP_DATA_4CH *DataArr);

Visual Basic:

APS_wait_trigger_sampling_async(ByVal Board_ID As Long, ByVal Length As Long, ByVal PreTrgLen As Long, ByVal TimeOutMs As Long, DataArr As STR_SAMP_DATA_4CH)As Long

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 Length: The number of sampling data. (array size)

I32 PreTrgLen: Pre-trigger length.

I32 TimeOutMs: Timeout time. Unit is millisecond.

STR_SAMP_DATA_4CH *DataArr: Get sampling data structure array. Array size must be larger than the parameter “Length”.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
//... initial card.

APS_set_sampling_param( Board_ID, SAMP_PA_RATE, 2    ); //Set sampling rate
APS_set_sampling_param( Board_ID, SAMP_PA_EDGE, 0    ); //Set trigger edge (rising edge)
APS_set_sampling_param( Board_ID, SAMP_PA_LEVEL, 1    ); //Set trigger level ( 1)
APS_set_sampling_param( Board_ID, SAMP_PA_TRIGCH, 0    ); //Set trigger channel
(channel 0)
APS_set_sampling_param( Board_ID, SAMP_PA_SRC_CH0, SAMP_CMD_VEL ); //Set
channel_0 sampling source.
APS_set_sampling_param( Board_ID, SAMP_PA_SRC_CH1, SAMP_MIO_INP    ); //Set
channel_1 sampling source.

//Start a asynchronous wait sampling.
I32 Length = 1024; //Total sampling data array size.
I32 PreTrgLen = 100; //The number of pre-trigger points
STR_SAMP_DATA_4CH DataArr[1024];
I32 TimeOutMs = 10000; //10 second timeout
I32 Ret;

Ret =APS_wait_trigger_sampling_async( Board_ID, Length, PreTrgLen, TimeOutMs,
DataArr );

if( Ret != ERR_NoError )
{
      //Show error message
}else
{
      while( count < Length )
      {
            APS_get_sampling_count( Board_ID, &count );
            If( count == -1 )
            {
                  //Sampling failed,
                  // Break program.;
            }

            If( ForceStop )
            {
                  APS_stop_wait_sampling(Board_ID);
            }
      }
      If( count == Length )
      {     //Sampling successed
            // DataArr are ready to used.
      }
}
```

See also:

APS_get_sampling_count(); APS_wait_trigger_sampling(); APS_stop_wait_sampling()

APS_get_sampling_count	Get sampled data count.

Support Products: PCI-8253/56, PCI-8392(H), MNET-4XMO, PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get asynchronous wait sampling dat count.

In the first way, start a trigger sampling operation using **APS_wait_trigger_sampling_async**, user need to get sampling count to check the operation is finish success or failed.

In the second way, start a sampling operation using **APS_auto_sampling**, user could get sampling count.

Syntax:

C/C++:

I32 FNTYPE APS_get_sampling_count(I32 Board_ID, I32 *SampCnt);

Visual Basic:

APS_get_sampling_count(ByVal Board_ID As Long, SampCnt As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 *SampCnt: Return sampled data count. If return -1 mean sampling failed.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Refer to **APS_wait_trigger_sampling_async** example.

See also:

APS_set_sampling_param();APS_get_sampling_param();APS_stop_wait_sampling();

APS_wait_trigger_sampling(); APS_wait_trigger_sampling_async()

APS_stop_wait_sampling	Force stop wait sampling

Support Products: PCI-8253/56, PCI-8392(H), MNET-4XMO, PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to forced stop **APS_wait_trigger_sampling** and **APS_wait_trigger_sampling_asnyc** function.

Syntax:

C/C++:

I32 FNTYPE APS_stop_wait_sampling(I32 Board_ID);

Visual Basic:

APS_stop_wait_sampling(ByVal Board_ID As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Refer to **APS_wait_trigger_sampling_asnyc** example

See also:

APS_wait_trigger_sampling(); APS_wait_trigger_sampling_async()

APS_auto_sampling	Start/Stop auto sampling

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to implement auto sampling operation. It creates a background thread to sample data.

User could use **APS_get_sampling_data** / **APS_get_sampling_data_ex** to get sampling data and monitor internal buffer status. Four states of the buffer could be monitored, that includes “STOP”, “WORK”, “EMPTY” and “FULL” states.

Use **APS_get_sampling_count** function to get the count of data be sampled. If sample count = -1, it means that auto sampling has already stopped.

After enabling, a thread of sampling data start to run and an internal buffer is also built to store sampling data. This buffer is finite space including up to 65535 sampling data, user needs to continuously get sampling data using **APS_get_sampling_data()**. It is necessary to continuously consume those data of the buffer, and the buffer could be reused to store more sampling data.

Generally speaking, the buffer is always in WORK state. It means that it is impossible to miss any sampling data and the polling frequency of getting data in user side is just suitable.

There is a chance of losing sampling data if the internal buffer is full of data, sampled from DSP side. In FULL state, for example, the sequential sampling data from DSP may be thrown until user gets buffer data to consume parts of data of the buffer.

On the other hand, if the internal buffer is in EMPTY status, it means that getting data form the buffer is faster than sampling from DSP side. User could eliminate frequency of getting data, changing polling timer to slower one. It would increase your CPU performance to do other things.

Caution: These is a set of API functions for auto sampling, including **APS_auto_sampling() and APS_get_sampling_data() / APS_get_sampling_data_ex()**. Don’t mix with other trigger functions like **APS_wait_trigger_sampling()**, **APS_wait_trigger_sampling_async() and APS_stop_wait_sampling().**

Syntax:

C/C++:

I32 FNTYPE APS_auto_sampling(I32 Board_ID, I32 StartStop);

Visual Basic:

APS_auto_sampling (ByVal Board_ID As Long, ByVal StartStop As Long)As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 StartStop: 1: Start auto sampling, 0: Stop auto sampling.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
APS_set_sampling_param( Board_ID, SAMP_PA_RATE, 1 ); //Set sampling rate
APS_set_sampling_param( Board_ID, SAMP_PA_SRC_CH0, SAMP_CMD_VEL ); //Set
channel_0 sampling source.
APS_set_sampling_param( Board_ID, SAMP_PA_SRC_CH1, SAMP_MIO_INP ); //Set
channel_1 sampling source.

//Start auto sampling.
STR_SAMP_DATA_4CH DataArr[500]; //Be the same size with length
I32 ret;
I32 length = 500; //User specifies a length to get data
I32 retLength = 0; //Physical length of returned data
I32 status; //Monitor buffer state
I32 start = 1;

APS_auto_sampling( Board_ID, start ); //Auto sampling start processing

timer( 10ms )
{
   if( start == 1 )
   {
      Length = 500; //User specifies a length to get data
      APS_get_sampling_data(Board_ID, &length, DataArr, & status ); //return physical length
      // Monitor buffer status
      if(status == 1 ) //buffer is in "WORK" state
      {
         //get data – returned length depends on remain data in buffer
      }
      else if(status == 2 ) //buffer is full
      {
         //get data
         //Sampling data may be lost
      }
      else if(status == 3) //buffer is empty
      {
         // get no data – returned length is 0
      }

      For( i=0; i<length; i++ )
      {
         // DataArr are ready to used.
      }
   }
}
```

APS_auto_sampling(Board_ID, 0); //Stop auto sampling

See also:

APS_get_sampling_data(); APS_get_sampling_count()

APS_get_sampling_data	Get sampling data in auto sampling mode

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to sample data after starting auto sampling. It is also used to monitor sampling status. Refer to **Aps_auto_sampling()** for details. Four states are defined below:

State	Description
STOP (0)	Auto sampling stopped
WORK (1)	Auto sampling started
FULL (2)	The internal buffer is full. It is a caution for lost sampling data. Because buffer is full of data, newer sampling data are automatically thrown until user consumes parts of data from buffer.
EMPTY (3)	The internal buffer is empty. Returned length must be zero and get no data

Caution: These is a set of API functions for auto sampling, including **Aps_auto_sampling() and APS_get_sampling_data()**. Don't mix with other trigger functions like **APS_wait_trigger_sampling**, **APS_wait_trigger_sampling_async and APS_stop_wait_sampling.**

Syntax:

C/C++:

I32 FNTYPE APS_get_sampling_data(I32 Board_ID, I32 *Length, STR_SAMP_DATA_4CH *DataArr, I32 *Status);

Visual Basic:

APS_get_sampling_data(ByVal Board_ID As Long, Length As Long, DataArr As STR_SAMP_DATA_4CH, Status As Long)As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 *Length: Bi-direction. User need to specify a maximum size to get data, usually the same with array size of "DataArr". Returned length is physical size of get back sampling data.

STR_SAMP_DATA_4CH *DataArr: Get sampling data structure array. Array size must be equal with or larger than the parameter "*Length".

I32 *Status: The buffer state. 0: STOP state, 1: WORK state, 2: EMPTY state, 3: FULL state.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
//... initial card.

APS_set_sampling_param( Board_ID, SAMP_PA_RATE, 1 ); //Set sampling rate
APS_set_sampling_param( Board_ID, SAMP_PA_SRC_CH0, SAMP_CMD_VEL ); //Set
channel_0 sampling source.
APS_set_sampling_param( Board_ID, SAMP_PA_SRC_CH1, SAMP_MIO_INP ); //Set
channel_1 sampling source.

//Start auto sampling.
STR_SAMP_DATA_4CH DataArr[500]; //Be the same size with length
I32 ret;
I32 length = 500; //User specifies a length to get data
I32 retLength = 0; //Physical length of returned data
I32 status; //Monitor buffer state
I32 start = 1;

APS_auto_sampling( Board_ID, start ); //Auto sampling start processing

Timer( 10ms )
{
  If( start == 1 )
  {
     Length = 500; //User specifies a length to get data
     APS_get_sampling_data(Board_ID, &length, DataArr, & status ); //return physical length
                                                                   //Monitor buffer status
     If(status == 1 ) //buffer is in “WORK” state
     {
       //get data – returned length depends on remain data in buffer
     }
     Else if(status == 2 ) //buffer is full
     {
       //get data
       //some sampling data may be lost
     }
     Else if(status == 3) //buffer is empty
     {
       // get no data – returned length is 0
     }

     For( i=0; i<length; i++ )
     {
       // DataArr are ready to used.
     }
  }
}
APS_auto_sampling( Board_ID, 0 ); //Stop auto sampling
```

See also:

APS_auto_sampling(); APS_get_sampling_count()

APS_set_sampling_param_ex	Set sampling parameter. It is an extension to 8 channels.

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set sampling parameters at once such as sampling rate, sampling channel source and so on. The related parameters of 8 channels are structured by SAMP_PARAM. There are common settings including sampling rate, sampling edge, sampling level and sampling channel in SAMP_PARAM structure. There are also other settings by channel, including sampling source & axis in SAMP_PARAM structure.

Sampling function is only for the boards have DSP inside. It is for real-time issue. The sampling functions guarantees each sampled point are record under hard realtime environment.

Syntax:

C/C++:

I32 FNTYPE APS_set_sampling_param_ex(I32 Board_ID, SAMP_PARAM *Param);

Visual Basic:

APS_set_sampling_param_ex (ByVal Board_ID As Long, ByRef Param As SAMP_PARAM) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

SAMP_PARAM *Param: A structure for setting sampling parameters

```
typedef struct _SAMP_PARAM
{
   I32 rate;        //Sampling rate [ 1 ~ 65535 times of cycle]
   I32 edge;              //Trigger edge [ 0: Rising edge, 1: Faling edge ]
   I32 level;             //Trigger level [ -214743648 ~ 2147483647 ]
   I32 trigCh;      //Trigger channel [ 0 ~ 7 ]
   I32 sourceByCh[8][2];    //Sampling source by channel, named sourceByCh[a][b],
          //a: Specify a Channel. Total channels are 8 to be configured.
          //b: 0: Sampling source, refer to following table 1: Sampling axis
          //Sampling source: F64 data occupies two channels, I32 data occupies one channel.
}
SAMP_PARAM, *PSAMP_PARAM;
```

Source	Symbol Define	Data type	Value range
0x00	SAMP_SRC_COM_POS	I32	command position
0x01	SAMP_SRC_FBK_POS	I32	feedback position
0x02	SAMP_SRC_CMD_VEL	I32	command velocity
0x03	SAMP_SRC_FBK_VEL	I32	feedback velocity
0x04	SAMP_SRC_MIO	I32	motion IO
0x05	SAMP_SRC_MSTS	I32	motion status
0x06	SAMP_SRC_MSTS_ACC	I32	motion status acc
0x07	SAMP_SRC_MSTS_MV	I32	motion status at max velocity
0x08	SAMP_SRC_MSTS_DEC	I32	motion status at dec
0x09	SAMP_SRC_MSTS_CSTP	I32	motion status CSTP
0x0A	SAMP_SRC_MSTS_MDN	I32	motion status MDN
0x0B	SAMP_SRC_MIO_INP	I32	motion status INP
0x0D	SAMP_SRC_MIO_ORG	I32	motion status OGR
0x20	SAMP_SRC_CONTROL_VOL	I32	Control command voltage
0x22	SAMP_SRC_ENCODER_RAW	I32	Encoder raw data
0x23	SAMP_SRC_ERR_POS	I32	Error position
0x10	SAMP_SRC_COM_POS_F64	F64	Command position
0x11	SAMP_SRC_FBK_POS_F64	F64	Feedback position
0x12	SAMP_SRC_CMD_VEL_F64	F64	Command velocity
0x13	SAMP_SRC_FBK_VEL_F64	F64	Feedback velocity
0x14	SAMP_SRC_CONTROL_VOL_F64	F64	Control command voltage
0x15	SAMP_SRC_ERR_POS_F64	F64	Error position

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
SAMP_PARAM Param;
Param.rate = 1; // Sampling rate
Param.edge = 0; // Rising edge
Param.level = 1000; // Trigger level
Param.trigCh = 0; //Channel 0
Param.sourceByCh[0][0] = 1; //Set axis 1 to sampling source of channel 0
Param.sourceByCh[0][1] = 0; //Set command position(I32) to sampling source of channel 0
Param.sourceByCh[1][0] = 0; //Set axis 0 to sampling source of channel 1
Param.sourceByCh[1][1] = 1; //Set feedback position(I32) to sampling source of channel 1
//….set other channels including channel 0 to channel 7

I32 Ret = APS_set_sampling_param_ex( Board_ID, &Param ); //Set sampling parameters
…
```

See also:

APS_get_sampling_param_ex(); APS_wait_trigger_sampling_ex()

APS_get_sampling_param_ex	Get sampling parameter. It is an extension to 8 channels.

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get sampling parameters at once such as sampling rate, sampling channel source and so on. Refer to APS_set_sampling_param_ex().

Syntax:

C/C++:

I32 FNTYPE APS_get_sampling_param_ex(I32 Board_ID, SAMP_PARAM *Param);

Visual Basic:

APS_get_sampling_param_ex (ByVal Board_ID As Long, ByRef Param As SAMP_PARAM) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

SAMP_PARAM *Param: A structure for setting sampling parameters

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
SAMP_PARAM Param;

Ret = APS_get_sampling_param_ex( Board_ID, &Param); //Get all paramters
…
```

See also:

APS_set_sampling_param_ex();APS_wait_trigger_sampling_ex()

APS_wait_trigger_sampling_ex	Waiting for sample data. It is an extension to 8 channels.

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to sample data from controller. When the function is issued, the program stating to sample the information and put the data to the internal buffer. Until the trigger signal is turned on, program fetched a mass of data which size is pre-trigger length from internal buffer to the user's data buffer and continuous sample the data until reach the length that users designated. In other hand, if the timeout time is reached and the trigger signal does not raised, this function will be timeout and return an error message.

Use **APS_stop_wait_sampling** to forced stop the wait sampling.

Caution:

APS_wait_trigger_sampling_ex, **APS_wait_trigger_sampling_async_ex and APS_auto_sampling_ex** functions cannot be used at the same time.

Syntax:

C/C++:

I32 FNTYPE APS_wait_trigger_sampling_ex(I32 Board_ID, I32 Length, I32 PreTrgLen, I32 TimeOutMs, STR_SAMP_DATA_8CH *DataArr);

Visual Basic:

APS_wait_trigger_sampling_ex(ByValBoard_ID As Long, ByVal Length As Long, ByVal PreTrgLen As Long, ByVal TimeOutMs As Long, DataArr As STR_SAMP_DATA_8CH) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Length: The number of sampling data. (array size)

I32 PreTrgLen: Pre-trigger length.

I32 TimeOutMs: Timeout time. Unit is millisecond.

STR_SAMP_DATA_8CH *DataArr: Get sampling data structure array. Array size must be larger than the parameter "Length".

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
//... initial card.
SAMP_PARAM Param;

Param.rate = 1; // Sampling rate
Param.edge = 0; // Rising edge
Param.level = 1000; // Trigger level
Param.trigCh = 0; //Channel 0
Param.sourceByCh[0][0] = 1; //Set axis 1 to sampling source of channel 0
Param.sourceByCh[0][1] = 0; //Set command position(I32) to sampling source of channel 0
Param.sourceByCh[1][0] = 0; //Set axis 0 to sampling source of channel 1
Param.sourceByCh[1][1] = 1; //Set feedback position(I32) to sampling source of channel 1
//….set other channels including channel 0 to channel 7

I32 Ret = APS_set_sampling_param_ex( Board_ID, &Param ); //Set sampling parameters

I32 Length = 1024; //Total sampling data array size.
I32 PreTrgLen = 100; //The number of pre-trigger points
STR_SAMP_DATA_8CH DataArr[1024];
I32 TimeOutMs = 10000; //10 second timeout

Ret =APS_wait_trigger_sampling_ex( Board_ID, Length, PreTrgLen, TimeOutMs, &DataArr );
If( Ret == ERR_NoError )
{ //Sampling successed
      // DataArr are ready to used.
}
```

See also:

APS_set_sampling_param_ex(); APS_get_sampling_param_ex();
APS_stop_wait_sampling_ex()

APS_wait_trigger_sampling_async_ex	Waiting for sample data asynchronously. It is an extension to 8 channels.

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to sample data from controller. This function will return immediately. And create a background thread to sampling the data.

Use **APS_get_sampling_count** function to get the count of data be sampled. When the sampled count reachs data **length**, it means sampling finish. If sample count = -1, it means wait failed.

Use **APS_stop_wait_sampling** to forced stop the asynchronous wait sampling. The sampling count than will become -1.

Caution:

APS_wait_trigger_sampling_ex, **APS_wait_trigger_sampling_async_ex and APS_auto_sampling_ex** functions cannot be used at the same time.

Syntax:

C/C++:

I32 FNTYPE APS_wait_trigger_sampling_async_ex(I32 Board_ID, I32 Length, I32 PreTrgLen, I32 TimeOutMs, STR_SAMP_DATA_8CH *DataArr);

Visual Basic:

APS_wait_trigger_sampling_async_ex(ByVal Board_ID As Long, ByVal Length As Long, ByVal PreTrgLen As Long, ByVal TimeOutMs As Long, DataArr As STR_SAMP_DATA_8CH)As Long

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 Length: The number of sampling data. (array size)

I32 PreTrgLen: Pre-trigger length.

I32 TimeOutMs: Timeout time. Unit is millisecond.

STR_SAMP_DATA_8CH *DataArr: Get sampling data structure array. Array size must be larger than the parameter “Length”.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
//... initial card.
SAMP_PARAM Param;


Param.rate = 1; // Sampling rate
Param.edge = 0; // Rising edge
Param.level = 1000; // Trigger level
Param.trigCh = 0; //Channel 0
Param.sourceByCh[0][0] = 1; //Set axis 1 to sampling source of channel 0
Param.sourceByCh[0][1] = 0; //Set command position(I32) to sampling source of channel 0
Param.sourceByCh[1][0] = 0; //Set axis 0 to sampling source of channel 1
Param.sourceByCh[1][1] = 1; //Set feedback position(I32) to sampling source of channel 1
//….set other channels including channel 0 to channel 7


I32 Ret = APS_set_sampling_param_ex( Board_ID, &Param ); //Set sampling parameters

//Start a asynchronous wait sampling.
I32 Length = 1024; //Total sampling data array size.
I32 PreTrgLen = 100; //The number of pre-trigger points
STR_SAMP_DATA_8CH DataArr[1024];
I32 TimeOutMs = 10000; //10 second timeout
I32 Ret;

Ret =APS_wait_trigger_sampling_async_ex( Board_ID, Length, PreTrgLen, TimeOutMs,
DataArr );

if( Ret != ERR_NoError )
{
	//Show error message
}else
{
	while( count < Length )
	{
		APS_get_sampling_count( Board_ID, &count );
		If( count == -1 )
		{
			//Sampling failed,
			// Break program.;
		}

		If( ForceStop )
		{
			APS_stop_wait_sampling(Board_ID);
		}
	}
	If( count == Length )
	{ //Sampling successed
		// DataArr are ready to used.
	}
}
```

See also:

APS_get_sampling_count(); APS_wait_trigger_sampling_ex(); APS_stop_wait_sampling

APS_get_sampling_data_ex	Get sampling data in auto sampling mode. It is an extension to 8 channels.

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to sample data after starting auto sampling. It is also used to monitor sampling status. Refer to **APS_auto_sampling()** for details. Four states are defined below:

State	Description
STOP (0)	Auto sampling stopped
WORK (1)	Auto sampling started
EMPTY (2)	The internal buffer is empty. Returned length must be zero and get no data
FULL (3)	The internal buffer is full. It is a caution for lost sampling data. Because buffer is full of data, newer sampling data are automatically thrown until user consumes parts of data from buffer.

Caution: These is a set of API functions for auto sampling, including **APS_auto_sampling() and APS_get_sampling_data_ex()**. Don't mix with other trigger functions like **APS_wait_trigger_sampling_ex**, **APS_wait_trigger_sampling_async_ex and APS_stop_wait_sampling.**

Syntax:

C/C++:

I32 FNTYPE APS_get_sampling_data_ex(I32 Board_ID, I32 *Length,
STR_SAMP_DATA_8CH *DataArr, I32 *Status);

Visual Basic:

APS_get_sampling_data_ex(ByVal Board_ID As Long, Length As Long, DataArr As STR_SAMP_DATA_8CH, Status As Long)As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 *Length: Bi-direction. User need to specify a maximum size to get data, usually the same with array size of "DataArr". Returned length is physical size of get back sampling data.

STR_SAMP_DATA_8CH *DataArr: Get sampling data structure array. Array size must be equal with or larger than the parameter “*Length”.

I32 *Status: The buffer state. 0: STOP state, 1: WORK state, 2: EMPTY state, 3: FULL state.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
//... initial card.
SAMP_PARAM Param;

Param.rate = 1; // Sampling rate
Param.edge = 0; // Rising edge
Param.level = 1000; // Trigger level
Param.trigCh = 0; //Channel 0
Param.sourceByCh[0][0] = 1; //Set axis 1 to sampling source of channel 0
Param.sourceByCh[0][1] = 0; //Set command position(I32) to sampling source of channel 0
Param.sourceByCh[1][0] = 0; //Set axis 0 to sampling source of channel 1
Param.sourceByCh[1][1] = 1; //Set feedback position(I32) to sampling source of channel 1
//….set other channels including channel 0 to channel 7

I32 Ret = APS_set_sampling_param_ex( Board_ID, &Param ); //Set sampling parameters

//Start auto sampling.
STR_SAMP_DATA_8CH DataArr[500]; //Be the same size with length
I32 ret;
I32 length = 500; //User specifies a length to get data
I32 retLength = 0; //Physical length of returned data
I32 status; //Monitor buffer state
I32 start = 1;

APS_auto_sampling( Board_ID, start ); //Auto sampling start processing

Timer( 10ms )
{
   If( start == 1 )
   {
      Length = 500; //User specifies a length to get data
      APS_get_sampling_data_ex(Board_ID, &length, DataArr, & status ); //return physical
                                                                       length //Monitor buffer
                                                                       status
      If(status == 1 ) //buffer is in “WORK” state
      {
         //get data – returned length depends on remain data in buffer
      }
      Else if(status == 2 ) //buffer is full
```

```
        {
           //get data
           //some sampling data may be lost
        }
        Else if(status == 3) //buffer is empty
        {
           // get no data – returned length is 0
        }

        For( i=0; i<length; i++ )
        {
           // DataArr are ready to used.
        }
    }
}
APS_auto_sampling( Board_ID, 0 ); //Stop auto sampling
```

See also:

APS_auto_sampling(); APS_get_sampling_count()

13. DIO & AIO

APS_set_field_bus_d_channel_output	Set field bus digital output by channel

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is use to set field bus digital output by channel.

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_d_channel_output(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 Ch_No, I32 DO_Value);

Visual Basic:

APS_set_field_bus_d_channel_output (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal Ch_No As Long, ByVal DO_Value As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0)

I32 MOD_No: The index of slave device. (start from 0)

I32 Ch_No: The index of digital output channel. (start from 0)

I32 DO_Value: The value of digital output.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_get_field_bus_d_channel_output	Get field bus digital output by channel

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is use to get field bus digital output by channel.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_d_channel_output(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 Ch_No, I32 *DO_Value);

Visual Basic:

APS_get_field_bus_d_channel_output(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal Ch_No As Long, ByRef DO_Value As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0)

I32 MOD_No: The index of slave device. (start from 0)

I32 Ch_No: The index of digital output channel. (start from 0)

I32 DO_Value: Return the value of digital output.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_get_field_bus_d_channel_input	Get field bus digital input by channel

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is use to get field bus digital input by channel.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_d_channel_input(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 Ch_No, I32 *DI_Value);

Visual Basic:

APS_get_field_bus_d_channel_input (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal Ch_No As Long, ByRef DI_Value As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0)

I32 MOD_No: The index of slave device. (start from 0)

I32 Ch_No: The index of digital output channel. (start from 0)

I32 *DI_Value: Return the value of digital input.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_set_field_bus_d_port_output	Set field bus digital output by port

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is use to set field bus digital output by port

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_d_port_output(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 Port_No, U32 DO_Value);

Visual Basic:

APS_set_field_bus_d_port_output (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal Port_No As Long, ByVal DO_Value As UInteger) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0)

I32 MOD_No: The index of slave device. (start from 0)

I32 Port_No: The index of digital output port. (start from 0)

U32 DO_Value: Set the value of digital output.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_get_field_bus_d_port_input	Get field bus digital input by port

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is use to get field bus digital output by port.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_d_port_input(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 Port_No, U32 *DI_Value);

Visual Basic:

APS_get_field_bus_d_port_input(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal Port_No As Long, ByRef DI_Value As UInteger) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0)

I32 MOD_No: The index of slave device. (start from 0)

I32 Port_No: The index of digital output port. (start from 0)

U32 *DI_Value: Return the value of digital input.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_get_field_bus_d_port_output	Get field bus digital output by port

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is use to get field bus digital output by port

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_d_port_output(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 Port_No, U32 *DO_Value);

Visual Basic:

APS_get_field_bus_d_port_output (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal Port_No As Long, ByRef DO_Value As UInteger) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0)

I32 MOD_No: The index of slave device. (start from 0)

I32 Port_No: The index of digital output port. (start from 0)

U32 *DO_Value: Return the value of digital output.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_write_d_output	Set digital output value

Support Products: PCI-8253/56, DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-8154/8158, PCI-8102/PCI-C154(+), EMX-100, PCI-8254/58 / AMP-204/8C , PCIe-833x, AMP-104C, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is use to access on board general purpose digital output. If the channels are more than 32, users must assign a group number to access more I/O.

The PCI-8256 has 8 (PCI-8253 has 4, DPAC-1000, DPAC-3000 has 4, PCI(e)-8154 has 4, PCI(e)-8158 has 8, PCI-8102 has 2,PCI-C154(+) has 4 multi-function DO) output channels, user can assign group number to be constant 0.

The PCI-8102 has 16 (PCIe-8154/8158, PCI-C154(+) has 16 channel extension DO)output channels, user can assign group number to be constant 1.

The PCI-8254/58 / AMP-204/8C has 24 output channels, user can assign group number to be constant 0.

Do channel defined by bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTL7	TTL6	TTL5	TTL4	TTL3	TTL2	TTL1	TTL0	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								TTL15	TTL14	TTL13	TTL12	TTL11	TTL10	TTL9	TTL8

The AMP-104C has 12 channel with isolated DO, user assign group number to be constant 0. Furthermore the AMP-104C has 15 channel with TTL DO, user assign group number to be constant 1.

Syntax:

C/C++:

I32 FNTYPE APS_write_d_output(I32 Board_ID, I32 DO_Group, I32 DO_Data);

Visual Basic:

APS_write_d_output (ByVal Board_ID As Long, ByVal DO_Grout As Long, ByVal DO_Data as Long) As Long;

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 DO_Group: The digit output group number. (only support group index 0)

I32 DO_Data: The digit output data (Data type is bit type).

For EMX-100:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 DO_Group: The digit output group number. (group index 0 ~ index1)

I32 DO_Data: The digit output data (group 0 data length is 8 bit, group 1 data length is 6 bit)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 DO_Group = 0;          // If DO channel less than 32
I32 DO_Data = 0x000F;      // Assign bit 0,1,2,3 output.
I32 returnCode;            // Function return code

returnCode = APS_write_d_output( Board_ID, DO_Group, DO_Data );
if( returnCode != 0 )
      return MessageBox( "Set digit output function failed" );
```

See also:

APS_read_d_input()

APS_read_d_output	Read digital output value

Support Products: PCI-8253/56, DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-8154/8158, PCI-8102/PCI-C154(+), EMX-100, PCI-8254/58 / AMP-204/8C, PCIe-833x, AMP-104C, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is use to get on board general purpose digital output. If the channels are more than 32, users must assign a group number to access more I/O.

The PCI-8256 has 8 (PCI-8253 has 4, DPAC-1000, DPAC-3000 has 4, PCI(e)-8154 has 4, PCI(e)-8158 has 8, PCI-8102 has 2, PCI-C154(+) has 4 multi-function DO) output channels, user can assign group number to be constant 0.

The PCI-8102 has 16 (PCIe-8154/8158, PCI-C154(+) has 16 channel extension DO) output channels, user can assign group number to be constant 1.

The PCI-8254/58 / AMP-204/8C has 24 output channels, user can assign group number to be constant 0.

Do channel defined by bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTL7	TTL6	TTL5	TTL4	TTL3	TTL2	TTL1	TTL0	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								TTL15	TTL14	TTL13	TTL12	TTL11	TTL10	TTL9	TTL8

The AMP-104C has 12 channel with isolated DO, user can assign group number to be constant 0.

Furthermore the AMP-104C has 15 channel with TTL DO, user can assign group number to be constant 1.

Syntax:

C/C++:

I32 FNTYPE APS_read_d_output(I32 Board_ID, I32 DO_Group, I32 *DO_Data);

Visual Basic:

APS_read_d_output (ByVal Board_ID As Long, ByVal DO_Grout As Long, DO_Data as Long) As Long;

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 DO_Group: The digit output group number. (only support group index 0)

I32 *DO_Data: The digit output data (Data type is bit type).

For EMX-100:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 DO_Group: The digit output group number. (group index 0 ~ index1)

I32 *DO_Data: The digit output data (group 0 data length is 8 bit, group 1 data length is 6 bit).

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_write_d_output()

APS_read_d_input	Read digital input value

Support Products: PCI-8253/56, DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-8154/8158, PCI-8102/PCI-C154(+),EMX-100 , PCI-8254/58 / AMP-204/8C, PCIe-833x, AMP-104C, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is use to get on board general purpose digital input. If the channels are more than 32, users must assign a group number to access more I/O.

The PCI-8256 has 8 (PCI-8253 has 4, DPAC-1000, DPAC-3000 has 4, PCI(e)-8154 has 4, PCI(e)-8158 has 8, PCI-8102 has 4 , PCI-C154(+) has 4 multi-function DO) input channels, user can assign group number to be constant 0.

The PCI-8102 has 16 (PCIe-8154/8158, PCI-C154(+) has 16 channel extension DO) input channels, user can assign group number to be constant 1.

The PCI-8254/58 / AMP-204/8C has 24 input channels, user can assign group number to be constant 0.

Do channel defined by bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTL7	TTL6	TTL5	TTL4	TTL3	TTL2	TTL1	TTL0	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								TTL15	TTL14	TTL13	TTL12	TTL11	TTL10	TTL9	TTL8

The AMP-104C has 16 channel with isolated DI, user can assign group number to be constant 0.

Furthermore the AMP-104C has 16 channel with TTL DI, user can assign group number to be constant 1.

Syntax:

C/C++:

I32 FNTYPE APS_read_d_input(I32 Board_ID, I32 DI_Group, I32 *DI_Data);

Visual Basic:

APS_read_d_input (ByVal Board_ID As Long, ByVal DI_Grout As Long, DI_Data as Long) As Long;

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 DI_Group: The digit input group number. (Only support group index 0)

I32 *DI_Data: The returned digit input data

For EMX-100:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 DI_Group: The digit input group number. (group index 0 ~ index3)

I32 *DI_Data: The returned digit input data (every group Data length is 8 bit)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for PCI-8254/58 / AMP-204/8C

```
I32 DI_Group = 0;            // If DI channel less than 32
I32 DI_Data = 0;                  // Di data
I32 returnCode;              // Function return code

returnCode = APS_read_d_ input( Board_ID, DI_Group, &DI_Data );
if( returnCode != 0 )
      MessageBox( “Get digit input function failed” );
```

See also:

APS_write_d_output()

APS_write_d_channel_output	Set digital output value by channel

Support Products: PCIe-8154/8158, PCI-C154(+), EMX-100, PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is use to access on board general purpose digital output by channel.

The PCIe-8154/8158, PCI-C154(+) has 16 channel extension DO output channels, user can assign group number to be constant 1.

The PCI-8254/58 / AMP-204/8C has 24 output channels in the group number 0.

The definition of Do channels in the group number 0:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Ch 0
TTL7	TTL6	TTL5	TTL4	TTL3	TTL2	TTL1	TTL0	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								TTL15	TTL14	TTL13	TTL12	TTL11	TTL10	TTL9	TTL8

Syntax:

C/C++:

I32 FNTYPE APS_write_d_channel_output(I32 Board_ID, I32 DO_Group, I32 Ch_No, I32 DO_Data);

Visual Basic:

APS_write_d_channel_output (ByVal Board_ID As Long, ByVal DO_Group As Long, ByVal Ch_No As Long, ByVal DO_Data as Long) As Long;

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 DO_Group: The digit output group number.

I32 Ch_No: The digit output channel.

I32 DO_Data: The digit output data (Data type is bit type).

For EMX-100:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 DO_Group: The digit output group number. (only support group index 0)

I32 Ch_No: The digit output channel (range is 0~13)

I32 DO_Data: The digit output data by channel (0~1).

For PCI-8254/58 / AMP-204/8C:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 DO_Group: The digit output group number. Set to 0 on PCI-8254/8.

I32 Ch_No: The digit output channel(0~23)

I32 DO_Data: The digit output data by channel (0~1).

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example1:

Below example is for PCI(e)-8154/58

```
I32 DO_Group = 1;          // If DO channel less than 32
I32 DO_Data =1;
I32 returnCode;            // Function return code
I32 Ch_No = 0;             // Assign bit 0 output.
returnCode = APS_write_d_output( Board_ID, DO_Group, Ch_No, DO_Data );
if( returnCode != 0 )
    return MessageBox( "Set digit output function failed" );
```

Example2:

Below example is for PCI-8254/58 / AMP-204/8C

```
I32 returnCode;            // Function return code
//Turn on Do output channel 3 in group number 0
returnCode = APS_write_d_channel_output( Board_ID, 0, 3, 1 );
if( returnCode != 0 )
    MessageBox( "Set digit channel output function failed" );
```

See also:

APS_read_d_channel_input()

APS_read_d_channel_output	Read digital output value by channel

Support Products: PCIe-8154/8158, PCI-C154(+), EMX-100, PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to access on board general purpose digital output by channel. If those channels are more than 32, users must assign a group number to access more I/O.

The PCIe-8154/8158, PCI-C154(+) has 16 channel extension DO output channels, user can assign group number to be constant 1.

The PCI-8254/58 / AMP-204/8C has 24 output channels in the group number 0.

The definition of Do channels in the group number 0:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Ch 0
TTL7	TTL6	TTL5	TTL4	TTL3	TTL2	TTL1	TTL0	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								TTL15	TTL14	TTL13	TTL12	TTL11	TTL10	TTL9	TTL8

Syntax:

C/C++:

I32 FNTYPE APS_read_d_channel_output(I32 Board_ID, I32 DO_Group, I32 Ch_No, I32 *DO_Data);

Visual Basic:

APS_read_d_channel_output (ByVal Board_ID As Long, ByVal DO_Grout As Long, ByVal Ch_No As Long, DO_Data As Long) As Long;

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 DO_Group: The digit output group number. (only support group index 0)

I32 Ch_No: The digit output channel.

I32 *DO_Data: The digit output data (Data type is bit type).

For EMX-100:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 DO_Group: The digit output group number. (only support group index 0)

I32 Ch_No: The digit output channel (range is 0~13)

I32 *DO_Data: The digit output data (0,1)

For PCI-8254/58 / AMP-204/8C:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 DO_Group: The digit output group number. Set to 0 on PCI-8254/8.

I32 Ch_No: The digit output channel(0~23)

I32 *DO_Data: The digit output data

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for PCI-8254/58 / AMP-204/8C

```
I32 DO_Data = 0;                    // Do data
I32 returnCode;                 // Function return code
//Get status of Do output channel 3 in group number 0
returnCode = APS_read_d_channel_output( Board_ID, 0, 3, &DO_Data );
if( returnCode != 0 )
    MessageBox( "Get digit channel output function failed" );
```

See also:

APS_write_d_channel_output()

APS_read_d_channel_input	Read digital input value by channel

Support Products: PCIe-8154/8158, PCI-C154(+),EMX-100 , PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to access on board general purpose digital input by channel.

The PCIe-8154/8158, PCI-C154(+) has 16 channel extension DI input channels, user can assign group number to be constant 1.

The definition of Do channels in the group number 0:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Ch 0
TTL7	TTL6	TTL5	TTL4	TTL3	TTL2	TTL1	TTL0	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								TTL15	TTL14	TTL13	TTL12	TTL11	TTL10	TTL9	TTL8

Syntax:

C/C++:

I32 FNTYPE APS_read_d_channel_input(I32 Board_ID, I32 DI_Group, I32 Ch_No, I32 *DI_Data);

Visual Basic:

APS_read_d_channel_input (ByVal Board_ID As Long, ByVal DI_Grout As Long, ByVal Ch_No As Long, DI_Data as Long) As Long;

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 DI_Group: The digit input group number. (only support group index 0)

I32 Ch_No: The digit input channel.

I32 *DI_Data: The returned digit input data

For EMX-100:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 DI_Group: The digit output group number. (only support group index 0)

I32 Ch_No: The digit output channel (range is 0~31)

I32 *DI_Data: The digit output data (0,1)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_write_d_channel_output()

APS_read_a_input_value	Read back analog input value by volt

Support Products: PCI-8253/56，PCI-8254/58 / AMP-204/8C, PCIe-833x

Descriptions:

There are two kinds of function for analog input. One is converted data. It could be voltage or current value. The other is raw data. It is relative to bit resolution of hardware design. This function is used to get on board general purpose analog input value of one axis, and the analog input value unit is volt. The conversion is one inside APS library according to hardware specifications and settings.

Notice: AMP series don't support this function.

Syntax:

C/C++:

I32 FNTYPE APS_read_a_input_value(I32 Board_ID, I32 Channel_No, F64 *Convert_Data);

Visual Basic:

APS_read_a_input_value (ByVal Board_ID As Long, ByVal Channel_No As Long,
Convert_Data as Double) As Long;

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Channel_No: The channel number. Range is from 0 to 65535.

F64 *Convert_Data: The returned converted analog data. Unit is volt and range is -10V to 10V.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for PCI-8254/58 / AMP-204/8C

```
I32 Board_ID = 0;
I32 Channel_No = 0;
F64 Convert_Data = 0.0;
I32 returnCode;                 // Function return code

returnCode = APS_read_a_input_value( Board_ID, Channel_No, & Convert_Data );
if( returnCode != 0 )
      MessageBox( “Get analog input function failed” );
```

See also:

APS_read_a_input_data()

APS_read_a_input_data	Read back analog input raw data

Support Products: PCI-8253/56

Descriptions:

There are two kinds of function for analog input. One is converted data. It could be voltage or current value. The other is raw data. It is relative to bit resolution of hardware design. This function is used to get on board general purpose analog input raw data of one axis.

Syntax:

C/C++:

I32 FNTYPE APS_read_a_input_data(I32 Board_ID, I32 Channel_No, I32 *Raw_Data);

Visual Basic:

APS_read_a_input_data (ByVal Board_ID As Long, ByVal Channel_No As Long, Raw_Data as Long) As Long;

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Channel_No: The channel number. Range is from 0 to 65535.

I32 *Raw_Data: The returned raw data of analog channel. Raw data definition:

*Raw_Data = -32768 => its mean -10V

*Raw_Data = 0 => its mean 0V

*Raw_Data = 32767 => its mean 10V

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_read_a_input_value()

APS_write_a_output_value	Set analog output value by volt

Support Products: PCI-8253/56, PCI-8254/58 / AMP-204/8C, PCIe-833x

Descriptions:

There are two kinds of function for analog output. One is converted data. It could be voltage or current value. The other is raw data. It is relative to bit resolution of hardware design.

This function is used to access on board general purpose analog output raw data of one axis and the analog output value unit is volt. Please make sure axis **servo on signal is turn off** relative to channel number before use analog output function.

Notice: AMP series don't support this function.

Syntax:

C/C++:

I32 FNTYPE APS_write_a_output_value(I32 Board_ID, I32 Channel_No, F64 Convert_Data);

Visual Basic:

APS_write_a_output_value (ByVal Board_ID As Long, ByVal Channel_No As Long, ByVal Convert_Data as Double) As Long;

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Channel_No: The channel number. Range is from 0 to 65535.

F64 Convert_Data: The converted analog data to be output. Unit is volt and range is -10V to 10V

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example1:

Below example is for PCI-8253/56

```
I32 Channel_No = 1;          // Assign channel 1 to be output channel
F32 Convert_Data;
I32 returnCode;              // Function return code
While( 1 )
{
    // From -10 …… +10 step 0.1
    Convert_Data = -10.0;
```

```
    do
    {
        APS_write_a_output_value( Board_ID, Channel_No, Convert_Data );
        Sleep(10);
        Convert_Data += 0.1;
    } while( Convert_Data < 10.0 )
}
```

Example2:

Below example is for PCI-8254/58 / AMP-204/8C

```
I32 Channel_No = 1;         // Assign channel 1 to be output channel
F32 Convert_Data = 5.2;     //Output 5.2 volt
I32 returnCode;             // Function return code

returnCode = APS_write_a_output_value( Board_ID, Channel_No, Convert_Data );
if( returnCode != 0 )
    MessageBox( “Write analog output function failed” );
```

See also:

APS_write_a_output_data()

APS_write_a_output_data	Set analog output value by raw data

Support Products: PCI-8253/56

Descriptions:

There are two kinds of function for analog output. One is converted data. It could be voltage or current value. The other is raw data. It is relative to bit resolution of hardware design. This function is used to access on board general purpose analog output raw data of one axis. Please make sure axis **servo on signal is turn off** relative to channel number before use analog output function.

Syntax:

C/C++:

I32 FNTYPE APS_write_a_output_data(I32 Board_ID, I32 Channel_No, I32 Raw_Data);

Visual Basic:

APS_write_a_output_data (ByVal Board_ID As Long, ByVal Channel_No As Long, ByVal Raw_Data as Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Channel_No: The channel number. Range is from 0 to 65535.

I32 Raw_Data: The raw analog data to be output. Raw data definition as below

Raw_Data = -32768 => its mean -10V

Raw_Data = 0 => its mean 0V

Raw_Data = 32767 => its mean 10V

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Channel_No = 2;          // Assign channel 1 to be output channel
I32 Raw_Data;
I32 returnCode;              // Function return code

While( 1 )
{
    // From -10 …… +10 step 1 bit
```

```
	Raw_Data = -32768;
	do
	{
		APS_write_a_output_Raw_Data( Board_ID, Channel_No, Raw_Data );
		Sleep(10);
		Raw_Data += 1;
	} while(Raw_Data < 0x7FFF)
}
```

See also:

APS_write_a_output_value()

14. Point table motion

APS_set_point_table	Set point table move parameters

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to set a set of point table parameters to specified axis. The point table defined in APS is not only a point table but also an instruction table. Users can link a move sequence by using this point table. The sequence can be used to different speed parameters and curve parameters. It can be assigned ending operation for next movement.
The maximum point can be downloaded to on board memory once refers to product specifications. By setting repeat movement, users can make dynamic loading regardless point quatity limitations.

Syntax:

C/C++:
I32 FNTYPE APS_set_point_table(I32 Axis_ID, I32 Index, POINT_DATA *Point);
Visual Basic:
APS_set_point_table(ByVal Axis_ID As Long, ByVal Index As Long, Point As POINT_DATA) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.
I32 Index: Specified point index to be set. Range
POINT_DATA *Point: Structure of point table parameters. Define in “type_def.h”

```
typedef struct
{
    I32 i32_pos;          //(Center)Position data (could be relative or absolute value) (pulse)
    I16 i16_accType;      //Acceleration pattern 0: T curve, 1:S curve
    I16 i16_decType;      // Deceleration pattern 0: T curve, 1:S curve
    I32 i32_acc;          //Acceleration rate ( pulse / sec2)
    I32 i32_dec;          //Deceleration rate ( pulse / sec2 )
    I32 i32_initSpeed;    //Start velocity   ( pulse / s )
    I32 i32_maxSpeed;     //Maximum velocity   ( pulse / s )
    I32 i32_endSpeed;     //End velocity    ( pulse / s )
    I32 i32_angle;        //Arc move angle ( degree, -360 ~ 360 )
```

U32 u32_dwell; //dwell times (unit: ms) ***Divided by system cycle time**.*

I32 i32_opt; //Point move option. **(*)**

} POINT_DATA;

(*) Point move option: ***i32_opt***

7	6	5	4	3	2	1	Bit : 0
-	-	Last point	Finish condition	Table_Ctrl	Linear/Arc	-	Absolute/Relative
15	14	13	12	11	10	9	Bit : 8
Table_No	Table_No	Table_No	Do_Ch	Do_Ch	Do_Ch	Do_OnOff	Do_En

Bit 0: 1:Relative move, 0:Absolute move

Bit 2: 1:Arc move, 0:Linear move

Bit 3: 1: Enable VAO table switching control (when it is enabled, the setting table is effective of bit13 to bit 15), 0: Disable

Bit 4: 1:INP ON(In position signal), 0:CSTP ON(command stop signal)

Bit 5: 1: Last point index. 0: Not Last point index. (if this bit is turned on, point table move will stop after this point.)

Bit 8: 1: Enable Do, 0: Disable Do

Bit 9: 1: Set Do on(set to 1), 0: Set Do off(set to 0)

Bit 10~12: Select a Do channel (0 ~ 7)

Bit 13~15: Select a table number from 0 to 7. It is effective when bit 3 is enabled. When point table is running on this point, it will automatically switch to specified VAO table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

I32 ret;
POINT_DATA Point;

Point.i32_pos = 10000;      //(Center)Position data (could be relative or absolute value) (pulse)
Point.i16_accType = 1;      //Acceleration pattern 0: T curve, 1:S curve
```

```
…
//Set point data to card memory.
Ret = APS_set_point_table(Axis_ID, 0, &Point );
if( ret != ERR_NoError )
{ //Error (C)
}
```

See also:

APS_get_point_table();APS_point_table_move();APS_get_next_point_index();
APS_get_start_point_index();APS_get_end_point_index()

APS_get_point_table	Get point table move parameters

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to get a set of point table parameters to specified axis.

Syntax:

C/C++:

I32 FNTYPE APS_get_point_table(I32 Axis_ID, I32 Index, POINT_DATA *Point);

Visual Basic:

APS_get_point_table(ByVal Axis_ID As Long, ByVal Index As Long, Point As POINT_DATA) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Index: Specified point index to be set. Range

POINT_DATA *Point: Structure of point table parameters. Define in “type_def.h”

```
typedef struct
{
    I32 i32_pos;          //(Center)Position data (could be relative or absolute value) (pulse)
    I16 i16_accType;      //Acceleration pattern 0: T curve, 1:S curve
    I16 i16_decType;      // Deceleration pattern 0: T curve, 1:S curve
    I32 i32_acc;          //Acceleration rate ( pulse / sec2)
    I32 i32_dec;          //Deceleration rate ( pulse / sec2 )
    I32 i32_initSpeed;    //Start velocity   ( pulse / s )
    I32 i32_maxSpeed;     //Maximum velocity   ( pulse / s )
    I32 i32_endSpeed;     //End velocity    ( pulse / s )
    I32 i32_angle;        //Arc move angle ( degree, -360 ~ 360 )
    U32 u32_dwell;        //dwell times ( unit: ms ) * Divided by system cycle time.
    I32 i32_opt;      //Point move option. (*)
} POINT_DATA;
```

(*) Point move option: ***i32_opt***

7	6	5	4	3	2	1	Bit : 0
-	-	Last point	Finish condition	-	Linear/Arc	-	Absolute/Relative
15	14	13	12	11	10	9	Bit : 8

			Do_Ch	Do_Ch	Do_Ch	Do_OnOff	Do_En

Bit 0: 1:Relative move, 0:Absolute move

Bit 2: 1:Arc move, 0:Linear move

Bit 4: 1:INP ON(In position signal), 0:CSTP ON(command stop signal)

Bit 5: 1: Last point index. 0: Not Last point index. (if this bit is turned on, point table move will stop after this point.)

Bit 8: 1: Enable Do, 0: Disable Do

Bit 9: 1: Set Do on(set to 1), 0: Set Do off(set to 0)

Bit 10~12: Do channel(0 ~ 7)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

//... initial card.
I32 ret ;
POINT_DATA Point;
ret =APS_get_point_table( Axis_ID, 0, &Point );
if( ret != ERR_NoError )
{
	//Error.
}
```

See also:

APS_set_point_table();APS_point_table_move();APS_get_next_point_index();
APS_get_start_point_index();APS_get_end_point_index()

APS_set_point_table_ex	Set point table move parameters with entend option

Support Products: PCI-8392(H)

Descriptions:

This function is used to set a set of point table parameters with entend option to specified axis. The point table defined in APS is not only a point table but also an instruction table. Users can link a move sequence by using this point table. The sequence can be used to different speed parameters and curve parameters. It can be assigned ending operation for next movement. The maximum point can be downloaded to on board memory once refers to product specifications. By setting repeat movement, users can make dynamic loading regardless point quatity limitations.

As depicts in APS_set_point_table, linear ad arc move are support. Helical move is additionally support by APS_set_point_table_ex with the extend option.

Multi-dimension move is support by setting entend option, which allows user change move dimension of move in a series of point moves.

Syntax:

C/C++:

I32 FNTYPE APS_set_point_table_ex(I32 Axis_ID, I32 Index, POINT_DATA_EX *Point);

Visual Basic:

APS_set_point_table_ex (ByVal Axis_ID As Integer, ByVal Index As Integer, ByRef Point As POINT_DATA_EX) As Integer

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Index: Specified point index to be set.

POINT_DATA_EX *Point: Structure of point table parameters. Define in “type_def.h”

```
typedef struct
{
  I32 i32_pos;          //(Center)Position data (could be relative or absolute value) (pulse)
  I16 i16_accType;      //Acceleration pattern 0: T curve, 1:S curve
  I16 i16_decType;      // Deceleration pattern 0: T curve, 1:S curve
  I32 i32_acc;          //Acceleration rate ( pulse / sec 2 )
  I32 i32_dec;          //Deceleration rate ( pulse / sec 2  )
  I32 i32_initSpeed;    //Start velocity ( pulse / s )
  I32 i32_maxSpeed;     //Maximum velocity     ( pulse / s )
  I32 i32_endSpeed;     //End velocity  ( pulse / s )
  I32 i32_angle;        //Arc move angle ( degree, -360 ~ 360 )
```

```
    U32 u32_dwell;          //dwell times ( unit: ms ) *Divided by system cycle time.
    I32 i32_opt;            //Point move option. (*)
    I32 i32_pitch;          // pitch for helical move
    I32 i32_totalheight;    // total hight
    I16 i16_cw;             // cw or ccw
    I16 i16_opt_ext;        // option extend (**)
} POINT_DATA_EX;
```

(*) Point move option: ***i32_opt***

7	6	5	4	3	2	1	Bit : 0
-	-	Last point	Finish condition	-	Linear/Arc	-	Absolute/Relative
15	14	13	12	11	10	9	Bit : 8
-	-	-	-	-	-	-	-

Bit 0: 1:Relative move, 0:Absolute move

Bit 2: 1:Arc move, 0:Linear move

Bit 3: 1: Enable VAO table switching control (when it is enabled, the setting table is effective of bit13 to bit 15), 0: Disable

Bit 4: 1:INP ON(In position signal), 0:CSTP ON(command stop signal)

Bit 5: 1: Last point index. 0: Not Last point index. (if this bit is turned on, point table move will stop after this point.)

Bit 8~15: Reserved.

(**) Point move option: ***i16_opt_ext***

7	6	5	4	3	2	1	Bit : 0
u	z	y	x	-	-	-	Helical
15	14	13	12	11	10	9	Bit : 8
-	-	-	-	-	-	-	-

Bit 0: 1: helical move, 0: linear or arc move

If Bit 0 is 1, the motion type is helical move.

If Bit 0 is 0, the motion type is defined by Bit 2 of ***i32_opt.***

Bit 4: 1: 1st axis move, 0: 1st axis not move

Bit 5: 1: 2nd axis move, 0: 2nd axis not move

Bit 6: 1: 3rd axis move, 0: 3rd axis not move

Bit 7: 1: 4th axis move, 0: 4th axis not move

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

I32 ret;
POINT_DATA_EX Point;

Point.i32_pos = 10000;      //(Center)Position data (could be relative or absolute value) (pulse)
Point.i16_accType = 1;      //Acceleration pattern 0: T curve, 1:S curve
…
//Set point data to card memory.
Ret = APS_set_point_table_ex(Axis_ID, 0, &Point );
if( ret != ERR_NoError )
{ //Error (C)
}
```

See also:

APS_set_point_table();APS_get_point_table();APS_get_point_table_ex();APS_point_table_move();APS_get_next_point_index();APS_get_start_point_index();APS_get_end_point_index()

APS_get_point_table_ex	Get point table move parameters with entend option

Support Products: PCI-8392(H)

Descriptions:

This function is used to get a set of point table parameters with entend option to specified axis.

Syntax:

C/C++:

I32 FNTYPE APS_get_point_table_ex(I32 Axis_ID, I32 Index, POINT_DATA_EX *Point);

Visual Basic:

APS_get_point_table_ex (ByVal Axis_ID As Integer, ByVal Index As Integer, ByRef Point As POINT_DATA_EX) As Integer

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Index: Specified point index to be set.

POINT_DATA_EX *Point: Structure of point table parameters. Define in “type_def.h”

```
typedef struct
{
  I32 i32_pos;            //(Center)Position data (could be relative or absolute value) (pulse)
  I16 i16_accType;        //Acceleration pattern 0: T curve, 1:S curve
  I16 i16_decType;        // Deceleration pattern 0: T curve, 1:S curve
  I32 i32_acc;            //Acceleration rate ( pulse / sec 2 )
  I32 i32_dec;            //Deceleration rate ( pulse / sec 2  )
  I32 i32_initSpeed;      //Start velocity ( pulse / s )
  I32 i32_maxSpeed;       //Maximum velocity    ( pulse / s )
  I32 i32_endSpeed;       //End velocity  ( pulse / s )
  I32 i32_angle;          //Arc move angle ( degree, -360 ~ 360 )
  U32 u32_dwell;          //dwell times ( unit: ms ) *Divided by system cycle time.
  I32 i32_opt;            //Point move option. (*)
  I32 i32_pitch;          // pitch for helical move
  I32 i32_totalheight;    // total hight
  I16 i16_cw;             // cw or ccw
  I16 i16_opt_ext;        // option extend (**)
} POINT_DATA_EX;
```

(*) Point move option: ***i32_opt***

7	6	5	4	3	2	1	Bit : 0
-	-	Last point	Finish condition	-	Linear/Arc	-	Absolute/Relative
15	14	13	12	11	10	9	Bit : 8
-	-	-	-	-	-	-	-

Bit 0: 1:Relative move, 0:Absolute move

Bit 2: 1:Arc move, 0:Linear move

Bit 3: 1: Enable VAO table switching control (when it is enabled, the setting table is effective of bit13 to bit 15), 0: Disable

Bit 4: 1:INP ON(In position signal), 0:CSTP ON(command stop signal)

Bit 5: 1: Last point index. 0: Not Last point index. (if this bit is turned on, point table move will stop after this point.)

Bit 8~15: Reserved.

(**) Point move option: ***i16_opt_ext***

7	6	5	4	3	2	1	Bit : 0
u	z	Y	x	-	-	-	Helical
15	14	13	12	11	10	9	Bit : 8
-	-	-	-	-	-	-	-

Bit 0: 1: helical move, 0: linear or arc move

If Bit 0 is 1, the motion type is helical move.

If Bit 0 is 0, the motion type is defined by Bit 2 of ***i32_opt.***

Bit 4: 1: 1st axis move, 0: 1st axis not move

Bit 5: 1: 2nd axis move, 0: 2nd axis not move

Bit 6: 1: 3rd axis move, 0: 3rd axis not move

Bit 7: 1: 4th axis move, 0: 4th axis not move

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
```

```
#include “APS168.h”
#include “ErrorCodeDef.h”

//... initial card.
I32 ret ;
POINT_DATA_EX Point;
ret =APS_get_point_table_ex( Axis_ID, 0, &Point );
if( ret != ERR_NoError )
{
        //Error.
}
```

See also:

APS_set_point_table();APS_get_point_table();APS_set_point_table_ex();APS_point_table_move();APS_get_next_point_index();APS_get_start_point_index();APS_get_end_point_index()

APS_point_table_move	Start a point table move

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to start a point table move. When point table move is started, the system will take the point parameters one by one from “StartIndex” to “EndIndex”. Therefore user must specified the point parameters to point table before perform point table move.

When the axis is in point table moving, user cannot perform others move until point table move is finish.

User could use stop_move, emg_stop, function to forced stop point table move.

Relative motion status description		
PMV	Point table move state	(Control axis) ON: in point table move state
PDW	Point table Dwell state	(Control axis) ON: in point table dwell state
PPS	Point table pause state	(Control axis) ON: in point table pause state
SLV	Slave axis move state	(Slave axis) ON: in slave axis move state

Reference axis: The first axis in axis array. User can specify it.

Control axis: The minimum axis ID will be the control axis.

Slave axis: Other axes except control axis.

For example:

Ex1.

I32 AxisArray[4] = {**3**, 1, 2, 4 };

Control axis is ID= 1.

Reference axis is ID = 3.

Slave axes are ID = 2, 3, 4

Ex2.

I32 AxisArray[3] = { **1**, 2, 4 };

Control axis is ID= 1.

Reference axis is ID = 1.

Slave axes are ID = 2, 4

Syntax:

C/C++:

I32 FNTYPE APS_point_table_move(I32 Dimension, I32 *Axis_ID_Array, I32 StartIndex, I32 EndIndex);

Visual Basic:

APS_point_table_move(ByVal Dimension As Long, Axis_ID_Array As Long, ByVal StartIndex As Long, ByVal EndIndex As Long) As Long

Parameters:

I32 Dimension: Dimension of axis array. (Linear move :1 ~ 4), (Arc move: 2)

I32 *Axis_ID_Array: Axis ID array.

I32 StartIndex: The first running point index.

I32 EndIndex: The end of point index. .

<Ex>

StartIndex = 3, EndIndex = 5.

The running sequence will be 3 -> 4 -> 5

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

I32 ret;
POINT_DATA Point;
I32 Axis_ID_Array;

Point.i32_pos = 10000;      //(Center)Position data (could be relative or absolute value) (pulse)
Point.i16_accType = 1;      //Acceleration pattern 0: T curve, 1:S curve
…
//Set point data to card memory.
Ret = APS_set_point_table(Axis_ID, 0, &Point );
…
if( ret != ERR_NoError )
```

```
{ //Error (C)
}

// Start a point table move.
Axis_ID_Array = Axis_ID;
ret = APS_point_table_move( 1, &Axis_ID_Array, 0 , 3 );
…
```

See also:

APS_set_point_table();APS_get_point_table();APS_point_table_move();APS_get_next_point_index();APS_get_start_point_index();APS_get_end_point_index()

APS_get_running_point_index	Get current point move index when axis is perform a point move

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to get the running point index when the axis is performing a point table move. For example, if the system is running index 3, this function will return index = 3.
If the operation is running at the last point, this function will return the “end point index”.
Note: When system’s state is at beginning, the default value is -1.

Syntax:

C/C++:

I32 FNTYPE APS_get_running_point_index(I32 Axis_ID, I32 *Index);

Visual Basic:

APS_get_running_point_index(ByVal Axis_ID As Long, Index As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Index: return running point index.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

//... initial card.
//…start network

I32 Index;
I32 ret = APS_get_running_point_index ( Axis_ID, &Index );
If( ret != ERR_NoError )
{ //Error (C)
}
```

See also:

APS_set_point_table();APS_get_point_table();APS_point_table_move();APS_get_start_point_index();APS_get_end_point_index()

APS_get_start_point_index	Get the first point move index when axis is perform a point move

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to get the first point index when the axis is performing a point table move.

Syntax:

C/C++:

I32 FNTYPE APS_get_start_point_index(I32 Axis_ID, I32 *Index);

Visual Basic:

APS_get_start_point_index(ByVal Axis_ID As Long, Index As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Index: return the first running point index.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

//... initial card.
//…start network

I32 Index;
I32 ret = APS_get_start_point_index ( Axis_ID, &Index );
If( ret != ERR_NoError )
{ //Error (C)
}
```

See also:

APS_set_point_table();APS_get_point_table();APS_point_table_move();APS_get_next_point_index();APS_get_end_point_index()

APS_get_end_point_index	Get the end of point move index when axis is perform a point move

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to get the end of point index when the axis is performing a point table move.

Syntax:

C/C++:

I32 FNTYPE APS_get_end_point_index(I32 Axis_ID, I32 *Index);

Visual Basic:

APS_get_end_point_index(ByVal Axis_ID As Long, Index As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Index: return the end of running point index.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

//... initial card.
//…start network

I32 Index;
I32 ret = APS_get_end_point_index( Axis_ID, &Index );
If( ret != ERR_NoError )
{ //Error (C)
}
```

See also:

APS_set_point_table();APS_get_point_table();APS_point_table_move();APS_get_next_point_index();APS_get_start_point_index()

APS_set_table_move_pause	Pause point table move

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to pauses the point table move. When pause command is issued, it will not stop current point but stop at next point index starting position.

Syntax:

C/C++:

I32 FNTYPE APS_set_table_move_pause(I32 Axis_ID, I32 Pause_en);

Visual Basic:

APS_set_table_move_pause(ByVal Axis_ID As Long, ByVal Pause_en As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Pause_en:

1: Pause. 0: Not pause.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

//... initial card.
//…start network

I32 Index;
I32 ret = APS_set_table_move_pause ( Axis_ID, 1 ); //Pause point table move
If( ret != ERR_NoError )
{ //Error (C)
}
```

See also:

APS_set_table_move_ex_pause	Decelerate to stop move and control I/O.

Support Products: PCI-8253/56

Descriptions:

This function is used to pauses move when running point table. When pause command is issued, it will decelerate to stop and control I/O. Other parameters included deceleration rate and I/O setting, could be configured by APS_set_axis_para().

Differences between APS_set_table_move_ex_pause() and APS_set_table_move_pause():

Function descriptions	APS_set_table_move_ex_pause()	APS_set_table_move_pause()
Motion status	NSTP(CSTP , INP)	PPS
Descriptions	Deceleration to stop & control I/O	Stop at next point index starting position.
Rollback	APS_set_table_move_ex_rollback()	N/A
Resume	APS_set_table_move_ex_resume()	APS_set_table_move_pause()

I/O could be controlled, such as disabling laser, while poiont table is pausing or normally stopping. Turning on/off specified I/O is configured in axis parameter table via APS_set_axis_para().

I/O setting in axis parameter table:

NO.	Define	Description	Value	Default
32h(50)	PRA_PT_STP_DO_EN	Enable Do when point table stoping/pausing	0: Disable 1: Enable	0
33h(51)	PRA_PT_STP_DO	Set Do value when Point table normally stopping/pausing	0: Set to 0 1: Set to 1	0

Syntax:

C/C++:

I32 FNTYPE APS_set_table_move_ex_pause(I32 Axis_ID);

Visual Basic:

APS_set_table_move_ex_pause(ByVal Axis_ID As Long, ByVal Pause_en As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

//... initial card.
// Pre-Configure parameters
// Enable Do when point table stoping/pausing
I32 ret = APS_set_axis_para( Axis_ID, 0x32, 1 );
// Set Do value to 1 (such as turning on laser) when Point table normally stopping/pausing
I32 ret = APS_set_axis_para( Axis_ID, 0x33, 1 );

//… move point table
I32 ret = APS_set_table_move_ex_pause( Axis_ID );
        //Stop point table move and control I/O.
If( ret != ERR_NoError )
{ //Error (C)
}
```

See also:

APS_set_table_move_ex_rollback();APS_set_table_move_ex_resume();APS_set_axis_para()

APS_set_table_move_ex_rollback	Rollback to starting position of current point index

Support Products: PCI-8253/56

Descriptions:

This function is used to rollback motion when point table paused. This function is used to rollback to starting position of the current index. Other parameters included start velocity, acceleration rate and deceleration rate could be configured by APS_set_axis_para().

Notice that this function will be used after APS_set_table_move_ex_pause() was called. Otherwise, it is possible to move to unexpected position.

Syntax:

C/C++:

I32 FNTYPE APS_set_table_move_ex_rollback(I32 Axis_ID, I32 Max_Speed);

Visual Basic:

APS_set_table_move_ex_rollback(ByVal Axis_ID As Long, ByVal Max_Speed As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Max_Speed: Maximum linear/circular interpolation speed. Unit: pulse/sec.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

//... initial card.
//… move point table, and pause it

I32 ret = APS_set_table_move_ex_rollback ( Axis_ID, Max_Speed );
        // Rollback move.
If( ret != ERR_NoError )
```

```
{ //Error (C)
}
```

See also:

APS_set_table_move_ex_pause();APS_set_table_move_ex_resume()

APS_set_table_move_ex_resume	Re-start point table move and keep I/O status.

Support Products: PCI-8253/56

Descriptions:

This function is used to resume move from current index to end index when point table paused. When resume command is issued, it will re-start point table move. When passing through the pause position, it will keep I/O status.

Notice that this function will be used after APS_set_table_move_ex_rollback() was called. Otherwise, it is possible to move to unexpected position.

Difference between APS_set_table_move_ex_resume() and APS_set_table_move_pause():

Function descriptions	APS_set_table_move_ex_resume()	APS_set_table_move_pause() (when resuming move)
Motion status	PMV, SLV	PMV, SLV
Descriptions	Resume move from current index to end index.	Resume move from next index to end index.
Pause	APS_set_table_move_ex_pause()	APS_set_table_move_pause() (when pausing move)

Syntax:

C/C++:

I32 FNTYPE APS_set_table_move_ex_resume(I32 Axis_ID);

Visual Basic:

APS_set_table_move_ex_resume(ByVal Axis_ID As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”
```

```
//... initial card.
//… move point table,
//…Pause it, then rollback.

I32 ret = APS_set_table_move_ex_resume( Axis_ID );
        // Re-start point table move and keep I/O status.
If( ret != ERR_NoError )
{ //Error (C)
}
```

See also:

APS_set_table_move_ex_pause();APS_set_table_move_ex_rollback()

APS_set_table_move_repeat	Set point table move repeat

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to set point table move repeat. When repeat function is enabled, it will repeat the point move until repeat function is disabled or stop function is issued.

Syntax:

C/C++:

I32 FNTYPE APS_set_table_move_repeat (I32 Axis_ID, I32 Repeat_en);

Visual Basic:

APS_set_table_move_repeat (ByVal Axis_ID As Long, ByVal Repeat_en As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Repeat_en:

1: Repeat. 0: Not repeat.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

//... initial card.
//…start network

I32 Index;
I32 ret = APS_set_table_move_repeat ( Axis_ID, 1 ); // Repeat point table move
If( ret != ERR_NoError )
{ //Error (C)
}
```

See also:

APS_set_point_table_mode2	Set point table mode

Support Products: MNET-4XMO-C

Descriptions:

This function is used to select a point table mode. There are two modes for point table: Single (Fast Index move) mode and Continuous (Path move) mode. Only one mode can be selected on a specified slave module at the same time. User should call this function to choose mode before using other point table functions.

For Single Mode – Fast Index Move (mode = 0):

It provides a fast way to start a move. Because MNET is using communication way to send/receive command and data, the access time depends on the network speed and the amount of data. It provides a fast way to let users to preset known data on SRAM. It can save much time on communication only by a point index command.

For Continuous Mode – Path Move (mode = 1):

It not only can make path locus running continuously without host PC's control but also can make path speed continuously by auto calculating from our software.

Users only need to give maximum speed and target position data and don't need to take care of starting speed for intercommand speed's continuity. This is so called auto speed profile feature.

A dwell move can be a part of path move. Dwell move means a certain time of axis still.

There is only one limitation for these piecewise point data: The distance for each segment must be long enough to support the time from current speed to be accelerated or decelerated to target maximum speed. Or it will return ERR_DistantEnough.

Syntax:

C/C++:

I32 FNTYPE APS_set_point_table_mode2 (I32 Axis_ID, I32 Mode);

Visual Basic:

APS_set_point_table_mode2 (ByVal Axis_ID As Long , ByVal Mode As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

For MENT-4XMO-C: Axis_ID on the specified slave module.

I32 Mode: Specified point table mode. (Default is 0)

0: Single Mode (Fast Index Move)

1: Continuous Mode (Path Move)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

I32 ret;
I32 Axis_ID = 1000;

ret = APS_set_point_table_mode2 (Axis_ID, 1); //Set to continuous mode
//... Set other point table functions
```

See also:

APS_set_point_table2	Set point table2 move parameters

Support Products: MNET-4XMO-C

Descriptions:

This function is used to set a set of point table parameters. The point table defined in APS is not only a point table but also an instruction table. Users can implement a move according to this point table. The table content can be used to different speed parameters.

For Single Mode – Fast Index Move (mode = 0):

When point table is running on single mode, the maximum number of points is 1024. It supports absolute and relative move for 1-axis motion. Notice that it only supports relative move for linear and arc muti-interpolation motion. It also supports dwell move. The point 0 to point N are not necessary in the same dimension and axis.

For Continuous Mode – Path Move (mode = 1):

When point table is running on continuous mode, the maximum number of points is 1,048,560. The SRAM buffer can preset 2048 points for 1-axis path single motion. If users need interpolation, 1024 points for 2-axis or 682 points for 3-axis or 512 points for 4-axis are possible includes circular motion. The circular motion is only for 2-axis setting. Notice that point 0 to point N are necessary in the same dimension and axis.

The starting speed, acceleration and deceleration rate are fixed from the beginning setting for whole path move. Please pre-set those value before path move.

Note: When point table is running on continuous mode, be sure to set each Point from index 0 to N in order.

Note: It will cause point table to re-initialize when setting Point to index 0,

Syntax:

C/C++:

I32 FNTYPE APS_set_point_table2 (I32 Dimension, I32 *Axis_ID_Array, I32 Index, POINT_DATA2 *Point);

Visual Basic:

APS_set_point_table2 (ByVal Dimension As Long , Axis_ID_Array As Long, ByVal Index As Long, Point As POINT_DATA2) As Long

Parameters:

I32 Dimension: Dimension of axis array. (Linear & Dwell move :1 ~ 4), (Arc move: 2)

I32 *Axis_ID_Array: Axis ID array on specified slave module

I32 Index: Specified point index to be set.

POINT_DATA2 *Point: Structure of point table parameters. Define in “type_def.h”

```
typedef struct
{
    I32 i32_pos[16];      //(Center)Position data (could be relative or absolute value) (pulse)
    I32 i32_initSpeed;    //Start velocity (Only available for single mode) ( pulse / s )
    I32 i32_maxSpeed;     //Maximum velocity    ( pulse / s )
    I32 i32_angle;        //Arc move angle ( degree, -360 ~ 360 )
    U32 u32_dwell;        //dwell times ( unit: ms )
    I32 i32_opt;          //Point move option. (*)
} POINT_DATA2;
```

(*) Point move option: ***i32_opt***

7	6	5	4	3	2	1	Bit : 0
-	-	Last point	Finish condition	-	Linear/Arc	-	Absolute/Relative

Bit 0: 1:Relative move, 0:Absolute move

Bit 2: 1:Arc move, 0:Linear move

Bit 4: 1:INP ON(In position signal), 0:CSTP ON(command stop signal)

Bit 5: 1: Last point index. 0: Not Last point index. (if this bit is turned on, point table move will stop after this point.) It is only available for continuous mode.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

I32 ret;
I32 Dimension = 2; // Interpolation for 2-axes.
I32 Axis_ID_Array[2] = { 1000, 1001 };
POINT_DATA2 Point;

…pre-set starting speed, acceleration and deceleration rate..
```

```
Point.i32_pos[0] = 10000;   //(Center)Position data (could be relative or absolute value) (pulse)
Point.i32_pos[1] = 20000;   //(Center)Position data (could be relative or absolute value) (pulse)
Point.i32_maxSpeed = 10000;    //Maximum velocity    ( pulse / s )
Point.i32_opt = 0; // Absolute, Linear, CSTP ON, Not Last point index

//Set point data to on-board SRAM.
Ret = APS_set_point_table2 (Dimension, Axis_ID_Array, 0, &Point ); //Index 0
//... set index in order.
If( ret != ERR_NoError )
{ //Error (C)
}
```

See also:

APS_point_table_continuous_move 2	Start a point table continuous move

Support Products: MNET-4XMO-C

Descriptions:

User must set point table to continuous mode with APS_set_point_table_mode2() before using this function. This function is used to start a point table continuous move. When point table move is started, the system will take the point parameters one by one from “0” to “LastPoint”. Therefore user must specify the point parameters to point table before perform point table move.

User could use stop_move, emg_stop, function to forced stop point table move.

Syntax:

C/C++:

I32 FNTYPE APS_point_table_continuous_move2(I32 Dimension, I32 *Axis_ID_Array);

Visual Basic:

APS_point_table_continuous_move2(ByVal Dimension As Long, Axis_ID_Array As Long) As Long

Parameters:

I32 Dimension: Dimension of axis array. (Linear & Dwell move :1 ~ 4), (Arc move: 2)

I32 *Axis_ID_Array: Axis ID array on specified slave module

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

I32 ret;
I32 Dimension = 2; // Interpolation for 2-axes.
I32 Axis_ID_Array[2] = { 1000, 1001 };
I32 Index = 0;
```

```
POINT_DATA2 Point;
I32 PointTableStatus;

…pre-set starting speed, acceleration and deceleration rate..

Point.i32_pos[0] = 10000;  //(Center)Position data (could be relative or absolute value) (pulse)
Point.i32_pos[1] = 20000;  //(Center)Position data (could be relative or absolute value) (pulse)
Point.i32_maxSpeed = 10000;   //Maximum velocity   ( pulse / s )
Point.i32_opt = 0; // Absolute, Linear, CSTP ON, Not Last point index

//Set point data to on-board SRAM.
Ret = APS_set_point_table2 (Dimension, Axis_ID_Array, 0, &Point ); //Index 0
Index++;
//...Preset Point(index) in order.
If( ret != ERR_NoError )
{ //Error (C)
}

ret = APS_set_point_table_mode2 (Axis_ID, 1); //Set to continuous mode

// Start a point table continuous move.
Ret = APS_point_table_continuous_move2 (Dimension, Axis_ID_Array );
…
//Check point table status & Re-load Point(index) in order
ret = APS_point_table_status2( Axis_ID_Array[0], &PointTableStatus );
if( PointTableStatus == 1 ) //SRAM is not full
{
      // Reload Point
      ret = APS_set_point_table2 (Dimension, Axis_ID_Array, 0, &Point ); //Index 0
      Index++;
}else
{
      //Cant Reload Point
}
```

See also:

APS_point_table_single_move2	Start a point table single move

Support Products: MNET-4XMO-C

Descriptions:

User must set point table to single mode with APS_set_point_table_mode2() before using this function. This function is used to start a point table single move. When point table move is started, the system will perform a single move according to specified index. Therefore user must specify the point parameters to point table before perform point table move.

User could use stop_move, emg_stop, function to forced stop point table move.

Syntax:

C/C++:

I32 FNTYPE APS_point_table_single_move2 (I32 Axis_ID, I32 Index);

Visual Basic:

APS_point_table_single_move2 (ByVal Axis_ID As Long, ByVal Index As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

For MENT-4XMO-C: Axis_ID on the specified slave module.

I32 Index: Specify point index to move.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

I32 ret;
I32 Dimension = 2; // Interpolation for 2-axes.
I32 Axis_ID_Array[2] = { 1000, 1001 };
POINT_DATA2 Point;

…pre-set acceleration and deceleration rate..
```

```
Point.i32_pos[0] = 10000;   //(Center)Position data (could be relative or absolute value) (pulse)
Point.i32_pos[1] = 20000;   //(Center)Position data (could be relative or absolute value) (pulse)
Point.i32_initSpeed = 0;      //Start velocity (Only available for single mode) ( pulse / s )
Point.i32_maxSpeed = 10000;    //Maximum velocity    ( pulse / s )
Point.i32_opt = 1; // Relative, Linear, CSTP ON, Not Last point index

ret = APS_set_point_table_mode2 (Axis_ID, 0); //Set to single mode

//Set point data to on-board SRAM.
Ret = APS_set_point_table2 (Dimension, Axis_ID_Array, 0, &Point ); //Set index 0
if( ret != ERR_NoError )
{ //Error (C)
}

// Start a point table single move.
Ret = APS_point_table_single_move2 (Axis_ID_Array[0], 0 ); //Move index 0
…
```

See also:

APS_get_running_point_index2	Get current point move index when point table move is running

Support Products: MNET-4XMO-C

Descriptions:

This function is used to get the running point index when performing a point table move. For example, if the system is running index 3, this function will return index = 3.

If the operation is running at the last point, this function will return the “last point index”.

Syntax:

C/C++:

I32 FNTYPE APS_get_running_point_index(I32 Axis_ID, I32 *Index);

Visual Basic:

APS_get_running_point_index(ByVal Axis_ID As Long, Index As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

For MENT-4XMO-C: Axis_ID on the specified slave module.

I32 *Index: return running point index.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

//... initial card.
//…start network

I32 Index;
I32 ret = APS_get_running_point_index2 ( Axis_ID, &Index );
If( ret != ERR_NoError )
{ //Error (C)
}
```

See also:

APS_point_table_status2	Get point table status when point table move is running

Support Products: MNET-4XMO-C

Descriptions:

MNET-4XMO-C provides one dedicated on-board SRAM to store point data and makes continuous path move standalone possible. This function is used to get SRAM status when performing a point table continuous move. User can reload point table when SRAM is not full.

Syntax:

C/C++:

I32 FNTYPE APS_point_table_status2(I32 Axis_ID, I32 *Status);

Visual Basic:

APS_point_table_status2(ByVal Axis_ID As Long, Status As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

For MENT-4XMO-C: Axis_ID on the specified slave module.

I32 *Status: get SRAM status for point table.

0: SRAM is full.

1: SRAM is not full.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

//... initial card.
//…start network

I32 Status;
I32 ret = APS_point_table_status2 ( Axis_ID, &Status );
If( ret != ERR_NoError )
```

```
{ //Error (C)
}
```

See also:

APS_set_point_table3	Set point table3 move parameters

Support Products: HSL-4XMO

Descriptions:

This function is used to set a set of point table parameters. The point table defined in APS is not only a point table but also an instruction table. Users can implement a move according to this point table. The table content can be used to different speed parameters.

The point table can store totally 2000 points (from 0 to 1999). Users can use the structure variable POINT_DATA3 provided by us to set data for each point. The POINT_DATA3 structure variable includes five components: position, max speed, end position, direction, and command function. It has to be noticed that the number of axis and the axes in axis array on each point must be equal under one movement. Move types are decided by command function and it must meet the number of axis ateach point set by user.

The starting speed, acceleration and deceleration rate are fixed from the beginning setting for whole path move. Please pre-set those value before path move via APS_set_point_table_param3 function..

Note: There are some notes in setting point table listed below:

1. Starting velocity must be smaller than max velocity.
2. The table has two points at least.
3. When previous point is arc move, the max velocity in next point must bigger the previous point.
4. Final point can‘t be a arc move.
5. The axis must be unique in axis array.
6. The axis in axis array must be in the same module.
7. The number of axis and axes number in axis array at each point must be equal under one movement.

Syntax:

C/C++:

I32 FNTYPE APS_set_point_table3(I32 Dimension, I32 *Axis_ID_Array, I32 Index, POINT_DATA3 *Point);

Visual Basic:

APS_set_point_table3 (ByVal Dimension As Long, Axis_ID_Array As Long, ByVal Index As Long, Point As POINT_DATA2) As Long

Parameters:

I32 Dimension: Dimension of axis array. (Line move:1 ~ 4), (Arc move: 2)

I32 *Axis_ID_Array: Axis ID array on specified slave module

I32 Index: Specified point index to be set.

POINT_DATA3 *Point: Structure of point table parameters. Define in “type_def.h”

```
typedef struct
{
    I32 i32_pos[4];         //(Center)Position data (could be relative or absolute value) (pulse)
    I32 i32_maxSpeed;   //Maximum velocity    ( pulse / s )
    I32 i32_endPos[2]    //For arc move
    I32 i32_dir;                 //For arc move
    I32 i32_opt;            //Point move option. (*)
} POINT_DATA3;
```

(*) Point move option: ***i32_opt***

Value	Move Type	Value	Move Type	Value	Move Type
0	start_tr_move	7	start_sa_line2	14	start_sr_line3
1	start_ta_move-	8	start_tr_arc2	15	start_sa_line3
2	start_sr_move	9	start_ta_arc2	16	start_sa_line3
3	start_sa_move	10	start_sr_arc2	17	start_ta_line4
4	start_tr_line2	11	start_sa_arc2	18	start_sr_line4
5	start_ta_line2	12	start_tr_line3	19	start_sa_line4
6	start_sr_line2	13	start_ta_line3		

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

I32 ret;
I32 Dimension = 2; // Interpolation for 2-axes.
I32 Axis_ID_Array[2] = { 0, 1};
POINT_DATA3 Point;
```

…pre-set starting speed, acceleration and deceleration rate..

Point.i32_pos[0] = 10000; //(Center)Position data (could be relative or absolute value) (pulse)
Point.i32_pos[1] = 20000; //(Center)Position data (could be relative or absolute value) (pulse)
Point.i32_maxSpeed = 10000; //Maximum velocity (pulse / s)
Point.i32_opt = 0; // Absolute, Linear, Not Last point index

See also:

APS_point_table_move3(); APS_set_point_table_param3()

APS_point_table_move3	Start a point table move

Support Products: HSL-4XMO

Descriptions:

This function is used to start a point table move. When point table move is started, the system will take the point parameters one by one from “StartIndex” to “EndIndex”. Therefore user must specify the point parameters to point table before perform point table move.

When the axis is in point table moving, user cannot perform others move until point table move is finish.

User could uses stop_move, emg_stop, function to forced stop point table move.

Syntax:

C/C++:

I32 FNTYPE APS_point_table_move3 (I32 Dimension, I32 *Axis_ID_Array, I32 StartIndex, I32 EndIndex)

Visual Basic:

APS_point_table_move3 (ByVal Dimension As Long, Axis_ID_Array As Long, StartIndex As Long, EndIndex As Long) As Long

Parameters:

I32 Dimension: Dimension of axis array. (Linear & Dwell move: 1 ~ 4), (Arc move: 2)

I32 *Axis_ID_Array: Axis ID array on specified slave module

Note:

1. The number of axis and axes number in axis array must be equal the axis array in the point.
2. The axis in axis array must be in the same module.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”
```

```
I32 ret;
I32 index;
I32 Dimension = 2; // Interpolation for 2-axes.
I32 Axis_ID_Array[2] = { 0, 1};
I32 StartIndex = 0;
I32 EndIndex = 1;
POINT_DATA3 Point;

…pre-set starting speed, acceleration and deceleration rate..

Point.i32_pos[0] = 10000;  //(Center)Position data (could be relative or absolute value) (pulse)
Point.i32_pos[1] = 20000;  //(Center)Position data (could be relative or absolute value) (pulse)
Point.i32_maxSpeed = 10000;   //Maximum velocity   ( pulse / s )
Point.i32_opt = 4; // start_tr_line2
Index = 0;
ret = APS_set_point_table3 (Dimension, Axis_ID_Array, index, &Point ); //Index 0

Point.i32_pos[0] = 20000;  //(Center)Position data (could be relative or absolute value) (pulse)
Point.i32_pos[1] = 10000;  //(Center)Position data (could be relative or absolute value) (pulse)
Point.i32_maxSpeed = 10000;   //Maximum velocity   ( pulse / s )
Point.i32_opt = 6; // start_sr_line2
Index = 1;
ret = APS_set_point_table3 (Dimension, Axis_ID_Array, index, &Point ); //Index 1

ret = APS_point_table_move3( Dimension, Axis_ID_Array, 0, 1 )
```

See also:

APS_set_point_table3(); APS_set_point_table_param3()

APS_set_point_table_param3	Set speed parameter for point table move

Support Products: HSL-4XMO

Descriptions:

This function is used to set the speed parameter for point table move including start velocity, acceleration, deceleration, scrve acceleration, and scrve deceleration. The numbers of each parameter are the same with axis parameter used by APS_set_axis_param. Users can refer to axis parameter table to set the speed parameter.

Syntax:

C/C++:

I32 FNTYPE APS_set_point_table_param3 (I32 FirstAxid, I32 ParaNum, I32 ParaDat);

Visual Basic:

APS_set_point_table_param3 (ByVal FirstAxid As Long, ParaNum As Long, ParaDat As Long) As Long;

Parameters:

I32 FirstAxid: The first axis in axis array set by APS_set_point_table3 function.

I32 ParaNum: The axis parameter please refer to axis table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “type_def.h”
#include “APS_define.h”
#include “APS168.h”
#include “ErrorCodeDef.h”

I32 ret;
I32 index;
I32 Dimension = 2; // Interpolation for 2-axes.
I32 Axis_ID_Array[2] = { 0, 1};
I32 StartIndex = 0;
I32 EndIndex = 1;
POINT_DATA3 Point;
```

…pre-set starting speed, acceleration and deceleration rate..

```
Point.i32_pos[0] = 10000;   //(Center)Position data (could be relative or absolute value) (pulse)
Point.i32_pos[1] = 20000;   //(Center)Position data (could be relative or absolute value) (pulse)
Point.i32_maxSpeed = 10000;    //Maximum velocity    ( pulse / s )
Point.i32_opt = 4; // start_tr_line2
Index = 0;
ret = APS_set_point_table3 (Dimension, Axis_ID_Array, index, &Point ); //Index 0

Point.i32_pos[0] = 20000;   //(Center)Position data (could be relative or absolute value) (pulse)
Point.i32_pos[1] = 10000;   //(Center)Position data (could be relative or absolute value) (pulse)
Point.i32_maxSpeed = 10000;    //Maximum velocity    ( pulse / s )
Point.i32_opt = 6; // start_sr_line2
Index = 1;
ret = APS_set_point_table3 (Dimension, Axis_ID_Array, index, &Point ); //Index 1

ret = APS_set_point_table_param3 ( 0, PRA_ACC, 50000 );    //Set acceleration for point table
move
ret = APS_set_point_table_param3 ( 0, PRA_DEC, 50000 );    //Set deceleration for point table
move
ret = APS_set_point_table_param3 ( 0, PRA_VS, 100 );    //Set start velocity for point table
move.
Ret = APS_set_point_table_param3 ( 0, PRA_SACC, 5000 ); //Set scurve acceleration for
point table move
ret = APS_set_point_table_param3 ( 0, PRA_SDEC, 50000); //Set scurve deceleration for point
table


ret = APS_point_table_move3( Dimension, Axis_ID_Array, 0, 1 )
```

See also:

APS_set_point_table3(); APS_point_table_move3()

APS_set_feeder_group	Set axes into a feeder group

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to set axes into a feeder group. Before you used any other feeder function, you should assign some axes to a feeder group. When you no longer use the feeder, you should free the group by *APS_free_feeder_group*() function.

Note:

The current feeder only support two dimension axis ID group.

Syntax:

C/C++:

I32 FNTYPE APS_set_feeder_group(I32 GroupId, I32 Dimension, I32 *Axis_ID_Array);

Visual Basic:

APS_set_feeder_group(ByVal GroupId As Long, ByVal Dimension As Long, Axis_ID_Array As Long) As Long

Parameters:

I32 GroupId: Group ID. Value range: 0~1.

I32 Dimension: The dimension of the axis ID array. Value range: 1~4

I32 *Axis_ID_Array: The Axis ID array from 0 to 65535. The array size must match the axis dimension. The axis-ID in Axis_ID_Array[0] represent as the control axis which must the minimum ID number in the array.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “APS168.h”
#include “ErrorCodeDef.h”

I32 ret; // Return code
I32 groupId = 0; // Feeder group ID [0,1]
I32 runIdx; // Which index of data is in operation
I32 fedIdx; // How much data is loaded into feeder module.
I32 msts;   // Motion status
I32 dim = 2; // Group dimension
I32 ax[2] = { 0, 1,}; // Axes ID array
```

```
PNT_DATA_2D* pPnt    = NULL;// Pointer of PNT_DATA_2D

ret = APS_set_feeder_group( groupId, dim, ax );
if( ret != ERR_NoError ){ //Exception handling }

ret = APS_reset_feeder_buffer(groupId );

ret = APS_set_feeder_point_2D(groupId, pPnt, cnt, 1 ); //or APS_set_feeder_point_2D_F64()
if( ret != ERR_NoError ) { //Exception handling }

// Start feeder and point table move
ret = APS_start_feeder_move( groupId );
if( ret != ERR_NoError ) { //Exception handling }


// Check whether the end of the point table move procedure
{
        ret = APS_get_feeder_running_index(groupId, &runIdx);
        if( ret != ERR_NoError ) break;
        ret = APS_get_feeder_feed_index(groupId, &fedIdx);
        if( ret != ERR_NoError ) break;
        msts = APS_motion_status( ax [0] );
        // Check motion status.

}while( runIdx != ( fedIdx -1 ) );


ret = APS_free_feeder_group(groupId);
if( ret != ERR_NoError ) { //Exception handling }
```

See also:

I32 FNTYPE APS_get_feeder_group(I32 GroupId, I32 *Dimension, I32 *Axis_ID_Array);

I32 FNTYPE APS_free_feeder_group(I32 GroupId);

I32 FNTYPE APS_reset_feeder_buffer(I32 GroupId);

I32 FNTYPE APS_set_feeder_point_2D (I32 GroupId, POINT_DATA_2D* PtArray, I32 Size,

I32 LastFlag);

I32 FNTYPE APS_start_feeder_move(I32 GroupId);

I32 FNTYPE APS_get_feeder_running_index(I32 GroupId, I32 *Index);

I32 FNTYPE APS_get_feeder_feed_index(I32 GroutId, I32 *Index);

APS_get_feeder_group	Return the configuration in one feeder group

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get the configuration of a specified feeder. The configuration include group dimension and which axis IDs in group.

Syntax:

C/C++:

I32 FNTYPE APS_get_feeder_group(I32 GroupId, I32 *Dimension, I32 *Axis_ID_Array);

Visual Basic:

APS_get_feeder_group (ByVal GroupId As Long, Dimension As Long, Axis_ID_Array As Long) As Long

Parameters:

I32 GroupId: Group ID. Value range: 0~1.

I32 *Dimension: Return group axes dimension. Possible return value [0~4].

I32 *Axis_ID_Array: Return the Axis ID from 0 to 65535. Please give a array of **constant size 4**.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Refer to the example of *APS_set_feeder_group*()

See also:

I32 FNTYPE APS_set_feeder_group(I32 GroupId, I32 Dimension, I32 *Axis_ID_Array);

I32 FNTYPE APS_free_feeder_group(I32 GroupId);

I32 FNTYPE APS_reset_feeder_buffer(I32 GroupId);

I32 FNTYPE APS_set_feeder_point_2D(I32 GroupId, POINT_DATA_2D* PtArray, I32 Size, I32 LastFlag);

I32 FNTYPE APS_start_feeder_move(I32 GroupId);

I32 FNTYPE APS_get_feeder_running_index(I32 GroupId, I32 *Index);

I32 FNTYPE APS_get_feeder_feed_index(I32 GroutId, I32 *Index);

APS_free_feeder_group	Free a feeder group and it's resources

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to free the axes from the feeder and free its resources. When you no long to use the feeder, you must use this function to release the resources or it will keep the resources until the process be terminated.

Syntax:

C/C++:

I32 FNTYPE APS_free_feeder_group(I32 GroupId);

Visual Basic:

APS_free_feeder_group(ByVal GroupId As Long) As Long

Parameters:

I32 GroupId: Group ID. Value range: 0~1.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Refer to the example of *APS_set_feeder_group*()

See also:

I32 FNTYPE APS_set_feeder_group(I32 GroupId, I32 Dimension, I32 *Axis_ID_Array);

I32 FNTYPE APS_get_feeder_group(I32 GroupId, I32 *Dimension, I32 *Axis_ID_Array);

I32 FNTYPE APS_reset_feeder_buffer(I32 GroupId);

I32 FNTYPE APS_set_feeder_point_2D(I32 GroupId, POINT_DATA_2D* PtArray, I32 Size, I32 LastFlag);

I32 FNTYPE APS_start_feeder_move(I32 GroupId);

I32 FNTYPE APS_get_feeder_running_index(I32 GroupId, I32 *Index);

I32 FNTYPE APS_get_feeder_feed_index(I32 GroutId, I32 *Index);

APS_reset_feeder_buffer	Reset the feeder's point buffer

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to reset the 2D point table data buffer of a feeder.

Note:

1. When feeder is loading the data to controller, you cannot use this function to reset the feeder buffer.
2. When issue the *APS_set_feeder_point_[n]D*() and the LastFlag is set. Use this function to reset the buffer and clear LastFlag.

Syntax:

C/C++:

I32 FNTYPE APS_reset_feeder_buffer(I32 GroupId);

Visual Basic:

APS_reset_feeder_buffer (ByVal GroupId As Long) As Long

Parameters:

I32 GroupId: Group ID. Value range: 0~1.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Refer to the example of *APS_set_feeder_group*()

See also:

I32 FNTYPE APS_set_feeder_group(I32 GroupId, I32 Dimension, I32 *Axis_ID_Array);

I32 FNTYPE APS_get_feeder_group(I32 GroupId, I32 *Dimension, I32 *Axis_ID_Array);

I32 FNTYPE APS_free_feeder_group(I32 GroupId);

I32 FNTYPE APS_set_feeder_point_2D(I32 GroupId, POINT_DATA_2D* PtArray, I32 Size, I32 LastFlag);

I32 FNTYPE APS_start_feeder_move(I32 GroupId);

I32 FNTYPE APS_get_feeder_running_index(I32 GroupId, I32 *Index);

I32 FNTYPE APS_get_feeder_feed_index(I32 GroutId, I32 *Index);

APS_set_feeder_point_2D	Add a point into feeder's buffer

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to set two dimension trajectory data into the buffer of a feeder. The parameter"LastFlag"must be set when the last piece of trajectory data is set. After "LastFlag"is be set, the function "*APS_start_feeder_move*()"can be execute. When "LastFlag"is set, the trajectory data cannot be set into buffer until *APS_reset_feeder_buffer*() is called.

Syntax:

C/C++:

I32 FNTYPE APS_set_feeder_point_2D(I32 GroupId, PNT_DATA_2D * PtArray, I32 Size, I32 LastFlag);

Visual Basic:

APS_set_feeder_point_2D (ByVal GroupId As Long, PtArray As PNT_DATA_2D, ByVal Size As Long, ByVal LastFlag As Long) As Long

Parameters:

I32 GroupId: Group ID. Value range: 0~1.

PNT_DATA_2D* PtArray: Two dimension trajectory information array.

I32 Size: PNT_DATA_2D array size. Value must large than 0. (Size > 0)

I32 LastFlag: Last point data flag. To notice the feeder the point array is the last one for feeder.

0: Not the last one

1: Last one.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Refer to the example of *APS_set_feeder_group*()

See also:

I32 FNTYPE APS_set_feeder_group(I32 GroupId, I32 Dimension, I32 *Axis_ID_Array);

I32 FNTYPE APS_get_feeder_group(I32 GroupId, I32 *Dimension, I32 *Axis_ID_Array);

I32 FNTYPE APS_free_feeder_group(I32 GroupId);

I32 FNTYPE APS_reset_feeder_buffer(I32 GroupId);

I32 FNTYPE APS_start_feeder_move(I32 GroupId);

```
I32 FNTYPE APS_get_feeder_running_index( I32 GroupId, I32 *Index );
I32 FNTYPE APS_get_feeder_feed_index( I32 GroutId, I32 *Index );
```

APS_set_feeder_point_2D_ex	Add a point into feeder's buffer

Support Products: PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to set two dimension trajectory data into the buffer of a feeder. The parameter"LastFlag"must be set when the last piece of trajectory data is set. After "LastFlag"is be set, the function "*APS_start_feeder_move*()"can be execute. When "LastFlag"is set, the trajectory data cannot be set into buffer until *APS_reset_feeder_buffer*() is called.

Caution:

APS_set_feeder_point_2D_ex() and APS_set_feeder_point_2D() functions cannot be used at the same time. APS_set_feeder_point_2D_ex() is used by F64 type, and APS_set_feeder_point_2D() is used by I32 type.

Syntax:

C/C++:

I32 FNTYPE APS_set_feeder_point_2D_ex(I32 GroupId, PNT_DATA_2D_F64 * PtArray, I32 Size, I32 LastFlag);

Visual Basic:

APS_set_feeder_point_2D_ex(ByVal GroupId As Long, PtArray As PNT_DATA_2D_F64, ByVal Size As Long, ByVal LastFlag As Long) As Long

Parameters:

I32 GroupId: Group ID. Value range: 0~1.

PNT_DATA_2D_F64* PtArray: Two dimension trajectory information array.

I32 Size: PNT_DATA_2D_F64 array size. Value must large than 0. (Size > 0)

I32 LastFlag: Last point data flag. To notice the feeder the point array is the last one for feeder.

0: Not the last one

1: Last one.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Refer to the example of *APS_set_feeder_group*()

See also:

I32 FNTYPE APS_set_feeder_group(I32 GroupId, I32 Dimension, I32 *Axis_ID_Array);

I32 FNTYPE APS_get_feeder_group(I32 GroupId, I32 *Dimension, I32 *Axis_ID_Array);

I32 FNTYPE APS_free_feeder_group(I32 GroupId);

I32 FNTYPE APS_reset_feeder_buffer(I32 GroupId);

I32 FNTYPE APS_start_feeder_move(I32 GroupId);

I32 FNTYPE APS_get_feeder_running_index(I32 GroupId, I32 *Index);

I32 FNTYPE APS_get_feeder_feed_index(I32 GroutId, I32 *Index);

APS_start_feeder_move	Start point table move and feed points.

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8254/58 / AMP-204/8C

Descriptions:

The following items will be executed when this function is issued.

1 Load points into controller (Point table).

2 Start point table move.

This function will fail when the parameter “LastFlag”of function *APS_set_feeder_point_[n]D()* does not be set.

Syntax:

C/C++:

I32 FNTYPE APS_start_feeder_move(I32 GroupId);

Visual Basic:

APS_start_feeder_move (ByVal GroupId As Long) As Long

Parameters:

I32 GroupId: Group ID. Value range: 0~1.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Refer to the example of *APS_set_feeder_group*()

See also:

I32 FNTYPE APS_set_feeder_group(I32 GroupId, I32 Dimension, I32 *Axis_ID_Array);

I32 FNTYPE APS_get_feeder_group(I32 GroupId, I32 *Dimension, I32 *Axis_ID_Array);

I32 FNTYPE APS_free_feeder_group(I32 GroupId);

I32 FNTYPE APS_reset_feeder_buffer(I32 GroupId);

I32 FNTYPE APS_set_feeder_point_2D (I32 GroupId, PNT_DATA_2D * PtArray, I32 Size, I32 LastFlag);

I32 FNTYPE APS_get_feeder_running_index(I32 GroupId, I32 *Index);

I32 FNTYPE APS_get_feeder_feed_index(I32 GroutId, I32 *Index);

APS_get_feeder_status	Get status of feeder

Support Products: PCI-8254/58 / AMP-204/8C

Descriptions:

User could monitor status of a feeder, including feeder states and feeder error code.

There are three states of a feeder as following:

0: Feeder_Stop: Feeder is stopped.

1: Feeder_Run: Feeder is running.

2: Feeder_Pause: Feeder is paused by APS_set_feeder_ex_pause().

Error code refers to APS Functions Return Code.

Syntax:

C/C++:

I32 FNTYPE APS_get_feeder_status(I32 GroupId, I32 *State, I32 *ErrCode);

Visual Basic:

APS_get_feeder_status(ByVal GroupId As Long, State As Long, ErrCode As Long) As Long

Parameters:

I32 GroupId: Group ID. Value range: 0~1.

I32 *State: State of a feeder

0: Feeder_Stop: Feeder is stopped.

1: Feeder_Run: Feeder is running.

2: Feeder_Pause: Feeder is paused.

I32 *ErrCode: runtime error code of a feeder. Refer to APS Functions Return Code.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 state = 0;
I32 errorCode = 0;
I32 ret = 0;

//Get feeder status
ret = APS_get_feeder_status( 0, &state, &errorCode );
```

See also:

APS_start_feeder_move()

APS_get_feeder_running_index	Get which point is in operation.

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to observe which buffer index currently the controller being processed. The index of the buffer is the array index you feed to buffer.

This function is similar with *APS_get_running_point_index*(), but the different is the order of the index. *APS_get_running_point_index*() return point table index which order by point table itself in operation ram; *APS_get_feeder_running_index*() return buffer index which order by feeder's buffer in host's ram.

Syntax:

C/C++:

I32 FNTYPE APS_get_feeder_running_index(I32 GroupId, I32 *Index);

Visual Basic:

APS_get_feeder_running_index (ByVal GroupId As Long, Index As Long) As Long

Parameters:

I32 GroupId: Group ID. Value range: 0~1.

I32 *Index: Return which point is in operation.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Refer to the example of *APS_set_feeder_group*()

See also:

I32 FNTYPE APS_set_feeder_group(I32 GroupId, I32 Dimension, I32 *Axis_ID_Array);

I32 FNTYPE APS_get_feeder_group(I32 GroupId, I32 *Dimension, I32 *Axis_ID_Array);

I32 FNTYPE APS_free_feeder_group(I32 GroupId);

I32 FNTYPE APS_reset_feeder_buffer(I32 GroupId);

I32 FNTYPE APS_set_feeder_point_2D (I32 GroupId, PNT_DATA_2D * PtArray, I32 Size, I32 LastFlag);

I32 FNTYPE APS_start_feeder_move(I32 GroupId);

I32 FNTYPE APS_get_feeder_feed_index(I32 GroutId, I32 *Index);

APS_get_feeder_feed_index	Get which point is set into point table.

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8254/58 / AMP-204/8C

Descriptions:

This function will return which the latest buffer index in feeder is loaded into controller.

Syntax:

C/C++:

I32 FNTYPE APS_get_feeder_feed_index(I32 GroutId, I32 *Index);

Visual Basic:

APS_get_feeder_feed_index (ByVal GroupId As Long, Index As Long) As Long

Parameters:

I32 GroupId: Group ID. Value range: 0~1.

I32 *Index: Return which buffer index is load into controller.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Refer to the example of *APS_set_feeder_group*()

See also:

I32 FNTYPE APS_set_feeder_group(I32 GroupId, I32 Dimension, I32 *Axis_ID_Array);

I32 FNTYPE APS_get_feeder_group(I32 GroupId, I32 *Dimension, I32 *Axis_ID_Array);

I32 FNTYPE APS_free_feeder_group(I32 GroupId);

I32 FNTYPE APS_reset_feeder_buffer(I32 GroupId);

I32 FNTYPE APS_set_feeder_point_2D(I32 GroupId, PNT_DATA_2D * PtArray, I32 Size, I32 LastFlag);

I32 FNTYPE APS_start_feeder_move(I32 GroupId);

I32 FNTYPE APS_get_feeder_running_index(I32 GroupId, I32 *Index);

APS_set_feeder_ex_pause	Motion paused(stopped) and feeder paused

Support Products: PCI-8253/56 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to pauses move when running point table. When pause command is issued, it will decelerate to stop and turn off I/O. The feeder also will be paused at the same time.

Syntax:

C/C++:

I32 FNTYPE APS_set_feeder_ex_pause(I32 GroupId);

Visual Basic:

APS_set_feeder_ex_pause (ByVal GroupId As Long) As Long

Parameters:

I32 GroupId: Group ID. Value range: 0~1.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “APS168.h”
#include “ErrorCodeDef.h”

//When push pause button on user interface.
I32 ret;
I32 groupId = 0;
ret = APS_set_feeder_ex_pause( groupId );
if( ret != ERR_NoError ) {//Exception handling }
// Check the motion status has stopped.
…
```

See also:

I32 FNTYPE APS_set_feeder_ex_pause(I32 GroupId);

I32 FNTYPE APS_set_feeder_ex_rollback(I32 GroupId, I32 Max_Speed);

I32 FNTYPE APS_set_feeder_ex_resume(I32 GroupId);

APS_set_feeder_ex_rollback	Move back to the starting position of paused index

Support Products: PCI-8253/56 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to let the group of axes back to the last point position which is paused by *APS_set_feeder_ex_pause()*.

This function can **ONLY** be called after *APS_set_feeder_ex_pause()*. The behavior is not defined when this function is be used in other situation.

Syntax:

C/C++:

I32 FNTYPE APS_set_feeder_ex_rollback(I32 GroupId, I32 Max_Speed);

Visual Basic:

APS_set_feeder_ex_rollback(ByVal GroupId As Long, ByVal Max_Speed As Long) As Long

Parameters:

I32 GroupId: Group ID. Value range: 0~1.

I32 Max_Speed: Maximum linear interpolation speed. Value > 0, Unit: pulse/sec.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “APS168.h”
#include “ErrorCodeDef.h”

//When push“Go back”button on user interface.
I32 ret;
I32 groupId = 0;
I32 max_speed = 5000; // Pulse/sec
ret = APS_set_feeder_ex_rollback( groupId, max_speed );
if( ret != ERR_NoError ) {//Exception handling }
// Check the motion status has done.
…
```

See also:

I32 FNTYPE APS_set_feeder_ex_pause(I32 GroupId);

I32 FNTYPE APS_set_feeder_ex_resume(I32 GroupId);

APS_set_feeder_ex_resume	Resume the point-table move.

Support Products: PCI-8253/56 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to resume move from paused feeder running index. When passing through the pause position, it will keep I/O status.

This function can **ONLY** be called after *APS_set_table_move_ex_rollback*(). The behavior is not defined when this function is be used in other situation.

Syntax:

C/C++:

I32 FNTYPE APS_set_feeder_ex_resume (I32 GroupId);

Visual Basic:

APS_set_feeder_ex_resume (ByVal GroupId As Long) As Long

Parameters:

I32 GroupId: Group ID. Value range: 0~1.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#include “APS168.h”
#include “ErrorCodeDef.h”

//When push“Resume”button on user interface.
I32 ret;
I32 groupId = 0;
ret = APS_set_feeder_ex_resume ( groupId );
if( ret != ERR_NoError ) {//Exception handling }
// Check the motion status has started.
…
```

See also:

I32 FNTYPE APS_set_feeder_ex_pause(I32 GroupId);

I32 FNTYPE APS_set_feeder_ex_rollback(I32 GroupId, I32 Max_Speed);

15. Advanced Point table

APS_pt_enable	Enable point table.

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to enable a point table. User could set related axes of board to specified point table. In a specified board, it is forbidden to set repeat axis to point table.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_enable(I32 Board_ID, I32 PtbId, I32 Dimension, I32 *AxisArr);

Visual Basic:

APS_pt_enable (ByVal Board_ID As Long, ByVal PtbId As Long, ByVal Dimension As Long, ByVal AxisArr() As Long) As Long

Parameters:

For PCI-8254/58 / AMP-204/8C:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

I32 Dimension:

I32 *AxisArr: the axis array of a specified board is from 0 to N.

For PCI-8254, it is from 0 to 3.

For PCI-8258, it is from 0 to 7.

For PCIe-833x, ECAT-4XMO :

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

I32 Dimension: The number of the axis's dimension.

I32 *AxisArr: the axis array of a specified board is from 0 to (N-1). The N is actual total axis numbers in topology in a specified board when not using EtherCAT manual slave ID. In other words, axis array will be "axis id" array when using EtherCAT manual slave ID. E.g.

Mode in APS_initial	Slave ID and axis number	AxisArr assign value
Auto card ID(Bit 0 = 0) AxisNo auto mode(Bit 1 = 0) Fieldbus Slave Axis No auto mode(Bit 2 = 0) Fieldbus Slave No auto mode(Bit 10 = 0) Mode = 0x0000	Slave 0 - Axis 0 Slave 1 - Axis 1	AxisArr[0] = 0; AxisArr[1] = 1;
Manual card ID(Bit 0 = 1). Card ID = 15. AxisNo fixed mode(Bit 1 = 1) Fieldbus Slave Axis No auto mode(Bit 2 = 0) Fieldbus Slave No auto mode(Bit 10 = 0) Mode = 0x0003	Slave 0 - Axis 960(15 * 64 + 0) Slave 1 - Axis 961(15 * 64 + 1)	AxisArr[0] = 0; AxisArr[1] = 1;
Auto card ID(Bit 0 = 0). AxisNo auto mode(Bit 1 = 0) Fieldbus Slave Axis No auto mode(Bit 2 = 0) Fieldbus Slave No fixed mode(Bit 10 = 1) Mode = 0x0400	Slave 1000 - Axis 5555 Slave 2000 - Axis 5556 *StartAxisID set to “5555”	AxisArr[0] = 5555; AxisArr[1] = 5556;
Auto card ID(Bit 0 = 0). AxisNo auto mode(Bit 1 = 0) Fieldbus Slave Axis No fixed mode(Bit 2 = 1) Fieldbus Slave No fixed mode(Bit 10 = 1) Mode = 0x0404	Slave 1000 - Axis 1000 Slave 2000 - Axis 2000	AxisArr[0] = 1000; AxisArr[1] = 2000;
Manual card ID(Bit 0 = 1). Card ID = 15. AxisNo auto mode(Bit 1 = 0) Fieldbus Slave Axis No fixed mode(Bit 2 = 1) Fieldbus Slave No fixed mode(Bit 10 = 1) Mode = 0x0405	Slave 1000 - Axis 1000 Slave 2000 - Axis 2000	AxisArr[0] = 1000; AxisArr[1] = 2000;

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0
I32 Dimension = 2; //2D Dimension
I32 AxisArr[2] = { 0, 1 }; //Set Axis 0 & Axis 1 to point table 0
//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
ret = APS_pt_enable(Board_ID , PtbId, Dimension, & AxisArr ); //Enable point table 0
```

See also:

APS_pt_disable	Disable point table.

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to disable a point table.

Note: Don’t invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_disable (I32 Board_ID, I32 PtbId);

Visual Basic:

APS_pt_disable (ByVal Board_ID As Long, ByVal PtbId As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0

//Disable
ret = APS_pt_disable(Board_ID , PtbId ); //Disable point table 0
```

See also:

APS_get_pt_info	Get information of point table.

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get information of point table. User could get information of dimension and axis array. If point table is disabled, the return information of dimension is defined to 0.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_get_pt_info(I32 Board_ID, I32 PtbId, PPTINFO Info);

Visual Basic:

APS_get_pt_info (ByVal Board_ID As Long, ByVal PtbId As Long, ByRef Info As PTINFO) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

PPTINFO Info: A structure pointer for getting information of point table

```
typedef struct
{
   I32 Dimension; //How many dimension in spcfied point table
   I32 AxisArr[6]; //Axis array of point talbe. Maximun to 6 axes depended on dimension.
} PTINFO, *PPTINFO;
```

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0
PTINFO Info;

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
```

ret = APS_get_pt_info(Board_ID , PtbId, &Info); //Get information of point table 0

…..

See also:

APS_pt_set_vs	Set configuration of Vs to point table

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set started speed (Vs) to point table. When point table is moving, Vs is only applicated to first point.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_set_vs(I32 Board_ID, I32 PtbId, F64 Vs);

Visual Basic:

APS_pt_set_vs (ByVal Board_ID As Long, ByVal PtbId As Long, ByVal Vs As Double) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

F64 Vs: The started speed of point table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 ret;

I32 Board_ID = 0;

I32 PtbId = 0; //Point table 0

F64 Vs = 100.0;

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.

….

//Configure Vs to point table 0

ret = APS_pt_set_vs(Board_ID , PtbId, Vs);

.....

See also:

APS_pt_get_vs	Get configuration of Vs in the point table

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get started speed (Vs) of point table.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_get_vs(I32 Board_ID, I32 PtbId, F64 *Vs);

Visual Basic:

APS_pt_get_vs (ByVal Board_ID As Long, ByVal PtbId As Long, ByRef Vs As Double) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

F64 *Vs: The started speed of point table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0
F64 Vs;

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
….
//Get vs of point table 0
ret = APS_pt_get_vs(Board_ID , PtbId, &Vs );
…..
```

See also:

APS_pt_start	Start point table

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to start point table.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_start(I32 Board_ID, I32 PtbId);

Visual Basic:

APS_pt_start (ByVal Board_ID As Long, ByVal PtbId As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Configure Vs to point table 0, Push point to point table
//Start point table to move
ret = APS_pt_start(Board_ID , PtbId );
```

See also:

APS_pt_stop	Stop point table

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to stop point table.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_stop(I32 Board_ID, I32 PtbId);

Visual Basic:

APS_pt_stop (ByVal Board_ID As Long, ByVal PtbId As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0

//Stop point table to move
ret = APS_pt_stop(Board_ID , PtbId );
```

See also:

APS_get_pt_status	Get status of point table

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get status of point table. There are information including the state of point table, the point buffer status, the usage and free space of point buffer, and the running counts. The detail is described as follows:

- The state includes START and STOP states of point table.
- The buffer status includes FULL or EMPTY status of point buffer.
- The usage space is a counter of comsuming buffer size.
- The free space is a counter of remaining buffer size.
- The running count means how many points are executed after point table is enabled.

There are totally 50 point buffers in a point table. User could push a specified move, including line move, arc move, or helical move, into the buffer. Then, user could monitor status of point table for buffer status & running status. If buffer is not full, user could push more moves in to buffer. If buffer is already full, invoking Sleep() for a while to wait points to be comsumed.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don’t invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_get_pt_status(I32 Board_ID, I32 PtbId, PPTSTS Status);

Visual Basic:

APS_get_pt_status (ByVal Board_ID As Long, ByVal PtbId As Long, ByRef Status As PTSTS) As Long

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

PPTSTS Status: The status of point table

typedef struct

{

```
    U16 BitSts;       //b0: Is PT work? [1:working, 0:Stopped]
```

```
                    //b1: Is point buffer full? [1:full, 0:not full]
                    //b2: Is point buffer empty? [1:empty, 0:not empty]
                    //b3, b4, b5: Reserved for future, Don't care.
    U16 PntBufFreeSpace; // Free space of point buffer
    U16 PntBufUsageSpace; //Usage space of point buffer
    U32 RunningCnt; //How many points be executed after point table is enabled
} PTSTS, *PPTSTS;
```

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0
PTSTS Status;

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Get status of point table 0
ret = APS_get_pt_status(Board_ID , PtbId, &Status );
.....
```

See also:

APS_reset_pt_buffer	Reset related buffer of point table

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to reset some buffers of point table. There are three buffers in the point table, including a move buffer, a command buffer, and a profile buffer.

The move buffer could queue up to 50 moves.

The command buffer could control Do with a move.

The profile buffer could change speed profile with a move, including Acc, Dec, S-factor, Vm, Ve and so on.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_reset_pt_buffer(I32 Board_ID, I32 PtbId);

Visual Basic:

APS_reset_pt_buffer (ByVal Board_ID As Long, ByVal PtbId As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Reset buffer of point table 0
ret = APS_reset_pt_buffer(Board_ID , PtbId );
```

See also:

APS_pt_roll_back	Rollback to previous point

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to rollback to previous point. After invoking APS_pt_stop() to pause point table, user could rollback point table back to previous point. Then, re-start point table to execute unfinished moves by invoking APS_pt_start().

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_roll_back(I32 Board_ID, I32 PtbId, F64 Max_Speed);

Visual Basic:

APS_pt_roll_back (ByVal Board_ID As Long, ByVal PtbId As Long, ByVal Max_Speed As Double) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

F64 Max_Speed: Max speed by float.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0
F64 Max_Speed = 10000.0;

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Push points into point table.
//Start point table. Then, pause point table.
//Rollback to previous point
```

```
ret = APS_pt_roll_back( Board_ID, PtbId, Max_Speed );
//Then, restart point table.
```

See also:

APS_pt_get_error	Get error code of point table

Support Products: PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get error code of point table. If point table has error in operation, error code will be recorded. Error code will be reset when re-eanbling point table. Error code refers to “ErrorCodeDef.h” to get physical meanings.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don’t invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_get_error (I32 Board_ID, I32 PtbId, I32 *ErrCode);

Visual Basic:

APS_pt_get_error (ByVal Board_ID As Long, ByVal PtbId As Long, ByRef ErrCode As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

I32 *ErrCode: Error code of running point table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0
I32 ErrCode = 0;

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Get error code of running point table
ret = APS_get_pt_error(Board_ID , PtbId, &ErrCode );
.....
```

See also:

APS_pt_dwell	Push a dwell move into point buffer of point table.

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to push a dwell move into point buffer. There are up to 50 point buffer to pre-stored points in a point table. User could monitor usage / free space of point buffer to push moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_dwell(I32 Board_ID, I32 PtbId, PPTDWL Prof, PPTSTS Status);

Visual Basic:

APS_pt_dwell (ByVal Board_ID As Long, ByVal PtbId As Long, ByRef Prof As PTDWL, ByRef Status As PTSTS) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

PPTDWL Prof: The profile of dwell move

```
typedef struct
{
    F64     DwTime; //Set dwell time, unit is ms.
} PTDWL, *PPTDWL;
```

PPTSTS Status: The status of point table

```
typedef struct
{
    U16 BitSts;     //b0: Is PT work? [1:working, 0:Stopped]
                    //b1: Is point buffer full? [1:full, 0:not full]
                    //b2: Is point buffer empty? [1:empty, 0:not empty]
                    //b3, b4, b5: Reserved for future. Don't care.
                    //b6~: Be always 0
    U16 PntBufFreeSpace; // Free space of point buffer
    U16 PntBufUsageSpace; //Usage space of point buffer
```

```
        U32 RunningCnt; //How many points be executed after point table is enabled
    } PTSTS, *PPTSTS;
```

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0
PTDWL Prof;
PTSTS Status;

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Get status of point table 0
ret = APS_get_pt_status(Board_ID , PtbId, &Status );
if ( !( Status.BitSts & 0x02 ) ) //Point buffer is not full
{
   //Push move into point buffer
   Prof. DwTime = 100; //100ms
   ret = APS_pt_dwell( Board_ID, PtbId, &Prof, &Status );
}
//Start point table move
APS_pt_start( Board_ID, PtbId, 0 );
```

See also:

APS_pt_line	Push a line move into point buffer of point table.

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to push a line move into point buffer. There are up to 50 point buffer to pre-stored points in a point table. User could monitor usage / free space of point buffer to push moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_line(I32 Board_ID, I32 PtbId, PPTLINE Prof, PPTSTS Status);

Visual Basic:

APS_pt_line (ByVal Board_ID As Long, ByVal PtbId As Long, ByRef Prof As PTLINE, ByRef Status As PTSTS) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

PPTLINE Prof: The profile of line move

```
typedef struct
{
    I32     Dim; //dimension
    F64     Pos[6]; //position array for line move
} PTLINE, *PPTLINE;
```

PPTSTS Status: The status of point table

```
typedef struct
{
    U16 BitSts;     //b0: Is PT work? [1:working, 0:Stopped]
                    //b1: Is point buffer full? [1:full, 0:not full]
                    //b2: Is point buffer empty? [1:empty, 0:not empty]
                    //b3, b4, b5: Reserved for future. Don't care.
                    //b6~: Be always 0
    U16 PntBufFreeSpace; // Free space of point buffer
```

```
        U16 PntBufUsageSpace; //Usage space of point buffer
        U32 RunningCnt; //How many points be executed after point table is enabled
    } PTSTS, *PPTSTS;
```

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0
PTLINE Prof;
PTSTS Status;

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Get status of point table 0
ret = APS_get_pt_status(Board_ID , PtbId, &Status );
if ( !( Status.BitSts & 0x02 ) ) //Point buffer is not full
{
   //Push move into point buffer
   Prof.Dim = 2;
   Prof.Pos[0] = 10000;
   Prof.Pos[1] = 10000;
   ret = APS_pt_line( Board_ID, PtbId, &Prof, &Status );
}
//Start point table move
APS_pt_start( Board_ID, PtbId, 0 );
```

See also:

APS_pt_arc2_ca	Push a 2d arc move into point buffer of point table.

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to push a 2d arc move with angle into point buffer. There are up to 50 point buffer to pre-stored points in a point table. User could monitor usage / free space of point buffer to push moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_arc2_ca(I32 Board_ID, I32 PtbId, PPTA2CA Prof, PPTSTS Status);

Visual Basic:

APS_pt_arc2_ca (ByVal Board_ID As Long, ByVal PtbId As Long, ByRef Prof As PTA2CA, ByRef Status As PTSTS) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

PPTA2CA Prof: The profile of arc move

```
typedef struct
{
    U8      Index[2]; //[0 ~ dimension of point table] Which axis index in point table
    F64     Center[2]; //Center position
    F64     Angle; //Angle, unit is radian
} PTA2CA, *PPTA2CA;
```

PPTSTS Status: The status of point table

```
typedef struct
{
    U16 BitSts;     //b0: Is PT work? [1:working, 0:Stopped]
                    //b1: Is point buffer full? [1:full, 0:not full]
                    //b2: Is point buffer empty? [1:empty, 0:not empty]
                    //b3, b4, b5: Reserved for future. Don't care.
                    //b6~: Be always 0
```

```
        U16 PntBufFreeSpace; // Free space of point buffer
        U16 PntBufUsageSpace; //Usage space of point buffer
        U32 RunningCnt; //How many points be executed after point table is enabled
    } PTSTS, *PPTSTS;
```

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0
PTA2CA Prof;
PTSTS Status;

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Get status of point table 0
ret = APS_get_pt_status(Board_ID , PtbId, &Status );
if ( !( Status.BitSts & 0x02 ) ) //Point buffer is not full
{
   //Push move into point buffer
   Prof.Index[0] = 0; //pick dimension 0
   Prof.Index[1] = 1; //pick dimension 1
   Prof.Center[0] = 10000;
   Prof.Center[1] = 10000;
   Prof.Angle = 3.14159265;
   ret = APS_pt_arc2_ca( Board_ID, PtbId, &Prof, &Status );
}
//Start point table move
APS_pt_start( Board_ID, PtbId );
```

See also:

APS_pt_arc2_ce	Push a 2d arc move into point buffer of point table.

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to push a 2d arc move with end position into point buffer. There are up to 50 point buffer to pre-stored points in a point table. User could monitor usage / free space of point buffer to push moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_arc2_ce(I32 Board_ID, I32 PtbId, PPTA2CE Prof, PPTSTS Status);

Visual Basic:

APS_pt_arc2_ce (ByVal Board_ID As Long, ByVal PtbId As Long, ByRef Prof As PTA2CE, ByRef Status As PTSTS) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

PPTA2CE Prof: The profile of arc move

```
typedef struct
{
    U8      Index[2]; //[0 ~ dimension of point table] Which axis index in point table
    F64     Center[2]; //Center position
    F64     End[2]; //End position
    I16     Dir; //A value specifies the rotate direction, If Dir set 0 means rotate in
                 positive direction, dir = -1 rotate in negative direction.
} PTA2CE, *PPTA2CE;
```

PPTSTS Status: The status of point table

```
typedef struct
{
    U16 BitSts;     //b0: Is PT work? [1:working, 0:Stopped]
                    //b1: Is point buffer full? [1:full, 0:not full]
                    //b2: Is point buffer empty? [1:empty, 0:not empty]
```

```
                    //b3, b4, b5: Reserved for future. Don't care.
                    //b6~: Be always 0
        U16 PntBufFreeSpace; // Free space of point buffer
        U16 PntBufUsageSpace; //Usage space of point buffer
        U32 RunningCnt; //How many points be executed after point table is enabled
    } PTSTS, *PPTSTS;
```

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0
PTA2CE Prof;
PTSTS Status;

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Get status of point table 0
ret = APS_get_pt_status(Board_ID , PtbId, &Status );
if ( !( Status.BitSts & 0x02 ) ) //Point buffer is not full
{
   //Push move into point buffer
   Prof.Index[0] = 0; //pick dimension 0
   Prof.Index[1] = 1; //pick dimension 1
   Prof.Center[0] = 10000;
   Prof.Center[1] = 10000;
   Prof.End[0] = 0;
   Prof.End[1] = 0;
   Prof.Dir = 0; //Positvie direction
  ret = APS_pt_arc2_ce( Board_ID, PtbId, &Prof, &Status );
}

//Start point table move
APS_pt_start( Board_ID, PtbId );
```

See also:

APS_pt_arc3_ca	Push a 3d arc move into point buffer of point table.

Support Products: PCI-8254/58 / AMP-204/8C, ECAT-4XMO

Descriptions:

This function is used to push a 3d arc move with angle into point buffer. There are up to 50 point buffer to pre-stored points in a point table. User could monitor usage / free space of point buffer to push moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_arc3_ca(I32 Board_ID, I32 PtbId, PPTA3CA Prof, PPTSTS Status);

Visual Basic:

APS_pt_arc3_ca (ByVal Board_ID As Long, ByVal PtbId As Long, ByRef Prof As PTA3CA, ByRef Status As PTSTS) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

PPTA2CA Prof: The profile of 3d arc move with angle

```
typedef struct
{
    U8      Index[3]; //[0 ~ dimension of point table] Which axis index in point table
    F64     Center[3]; //Center position
    F64     Noraml[3]; //Normal vector
    F64     Angle; //Angle, unit is radian
} PTA3CA, *PPTA3CA;
```

PPTSTS Status: The status of point table

```
typedef struct
{
    U16 BitSts;     //b0: Is PT work? [1:working, 0:Stopped]
                    //b1: Is point buffer full? [1:full, 0:not full]
                    //b2: Is point buffer empty? [1:empty, 0:not empty]
                    //b3, b4, b5: Reserved for future. Don't care.
```

```
                                    //b6~: Be always 0
            U16 PntBufFreeSpace; // Free space of point buffer
            U16 PntBufUsageSpace; //Usage space of point buffer
            U32 RunningCnt; //How many points be executed after point table is enabled
        } PTSTS, *PPTSTS;
```

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0
PTA3CA Prof;
PTSTS Status;

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Get status of point table 0
ret = APS_get_pt_status(Board_ID , PtbId, &Status );
if ( !( Status.BitSts & 0x02 ) ) //Point buffer is not full
{
   //Push move into point buffer
   Prof.Index[0] = 0; //pick dimension index 0
   Prof.Index[1] = 1; //pick dimension index 1
   Prof.Index[2] = 2; //pick dimension index 2
   Prof.Center[0] = 10000;
   Prof.Center[1] = 10000;
   Prof.Center[2] = 10000;
   Prof.Normal[0] = 0;
   Prof.Normal[1] = 0;
   Prof.Normal[2] = 1;
   Prof.Angle = 3.14159265; //In radian
  ret = APS_pt_arc3_ca( Board_ID, PtbId, &Prof, &Status );
}

//Start point table move
APS_pt_start( Board_ID, PtbId );
```

See also:

APS_pt_arc3_ce	Push a 3d arc move into point buffer of point table.

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to push a 3d arc move with end position into point buffer. There are up to 50 point buffer to pre-stored points in a point table. User could monitor usage / free space of point buffer to push moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_arc3_ce(I32 Board_ID, I32 PtbId, PPTA3CE Prof, PPTSTS Status);

Visual Basic:

APS_pt_arc3_ce (ByVal Board_ID As Long, ByVal PtbId As Long, ByRef Prof As PTA3CE, ByRef Status As PTSTS) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

PPTA3CE Prof: The profile of 3d arc move with end position

```
typedef struct
{
    U8      Index[3]; //[0 ~ dimension of point table] Which axis index in point table
    F64     Center[3]; //Center position
    F64     End[3]; //End position
    I16     Dir; //A value specifies the rotate direction, If Dir set 0 means rotate in
                 positive direction, dir = -1 rotate in negative direction.
} PTA3CE, *PPTA3CE;
```

PPTSTS Status: The status of point table

```
typedef struct
{
    U16 BitSts;     //b0: Is PT work? [1:working, 0:Stopped]
                    //b1: Is point buffer full? [1:full, 0:not full]
                    //b2: Is point buffer empty? [1:empty, 0:not empty]
```

```
                    //b3, b4, b5: Reserved for future. Don't care.
                    //b6~: Be always 0
        U16 PntBufFreeSpace; // Free space of point buffer
        U16 PntBufUsageSpace; //Usage space of point buffer
        U32 RunningCnt; //How many points be executed after point table is enabled
} PTSTS, *PPTSTS;
```

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Refer to APS_pt_arc3_ca().

See also:

APS_pt_spiral_ca	Push a helical move into point buffer of point table.

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to push a helical move with angle into point buffer. There are up to 50 point buffer to pre-stored points in a point table. User could monitor usage / free space of point buffer to push moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_spiral_ca(I32 Board_ID, I32 PtbId, PPTHCA Prof, PPTSTS Status);

Visual Basic:

APS_pt_spiral_ca (ByVal Board_ID As Long, ByVal PtbId As Long, ByRef Prof As PTHCA, ByRef Status As PTSTS) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

PPTHCA Prof: The profile of helical move with angle

```
typedef struct
{
    U8      Index[3]; //[0 ~ dimension of point table] Which axis index in point table
    F64     Center[3]; //Center position
    F64     Noraml[3]; //Normal vector
    F64     Angle; //Angle, unit is radian
    F64     DeltaH; //The height of helical move, User unit,
    F64     FinalR; //The distant from end position to normal vector, User unit
} PTHCA, *PPTHCA;
```

PPTSTS Status: The status of point table

```
typedef struct
{
    U16 BitSts;     //b0: Is PT work? [1:working, 0:Stopped]
                    //b1: Is point buffer full? [1:full, 0:not full]
```

```
                    //b2: Is point buffer empty? [1:empty, 0:not empty]
                    //b3, b4, b5: Reserved for future. Don't care.
                    //b6~: Be always 0
        U16 PntBufFreeSpace; // Free space of point buffer
        U16 PntBufUsageSpace; //Usage space of point buffer
        U32 RunningCnt; //How many points be executed after point table is enabled
} PTSTS, *PPTSTS;
```

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Refer to APS_pt_arc3_ca().

See also:

APS_pt_spiral_ce	Push a helical move into point buffer of point table.

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to push a helical move with end position into point buffer. There are up to 50 point buffer to pre-stored points in a point table. User could monitor usage / free space of point buffer to push moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_spiral_ce(I32 Board_ID, I32 PtbId, PPTHCE Prof, PPTSTS Status);

Visual Basic:

APS_pt_spiral_ce (ByVal Board_ID As Long, ByVal PtbId As Long, ByRef Prof As PTHCE, ByRef Status As PTSTS) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

PPTHCE Prof: The profile of helical move with end position

```
typedef struct
{
    U8      Index[3]; //[0 ~ dimension of point table] Which axis index in point table
    F64     Center[3]; //Center positioi
    F64     Noraml[3]; //Normal vector
    F64     End[3]; //End position
    I16     Dir; //A value specifies the rotate direction, If Dir set 0 means rotate in
                 positive direction, dir = -1 rotate in negative direction.
} PTHCE, *PPTHCE;
```

PPTSTS Status: The status of point table

```
typedef struct
{
    U16 BitSts;     //b0: Is PT work? [1:working, 0:Stopped]
                    //b1: Is point buffer full? [1:full, 0:not full]
```

```
                    //b2: Is point buffer empty? [1:empty, 0:not empty]
                    //b3, b4, b5: Reserved for future. Don't care.
                    //b6~: Be always 0
        U16 PntBufFreeSpace; // Free space of point buffer
        U16 PntBufUsageSpace; //Usage space of point buffer
        U32 RunningCnt; //How many points be executed after point table is enabled
    } PTSTS, *PPTSTS;
```

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Refer to APS_pt_arc3_ca().

See also:

APS_pt_ext_set_do_ch	Set do extension command into command buffer. Command buffer is active when pushing a move into point table./ Control digital output within advanced point-table in EPS-6000 slave

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x

Descriptions:

For PCI-8254/58 / AMP-204/8C , this function is used to set do extension command into command buffer. Command buffer is active when pushing a move into point table. After pushing a move, command buffer will be automatically cleared. User could write up to 7 commands into command buffer. Then, pushing a move into point table will take those commands into point table together. Now, do command is supported. Other commands are reserved for future.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

For PCIe-833x , this function only support a ADLINK EPS-6000 slave within digital output module in topology and this slave should be placed at the first position in topology. This function is used to control digital output when execute advanced point-table function. User can use this API to control digital output when the command position reach to the target position.

Syntax:

C/C++:

I32 FNTYPE APS_pt_ext_set_do_ch(I32 Board_ID, I32 PtbId, I32 Channel, I32 OnOff);

Visual Basic:

APS_pt_ext_set_do_ch (ByVal Board_ID As Long, ByVal PtbId As Long, ByVal Channel As Long, ByVal OnOff As Long) As Long

Parameters:

For PCI-8254/58 / AMP-204/8C:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

I32 Channel: Do channel

I32 OnOff: Do on/off. 1: On, 0: Off.

For PCIe-833x :

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

I32 Channel: the channel number of digital output module.

I32 OnOff: 0: set digital output value to 0.

1: set digital output value to 1.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Set do extension command to command buffer //Set do channel 0 to turn on
ret = APS_pt_ext_set_do_ch( Board_ID, PtbId, 0, 1 );
//Push a move into point buffer
```

See also:

APS_pt_ext_set_table_no	Set VAO table No. extension command into command buffer. Command buffer is active when pushing a move into point table.

Support Products: PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to set VAO table No. extension command into command buffer. Command buffer is active when pushing a move into point table. After pushing a move, command buffer will be automatically cleared. User could write up to 7 commands into command buffer. Then, pushing a move into point table will take those commands into point table together. Now, table No. command is supported. Other commands are reserved for future.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_ext_set_table_no(I32 Board_ID, I32 PtbId, I32 CtrlNo, I32 TableNo);

Visual Basic:

APS_pt_ext_set_table_no (ByVal Board_ID As Long, ByVal PtbId As Long, ByVal CtrlNo As Long, ByVal TableNo As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

I32 CtrlNo: Control No. is from 0 to 1.

I32 TableNo: VAO Table No. is from -1 to 7.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0
```

```
//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Set table No. extension command to command buffer //Set VAO table No. to 0, and set
control No. to 0
ret = APS_pt_ext_set_table_no( Board_ID, PtbId, 0, 0 );
//Push a move into point buffer
```

See also:

APS_pt_set_absolute	Set absolute profile into profile buffer.

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set absolute profile into profile buffer. Profile buffer is active when pushing a move into point table. User usually sets profile all together in the beginning after enabling point table. If user wants to change some profile, user could modify them before pushing a move into point buffer. On the contrary, if user doesn't want to modify profile, the same profile will be automatically maintained for following moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.
Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:
I32 FNTYPE APS_pt_set_absolute (I32 Board_ID, I32 PtbId);
Visual Basic:
APS_pt_set_absolute (ByVal Board_ID As Long, ByVal PtbId As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().
I32 PtbId: the point table id is from 0 to 1.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Set absolute profile to profilr buffer
ret = APS_pt_set_absolute ( Board_ID, PtbId );
//Push a move into point buffer
```

See also:

APS_pt_set_relative	Set relative profile into profile buffer.

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set relative profile into profile buffer. Profile buffer is active when pushing a move into point table. User usually sets profile all together in the beginning after enabling point table. If user wants to change some profile, user could modify them before pushing a move into point buffer. On the contrary, if user doesn't want to modify profile, the same profile will be automatically maintained for following moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_set_relative (I32 Board_ID, I32 PtbId);

Visual Basic:

APS_pt_set_relative (ByVal Board_ID As Long, ByVal PtbId As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0
//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Set relative profile to profilr buffer
ret = APS_pt_set_relative ( Board_ID, PtbId );
//Push a move into point buffer
```

See also:

APS_pt_set_trans_buffered	Set transition to buffer mode in profile buffer

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set transition to buffer mode in profile buffer. Profile buffer is active when pushing a move into point table. User usually sets profile all together in the beginning after enabling point table. If user wants to change some profile, user could modify them before pushing a move into point buffer. On the contrary, if user doesn't want to modify profile, the same profile will be automatically maintained for following moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_set_trans_buffered(I32 Board_ID, I32 PtbId);

Visual Basic:

APS_pt_set_trans_buffered (ByVal Board_ID As Long, ByVal PtbId As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0
//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
// Set to buffered mode
ret = APS_pt_set_trans_buffered( Board_ID, PtbId );
//Push a move into point buffer
```

See also:

APS_pt_set_trans_inp	Set transition to in-position mode in profile buffer

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set transition to in-position mode in profile buffer. Profile buffer is active when pushing a move into point table. User usually sets profile all together in the beginning after enabling point table. If user wants to change some profile, user could modify them before pushing a move into point buffer. On the contrary, if user doesn't want to modify profile, the same profile will be automatically maintained for following moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_set_trans_inp(I32 Board_ID, I32 PtbId);

Visual Basic:

APS_pt_set_trans_inp (ByVal Board_ID As Long, ByVal PtbId As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0
//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
// Set to in-position mode
ret = APS_pt_set_trans_inp( Board_ID, PtbId );
//Push a move into point buffer
```

See also:

APS_pt_set_trans_blend_dec	Set transition to blending mode with deceleration in profile buffer

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set transition to blending mode with deceleration in profile buffer. Profile buffer is active when pushing a move into point table. User usually sets profile all together in the beginning after enabling point table. If user wants to change some profile, user could modify them before pushing a move into point buffer. On the contrary, if user doesn't want to modify profile, the same profile will be automatically maintained for following moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.
Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:
I32 FNTYPE APS_pt_set_trans_blend_dec(I32 Board_ID, I32 PtbId, F64 Bp);
Visual Basic:
APS_pt_set_trans_blend_dec (ByVal Board_ID As Long, ByVal PtbId As Long, ByVal Bp As Double) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().
I32 PtbId: the point table id is from 0 to 1.
F64 Bp: Deceleration rate. [Bp > 0, unit/s^2]

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
// Set to blending mode with deceleration.
```

```
ret = APS_pt_set_trans_blend_dec( Board_ID, PtbId, 10000 );
//Push a move into point buffer
```

See also:

APS_pt_set_trans_blend_dist	Set transition to blending mode with residue distant in profile buffer

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set transition to blending mode with residue distant in profile buffer. Profile buffer is active when pushing a move into point table. User usually sets profile all together in the beginning after enabling point table. If user wants to change some profile, user could modify them before pushing a move into point buffer. On the contrary, if user doesn't want to modify profile, the same profile will be automatically maintained for following moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.
Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:
I32 FNTYPE APS_pt_set_trans_blend_dist(I32 Board_ID, I32 PtbId, F64 Bp);
Visual Basic:
APS_pt_set_trans_blend_dist (ByVal Board_ID As Long, ByVal PtbId As Long, ByVal Bp As Double) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().
I32 PtbId: the point table id is from 0 to 1.
F64 Bp: Residue distance. Unit is user unit, generally is pulse. [Bp >= 0]

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
// Set to blending mode with residue distant.
```

```
ret = APS_pt_set_trans_blend_dist( Board_ID, PtbId, 100 );
//Push a move into point buffer
```

See also:

APS_pt_set_trans_blend_pcnt	Set transition to blending mode with residue distant percentage in profile buffer

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set transition to blending mode with residue distant percentage in profile buffer. Profile buffer is active when pushing a move into point table. User usually sets profile all together in the beginning after enabling point table. If user wants to change some profile, user could modify them before pushing a move into point buffer. On the contrary, if user doesn't want to modify profile, the same profile will be automatically maintained for following moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_set_trans_blend_pcnt(I32 Board_ID, I32 PtbId, F64 Bp);

Visual Basic:

APS_pt_set_trans_blend_pcnt (ByVal Board_ID As Long, ByVal PtbId As Long, ByVal Bp As Double) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

F64 Bp: Residue distance in travel distance's percentage. Unit is %. [Bp: 0.0 ~ 1.0]

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
```

```
// Set to blending mode with residue distant percentage
ret = APS_pt_set_trans_blend_pcnt( Board_ID, PtbId, 0.05 );
//Push a move into point buffer
```

See also:

APS_pt_set_acc	Set accerlation profile into profile buffer.

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set acceleration profile into profile buffer. Profile buffer is active when pushing a move into point table. User usually sets profile all together in the beginning after enabling point table. If user wants to change some profile, user could modify them before pushing a move into point buffer. On the contrary, if user doesn't want to modify profile, the same profile will be automatically maintained for following moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.
Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:
I32 FNTYPE APS_pt_set_acc(I32 Board_ID, I32 PtbId, F64 Acc);
Visual Basic:
APS_pt_set_acc (ByVal Board_ID As Long, ByVal PtbId As Long, ByVal Acc As Double) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().
I32 PtbId: the point table id is from 0 to 1.
F64 Acc: Acceleration rate. [unit/s^2, > 0]

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Set acceleration to 10000
```

```
ret = APS_pt_set_acc( Board_ID, PtbId, 10000 );
//Push a move into point buffer
```

See also:

APS_pt_set_dec	Set deceleration profile into profile buffer.

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set deceleration profile into profile buffer. Profile buffer is active when pushing a move into point table. User usually sets profile all together in the beginning after enabling point table. If user wants to change some profile, user could modify them before pushing a move into point buffer. On the contrary, if user doesn't want to modify profile, the same profile will be automatically maintained for following moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.
Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:
I32 FNTYPE APS_pt_set_dec(I32 Board_ID, I32 PtbId, F64 Dec);
Visual Basic:
APS_pt_set_dec (ByVal Board_ID As Long, ByVal PtbId As Long, ByVal Dec As Double) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().
I32 PtbId: the point table id is from 0 to 1.
F64 Dec: Deceleration rate. [unit/s^2, > 0]

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Set deceleration to 10000
```

```
ret = APS_pt_set_dec( Board_ID, PtbId, 10000 );
//Push a move into point buffer
```

See also:

APS_pt_set_acc_dec	Set accerlation & deceleration profile into profile buffer.

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set acceleration and deceleration profile into profile buffer. Profile buffer is active when pushing a move into point table. User usually sets profile all together in the beginning after enabling point table. If user wants to change some profile, user could modify them before pushing a move into point buffer. On the contrary, if user doesn't want to modify profile, the same profile will be automatically maintained for following moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.
Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:
I32 FNTYPE APS_pt_set_acc_dec(I32 Board_ID, I32 PtbId, F64 AccDec);
Visual Basic:
APS_pt_set_acc_dec (ByVal Board_ID As Long, ByVal PtbId As Long, ByVal AccDec As Double) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().
I32 PtbId: the point table id is from 0 to 1.
F64 AccDec: Acceleration/deceleration rate. [unit/s^2, > 0]

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Set acceleration / deceleration to 10000
```

```
ret = APS_pt_set_acc_dec( Board_ID, PtbId, 10000 );
//Push a move into point buffer
```

See also:

APS_pt_set_s	Set S-factor profile into profile buffer.

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set S-factor profile into profile buffer. Profile buffer is active when pushing a move into point table. User usually sets profile all together in the beginning after enabling point table. If user wants to change some profile, user could modify them before pushing a move into point buffer. On the contrary, if user doesn't want to modify profile, the same profile will be automatically maintained for following moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.

Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:

I32 FNTYPE APS_pt_set_s(I32 Board_ID, I32 PtbId, F64 Sf);

Visual Basic:

APS_pt_set_s (ByVal Board_ID As Long, ByVal PtbId As Long, ByVal Sf As Double) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PtbId: the point table id is from 0 to 1.

F64 Sf: s-factor [0 ~ 1]

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Set s-factor to 0.5
ret = APS_pt_set_s( Board_ID, PtbId, 0.5 );
//Push a move into point buffer
```

See also:

APS_pt_set_vm	Set maximum velocity profile into profile buffer.

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set maximum velocity profile into profile buffer. Profile buffer is active when pushing a move into point table. User usually sets profile all together in the beginning after enabling point table. If user wants to change some profile, user could modify them before pushing a move into point buffer. On the contrary, if user doesn't want to modify profile, the same profile will be automatically maintained for following moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.
Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:
I32 FNTYPE APS_pt_set_vm(I32 Board_ID, I32 PtbId, F64 Vm);
Visual Basic:
APS_pt_set_vm (ByVal Board_ID As Long, ByVal PtbId As Long, ByVal Vm As Double) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().
I32 PtbId: the point table id is from 0 to 1.
F64 Vm : Max. velocity [Vm >= 0]

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Set Vm to 10000
```

```
ret = APS_pt_set_vm( Board_ID, PtbId, 10000 );
//Push a move into point buffer
```

See also:

APS_pt_set_ve	Set end velocity profile into profile buffer.

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to set end velocity profile into profile buffer. Profile buffer is active when pushing a move into point table. User usually sets profile all together in the beginning after enabling point table. If user wants to change some profile, user could modify them before pushing a move into point buffer. On the contrary, if user doesn't want to modify profile, the same profile will be automatically maintained for following moves.

Note: It is necessary to invoke APS_pt_enable() first to enable point table. Otherwise, it will be return error code.
Note: Don't invoke Pt functions with Feeder fuctions at the same time. They are exclusive.

Syntax:

C/C++:
I32 FNTYPE APS_pt_set_ve(I32 Board_ID, I32 PtbId, F64 Ve);
Visual Basic:
APS_pt_set_ve (ByVal Board_ID As Long, ByVal PtbId As Long, ByVal Ve As Double) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().
I32 PtbId: the point table id is from 0 to 1.
F64 Ve: end velocity [Ve >= 0]

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 PtbId = 0; //Point table 0

//Enable point table 0 to 2d dimension with aixs 0 and axis 1.
//Set Ve to 100
ret = APS_pt_set_ve( Board_ID, PtbId, 100 );
```

//Push a move into point buffer

See also:

16. Field bus functions

APS_set_field_bus_param	Set field bus related parameters

Support Products: PCI-8392H, DPAC-3000, PCI(e)-7856, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, HSL-DIO

Descriptions:

This function is used to set field bus system parameters. Users must use this function before starting field bus communication. Otherwise, the field bus will be started by default. For parameter details, you can refer to field bus parameter table.

The field bus is a kind of serial network bus using in industrial field. The most popular one is CAN bus.

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_param(I32 Board_ID, I32 BUS_No, I32 BUS_Param_No, I32 BUS_Param);

Visual Basic:

APS_set_field_bus_param(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal BUS_Param_No As Long, ByVal BUS_Param As Long)As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 BUS_Param_No: Field bus parameter number, Refer to table 569peration569

I32 BUS_Param: Field bus parameter data. Refer to table definition.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_get_field_bus_param();APS_start_field_bus()

APS_get_field_bus_param	Get field bus related parameters

Support Products: PCI-8392H, DPAC-3000, PCI(e)-7856, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, HSL-DIO

Descriptions:

This function is used to get field bus system parameters. Please refer to field bus parameter table.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_param(I32 Board_ID, I32 BUS_No, I32 BUS_Param_No, I32 *BUS_Param);

Visual Basic:

APS_get_field_bus_param(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal BUS_Param_No As Long, BUS_Param As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 BUS_Param_No: Field bus parameter number, Refer to table 570peration570

I32 *BUS_Param: Return field bus parameter data. Refer to table definition.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_set_field_bus_param()

APS_scan_field_bus	Scan field bus and generate ENI file

Support Products: PCIe-833x

Descriptions:

This function is used to scan field bus and generate ENI file. In the first time to use, user should call this function before use APS_start_field_bus() .

Syntax:

C/C++:

I32 FNTYPE APS_scan_field_bus(I32 Board_ID, I32 BUS_No)

Visual Basic:

APS_scan_field_bus (ByVal Board_ID As Long, ByVal BUS_No As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: only support number 0.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

CASE 1:

For first time to use PCIe-833x (The ENI file doesn't exist)

```
APS_scan_field_bus(    0, 0 );    // scan field bus and generate ENI file firstly
APS_start_field_bus( 0, 0, 0 );    // start field bus communication
…
Field bus operation…
Do something….
….
APS_stop_field_bus( 0, 0 );    // stop field bus communication
```

CASE 2:

The ENI file does exist and the topology does not change.

```
APS_start_field_bus( 0, 0, 0 );    // start field bus communication
…
Field bus operation…
```

Do something….

….

APS_stop_field_bus(0, 0); // stop field bus communication

CASE 3:

The ENI file does exist and the topology does change.

APS_scan_field_bus(0, 0); // scan field bus and generate new ENI file firstly

APS_start_field_bus(0, 0, 0); // start field bus communication

…

Field bus operation…

Do something….

….

APS_stop_field_bus(0, 0); // stop field bus communication

See also:

APS_start_field_bus(); APS_stop_field_bus()

Note: Before ENI generation process, this function will remove EniBuilderForCpp.log and ADLINK_Config2.xml respectively. If ENI generation process successes, user can find the ADLINK_Config2.xml (ENI file) in ENI folder for connecting EtherCAT network. Otherwise, this function will return an error code and user can refer the EniBuilderForCpp.log for details.

APS_start_field_bus	Start the network of specified field bus

Support Products: PCI-8392H, DPAC-3000, PCI(e)-7856, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, HSL-DIO, PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to start field bus communication. Once it is started, it will search all modules connected to the port. Because there could be motion slaves on the port, users should assign a starting axis ID when using this function. All axes of the port will start axis ID arrangement from the starting axis ID.

You should call this function before using field bus even you have only I/O slaves on the port. Notice that because the slaves are automatically searched, some slaves may be lost due to communication quality. Users must check all the slaves are found and types are correct before field bus opertation.

APS_stop_field_bus() must be called at the end of filed bus operation.

For PCIe-833x, ECAT-4XMO , ECAT-TRG4 , this function is used to start field bus communication. Once it is started, it will search all modules connected to the port. User should call this function before using field bus.

APS_stop_field_bus() must be called at the end of filed bus operation.

In the first time to use PCIe-833x, user should call APS_scan_field_bus() before use APS_start_field_bus() .

Syntax:

C/C++:

I32 FNTYPE APS_start_field_bus(I32 Board_ID, I32 BUS_No, I32 Starting_Axis_ID);

Visual Basic:

APS_start_field_bus(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal Starting_Axis_ID As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

For PCI(e)-7856, HSL field bus is Bus_No 0 and MNET field bus is Bus_No 1.

I32 Starting_Axis_ID: Starting axis ID number of this field bus number.

For PCIe-833x or ECAT-4XMO , ECAT-TRG4:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: only support number 0.

I32 Starting_Axis_ID: Don't care.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example1:

```
I32 ret; //Return error code.
I32 boardId = 0;
I32 busNum = 0; //Bus number.
I32 startingAxisId = 1000; //Startin axis ID of the filed bus.

Ret = APS_start_field_bus( boardId, busNum, startingAxisId );
// Field bus operation…
APS_stop_field_bus(boardId, busNum ); //Stop field bus.
```

Example2:

CASE 1:

```
For first time to use PCIe-833x (The ENI file doesn't exist)
APS_scan_field_bus(    0, 0 );    // scan field bus and generate ENI file firstly
APS_start_field_bus( 0, 0, 0 );    // start field bus communication
…
Field bus operation…
Do something….
….
APS_stop_field_bus( 0, 0 );    // stop field bus communication
```

CASE 2:

```
The ENI file does exist and the topology does not change.
APS_start_field_bus( 0, 0, 0 );    // start field bus communication
…
Field bus operation…
Do something….
….
APS_stop_field_bus( 0, 0 );    // stop field bus communication
```

CASE 3:

The ENI file does exist and the topology does change.

```
APS_scan_field_bus(    0, 0 );    // scan field bus and generate new ENI file firstly
APS_start_field_bus( 0, 0, 0 );    // start field bus communication
…
Field bus operation…
Do something….
....
APS_stop_field_bus( 0, 0 );    // stop field bus communication
```

See also:

APS_stop_field_bus()

APS_stop_field_bus	Stop the network of specified field bus

Support Products: PCI-8392H, DPAC-3000, PCI(e)-7856, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, HSL-DIO , PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to stop field bus communication and release its resource.

This function must be called at end of process, if user ever used APS_start_field_bus() to start network.

Syntax:

C/C++:

I32 FNTYPE APS_stop_field_bus(I32 Board_ID, I32 BUS_No);

Visual Basic:

APS_stop_field_bus(ByVal Board_ID As Long, ByVal BUS_No As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

For PCI(e)-7856, HSL field bus is Bus_No 0 and MNET field bus is Bus_No 1

For PCIe-833x or ECAT-4XMO , ECAT-TRG4,I32 BUS_No: Field bus number.(Port number) value: only support number 0.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example1:

```
I32 ret; //Return error code.
I32 boardId = 0;
I32 busNum = 0; //Bus number.
I32 startingAxisId = 1000; //Startin axis ID of the filed bus.

Ret = APS_start_field_bus(boardId, busNum, startingAxisId );
// Field bus operation…
APS_stop_field_bus(boardId, busNum ); //Stop field bus.
```

Example2:

CASE 1:

For first time to use PCIe-833x (The ENI file doesn't exist)

```
APS_scan_field_bus(    0, 0 );    // scan field bus and generate ENI file firstly
APS_start_field_bus( 0, 0, 0 );    // start field bus communication
…
Field bus operation…
Do something….
….
APS_stop_field_bus( 0, 0 );    // stop field bus communication
```

CASE 2:

The ENI file does exist and the topology does not change.

```
APS_start_field_bus( 0, 0, 0 );    // start field bus communication
…
Field bus operation…
Do something….
….
APS_stop_field_bus( 0, 0 );    // stop field bus communication
```

CASE 3:

The ENI file does exist and the topology does change.

```
APS_scan_field_bus(    0, 0 );    // scan field bus and generate new ENI file firstly
APS_start_field_bus( 0, 0, 0 );    // start field bus communication
…
Field bus operation…
Do something….
….
APS_stop_field_bus( 0, 0 );    // stop field bus communication
```

See also:

APS_start_field_bus()

APS_field_bus_d_set_output	Set field bus digital output

Support Products: PCI-8392H, DPAC-3000, PCI(e)-7856, MNET-4XMO-(C) , MNET-1XMO , HSL-4XMO, HSL-DIO

Descriptions:

This function is used to set field bus digital output on slave modules. The maximum data length of one module ID is 32-bit. If the module ID has fewer channels than 32, the higher bit must be remained zero when outputting. The read back data of higher bit will be zero

Notice: For HSL_DI56DO32_FCN module, users should call APS_field_bus_d_set_output_ex() for 64 bits DIO operation.

Syntax:

C/C++:

I32 FNTYPE APS_field_bus_d_set_output(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 DO_Value);

Visual Basic:

APS_field_bus_d_set_output(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal DO_Value As Long)As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For High Speed Link(HSL) type field bus, the range of module number is 1 to 63. Note: In HSL, the MOD_No is the first id occupied by the module.

For MNET type field bus, the range of module number is 0 to 63.

I32 DO_Value: Digital output value. In bit format. Bit 0 corresponding to digital output channel 0 and the rest may be deduced by analogy.

For MNET-4XMO the definitions of DO bits are as follows. The default value is 0xff.

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOIF4.Do2	IOIF3.Do2	IOIF2.Do2	IOIF1.Do2	IOIF4.Do1	IOIF3.Do1	IOIF2.Do1	IOIF1.Do1

For MNET-4XMO-C and HSL-4XMO, the definitions of DO bits are as follows. The default value is 0xf.

Bit3	Bit2	Bit1	Bit0
IOIF4.Do1	IOIF3.Do1	IOIF2.Do1	IOIF1.Do1

For MNET-1XMO the definitions of DO bits are as follows. The default value is 0x0.

Bit3	Bit2	Bit1	Bit0
N/A	SZST 0(Low) 1(High)	STL 0(Low) 1(High)	AlmReset 0(Low) 1(High)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret; //return error code.
I32 boardId = 0;
I32 busNum = 0;
I32 moduleNum = 0;
I32 DO_Value = 0;

//Start Field bus first.
//    ret = APS_start_field_bus( boardId, busNum, startingAxisId );
DO_Value = 0xF;
ret = APS_field_bus_d_set_output(boardId, busNum,, moduleNum, DO_Value );
```

See also:

APS_field_bus_d_get_output()

APS_field_bus_d_get_output	Get field bus digital output

Support Products: PCI-8392H, DPAC-3000, PCI(e)-7856, MNET-4XMO-(C) , MNET-1XMO, HSL-4XMO, HSL-DIO

Descriptions:

This function is used to get field bus digital output on slave modules. Some module ID can't be read back the output information. Please check each module's hardware specification. The maximum data length of one module ID is 32-bit. If the module ID has fewer channels than 32, the higher bit must be remained zero when outputting. The read back data of higher bit will be zero.

Notice: For HSL_DI56DO32_FCN module, users should call APS_field_bus_d_get_output_ex() for 64 bits DIO operation.

Syntax:

C/C++:

I32 FNTYPE APS_field_bus_d_get_output(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 *DO_Value);

Visual Basic:

APS_field_bus_d_get_output(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, DO_Value As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For High Speed Link(HSL) type field bus, the range of module number is 1 to 63. Note: In HSL, the Module_No is the first id occupied by the module.

For MNET type field bus, the range of module number is 0 to 63.

I32 *DO_Value: Return digital output value. Bit 0 corresponding to digital output channel 0 and the rest may be deduced by analogy.

For MNET-4XMO, the definitions of DO bits are as follows. The default value is 0xff.

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOIF4.Do2	IOIF3.Do2	IOIF2.Do2	IOIF1.Do2	IOIF4.Do1	IOIF3.Do1	IOIF2.Do1	IOIF1.Do1

For MNET-4XMO-C and HSL-4XMO, the definitions of DO bits are as follows. The default value is 0xf.

Bit3	Bit2	Bit1	Bit0
IOIF4.Do1	IOIF3.Do1	IOIF2.Do1	IOIF1.Do1

For MNET-1XMO the definitions of DO bits are as follows. The default value is 0x0.

Bit3	Bit2	Bit1	Bit0
N/A	SZST 0(Low) 1(High)	STL 0(Low) 1(High)	AlmReset 0(Low) 1(High)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret; //return error code.
I32 boardId = 0;
I32 busNum = 0;
I32 moduleNum = 0;
I32 DO_Value = 0;

//Start Field bus first.
//    ret = APS_start_field_bus( boardId, busNum, startingAxisId );

ret = APS_field_bus_d_get_output(boardId, busNum, moduleNum, &DO_Value );
```

See also:

APS_field_bus_d_set_output()

APS_field_bus_d_get_input	Get field bus digital input

Support Products: PCI-8392H, DPAC-3000, PCI(e)-7856, MNET-4XMO-(C), HSL-4XMO, HSL-DIO

Descriptions:

This function is used to get input data from field bus digital input on slave modules. The maximum data length of one module ID is 32-bit. If the module ID has fewer channels than 32, the higher bit must be remained zero.

Notice: For HSL_DI56DO32_FCN module, users should call APS_field_bus_d_get_input_ex() for 64 bits DIO operation.

Syntax:

C/C++:

I32 FNTYPE APS_field_bus_d_get_input(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 *DI_Value);

Visual Basic:

APS_field_bus_d_get_input(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, DI_Value As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For High Speed Link(HSL) type field bus, the range of module number is 1 to 63. Note: In HSL, the Module_No is the first id occupied by the module.

For MNET type field bus, the range of module number is 0 to 63.

I32 *DI_Value: Return digital input value.

For MNET-4XMO, the definitions of DI bits are as follows.

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOIF4.Di2	IOIF3.Di2	IOIF2.Di2	IOIF1.Di2	IOIF4.Di1	IOIF3.Di1	IOIF2.Di1	IOIF1.Di1

For MNET-4XMO-C and 4XMO, the definitions of DI bits are as follows.

Bit3	Bit2	Bit1	Bit0
IOIF4.Di1	IOIF3.Di1	IOIF2.Di1	IOIF1.Di1

For MNET-1XMO the definitions of DI bits are as follows.

Bit3	Bit2	Bit1	Bit0

N/A	N/A	N/A	STLOV

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret; //return error code.
I32 boardId = 0;
I32 busNum = 0;
I32 moduleNum = 0;
I32 DI_Value = 0;

//Start Field bus first.
//    ret = APS_start_field_bus( boardId, busNum, startingAxisId );
ret = APS_field_bus_d_get_input( boardId, busNum,, moduleNum, &DI_Value );
```

See also:

APS_field_bus_d_set_output();APS_field_bus_d_get_output()

APS_field_bus_d_set_output_ex	Set field bus digital output for 64 bit operation

Support Products: PCI(e)-7856

Descriptions:

This function is used to set field bus digital output on slave modules for 64 bit DIO operation. The maximum data length of one module ID is 64-bit. If the module ID has fewer channels than 64, the higher bit must be remained zero when outputting. The read back data of higher bit will be zero.

Notice: Only be available on HSL_DI56DO32_FCN module for 64bit DIO operation.

Syntax:

C/C++:

I32 FNTYPE APS_field_bus_d_set_output_ex(I32 Board_ID, I32 BUS_No, I32 MOD_No , DO_DATA_EX DO_Value);

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For High Speed Link(HSL) type field bus, the range of module number is 1 to 63. Note: In HSL, the MOD_No is the first id occupied by the module.

For MNET type field bus, the range of module number is 0 to 63.

DO_DATA_EX DO_Value: Digital output value. In bit format. Bit 0 corresponding to digital output channel 0 and the rest may be deduced by analogy. The definition of its structure is shown below:

```
typedef struct
{
    U32 Do_ValueL;    //bit[0~31]
    U32 Do_ValueH;    //bit[32~63]
} DO_DATA_EX, *PDO_DATA_EX;
```

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret; //return error code.
```

```
I32 boardId = 0;
I32 busNum = 0;
I32 moduleNum = 0;
DO_DATA_EX DO_Value = {0, 0};

//Start Field bus first.
//    ret = APS_start_field_bus(  585  boardId, busNum, startingAxisId );
DO_Value. Do_ValueL = 0x0F; // Turn on bit 0 ~ 3
DO_Value. Do_ValueH = 0x00;
ret = APS_field_bus_d_set_output_ex(585boardId, busNum,, moduleNum, DO_Value );
```

See also:

APS_field_bus_d_get_output_ex()

APS_field_bus_d_get_output_ex	Get field bus digital output for 64 bit operation

Support Products: PCI(e)-7856

Descriptions:

This function is used to get field bus digital output on slave modules for 64 bit DIO operation. The maximum data length of one module ID is 64-bit. If the module ID has fewer channels than 64, the higher bit must be remained zero when outputting. The read back data of higher bit will be zero.

Notice: Only be available on HSL_DI56DO32_FCN module for 64bit DIO operation.

Syntax:

C/C++:

I32 FNTYPE APS_field_bus_d_get_output_ex(I32 Board_ID, I32 BUS_No, I32 MOD_No, DO_DATA_EX *DO_Value);

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For High Speed Link(HSL) type field bus, the range of module number is 1 to 63. Note: In HSL, the Module_No is the first id occupied by the module.

For MNET type field bus, the range of module number is 0 to 63.

DO_DATA_EX *DO_Value: Return digital output value. Bit 0 corresponding to digital output channel 0 and the rest may be deduced by analogy. The definition of its structure is shown below:

```
typedef struct
{
    U32 Do_ValueL;    //bit[0~31]
    U32 Do_ValueH;    //bit[32~63]
} DO_DATA_EX, *PDO_DATA_EX;
```

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret; //return error code.
```

```
I32 boardId = 0;
I32 busNum = 0;
I32 moduleNum = 0;
DO_DATA_EX DO_Value = {0, 0};

//Start Field bus first.
//    ret = APS_start_field_bus(  587  boardId, busNum, startingAxisId );

ret = APS_field_bus_d_get_output_ex(587boardId, busNum, moduleNum, &DO_Value );
```

See also:

APS_field_bus_d_set_output_ex()

APS_field_bus_d_get_input_ex	Get field bus digital input for 64 bit DIO operation

Support Products: PCI(e)-7856

Descriptions:

This function is used to get input data from field bus digital input on slave modules for 64 bit DIO operation. The maximum data length of one module ID is 64-bit. If the module ID has fewer channels than 64, the higher bit must be remained zero.

Notice: Only be available on HSL_DI56DO32_FCN module for 64bit DIO operation.

Syntax:

C/C++:

I32 FNTYPE APS_field_bus_d_get_input_ex(I32 Board_ID, I32 BUS_No, I32 MOD_No, DI_DATA_EX *DI_Value);

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For High Speed Link(HSL) type field bus, the range of module number is 1 to 63. Note: In HSL, the Module_No is the first id occupied by the module.

For MNET type field bus, the range of module number is 0 to 63.

I32 *DI_Value: Return digital input value. Bit 0 corresponding to digital input channel 0 and the rest may be deduced by analogy. The definition of its structure is shown below:

```
typedef struct
{
    U32 Di_ValueL;    //bit[0~31]
    U32 Di_ValueH;    //bit[32~63]
} DI_DATA_EX, *PDI_DATA_EX;
```

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret; //return error code.
I32 boardId = 0;
I32 busNum = 0;
```

```
I32 moduleNum = 0;
DI_DATA_EX DI_Value = { 0, 0 };

//Start Field bus first.
ret = APS_start_field_bus(    boardId, busNum, startingAxisId );

//Get 64 bit DI data
ret = APS_field_bus_d_get_input_ex(589boardId, busNum, moduleNum, &DI_Value );
```

See also:

APS_field_bus_d_set_output_ex(); APS_field_bus_d_get_output_ex()

APS_set_field_bus_slave_param	Set parameter to field bus slave module

Support Products: PCI-8392H, DPAC-3000, PCI(e)-7856

Descriptions:

This function is used to set field bus slave parameter.

Some parameters are for slave module itself and some are for channels of slave. It is depend on input-parameter “I32 Ch_no”. When you set -1 to Ch_no, it means you set parameter to specified module (module layer parameter). Otherwise you set channel number to CH_no to set parameter to specified channel

The detail of field bus slave parameters, please refer to slave parameter table.

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_slave_param(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 Ch_No, I32 ParaNum, I32 ParaDat);

Visual Basic:

APS_set_field_bus_slave_param(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal Ch_No As Long, ByVal ParaNum As Long, ByVal ParaDat As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Slave Module number.

For HSL slave module, depend on slave ID : 1 ~ 63. Note: In HSL, the Module_No is the first id occupied by the module.

I32 Ch_No: Channel number. If set this parameter to -1 mean set slave parameter.

-1 : Set parameter to specified slave module number.

0 ~ : Set parameter to specified channel number (AIO channel , DIO channel etc.)

I32 ParaNum: Slave / Channel parameter number.

Refer to fieldbus slave parameter definition table.

I32 ParaDat: Slave / Channel parameter data.

Refer to fieldbus slave parameter definition table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_get_field_bus_slave_param()

APS_get_field_bus_slave_param	Get parameter from field bus slave module

Support Products: PCI-8392H, DPAC-3000, PCI(e)-7856

Descriptions:

This function is used to get field bus slave parameter.

Some parameters are for slave module itself and some are for channels of slave. It is depened on input-parameter “I32 Ch_no”. When you set **-1** to Ch_no, it means you set parameter to specified module. Otherwise you set channel number to CH_no to set parameter to specified channel.

The detail of field bus slave parameters, please refer to slave parameter table.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_slave_param(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 Ch_No, I32 ParaNum, I32 *ParaDat);

Visual Basic:

APS_get_field_bus_slave_param(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 Ch_No, I32 ParaNum, I32 *ParaDat);

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Slave Module number.

For HSL slave module, depend on slave ID : 1 ~ 63. Note: In HSL, the Module_No is the first id occupied by the module.

I32 Ch_No: Channel number. If set this parameter to -1 mean set slave parameter.

-1 : Set parameter to specified slave module number.

0 ~ : Set parameter to specified channel number (AIO channel , DIO channel etc.)

I32 ParaNum: Slave / Channel parameter number.

Refer to fieldbus slave parameter definition table.

I32 *ParaDat: Return Slave / Channel parameter data.

Refer to fieldbus slave parameter definition table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_set_field_bus_slave_param()

APS_set_field_bus_a_output	Set field bus analog output

Support Products: PCI-8392(H) , DPAC-3000, PCI(e)-7856, PCIe-833x

Descriptions:

This function is used to set analog type of field bus salve analog output value. The conversion from digital value to floating point value is according to hardware specifications and built-in in APS.

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_a_output(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 Ch_No, F64 AO_Value);

Visual Basic:

APS_set_field_bus_a_output(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal Ch_No As Long, ByVal AO_Value As Double) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Slave Module number.

For HSL slave module, depend on slave ID : 1 ~ 63. Note: In HSL, the Module_No is the first id occupied by the module.

I32 Ch_No: Channel number. Value range 0 ~ n (n = max. channel number – 1)

F64 AO_Value: Analog output. Unit of value is depended on slave type. [V] for voltage / [A] for current.

For PCIe-833x :

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number. (only support index 0)

I32 MOD_No: The index of slave device (start from 0)

I32 Ch_No: Channel number. (start from 0)

F64 AO_Value: Analog output. Unit of value is depended on slave type. [V] for voltage / [A] for current.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_get_field_bus_a_output( ); APS_get_field_bus_a_input()

APS_get_field_bus_a_output	Get field bus analog output

Support Products: PCI-8392(H) , DPAC-3000, PCI(e)-7856, PCIe-833x

Descriptions:

This function is used to get analog output of analog type field bus salve. The conversion from digital value to floating point value is according to hardware specifications and built-in in APS.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_a_output(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 Ch_No, F64 *AO_Value);

Visual Basic:

APS_get_field_bus_a_output(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal Ch_No As Long, AO_Value As Double) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Slave Module number.

For HSL slave module, depend on slave ID : 1 ~ 63. Note: In HSL, the Module_No is the first id occupied by the module.

I32 Ch_No: Channel number. Value range 0 ~ n (n = max. channel number – 1)

F64 *AO_Value: Return analog output. Unit of value is depended on slave type. [V] for voltage / [A] for current.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_set_field_bus_a_output(); APS_get_field_bus_a_input()

APS_get_field_bus_a_input	Get field bus analog input

Support Products: PCI-8392(H) , DPAC-3000, PCI(e)-7856 , PCIe-833x

Descriptions:

This function is used to get analog input of analog type field bus salve. The conversion from digital value to floating point value is according to hardware specifications and built-in in APS.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_a_input(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 Ch_No, F64 *AI_Value);

Visual Basic:

APS_get_field_bus_a_input(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal Ch_No As Long, AI_Value As Double) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Slave Module number.

For HSL slave module, depend on slave ID : 1 ~ 63. Note: In HSL, the Module_No is the first id occupied by the module.

I32 Ch_No: Channel number. Value range 0 ~ n (n = max. channel number – 1)

F64 *AI_Value: Return analog input. Unit of value is depended on slave type. [V] for voltage / [A] for current.

For PCIe-833x:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number. (only support index 0)

I32 MOD_No: The index of slave device (start from 0)

I32 Ch_No: Channel number.

F64 *AI_Value: Return analog input. Unit of value is depended on slave type. [V] for voltage / [A] for current.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_set_field_bus_a_output(); APS_get_field_bus_a_output( )

APS_get_slave_connect_quality	Get the connected quality of slave

Support Products: PCI-8392(H), DPAC-3000, PCI(e)-7856

Descriptions:

This function is used to get the connected quality of slave.

After starting to scan slave module, this function can be used to check if any error of communication occurred. This result only shows the status at the moment when executing, not showing the status in the history. User can set the checking degree by PRF_CHKERRCNT_LAYER parameter. The range of return value is according to the number of id occupied by the module.

It must be remained again that this function just shows the quality of connection at this moment.

Note: This function supports HSL bus.

Note: This function doesn't support MotionNet bus.

Syntax:

C/C++:

I32 FNTYPE APS_get_slave_connect_quality(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 *Sts_data);

Visual Basic:

APS_get_slave_connect_quality (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByRef Sts_data As Long);

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1. This function only supports HSL bus now.

I32 MOD_No: Slave Module number.

For HSL slave module, depend on slave ID : 1 ~ 63. Note: In HSL, the Module_No is the first id occupied by the module.

I32 *Sts_data : Return status value. The return value is bit form. Each bit decriebes the communication status for each id respectively. Zero is normal, one is abnormal.

For example:

HSL module may occupy id more than one. You can recognize the state of each id via the retun value. However, if the return value is bigger than zero, it means that the communication isn't stable in the module.

0x00(0): All id is normal.

0x01(1): The first id is abnormal.

0x05(5): The first and the third ids are abnormal.

0x0f(15) : All ids are abnormal

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example for HSL bus:

```
//If the module occupies 4 ids.
I32 ret; //return error code.
I32 boardId = 0;
I32 busNum = 0;
I32 moduleNum = 1;
I32 Sts_data = 0;
I32 bus_param = 5;
I32 startingAxisId = 0;
//Start Field bus first.
Ret = APS_start_field_bus(boardId, busNum, startingAxisId );
ret = APS_set_field_bus_param (boardId, busNum, PRF_CHKERRCNT_LAYER, bus_param );
ret = APS_get_slave_connect_quality(boardId, busNum, moduleNum, &Sts_data );
//if Sts_data is 5, it means that first and third ids are abnormal.
```

See also:

APS_get_slave_online_status()

APS_get_slave_online_status	Get the connected quality of slave/ Get the status of slave

Support Products: PCI-8392(H) , DPAC-3000, PCI(e)-7856, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, HSL-DIO , PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to get the status of online.

After starting to scan slave module, this function can be used to check if the slave module is online or offline.

It must be noted that this function just shows the status of communication at this moment.

Note: This function supports both HSL & MotionNet bus.

Note: For the HSL bus, the range of return value is according to the number of bit occupied by the module.

For PCIe-833x or ECAT-4XMO , ECAT-TRG4 , this function is used to get the status of slave. This function should be executed after starting field bus.

Syntax:

C/C++:

I32 FNTYPE APS_get_slave_online_status (I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 * Live);

Visual Basic:

APS_get_slave_online_status (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByRef Live);

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Slave Module number.

For HSL slave module, depend on slave ID : 1 ~ 63.

Note: In HSL, the Module_No is the first id occupied by the module.

For MNET slave module, depend on slave ID : 0 ~ 63

I32 * Live : Return status value. The return value is bit form. Each bit decriebes the status for each id respectively. 0 is offline, 1 is online

Example for HSL bus:

HSL module may occupy id more than one. You can recognize the state of each id via the retun value.

0x00(0): All ids are offline

0x01(1): The first id is online

0x05(5): The first and the third ids are online

0x0f(15) : All ids are online

Example for Mnet bus:

User could identify communication error for specific SlaveId by invoking this function at this moment. If a communication error occurs for a specific SlaveId on three consecutive communication cycles, it will issue a communication error.

0x00(0): This id is offline. That is, this id issues a communication error.

0x01(1): This id is online. That is, the communication of this id is good.

For PCIe-833x :

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) only support number 0.

I32 MOD_No: The number ID of slave.

I32 *Live : Return the status of slave by bits definitions.

As follows are bits definitions:

Bit 0:

0: Slave Absense 1: Slave Presence

Bit 1:

0: Not Bus Scan 1: Bus Scan

Bit 2:

0: Not Initial 1: Initial

Bit 3:

0: Not PreOP 1: PreOP

Bit 4:

0: Not SafeOP 1: SafeOP

Bit 5:

0: Not OP 1: OP

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example

Below example is for HSL bus:

```
//If the module occupies 4 ids.
I32 ret; //return error code.
I32 boardId = 0;
```

```
I32 busNum = 0; //HSL bus number
I32 moduleNum = 1;
I32 on_line = 0;
I32 bus_param = 5;
I32 startingAxisId = 0;
//Start Field bus first.
Ret = APS_start_field_bus(boardId, busNum, startingAxisId );
ret = APS_get_slave_online_status (boardId, busNum, moduleNum, & on_line );
//if on_line is 5, it means that first and third ids are online.
```

Example2:

Below example is for MotionNet bus:

```
I32 ret; //return error code.
I32 boardId = 0;
I32 busNum = 1; //MotionNet Bus number
I32 moduleNo = 10;
I32 on_line = 0;

//Start Field bus..
//Check if communication has error for specific ModuleId at this moment.
Ret = APS_get_slave_online_status (boardId, busNum, moduleNo, & on_line );
```

Example3:

Below example is for PCIe-833x

```
I32 ret;
I32 Board_ID = 0;
I32 BUS_No = 0;
I32 MOD_No = 0;
I32 Live = 0;

ret = APS_get_slave_online_status ( Board_ID, BUS_No, MOD_No, &Live );
if( ret == ERR_NoError )
{
    if ( Live & 0x1 )
        printf("This slave is present.\n");
    else
        printf("This slave is absent.\n");
}
```

See also:

APS_get_slave_connect_quality()

APS_get_field_bus_master_status	Get field bus master status

Support Products: PCIe-833x

Descriptions:

To Get field bus master status such INIT state、SAFEOP state and OP state in the EtherCAT definition.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_master_status(I32 Board_ID, I32 BUS_No, U32 *Status)

Visual Basic:

APS_get_field_bus_master_status (ByVal Board_ID As Long, ByVal BUS_No As Long, ByRef Status As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: only support number 0.

U32 *Status: Return status of the field bus master.

The status of the field bus master are as follows:

EC_STATE_NOT_RDY (0x0000)

EC_STATE_RDY (0x0001)

EC_STATE_BUS_SCAN (0x0002)

EC_STATE_INIT (0x0003)

EC_STATE_PREOP (0x0004)

EC_STATE_SAFEOP (0x0005)

EC_STATE_OP (0x0006)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_get_field_bus_last_scan_info	Get fieldbus info after system scanning.

Support Products: PCI-8392(H) , DPAC-3000, PCI(e)-7856, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, HSL-DIO , PCIe-833x

Descriptions:

This function is used to get the fieldbus info after system scanning. Please refer to the **fieldbus Info table**.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_last_scan_info (I32 Board_ID, I32 BUS_No, I32 * Info_Array, I32 Array_Size, I32 *Info_Count);

Visual Basic:

APS_get_field_bus_last_scan_info (ByVal Board_ID As Long, ByVal BUS_No As Long, ByRef Info_Array As Long, ByVal Array_Size As Long, ByRef Info_Count As Long);

Parameters:

For MNET:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 * Info_Array: return scanning info. Refer to **fieldbus Info table.**

I32 Array_Size: The array size which user want to get.

I32 * Info_Count: return the actual size.

For MNET fieldbus info table

Array Index	Return scanning fieldbus Info
0	Total numbers of Slaves after scanning.
1	Total numbers of axes after scanning.

For PCIe-833x:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) only support number 0.

I32 * Info_Array: return scanning info. Refer to **fieldbus Info table.**

I32 Array_Size: The array size which user want to get.

I32 * Info_Count: return the actual size.

For PCIe-833x fieldbus info table

Array Index	Return scanning fieldbus Info
0	Total numbers of Slaves after scanning.
1	Not support.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example1:

Below example is for MNET

```
I32 ret;
I32 Info_Array[2];
I32 Info_Count;
ret = APS_get_field_bus_last_scan_info ( 0, 1, & Info_Array, 2, & Info_Count );
if( ret != ERR_NoError )
{
    //Get fieldbus info
}
```

Example2:

Below example is for PCIe-833x

```
I32 ret;
I32 Info_Array[1];
I32 Info_Count;
I32 Slave_Count;
ret = APS_get_field_bus_last_scan_info ( 0, 1, & Info_Array, 1, & Info_Count );
if( ret == ERR_NoError )
{
    //To get how many slaves number in the fieldbus
    Slave_Count = Info_Array[0];
}
```

See also:

APS_get_field_bus_master_type	Get master type of the fieldbus

Support Products: PCI-8392(H) , DPAC-3000, PCI(e)-7856, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, HSL-DIO

Descriptions:

This function is used to get the master type of the fieldbus.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_master_type(I32 Board_ID, I32 BUS_No, I32 *BUS_Type);

Visual Basic:

APS_get_field_bus_master_type(ByVal Board_ID As Long, ByVal BUS_No As Long, ByRef BUS_Type As Long);

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 * BUS_Type: Return .

0 : Reserved

1 : HSL

2 : MNET

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 BUS_Type;
ret = APS_get_field_bus_master_type ( 0, 1, & BUS_Type );
if( ret != ERR_NoError )
{
    // get the master type of the fieldbus
}
```

See also:

APS_get_field_bus_slave_type	Get slave type on the fieldbus

Support Products: PCI-8392(H) , DPAC-3000, PCI(e)-7856, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, HSL-DIO

Descriptions:

This function is used to get the slave type on the fieldbus.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_slave_type(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 *MOD_Type);

Visual Basic:

APS_get_field_bus_slave_type(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long , ByRef MOD_Type As Long);

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Slave Module number.

For HSL slave module, depend on slave ID : 1 ~ 63. In HSL, the Module_No is the first id occupied by the module.

For MNET slave module, depend on slave ID : 0 ~ 63

I32 * MOD_Type: Return .

0 : Reserved

1 : HSL

2 : MNET

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 MOD_Type;
ret = APS_get_field_bus_slave_type ( 0, 1, 10, & MOD_Type );
if( ret != ERR_NoError )
{
    // get the slave type on the fieldbus
```

}

See also:

APS_get_field_bus_slave_name	Get slave name on the fieldbus

Support Products: PCI-8392(H) , DPAC-3000 , PCI(e)-7856, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO, HSL-DIO

Descriptions:

This function is used to get the slave name on the fieldbus.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_slave_name(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 *MOD_Name);

Visual Basic:

APS_get_field_bus_slave_name (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long , ByRef MOD_Type As Long);

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Slave Module number.

For HSL slave module, depend on slave ID : 1 ~ 63. In HSL, the Module_No is the first id occupied by the module.

For MNET slave module, depend on slave ID : 0 ~ 63

I32 * MOD_ Name: Return module name.

0x000: UNKNOWN

0x100: HSL_DI32

0x101: HSL_DO32

0x102: HSL_DI16DO16

0x103: HSL_AO4

0x104: HSL_AI16AO2VV

0x105: HSL_AI16AO2_AV

0x106: HSL_DI16UL

0x107: HSL_DI16RO8

0x108: HSL 4XMO

0x109: HSL_DI16_UCT

0x10A: HSL_DO16_UCT

0x10B: HSL_DI8DO8

0x10C: HSL_DI56DO32_FCN

0x200: MNET_1XMO

0x201: MENT-4XMO

0x202: MENT-4XMO-C

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 MOD_ Name;
ret = APS_get_field_bus_slave_type ( 0, 1, 10, & MOD_ Name );
if( ret != ERR_NoError )
{
    // get the slave name on the fieldbus
}
```

See also:

APS_get_field_bus_slave_first_axis no	Get first axis of the slave module

Support Products: PCI-8392(H) , DPAC-3000 , PCI(e)-7856, MNET-4XMO-(C), MNET-1XMO, HSL-4XMO

Descriptions:

This function is used to get first axis of the slave module. After starting to scan slave module, this function can be used to get what axisID is allocated to the slave module.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_slave_first_axisno (I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 *AxisNo, I32 *Totalaxes);

Visual Basic:

APS_get_field_bus_slave_first_axisno (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long , ByRef AxisNo As Long, ByRef TotalAxes As Long);

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Slave Module number.

For HSL slave module, depend on slave ID : 1 ~ 63. In HSL, the Module_No is the first id occupied by the module.

For MNET slave module, depend on slave ID : 0 ~ 63

I32 *AxisNo: return first axis of the slave module.

I32 *TotalAxes: return total axes of this module

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 AxisID;
I32 Totalaxes;
ret = APS_get_field_bus_slave_first_axisno ( 0, 1, 10, & AxisID,& Totalaxes );
if( ret != ERR_NoError )
{
```

```
        // get first axis of the slave module
}
```

See also:

APS_get_field_bus_device_info	Get device(slave) information on a specified field bus

Support Products: PCI-8392(H) , DPAC-3000 , PCI(e)-7856, MNET-4XMO-(C), HSL-4XMO

Descriptions:

This function is used to get specified device (Slave) information. The information includes firmware version, PCB version and so on. Refer to device information table.

Syntax:

C/C++

I32 FNTYPE APS_get_field_bus_device_info(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 Info_No, I32 *Info);

Visual Basic:

APS_get_field_bus_device_info (ByVal Board_ID As Long, ByVal BUS_No As Long , ByVal MOD_No As Long , ByVal Info_No As Long, Info As Long) As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Slave Module number.

For HSL slave module, depend on slave ID : 1 ~ 63. In HSL, the Module_No is the first id occupied by the module.

For MNET slave module, depend on slave ID : 0 ~ 63

I32 Info_No: Reference to device information table.

I32 *Info: Reference to device information table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Board_ID = 0;
I32 BUS_No = 1;
I32 MOD_No = 0;
I32 ret;
I32 Info;
ret = APS_get_field_bus_device_info (Board_ID, BUS_No, MOD_No , 0x20, &Info );
if( ret != ERR_NoError )
{
```

```
        //Show device information.
}
```

See also:

APS_get_field_bus_module_info	Get slave information

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to get the slave device information after system starting. You can use this function to get such as vendorID、product code、Total Axis number、IO number etc.,

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_module_info(I32 Board_ID, I32 BUS_No, I32 MOD_No, PEC_MODULE_INFO Module_info);

Visual Basic:

APS_get_field_bus_module_info (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByRef Module_info As EC_MODULE_INFO) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) only support number 0.

I32 MOD_No: slave number (start from number 0)

PEC_MODULE_INFO Module_info : structure of slave information.
For detail of the members parameters as below:

I32 VendorID: the vender ID number of slave.

I32 ProductCode: the product number of slave.

I32 RevisionNo: the revision number of slave.

I32 TotalAxisNum: the total axes of slave.

I32 Axis_ID[64]: the axis ID number of auto slave ID mode.

I32 Axis_ID_manual[64]: the axis ID number of manual slave ID mode.

I32 All_ModuleType[32]: the sub module ID by sequence.

I32 DI_ModuleNum: the number of digital input module in slave.

I32 DI_ModuleType[32]: the type of digital input module in slave.

I32 DO_ModuleNum: the number of digital output module in slave.

I32 DO_ModuleType[32]: the type of digital output module in slave.

I32 AI_ModuleNum: the number of analog input module in slave.

I32 AI_ModuleType[32]: the type of analog input module in slave.

I32 AO_ModuleNum: the number of analog output module in slave.

I32 AO_ModuleType[32]: the type of analog output module in slave.

Char Name[128]: Reserve.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 BUS_No = 0;
I32 MOD_No = 0;
EC_MODULE_INFO Module_info;
ret = APS_get_field_bus_module_info(Board_ID, BUS_No, MOD_No,&Module_info);
if( ret == ERR_NoError )
{
    printf("Vendor ID is: 0x%x.\n", Module_info.VendorID);
    printf("Total axis number is: %d.\n", Module_info.TotalAxisNum);
}
```

See also:

APS_reset_field_bus_alarm	Reset the alarm signal of slave.

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

When servo drives occured alarm, and alarm severity is not critical you can reset the alarm signal by this function.

Syntax:

C/C++:

I32 FNTYPE APS_reset_field_bus_alarm(I32 Axis_ID);

Visual Basic:

APS_reset_field_bus_alarm (ByVal Axis_ID As Long) As Long

Parameters:

I32 Axis_ID: Number of axis.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Axis_ID = 0;
ret = APS_reset_field_bus_alarm( Axis_ID);
if( ret == ERR_NoError )
{
    printf("Reset alarm successful.\n");
}
```

See also:

APS_get_field_bus_alarm	Get alarm code of slave

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

When servo drives occured alarm, you can get alarm code by calling this function which to get value in OD(Error code,0x603F).The alarm code definition depends on each vendor of servo drive, you may reference to vendor's servo drive manual.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_alarm(I32 Axis_ID, U32 *AlarmCode);

Visual Basic:

APS_get_field_bus_alarm (ByVal Axis_ID As Long, ByRef AlarmCode As UInteger) As Long

Parameters:

I32 Axis_ID: Number of axis.

U32 *AlarmCode: return alarm status from slave.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Axis_ID = 0;
U32 AlarmCode;
ret = APS_get_field_bus_alarm( Axis_ID, &AlarmCode );
if( ret == ERR_NoError )
{
    printf("Display alarm code= %d\n", AlarmCode);
}
```

See also:

APS_get_field_bus_pdo	Get value from PDO memory

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This is the lowest level function which you can directely get value from EtherCAT PDO memory and align to EtherCAT cycle time.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_pdo(I32 Board_ID, I32 BUS_No, U16 ByteOffset, U16 Size, U32 *Value);

Visual Basic:

APS_get_field_bus_pdo (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal ByteOffset As Long, ByVal Size As Long, ByRef Value As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) only support number 0.

U16 ByteOffset: The offset address of specific PDO data, unit is byte.

U16 Size: The size value of PDO data, unit is byte.

U32 *Value: Return the value of PDO data.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 BUS_No = 0;
U16 ByteOffset = 16;//the OD offset of PDO is 16 bytes
U16 Size = 4;//to get 4 bytes data back
U32 Value = 0;
ret=APS_get_field_bus_pdo(Board_ID, BUS_No, ByteOffset, Size, &Value )
if( ret == ERR_NoError )
{
    printf("Display PDO value= %d\n", Value);
}
```

See also:

APS_set_field_bus_pdo	Set value to PDO memory

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This is the lowest level function which you can directely set value to EtherCAT PDO memory and align to EtherCAT cycle time.

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_pdo(I32 Board_ID, I32 BUS_No, U16 ByteOffset, U16 Size, U32 Value);

Visual Basic:

APS_set_field_bus_pdo(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal ByteOffset As Long, ByVal Size As Long, ByVal Value As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) only support number 0.

U16 ByteOffset: The offset address of specific PDO data, unit is byte.

U16 Size: The size value of PDO data, unit is byte.

U32 Value: The value set to the PDO.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 BUS_No = 0;
U16 ByteOffset = 16;//the OD offset of PDO is 16 bytes
U16 Size = 4;//to set 4 bytes data
U32 Value = 65535;
ret=APS_set_field_bus_pdo(Board_ID, BUS_No, ByteOffset, Size, Value )
if( ret == ERR_NoError )
{
    printf("Set data to PDO value successful\n");
}
```

See also:

APS_get_field_bus_pdo_offset	Get PDO information

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This is the lowest level function which you can directely get information from all EtherCAT PDO, like numbers, datatype, size, index and name.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_pdo_offset(I32 Board_ID, I32 BUS_No, I32 MOD_No, PPDO_OFFSET *PPTx, U32 *NumOfTx, PPDO_OFFSET *PPRx, U32 *NumOfRx);

Visual Basic:

APS_get_field_bus_pdo_offset (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByRef PPTx As IntPtr, ByRef NumOfTx As UInteger, ByRef PPRx As IntPtr, ByRef NumOfRx As UInteger) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) only support number 0.

I32 MOD_No: slave number (start from number 0)

PPDO_OFFSET *PPTx: Return information of Tx PDO

U32 *NumOfTx: Number of slave PDO Tx

PPDO_OFFSET* PPRx: Return information of Rx PDO

U32 *NumOfRx:Number of slave PDO Rx

typedef struct

{

U16 DataType; : The type of PDO data.

U32 ByteSize; : The size of PDO data, unit is byte.

U32 ByteOffset; : The offset address of specific PDO data, unit is byte.

U32 Index; : The index of PDO object

U8 NameArr[128]; : The name of PDO object

} PDO_OFFSET, *PPDO_OFFSET;

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 BUS_No = 0;
I32 MOD_No= 0;
PPDO_OFFSET PPTx;
PPDO_OFFSET PPRx;
U32 Tx_cnt, Rx_cnt;
I32 i;

ret = APS_get_field_bus_pdo_offset(Board_ID, BUS_No, MOD_No, &PPTx, &Tx_cnt, &PPRx,
&Rx_cnt);

if(ret == ERR_NoError)
{
	//	load data from PPDO_OFFSET struct
	for(i=0;i++;i<Tx_cnt)
	{
		printf("DataType : %d\n",(PPTx+i)-> DataType)
		printf("ByteSize : %d\n",(PPTx+i)-> ByteSize)
		printf("ByteOffset : %d\n",(PPTx+i)-> ByteOffset)
		printf("Name : %s\n",(PPTx+i)-> NameArr)
	}

	for(i=0;i++;i<Rx_cnt)
	{
		printf("DataType : %d\n",(PPRx+i)-> DataType)
		printf("ByteSize : %d\n",(PPRx+i)-> ByteSize)
		printf("ByteOffset : %d\n",(PPRx+i)-> ByteOffset)
		printf("Name : %s\n",(PPRx+i)-> NameArr)
	}
}
```

See also:

APS_get_field_bus_sdo	Get SDO data from slave

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

Use this function to get OD data from specific slave by SDO method.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_sdo(I32 Board_ID, I32 BUS_No, I32 MOD_No, U16 ODIndex, U16 ODSubIndex,U8 *Data, U32 DataLen, U32 *OutDatalen, U32 Timeout, U32 Flags);

Visual Basic:

APS_get_field_bus_sdo (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal ODIndex As UShort, ByVal ODSubIndex As UShort, ByRef Data As Byte, ByVal DataLen As UInteger, ByRef OutDatalen As UInteger, ByVal Timeout As UInteger, ByVal Flags As UInteger) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) only support number 0.

I32 MOD_No: slave number (start from number 0).

U16 ODIndex: The index of object dictionary.

U16 ODSubIndex: The sub index of object dictionary.

U8 *Data: Return the data value of specific OD.

U32 DataLen: The data length of specific OD, unit is byte.

U32 *OutDatalen: Return the actual data length of specific OD, unit is byte.

U32 Timeout: The maximum waiting time to get data from slave, unit is ms.

U32 Flags: reserve to 0.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 BUS_No = 0;
I32 MOD_No= 0;
U16 ODIndex = 0x60fd;
U16    ODSubIndex = 0;
U8 Data = 0;
U32 DataLen = 4;
U32 OutDatalen = 0;
U32 Timeout = 5000;
U32 Flags = 0;
ret= APS_get_field_bus_sdo( Board_ID,
                            BUS_No,
                            MOD_No,
                            ODIndex,
                            ODSubIndex,
                            &Data,
                            DataLen,
                            &OutDatalen,
                            Timeout,
                            Flags
                            );

if( ret == ERR_NoError )
{
        printf("The OD data value =%d\n",Data);
}
```

See also:

APS_set_field_bus_sdo	Set SDO data to slave

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

Use this function to set OD data to specific slave by SDO method.

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_sdo(I32 Board_ID, I32 BUS_No, I32 MOD_No, U16 ODIndex, U16 ODSubIndex, U8 *Data, U32 DataLen, U32 Timeout, U32 Flags);

Visual Basic:

APS_set_field_bus_sdo (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal ODIndex As UShort, ByVal ODSubIndex As UShort, ByRef Data As Byte, ByVal DataLen As UInteger, ByVal Timeout As UInteger, ByVal Flags As UInteger) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) only support number 0.

I32 MOD_No: slave number (start from number 0).

U16 ODIndex: The index of object dictionary.

U16 ODSubIndex: The sub index of object dictionary.

U8 *Data: The data value of specific OD.

U32 DataLen: The data length of specific OD, unit is byte.

U32 Timeout: The maximum waiting time to get data from slave, unit is ms.

U32 Flags: reserve to 0.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 BUS_No = 0;
I32 MOD_No= 0;
U16 ODIndex = 0x60fe;
U16   ODSubIndex = 1;
U8 Data = 256;
```

```
U32 DataLen = 4;
U32 Timeout = 5000;
U32 Flags = 0;
ret= APS_set_field_bus_sdo( Board_ID,
                            BUS_No,
                            MOD_No,
                            ODIndex,
                            ODSubIndex,
                            &Data,
                            DataLen,
                            Timeout,
                            Flags
                            );

if( ret == ERR_NoError )
{
    printf("Set OD data to slave successful.\n");
}
```

See also:

APS_set_field_bus_od_data	Set EtherCAT OD raw data

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to set EtherCAT OD data in PDO by operates specific slave device.

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_od_data(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 SubMOD_No, I32 ODIndex, U32 RawData);

Visual Basic:

APS_set_field_bus_od_data (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal SubMOD_No As Long, ByVal ODIndex As Long, ByVal RawData As UInteger) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) only support number 0.

I32 MOD_No: The number ID of slave.

I32 SubMOD_No:The sub module in one slave.

I32 ODIndex: The EtherCAT OD data index.

U32 RawData: The EtherCAT OD data

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 BUS_No = 0;
I32 MOD_No = 0;
I32 SubMOD_No = 0;
I32 ODIndex = 0;
U32 ODValue = 2048;

ret = APS_set_field_bus_od_data(Board_ID, BUS_No, MOD_No, SubMOD_No,ODIndex,
  ODValue);
```

See also:

APS_get_field_bus_od_data	Get EtherCAT OD raw data

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to get EtherCAT OD data in PDO by operates specific slave device.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_od_data(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 SubMOD_No, I32 ODIndex, U32 *RawData);

Visual Basic:

APS_get_field_bus_od_data (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal SubMOD_No As Long, ByVal ODIndex As Long, ByRef RawData As UInteger) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) only support number 0.

I32 MOD_No: The number ID of slave.

I32 SubMOD_No: The sub module in one slave.

I32 ODIndex: The EtherCAT OD data index.

U32 *RawData: Return the EtherCAT OD data

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 BUS_No = 0;
I32 MOD_No = 0;
I32 SubMOD_No = 0;
I32 ODIndex = 0;
U32 RawData ;

ret = APS_get_field_bus_od_data( Board_ID, BUS_No, MOD_No, SubMOD_No, ODIndex, &RawData );
```

```
if( ret == ERR_NoError )
{
        printf(“OD value is = %d\n”, RawData);
}
```

See also:

APS_get_field_bus_od_module_info	Get EtherCAT slave information

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to get EtherCAT slave information such as vendorID、product code and module ID.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_od_module_info(I32 Board_ID, I32 BUS_No, I32 MOD_No, PEC_Sub_MODULE_INFO Sub_Module_info);

Visual Basic:

APS_get_field_bus_od_module_info (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByRef Sub_Module_info As EC_Sub_MODULE_INFO) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) only support number 0.

I32 MOD_No: The number ID of slave.

The define of struct EC_Sub_MODULE_INFO as follows:

I32 VendorID: The vendor ID number of slave

I32 ProductCode:The ProductCode number of slave

I32 RevisionNo: The RevisionNo number of slave

I32 TotalSubModuleNum: The maximum sub module number of slave

I32 SubModuleID[32]:The ID number array of sub module

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 BUS_No = 0;
I32 MOD_No = 0;
EC_Sub_MODULE_INFO Sub_Module_info;
I32 i = 0;
```

```
ret = APS_get_field_bus_od_module_info( Board_ID, BUS_No, MOD_No,
&Sub_Module_info );
if( ret == ERR_NoError )
{
        for ( i = 0 ; i < Sub_Module_info.TotalSubModuleNum ; i++)
        {
                if ( Sub_Module_info.SubModuleID[i] != 0 )
                        printf("SubModuleID is = 0x%x\n", Sub_Module_info.SubModuleID[i]);
        }
}
```

See also:

APS_get_field_bus_module_map	Get mapped slave ID in manual ID mode

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to get total mapped Slave ID when using manual ID mode.

Figure 1 is an example to illustrate slave ID representation in auto mode and manual ID mode. First, user can use *APS_get_field_bus_last_scan_info()* to get how many slaves exists in EtherCAT network now. Here it is assumed 40 slaves are used. Second, user will get an actual mapped slave ID array by this function. In this array, the array index denotes the slave ID in auto mode and the array value denotes the slave ID in manual ID mode. For example, if user's array MOD_No_Arr get MOD_No_Arr[0] = 100, MOD_No_Arr[1] = 200, ... and MOD_No_Arr[39] = 4000 by this function, it shows the array index 0, 1, ... 39 are slave ID in auto mode, and the array value 100, 200, ... 4000 are slave ID in manual ID mode.

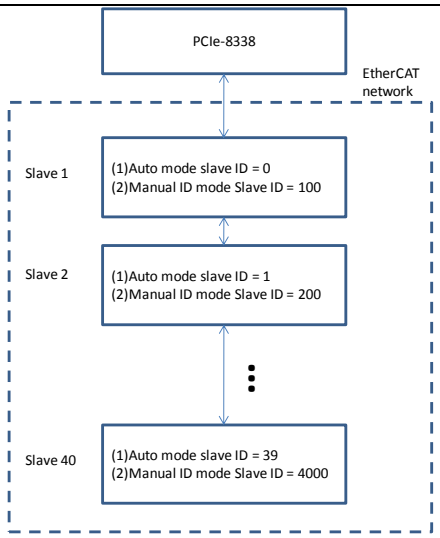


Figure 1 Representation of slave IDs in auto mode and manual ID mode

Syntax:

C/C++:

I32 APS_get_field_bus_module_map(I32 Board_ID, I32 BUS_No, U32 *MOD_No_Arr, U32 Size);

Visual Basic:

APS_get_field_bus_module_map(ByVal Board_ID As Long, ByVal BUS_No As Long, MOD_No_Arr As Int, ByVal Size As Int);

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) only support number 0.

U32* MOD_No_Arr: Mapped slave ID array in manual ID mode

U32 Size: Total slaves exist in EtherCAT network

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 BUS_No = 0;
U32 * MOD_No_Arr = NULL;
U32 Size = 0;
//if total slave number are 5, Size = 5;
MOD_No_Arr = (U32 *) malloc(    sizeof(U32) * Size    );
ret = APS_get_field_bus_module_map ( Board_ID, BUS_No, MOD_No_Arr, Size );
```

See also:

APS_get_field_bus_last_scan_info()

APS_set_field_bus_module_map	Set mapped slave ID in manual ID mode

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to set mapped slave ID in manual slave ID mode.

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_module_map (I32 Board_ID, I32 BUS_No, U32*

MOD_No_Arr, U32 Size);

Visual Basic:

APS_set_field_bus_module_map (ByVal Board_ID As Long, ByVal Bus_No As Long, ByVal

MOD_No_Arr() As UInteger, ByVal Size As UInteger) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) only support number 0.

U32 *MOD_No_Arr: Mapped slave ID array in manual slave ID mode.

U32 Size: Total slave number exists in field bus network.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 BUS_No = 0;
U32 * MOD_No_Arr = NULL;
U32 Size = 0;
//if total slave number are 5, Size = 5;
MOD_No_Arr = (U32 *) malloc(    sizeof(U32) * Size    );
MOD_No_Arr[0] = 111; // first slave of topology manual ID is 111
MOD_No_Arr[1] = 222; // second slave of topology manual ID is 222
MOD_No_Arr[2] = 333; // third slave of topology manual ID is 333
MOD_No_Arr[3] = 444; // fourth slave of topology manual ID is 444
MOD_No_Arr[4] = 555; // fifth slave of topology manual ID is 555
ret = APS_set_field_bus_module_map ( Board_ID, BUS_No, MOD_No_Arr, Size );
```

See also:

APS_get_field_bus_slave_state	Get the status of slave's state machine

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to get the status of slave's state machine.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_slave_state (I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 *State);

Visual Basic:

APS_get_field_bus_slave_state (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByRef State As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) only support number 0.

I32 MOD_No: The number ID of slave.

I32 *State: the status of slave's state machine.

value	State define
3	EC_STATE_INIT
4	EC_STATE_PREOP
5	EC_STATE_SAFEOP
6	EC_STATE_OP

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 BUS_No = 0;
I32 MOD_No = 0;
I32 State = 0;

ret = APS_get_field_bus_slave_state (Board_ID,BUS_No,MOD_No, &State);
```

// Please refer APS_set_field_bus_slave_state of state table.

See also:

APS_set_field_bus_slave_state()

APS_set_field_bus_slave_state	Set the status of slave's state machine

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to set the status of slave's state machine.

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_slave_state(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 State);

Visual Basic:

APS_set_field_bus_slave_state (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal State As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) only support number 0.

I32 MOD_No: The number ID of slave.

I32 State: The status of slave's state machine.

value	State define
3	EC_STATE_INIT
4	EC_STATE_PREOP
5	EC_STATE_SAFEOP
6	EC_STATE_OP

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 BUS_No = 0;
I32 MOD_No = 0;
I32 State = 0 ;
State = 5 ; //Safe Op mode
ret = APS_set_field_bus_slave_state ( Board_ID, BUS_No, MOD_No, State);
```

See also:

APS_get_field_bus_ESC_register	Get EtherCAT Slave Controller register

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to get EtherCAT slave controller(ESC) register.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_ESC_register(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 RegOffset, I32 DataSize, I32 *DataValue);

Visual Basic:

APS_get_field_bus_ESC_register (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal RegOffset As Long, ByVal DataSize As Long, ByRef DataValue As UInteger) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number (Port number) that only supports number 0.

I32 MOD_No: The number ID of slave.

I32 RegOffset: The address offset of ESC register.

I32 DataSize: The length of ESC register, unit is byte. Its range should be between 1 and 8 bytes.

I32 *DataValue: Get ESC data buffer. If range is over 4 bytes, need two dimension I32 array to operate.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 BUS_No = 0;
I32 MOD_No = 0;
I32 RegOffset = 0x920;
I32 DataSize = 8;
I32 GetDataValue[2];
```

ret = APS_get_field_bus_ESC_register (Board_ID, BUS_No, MOD_No, RegOffset, DataSize, &GetDataValue);

See also:

APS_set_field_bus_ESC_register()

APS_set_field_bus_ESC_register	Set EtherCAT Slave Controller register

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to set EtherCAT slave controller(ESC) register.

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_ESC_Register(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 RegOffset, I32 DataSize, I32 *DataValue);

Visual Basic:

APS_set_field_bus_ESC_register (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal RegOffset As Long, ByVal DataSize As Long, ByRef DataValue As UInteger) As Long

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number (Port number) that only supports number 0.

I32 MOD_No: The number ID of slave.

I32 RegOffset: The address offset of ESC register.

I32 DataSize: The length of ESC register, unit is byte. It's range should be between 1 and 8 bytes.

I32 *DataValue: Get ESC data buffer. If range is over 4 bytes, need two dimension I32 array to operate.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Board_ID = 0;
I32 BUS_No = 0;
I32 MOD_No = 0;
I32 RegOffset = 0x300;
I32 DataSize = 1;
I32 DataValue = 0;
```

ret = APS_set_field_bus_ESC_register (Board_ID, BUS_No, MOD_No, RegOffset, DataSize, &DataValue);

See also:

APS_get_field_bus_ESC_register ()

APS_get_system_loading	Get system loop loading

Support Products: PCIe-833x,

Descriptions:

This function is used to get system loop loading.

Syntax:

C/C++:

I32 FNTYPE APS_get_system_loading(I32 Board_ID, F64* Loading1, F64* Loading2, F64* Loading3, F64* Loading4);

Visual Basic:

APS_get_system_loading (ByVal Board_ID As Long, ByRef Loading1 As Double, ByRef Loading2 As Double, ByRef Loading3 As Double, ByRef Loading4 As Double) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

F64* Loading1: Calculate PCIe-8334/8 motion loop cosume time loading, unit is %.

F64* Loading2: Calculate PCIe-8334/8 EtherCAT loop cosume time loading, unit is %.

F64* Loading3: Reserve.

F64* Loading4: Reserve.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
F64 motion_loading;
F64 ECAT_loading;
F64 no_data1, no_data2;

ret = APS_get_system_loading (Board_ID, &motion_loading, &ECAT_loading, &no_data1, &no_data2);
```

See also:

APS_get_field_bus_analysis_topology	Get current and past topology then analysis.

Support Products: PCIe-833x, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to analysis current and past slave topology when APS_start_filed_bus API retrun -4013 or –4043 error. The analysis condition includes vendor ID, product code, revision number and sub-module ID with slave. If topology chanes between scan file bus and start field bus process, API will return error slave number for user reference.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_analysis_topology(I32 Board_ID, I32 BUS_No, I32 *Error_Slave_No,PEC_MODULE_INFO Current_slave_info,I32 *Current_slave_num,PEC_MODULE_INFO Past_slave_info,I32 * Past_slave_num);

Visual Basic:

APS_get_field_bus_analysis_topology (ByVal Board_ID As Long, ByVal Bus_No As Long, ByRef Error_Slave_No As Long, ByRef Current_slave_info As EC_Sub_MODULE_INFO, ByRef Current_slave_num As Long, ByRef Past_slave_info As EC_Sub_MODULE_INFO, ByRef Past_slave_num As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number (Port number) that only supports number 0.

I32* Error_Slave_No: Error slave number with auto slave ID mode.If Error_Slave_No = 0, there is no error, past slaves and current slaves are the same. If return 1, the current slave 1 is different from past slave 1. If return 5, the current slave 5 is different from past slave 5.

PEC_MODULE_INFO Current_slave_info :

Structure of slave information with current topology. For detail of the members parameters as below:

I32 VendorID:	The vender ID number of slave.
I32 ProductCode:	The product code of slave.
I32 RevisionNo:	The revision number of slave.
I32 TotalAxisNum:	Reserved.
I32 Axis_ID[64]:	Reserved.
I32 Axis_ID_manual[64]:	Reserved.
I32 All_ModuleType[32]:	The sub module ID by sequence.
I32 DI_ModuleNum:	Reserved.

I32 DI_ModuleType[32]: Reserved.
I32 DO_ModuleNum: Reserved.
I32 DO_ModuleType[32]: Reserved.
I32 AI_ModuleNum: Reserved.
I32 AI_ModuleType[32]: Reserved.
I32 AO_ModuleNum: Reserved.
I32 AO_ModuleType[32]: Reserved.
Char Name[128]: Reserved.

I32 *Current_slave_num : Slave numbers of current topology.
PEC_MODULE_INFO Past_slave_info : Structure of slave information with past topology.
I32 * Past_slave_num : Slave numbers of past topology.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 BoardID = 0;
I32 BusNo = 0 ;
I32 Error_Slave_No = 0;
I32 Current_slave_num =0 ;
I32 Past_slave_num =0;
EC_MODULE_INFO Current_slave_info[64] = {0};
EC_MODULE_INFO Past_slave_info[64] = {0};

ret = APS_get_field_bus_analysis_topology( BoardID,BusNo,
&Error_Slave_No,Current_slave_info,&Current_slave_num,Past_slave_info,&Past_slave_num
);
```

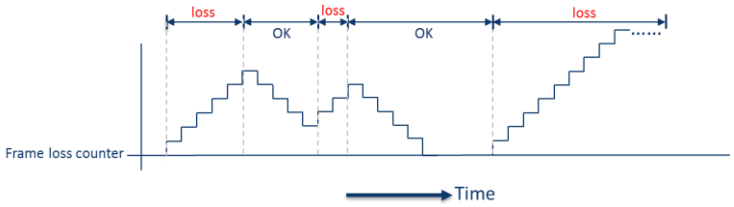
See also:

APS_get_field_bus_loss_package	Get the loss of EtherCAT frame count on receive bus direction

Support Products: PCIe-833x

Descriptions:

This function is used to get the loss of EtherCAT frame count on receive bus direction. The count behavior is shows below:



If system detect frame losing that the frame loss count will increase one by every EtherCAT loop timing. The frame loss count will decrese until zero when system getting frame normally.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_loss_package (I32 Board_ID, I32 BUS_No,I32 *Loss_Count);

Visual Basic:

APS_get_field_bus_loss_package (ByVal Board_ID As Integer, ByVal BUS_No As Integer, ByRef Loss_Count As Integer) As Integer

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number (Port number) that only supports number 0.

I32 *Loss_Count: The count value of package loss.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 BoardID = 0;
I32 BusNo = 0;
```

```
I32 lossCount = 0;

ret = APS_get_field_bus_loss_package( Board_ID, BusNo, & lossCount );
// Stop X, Y and Z moving when loss package count bigger than 10.
if( lossCount >= 10 )
{
	ret = APS_emg_stop( X );
	ret = APS_emg_stop( Y );
	ret = APS_emg_stop( Z );
}
```

See also:

17. Gear / Gantry functions

APS_set_gantry_param	Set gantry function related parameter / Enable/Disable a specified gear mode

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to set parameters to a specified gantry group.

The parameter number and the corresponding parameter data, please refer to the Gantry parameters table.

Syntax:

C/C++:

I32 FNTYPE APS_set_gantry_param(I32 Board_ID, I32 GroupNum, I32 ParaNum, I32 ParaDat);

Visual Basic:

APS_set_gantry_param(ByVal Board_ID As Long, ByVal GroupNum As Long, ByVal ParaNum As Long, I32 ParaDat As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 GroupNum: Specified a gantry group number.

I32 ParaNum: Parameter number. Please refer to Gantry parameters table.

I32 ParaDat: Parameter data. Please refer to Gantry parameters table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 ret; //return error code.

I32 boardId = 0;

See also:

APS_get_gantry_param();APS_set_gantry_axis();APS_get_gantry_axis()

APS_get_gantry_param	Get gantry function related parameter

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to get parameters from a specified gantry group.

The parameter number and the corresponding parameter data, please refer to the Gantry parameters table.

Syntax:

C/C++:

I32 FNTYPE APS_get_gantry_param(I32 Board_ID, I32 GroupNum, I32 ParaNum, I32 *ParaDat);

Visual Basic:

APS_get_gantry_param(ByVal Board_ID As Long, ByVal GroupNum As Long, ByVal ParaNum As Long, ParaDat As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 GroupNum: Specified a gantry group number.

I32 ParaNum: Specified a parameter number. Please refer to Gantry parameters table.

I32 *ParaDat: Return a parameter data. Please refer to Gantry parameters table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 ret; //return error code.

I32 boardId = 0;

See also:

APS_set_gantry_param(); APS_set_gantry_axis();APS_get_gantry_axis()

APS_set_gantry_axis	Set two axes in a gantry group

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to specify any two axes into a gantry group. Once the gantry mode of this group is enabled, those two axes will have gantry behavior. You can't change gantry axis setting when gantry mode is enabled.

Syntax:

C/C++:

I32 FNTYPE APS_set_gantry_axis(I32 Board_ID, I32 GroupNum, I32 Master_Axis_ID, I32 Slave_Axis_ID);

Visual Basic:

APS_set_gantry_axis(ByValBoard_ID As Long, ByVal GroupNum As Long, ByVal Master_Axis_ID As Long, ByVal Slave_Axis_ID As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 GroupNum: Specified a gantry group number. The maximum group number refers to specification.

I32 Master_Axis_ID: Specified an axis ID as a gantry master axis.

I32 Slave_Axis_ID: Specified an axis ID as a gantry slave axis.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret; //return error code.
I32 boardId = 0;
I32 GroupNum = 0;
I32 Master_Axis_ID = 0, Slave_Axis_ID = 1;

//Gantry mode muse be disable before you set the gantry axes.
Ret = APS_set_gantry_axis(Board_ID, GroupNum, Master_Axis_ID, Slave_Axis_ID );
if( ret != ERR_NoError )
        //…check error code.
```

```
Ret = APS_get_gantry_axis(Board_ID, GroupNum, &Master_Axis_ID, &Slave_Axis_ID );
if( ret != ERR_NoError )
        //…check error code.
```

See also:

APS_get_gantry_axis(); APS_set_gantry_param(); APS_get_gantry_param()

APS_get_gantry_axis	Get which axes in a gantry group

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to get gantry master axis ID and slave axis ID in a specify gantry group.

Syntax:

C/C++:

I32 FNTYPE APS_get_gantry_axis(I32 Board_ID, I32 GroupNum, I32 *Master_Axis_ID, I32 *Slave_Axis_ID);

Visual Basic:

APS_get_gantry_axis(ByVal Board_ID As Long, ByVal GroupNum As Long, Master_Axis_ID As Long, Slave_Axis_ID As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 GroupNum: Specified a gantry group number.

I32 *Master_Axis_ID: Return the master axis ID in a specify gantry group.

I32 *Slave_Axis_ID: Return the slave axis ID in a specify gantry group.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret; //return error code.
I32 boardId = 0;
I32 GroupNum = 0;
I32 Master_Axis_ID = 0, Slave_Axis_ID = 1;

//Gantry mode muse be disable before you set the gantry axes.
Ret = APS_set_gantry_axis(Board_ID, GroupNum, Master_Axis_ID, Slave_Axis_ID );
if( ret != ERR_NoError )
    //…check error code.

Ret = APS_get_gantry_axis(Board_ID, GroupNum, &Master_Axis_ID, &Slave_Axis_ID );
if( ret != ERR_NoError )
    //…check error code.
```

See also:

APS_set_gantry_axis(); APS_set_gantry_param(); APS_get_gantry_param()

APS_get_gantry_error	Get gantry axes deviation error

Support Products: PCI-8253/56, PCI-8392(H)

Descriptions:

This function is used to get gantry axes deviation error.

Deviation error = Master axis feedback position – Slave axis feedback position

Syntax:

C/C++:

I32 FNTYPE APS_get_gantry_error(I32 Board_ID, I32 GroupNum, I32 *GentryError);

Visual Basic:

APS_get_gantry_error (ByVal Board_ID As Long, ByVal GroupNum As Long, GentryError As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 GroupNum: Specified a gantry group number.

I32 *GentryError: Return gantry axes deviation error.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret; //return error code.
I32 boardId = 0;
I32 GroupNum = 0;
I32 GentryError;

ret = APS_get_gantry_error(boardId, GroupNum, &GentryError );
if( ret == ERR_NoError)
    // Display GantryError
```

See also:

APS_set_gantry_axis(); APS_set_gantry_param(); APS_get_gantry_param()

APS_get_encoder	Get encoder

Support Products: PCI-8253/56

Descriptions:

This function is used to get encoder counter of one axis. The counter is in unit of pulse.

Generally speaking, it is used for compensation of gantry home return.

Syntax:

C/C++:

I32 FNTYPE APS_get_encoder(I32 Axis_ID, I32 *Encoder);

Visual Basic:

APS_get_encoder(ByVal Axis_ID As Long, Encoder As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Encoder: Encoder counter. Unit in pulse.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Encoder;
APS_get_encoder(Axis_ID, &Encoder ); //Get encoder counter.
…//
```

See also:

APS_ get_latch_event(); APS_get_latch_counter()

APS_get_latch_event	Get latch event by axis

Support Products: PCI-8253/56

Descriptions:

This function is used to get latch event. There are two sources including Ez and Org signal latch. If a latch is occurring, the event turns on. User could clear the latch event by invoking APS_get_latch_counter().

Generally speaking, it is used for compensation of gantry home return.

Syntax:

C/C++:

I32 FNTYPE APS_get_latch_event(I32 Axis_ID, I32 Src, I32 *Event);

Visual Basic:

APS_get_latch_event(ByVal Axis_ID As Long, ByVal Src As Long, Event As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Src: Specify a latch source.

0: Ez latch, 1: Org latch.

I32 *Event: latch event.

0: No any latch occurred. 1: A latch occurred.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Event, latchCounter;
I32 SrcOrg = 1; //Specify Org

APS_get_latch_event(Axis_ID, SrcOrg, &Event ); //Get ORG latch event
If( Event == 1 ) //ORG is latched
{       //Reset latch event & Read latch counter
        APS_get_latch_counter(Axis_ID, SrcOrg, &latchCounter);
}
```

See also:

APS_get_latch_counter(); APS_get_encoder()

APS_get_latch_counter	Get latch counter by axis

Support Products: PCI-8253/56

Descriptions:

This function is used to get latch counter. There are two sources including Ez and Org signal latch. If a latch is occurring, the event turns on and the encoder counter is latched. User could get latch counter and reset (turn off) the event by invoking this function.
Generally speaking, it is used for compensation of gantry home return.

Syntax:

C/C++:

I32 FNTYPE APS_get_latch_counter(I32 Axis_ID, I32 Src, I32 *Counter);

Visual Basic:

APS_get_latch_counter(ByVal Axis_ID As Long, ByVal Src As Long, Counter As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Src: Specify a latch source.

0: Ez latch, 1: Org latch.

I32 *Counter: Latch counter.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Event, latchCounter;
I32 SrcOrg = 1; //Specify Org

APS_get_latch_event(Axis_ID, SrcOrg, &Event ); //Get ORG latch event
If( Event == 1 ) //ORG is latched
{
    //Reset latch event & Read latch counter
    APS_get_latch_counter(Axis_ID, SrcOrg, &latchCounter);
}
```

See also:

APS_set_latch_event(); APS_get_encoder()

APS_start_gear	Enable/Disable a specified gear mode

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to enable a specified gear mode. Two gear modes, including stardard and gantry, are available for specified application.

Syntax:

C/C++:

I32 FNTYPE APS_start_gear (I32 Axis_ID, I32 Mode);

Visual Basic:

APS_start_gear (ByVal Axis_ID As Long, ByVal Mode As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Mode: Gear mode.

0: Disable, 1: Standard mode, 2: Gantry mode.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

…//

APS_start_gear(Axis_ID, 0); //Disable gear.

…//

APS_start_gear(Axis_ID, 1); //Enable a standard gear mode.

See also:

APS_get_gear_status()

APS_get_gear_status	Get gear status

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get status of gear applicaiton.

Syntax:

C/C++:

I32 FNTYPE APS_get_gear_status(I32 Axis_ID, I32 *Status);

Visual Basic:

APS_get_gear_status(ByVal Axis_ID As Long, Status As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Status: Gear status.

0: In disabling status.

1: In enabling status of standard mode.

2: In enabling status of gantry mode.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Status;

APS_get_gear_status(Axis_ID, &Status ); //Get Gear status
…//
```

See also:

APS_start_gear()

APS_get_gantry_number	Get number of this master's corresponding slaves

Support Products: PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get the total number of this master's corresponding slaves in gantry mode.User need to use axis parameters PRA_EGEAR_MASTER (0x65) and PRA_EGEAR_SOURCE (0x66) and APS_set_axis_param() function to specify master and corresponding slaves and use APS_start_gear() to enable gantry mode.

Then user can use these functions APS_get_gantry_number() to get total number of slaves and APS_get_gantry_info() to get slave axis ID array.

Syntax:

C/C++:

I32 APS_get_gantry_number(I32 MasterAxisID, I32 *SlaveAxisIDSize);

Visual Basic:

APS_get_gantry_number(ByVal MasterAxisID As Long, SlaveAxisIDSize As Long) As Long

Parameters:

I32 MasterAxisID: Master axis ID; The Axis ID is from 0 to 65535.

I32* SlaveAxisIDSize: Total number of this master's corresponding slaves

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_get_gantry_info()

APS_get_gantry_info	Get slave axis ID array

Support Products: PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get slave axis ID array in gantry mode. Please refer the description of APS_get_gantry_number() for details.

Syntax:

C/C++:

I32 APS_get_gantry_info (I32 MasterAxisID, I32 SlaveAxisIDSize, I32 *SlaveAxisIDArray);

Visual Basic:

APS_get_gantry_info (ByVal MasterAxisID As Long, ByVal SlaveAxisIDSize As Long, SlaveAxisIDArray As Long) As Long

Parameters:

I32 MasterAxisID: Master axis ID; The Axis ID is from 0 to 65535.

I32 SlaveAxisIDSize: Total number of slaves

I32* SlaveAxisIDArray: Slave axis ID array

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_get_gantry_number()

APS_get_gantry_deviation	Get position deviation between master and slaves

Support Products: PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get position deviation between master and slaves. This function is implemented in ASYNC mode.

Syntax:

C/C++:

I32 APS_get_gantry_deviation (I32 MasterAxisID, I32 SlaveAxisIDSize, I32 *SlaveAxisIDArray, F64 *DeviationArray);

Visual Basic:

APS_get_gantry_deviation (ByVal MasterAxisID As Long, ByVal SlaveAxisIDSize As Long, SlaveAxisIDArray As Long, DeviationArray As Long) As Long

Parameters:

I32 MasterAxisID: Master axis ID; The Axis ID is from 0 to 65535.

I32 SlaveAxisIDSize: Total number of slaves

I32* SlaveAxisIDArray: Slave axis ID array

F64* DeviationArray: Position deviation array between master and slave

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

18. Compare trigger

APS_set_trigger_param	Set compare trigger related parameter

Support Products: PCI-8253/56,PCI-C154(+), PCI-8154/8158(DB-8150), EMX-100, PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to set comparing trigger related parameters. All definitions of trigger parameters are described in trigger parameter table.

You can also get parameter setting using “APS_get_trigger_param()” function.

Syntax:

C/C++:

I32 FNTYPE APS_set_trigger_param(I32 Board_ID, I32 Param_No, I32 Param_Val);

Visual Basic:

APS_set_trigger_param(ByVal Board_ID As Long, ByVal Param_No As Long, ByVal Param_Val As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 Param_No: Parameter number. Refer to trigger parameter table.

I32 Param_Val: Parameter value. Refer to trigger parameter table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example1:

Refer to example of “APS_set_trigger_linear”, “APS_set_trigger_table”

Example2:

Below example is for EMX-100

```
I32 BoardId = 0;
APS_set_trigger_param(BoardId, TGR0_CMP_ENC, 0 ); // Set axis 0 to compare command position
```

Example3:

Below example is for PCI-8254/58 / AMP-204/8C

```
I32 BoardId = 0;
APS_set_trigger_param(BoardId, 0x0, 0 ); //Set linear compare source
```

See also:

APS_get_trigger_param()

APS_get_trigger_param	Get compare trigger related parameter

Support Products: PCI-8253/56,PCI-C154(+), PCI-8154/8158(DB-8150), EMX-100 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get comparing trigger related parameters. All definitions of trigger parameters are described in trigger parameter table.

You can also set parameter using “APS_set_trigger_param()” function.

Syntax:

C/C++:

I32 FNTYPE APS_get_trigger_param(I32 Board_ID, I32 Param_No, I32 *Param_Val);

Visual Basic:

APS_get_trigger_param(ByVal Board_ID As Long, ByVal Param_No As Long, Param_Val As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 Param_No: Parameter number. Refer to trigger parameter table.

I32 Param_Val: Return parameter value. Refer to trigger parameter table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example1:

Below example is for EMX-100

```
I32 BoardId = 0;
I32 Param_Val = 0;
APS_get_trigger_param(BoardId, TGR0_CMP_ENC, &Param_Val );
```

Example2:

Below example is for PCI-8254/58 / AMP-204/8C

```
I32 BoardId = 0;
I32 Param_Val = 0;
APS_get_trigger_param(BoardId, 0x0, &Param_Val ); //Get linear compare source
```

See also:

APS_set_trigger_param()

APS_set_trigger_linear	Set linear comparing function

Support Products: PCI-8253/56,PCI-C154(+), PCI-8154/8158(DB-8150) , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to set linear comparing function.

When the linear trigger operation is completed, the total compared point will be:

Total compared point number = RepeatTimes. (StartPoint as first trigger point)

Syntax:

C/C++:

I32 FNTYPE APS_set_trigger_linear(I32 Board_ID, I32 LCmpCh, I32 StartPoint, I32 RepeatTimes, I32 Interval);

Visual Basic:

APS_set_trigger_linear(ByVal Board_ID As Long, ByVal LCmpCh As Long, ByVal StartPoint As Long, ByVal RepeatTimes As Long, ByVal Interval As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 LCmpCh: Linear compare set channel. Zero base.

For PCI-8254/58 / AMP-204/8C, I32 LCmpCh: Linear compare set channel. Zero base. Range is from 0 to 3.

I32 StartPoint: Start linear trigger point.

I32 RepeatTimes: Trigger repeat times.

I32 Interval: Trigger interval.

For PCI-8253/56, Interval: 24bit unsigned value.

For PCI-8254/58 / AMP-204/8C, I32 Interval: Trigger interval. (-16777215 ~ 16777215, unit is pulse)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 BoardId = 0;
APS_set_trigger_param(BoardId, 0x0, 0 ); //Set linear compare source
APS_set_trigger_param(BoardId, 0x10, 0 ); //Set LCMP0 as TRG0's source
APS_set_trigger_linear(BoardId, 0, 100, 49999, 10 ); //Set LCMP0 linear compare algorithm.
```

```
// Start point = 100, RepeatTimes = 49999, Interval = 10.
APS_set_trigger_param(BoardId, 0x04, 1 ); //Enable LCMP0
  // Trigger operation.

APS_set_trigger_param( 0, 0x04, 0 ); //Disable LCMP0
```

See also:

APS_set_trigger_table()

APS_set_trigger_table	Set table comparing function

Support Products: PCI-8253/56 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to configure the specified comparing table.

Syntax:

C/C++:

I32 FNTYPE APS_set_trigger_table(I32 Board_ID, I32 TCmpCh, I32 *DataArr, I32 ArraySize);

Visual Basic:

APS_set_trigger_table(ByVal Board_ID As Long, ByVal TCmpCh As Long, DataArr As Long, ByVal ArraySize As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TCmpCh: Specified comparing table number. Zero base.

For PCI-8253/56, there two comparing table.

For PCI-8254/58 / AMP-204/8C, I32 TCmpCh: Specified comparing table number. Zero base. Range is from 0 to 3.

I32 *DataArr: Comparing data array.

I32 ArraySize The size of comparing data array. Please refer to product's specification.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
#define POINTS        1000
I32 ret;
I32 data[POINTS];
I32 i;
for( i = 0; i < POINTS; i++ )
        data[i] = 10 + (C) * 10;

APS_set_trigger_param(BoardId, 0x2, 0 ); //Set encoder counter 0 as TCMP0's source.
APS_set_trigger_param(BoardId, 0x10, 4 ); //Set TCMP0 as TRG0's source
```

```
ret = APS_set_trigger_table( 0, 0, data, POINTS );
APS_set_trigger_param(BoardId, 0x06, 1 ); //Enable TCMP0
// Trigger operation…
//When finish the trigger operation.
APS_set_trigger_param(BoardId, 0x06, 0 ); //Enable TCMP0
```

See also:

APS_set_trigger_linear()

APS_set_trigger_manual	Manual output trigger

Support Products: PCI-8253/56,PCI-C154(+), PCI-8154/8158(DB-8150) , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to forced output a trigger at specified trigger output channel.

Syntax:

C/C++:

I32 FNTYPE APS_set_trigger_manual(I32 Board_ID, I32 TrgCh);

Visual Basic:

APS_set_trigger_manual(ByVal Board_ID As Long, ByVal TrgCh As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TrgCh: Trigger output channel (TRG) number. Zero based.

For PCI-8254/58 / AMP-204/8C, I32 TrgCh: Trigger output channel (TRG) number. Zero based. Range is from 0 to 3.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Board_ID = 0;
I32 ret;
ret = APS_set_trigger_manual( Board_ID, 1); //TRG1
```

See also:

APS_set_trigger_manual_s()

APS_set_trigger_manual_s	Manual output trigger synchronously

Support Products: PCI-8253/56/58A/PCI-C154(+), PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to forced to output a trigger pulse. It is designed to output one or more channels of trigger synchronously and manually.

Syntax:

C/C++:

I32 FNTYPE APS_set_trigger_manual_s(I32 Board_ID, I32 TrgChInBit);

Visual Basic:

APS_set_trigger_manual_s(ByValBoard_ID As Long, ByValTrgChInBit As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
ret = APS_set_trigger_manual_s( 0, 0xF ); //4 channels output trigger simultaneously.
Ret = APS_set_trigger_manual_s( 0, 0x2 ); //TRG1 outputs trigger.
Ret = APS_set_trigger_manual_s( 0, 0x3 ); //TRG0 and TRG1 output trigger simultaneously.
//…
```

See also:

APS_set_trigger_manual()

APS_get_trigger_table_cmp	Get current table comparing value

Support Products: PCI-8253/56 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get current comparing value in the specified table comparator.

Syntax:

C/C++:

I32 FNTYPE APS_get_trigger_table_cmp(I32 Board_ID, I32 TCmpCh, I32 *CmpVal);

Visual Basic:

APS_get_trigger_table_cmp(ByVal Board_ID As Long, ByVal TCmpCh As Long, CmpVal As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TCmpCh: Specified the table comparator channel number. Zero base.

For PCI-8254/58 / AMP-204/8C, I32 TCmpCh: Specified the table comparator channel number. Zero base. Range is from 0 to 3.

I32 *CmpVal: Return the current comparing value in the comparator.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 CmpVal;
ret = APS_get_trigger_table_cmp ( 0, 0, &CmpVal );
If( ret != ERR_NoError )
{ // Error, show message.
}
```

See also:

APS_get_trigger_linear_cmp()

APS_get_trigger_linear_cmp	Get current linear comparing value

Support Products: PCI-8253/56,PCI-C154(+), PCI-8154/8158(DB-8150) , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get current comparing value in the specified linear comparator.

Syntax:

C/C++:

I32 FNTYPE APS_get_trigger_linear_cmp(I32 Board_ID, I32 LCmpCh, I32 *CmpVal);

Visual Basic:

APS_get_trigger_linear_cmp(ByVal Board_ID As Long, ByVal LCmpCh As Long, CmpVal As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 LCmpCh: Specified the linear comparator channel number. Zero base.

For PCI-8254/58 / AMP-204/8C, I32 LCmpCh: Specified the linear comparator channel number. Zero base. Range is from 0 to 3.

I32 *CmpVal: Return the current comparing value in the comparator.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 CmpVal;
ret = APS_get_trigger_linear_cmp( 0, 0, &CmpVal );
If( ret != ERR_NoError )
{ // Error, show message.
}
```

See also:

APS_get_trigger_table_cmp()

APS_get_trigger_count	Get triggered count.

Support Products: PCI-8253/56, PCI-C154(+), PCI-8154/8158(DB-8150), EMX-100 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get the triggered counter value. This value means total triggered pulses from last counter reset. It is useful to check compared times.

Syntax:

C/C++:

I32 FNTYPE APS_get_trigger_count(I32 Board_ID, I32 TrgCh, I32 *TrgCnt);

Visual Basic:

APS_get_trigger_count(ByVal Board_ID As Long, ByVal TrgCh As Long, TrgCnt As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TrgCh: Specified trigger output counter channel number. Zero base.

For PCI-8254/58 / AMP-204/8C, I32 TrgCh: Specified trigger output counter channel number. Zero base. Range is from 0 to 3.

For EMX-100: I32 TrgCh: Specified trigger output counter channel number (0 or 1) of device.

I32 *TrgCnt: Return trigger counter value.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Ret;
I32 TrgCnt;
Ret = APS_get_trigger_count( 0, 0, &TrgCnt );
If( ret != ERR_NoError )
{ // Error, show message.
}
```

See also:

APS_reset_trigger_count()

APS_reset_trigger_count	Reset triggered count.

Support Products: PCI-8253/56,PCI-C154(+), PCI-8154/8158(DB-8150), EMX-100 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to reset the triggered counter to zero.

Syntax:

C/C++:

I32 FNTYPE APS_reset_trigger_count(I32 Board_ID, I32 TrgCh);

Visual Basic:

APS_reset_trigger_count(ByVal Board_ID As Long, ByVal TrgCh As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TrgCh: Trigger counter channel number. Zero based.

For EMX-100: I32 TrgCh: Trigger counter channel number(0 or 1) of device.

For PCI-8254/58 / AMP-204/8C, I32 TrgCh: Trigger counter channel number. Zero based. Range is from 0 to 3.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;

ret = APS_reset_trigger_count( 0, 0 );
ret = APS_reset_trigger_count( 0, 1 );
ret = APS_reset_trigger_count( 0, 2 );
ret = APS_reset_trigger_count( 0, 3 );
…
```

See also:

APS_get_trigger_count()

APS_enable_trigger_fifo_cmp	Enable trigger fifo comparator

Support Products: PCI-C154(+), PCI-8154/8158(DB-8150) , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to enable/disable fifo comparator. When user disable the fifo comparator , the fifo data will be reset.

Syntax:

C/C++:

I32 FNTYPE APS_enable_trigger_fifo_cmp(I32 Board_ID, I32 FCmpCh, I32 Enable);

Visual Basic:

APS_enable_trigger_fifo_cmp (ByVal Board_ID As Long, ByVal FCmpCh As Long, ByVal Enable As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 FCmpCh: The specified channel number. (Only support channel 0 in DB-8150)

I32 Enable: Enable/Disable fifo comparator.

0: Disable fifo comparator

1: Enable fifo comparator

Note: Before start FIFO comparing,user must enable fifo comparator first.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Board_ID = 0;
I32 FCmpCh = 0;
I32 Enable = 1 // Enable fifo comparator.
I32 ret = 0;
I32 DataArr[3]={1000,2000,3000};
I32 ArraySize=3;
I32 ShiftFlag = 1; //Auto shift one data to FIFO comparator

ret = APS_set_trigger_fifo_data(Board_ID, FCmpCh, DataArr, ArraySize, ShiftFlag );
ret = APS_enable_trigger_fifo_cmp(Board_ID, FCmpCh, Enable );
```

See also:

APS_get_trigger_fifo_cmp	Get trigger fifo comparator data

Support Products: PCI-C154(+), PCI-8154/8158(DB-8150)

Descriptions:

This function is used to get the current comparing data from FIFO comparator.

Syntax:

C/C++:

I32 FNTYPE APS_get_trigger_fifo_cmp(I32 Board_ID, I32 FCmpCh, I32 *CmpVal);

Visual Basic:

APS_get_trigger_fifo_cmp (ByVal Board_ID As Long, ByVal FCmpCh As Long, *CmpVal As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 FCmpCh: The specified channel number. (Only support channel 0 in DB-8150)

I32 *CmpVal: The current comparing data in comparator.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 Board_ID = 0;

I32 FCmpCh = 0;

I32 CmpVal = 0

I32 ret = 0;

ret = APS_get_trigger_fifo_cmp(Board_ID, FCmpCh, &CmpVal);

See also:

APS_get_trigger_fifo_status	Get fifo status

Support Products: PCI-C154(+), PCI-8154/8158(DB-8150)

Descriptions:

Get the current status of fifo data.

Syntax:

C/C++:

I32 FNTYPE APS_get_trigger_fifo_status(I32 Board_ID, I32 FCmpCh, I32 *FifoSts);

Visual Basic:

APS_get_trigger_fifo_status (ByVal Board_ID As Long, ByVal FCmpCh As Long, FifoSts As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 FCmpCh: The specified channel number. (Only support channel 0 in DB-8150)

I32 * FifoSts: The current status of fifo data.

Bit0=0: not empty , Bit0=1: empty

Bit1=0: not full , Bit1=1; full

Bit2=0: equal or greater than the preset level,

Bit2=1: below the preset level

Other bits be reserved

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 Board_ID = 0;

I32 FCmpCh = 0;

I32 FifoSts = 0

I32 ret = 0;

ret = APS_get_trigger_fifo_status(Board_ID, FCmpCh, & FifoSts);

See also:

APS_set_trigger_fifo_data	Set trigger fifo data

Support Products: PCI-C154(+), PCI-8154/8158(DB-8150)

Descriptions:

This function is used to set comparing data array to the FIFO. The capacity of FIFO is 2097151.

When the status of FIFO is full, the data cannot be set into FIFO. This function won't check the FIFO status. When using this function, you should also enable fifo comparator by"APS_enable_trigger_fifo_cmp" function.

Syntax:

C/C++:

I32 FNTYPE APS_set_trigger_fifo_data(I32 Board_ID, I32 FCmpCh, I32 *DataArr, I32 ArraySize, I32 ShiftFlag);

Visual Basic:

APS_set_trigger_fifo_data (ByVal Board_ID As Long, ByVal FCmpCh As Long, DataArr As Long, ByVal ArraySize As Long, ByVal ShiftFlag As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 FCmpCh: The specified channel number. (Only support channel 0 in DB-8150)

I32 *DataArr : The index pointer of FIFO's data array.

I32 ArraySize : The size of FIFO data array. (1 – 1026)

I32 ShiftFlag : Auto shift one FIFO data to comparator.

0: Disable auto shift one FIFO data to comparator.

1: Enable auto shift one FIFO data to comparator.

Note: Before start FIFO comparing, user must enable auto shift one FIFO data to comparator first.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 Board_ID = 0;

I32 FCmpCh = 0;

I32 DataArr ={1000,2000,3000}

I32 ArraySize = 3;

```
I32 Enable = 1;    // Start FIFO comparing
```

```
I32 ret = 0;
I32 ShiftFlag = 1;    // Enable auto shift one data to FIFO comparator

ret = APS_set_trigger_fifo_data(Board_ID, FCmpCh, DataArr, ArraySize, ShiftFlag );
ret = APS_enable_trigger_fifo_cmp(Board_ID, FCmpCh, Enable );
```

Note : Please do set trigger fifo data first then enable fifo comparator ,the comparator will trigger interrupt normally.

See also:

APS_start_timer	Start / Stop timer

Support Products: PCI-C154(+), PCI-8154/8158(DB-8150)

Descriptions:

In PCI-C154(+), this function is used to Start / Stop timer 8. The timer 8 is used to simulate for encoder, that is used to be comparator source.

Syntax:

C/C++:

32 FNTYPE APS_start_timer(I32 Board_ID, I32 TrgCh, I32 Start);

Visual Basic:

APS_start_timer (ByVal Board_ID As Long, ByVal TrgCh As Long, ByVal Start As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TrgCh: The specified channel number. (In **PCI-C154(+): only support CH0**)

I32 Start: start/stop timer

0: stop timer

1: start timer

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 Board_ID = 0;

I32 TrgCh = 0;

I32 Start = 1 // start timer

I32 ret = 0;

ret = APS_start_timer(Board_ID, TrgCh, Start);

See also:

APS_get_timer_counter	Get timer count value

Support Products: PCI-C154(+), PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get the timer counter value.

In PCI-C154(+), this function is used to get timer 8 count value. The timer 8 is used to simulate for encoder, that is used to be comparator source.

Syntax:

C/C++:

I32 FNTYPE APS_get_timer_counter (I32 Board_ID, I32 TmrCh, I32 *Cnt);

Visual Basic:

APS_get_timer_counter(ByVal Board_ID As Long, ByVal TmrCh As Long, Cnt As Long)As Long

Parameters:

For PCI-C154(+):

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TmrCh: The specified channel number. (In **C154(+): only support CH0**)

I32 *Cnt: Get timer count value.

For PCI-8254/58 / AMP-204/8C:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TmrCh: Specified timer channel number. Zero base.

Only channel 0 is available.

I32 *TmrCnt: Return timer counter value.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example1:

Below example is for PCI-C154(+)

```
I32 ret = 0;
I32 Board_ID = 0;
I32 TmrCh = 0;
I32 Cnt = 0;
ret = APS_get_timer_counter ( Board_ID, TmrCh, &Cnt );
```

Example2:

Below example is for PCI-8254/58 / AMP-204/8C

```
I32 Ret;
I32 TmrCnt;
Ret = APS_get_timer_counter( 0, 0, &TmrCnt ); //Get counter from timer channel 0
If( ret != ERR_NoError )
{ // Error, show message.
}
```

See also:

APS_set_timer_counter()

APS_set_timer_counter	Set timer count value

Support Products: PCI-C154(+), PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to set timer counter.

For PCI-C154(+), this function is used to set timer 8 count value. The timer 8 is used to simulate for encoder, that is used to be comparator source.

Syntax:

C/C++:

I32 FNTYPE APS_set_timer_counter (I32 Board_ID, I32 TmrCh, I32 Cnt);

Visual Basic:

APS_set_timer_counter(ByVal Board_ID As Long, ByVal TmrCh As Long, ByVal Cnt As Long)As Long

Parameters:

For PCI-C154(+):

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TmrCh: The specified channel number. (In **PCI-C154(+): only support CH0**)

I32 Cnt: Set timer count value.

For PCI-8254/58 / AMP-204/8C:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TmrCh: Timer counter channel number. Zero based.

Only one channel is available in PCI-8258.

I32 TmrCnt: Specify timer counter value.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example1:

Below example is for PCI-C154(+)

```
I32 ret = 0;
I32 Board_ID = 0;
I32 TmrCh = 0;
I32 Cnt = 0;
ret = APS_set_timer_counter ( Board_ID, TmrCh, Cnt );
```

Example2:

Below example is for PCI-8254/58 / AMP-204/8C

```
I32 ret;

//set timer counter channel 0 to 100
ret = APS_set_timer_counter( 0, 0, 100 );
…
```

See also:

APS_get_timer_counter()

APS_start_trigger_timer	Start trigger timer

Support Products: PCI-C154(+)

Descriptions:

This function is used to start/stop timers that generate trigger signal periodically

Syntax:

C/C++:

I32 FNTYPE APS_start_trigger_timer (I32 Board_ID, I32 TrgCh, I32 Start);

Visual Basic:

APS_start_trigger_timer (ByVal Board_ID As Long, ByVal TrgCh As Long, ByVal Start As Long)As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TrgCh: The specified channel number.

In **PCI-C154(+): Support CH0 ~ CH3**

I32 Start: Start=1; Start timer

Start=0; Stop timer

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret = 0;
I32 Board_ID = 0;
I32 TrgCh = 0;
I32 Start = 1; // Start timer
ret = APS_set_timer_counter ( Board_ID, TrgCh, Start);
……
Start = 0;// Stop timer
ret = APS_set_timer_counter ( Board_ID, TrgCh, Start);
```

See also:

APS_get_trigger_timer_counter()

APS_get_trigger_timer_counter	Get trigger timer count value

Support Products: PCI-C154(+)

Descriptions:

This function is used to get trigger timer count value that generate trigger signal periodically.

Syntax:

C/C++:

I32 FNTYPE APS_get_trigger_timer_counter (I32 Board_ID, I32 TmrCh, I32 *TmrCnt);

Visual Basic:

APS_get_trigger_timer_counter (ByVal Board_ID As Long, ByVal TmrCh As Long, TmrCnt As Long)As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TmrCh: The specified channel number.

In **PCI-C154(+): Support CH0 ~ CH3**)

I32 *TmrCnt: Get trigger timer count value.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret=0;
I32 Board_ID = 0;
I32 TmrCh = 0;
I32 TmrCnt=0;
ret = APS_get_trigger_timer_counter ( Board_ID, TmrCh, &TmrCnt );
```

See also:

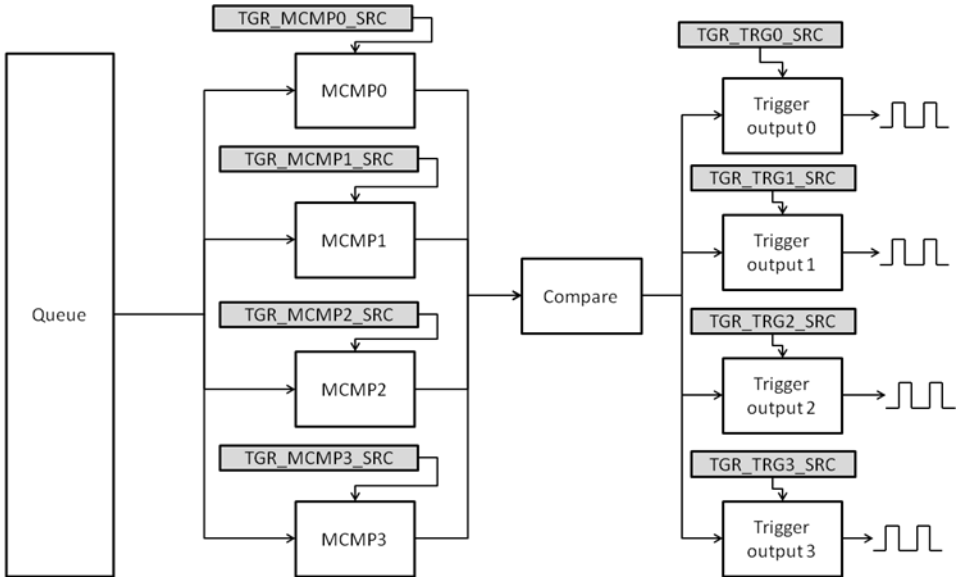
APS_start_trigger_timer()

APS_set_multi_trigger_table	Set table for comparing

Support Products: PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to push data in table (FIFO) for comparing. There are four comparators designed for multi-dimension comparing application. The comparing points are pushed in the queue initially. User can use trigger parameter to select comparator source from encoder 0~7. Specify arbitrary trigger channel to generate PWM is allowed. Once the point is compared, the specified trigger channel will generate one PWM signal and its corresponding counter will add 1 simultaneously.



Comparator Configuration:

Dimension	Configuration
2	Select source in trigger parameter TGR_MCMP0_SRC / TGR_MCMP1_SRC
3	Select source in trigger parameter TGR_MCMP0_SRC / TGR_MCMP1_SRC / TGR_MCMP2_SRC
4	Select source in trigger parameter TGR_MCMP0_SRC / TGR_MCMP1_SRC / TGR_MCMP2_SRC / TGR_MCMP3_SRC

Trigger Output Configuration

Channel	Configuration
0	Set bit 6 in trigger parameter TGR_TRG0_SRC

1	Set bit 6 in trigger parameter TGR_TRG1_SRC
2	Set bit 6 in trigger parameter TGR_TRG2_SRC
3	Set bit 6 in trigger parameter TGR_TRG3_SRC

Syntax:

C/C++:

I32 FNTYPE APS_set_multi_trigger_table(I32 Board_ID, I32 Dimension, MCMP_POINT *Point, I32 PointSize, I32 Window);

Visual Basic:

APS_set_multi_trigger_table(ByVal Board_ID As Long, ByVal Dimension As Long, ByVal DataArr() As MCMP_POINT, ByVal ArraySize As Long, ByVal Window As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Dimention: 2~4 dimension

MCMP_POINT *Point: Point array used for comparator. See description below for details.

```
// Multi-dimension comparator
typedef struct
{
	F64 axisX; // x axis data for multi-dimension comparator 0
	F64 axisY; // y axis data for multi-dimension comparator 1
	F64 axisZ; // z axis data for multi-dimension comparator 2
	F64 axisU; // u axis data for multi-dimension comparator 3
	U32 chInBit; // pwm output channel in bit format
}MCMP_POINT;
```

I32 PointSize: The size of point array.

I32 Window: Specify comparing range

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
void main()
{
	I32 ret = 0;
```

```
U32 i = 0;
I32 BoardID_InBits;
I32 BoardID = 0;
I32 Mode = 0; //By system assigned
I32 msts; // Motion status
MCMP_POINT DataArr[10000];
I32 data = 0;
U32 totalPoint = 5000;
U32 window = 10;
U32 dimension = 2;
I32 Axis_ID_Array[2] = {0, 1};
I32 Distance_Array[2] = {1100, 2200 };
I32 Max_Linear_Speed = 20000;
MCMP_POINT Point;
printf("\n");
// ********************************************************************************
// Initialization
// ********************************************************************************
ret = APS_initial( &BoardID_InBits, Mode);
if(ret)
{
	printf("APS initial fail\n");
	goto TEST_END;
}
printf("APS version = %d\n", (I32)APS_version() );
// ********************************************************************************
// Set trigger parameter
// ********************************************************************************
// Set comparator source: encode 0 for comparator 0 and encoder 1 for comparator 1
ret = APS_set_trigger_param( BoardID, TGR_MCMP0_SRC, 0 );
if(ret)
{
	printf("APS_set_trigger_paraml fail\n");
	goto TEST_END;
}
ret = APS_set_trigger_param( BoardID, TGR_MCMP1_SRC, 1 );
if(ret)
{
	printf("APS_set_trigger_param fail\n");
```

```
        goto TEST_END;
    }

    // Set PWM output channel 0
    ret = APS_set_trigger_param( BoardID, TGR_TRG0_SRC, 0x40 );
    if(ret)
    {
        printf("APS_set_trigger_param fail\n");
        goto TEST_END;
    }

    // Set PWM output channel 1
    ret = APS_set_trigger_param( BoardID, TGR_TRG1_SRC, 0x40 );
    if(ret)
    {
        printf("APS_set_trigger_param fail\n");
        goto TEST_END;
    }

    ret = APS_set_trigger_param( BoardID, TGR_TRG_EN, 0xF );
    ret = APS_set_trigger_param( BoardID, TGR_TRG2_SRC, 0x40 );
    ret = APS_set_trigger_param( BoardID, TGR_TRG3_SRC, 0x40 );

    // Enable all trigger output channel
    ret = APS_set_trigger_param( BoardID, TGR_TRG_EN, 0xF );

    // ****************************************************************************
    // Reset and read trigger count
    // ****************************************************************************
    // Reset PWM channel 0 trigger count
    ret = APS_reset_trigger_count( BoardID, 0 );
    ret = APS_reset_trigger_count( BoardID, 1 );
    ret = APS_reset_trigger_count( BoardID, 2 );
    ret = APS_reset_trigger_count( BoardID, 3 );
    if(ret)
    {
        printf("APS_reset_trigger_count fail\n");
        goto TEST_END;
    }
```

```
// ******************************************************************************
// Set servo on
// ******************************************************************************

// Set axes servo ON
ret = APS_set_servo_on( 0, 1 );
if(ret)
{
	printf("Servo on fail\n");
	goto TEST_END;
}
ret = APS_set_servo_on( 1, 1 );
if(ret)
{
	printf("Servo on fail\n");
	goto TEST_END;
}

// Reset command
ret = APS_set_command( 0, 0 );
if(ret)
{
	printf("APS_set_command fail\n");
	goto TEST_END;
}
ret = APS_set_command( 1, 0 );
if(ret)
{
	printf("APS_set_command fail\n");
	goto TEST_END;
}

// ******************************************************************************
// Set compare points
// ******************************************************************************

// Prepare compare points
for(i=0; i<totalPoint; i++)
```

```
	{
		DataArr[i].axisX = i * 10 + 10;
		DataArr[i].axisY = i * 20 + 20;
		DataArr[i].axisZ = 0;
		DataArr[i].axisU = 0;
		DataArr[i].chInBit = 0xF;
	}

	// Set compare points into queue
	ret = APS_set_multi_trigger_table( BoardID, dimension, DataArr, totalPoint, window );
	if(ret)
	{
		printf("APS_set_multi_trigger_table fail\n");
		goto TEST_END;
	}

	// Check comparator data
	ret = APS_get_multi_trigger_table_cmp( BoardID, dimension, &Point );
	if(ret)
	{
		printf("APS_get_trigger_table_cmp fail\n");
		goto TEST_END;
	}
	printf("Point in comparator: axisX = %f    axisY = %f\n", Point.axisX, Point.axisY );

	// ********************************************************************************
	// Start motor and read status
	// ********************************************************************************

	// Start interpolation
	ret = APS_relative_linear_move( dimension, Axis_ID_Array, Distance_Array,
Max_Linear_Speed );
	if(ret)
	{
		printf("APS_relative_linear_move fail\n");
		goto TEST_END;
	}

	// Check CSTP
```

```
	while(1)
	{
		F64 data1,data2;
		I32 data3, data4, data6, data7;
		U32 data5 = 0;
		U32 data8;
		msts = APS_motion_status(0);

		//ret = APSI_8258_read_fpga( 0, 1, 0x37c, &data5 );
		APS_get_trigger_count( BoardID, 0, &data3 );
		APS_get_trigger_count( BoardID, 1, &data4 );
		APS_get_trigger_count( BoardID, 2, &data6 );
		APS_get_trigger_count( BoardID, 3, &data7 );
		APS_get_position_f( 0, &data1 );
		APS_get_position_f( 1, &data2 );
		printf("fbk0 = %f fbk1 = %f cnt0 = %d cnt1 = %d cnt2 = %d cnt3 = %d ch =
0x%x\n", data1, data2, data3, data4,data6,data7, data5);
		if( msts & 0x1)
			break;
		Sleep(10);
	}

	// ***************************************************************************
	// Read final PWM count
	// ***************************************************************************
	// Check PWM channel 0 trigger counter
	ret = APS_get_trigger_count( BoardID, 0, &data );
	if(ret)
	{
		printf("APS_get_trigger_count fail\n");
		goto TEST_END;
	}
	printf("Final pwm count 0 = %d\n", data );

	// Check PWM channel 1 trigger counter
	ret = APS_get_trigger_count( BoardID, 1, &data );
	if(ret)
	{
		printf("APS_get_trigger_count fail\n");
```

```
            goto TEST_END;
        }
        printf("Final pwm count 1 = %d\n", data );

        // Check PWM channel 2 trigger counter
        ret = APS_get_trigger_count( BoardID, 2, &data );
        if(ret)
        {
            printf("APS_get_trigger_count fail\n");
            goto TEST_END;
        }
        printf("Final pwm count 2 = %d\n", data );

        // Check PWM channel 3 trigger counter
        ret = APS_get_trigger_count( BoardID, 3, &data );
        if(ret)
        {
            printf("APS_get_trigger_count fail\n");
            goto TEST_END;
        }
        printf("Final pwm count 3 = %d\n", data );

TEST_END:

        // Set axes servo off
        ret = APS_set_servo_on( 0, 0 );
        ret = APS_set_servo_on( 1, 0 );
        ret = APS_close();
        system("PAUSE");
}
```

APS_get_multi_trigger_table_cmp	Get current table comparing value

Support Products: PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get current comparing value in the specified table comparator.

Syntax:

C/C++:

I32 FNTYPE APS_get_trigger_table_cmp(I32 Board_ID, I32 Dimension, MCMP_POINT *Point);

Visual Basic:

APS_get_trigger_table_cmp (ByVal Board_ID As Long, ByVal TCmpCh As Long, ByRef CmpVal As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Dimension: 2~4 dimension

MCMP_POINT *Point: Return the current comparing value in comparator. See type_define.h for details.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

APS_set_trigger_table_data	Set table compator data (Fast table compare trigger function)

Below example is for PCI-8254/58 / AMP-204/8C

Descriptions:

This function is belong to fast table compare trigger function and it is used to set comparing data to comparing table. The size of comparing data is constrainted by the FIFO free size, and its maximum value is 40.

Syntax:

C/C++:

I32 FNTYPE APS_set_trigger_table_data(I32 Board_ID, I32 TCmpCh, I32 *DataArr, I32 ArraySize);

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TCmpCh: Specified comparing table number. Zero base. Range is from 0 to 3.

I32 *DataArr: Comparing data array.

I32 ArraySize The size of comparing data array. Its range is 1~40.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 FreeSize = 0;
I32 FifoSts = 0
I32 Data = 0;
I32 DataArr[1000];
U32 usedPoint=0;
U32 residualPoint=0;
U32 totalPoint = 500;

// Enable table compare trigger
ret = APS_enable_trigger_table( 0, 0, 1 );

// Reset FIFO compare data
```

```
ret = APS_reset_trigger_table( 0, 0 );

// Generate compare data
for(i=0; i<totalPoint; i++)
	DataArr[i] = i * 100 + 100;

while(1)
{
	// Get residual data size
	residualPoint = totalPoint - usedPoint;

	// Get FIFO status
	ret = APS_get_trigger_table_status( 0, 0, &FreeSize, &FifoSts );

	// Get current FIFO compare data
	APS_get_trigger_cmp_value( 0, 0, &Data );

	// // Set compare data to FIFO
	if(FreeSize >= 40)
	{
		if(residualPoint >= 40)
		{
			ret = APS_set_trigger_table_data( 0, 0, &DataArr[usedPoint], 40 );
			if(ret ==0)
				usedPoint += 40;
		}
		else
		{
			ret = APS_set_trigger_table_data( 0, 0, &DataArr[usedPoint], residualPoint );
			if(ret ==0)
				usedPoint += residualPoint;
		}
	}
	else
	{
		if(FreeSize >= residualPoint)
		{
			ret = APS_set_trigger_table_data( 0, 0, &DataArr[usedPoint], residualPoint );
```

```
                    if(ret ==0)
                        usedPoint += residualPoint;
            }
            else
            {
                    ret = APS_set_trigger_table_data( 0, 0, &DataArr[usedPoint], FreeSize );
                    if(ret ==0)
                        usedPoint += FreeSize;
            }
        }

        // Complete set compare data to FIFO
        if(usedPoint == totalPoint)
            break;

        Sleep(1);
}
```

See also:

APS_set_trigger_table_data();APS_get_trigger_table_status();APS_get_trigger_cmp_value();
APS_enable_trigger_table();APS_reset_trigger_table()

APS_get_trigger_table_status	Get table comparator status (Fast table compare trigger function)

Below example is for PCI-8254/58 / AMP-204/8C

Descriptions:

This function is belong to fast table compare trigger function and it is used to get the FIFO status of table comparator.

Syntax:

C/C++:

I32 FNTYPE APS_get_trigger_table_status(I32 Board_ID, I32 TCmpCh, I32 *FreeSpace, I32 *FifoSts);

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TCmpCh: Specified comparing table number. Zero base. Range is from 0 to 3.

I32 *FreeSpace: The free size of FIFO. The total free size is 1254.

I32 *FifoSts: The FIFO status: bit 0 = 1 indicates FIFO is full and bit 1 = 1 indicates FIFO is empty.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_set_trigger_table_data();APS_get_trigger_table_status();APS_get_trigger_cmp_value(); APS_enable_trigger_table();APS_reset_trigger_table()

APS_get_trigger_cmp_value	Get table comparator value (Fast table compare trigger function)

Below example is for PCI-8254/58 / AMP-204/8C

Descriptions:

This function is belong to fast table compare trigger function and it is used to get the current comparing value of table comparator.

Syntax:

C/C++:

I32 FNTYPE APS_get_trigger_cmp_value(I32 Board_ID, I32 TCmpCh, I32 *CmpVal);

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TCmpCh: Specified comparing table number. Zero base. Range is from 0 to 3.

I32 *CmpVal: The current coparing value of table comparator.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_set_trigger_table_data();APS_get_trigger_table_status();APS_get_trigger_cmp_value();
APS_enable_trigger_table();APS_reset_trigger_table()

APS_enable_trigger_table	Enable table comparator (Fast table compare trigger function)

Below example is for PCI-8254/58 / AMP-204/8C

Descriptions:

This function is belong to fast table compare trigger function and it is used to enable the table comparator.

Syntax:

C/C++:

I32 FNTYPE APS_enable_trigger_table(I32 Board_ID, I32 TCmpCh, I32 Enable);

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TCmpCh: Specified comparing table number. Zero base. Range is from 0 to 3.

I32 Enable: Set Enable = 1 to begin table comparator.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_set_trigger_table_data();APS_get_trigger_table_status();APS_get_trigger_cmp_value();
APS_enable_trigger_table();APS_reset_trigger_table()

APS_reset_trigger_table	Reset table comparator (Fast table compare trigger function)

Support Products: PCI-8254/58 / AMP-204/8C

Descriptions:

This function is belong to fast table compare trigger function and it used to reset the FIFO of table comparator.

Syntax:

C/C++:

I32 FNTYPE APS_reset_trigger_table(I32 Board_ID, I32 TCmpCh);

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TCmpCh: Specified comparing table number. Zero base. Range is from 0 to 3.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_set_trigger_table_data();APS_get_trigger_table_status();APS_get_trigger_cmp_value();
APS_enable_trigger_table();APS_reset_trigger_table()

19. Program download

APS_load_vmc_program	Load VMC file to task memory

Support Products: PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get VMC file to task memory.

Notice: AMP series don't support this function.

Syntax:

C/C++:

I32 FNTYPE APS_load_vmc_program (I32 Board_ID, I32 TaskNum, const char *pFile, I32 Password);

Visual Basic:

APS_load_vmc_program (ByVal Board_ID As Long, ByVal TaskNum As Long, pFile As String, ByVal Password As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TaskNum: Specify a task number from 0 to 7.

I32 *pFile: Specified a VMC file which created by MCPro2.exe.

I32 Password: Input a specified password for security.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret = 0;
I32 boardId = 0;
I32 taskNum = 0;

//Load a VMC file named "BubbleSort.txt" to specified task.
ret = APS_load_vmc_program( boardId, taskNum, "BubbleSort.txt", 0 );
```

See also:

APS_save_vmc_program()

APS_save_vmc_program	Save to VMC file from task memory

Support Products: PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to save task program from task memory to VMC file.

Notice: AMP series don't support this function.

Syntax:

C/C++:

I32 FNTYPE APS_save_vmc_program(I32 Board_ID, I32 TaskNum, const char *pFile, I32 Password);

Visual Basic:

APS_save_vmc_program (ByVal Board_ID As Long, ByVal TaskNum As Long, pFile As String, ByVal Password As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TaskNum: Specify a task number from 0 to 7.

I32 *pFile: Specify a VMC file to save.

I32 Password: Input a specified password for security.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret = 0;
I32 boardId = 0;
I32 taskNum = 0;

//Save task program to a VMC file named "BubbleSort.txt" .
ret = APS_save_vmc_program( boardId, taskNum, "BubbleSort.txt", 0 );
```

See also:

APS_load_vmc_program()

APS_set_task_mode	Set task run mode

Support Products: PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to set task run mode.

Notice: AMP series don't support this function.

Syntax:

C/C++:

I32 FNTYPE APS_set_task_mode(I32 Board_ID, I32 TaskNum, U8 Mode, U16 LastIP);

Visual Basic:

APS_set_task_mode (ByVal Board_ID As Long, ByVal TaskNum As Long, ByVal Mode As Byte, ByVal LastIP As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TaskNum: Specify a task number from 0 to 7.

U8 Mode: Two run mode to set.

0: Normal mode. 1: Repeat mode.

U16 LastIP: Last instruction offset. It is only available in repeat mode.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret = 0;
I32 boardId = 0;
I32 taskNum = 0;
U16 lastIP = 0;

//Set task 0 to normal mode. IP is igonored in normal mode.
ret = APS_set_task_mode ( boardId, taskNum, 0, &lastIP );
```

See also:

APS_get_task_mode ()

APS_get_task_mode	Get task run mode

Support Products: PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get task run mode.

Notice: AMP series don't support this function.

Syntax:

C/C++:

I32 FNTYPE APS_get_task_mode(I32 Board_ID, I32 TaskNum, U8 *Mode, U16 *LastIP);

Visual Basic:

APS_get_task_mode (ByVal Board_ID As Long, ByVal TaskNum As Long, Mode As Byte, ByVal LastIP As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TaskNum: Specify a task number from 0 to 7.

U8 *Mode: Two run mode.

0: Normal mode. 1: Repeat mode.

U16 *LastIP: Last instruction offset. It is only available in repeat mode.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret = 0;
I32 boardId = 0;
I32 taskNum = 0;
U8 mode = 0;
U16 lastIP = 0;

//Get run mode from task 0. IP is igonored in normal mode.
ret = APS_get_task_mode ( boardId, taskNum, &mode, &lastIP);
```

See also:

APS_set_task_mode ()

APS_start_task	Start task control command

Support Products: PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to start task control command.

Notice: AMP series don't support this function.

Syntax:

C/C++:

I32 FNTYPE APS_start_task(I32 Board_ID, I32 TaskNum, I32 CtrlCmd);

Visual Basic:

APS_start_task (ByVal Board_ID As Long, ByVal TaskNum As Long, ByVal CtrlCmd As Long)

As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TaskNum: Specify a task number from 0 to 7.

I32 CtrlCmd: Control command.

0: TSK_RESET. Reset task. (Not start Program)

1: TSK_RESTART. Restart task. (Start program at the same time)

2: TSK_STOP. Stop Program.

3: TSK_RUN. Start program.

4: TSK_STEP. Step(Run) one instrucation. Then stop.

5: TSK_STEP_P. Run until parallel bit == 0

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret = 0;
I32 boardId = 0;
I32 taskNum = 0;
I32 CtrlCmd = 3;

//Run the program of task 0
ret = APS_start_task ( boardId, taskNum, CtrlCmd );
```

See also:

APS_get_task_info(); APS_get_task_msg()

APS_get_task_info	Get task information

Support Products: PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get task information.

Notice: AMP series don't support this function.

Syntax:

C/C++:

I32 FNTYPE APS_get_task_info(I32 Board_ID, I32 TaskNum, TSK_INFO *Info);

Visual Basic:

APS_get_task_info (ByVal Board_ID As Long, ByVal TaskNum As Long, pFile As String, ByRef Info As TSK_INFO) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TaskNum: Specify a task number from 0 to 7.

TSK_INFO *Info: Task information.

```
typedef struct _TSK_INFO
{
    U16 State;              //Task state: 0:Stop, 1: Run, 2: Step, 3:Step_p 4: ??
    U16 RunTimeErr;    //runtime error code when state is in ERROR state.
    U16 IP; //Register IP
    U16 SP; //Register SP
    U16 BP; //Register BP
    U16 MsgQueueSts; //Message queue status, refer to following definition
} TSK_INFO, *PTSK_INFO;
```

U16 MsgQueueSts: (Note: All tasks share only one massage queue.)

BitNum	Status description
0	MPU_MSG_EMPTY: Message queue empty
1	MPU_MSG_FULL: Queue full
2	MPU_MSG_NOT_EMPTY
3~15	Reserved. 0

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret = 0;
I32 boardId = 0;
I32 taskNum = 0;
TSK_INFO info;

//Get information of task 0
 ret = APS_get_task_info( boardId, taskNum, &info );
```

See also:

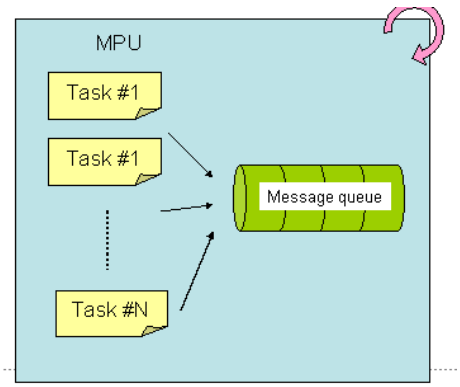
APS_start_task(); APS_get_task_msg()

APS_get_task_msg	Get message of all tasks

Support Products: PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get task message. All tasks share only one message queue. This is useful for debug. User could output some debug string to message queue.



Notice: AMP series don't support this function.

Syntax:

C/C++:

I32 FNTYPE APS_get_task_msg(I32 Board_ID, U16 *QueueSts, U16 *ActualSize, U8 *CharArr);

Visual Basic:

APS_get_task_msg (ByVal Board_ID As Long, ByRef QueueSts As UShort, ByRef ActualSize As UShort, ByRef CharArr As Byte) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

U16 *QueueSts: Message queue status. Refer to following chart.

U16 *ActualSize: Actual return message size. [0~128]

U8 *CharArr: Return char message. Maximum array size is 128 bytes.(U8 CharArr[n = 128])

It depends on ActualSize, if n > = ActualSize, data is meaningless and could be ignored.

QueueSts definition:

BitNum	Status description
0	MPU_MSG_EMPTY: Message queue empty

1	MPU_MSG_FULL: Queue full
2	MPU_MSG_NOT_EMPTY
3~15	Reserved. 0

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret = 0;
I32 boardId = 0;
U16 queueSts = 0;
U16 actualSize = 0;
U8 charArr[128];

//Get message of all tasks
  ret = APS_get_task_msg(boardId, &queueSts, actualSize, &charArr );;
```

See also:

APS_start_task(); APS_get_task_info()

20. Manual Pulse Generator functions

APS_manual_pulser_start	Enable/Disable PA/PB input./ Start manual pulser operation

Support Products: PCIe-8154/58, PCI-C154(+), PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

For PCIe-8154/58, PCI-C154(+) , this function is used to enable/disable PA/PB input. When disabling manual pulser, the pulse signal inputted from PA/PB pin is ignored.
For PCI-C154+, default setting is enabling mode.

For PCI-8254/58 / AMP-204/8C , this function is used to start manual pulser operation. It supports one set PA/PB pin to connect handy manual pulse generator and decoder for single axis position control. The decoder allows the input signal type from PA and PB pins being plus and minus pulses (CW/CCW), OUT/DIR, or 90 degrees phase difference signals (AB phase) respectively. User should select correct input signal type based on the specification of handy manual pulser generator carefully. If necessary, inversing PA and PB signals or changing counting direction are also allowed. These setting can be configured by axis parameters.
After issuing “enable” command, the manual pulser operation will keep waiting to receive new input signals and then move motor immediately. The bit 29 of motion status is able to indicate the manual pulser operation is enabled or disabled. Befor using this function, user should use APS_manual_pulser_velocity_move() to specify which axis and its maximum velocity. For safety consideration, this pulser operation will terminate immediately when

1. The “disable” command is issued by user.
2. The motion IO such that PEL/MEL, ALM, and EMG are triggered.

For PCIe-833x or ECAT-4XMO, this function is used to start manual pulser operation. It supports one set PA/PB pin to connect handy manual pulse generator and decoder for single axis position control. The decoder allows the input signal type from PA and PB pins being plus and minus pulses (CW/CCW), OUT/DIR, or 90 degrees phase difference signals (AB phase) respectively. User should select correct input signal type based on the specification of handy manual pulser generator carefully. If necessary, inversing PA and PB signals or changing counting direction are also allowed. These setting can be configured by axis parameters.
After issuing “enable” command, the manual pulser operation will keep waiting to receive new input signals and then move motor immediately. The bit 29 of motion status is able to indicate

the manual pulser operation is enabled or disabled. Befor using this function, user should use APS_manual_pulser_velocity_move() to specify which axis and its maximum velocity. For safety consideration, this pulser operation will terminate immediately when

1. The “disable” command is issued by user.
2. The motion IO such that PEL/MEL, ALM, and EMG are triggered.

For PCI-8254/58 / AMP-204/8C / PCIe-833x or ECAT-4XMO, this function must disable and enable again while any one axis parameter related pulser has been modified.

Syntax:

C/C++:

I32 FNTYPE APS_manual_pulser_start (I32 Board_ID, I32 Enable);

Visual Basic:

APS_ manual_pulser_start (ByVal Board_ID As Long, ByVal Enable As Long) As Long

Parameters:

For PCIe-8154/58, PCI-C154(+):

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 Enable: Enable/disable PA/PB input. 0: disable, 1: enable.

For PCI-8254/58 / AMP-204/8C or PCIe-833x, ECAT-4XMO:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial ().

I32 Enable: Enable pulser operation.

1: enable manual pulser operation

0: disable manual pulser operation

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example1:

Below example is for PCIe-8154/58, PCI-C154(+)

```
I32 ret;

ret = APS_manual_pulser_start (0, 1 ); //enable pulser input.
```

Example2:

Below example is for PCI-8254/58 / AMP-204/8C or PCIe-833x, ECAT-4XMO

```
ret = APS_manual_pulser_start( BoardID, 0 ); // Disable pulser process
ret = APS_set_axis_param( Axis, PRA_PSR_IPT_MODE, 2 );
```

```
// Set input mode: 0: 1xAB; 2: 4xAB
ret = APS_set_axis_param( Axis, PRA_PSR_IPT_LOGIC, 0 );
//Set logic: 0: InvPA = 0, InvPB = 0
ret = APS_set_axis_param( Axis, PRA_PSR_IPT_DIR, 0 );
  // Set direction: 0: InvPA = 0, InvPB = 0

ret = APS_set_axis_param_f( Axis, PRA_PSR_RATIO_VALUE, 1 ); // Set ratio
ret = APS_set_axis_param_f( Axis, PRA_PSR_ACC, 123456 ); // Set acceleration
ret = APS_set_axis_param_f( Axis, PRA_PSR_JERK, 12345678 ); // Set jerk

ret = APS_manual_pulser_velocity_move( Axis, 12345 ); // Start velocity move
ret = APS_manual_pulser_start( BoardID, 1 ); // Enable pulser

// If reset pulser related parameter….
ret = APS_manual_pulser_start( BoardID, 0 ); // Disable pulser process
ret = APS_set_axis_param_f( Axis, PRA_PSR_RATIO_VALUE, 2 ); // Reset ratio
ret = APS_manual_pulser_start( BoarID, 1 ); // Re-Enable pulser process
```

See also:

APS_manual_pulser_velocity_move()

APS_manual_pulser_velocity_move	Begin a pulser velocity move/ Start velocity move in manual pulser operation

Support Products: PCIe-8154/58, PCI-C154(+), PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

For PCIe-8154/58, PCI-C154(+) , this function is used to start a pulser velocity move. The axis will output one pulse when receiving one pulse from pulser input with default value, user can set the ratio between output and input pulse via axis parameter 164h and 165h. The axis could stop pulser function when users issue a stop move command.

User could specify a limited speed to pulser function. For example, if SpeedLimit is set to be 100 pps, then the axis would move at fastest 100 pps, even the input pulser signal rate is more than 100 pps.

When a pulser move function is issued, there are some situations of motion status as below:

1. When pulser function is issued, bit 29(PAPB) of motion status will turn on. It means to wait the signal from PA/PB input.

2.When continuous signal is inputted from PA/PB pin, bit 10(VS) of motion status will turn on. It means the axis output pulse according pulser input by user's specified speed.

For PCI-8254/58 / AMP-204/8C and PCIe-833x or ECAT-4XMO, this function will start velocity move in manual pulser operation.

Syntax:

For PCIe-8154/58, PCI-C154(+):

C/C++:

I32 FNTYPE APS_manual_pulser_velocity_move(I32 Axis_ID, F64 SpeedLimit);

Visual Basic:

APS_manual_pulser_velocity_move (ByVal Axis_ID As Long, ByVal SpeedLimit As Double) As Long

For PCI-8254/58 / AMP-204/8C:

I32 FNTYPE APS_manual_pulser_velocity_move (I32 Axis_ID, F64 MaxVelocity)

APS_manual_pulser_velocity_move (ByVal Axis_ID As Long, ByVal MaxVelocity As Double) As Long

Parameters:

For PCIe-8154/58, PCI-C154(+):

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 SpeedLimit: The maximum limited speed of this move profile. Unit: pulse/sec

For PCI-8254/58 / AMP-204/8C or PCIe-833x, ECAT-4XMO:

I32 Axis_ID: Axis number 0~7

F64 MaxVelocity: Maximum speed in manual pulser operation. It's value should be larger than zero.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example1:

Below example is for PCIe-8154/58, PCI-C154(+)

```
I32 ret;

ret = APS_manual_pulser_start (0, 1 ); //enable pulser input.
ret =APS_manual_pulser_velocity_move (Axis_ID, 1000 ); // Begin a pulser velocity move
```

Example2:

Below example is for PCI-8254/58 / AMP-204/8C or PCIe-833x, ECAT-4XMO:

```
ret = APS_manual_pulser_start( BoardID, 0 ); // Disable pulser process
ret = APS_set_axis_param( Axis, PRA_PSR_IPT_MODE, 2 );
// Set input mode: 0: 1xAB; 2: 4xAB
ret = APS_set_axis_param( Axis, PRA_PSR_IPT_LOGIC, 0 );
//Set logic: 0: InvPA = 0, InvPB = 0
ret = APS_set_axis_param( Axis, PRA_PSR_IPT_DIR, 0 );
  // Set direction: 0: InvPA = 0, InvPB = 0

ret = APS_set_axis_param_f( Axis, PRA_PSR_RATIO_VALUE, 1 ); // Set ratio
ret = APS_set_axis_param_f( Axis, PRA_PSR_ACC, 123456 ); // Set acceleration
ret = APS_set_axis_param_f( Axis, PRA_PSR_JERK, 12345678 ); // Set jerk

ret = APS_manual_pulser_velocity_move( Axis, 12345 ); // Start velocity move
ret = APS_manual_pulser_start( BoardID, 1 ); // Enable pulser
```

See also:

APS_manual_pulser_start()

APS_manual_pulser_relative_move	Begin a pulser relative distance move

Support Products: PCIe-8154/58, PCI-C154(+)

Descriptions:

This function is used to start a pulser relative move. The axis will output one pulse when receiving one pulse from pulser input with default value, user can set the ratio between output and input pulse via axis parameter 164h and 165h. The axis could stop pulser function when users issue a stop move command.

User could specify a limited speed to pulser function. For example, if SpeedLimit is set to be 100 pps, then the axis would move at fastest 100 pps, even the input pulser signal rate is more than 100 pps.

When a pulser move function is issued, there are some situations of motion status as below:

1. When pulser function is issued, bit 29(PAPB) of motion status will turn on. It means to wait the signal from PA/PB input.

2.When continuous signal is inputted from PA/PB pin, bit 10(VS) of motion status will turn on. It means the axis output pulse according pulser input by user's specified speed.

Syntax:

C/C++:

I32 FNTYPE APS_manual_pulser_relative_move(I32 Axis_ID, F64 Distance, F64 SpeedLimit);

Visual Basic:

APS_manual_pulser_relative_move (ByVal Axis_ID As Long, ByVal Distance As Double, ByVal SpeedLimit As Double) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Distance: Relative distance. Unit is pulse.

I32 SpeedLimit: The maximum limited speed of this move profile. Unit: pulse/sec.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 ret;

```
ret = APS_manual_pulser_start (0, 1 ); //enable pulser input.
ret =APS_manual_pulser_relative_move (Axis_ID, 100, 1000 ); // Begin a pulser relative move
```

APS_manual_pulser_home_move	Begin a pulser home move

Support Products: PCIe-8154/58, PCI-C154(+)

Descriptions:

This function is used to start a pulser home move. The axis will output one pulse when receiving one pulse from pulser input with default value, user can set the ratio between output and input pulse via Axis Parameter 164h and 165h. The axis could stop pulser function when users issue a stop move command.

For pulser home function, user would refer to axis parameter table to set a specified home type (166h) & home limited speed (167h).

User could specify a limited speed to pulser function. For example, if SpeedLimit is set to be 100 pps, then the axis would move at fastest 100 pps, even the input pulser signal rate is more than 100 pps.

When a pulser move function is issued, there are some situations of motion status as below:

1. When pulser function is issued, bit 29(PAPB) of motion status will turn on. It means to wait the signal from PA/PB input.
2. When continuous signal is inputted from PA/PB pin, bit 10(VS) of motion status will turn on. It means the axis output pulse according pulser input by user's specified speed.

Syntax:

C/C++:

I32 FNTYPE APS_manual_pulser_home_move(I32 Axis_ID);

Visual Basic:

APS_manual_pulser_home_move (ByVal Axis_ID As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;

ret = APS_manual_pulser_start (0, 1 ); //enable pulser input.
ret =APS_manual_pulser_home_move (Axis_ID ); // Begin a pulser home move
```

APS_get_pulser_counter	Get pulser counter

Support Products: PCI-8253/56, DPAC-1000, DPAC-3000

Descriptions:

This function is used to get the counter value of pulser. Pulser is a short term of manual pulse generator. It is a device for manually generating industrial counter pulses. The device sometime calls “hand wheel”.

Syntax:

C/C++:

I32 FNTYPE APS_get_pulser_counter(I32 Board_ID, I32 *Counter);

Visual Basic:

APS_get_pulser_counter(ByVal Board_ID As Long, Counter As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 *Counter: Return the value of pulser counter.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Counter;

ret = APS_get_pulser_counter(0, &Counter );
if( ret == ERR_NoError )
    //Show counter value.
```

APS_set_pulser_counter	Set DPAC pluse input counter

Support Products: DPAC-1000, DPAC-3000

Descriptions:

For DPAC, This function is used to set input pulses counter's numbers.

Syntax:

C/C++:

I32 FNTYPE APS_set_pulser_counter (I32 Board_ID, I32 Counter);

Visual Basic:

APS_set_pulser_counter (ByVal Board_ID As Long, ByVal Counter As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Counter: Input pulses counter's numbers.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Counter;

Counter = 0; //Set Input pulses counter=0
ret = APS_set_pulser_counter (0, Counter );
if( ret == ERR_NoError )
    //Show counter value.
```

See also:

APS_get_pls_iptcounter()

21. Pitch error compensation functions

APS_set_pitch_table	Set data for pitch error compensation table.

Support Products: PCI-8254/58 / AMP-204/8C

Description:

Figure 1 introduces two coordinate systems in PCI-8254/8: one is user coordinate, and the other is motor coordinate. Specifying arbitrary values for command position and feedback position in user coordinate are allowed here. Usually, after machine returns to its home position, both of them will be set to zero immediately. On the other hand, user is forbidden to change command counter and encoder counter in motor coordinate since they are used for servo loop control. It is noticed that the data type of these two coordinate are double and integer respectively. The process of pitch error compensation is introduced in Figure 1 now. Its input is current command position, and its output obtained from a look-up table is error compensation. The actual value of command counter will be the resultant of command position and pitch error compensation.

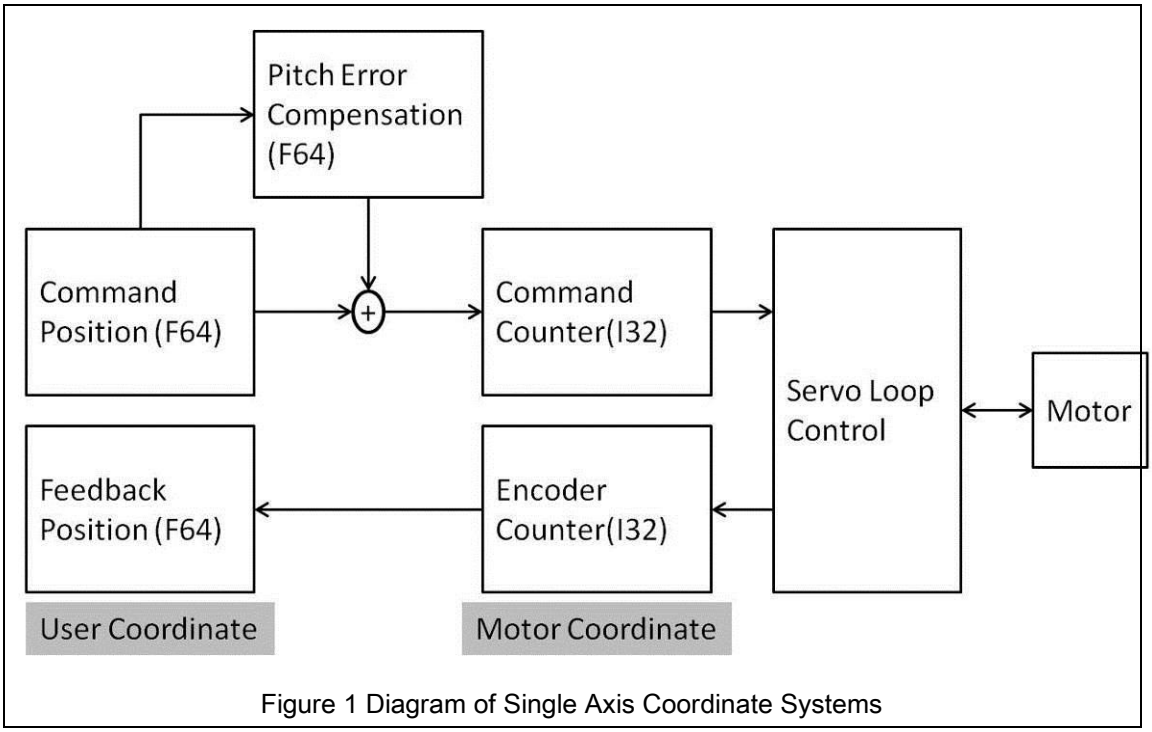


Figure 1 Diagram of Single Axis Coordinate Systems

The pitch error compensation data is used for each compensation position at the intervals specified for each axis. The origin of compensation is the home position that the machine is returned. The compensation data is signed value and it is set with respect to home position

(usually, compensation data at home position is zero). In order to perform pitch error compensation, it is also necessary to set configurations such that minimum position, interval of compensation position, and total points. A pitch error compensation table can be built successfully based on these configurations and compensation data. There are two types of compensation methods for user: constant type and linear type. After finishing all settings above, user can enable the pitch error compensation via APS function. It should be noticed that if the machine stroke exceeds the specified range on either the positive direction or the negative direction, the pitch error compensation does not apply beyond the range, so the compensation value will be zero. The unit used in pitch error compensation is pulse (count). Figure 2 is an example of pitch error compensation table. The solid line and dashed line denotes the two compensation types. User specifies minimum position, interval, and total points are 0, 100, and 5. So the maximum position will be 500. And the compensation data at each compensation position are 0, 1, 2, -1, and 1. There is no compensation if command position is outside the range from 0 to 500. After machine returns home position (command position is zero), the error compensation is also zero.

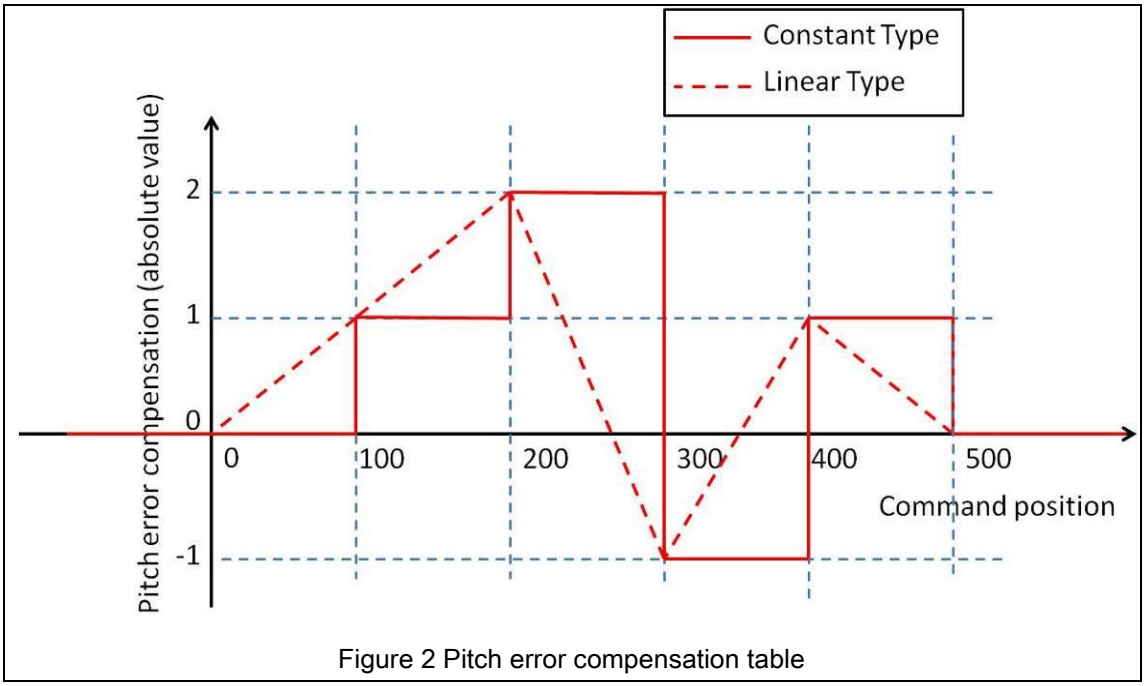


Figure 2 Pitch error compensation table

Table 1 shows the results of using pitch error compensation table in Figure 2. For example, if command position is set to 150.0, for constant compensation type the command counter will be 151; on the contrary, for linear compensation type, the command counter will be 152. This is due to truncation error.

	Command position(F64)	0.0	50.0	100.0	150.0	200.0	250.0	300.0	350.0	400.0	450.0	500.0
Constant compensation	Error Compensation (F64)	0.0	0.0	1.0	1.0	2.0	2.0	-1.0	-1.0	1.0	1.0	0.0
	Command counter(I32)	0	50	101	151	202	252	299	349	401	451	500
Linear compensation	Error Compensation (F64)	0.0	0.5	1.0	1.5	2.0	0.5	-1.0	0.0	1.0	0.5	0.0
	Command counter (I32)	0	51	101	152	202	251	299	350	401	451	500

Table 1 Results using different compensation method

Syntax:

C/C++:

I32 FNTYPE APS_set_pitch_table(I32 Axis_ID, I32 Comp_Type, I32 Total_Points, I32 MinPosition, U32 Interval, I32 *Comp_Data);

Visual Basic:

APS_set_pitch_table (ByVal Axis_ID As Long, ByVal Comp_Type As Long, ByVal Total_Points As Long, ByVal MinPosition As Long, ByVal Interval As UInteger, ByVal Comp_Data() As UInteger) As Long

Parameters:

I32 Axis_ID: 0~7

I32 Comp_Type: Compensation type; 0: Constant compensation; 1: Linear compensation

I32 Total_Points: Total amounts of compensation data; Maximum value is 500.

I32 MinPosition: Minimum position in pitch error compensation table

U32 Interval: Interval between two compensation points in pitch error compensation table

I32 *Comp_Data: Compensation data in pitch error compensation table

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
void main()
{
	I32 ret; // function return
	I32 BoardID_InBits; // for initialization
	I32 Axis_ID = 0; // axis ID
	I32 MotionStatus; // motion status in bits
```

```
        I32 Comp_Data[5] = {0, 1, 2, -1, 1}; // pitch error compensation data
        I32 CommandPosition; // command position
        I32 CommandCount; // command counter
        I32 Comp_Type = 0; // compensation type
        I32 Total_Points = 5; // total points
        I32 MinPosition = 0; // minimum command position
        U32 Interval = 100; // interval
        I32 i;

        printf("/* Start pitch error compensation demo */ \n");

        // Initialization
        ret = APS_initial( &BoardID_InBits, 0 );
        if(ret)
        {
                printf("APS library initial fail! \n");
                goto END_PROGRAM;
        }

        // Set servo on
        ret = APS_set_servo_on( Axis_ID, 1 );
        if(ret)
        {
                printf("Set servo on fail! \n");
                goto END_PROGRAM;
        }

        // Get current command position and command counter
        APS_get_command(Axis_ID, &CommandPosition );
        APS_get_command_counter(Axis_ID, &CommandCount );
        printf("Command position = %d Command count = %d \n",CommandPosition,
CommandCount );

        // Start home process
        printf("Return to home position... \n");
        ret = APS_home_move( Axis_ID );
        if(ret)
        {
                printf("Start home fail! \n");
```

```
            goto END_PROGRAM;
        }

        // Check home is done
        do{
            MotionStatus = APS_motion_status( Axis_ID ); //Get Motion status
        }while ( ( MotionStatus>>5 & 0x1 ) == 0 );

        // Get command position and command counter at home position
        APS_get_command(Axis_ID, &CommandPosition );
        APS_get_command_counter(Axis_ID, &CommandCount );
        printf("Command position = %d Command count = %d \n",CommandPosition,
CommandCount );

        // Set pitch error compensation table
        ret = APS_set_pitch_table( Axis_ID, Comp_Type, Total_Points, MinPosition, Interval,
Comp_Data);
        if(ret)
        {
            printf("Set pitch error compensation data and configuration fail! \n");
            goto END_PROGRAM;
        }

        // Start pitch error compensation
        ret = APS_start_pitch_comp( Axis_ID, 1 );
        if(ret)
        {
            printf("Start pitch error compensation fail! \n");
            goto END_PROGRAM;
        }

        // Start PTP to test pitch error compensation
        for( i=0; i<Total_Points; i++ )
        {
            ret = APS_absolute_move( Axis_ID, 100+i*100, 10000 );
            if(ret)
            {
                printf("Start PTP fail! \n");
                goto END_PROGRAM;
```

```
            }

            // Check PTP is done
            do{
                MotionStatus = APS_motion_status( Axis_ID ); //Get Motion status
            }while ( ( MotionStatus>>5 & 0x1 ) == 0 );

            // Get command position and command counter
            APS_get_command(Axis_ID, &CommandPosition );
            APS_get_command_counter(Axis_ID, &CommandCount );
            printf("Command position = %d Command count = %d \n",CommandPosition,
CommandCount );
        }

        // Set servo off
        ret = APS_set_servo_on( Axis_ID, 0 );
        if(ret)
        {
            printf("Set servo off fail! \n");
            goto END_PROGRAM;
        }

        // Stop pitch error compensation
        ret = APS_start_pitch_comp( Axis_ID, 0 );
        if(ret)
        {
            printf("Stop pitch error compensation fail! \n");
            goto END_PROGRAM;
        }

END_PROGRAM:
        printf("/* Stop pitch error compensation demo */ \n");
        system("pause");
```

See also:

APS_get_pitch_table();APS_start_pitch_comp()

APS_get_pitch_table	Get data from pitch error compensation table.

Support Products: PCI-8254/58 / AMP-204/8C

Description:

Get configurations and data of pitch error compensation table

Syntax:

C/C++:

I32 FNTYPE APS_get_pitch_table(I32 Axis_ID, I32 *Comp_Type, I32 *Total_Points, I32 *MinPosition, U32 *Interval, I32 *Comp_Data);

Visual Basic:

APS_get_pitch_table (ByVal Axis_ID As Long, ByRef Comp_Type As Long, ByRef Total_Points As Long, ByRef MinPosition As Long, ByRef Interval As UInteger, ByRef Comp_Data As UInteger) As Long

Parameters:

I32 Axis_ID: 0~7

I32 *Comp_Type: Compensation type; 0: Constant compensation; 1: Linear compensation

I32 *Total_Points: Amounts of compensation data

I32 *MinPosition: Minimum position in pitch error compensation table

U32 *Interval: Interval between two compensation points in pitch error compensation table

I32 *Comp_Data: Compensation data in pitch error compensation table

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See sample program in APS_set_pitch_table

See also:

APS_set_pitch_table();APS_start_pitch_comp()

APS_start_pitch_comp	Start pitch error compensation

Support Products: PCI-8254/58 / AMP-204/8C

Description:

Start pitch error compensation table

Syntax:

C/C++:

I32 FNTYPE APS_start_pitch_comp(I32 Axis_ID, I32 Enable);

Visual Basic:

APS_start_pitch_comp (ByVal Axis_ID As Long, ByVal Enable As Long) As Long

Parameters:

I32 Axis_ID: 0~7

I32 Enable: 0: Disable error compensation: 1: Enable error compensation

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See sample program in APS_set_pitch_table

See also:

APS_set_pitch_table();APS_get_pitch_table()

22. DPAC System functions

APS_rescan_CF	Rescan DPAC Slave CF slot

Support Products: DPAC-1000, DPAC-3000

Descriptions:

This function is used to rescan DPAC external slave CF slot. When system is started into Windows, the right-down corner has an icon to manage removable devices like USB flash. If users remove DPAC's external CF which is an USB device from the management icon, there is not possible to re-scan it by un-plug and plug CF card from external CF slot. Users must call this function to activate the re-scan.

Syntax:

C/C++:

I32 FNTYPE APS_rescan_CF (I32 Board_ID);

Visual Basic:

APS_rescan_CF (ByVal Board_ID As Long) As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
ret = APS_rescan_CF ( 0 );
if( ret != ERR_NoError )
{
	// Error, show message.
}
```

See also:

APS_get_battery_status	Get DPAC SRAM Battery status

Support Products: DPAC-1000, DPAC-3000

Descriptions:

This function is used to get DPAC SRAM battery status. There is a SRAM on DPAC which is for users to store in a very fast way. The SRAM can be a non-volatile storage if the battery is installed on DPAC. Users can use this function to know the status of the battery. Notice that if there is no battery installed on DPAC, this function will return you battery high status but actually SRAM has no function for non-volatile storage. Please check the bettery exists first.

Syntax:

C/C++:

I32 FNTYPE APS_get_battery_status(I32 Board_ID, I32 *Battery_status);

Visual Basic:

APS_get_battery_status(ByVal Board_ID As Long, Battery_status As Long) As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

I32 *Battery_status: 1: Normal, 0: Low.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Battery_status;
ret = APS_get_battery_status ( 0, &Battery_status );
if( ret == ERR_NoError )
{
	//Show Battery status.
}
```

See also:

APS_get_display_data	Get 7-Segment LED Data

Support Products: DPAC-1000, DPAC-3000

Descriptions:

This function is used to get 7-Segment LED's data. There are five digits on DPAC LED. Each digit can display one character. If the character is a number, it can display one character and an additional dot sign too.

Syntax:

C/C++:

I32 FNTYPE APS_get_display_data(I32 Board_ID, I32 displayDigit, I32 *displayIndex);

Visual Basic:

APS_get_display_data (ByVal Board_ID As Long, ByVal displayDigit As Long, displayIndex As Long) As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

I32 displayDigit: 7-Segment No. (1~5)

I32 * displayIndex: Reference to DPAC display index table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 displayNum;
ret = APS_get_display_data( 0, 1, &displayNum );
if( ret == ERR_NoError )
{
    //  The displayNum variable shows digit one's display number
}
```

See also:

APS_set_display_data();DPAC diplay index table()

APS_set_display_data	Set 7-Segment LED Data

Support Products: DPAC-1000, DPAC-3000

Descriptions:

This function is used to set 7-Segment LED's data and display. There are five digits on DPAC LED. Each digit can display one character. If the character is a number, it can display one character and an additional dot sign too.

Syntax:

C/C++:

I32 FNTYPE APS_set_display_data(I32 Board_ID, I32 displayDigit, I32 displayIndex);

Visual Basic:

APS_set_display_data (ByVal Board_ID As Long, ByVal displayDigit As Long, ByVal displayIndex As Long) As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

I32 displayDigit: 7-Segment No. (1~5)

I32 displayIndex: Reference to displayIndex table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 displayNum;

displayNum=0x01;
ret = APS_set_display_data ( 0, 1, displayNum ); // It will display '1' on first digit of LEDs
if( ret != ERR_NoError )
{
    // Error, show message.
}
```

See also:

APS_get_display_data()

APS_get_button_status	Get the Push Button Input Status

Support Products: DPAC-1000, DPAC-3000

Descriptions:

This function is used to get push button Istatus of DPAC. There are 4 buttons on DPAC. Each button is click type. That means when you release the pushing, the button will be back to its original position.

Syntax:

C/C++:

I32 FNTYPE APS_get_button_status (I32 Board_ID, I32 *buttonstatus);

Visual Basic:

APS_get_button_status (ByVal Board_ID As Long, buttonstatus As Long) As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

I32 *buttonstatus: Reference to buttonstatus table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 buttonstatus;
ret = APS_get_button_status ( 0, &buttonstatus );
if( ret == ERR_NoError )
{
        //Show button status.
}
Else
{
    "check B3 ON/OFF"
    1) Read button status
    2) To get a new button status by 'NOT' button status
    3) Maps B3 to Bit# by "Bit#=(4 – B#)'. We get Bit1.
    4) Use Bit1 (0010b) to 'AND' new button status
    5) If the result is zero, it means B3 is not pushed.
    6) If the result is non-zero, it means B3 is pushed.
```

}

See also:

DPAC push button status table()

23. Non-Volatile RAM

APS_set_nv_ram	Set NVRAM data

Support Products: DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-7856

Descriptions:

This function is used to write a value to NVRAM. NVRAM means non-volatile memory. It can store user's data permanently even system power is off.

PCI-8144 uses EEPROM as NVRAM. It ganrentee 1,000,000 times write access.

Syntax:

C/C++:

I32 FNTYPE APS_set_nv_ram(I32 Board_ID, I32 RamNo, I32 DataWidth, I32 Offset, I32 Data);

Visual Basic:

APS_set_nv_ram (ByVal Board_ID As Long, ByVal RamNo As Long, ByVal DataWidth As Long, ByVal Offset As Long, ByVal Data As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 RamNo: RamNo=0(DPAC,PCI(e)-7856)

I32 DataWidth: 0: RW_WIDTH_8; 1: RW_WIDTH_16; 2: RW_WIDTH_32(PCI(e)-7856 Only)

I32 Offset: The Offset from 0x0000 to 0x75FF(DPAC).; The Offset from 0x0000 to 0x7FFF(PCI(e)-7856)

I32 Data: DataWidth: 0 The Data from -128 to 127.; (DPAC,PCI(e)-7856)

DataWidth: 1 The Data from -32768 to 32767.; (DPAC,PCI(e)-7856)

DataWidth: 2 The Data from -2147483648 to 2147483647.; (PCI(e)-7856 Only)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 ret;

I16 Data;

```
Data=0x5168;
ret = APS_set_nv_ram (0, 0, 1, 0x1000,Data );
//Write RAM (offset =0x1000) value=0x5168. DataWidth: 1
if( ret != ERR_NoError )
{
	// Error, show message.
}
```

See also:

APS_get_nv_ram()

APS_get_nv_ram	Get NVRAM data

Support Products: DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-7856

Descriptions:

This function is used to read a value from NVRAM. NVRAM means non-volatile memory. It can store user's data permanently. It means even system power is off, the data is still in the memory. Next time when system is recovered, users can get the data by this function.

Syntax:

C/C++:

I32 FNTYPE APS_get_nv_ram(I32 Board_ID, I32 RamNo, I32 DataWidth, I32 Offset, I32 *Data);

Visual Basic:

APS_get_nv_ram (ByVal Board_ID As Long, ByVal RamNo As Long, ByVal DataWidth As Long, ByVal Offset As Long, Data As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 RamNo: RamNo=0(DPAC,PCI(e)-7856)

I32 DataWidth: 0: RW_WIDTH_8; 1: RW_WIDTH_16; 2: RW_WIDTH_32(PCI(e)-7856 Only)

I32 Offset: The Offset from 0x0000 to 0x75FF(DPAC).; The Offset from 0x0000 to 0x7FFF(PCI(e)-7856)

I32 *Data: DataWidth: 0 The Data from -128 to 127.; (DPAC,PCI(e)-7856)

DataWidth: 1 The Data from -32768 to 32767.; (DPAC,PCI(e)-7856)

DataWidth: 2 The Data from -2147483648 to 2147483647.; (PCI(e)-7856 Only)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Data;

ret = APS_get_nv_ram (0, 0, 1, 0x1000,&Data );
if( ret == ERR_NoError )
    //Show RAM (offset =0x1000) DataWidth: 1 value.
```

See also:

APS_set_nv_ram()

APS_clear_nv_ram	Clear NVRAM data

Support Products: DPAC-1000, DPAC-3000, PCI-8144, PCI(e)-7856

Descriptions:

This function is used to clear all values on NVRAM. NVRAM means non-volatile memory. It can store user's data permanently even system power is off. Once this function is issued, all data stored in this memory will be clear.

PCI-8144 uses EEPROM as NVRAM. It ganrentee 1,000,000 times write access.

Syntax:

C/C++:

I32 FNTYPE APS_clear_nv_ram(I32 Board_ID, I32 RamNo);

Visual Basic:

APS_clear_nv_ram (ByVal Board_ID As Long, ByVal RamNo As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 RamNo: RamNo=0(DPAC,PCI(e)-7856)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
ret = APS_clear_nv_ram ( 0, 0 ); //Clear RamNo=0 data
if( ret != ERR_NoError )
{
    // Error, show message.
}
```

See also:

APS_set_nv_ram();APS_get_nv_ram();APS_clear_nv_ram()

24. Field bus compare trigger

APS_set_field_bus_trigger_param	Set compare trigger related parameter

Support Products: MNET-4XMO-C, HSL-4XMO, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to set comparing trigger related parameters. All definitions of trigger parameters are described in trigger parameter table.

You can also get parameter setting using “APS_get_field_bus_trigger_param ()” function.

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_trigger_param(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 Param_No, I32 Param_Val);

Visual Basic:

APS_set_field_bus_trigger_param (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal Param_No As Long, ByVal Param_Val As Long) As Long

Parameters:

For MNET-4XMO-C, HSL-4XMO:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For HSL field bus, the range fo module number is 1 to 63. In HSL, the Module_No is the first id occupied by the module.

For MNET field bus, the range of module number is 0 to 63.

I32 Param_No: Parameter number. Refer to trigger parameter table.

I32 Param_Val: Parameter value. Refer to trigger parameter table.

For ECAT-4XMO , ECAT-TRG4:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (start from 0).

I32 Param_No: Parameter number. Refer to trigger parameter table.

I32 Param_Val: Parameter value. Refer to trigger parameter table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for MNET-4XMO-C, HSL-4XMO:

Refer to example of “APS_set_field_bus_trigger_linear”, “APS_set_field_bus_trigger_table”

Below example is for ECAT-4XMO , ECAT-TRG4:

I32 BoardId = 0, BusNo = 0, ModNo = 0;

APS_set_field_bus_trigger_param (BoardId, BusNo, ModNo, 0x0, 0); //Set linear compare source

See also:

APS_get_field_bus_trigger_param()

APS_get_field_bus_trigger_param	Get compare trigger related parameter

Support Products: MNET-4XMO-C, HSL-4XMO, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to get comparing trigger related parameters. All definitions of trigger parameters are described in trigger parameter table.

You can also set parameter using “APS_set_field_bus_trigger_param()” function.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_trigger_param(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 Param_No, I32 *Param_Val);

Visual Basic:

APS_get_field_bus_trigger_param(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal Param_No As Long, Param_Val As Long) As Long

Parameters:

For MNET-4XMO-C, HSL-4XMO:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For HSL field bus, the range of module number is 1 to 63. In HSL, the Module_No is the first id occupied by the module.

For MNET field bus, the range of module number is 0 to 63.

I32 Param_No: Parameter number. Refer to trigger parameter table.

I32 Param_Val: Return parameter value. Refer to trigger parameter table.

For ECAT-4XMO , ECAT-TRG4:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (start from 0).

I32 Param_No: Parameter number. Refer to trigger parameter table.

I32 Param_Val: Return parameter value. Refer to trigger parameter table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for ECAT-4XMO , ECAT-TRG4:

```
I32 BoardId = 0, BusNo = 0, ModNo = 0;
I32 Param_Val = 0;
APS_get_field_bus_trigger_param (BoardId, BusNo, ModNo, 0x0, &Param_Val ); //Get linear compare source
```

See also:

APS_set_field_bus_trigger_param()

APS_set_field_bus_trigger_linear	Set linear comparing function

Support Products: MNET-4XMO-C, HSL-4XMO, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to set linear comparing function.

When the linear trigger operation is completed, the total compared point will be:

For MNET-4XMO-C, Total compared point number = RepeatTimes.

For ECAT-4XMO , ECAT-TRG4: Total compared point number = RepeatTimes. (StartPoint as first trigger point)If any compare/trigger setting be changed, need to use this function again.

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_trigger_linear(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 LCmpCh, I32 StartPoint, I32 RepeatTimes, I32 Interval);

Visual Basic:

APS_set_field_bus_trigger_linear(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No As Long, ByVal LCmpCh As Long, ByVal StartPoint As Long, ByVal RepeatTimes As Long, ByVal Interval As Long) As Long

Parameters:

For MNET-4XMO-C, HSL-4XMO:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For HSL field bus, the range of module number is 1 to 63. In HSL, the Module_No is the first id occupied by the module.

For MNET field bus, the range of module number is 0 to 63.

I32 LCmpCh: Linear compare set channel.

For MNET-4XMO-C, the range of LCmpCh is 0 to 4. (LCmpCh 0~3 is used for general comparator, and LCmpCh 4 is used for high speed comparator.)

I32 StartPoint: Start linear trigger point.

I32 RepeatTimes: Trigger repeat times.

For MNET_4XMO-C, Interval: 31bit unsigned value. (Value: 1 ~ 0x7fffffff)

I32 Interval: Trigger interval.

For ECAT-4XMO , ECAT-TRG4:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (start from 0).

I32 LCmpCh: Specified the linear comparator channel number. Zero base. Range is from 0 to 3.

I32 StartPoint: Start linear trigger point.

I32 RepeatTimes: Trigger repeat times.

I32 Interval: Trigger interval.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for MNET-4XMO-C, HSL-4XMO:

I32 BoardId = 0, Bus_No = 1, Mod_No = 0

APS_set_field_bus_trigger_param(BoardId, Bus_No, Mod_No, 0x0, 1); //Set CMP0 as linear type

APS_set_field_bus_trigger_param(BoardId, Bus_No, Mod_No, 0x10, 1); //Set CMP0 as TRG0's source

APS_set_field_bus_trigger_linear(BoardId, Bus_No, Mod_No, 0, 1000, 100000, 100); //Set CMP0 linear compare algorithm.

// Start point = 1000, RepeatTimes = 100000, Interval = 100.

APS_set_field_bus_trigger_param(BoardId, Bus_No, Mod_No, 0x04, 1); //Enable CMP0

// Trigger operation…

//When finish the trigger operation.

APS_set_field_bus_trigger_param(BoardId, Bus_No, Mod_No, 0x04, 0); //Disable CMP0

Below example is for ECAT-4XMO:

I32 BoardId = 0, Bus_No = 0, Mod_No = 0

APS_set_field_bus_trigger_param(BoardId, Bus_No, Mod_No, 0x0, 0); //Set linear compare source

APS_set_field_bus_trigger_param(BoardId, Bus_No, Mod_No, 0x10, 1); //Set LCMP0 as TRG0's source

// Set LCMP0 linear compare algorithm. Start point = 1000, RepeatTimes = 100000, Interval = 100.

APS_set_field_bus_trigger_linear(BoardId, Bus_No, Mod_No, 0, 1000, 100000, 100);

Below example is for ECAT-ECAT-TRG4:

I32 BoardId = 0, Bus_No = 0, Mod_No = 0

APS_set_field_bus_trigger_param(BoardId, Bus_No, Mod_No, 0x0, 0); //Set linear compare source

APS_set_field_bus_trigger_param(BoardId, Bus_No, Mod_No, 0x10, 1); //Set LCMP0 as TRG0's source

APS_set_field_bus_trigger_param(BoardId, Bus_No, Mod_No, 0x50, 5); //Set external source from which Axis ID = 5

// Set LCMP0 linear compare algorithm. Start point = 1000, RepeatTimes = 100000, Interval = 100.

APS_set_field_bus_trigger_linear(BoardId, Bus_No, Mod_No, 0, 1000, 100000, 100);

See also:

APS_set_field_bus_trigger_table()

APS_set_field_bus_trigger_table	Set table comparing function

Support Products: MNET-4XMO-C, HSL-4XMO, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to configure the specified comparing table.

For ECAT-4XMO , ECAT-TRG4:

The comparing table would be consumed while comparing matched point. If any compare/trigger setting be changed, need to configure the specified comparing table.

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_trigger_table(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 TCmpCh, I32 *DataArr, I32 ArraySize);

Visual Basic:

APS_set_field_bus_trigger_table(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No, ByVal TCmpCh As Long, DataArr As Long, ByVal ArraySize As Long) As Long

Parameters:

For MNET-4XMO-C, HSL-4XMO:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For HSL field bus, the range of module number is 1 to 63. In HSL, the Module_No is the first id occupied by the module.

For MNET field bus, the range of module number is 0 to 63.

I32 TCmpCh: Specified comparing table number.

For MNET-4XMO-C, the range of TCmpCh is 0 to 3. (TCmpCh 0~3 is used for general comparator.)

I32 *DataArr: Comparing data array.

I32 ArraySize: Size of comparing data array.

For MNET-4XMO-C, the maximum size of each channel = 8192.

For ECAT-4XMO , ECAT-TRG4:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (start from 0).

I32 TCmpCh: Specified the table comparator channel number. Zero base. Range is from 0 to 3.

I32 *DataArr: Comparing data array.

I32 ArraySize: The size of comparing data array. Please refer to product's specification.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for MNET-4XMO-C, HSL-4XMO:

```
#define POINTS 5000
I32 BoardId = 0;
I32 Bus_No = 1;
I32 Mod_No = 0;
I32 ret;
I32 data[POINTS];
I32 i;
for( i = 0; i < POINTS; i++ )
    data[i] = 10 +( i * 10 );
APS_set_field_bus_trigger_param(BoardId, Bus_No, Mod_No, 0x0, 0 ); //Set CMP0 as table type
APS_set_field_bus_trigger_param(BoardId, Bus_No, Mod_No, 0x10, 1 ); //Set CMP0 as TRG0's source
ret = APS_set_field_bus_trigger_table(BoardId, Bus_No, Mod_No, 0, data, POINTS );
APS_set_field_bus_trigger_param(BoardId, Bus_No, Mod_No, 0x04, 1 ); //Enable CMP0
// Trigger operation…
//When finish the trigger operation.
APS_set_field_bus_trigger_param(BoardId, Bus_No, Mod_No, 0x04, 0 ); //Disable CMP0
```

Below example is for ECAT-4XMO:

```
#define POINTS 5000
I32 BoardId = 0, Bus_No = 0, Mod_No = 0, ret = 0, i = 0;
I32 data[POINTS];
for( i = 0; i < POINTS; i++ )
    data[i] = 10 + ( i * 10 );
// Set encoder counter 0 as TCMP0's source.
APS_set_field_bus_trigger_param( BoardId, Bus_No, Mod_No, 0x2, 0 );
// Set TCMP0 as TRG0's source.
APS_set_field_bus_trigger_param( BoardId, Bus_No, Mod_No, 0x10, 4 );
// Set TCMP0 compare direction to bi-direction
```

```
APS_set_field_bus_trigger_param( BoardId, Bus_No, Mod_No, 0x4, 2 );

// Start table compare.
ret = APS_set_field_bus_trigger_table( BoardId, Bus_No, Mod_No, 0, data, POINTS );
```

Below example is for ECAT-ECAT-TRG4:

```
#define POINTS 5000
I32 BoardId = 0, Bus_No = 0, Mod_No = 0, ret = 0, i = 0;
I32 data[POINTS];
for( i = 0; i < POINTS; i++ )
        data[i] = 10 + ( i * 10 );
// Set encoder counter 0 as TCMP0's source.
APS_set_field_bus_trigger_param( BoardId, Bus_No, Mod_No, 0x2, 0 );
// Set TCMP0 as TRG0's source.
APS_set_field_bus_trigger_param( BoardId, Bus_No, Mod_No, 0x10, 4 );
// Set TCMP0 compare direction to bi-direction
APS_set_field_bus_trigger_param( BoardId, Bus_No, Mod_No, 0x4, 2 );
//Set external source from which Axis ID = 5
APS_set_field_bus_trigger_param( BoardId, Bus_No, Mod_No, 0x50, 5 );

// Start table compare.
ret = APS_set_field_bus_trigger_table( BoardId, Bus_No, Mod_No, 0, data, POINTS );
```

See also:

APS_set_field_bus_trigger_linear()

APS_set_field_bus_trigger_manual	Manual output trigger

Support Products: MNET-4XMO-C, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to forced output a trigger at specified trigger output channel.

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_trigger_manual(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 TrgCh);

Visual Basic:

APS_set_field_bus_trigger_manual(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No, ByVal TrgCh As Long) As Long

Parameters:

For MNET-4XMO-C, HSL-4XMO:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For MNET field bus, the range of module number is 0 to 63.

I32 TrgCh: Trigger output channel (TRG) number. Zero based.

For MNET-4XMO-C, the range of TrgCh is 0 to 3.

For ECAT-4XMO , ECAT-TRG4:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (start from 0).

I32 TrgCh: Trigger output channel (TRG) number. Zero based. Range is from 0 to 3.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for MNET-4XMO-C, HSL-4XMO:

```
I32 BoardId = 0;
I32 Bus_No = 1;
I32 Mod_No = 0;
I32 ret;
ret = APS_set_field_bus_trigger_manual(BoardId, Bus_No, Mod_No, 0); //TRG0
```

Below example is for ECAT-4XMO , ECAT-TRG4:

```
I32 BoardId = 0, Bus_No = 0, Mod_No = 0;
I32 ret;
ret = APS_set_field_bus_trigger_manual( BoardId, Bus_No, Mod_No, 0 ); //TRG0
```

See also:

APS_set_field_bus_trigger_manual_s()

APS_set_field_bus_trigger_manual_s	Manual output trigger synchronously

Support Products: MNET-4XMO-C, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to forced output a trigger.

By this function, all output channels output trigger synchronously is possible.

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_trigger_manual_s(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 TrgChInBit);

Visual Basic:

APS_set_field_bus_trigger_manual_s(ByValBoard_ID As Long, ByVal BUS_No As Long, ByVal MOD_No, ByValTrgChInBit As Long) As Long

Parameters:

For MNET-4XMO-C, HSL-4XMO:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For MNET field bus, the range of module number is 0 to 63.

I32 TrgChInBit: 1: Output trigger, 0: Don't output trigger

For MNET-4XMO-C : Bit0: TRG0, Bit1: TRG1, Bit2: TRG2, Bit3: TRG3

For ECAT-4XMO , ECAT-TRG4:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (start from 0).

I32 TrgChInBit: Assign trigger channel by bit define. Define as bellows:

Bit0: TRG0, Bit1: TRG1, Bit2: TRG2, Bit3: TRG3

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for MNET-4XMO-C, HSL-4XMO:

```
I32 BoardId = 0;
I32 Bus_No = 1;
I32 Mod_No = 0;
I32 ret;
ret = APS_set_field_bus_trigger_manual_s(BoardId, Bus_No, Mod_No, 0xF ); //4 channels output trigger simultaneously.
ret = APS_set_field_bus_trigger_manual_s(BoardId, Bus_No, Mod_No, 0x2 ); //TRG1 outputs trigger.
ret = APS_set_field_bus_trigger_manual_s( 0, 0x3 ); //TRG0 and TRG1 output trigger simultaneously.
//…
```

Below example is for ECAT-4XMO , ECAT-TRG4:

```
I32 BoardId = 0, Bus_No = 0, Mod_No = 0, ret = 0;
// 4 channels output trigger simultaneously.
ret = APS_set_field_bus_trigger_manual_s( BoardId, Bus_No, Mod_No, 0xF );
// TRG1 outputs trigger.
ret = APS_set_field_bus_trigger_manual_s( BoardId, Bus_No, Mod_No, 0x2 );
// TRG0 and TRG1 output trigger simultaneously.
ret = APS_set_field_bus_trigger_manual_s( BoardId, Bus_No, Mod_No, 0x3 );
```

See also:

APS_set_field_bus_trigger_manual()

APS_get_field_bus_trigger_table_cmp	Get current table comparing value

Support Products: MNET-4XMO-C, HSL-4XMO, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to get current comparing value in the specified table comparator.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_trigger_table_cmp(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 TCmpCh, I32 *CmpVal);

Visual Basic:

APS_get_field_bus_trigger_table_cmp(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No, ByVal TCmpCh As Long, CmpVal As Long) As Long

Parameters:

For MNET-4XMO-C, HSL-4XMO:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For HSL field bus, the range of module number is 1 to 63. In HSL, the Module_No is the first id occupied by the module.

For MNET field bus, the range of module number is 0 to 63.

I32 TCmpCh: Specified the table comparator channel number. Zero base.

For MNET-4XMO-C, the range of TCmpCh is 0 to 3. (TCmpCh 0~3 is used for general comparator.)

I32 *CmpVal: Return the current comparing value in the comparator.

For ECAT-4XMO , ECAT-TRG4:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (start from 0).

I32 TCmpCh: Specified the table comparator channel number. Zero base. Range is from 0 to 3.

I32 *CmpVal: Return the current comparing value in the comparator.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for MNET-4XMO-C, HSL-4XMO:

```
I32 BoardId = 0;
I32 Bus_No = 1;
I32 Mod_No = 0;
I32 ret;
I32 CmpVal;
ret = APS_get_field_bus_trigger_table_cmp (BoardId, Bus_No, Mod_No, 0, &CmpVal );
If( ret != ERR_NoError )
{ // Error, show message.}
```

Below example is for ECAT-4XMO , ECAT-TRG4:

```
I32 BoardId = 0, Bus_No = 0, Mod_No = 0, ret = 0, CmpVal = 0;
// Get TCMP0 current compare point.
ret = APS_get_field_bus_trigger_table_cmp( BoardId, Bus_No, Mod_No, 0, &CmpVal );
If( ret != ERR_NoError )
{ // Error, show message. }
```

See also:

APS_get_field_bus_trigger_linear_cmp()

APS_get_field_bus_trigger_linear_cmp	Get current linear comparing value

Support Products: MNET-4XMO-C, HSL-4XMO, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to get current comparing value in the specified linear comparator.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_trigger_linear_cmp(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 LCmpCh, I32 *CmpVal);

Visual Basic:

APS_get_field_bus_trigger_linear_cmp(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No, ByVal LCmpCh As Long, CmpVal As Long) As Long

Parameters:

For MNET-4XMO-C, HSL-4XMO:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For HSL field bus, the range of module number is 1 to 63. In HSL, the Module_No is the first id occupied by the module.

For MNET field bus, the range of module number is 0 to 63.

I32 LCmpCh: Specified the linear comparator channel number. Zero base.

For MNET-4XMO-C, the range of LCmpCh is 0 to 4. (LCmpCh 0~3 is used for general comparator, and LCmpCh 4 is used for high speed comparator.)

I32 *CmpVal: Return the current comparing value in the comparator.

For ECAT-4XMO , ECAT-TRG4:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (start from 0).

I32 LCmpCh: Specified the linear comparator channel number. Zero base. Range is from 0 to 3.

I32 *CmpVal: Return the current comparing value in the comparator.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for MNET-4XMO-C, HSL-4XMO:

```
I32 BoardId = 0;
I32 Bus_No = 1;
I32 Mod_No = 0;
I32 ret;
I32 CmpVal;
ret = APS_get_field_bus_trigger_linear_cmp(BoardId, Bus_No, Mod_No, 0, &CmpVal );
If( ret != ERR_NoError )
{ // Error, show message.}
```

Below example is for ECAT-4XMO , ECAT-TRG4:

```
I32 BoardId = 0, Bus_No = 0, Mod_No = 0, ret = 0, CmpVal = 0;
// Get LCMP0 current compare point.
ret = APS_get_field_bus_trigger_linear_cmp(BoardId, Bus_No, Mod_No, 0, &CmpVal );
If( ret != ERR_NoError )
{// Error, show message.}
```

See also:

APS_get_field_bus_trigger_table_cmp()

APS_get_field_bus_trigger_count	Get triggered count.

Support Products: MNET-4XMO-C, ECAT-4XMO , ECAT-TRG4

Descriptions:

For MNET-4XMO-C, HSL-4XMO:

This function is used to get the triggered counter.

You can use this function to check how many trigger pulse be output.

Using **APS_reset_field_bus_trigger_count()** to reset the counter to zero.

For ECAT-4XMO , ECAT-TRG4:

This function is used to get the triggered counter value.

This value means total triggered pulses from last counter reset.

It is useful to check compared times.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_trigger_count(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 TrgCh, I32 *TrgCnt);

Visual Basic:

APS_get_field_bus_trigger_count(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No, ByVal TrgCh As Long, TrgCnt As Long) As Long

Parameters:

For MNET-4XMO-C, HSL-4XMO:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For MNET field bus, the range of module number is 0 to 63.

I32 TrgCh: Specified trigger output counter channel number. Zero base.

For MNET-4XMO-C, the range of TrgCh is 0 to 3.

I32 *TrgCnt: Return trigger counter value.

For ECAT-4XMO , ECAT-TRG4:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (start from 0).

I32 TrgCh: Specified trigger output counter channel number. Zero base. Range is from 0 to 3.

I32 *TrgCnt: Return trigger counter value.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for MNET-4XMO-C, HSL-4XMO:

```
I32 BoardId = 0;
I32 Bus_No = 1;
I32 Mod_No = 0;
I32 Ret;
I32 TrgCnt;
Ret = APS_get_field_bus_trigger_count(BoardId, Bus_No, Mod_No, 0, &TrgCnt );
If( ret != ERR_NoError )
{ // Error, show message.}
```

Below example is for ECAT-4XMO , ECAT-TRG4:

```
I32 BoardId = 0, Bus_No = 0, Mod_No = 0, ret = 0, TrgCnt = 0;
// Get TRG0 current trigger count.
ret = APS_get_field_bus_trigger_count( BoardId, Bus_No, Mod_No, 0, &TrgCnt );
If( ret != ERR_NoError )
{// Error, show message.}
```

See also:

APS_reset_field_bus_trigger_count()

APS_reset_field_bus_trigger_count	Reset triggered count.

Support Products: MNET-4XMO-C, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to reset the triggered counter to zero.

Syntax:

C/C++:

I32 FNTYPE APS_reset_field_bus_trigger_count(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 TrgCh);

Visual Basic:

APS_reset_field_bus_trigger_count(ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No, ByVal TrgCh As Long) As Long

Parameters:

For MNET-4XMO-C, HSL-4XMO:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For MNET field bus, the range of module number is 0 to 63.

I32 TrgCh: Trigger counter channel number. Zero based.

For MNET-4XMO-C, the range of TrgCh is 0 to 3.

For ECAT-4XMO , ECAT-TRG4:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (start from 0).

I32 TrgCh: Specified trigger output counter channel number. Zero base. Range is from 0 to 3.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for MNET-4XMO-C, HSL-4XMO:

```
I32 BoardId = 0;
I32 Bus_No = 1;
I32 Mod_No = 0;
I32 ret;
ret = APS_reset_field_bus_trigger_count(BoardId, Bus_No, Mod_No, 0 );
ret = APS_reset_field_bus_trigger_count(BoardId, Bus_No, Mod_No, 1 );
ret = APS_reset_field_bus_trigger_count(BoardId, Bus_No, Mod_No, 2 );
ret = APS_reset_field_bus_trigger_count(BoardId, Bus_No, Mod_No, 3 );
…
```

Below example is for ECAT-4XMO , ECAT-TRG4:

```
I32 BoardId = 0, Bus_No = 0, Mod_No = 0, ret = 0;
ret = APS_reset_field_bus_trigger_count(BoardId, Bus_No, Mod_No, 0 );
ret = APS_reset_field_bus_trigger_count(BoardId, Bus_No, Mod_No, 1 );
ret = APS_reset_field_bus_trigger_count(BoardId, Bus_No, Mod_No, 2 );
ret = APS_reset_field_bus_trigger_count(BoardId, Bus_No, Mod_No, 3 );
```

See also:

APS_get_field_bus_trigger_count()

APS_get_field_bus_linear_cmp_remain_count	Get remaining counter of linear comparator

Support Products: MNET-4XMO-C, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to get remaining counter of linear comparator.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_linear_cmp_remain_count(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 LCmpCh, I32 *Cnt);

Visual Basic:

APS_get_field_bus_linear_cmp_remain_count (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No, ByVal LCmpCh As Long, Cnt As Long) As Long

Parameters:

For MNET-4XMO-C, HSL-4XMO:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For MNET field bus, the range of module number is 0 to 63.

I32 LCmpCh: Specified the linear comparator channel number. Zero base.

For MNET-4XMO-C, the range of LCmpCh is 0 to 4. (LCmpCh 0~3 is used for general comparator, and LCmpCh 4 is used for high speed comparator.)

I32 *Cnt: Remaining counter.

For ECAT-4XMO , ECAT-TRG4:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (start from 0).

I32 LCmpCh: Specified the linear comparator channel number. Zero base. Range is from 0 to 3.

I32 Cnt: Remaining count value.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for MNET-4XMO-C, HSL-4XMO:

```
I32 BoardId = 0;
I32 Bus_No = 1;
I32 Mod_No = 0;
I32 ret;
I32 Cnt;
ret = APS_get_field_bus_linear_cmp_remain_count (BoardId, Bus_No, Mod_No, 0, &Cnt );
If( ret != ERR_NoError )
{ // Error, show message.}
```

Below example is for ECAT-4XMO , ECAT-TRG4:

```
I32 BoardId = 0, Bus_No = 0, Mod_No = 0, ret = 0, Cnt = 0;
// Get LCMP0 remain count.
ret = APS_get_field_bus_linear_cmp_remain_count ( BoardId, Bus_No, Mod_No, 0, &Cnt );
If( ret != ERR_NoError )
{ // Error, show message.}
```

See also:

APS_get_field_bus_table_cmp_remain_count()

APS_get_field_bus_table_cmp_remain_count	Get remaining counter of table comparator

Support Products: MNET-4XMO-C, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to get remaining counter of table comparator.

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_table_cmp_remain_count(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 TCmpCh, I32 *Cnt);

Visual Basic:

APS_get_field_bus_table_cmp_remain_count (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No, ByVal TCmpCh As Long, Cnt As Long) As Long

Parameters:

For MNET-4XMO-C, HSL-4XMO:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For MNET field bus, the range of module number is 0 to 63.

I32 TCmpCh: Specified the table comparator channel number. Zero base.

For MNET-4XMO-C, the range of TCmpCh is 0 to 3. (TCmpCh 0~3 is used for general comparator.)

I32 *Cnt: Remaining counter.

For ECAT-4XMO , ECAT-TRG4:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (start from 0).

I32 TCmpCh: Specified the table comparator channel number. Zero base. Range is from 0 to 3.

I32 Cnt: Remaining count value.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for MNET-4XMO-C, HSL-4XMO:

```
I32 BoardId = 0;
I32 Bus_No = 1;
I32 Mod_No = 0;
I32 ret;
I32 Cnt;
ret = APS_get_field_bus_table _cmp_remain_count (BoardId, Bus_No, Mod_No, 0, &Cnt );
If( ret != ERR_NoError )
{ // Error, show message.}
```

Below example is for ECAT-4XMO , ECAT-TRG4:

```
I32 BoardId = 0, Bus_No = 0, Mod_No = 0, ret = 0, Cnt = 0;
// Get LCMP0 remain count.
ret = APS_get_field_bus_linear_cmp_remain_count ( BoardId, Bus_No, Mod_No, 0, &Cnt );
If( ret != ERR_NoError )
{ // Error, show message. }
```

See also:

APS_get_field_bus_linear_cmp_remain_count()

APS_get_field_bus_encoder	Get encoder count.

Support Products: MNET-4XMO-C, ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to get encoder count

Syntax:

C/C++:

I32 FNTYPE APS_get_field_bus_encoder(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 EncCh, I32 *EncCnt);

Visual Basic:

APS_get_field_bus_encoder (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No, ByVal EncCh As Long, EncCnt As Long) As Long

Parameters:

For MNET-4XMO-C, HSL-4XMO:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For MNET field bus, the range of module number is 0 to 63.

I32 EncCh: Specified the encoder channel number. Zero base.

For MNET-4XMO-C, the range of EncCh is 0 to 4. (EncCh 0~3 is used for general comparator, and LCmpCh 4 is used for high speed comparator.)

I32 * EncCnt: Encoder count.

For ECAT-4XMO , ECAT-TRG4:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (start from 0).

I32 EncCh: Specified the encoder channel number. Zero base. Range is from 0 to 3.

I32 EncCnt: Encoder count.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

Below example is for MNET-4XMO-C, HSL-4XMO:

```
I32 BoardId = 0;
I32 Bus_No = 1;
I32 Mod_No = 0;
I32 ret;
I32 EncCnt;
ret = APS_get_field_bus_encoder (BoardId, Bus_No, Mod_No, 0, & EncCnt);
If( ret != ERR_NoError )
{ // Error, show message.}
```

Below example is for ECAT-4XMO , ECAT-TRG4:

```
I32 BoardId = 0, Bus_No = 0, Mod_No = 0, ret = 0, EncCnt = 0;
// Get Encoder 0 count.
ret = APS_get_field_bus_encoder( BoardId, Bus_No, Mod_No, 0, &EncCnt );
If( ret != ERR_NoError )
{ // Error, show message. }
```

See also:

APS_set_field_bus_encoder()

APS_set_field_bus_encoder	Set encoder count.

Support Products: MNET-4XMO-C

Descriptions:

This function is used to set encoder count

Syntax:

C/C++:

I32 FNTYPE APS_set_field_bus_encoder(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 EncCh, I32 EncCnt);

Visual Basic:

APS_set_field_bus_encoder (ByVal Board_ID As Long, ByVal BUS_No As Long, ByVal MOD_No, ByVal EncCh As Long, ByVal EncCnt As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: Field bus number.(Port number) value: 0~1

I32 MOD_No: Module number.

For MNET field bus, the range of module number is 0 to 63.

I32 EncCh: Specified the encoder channel number. Zero base.

For MNET-4XMO-C, the range of EncCh is 0 to 4. (EncCh 0~3 is used for general comparator, and LCmpCh 4 is used for high speed comparator.)

I32 EncCnt: Encoder count.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 BoardId = 0;
I32 Bus_No = 1;
I32 Mod_No = 0;
I32 ret;

ret = APS_set_field_bus_encoder (BoardId, Bus_No, Mod_No, 0, 0);
If( ret != ERR_NoError )
{ // Error, show message.
}
```

See also:

APS_set_field_bus_encoder()

APS_get_field_bus_timer_counter	Get timer count

Support Products: ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to get the timer counter value.

Syntax:

C/C++:

I32 APS_get_field_bus_timer_counter(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 TmrCh, I32 *TmrCnt);

Visual Basic:

APS_get_field_bus_timer_counter (ByVal Board_ID As Integer, ByVal BUS_No As Integer, ByVal MOD_No As Integer, ByVal TmrCh As Integer, ByRef TmrCnt As Integer) As Integer

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (start from 0).

I32 TmrCh: The channel of timer. (only support index 0 now)

I32 *TmrCnt: Return timer counter value.

Return Values:

I32 error code. Refer to error code table.

Example:

```
I32 BoardId = 0, Bus_No = 0, Mod_No = 0, ret = 0, TmrCnt = 0;
// Get Timer count value.
ret = APS_get_field_bus_timer_counter( BoardId, Bus_No, Mod_No, &TmrCnt );
If( ret != ERR_NoError )
{ // Error, show message.}
```

See also:

APS_set_field_bus_timer_counter();

APS_set_field_bus_timer_counter	Set timer count

Support Products: ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to set timer count value. The timer is used to simulate for encoder, and be comparator source.

Syntax:

C/C++:

I32 APS_set_field_bus_timer_counter(I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 TmrCh, I32 TmrCnt);

Visual Basic:

APS_set_field_bus_timer_counter (ByVal Board_ID As Integer, ByVal BUS_No As Integer, ByVal MOD_No As Integer, ByVal TmrCh As Integer, ByVal TmrCnt As Integer) As Integer

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (start from 0).

I32 TmrCh: The channel of timer. (only support index 0 now)

I32 TmrCnt: Specify timer counter value.

Return Values:

I32 error code. Refer to error code table.

Example:

```
I32 BoardId = 0, Bus_No = 0, Mod_No = 0, ret = 0;
// Set timer counter.
ret = APS_set_field_bus_timer_counter( BoardId, Bus_No, Mod_No, 0 , 100);
If( ret != ERR_NoError )
{ // Error, show message.}
```

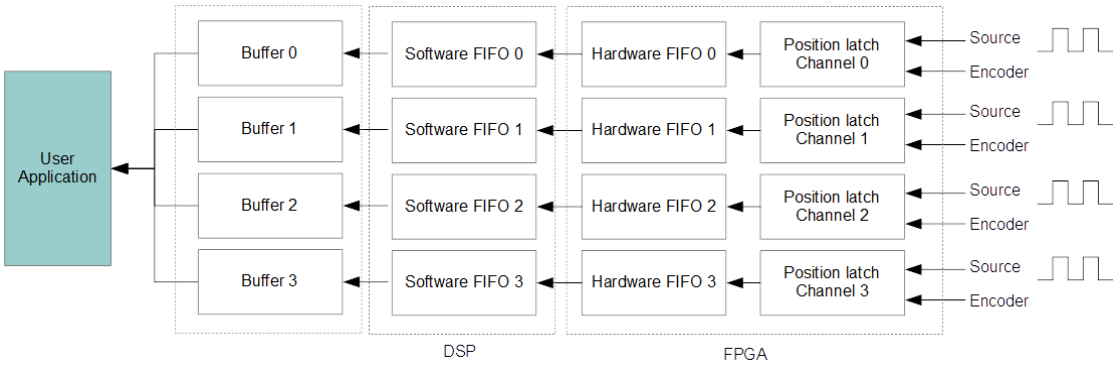
See also:

APS_get_field_bus_timer_counter()

25. Field bus position latch functions

Latch process Descriptions:

There are 4 position latch channels and each of them has hardware FIFO and software FIFO. User has to select the trigger source and encoder number for latch channel. Trigger source can be LTC, which is hardware digital input signals, or PWM pulse out module. The rising, falling or both edge trigger modes are also supported here. Before enable latch process, these setting must be set refer from “Latch parameter table”. Then set LTC_EN (0x13) to enable latch process



After enable position latch process, user can get software FIFO state and information from other functions. It can help user manage its program flow and calculation.

When the latch event occurs, the latched data will immediately be stored into hardware FIFO, which space is 255. This data will then be cyclically moved to software FIFO, which space is 5000.

Between user application and software FIFO, there are buffers to place point temporarily, so that user can get this data with software layer API.

If any latch parameter be revised, fieldbus disconnection or power off, operators need to set LTC_EN (0x13) disable latch process and enable it again.

Latch process enable ECAT-4XMO example:

I32 Board_ID = 0, BUS_NO = 0, ret = 0, MOD_NO = 1, FLtcCh = 0, Param_No = 0, Param_Val = 0,

FLtcCh = 0;

Param_No = LTC_EN;

Param_Val = 0;

ret = APS_set_field_bus_ltc_fifo_param(Board_ID,BUS_NO,MOD_NO,FLtcCh,Param_No,Param_Val);

```
ret = APS_reset_field_bus_ltc_fifo(Board_ID,BUS_NO,MOD_NO,FLtcCh);

Param_No = LTC_IPT;
Param_Val = 5;                  //      bit 0, bit 2 set means 2 ltc source
ret =
APS_set_field_bus_ltc_fifo_param(Board_ID,BUS_NO,MOD_NO,FLtcCh,Param_No,Param_V
al);
Param_No = LTC_ENC;
Param_Val = 0;
ret =
APS_set_field_bus_ltc_fifo_param(Board_ID,BUS_NO,MOD_NO,FLtcCh,Param_No,Param_V
al);
Param_No = LTC_LOGIC;
Param_Val = 1;
ret =
APS_set_field_bus_ltc_fifo_param(Board_ID,BUS_NO,MOD_NO,FLtcCh,Param_No,Param_V
al);
Param_No = LTC_EN;
Param_Val = 1;
ret =
APS_set_field_bus_ltc_fifo_param(Board_ID,BUS_NO,MOD_NO,FLtcCh,Param_No,Param_V
al);
If( ret != ERR_NoError )
{ // Error, show message.
}
```

Latch process enable ECAT-ECAT-TRG4 example:

```
I32 Board_ID = 0, BUS_NO = 0, ret = 0, MOD_NO = 1, FLtcCh = 0, Param_No = 0, Param_Val
= 0, Axis_ID = 5;
FLtcCh = 0;
Param_No = LTC_EN;
Param_Val = 0;
ret =
APS_set_field_bus_ltc_fifo_param(Board_ID,BUS_NO,MOD_NO,FLtcCh,Param_No,Param_V
al);

ret = APS_reset_field_bus_ltc_fifo(Board_ID,BUS_NO,MOD_NO,FLtcCh);
```

```
Param_No = LTC_IPT;
Param_Val = 5;                  //      bit 0, bit 2 set means 2 ltc source
ret =
APS_set_field_bus_ltc_fifo_param(Board_ID,BUS_NO,MOD_NO,FLtcCh,Param_No,Param_V
al);
Param_No = LTC_ENC;
Param_Val = 0;
ret =
APS_set_field_bus_ltc_fifo_param(Board_ID,BUS_NO,MOD_NO,FLtcCh,Param_No,Param_V
al);
Param_No = LTC_LOGIC;
Param_Val = 1;
ret =
APS_set_field_bus_ltc_fifo_param(Board_ID,BUS_NO,MOD_NO,FLtcCh,Param_No,Param_V
al);

Param_No = LTC_EXTENC_SRC;
Param_Val = Axis_ID;
ret =
APS_set_field_bus_ltc_fifo_param(Board_ID,BUS_NO,MOD_NO,FLtcCh,Param_No,Param_V
al);

Param_No = LTC_EN;
Param_Val = 1;
ret =
APS_set_field_bus_ltc_fifo_param(Board_ID,BUS_NO,MOD_NO,FLtcCh,Param_No,Param_V
al);
If( ret != ERR_NoError )
{ // Error, show message.
}
```

See also:

APS_set_field_bus_ltc_fifo_param(),APS_get_field_bus_ltc_fifo_param(),APS_reset_field_bus_ltc_fifo()

APS_get_field_bus_ltc_fifo_point	Get latch point array.

Support Products: ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to get latch point. Each latch point will include position in user coordinate and corresponding trigger source. The maximum latch point array size is 1.

Syntax:

C/C++:

I32 APS_get_field_bus_ltc_fifo_point (I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 FLtcCh, I32 *ArraySize, LATCH_POINT *LatchPoint);

Visual Basic:

APS_get_field_bus_ltc_fifo_point (ByVal Board_ID As Integer, ByVal BUS_No As Integer, ByVal MOD_No As Integer, ByVal FLtcCh As Integer, ByRef ArraySize As Integer, ByRef LatchPoint As LATCH_POINT) As Integer

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (start from 0).

I32 FLtcCh: Specified the latch channel number. Zero base. Range is from 0 to 3.

I32 *ArraySize: The size of latch point array, maximum value is 1.

```
typedef struct {
        F64 position;
        I32 ltcSrcInBit;
} LATCH_POINT;
```

Struct members:

F64 position: Latched position from specified encoder

I32 ltcSrcInBit: bit 0~3 means source from digital input signal 0~3; bit 8~11 means source from PWM output channel 0~3.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
//After enable latch process
// Thread 1 polling status,
I32 ret = 0, usage = 0;
I32 Board_ID = 0, BUS_NO = 0, ret = 0, MOD_NO = 1, FLtcCh = 0
I32 flag = 0;
While(1)
{
   ret = APS_get_field_bus_ltc_fifo_usage(Board_ID,BUS_NO,MOD_NO,FLtcCh, &usage)
   If(usage >= 1)
   {
      flag = 1;
   }
   else
   {
      flag = 0;
   }
}

//-----------------------------------------------------------------------------------------------
// Thread 2 judge flag to get point
I32 Board_ID = 0, BUS_NO = 0, ret = 0, MOD_NO = 1, FLtcCh = 0, size = 0;
LATCH_POINT pt;

if(flag == 1)
{
   ret = APS_get_field_bus_ltc_fifo_point(Board_ID,BUS_NO,MOD_NO,FLtcCh,&size,&pt);
   If( ret != ERR_NoError )
   { // Error, show message.
   }
}
```

See also:

APS_get_field_bus_ltc_fifo_usage(), APS_get_field_bus_ltc_fifo_status().

APS_set_field_bus_ltc_fifo_param	Set latch parameter value.

Support Products: ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to set latch parameter value into Latch parameter table.

Syntax:

C/C++:

I32 APS_set_field_bus_ltc_fifo_param (I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 FLtcCh, I32 Param_No, I32 Param_Val);

Visual Basic:

APS_set_field_bus_ltc_fifo_param (ByVal Board_ID As Integer, ByVal BUS_No As Integer, ByVal MOD_No As Integer, ByVal FLtcCh As Integer, ByVal Param_No As Integer, ByVal param_val As Integer) As Integer

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial ().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (Start from 0).

I32 FLtcCh: Specified the latch channel number. Zero base. Range is from 0 to 3.

I32 Param_No: Parameter number. Refer to latch parameter table.

I32 Param_Val: Parameter value. Refer to latch parameter table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Board_ID = 0, BUS_NO = 0, ret = 0, MOD_NO = 1, FLtcCh = 0;
I32 Param_Val = 0, Param_No = 0;
Param_No = LTC_IPT;
Param_Val = 5;                //      bit 0, bit 2 set
ret =
APS_set_field_bus_ltc_fifo_param(Board_ID,BUS_NO,MOD_NO,FLtcCh,Param_No,Param_Val);    // set input trigger source is which source
```

See also:

APS_get_field_bus_ltc_fifo_param()

APS_get_field_bus_ltc_fifo_param	Get latch parameter value.

Support Products: ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to get latch parameter value into Latch parameter table.

Syntax:

C/C++:

I32 APS_get_field_bus_ltc_fifo_param (I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 FLtcCh, I32 Param_No, I32 *Param_Val);

Visual Basic:

APS_set_field_bus_ltc_fifo_param (ByVal Board_ID As Integer, ByVal BUS_No As Integer, ByVal MOD_No As Integer, ByVal FLtcCh As Integer, ByVal Param_No As Integer, ByRef param_val As Integer) As Integer

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial ().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (Start from 0).

I32 FLtcCh: Specified the latch channel number. Zero base. Range is from 0 to 3.

I32 Param_No: Parameter number. Refer to latch parameter table.

I32 *Param_Val: Parameter value. Refer to latch parameter table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Board_ID = 0, BUS_NO = 0, ret = 0, MOD_NO = 1, FLtcCh = 0;
I32 Param_Val = 0, Param_No = 0;
Param_No = LTC_IPT;
Param_Val = 0;
ret=APS_get_field_bus_ltc_fifo_param(Board_ID,BUS_NO,MOD_NO,FLtcCh,Param_No,&Param_Val);
    // get input trigger source is which source
```

See also:

APS_set_field_bus_ltc_fifo_param()

APS_reset_field_bus_ltc_fifo	Reset latch queue and fifo.

Support Products: ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used before starting position latch to reset or clear buffer and both FIFO that has been introduced in position patch introduction. It is noticed that the position latch is also cleared simultaneously.

Syntax:

C/C++:

I32 APS_reset_field_bus_ltc_fifo (I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 FLtcCh);

Visual Basic:

APS_reset_field_bus_ltc_fifo (ByVal Board_ID As Integer, ByVal BUS_No As Integer, ByVal MOD_No As Integer, ByVal FLtcCh As Integer) As Integer

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial ().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (Start from 0).

I32 FLtcCh: Specified the latch channel number. Zero base. Range is from 0 to 3.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 Board_ID = 0, BUS_NO = 0, ret = 0, MOD_NO = 1, FLtcCh = 0;

ret = APS_reset_field_bus_ltc_fifo(Board_ID,BUS_NO,MOD_NO,FLtcCh); //Reset latch fifo

See also:

APS_get_field_bus_ltc_fifo_status().

APS_get_field_bus_ltc_fifo_usage	Get latch queue used space.

Support Products: ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to get the latch buffer and FIFO used space which is introduced in position patch introduction.

Syntax:

C/C++:

I32 APS_get_field_bus_ltc_fifo_usage (I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 FLtcCh , I32 *Usage);

Visual Basic:

APS_get_field_bus_ltc_fifo_usage (ByVal Board_ID As Integer, ByVal BUS_No As Integer, ByVal MOD_No As Integer, ByVal FLtcCh As Integer, ByRef Usage As Integer) As Integer

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (Start from 0).

I32 FLtcCh: Specified the latch channel number. Zero base. Range is from 0 to 3.

I32 *Usage: Software FIFO and hardware FIFO used space. The maximum value is 5255

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
//      After enable latch process
I32 Board_ID = 0, BUS_NO = 0, ret = 0, MOD_NO = 1, FLtcCh = 0, usage = 0;
ret = APS_get_field_bus_ltc_fifo_usage (Board_ID,BUS_NO,MOD_NO,FLtcCh, & usage);
If( ret != ERR_NoError )
{ // Error, show message.
}
```

See also:

APS_get_filed_bus_ltc_fifo_point(),APS_get_field_bus_ltc_fifo_free_space(),APS_get_field_bus_ltc_fifo_status().

APS_get_field_bus_ltc_fifo_free_space	Get latch queue free space.

Support Products: ECAT-4XMO , ECAT-TRG4

Descriptions:

This function is used to get the latch buffer and FIFO free space which is introduced in position patch introduction.

Syntax:

C/C++:

I32 APS_get_field_bus_ltc_fifo_free_space (I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 FLtcCh , I32 * FreeSpace);

Visual Basic:

APS_get_field_bus_ltc_fifo_free_space (ByVal Board_ID As Integer, ByVal BUS_No As Integer, ByVal MOD_No As Integer, ByVal FLtcCh As Integer, ByRef FreeSpace As Integer) As Integer

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial ().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (Start from 0).

I32 FLtcCh: Specified the latch channel number. Zero base. Range is from 0 to 3.

I32 * FreeSpace: Software FIFO and hardware FIFO free space. The value is 5255 – usage which is from APS_get_field_bus_ltc_fifo_usage ().

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
//      After enable latch process
I32 Board_ID = 0, BUS_NO = 0, ret = 0, MOD_NO = 1, FLtcCh = 0, freespace = 0;
ret = APS_get_field_bus_ltc_fifo_ free_space (Board_ID,BUS_NO,MOD_NO,FLtcCh, &
freespace);
If( ret != ERR_NoError )
{ // Error, show message.
}
```

See also:

APS_get_filed_bus_ltc_fifo_point(),APS_get_field_bus_ltc_fifo_usage(),APS_get_field_bus_ltc_fifo_status().

APS_get_field_bus_ltc_fifo_status	Get latch queue and fifo status.

Support Products: ECAT-4XMO , ECAT-TRG4

Descriptions:

This function can return the status of buffer, software and hardware FIFO, and usually used to prevent application fail (E.g., FIFO overflow).

Syntax :

C/C++:

I32 APS_get_field_bus_ltc_fifo_status (I32 Board_ID, I32 BUS_No, I32 MOD_No, I32 FLtcCh , I32 * Status);

Visual Basic:

APS_get_field_bus_ltc_fifo_status (ByVal Board_ID As Integer, ByVal BUS_No As Integer, ByVal MOD_No As Integer, ByVal FLtcCh As Integer, ByRef Status As Integer) As Integer

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial ().

I32 BUS_No: The index of field bus (only support index 0).

I32 MOD_No: The index of slave device. (Start from 0).

I32 FLtcCh: Specified the latch channel number. Zero base. Range is from 0 to 3.

I32 * Status: the bit define of status as follows:

In FIFO mode (LTC parameter : LTC_FIFO_MODE = 0)

Bit 0 = 0: Hardware FIFO is not empty; 1: Hardware FIFO is empty.

Bit 1 = 0: Hardware FIFO is not full; 1: Hardware FIFO is full.

Bit 2 = X, not used

Bit 3 = 0: Hardware FIFO is not overflow; 1: Hardware FIFO is overflow.

Bit 4 = 0: Software FIFO is not empty; 1: Software FIFO is empty.

Bit 5 = 0: Software FIFO is not full; 1: Software FIFO is full.

Bit 6 = 0: Software FIFO is not overflow; 1: Software FIFO is overflow

Bit 7 = 0: Buffer is not full; 1: Buffer is full.

In Single point mode (LTC parameter : LTC_FIFO_MODE = 1)

Bit 0~6 = X: not used

Bit 7 = 0: Buffer is not full; 1: Buffer is full.

Noted, once bit 3 or bit 6 be set 1, these status won't be clear to be 0 only if user manually reset FIFO by APS_reset_field_bus_ltc_fifo().

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 Board_ID = 0, BUS_NO = 0, ret = 0, MOD_NO = 1, FLtcCh = 0, status = 0;

ret = APS_get_field_bus_ltc_fifo_ status (Board_ID,BUS_NO,MOD_NO,FLtcCh, & status);

If(ret != ERR_NoError)

{ // Error, show message. }

See also:

APS_get_filed_bus_ltc_fifo_point(), APS_get_field_bus_ltc_fifo_free_space(),

APS_get_field_bus_ltc_fifo_usage(), APS_reset_field_bus_ltc_fifo().

26. Watch dog timer

APS_wdt_start	Start / Stop watch dog timer

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x

Descriptions:

This function is used to start / stop watch dog timer.

Syntax:

C/C++:

I32 FNTYPE APS_wdt_start(I32 Board_ID, I32 TimerNo, I32 TimeOut);

Visual Basic:

APS_wdt_start (ByVal Board_ID As Long, ByVal TimerNo As Long, ByVal TimeOut As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TimerNo: Specify a timer number.

In PCI-8254/58 or PCIe-833x, Timer No is 0.

I32 TimeOut:

Set 0 to diable watch dog timer.

Set a value by N(1 ~ 100) to enable watch dog timer.

TimeOut = N * 100 ms

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret = 0;
I32 boardId = 0;

//Enable watch dog timer, TimeOut is 2 sec.
ret = APS_wdt_start( boardId, 0, 20 );
```

See also:

APS_wdt_get_timeout_period	Get a timeout period of watch dog timer

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x

Descriptions:

This function is used to get a timeout period of watch dog timer. If timeout period is 0, watch dog timer is disabled. If timeout period is not 0, watch dog timer is enabled.

Syntax:

C/C++:

I32 FNTYPE APS_wdt_get_timeout_period(I32 Board_ID, I32 TimerNo, I32 *TimeOut);

Visual Basic:

APS_wdt_get_timeout_period(ByVal Board_ID As Long, ByVal TimerNo As Long, TimeOut As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TimerNo: Specify a timer number.

In PCI-8254/58 or PCIe-833x, Timer No is 0.

I32 TimeOut:

0 means that watch dog timer is disabled.

A value N(1 ~ 100) means that watch dog timer is enabled.

TimeOut = N * 100 ms

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret = 0;
I32 boardId = 0;
I32 timeOut = 0;

// Get a timeout period of watch dog timer
ret = APS_wdt_get_timeout_period ( boardId, 0, &timeOut );
```

See also:

APS_wdt_reset_counter	Reset counter of watch dog timer

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x

Descriptions:

This function is reset counter of watch dog timer to 0. The counter adds one count every DSP cycle.

When the watch dog timer is enabled, user could periodly reset the counter of watch dog timer to advoid trigger action event.

Syntax:

C/C++:

I32 FNTYPE APS_wdt_reset_counter(I32 Board_ID, I32 TimerNo);

Visual Basic:

APS_wdt_reset_counter (ByVal Board_ID As Long, ByVal TimerNo As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TimerNo: Specify a timer number.

In PCI-8254/58 or PCIe-833x, Timer No is 0.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret = 0;
I32 boardId = 0;

//Reset the counter of watch dog timer
ret = APS_wdt_reset_counter ( boardId, 0);
```

See also:

APS_wdt_get_counter	Get counter of watch dog timer

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x

Descriptions:

This function is used to get counter of watch dog timer. If enabled, the counter adds one count every DSP cycle. If disabled, the counter shows zero.

Syntax:

C/C++:

I32 FNTYPE APS_wdt_get_counter(I32 Board_ID, I32 TimerNo, I32 *Counter);

Visual Basic:

APS_wdt_get_counter (ByVal Board_ID As Long, ByVal TimerNo As Long, Counter As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TimerNo: Specify a timer number.

In PCI-8254/58 or PCIe-833x, Timer No is 0.

For PCI-8253/56 :

I32 Counter: If enabled, the counter adds one count every DSP cycle. If disabled, the counter shows zero.

For PCIe-833x :

I32 Counter: If enabled, the counter adds one count every system cycle time. If disabled, the counter shows zero.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret = 0;
I32 boardId = 0;
I32 Counter = 0;
// Get the counter of watch dog timer
ret = APS_wdt_get_counter ( boardId, 0, &Counter );
```

See also:

APS_wdt_set_action_event	Set action event of watch dog timer

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x

Descriptions:

This function is used to set action event of watch dog timer. If time out, action event will be triggered.

Syntax:

C/C++:

I32 FNTYPE APS_wdt_set_action_event(I32 Board_ID, I32 TimerNo, I32 EventByBit);

Visual Basic:

APS_wdt_set_action_event (ByVal Board_ID As Long, ByVal TimerNo As Long, ByVal EventByBit As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TimerNo: Specify a timer number.

In PCI-8254/58 or PCIe-833x, Timer No is 0.

For PCI-8253/56 :

I32 EventByBit: Set events

Bit0: Motor servo off

Bit1: Digital output off

Bit2: PWM off

For PCIe-833x :

I32 EventByBit: Set events

Bit0: All axes invoke EMG stop function

Bit1: All digital output of slaves are turned off

Bit2: All digital output of slaves are turned on

Bit3: All axes invoke servo off function

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret = 0;
I32 boardId = 0;
```

```
//Set action event. If time out, motor servo turns off.
ret = APS_wdt_set_action_event( boardId, 0, 1);
```

See also:

APS_wdt_get_action_event	Get action event of watch dog timer

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x

Descriptions:

This function is used to get action event of watch dog timer.

Syntax:

C/C++:

I32 FNTYPE APS_wdt_get_action_event(I32 Board_ID, I32 TimerNo, I32 *EventByBit);

Visual Basic:

APS_wdt_get_action_event (ByVal Board_ID As Long, ByVal TimerNo As Long, EventByBit As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 TimerNo: Specify a timer number.

In PCI-8254/58 or PCIe-833x, Timer No is 0.

For PCI-8253/56 :

I32 *EventByBit: Get events

Bit0: Motor servo off

Bit1: Digital output off

Bit2: PWM off

For PCIe-833x :

I32 *EventByBit: Get events

Bit0: All axes invoke EMG stop function

Bit1: All digital output of slaves are turned off

Bit2: All digital output of slaves are turned on

Bit3: All axes invoke servo off function

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret = 0;
I32 boardId = 0;
I32 EventByBit = 0;
```

```
// Get action event of watch dog timer
ret = APS_wdt_get_action_event( boardId, 0, &EventByBit );
```

See also:

27. VAO/PWM functions (Laser function)

APS_set_vao_param	Set parameter to VAO table

Support Products: PCI-8253/56, PCI-8254/58 / AMP-204/8C

Descriptions:

The VAO module is a laser control application. It provides analog output and PWM signal according to corresponding linear speed.

This function is used to set VAO related parameters. All definitions of VAO parameters are described in VAO parameter table.

You can also get VAO parameter setting using “APS_get_vao_param ()” function.

Syntax:

C/C++:

I32 FNTYPE APS_set_vao_param(I32 Board_ID, I32 Param_No, I32 Param_Val);

Visual Basic:

APS_set_vao_param (ByVal Board_ID As Long, ByVal Param_No As Long, ByVal Param_Val As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 Param_No: Parameter number. Refer to VAO parameter table.

I32 Param_Val: Parameter value. Refer to VAO parameter table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;

//Set output type of voltage mode to VAO table 0
ret = APS_set_vao_param(Board_ID, 0x00, 1);
```

See also:

APS_get_vao_param()

APS_get_vao_param	Get parameter of VAO table

Support Products: PCI-8253/56, PCI-8254/58 / AMP-204/8C

Descriptions:

The VAO module is a laser control application. It provides analog output and PWM signal according to corresponding linear speed.

This function is used to get VAO related parameters. All definitions of VAO parameters are described in VAO parameter table.

You can also set VAO parameter using “APS_set_vao_param()” function.

Syntax:

C/C++:

I32 FNTYPE APS_get_vao_param(I32 Board_ID, I32 Param_No, I32 *Param_Val);

Visual Basic:

APS_get_vao_param(ByVal Board_ID As Long, ByVal Param_No As Long, Param_Val As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 Param_No: Parameter number. Refer to VAO parameter table.

I32 Param_Val: Return parameter value. Refer to VAO parameter table.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Output_Type;

//Get output type of VAO table 0
ret = APS_set_vao_param(Board_ID, 0x00, &Output_Type );
```

See also:

APS_set_vao_param()

APS_set_vao_table	Set VAO table

Support Products: PCI-8253/56, PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to set a set of VAO table. Users can implement a VAO application according to this table. User configures related minimum velocity, velocity interval, total points, and mapping output value for laser application. Therefore, “Velocity to Power” mapping lookup table will be built.

Notice that the mapping output value will be checked according to VAO output type when executing APS_check_vao_param(). If the mapping output value is invalid, it returns “ERR_ParametersInvalid”.

For example, if output type was set to voltage mode, the mapping output voltage cant large than 10000 mV. The range of the mapping output value is described as below:

For PCI-8253/56:

Output type (0~3)	Output Range(範圍)
0: Voltage	0 ~ 10000 mv Unit: 1 mv
1: PWM mode	0 ~ 2000 (0.0% ~ 100%) Unit: 0.05%
2: PWM frequency mode with fixed width	1 ~ 25M Hz Unit: 1 Hz
3: PWM frequency mode with fixed duty cycle	1 ~ 25M Hz Unit: 1 Hz

For PCI-8254/58 / AMP-204/8C:

Output type (0~3)	Output Range
0: Voltage	0 ~ 10000 mv Unit: 1 mv
1: PWM mode	0 ~ 2000 (0.0% ~ 100%) Unit: 0.05%
2: PWM frequency mode with fixed width	3 ~ 50M Hz Unit: 1 Hz
3: PWM frequency mode with fixed	3 ~ 50M Hz

duty cycle	Unit: 1 Hz

Syntax:

C/C++:

I32 FNTYPE APS_set_vao_table(I32 Board_ID, I32 Table_No, I32 MinVelocity, I32 VelInterval, I32 TotalPoints, I32 *MappingDataArray);

Visual Basic:

APS_set_vao_table (ByVal Board_ID As Long , ByVal Table_No As Long, ByVal MinVelocity As Long, ByVal VelInterval As Long, ByVal TotalPoints As Long, MappingDataArray As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Table_No: VAO table number. Range is 0 ~ 7.

I32 MinVelocity: Minimum linear speed.

I32 VelInterval: Speed interval.

I32 TotalPoints : Total points. Range is 1 ~ 32.

I32 *MappingDataArray: Output data array.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Minimum_Velocity;
I32 Velocity_Interval;
I32 TotalPoints = 32;
I32 OutputVoltageData[32];

//Configure linear speed
//1st speed: 10000, 2nd speed: 20000, ….., 32th speed: 320000
Minimum_Velocity = 10000;
Velocity_Interval = 10000;
TotalPoints = 32;

//Configure mapping output voltage
OutputVoltageData[0] = 500; // 1st voltage: 500 mv
OutputVoltageData[1] = 600; // 2nd voltage: 600 mv
```

……..

OutputVoltageData[31] = 8600; // 32th voltage: 8600 mv

//Set mapping table of Vao table 0

Ret = APS_set_vao_table(Board_ID, 0, MinVelocity, VelInterval, TotalPoints,

OutputVoltageData);

See also:

APS_set_vao_param(); APS_get_vao_param(); APS_switch_vao_table( ); APS_start_vao()

APS_set_vao_param_ex	Set parameters via VAO structure

Support Products: PCI-8253/56，PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to set parameters via VAO structure. This is a extension of APS_set_vao_param() and APS_set_vao_table(). By invoking APS_set_vao_param_ex(), user could set all parameters via VAO structure at once. By invoking APS_set_vao_param(), user could set a specified parameter one by one.

This function is also used to set mapping table to replace APS_set_vao_table().User could configure related minimum velocity, velocity interval, total points, and mapping output value for laser application. Then, “Velocity to Power” mapping lookup table will be built.

Notice that both functions of APS_set_vao_param() and APS_set_vao_table() could be replaced by APS_set_vao_param_ex(). This is an option between them.

Syntax:

C/C++:

I32 FNTYPE APS_set_vao_param_ex(I32 Board_ID, I32 Table_No, VAO_DATA* VaoData);

Visual Basic:

APS_set_vao_param_ex (ByVal Board_ID As Integer, ByVal Table_No As Integer, ByRef VaoData As VAO_DATA) As Integer

Parameters:

I32 Board_ID: ID of the target controller. It’s retrieved by successful call to APS_initial().

I32 Table_No: VAO table number. Range is 0 ~ 7.

VAO_DATA *VaoData: Vao structure for setting all parameters.

```
Typedef struct _VAO_DATA
{
    //Parameters
    I32 outputType; //Output type, [0, 3]
    I32 inputType;   //Input type, [0, 1]
    I32 config;      //PWM configuration according to output type
    I32 inputSrc;    //Input source by axis, [0, 0xf]

    //Mapping table
    I32 minVel;      //Minimum linear speed, [ positive ]
```

```
    I32 VelInterval;  //Speed interval, [ positive ]
    I32 totalPoints;  //Total points, [1, 32]
    I32 mappingDataArr[32];     //mapping data array
}
VAO_DATA, *PVAO_DATA;
```

For PCI-8253/56:

VAO_DATA structure definition for setting VAO parameters

Varible name	Description	Value
outputType	Table output type (*1)	0: Voltage 1: PWM mode 2: PWM frequency mode with fixed width 3. PWM frequency mode with fixed duty cycle
inputType	Table input type	0: Feedback speed 1: Command speed
config	Configure PWM according to output type.	a. Mode 0 – Don't care b. Mode 1 – set a fixed frequency (1 ~ 25M Hz) c. Mode 2 – set a fixed Pulse Width (40 ~ 335544340 ns) d. Mode 3 – set a fixed duty cycle: N * 0.05 %. (N: 1 ~ 2000)
inputSrc	Specify axisID for VAO table. (linear speed on multi- axes) (*2)	Bit0: Axis 0 On Bit1: Axis 1 On Bit2: Axis 2 On Bit3: Axis 3 On

(*1) : PCI-8253 don't support voltage mode.

(*2): PCI-8253 supports 3 axes. Bit 0, bit 1 and bit 2 are available.

VAO_DATA structure definition for setting VAO mapping table

Varible name	Description	Value
minVel	Minimum linear speed	positive
velInterval	Speed interval	positive
totalPoints	Total points	1 ~ 32
mappingDataArr	mapping data array	Refer to following chart

The mapping data of VAO_DATA structure will be checked according to VAO output type. If the mapping data is invalid, it returns “ERR_ParametersInvalid”.
For example, if output type was set to voltage mode, the mapping output voltage cant large than 10000 mV. The range of mapping data is described as below:

Output type (0~3)	Output Range of mapping data
0: Voltage	0 ~ 10000 mv Unit: 1 mv
1: PWM mode	0 ~ 2000 (0.0% ~ 100%) Unit: 0.05%
2: PWM frequency mode with fixed width	1 ~ 25M Hz Unit: 1 Hz
3: PWM frequency mode with fixed duty cycle	1 ~ 25M Hz Unit: 1 Hz

For PCI-8254/58 / AMP-204/8C:

VAO_DATA structure definition for setting VAO parameters

Varible name	Description	Value
outputType	Table output type	0: Voltage 1: PWM mode 2: PWM frequency mode with fixed width 3. PWM frequency mode with fixed duty cycle
inputType	Table input type	0: Feedback speed 1: Command speed
config	Configure PWM according to output	a. Mode 0 – Don’t care b. Mode 1 – set a fixed

	type.	frequency (3 ~ 50M Hz) c. Mode 2 – set a fixed Pulse Width (20 ~ 335544300 ns) d. Mode 3 – set a fixed duty cycle: N * 0.05 %. (N: 1 ~ 2000)
inputSrc	Specify axisID for VAO table. (linear speed on multi- axes)	Bit0: Axis 0 On Bit1: Axis 1 On Bit2: Axis 2 On Bit3: Axis 3 On

VAO_DATA structure definition for setting VAO mapping table

Varible name	Description	Value
minVel	Minimum linear speed	positive
velInterval	Speed interval	positive
totalPoints	Total points	1 ~ 32
mappingDataArr	mapping data array	Refer to following chart

The mapping data of VAO_DATA structure will be checked according to VAO output type. If the mapping data is invalid, it returns “ERR_ParametersInvalid”.

For example, if output type was set to voltage mode, the mapping output voltage cant large than 10000 mV. The range of mapping data is described as below:

Output type (0~3)	Output Range of mapping data
0: Voltage(Reserved)	0 ~ 10000 mv Unit: 1 mv
1: PWM mode	0 ~ 2000 (0.0% ~ 100%) Unit: 0.05%
2: PWM frequency mode with fixed width	3 ~ 50M Hz Unit: 1 Hz
3: PWM frequency mode with fixed	3 ~ 50M Hz

duty cycle	Unit: 1 Hz

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
VAO_DATA VaoData;

VaoData. outputType = 1; // PWM mode
VaoData. inputType = 0; //Feedback speed
VaoData. Config = 1000; // set a fixed frequency of PWM mode, 1000hz
VaoData. inputSrc = 0x03; //axis 0 & axis 1
VaoData. minVel = 1000; // Minimum linear speed
VaoData. velInterval = 100; //Speed interval
VaoData. totalPoints = 2; //Two points
//10% ~ 15% of PWM mode
VaoData. mappingDataArr[0] = 200;
VaoData. mappingDataArr[1] = 300;

//Set parameters to table 0
ret = APS_set_vao_param_ex(Board_ID, 0, &VaoData);
```

See also:

APS_get_vao_param_ex(); APS_switch_vao_table(); APS_start_vao()

APS_get_vao_param_ex	Get parameters via VAO structure

Support Products: PCI-8253/56 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get parameters via VAO structure.

Refer to APS_set_vao_param_ex() for details.

Syntax:

C/C++:

I32 FNTYPE APS_get_vao_param_ex(I32 Board_ID, I32 Table_No, VAO_DATA* VaoData);

Visual Basic:

APS_get_vao_param_ex (ByVal Board_ID As Integer, ByVal Table_No As Integer, ByRef VaoData As VAO_DATA) As Integer

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Table_No: VAO table number. Range is 0 ~ 7.

VAO_DATA *VaoData: Vao structure for setting all parameters. Refer to APS_set_vao_param_ex() for more details.

```
Typedef struct _VAO_DATA
{
    //Parameters
    I32 outputType; //Output type, [0, 3]
    I32 inputType;   //Input type, [0, 1]
    I32 config;      //PWM configuration according to output type
    I32 inputSrc;    //Input source by axis, [0, 0xf]

    //Mapping table
    I32 minVel;            //Minimum linear speed, [ positive ]
    I32 VelInterval;       //Speed interval, [ positive ]
    I32 totalPoints;       //Total points, [1, 32]
    I32 *mappingDataArr; //mapping data array
}
VAO_DATA, *PVAO_DATA;
```

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
VAO_DATA VaoData;

//Get VAO param structure of VAO table 0
ret = APS_get_vao_param_ex(Board_ID, 0, &VaoData );
```

See also:

APS_set_vao_param_ex(); APS_start_vao()

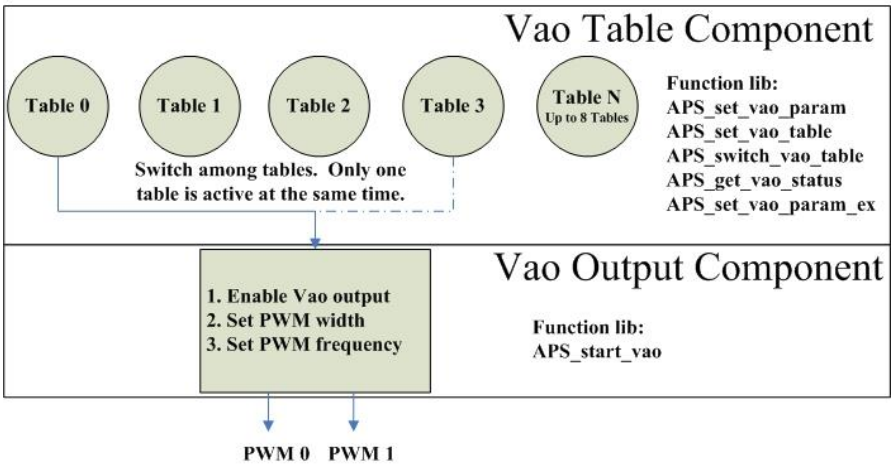
APS_switch_vao_table	Switch to specified VAO table

Support Products: PCI-8253/56

Descriptions:

The VAO module is a laser control application. It provides analog output and PWM signal according to corresponding linear speed.

This function is used to switch to specified VAO table as following figure. There are up to 8 tables to be configurated. User could switch to each table among them. Only one table is active at the same time.



Notice that if point table is running on this point, it will automatically switch to the specified table by setting "opt" variable. Refer to APS_set_point_table(). In the other way, user also could manually switch to specified table by APS_switch_vao_table().

Syntax:

C/C++:

I32 FNTYPE APS_switch_vao_table(I32 Board_ID, I32 Table_No);

Visual Basic:

APS_switch_vao_table(ByVal Board_ID As Long, ByVal Table_No As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Table_No: VAO table number.

0 ~ 7: Table number.

-1: Disable all tables.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 ret;

ret = APS_switch_vao_table(Board_ID, 0); //Swtich to table 0

See also:

APS_set_vao_param(); APS_get_vao_param(); APS_set_vao_table (); APS_start_vao()

APS_start_vao	Enable VAO output channel.

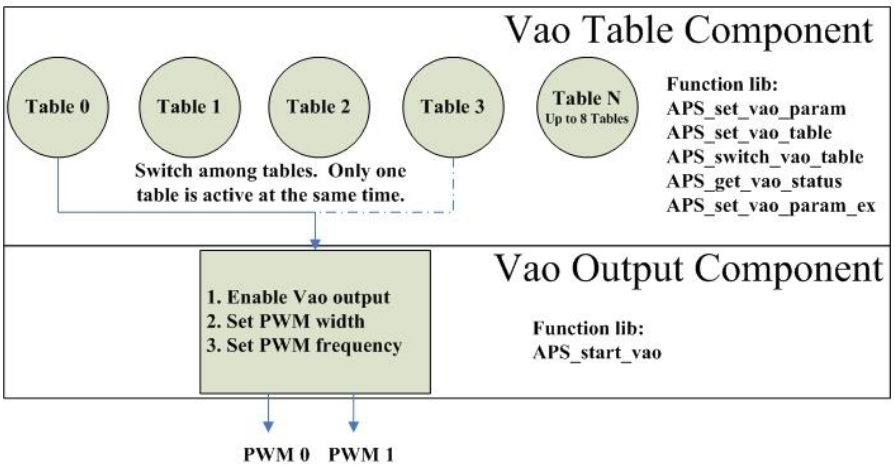
Support Products: PCI-8253/56，PCI-8254/58 / AMP-204/8C

Descriptions:

The VAO module is a laser control application. It provides analog output and PWM signal according to corresponding linear speed.

This function is used to enable VAO output channel as following figure. When VAO Output is enabling, analog voltage or PWM signal will output continuously according to corresponding linear speed.

User could also use APS_start_vao() to disable VAO output channel.



Syntax:

C/C++:

I32 FNTYPE APS_start_vao(I32 Board_ID, I32 Output_Ch, I32 Enable);

Visual Basic:

APS_start_vao (ByVal Board_ID As Long, ByVal Output_Ch As Long, ByVal Enable As Long) As Long

Parameters:

For PCI-8253/56:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Output_Ch: PWM or Analog channel. Range is 0 ~ 1.

0: PWM channel 0 or Aout 4

1: PWM channel 1 or Aout 5

I32 Enable: Enable specified channel to output PWM/Voltage.

0: Disable. 1: Enable

For PCI-8254/58 / AMP-204/8C:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Output_Ch: PWM or Analog channel. Range is 0 ~ 5.

0: PWM channel 0

1: PWM channel 1

2: PWM channel 2 (only 8258)

3: PWM channel 3 (only 8258)

4: Analog output 3 (Pulse mode)

5: Analog output 7 (Pulse mode)

I32 Enable: Enable specified channel to output PWM/Voltage.

0: Disable. 1: Enable

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;

ret = APS_start_vao( Board_ID, 0, 1 ); // Enable PWM channel 0 to output
```

See also:

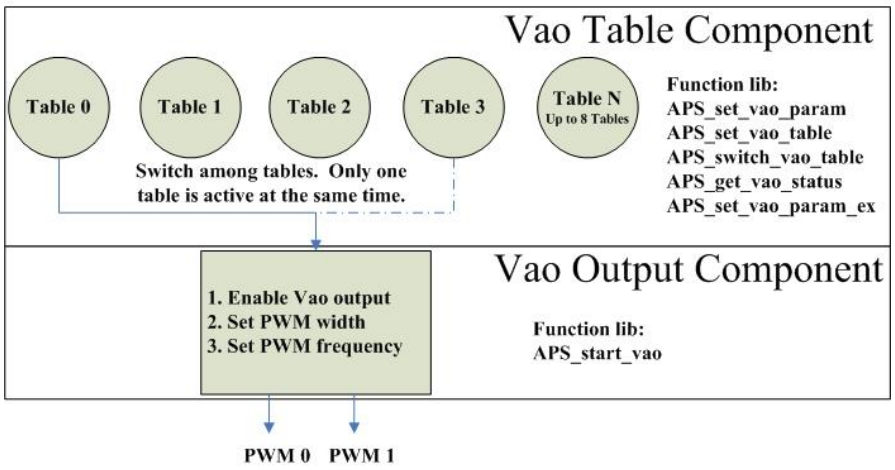
APS_set_vao_param();APS_get_vao_param(); APS_set_vao_table ();APS_switch_vao_table()

APS_get_vao_status	Get VAO status

Support Products: PCI-8253/56 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get VAO status. User could monitor which table is active and which PWM is enabling as following figure.



Syntax:

C/C++:

I32 FNTYPE APS_get_vao_status(I32 Board_ID, I32 *Status);

Visual Basic:

APS_get_vao_status (ByVal Board_ID As Long, Status As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 *Status: Get VAO status by bit.

Bit 0~7: Table 0~7 is active.

Bit 8~15: Reserved

Bit 16: PWM 0 or Analog 4 is enabling.

Bit 17: PWM 1 or Analog 5 is enabling.

Bit 18~: Reserved

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 ret;

```
I32 status;

//Get VAO status.
Ret = APS_get_vao_status(Board_ID, &status );
…….
```

See also:

APS_start_vao( ); APS_switch_vao_table( ); APS_start_vao

APS_check_vao_param	Check parameters setting of specified VAO table

Support Products: PCI-8253/56 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to check table parameters of specidied VAO table.

Syntax:

C/C++:

I32 FNTYPE APS_check_vao_param(I32 Board_ID, I32 Table_No, I32 *Status);

Visual Basic:

APS_check_vao_param (ByVal Board_ID As Long, ByVal Table_No As Long, Status As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 Table_No: VAO table number. Range is 0 ~ 7.

I32 *Status: The checking status of parameters. Refer to VAO parameter table definition.

0: No any parameters error

1: Parameter of table input type is out of range. (VAO_TABLE_ INPUT _TYPE)

2: Parameter of table output type is out of range. (VAO_TABLE_OUTPUT_TYPE)

3: Parameter of table input source is out of range. (VAO_TABLE_SRC)

4: Parameter of table pwm perationion is out of range.(VAO_TABLE_PWM_CONFIG)

5: Mapping table data is out of range. (Refer to APS_set_vao_table())

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 Sts;

// Check parameters setting of specified VAO table
// Check parameters setting of VAO table 0
ret = APS_check_vao_param (Board_ID, 0, & Sts );
…….
```

See also:

APS_set_vao_param(); APS_set_vao_table()

APS_set_pwm_on	Start to output PWM signal

Support Products: PCI-8253/56，PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to output PWM signal. It is applied to activate laser, trigger, etc. There are two PWM channels which are TRG1 and TRG2 on main connector

Note that the PWM output (TRG) is used by two function APIs, that are APS_set_pwm_on() and APS_start_vao() . Don't mix using them at the same time. Be sure that only one of them is enabled, specified PWM channel could rightly work.

Syntax:

C/C++:

I32 FNTYPE APS_set_pwm_on(I32 Board_ID, I32 PWM _Ch, I32 PWM_On);

Visual Basic:

APS_set_pwm_on(ByVal Board_ID As Long , ByVal PWM_Ch As Long , ByVal PWM_On As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PWM_Ch: PWM output channel (TRG) number. Zero based. Range is from 0 to 1.

I32 PWM_On: 0: PWM OFF, 1: PWM ON

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 PWM_Ch = 0; //TRG 0 is used
I32 Width = 2000 ns; //Pulse width is 2 us.
I32 Frequency = 10000 Hz;//pulse frequency is 10K Hz.

// Set pulse width to PWM channel 0
ret = APS_set_pwm_width( Board_ID, PWM_Ch, Width );
// Set pulse frequency to PWM channel 0
ret = APS_set_pwm_ frequency( Board_ID, PWM_Ch, Frequency);
// Output PWM signal to activate laser
```

```
ret = APS_set_pwm_on ( Board_ID, PWM_Ch, 1 );
.....
// Stop outputting PWM signal
Ret = APS_set_pwm_on ( Board_ID, PWM_Ch, 0 );
```

See also:

APS_set_pwm_width();APS_set_pwm_frequency();APS_get_pwm_width();
APS_get_pwm_frequency()

APS_set_pwm_width	Set pulse width to a PWM channel

Support Products: PCI-8253/56 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to set pulse width to specialized PWM channel.

For PCI-8253/56 :

Note that the range of pulse width is form 40 to 335544340. The unit is nano-second. The resolution of pulse width is 20 ns.

For PCI-8254/58 / AMP-204/8C:

Note that the range of pulse width is form 20 to 335544300. The unit is nano-second. The resolution of pulse width is 20 ns.

Syntax:

C/C++:

I32 FNTYPE APS_set_pwm_width(I32 Board_ID, I32 PWM _Ch, I32 Width);

Visual Basic:

I32 FNTYPE APS_set_pwm_width(ByVal Board_ID As Long , ByVal PWM_Ch As Long , ByVal Width As Long) As Long

Parameters:

For PCI-8253/56 :

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PWM_Ch: PWM output channel (TRG) number. Zero based. Range is from 0 to 1.

I32 Width: Pulse width. Unit: ns. Range is from 40 to 335544340.

For PCI-8254/58 / AMP-204/8C:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PWM_Ch: PWM output channel (TRG) number. Zero based. Range is from 0 to 1.

I32 Width: Pulse width. Unit: ns. Range is from 20 to 335544300.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 PWM_Ch = 0; // TRG 0 is used.
```

I32 Width = 2000 ns; //Pulse width is 2 us.

// Set pulse width to PWM channel 0
ret = APS_set_pwm_width(Board_ID, PWM_Ch, Width);

See also:

APS_set_pwm_on(); APS_set_pwm_frequency(); APS_get_pwm_width();
APS_get_pwm_frequency()

APS_set_pwm_frequency	Set pulse frequency to a PWM channel

Support Products: PCI-8253/56, PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to set pulse frequency to specialized PWM channel.

For PCI-8253/56:

Note that the range of pulse frequency is form 1 to 25000000. The unit is Hz.

It may have slightly offset between actual output frequency and the frequency you set.

The actual frequency is according to following formula:

$$\text{Frequency} = \frac{100{,}000{,}000}{2 \times N + 4}$$

N: 0 ~ 2147483647 (a positive 32 bit value)

For example, User could set the frequency = 10005 Hz to the card by this function.

In side the function, It get the *N* = 4988 from the formula and send it to the controller, and the actual frequency output from the PWM will be 10000 Hz (According above formula).

For PCI-8254/58 / AMP-204/8C:

Note that the range of pulse frequency is form 3 to 50,000,000. The unit is Hz.

It may have slightly offset between actual output frequency and the frequency you set.

The actual frequency is according to following formula:

$$\text{Frequency} = \frac{1{,}000{,}000{,}000}{20 \times N}$$

N: 0 ~ 16777215 (a positive 32 bit value)

For example, User could set the frequency = 10005 Hz to the card by this function.

In side the function, It get the *N* = 5000 from the formula and send it to the controller, and the actual frequency output from the PWM will be 10000 Hz (According above formula).

Syntax:

C/C++:

I32 FNTYPE APS_set_pwm_frequency(I32 Board_ID, I32 PWM _Ch, I32 Frequency);

Visual Basic:

APS_set_pwm_frequency(ByVal Board_ID As Long , ByVal PWM_Ch As Long , ByVal Frequency As Long) As Long

Parameters:

For PCI-8253/56:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PWM_Ch: PWM output channel (TRG) number. Zero based. Range is from 0 to 1.

I32 Frequency: Pulse frequency. Unit: Hz. Range is from 1 to 25000000.

For PCI-8254/58 / AMP-204/8C:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PWM_Ch: PWM output channel (TRG) number. Zero based. Range is from 0 to 1.

I32 Frequency: Pulse frequency. Unit: Hz. Range is from 3 to 50000000.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 PWM_Ch = 0; // TRG    0 is used.
I32 Frequency = 10000 Hz; //Pulse frequency is 10k Hz.

// Set pulse frequency to PWM channel 0
ret = APS_set_pwm_ frequency( Board_ID, PWM_Ch, Frequency);
```

See also:

APS_set_pwm_on(); APS_set_pwm_width(); APS_get_pwm_width();

APS_get_pwm_frequency()

APS_get_pwm_width	Get pulse width from a PWM channel

Support Products: PCI-8253/56 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get pulse width from specialized PWM channel.

Syntax:

C/C++:

I32 FNTYPE APS_get_pwm_width(I32 Board_ID, I32 PWM _Ch, I32 *Width);

Visual Basic:

I32 FNTYPE APS_get_pwm_width(ByVal Board_ID As Long , ByVal PWM_Ch As Long , Width As Long) As Long

Parameters:

For PCI-8253/56:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PWM_Ch: PWM output channel (TRG) number. Zero based. Range is from 0 to 1.

I32 Width: Pulse width. Unit: ns. Range is from 40 to 335544340.

For PCI-8254/58 / AMP-204/8C:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PWM_Ch: PWM output channel (TRG) number. Zero based. Range is from 0 to 1.

I32 Width: Pulse width. Unit: ns. Range is from 20 to 335544300.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 PWM_Ch = 0; // TRG 0 is used.
I32 Width;

// Get pulse width from PWM channel 0
ret = APS_get_pwm_width( Board_ID, PWM_Ch, &Width );
```

See also:

APS_set_pwm_on(); APS_set_pwm_width(); APS_set_pwm_frequency();

APS_get_pwm_frequency()

APS_get_pwm_frequency	Get pulse frequency from a PWM channel

Support Products: PCI-8253/56 , PCI-8254/58 / AMP-204/8C

Descriptions:

This function is used to get pulse frequency from specialized PWM channel.

Syntax:

C/C++:

I32 FNTYPE APS_get_pwm_frequency(I32 Board_ID, I32 PWM _Ch, I32 *Frequency);

Visual Basic:

APS_get_pwm_frequency(ByVal Board_ID As Long , ByVal PWM_Ch As Long , Frequency As Long) As Long

Parameters:

For PCI-8253/56:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PWM_Ch: PWM output channel (TRG) number. Zero based. Range is from 0 to 1.

I32 Frequency: Pulse frequency. Unit: Hz. Range is from 1 to 25000000.

For PCI-8254/58 / AMP-204/8C:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 PWM_Ch: PWM output channel (TRG) number. Zero based. Range is from 0 to 1.

I32 Frequency: Pulse frequency. Unit: Hz. Range is from 3 to 50000000.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
I32 PWM_Ch = 0; // TRG    0 is used.
I32 Frequency;

// Get pulse frequency from PWM channel 0
ret = APS_get_pwm_ frequency( Board_ID, PWM_Ch, &Frequency);
```

See also:

APS_set_pwm_on(); APS_set_pwm_frequency(); APS_set_pwm_width();

APS_get_pwm_width()

28. Circular limit functions

APS_set_circular_limit	Set configuration for circular limit

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

Set configuration for circular limit.

Syntax:

C/C++:

I32 FNTYPE APS_set_circular_limit (I32 Axis_A, I32 Axis_B, F64 Center_A, F64 Center_B, F64 Radius, I32 Stop_Mode, I32 Enable);

Visual Basic:

APS_set_circular_limit (ByVal Axis_ID_A As Integer, ByVal Axis_ID_B As Integer, ByVal Center_A As Double, ByVal Center_B As Double, ByVal Radius As Double, ByVal Stop_mode As Integer, ByVal Enable As Integer) As Integer

Parameters:

I32 Axis_A: axis ID 0~7

I32 Axis_B: axis ID 0~7

F64 Center_A: Center position of Axis_A.

F64 Center_B: Center position of Axis_B.

F64 Radius: Distance between circular limit boundary and center.

I32 Stop_Mode: Only Axis_A and Axis_B stop or all axes stop when circular limit is triggered.

I32 Enable: 0: Disable circular limit; 1: Enable circular limit

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
ret = APS_initial( &BoardID, 0 );

// Set servo on
ret = APS_set_servo_on( 0, 1 );
ret = APS_set_servo_on( 1, 1 );
```

```
// Set command
ret = APS_set_command(0, 0);
ret = APS_set_command(1, 0);

// Enable circular limit
ret = APS_set_circular_limit( 0, 1, 0, 0, 100, 1, 1 );

// Set interrupt
Int_No = APS_set_int_factor( 0, 0, 17, 1 ); //Enable the interrupt factor
APS_int_enable( 0, 1 ); //Enable the interrupt main switch
// Start move
ret = APS_relative_move( 0, 500, 1000 );

// Wait interrupt
returnCode = APS_wait_single_int( Int_No, -1 );
if( returnCode == ERR_NoError )
{ //Interrupt occurred
        APS_reset_int( Int_No );
}

// Disable circular limit
ret = APS_set_circular_limit( 0, 1, 0, 0, 100, 1, 0 );
```

See also:

APS_get_circular_limit()

APS_get_circular_limit	Get configuration for circular limit

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

Get configuration for circular limit.

Syntax:

C/C++:

I32 FNTYPE APS_get_circular_limit (I32 Axis_A, I32 Axis_B, F64 *Center_A, F64 *Center_B, F64 *Radius, I32 *Stop_Mode, I32 *Enable);

Visual Basic:

APS_get_circular_limit (ByVal Axis_ID_A As Integer, ByVal Axis_ID_B As Integer, ByRef Center_A As Double, ByRef Center_B As Double, ByRef Radius As Double, ByRef Stop_mode As Integer, ByRef Enable As Integer) As Integer

Parameters:

I32 Axis_A: axis ID 0~7

I32 Axis_B: axis ID 0~7

F64 *Center_A: Center position of Axis_A.

F64 *Center_B: Center position of Axis_B.

F64 *Radius: Distance between circular limit boundary and center.

I32 *Stop_Mode: Only Axis_A and Axis_B stop or all axes stop when circular limit is triggered.

I32 *Enable: 0: Disable circular limit; 1: Enable circular limit

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_set_circular_limit()

29. Simultaneous move functions

APS_set_absolute_simultaneous_move	Setup a absolute simultaneous move

Support Products: MNET-4XMO-(C), PCIe-8154/8158, PCI-C154(+)

Descriptions:

The function is used to setup an absolute simultaneous move. User could setup specified axes to implement simultaneous move. The parameters of Distance_Array and Max_Speed_Array are applied to specified axes. After that, user could invoke "APS_start_simultaneous_move()/APS_stop_simultaneous_move()" to start/stop simultaneous operation for starting/stopping specified axes at the same time.

Note: The axes specified in Axis_ID_Array must be of the same card/module.

Syntax:

C/C++:

I32 FNTYPE APS_set_absolute_simultaneous_move(I32 Dimension, I32 *Axis_ID_Array, I32 *Position_Array, I32 *Max_Speed_Array);

Visual Basic:

APS_set_absolute_simultaneous_move (ByVal Dimension As Long, Axis_ID_Array As Long, Position_Array As Long, Max_Speed_Array As Long) As Long

Parameters:

I32 Dimension: The dimension of simultaneous axes. (1~4 axes)

I32 *Axis_ID_Array: The axis ID array from 0 to 65535.

I32 Position_Array: Absolute position array. (unit: pulse)

I32 Max_Speed_Array: Maximum speed array. (unit: pulse/sec)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

//…Initial card

I32 Dimension = 4;

I32 Axis_ID_Array[4] = { 0, 1, 2, 3};

```
I32 Position_Array = {10000, 10000, 10000, 10000};
I32 Max_Speed_Array = {10000, 10000, 10000, 10000};
I32 Ret;

// Setup a absolute simultaneous move
Ret = APS_set_absolute_simultaneous_move ( Dimension, Axis_ID_Array, Position_Array,
Max_Speed_Array );
// Start a simultaneous move
Ret = APS_start_simultaneous_move( Axis_ID_Array[0] );
...
// Stop a simultaneous move
Ret = APS_stop_simultaneous_move( Axis_ID_Array[0] );
```

See also:

APS_set_relative_simultaneous_move();APS_start_simultaneous_move();
APS_stop_simultaneous_move()

APS_set_relative_simultaneous_move	Setup a relative simultaneous move

Support Products: MNET-4XMO-(C), PCIe-8154/8158, PCI-C154(+)

Descriptions:

The function is used to setup a relative simultaneous move. User could setup specified axes to implement simultaneous move. The parameters of Distance_Array and Max_Speed_Array are applied to specified axes.

After that, user could invoke

“APS_start_simultaneous_move()/APS_stop_simultaneous_move()” to start/stop a simultaneous operation for starting/stopping specified axes at the same time.

Note: The axes specified in Axis_ID_Array must be of the same card/module.

Syntax:

C/C++:

I32 FNTYPE APS_set_relative_simultaneous_move(I32 Dimension, I32 *Axis_ID_Array, I32 *Distance_Array, I32 *Max_Speed_Array);

Visual Basic:

APS_set_relative_simultaneous_move (ByVal Dimension As Long, Axis_ID_Array As Long, Distance_Array As Long, Max_Speed_Array As Long) As Long

Parameters:

I32 Dimension: The dimension of simultaneous axes. (1~4 axes)

I32 *Axis_ID_Array: The axis ID array from 0 to 65535.

I32 Distance_Array: Relative distance array. (unit: pulse)

I32 Max_Speed_Array: Maximum speed array. (unit: pulse/sec)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

//…Initial card

I32 Dimension = 4;

I32 Axis_ID_Array[4] = { 0, 1, 2, 3};

I32 Distance_Array = {10000, 10000, 10000, 10000};

I32 Max_Speed_Array = {10000, 10000, 10000, 10000};

```
I32 Ret;

// Setup a relative simultaneous move
Ret = APS_set_relative_simultaneous_move ( Dimension, Axis_ID_Array, Distance_Array,
Max_Speed_Array );
// Start a simultaneous move
Ret = APS_start_simultaneous_move( Axis_ID_Array[0] );
...
// Stop a simultaneous move
Ret = APS_stop_simultaneous_move( Axis_ID_Array[0] );
```

See also:

APS_set_absolute_simultaneous_move();APS_start_simultaneous_move();APS_stop_simultaneous_move()

APS_start_simultaneous_move	Begin a simultaneous move

Support Products: MNET-4XMO-(C), PCIe-8154/8158, PCI-C154(+)

Descriptions:

The function is used to start a simultaneous operation for starting specified axes at the same time.

Syntax:

C/C++

I32 FNTYPE APS_start_simultaneous_move (I32 Axis_ID);

Visual Basic:

APS_start_simultaneous_move (ByVal Axis_ID As Long) As Long

Parameters:

I32 Axis_ID: Specify first axis of simultaneous axes. The Axis ID is from 0 to 65535.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
//…Initial card
I32 Dimension = 4;
I32 Axis_ID_Array[4] = { 0, 1, 2, 3};
I32 Distance_Array = {10000, 10000, 10000, 10000};
I32 Max_Speed_Array = {10000, 10000, 10000, 10000};
I32 Ret;

// Setup a relative simultaneous move
Ret = APS_set_relative_simultaneous_move ( Dimension, Axis_ID_Array, Distance_Array,
Max_Speed_Array );
// Start a simultaneous move
Ret = APS_start_simultaneous_move( Axis_ID_Array[0] );
...
// Stop a simultaneous move
Ret = APS_stop_simultaneous_move( Axis_ID_Array[0] );
```

See also:

APS_set_absolute_simultaneous_move();APS_set_relative_simultaneous_move();
APS_stop_simultaneous_move()

APS_stop_simultaneous_move	Stop a simultaneous move

Support Products: MNET-4XMO-(C), PCIe-8154/8158, PCI-C154(+)

Descriptions:

The function is used to stop a simultaneous operation for stopping specified axes at the same time.

Syntax:

C/C++

I32 FNTYPE APS_stop_simultaneous_move (I32 Axis_ID);

Visual Basic:

APS_stop_simultaneous_move (ByVal Axis_ID As Long) As Long

Parameters:

I32 Axis_ID: Specify first axis of simultaneous axes. The Axis ID is from 0 to 65535.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

//…Initial card

I32 Dimension = 4;

I32 Axis_ID_Array[4] = { 0, 1, 2, 3};

I32 Distance_Array = {10000, 10000, 10000, 10000};

I32 Max_Speed_Array = {10000, 10000, 10000, 10000};

I32 Ret;

// Setup a relative simultaneous move

Ret = APS_set_relative_simultaneous_move (Dimension, Axis_ID_Array, Distance_Array, Max_Speed_Array);

// Start a simultaneous move

Ret = APS_start_simultaneous_move(Axis_ID_Array[0]);

...

// Stop a simultaneous move

Ret = APS_stop_simultaneous_move(Axis_ID_Array[0]);

See also:

APS_set_absolute_simultaneous_move();APS_set_relative_simultaneous_move();
APS_start_simultaneous_move()

30. Single latch functions

APS_manual_latch2	Manual latch for a axis

Support Products:MNET-4XMO-(C), MNET-1XMO, PCI(e)-8154/8158, PCI-8102/PCI-C154(+)

Descriptions:

The function is used to produce a manual latch signal.

Syntax:

C/C++:

I32 FNTYPE APS_manual_latch2(I32 Axis_ID);

Visual Basic:

APS_manual_latch2 (ByVal Axis_ID As Long) As Long

Parameters:

I32 Axis_ID: The axis ID array from 0 to 65535.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 axisID = 0;
I32 ret = 0;
I32 LatchData = 0;

ret = APS_manual_latch2( axisID );
//latch data is command counter
ret = APS_get_latch_data2( axisID, 0, &LatchData );
```

See also:

APS_get_latch_data2()

APS_get_latch_data2	Get latch data for a axis

Support Products: MNET-4XMO(C), MNET-1XMO, PCI(e)-8154/8158, PCI-8102/PCI-C154(+)

Descriptions:

The function is used to get latch data. There are two methods to latch data. One is input signal from physical latch pin. The other is internal latch signal from manual latch. There are four kinds of data including that user could latch. They are:

1. Command counter (Command position)
2. Feedback counter (Feedback position)
3. Error counter (Error position) / current speed
4. General-purpose counter

Syntax:

C/C++:

I32 FNTYPE APS_get_latch_data2(I32 Axis_ID, I32 LatchNum, I32 *LatchData);

Visual Basic:

APS_get_latch_data2 (ByVal Axis_ID As Long) As Long

Parameters:

I32 Axis_ID: The axis ID array from 0 to 65535.

I32 LatchNum:

0: Command counter

1: Feedback counter

2: Error counter / Current speed (via axis parameter 22Dh PRA_LATCH_DATA_SPD)

3: General-purpose counter

I32 *LatchData: Latch data

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 axisID = 0;
I32 ret = 0;
I32 LatchData = 0;

ret = APS_manual_latch2( axisID );
//latch data is command counter
```

```
ret = APS_get_latch_data2( axisID, 0, &LatchData );
```

See also:

APS_manual_latch2()

31. Multi-latch functions

APS_set_ltc_counter	Set encoder counter value

Support Products: PCI-C154(+)

Descriptions:

The function is used to set encoder counter value.

Syntax:

C/C++:

I32 FNTYPE APS_set_ltc_counter (I32 Board_ID, I32 LtcCh, I32 CntValue);

Visual Basic:

APS_set_ltc_counter (ByVal Board_ID As Long, ByVal LtcCh As Long, ByVal CntValue As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 LtcCh: The specified channel number.

I32 CntValue: The encoder (counter) value.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 Board_ID = 0;

I32 ret = 0;

//Set latch counter 0 to 100

ret = APS_set_ltc_counter (Board_ID, 0, 100);

See also:

APS_get_ltc_counter ()

APS_get_ltc_counter	Get encoder counter value

Support Products: PCI-C154(+)

Descriptions:

The function is used to get encoder counter value.

Syntax:

C/C++:

I32 FNTYPE APS_get_latch_counter (I32 Board_ID, I32 LtcCh, I32 *CntValue);

Visual Basic:

APS_get_latch_counter (ByVal Board_ID As Long, ByVal LtcCh As Long, CntValue As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 LtcCh: The specified channel number.

I32 CntValue: The encoder (counter) value.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Board_ID = 0;
I32 ret = 0;
I32 CntValue= 0;

//Get counter value from latch counter 0
ret = APS_get_ltc_counter ( Board_ID, 0, &CntValue );
```

See also:

APS_set_ltc_counter ()

APS_set_ltc_fifo_param	Set latch parameter

Support Products: PCI-C154(+), PCI-8254/58 / AMP-204/8C, AMP-104C

Descriptions:

This function is used to set all kinds of latch parameter. Please refer to the latch parameter table for the definition and detail descriptions.

Syntax:

C/C++

I32 FNTYPE APS_set_ltc_fifo_param(I32 Board_ID, I32 FLtcCh, I32 Param_No, I32 Param_Val);

Visual Basic:

APS_set_ltc_fifo_param (ByVal Board_ID As Long, ByVal FLtcCh As Long, ByVal Param_No As Long, ByVal Param_Val As Long) As Long

Parameters:

For PCI-C154(+)/AMP-104C:

I32 Board_ID: The Board's ID from 0 to 31.

I32 FLtcCh: The specified latch channel.

I32 Param_No: Latch parameter number. Please refer the latch parameter table for definition.

I32 Param_Val: Latch parameter value. Refer to the latch parameter table for detail.

For PCI-8254/58 / AMP-204/8C:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 FLtcCh: The range of latch channel is 0~3

I32 Param_No: Latch parameter number; Please refer the latch parameter table for definition.

I32 Param_Val: Latch parameter value. Refer to the latch parameter table for detail.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Board_ID = 0;
I32 ret = 0;

//Set encoder input mode(0x00) to 4xAB-Phase(4) in channel 0
ret = APS_set_ltc_fifo_param ( Board_ID, 0, 0, 4 );
```

See also:

APS_get_ltc_fifo_param ()

APS_get_ltc_fifo_param	Get latch parameter

Support Products: PCI-C154(+), PCI-8254/58 / AMP-204/8C, AMP-104C

Descriptions:

This function is used to get all kinds of latch parameter. Please refer to the latch parameter table for the definition and detail descriptions.

Syntax:

C/C++:

I32 FNTYPE APS_get_ltc_fifo_param(I32 Board_ID, I32 FLtcCh, I32 Param_No, I32 *Param_Val);

Visual Basic:

APS_get_ltc_fifo_param (ByVal Board_ID As Long, ByVal FLtcCh As Long, ByVal Param_No As Long, Param_Val As Long) As Long

Parameters:

For PCI-C154(+)/ AMP-104C:

I32 Board_ID: The Board's ID from 0 to 31.

I32 FLtcCh: The specified latch channel.

I32 Param_No: Latch parameter number. Please refer the latch parameter table for definition.

I32 *Param_Val: Latch parameter value. Refer to the latch parameter table for detail.

For PCI-8254/58 / AMP-204/8C:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 FLtcCh: The range of latch channel is 0~3

I32 Param_No: Latch parameter number; Please refer the latch parameter table for definition.

I32 Param_Val: Latch parameter value. Refer to the latch parameter table for detail.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 Board_ID = 0;

I32 ret = 0;

I32 Param_Val = 0;

```
//Get encoder input mode(0x00) in channel 0
ret = APS_get_ltc_fifo_param ( Board_ID, 0, 0, &Param_Val );
```

See also:

APS_set_ltc_fifo_param ()

APS_manual_latch	Latch data manually and synchronously.

Support Products: PCI-C154(+)

Descriptions:

This function is used to latch data manually. It is designed to latch one or more channels synchronously.

Syntax:

C/C++:

I32 FNTYPE APS_manual_latch (I32 Board_ID, I32 LtcChInBit);

Visual Basic:

APS_manual_latch (ByVal Board_ID As Long, ByVal LtcChInBit As Long) As Long

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 LtcChInBit: The specified latch channel by bit.

Bit 0: Channel 0, Bit 1: Channel 1, …, Bit 8: Channel 8

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;
//Latch data of channel 0 ~ 3 synchronously.
ret = APS_manual_latch( Board_ID, 0xf );
```

See also:

APS_enable_ltc_fifo	Enable/Disable ltc fifo/ Enable position latch process

Support Products: PCI-C154(+), PCI-8254/58 / AMP-204/8C

Descriptions:

For PCI-C154(+) , this function is used to enable/disable latch fifo. Once it is disabled, Latch pin will ignore any latch signal. Users must enable this function before using any latch relative functions and disable this function when users do not use latch anymore. The latch fifo can store up to 258 (PCI-C154+)pieces of latch data, user could monitor fifo status and get latch data.

For PCI-8254/58 / AMP-204/8C , this function is used to enable position latch process. Figure 1 shows the position latch module architecture. There are four position latch channels and each of them has dedicated FIFO and queue. User has to specify the trigger source and encoder no. for position latch channel. The trigger source can be digital input signals or PWM pulse out. The rising, falling or both edge trigger modes are also supported here. These setting can be configured by the latch parameter table.

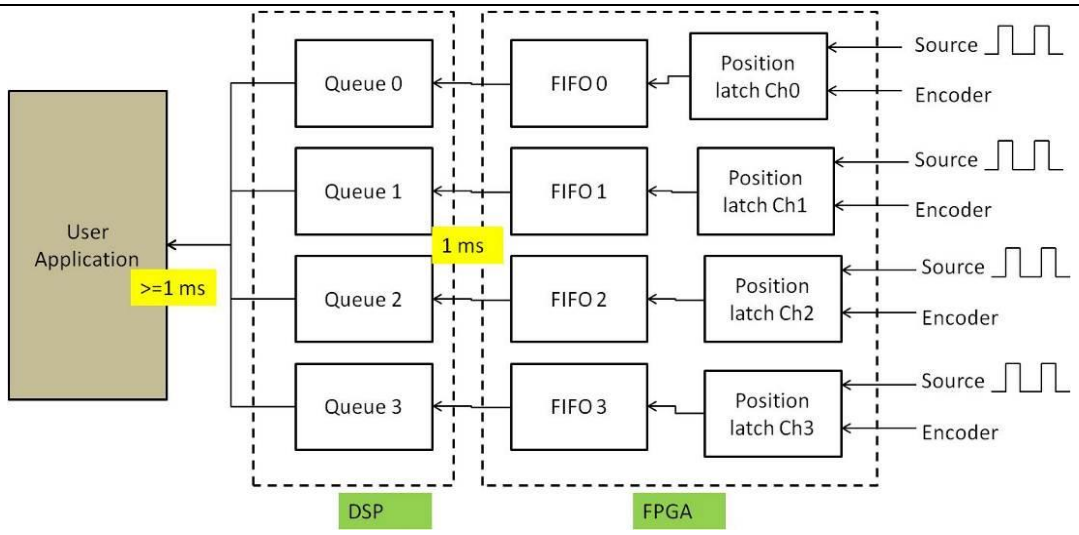


Figure 1 Position latch architecture

Syntax:

C/C++:

I32 FNTYPE APS_enable_ltc_fifo(I32 Board_ID, I32 FLtcCh, I32 Enable);

Visual Basic:

APS_enable_ltc_fifo (ByVal Board_ID As Long, ByVal FLtcCh As Long, ByVal Enable As Long) As Long

Parameters:

For PCI-C154(+):

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 FLtcCh: The specified latch channel from 0 to 3.

I32 Enable: Enable/Disable latch fifo.

0: Disable. 1: Enable

For PCI-8254/58 / AMP-204/8C:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 FLtcCh: The range of latch channel is 0~3

I32 Enable: 0: Disable; 1: Enable

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;

//Enable latch fifo channel 0
ret = APS_enable_ltc_fifo( BoardID, 0, 0 ); // Disable position latch

ret = APS_reset_ltc_fifo( BoardID, 0 ); // Reset position latch queue

ret = APS_set_ltc_fifo_param(BoardID, 0, LTC_IPT, 0xFFF ); // Set input source
ret = APS_set_ltc_fifo_param(BoardID, 0, LTC_ENC, 0 ); // Set EncoderNo
ret = APS_set_ltc_fifo_param(BoardID, 0, LTC_LOGIC, 0 ); // Set Logic

ret = APS_enable_ltc_fifo( BoardID, 0, 1 ); // Start position latch
```

See also:

APS_set_ltc_fifo_param(); APS_get_ltc_fifo_param(); APS_reset_ltc_fifo()

APS_reset_ltc_fifo	Reset ltc fifo/ Reset latch queue and fifo

Support Products: PCI-C154(+), PCI-8254/58 / AMP-204/8C, AMP-104C

Descriptions:

For PCI-C154(+)/AMP-104C , this function is used to reset latch fifo. Latch fifo will clear all data, and the status of fifo is in empty status.

For PCI-8254/58 / AMP-204/8C , this function is used before starting position latch to reset or clear both Queue and FIFO that has been introduced in APS_enable_ltc_fifo. It is noticed that the position latch is also cleared simutaneously.

Syntax:

C/C++:

I32 FNTYPE APS_reset_ltc_fifo(I32 Board_ID, I32 FLtcCh);

Visual Basic:

APS_reset_ltc_fifo (ByVal Board_ID As Long, ByVal FLtcCh As Long) As Long

Parameters:

For PCI-C154(+):

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 FLtcCh: The specified latch channel.

For PCI-8254/58 / AMP-204/8C:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 FLtcCh: The range of latch channel is 0~3

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret;

//Reset latch fifo channel 0
ret = APS_reset_ltc_fifo ( Board_ID, 0 );
```

See also:

APS_get_ltc_fifo_data	Get one latch data from fifo

Support Products: PCI-C154(+)

Descriptions:

This function is used to get one latch data from fifo.

Syntax:

C/C++:

I32 FNTYPE APS_get_ltc_fifo_data(I32 Board_ID, I32 FLtcCh, I32 Data);

Visual Basic:

APS_get_ltc_fifo_data (ByVal Board_ID As Long, ByVal FLtcCh As Long, Data As Long) As Long

Parameters:

I32 Board_ID: The Board's ID from 0 to 31.

I32 FLtcCh: The specified latch channel.

I32 Data: Get latch data stored in fifo.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Board_ID = 0;
I32 ret = 0;
I32 data= 0;

//Get data from latch fifo channel 0
ret = APS_get_ltc_fifo_data ( Board_ID, 0, & data );
```

See also:

APS_get_ltc_fifo_usage	Get usage of latch fifo

Support Products: PCI-C154(+), PCI-8254/58 / AMP-204/8C, AMP-104C

Descriptions:

For PCI-C154(+) ,this function is used to get usage of latch fifo. The usage means how many fifo spaces is already used. The range of fifo usage is from 0 to 258 (PCI-C154+).

The range of fifo usage is from 0 to 255 (AMP-104C).

For PCI-8254/58 / AMP-204/8C,this function is used to get the latch queue used space which is introduced in APS_enable_ltc_fifo.

Syntax:

C/C++:

I32 FNTYPE APS_get_ltc_fifo_usage(I32 Board_ID, I32 FLtcCh, I32 Usage);

Visual Basic:

APS_get_ltc_fifo_usage (ByVal Board_ID As Long, ByVal FLtcCh As Long, Usage As Long) As Long

Parameters:

For PCI-C154(+)/AMP-104C :

I32 Board_ID: The Board's ID from 0 to 31.

I32 FLtcCh: The specified latch channel from 0 ~ 3.

I32 Usage: Get usage of latch fifo.

For PCI-8254/58 / AMP-204/8C:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 FLtcCh: The range of latch channel is 0~3

I32 *Usage: Queue used space

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 Board_ID = 0;

I32 ret = 0;

I32 usage = 0;

//Get usage of latch fifo channel 0

```
ret = APS_get_ltc_fifo_usage ( Board_ID, 0, &usage );
```

See also:

APS_get_ltc_fifo_free_space	Get free space of latch fifo

Support Products: PCI-C154(+), PCI-8254/58 / AMP-204/8C, AMP-104C

Descriptions:

For PCI-C154(+)/AMP-104C , this function is used to get free space of latch fifo. The range of free space is from 0 to 258(PCI-C154+). The free space means remaining space to store data in fifo.

For PCI-8254/58 / AMP-204/8C , this function is used to get latch queue used space which is introduced in APS_enable_ltc_fifo.

Syntax:

C/C++:

I32 FNTYPE APS_get_ltc_fifo_free_space(I32 Board_ID, I32 FLtcCh, I32 *FreeSpace);

Visual Basic:

APS_get_ltc_fifo_free_space (ByVal Board_ID As Long, ByVal FLtcCh As Long, FreeSpace As Long) As Long

Parameters:

For PCI-C154(+) :

I32 Board_ID: The Board's ID from 0 to 31.

I32 FLtcCh: The specified latch channel.

I32 FreeSpace: Get free space of latch fifo.

For PCI(e)-8154/58:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 FLtcCh: The range of latch channel is 0~3

I32 * FreeSpace: Queue free space

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 Board_ID = 0;

I32 ret = 0;

I32 space = 0;

```
//Get free space of latch fifo channel 0
ret = APS_get_ltc_fifo_free_space ( Board_ID, 0, &space );
```

See also:

APS_get_ltc_fifo_status	Get fifo status/ Get latch queue and fifo status

Support Products: PCI-C154(+), PCI-8254/58 / AMP-204/8C, AMP-104C

Descriptions:

This function is used to get latch queue and fifo status. User could monitor fifo status including empty, full, level and overflow status.

Syntax:

C/C++:

I32 FNTYPE APS_get_ltc_fifo_status(I32 Board_ID, I32 FLtcCh, I32 Status);

Visual Basic:

APS_get_ltc_fifo_status (ByVal Board_ID As Long, ByVal FLtcCh As Long, Status As Long) As Long

Parameters:

For PCI-C154(+):

I32 Board_ID: The Board's ID from 0 to 31.

I32 FLtcCh: The specified latch channel from 0 to 3.

I32 Status: Get status of latch fifo.

Bit0➔ 0: Not empty, 1: Empty (PCI-C154(+))

Bit1➔ 0: Not full, 1: Full (PCI-C154(+))

Bit2➔ 0: Under high level, 1: Above high level (PCI-C154(+))

For AMP-104C:

I32 Board_ID: The Board's ID from 0 to 31.

I32 FLtcCh: The specified latch channel from 0 to 3.

I32 Status: Get status of latch fifo.

Bit0➔ 0: FIFO is not empty, 1: FIFO is Empty (AMP-104C)

Bit1➔ 0: FIFO is not full, 1: FIFO is full (AMP-104C)

Bit2➔ 0: FIFO is not overflow, 1: FIFO is Overflow (AMP-104C)

For PCI-8254/58 / AMP-204/8C:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 FLtcCh: The range of latch channel is 0~3

I32 * Status: the bit define of status is

Bit0 = 0: FIFO is not empty, 1: FIFO is Empty (cyclic update)

Bit1 = 0: FIFO is not full, 1: FIFO is full (cyclic update)

Bit2 = X

Bit3 = 0: FIFO is not overflow, 1: FIFO is Overflow (clear by reset Queue and FIFO)

Bit4 = 0: Queue is not empty, 1: Queue is empty (cyclic update)

Bit5 = 0: Queue is not full, 1: Queue is full (cyclic update)

Bit6 = 0: Queue is not overflow, 1: Queue is overflow (clear by reset Queue and FIFO)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Board_ID = 0;
I32 ret = 0;
I32 status = 0;

//Get status of latch fifo channel 0
ret = APS_get_ltc_fifo_free_status ( Board_ID, 0, &status );
```

See also:

APS_get_ltc_fifo_point	Get latch point array

Support Products: PCI-8254/58 / AMP-204/8C, AMP-104C

Descriptions:

This function is used to get latch point array. Each latch point will include position in user coordinate and corresponding trigger source. The maximum latch point array size is 16.

Syntax:

C/C++:

I32 FNTYPE APS_get_ltc_fifo_point(I32 Board_ID, I32 FLtcCh, I32 *ArraySize, LATCH_POINT *LatchPoint)

APS_get_ltc_fifo_point (ByVal Board_ID As Integer, ByVal FLtcCh As Integer, ByRef ArraySize As Integer, ByRef LatchPoint As LATCH_POINT) As Integer

Parameters:

I32 Board_ID: ID of the target controller. It's retrieved by successful call to APS_initial().

I32 FLtcCh: The range of latch channel is 0~3

I32 * ArraySize: The size of latch point array; the maximum value of ArraySize is less than 255.

LATCH_POINT * LatchPoint: Latched point array. The define of LATCH_POINT is shown below.

For PCI-8254/58 / AMP-204/8C:

```
typedef struct
{
    F64 position;    // Latched position
    I32 ltcSrcInBit; // Latch source: bit 0~7: DI; bit 8~11: trigger channel
} LATCH_POINT;
```

For AMP-104C:

```
typedef struct
{
    F64 position;    // Latched position
    I32 ltcSrcInBit; // Latch source: bit 0~3: SISC isolated DI; bit 4~7: TTL DI
} LATCH_POINT;
```

Members:

For PCI-8254/58 / AMP-204/8C:

position

Latched position

ltcSrcInBit

(1) bit 0~7: Digital input signal

(2) bit 8~11: trigger channel

For AMP-104C:

position

Latched position

ltcSrcInBit

(1) bit 0~3: Digital input signal

(2) bit 4~7: trigger channel

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 I32 Board_ID =0;
I32 FLtcCh: The specified latch channel from 0 to 3.
LATCH_POINT LatchPoint[255];// Maximum latch FIFO size are 255.
I32 ArraySize = 0;
APS_get_ltc_fifo_point( BoardID, 0, &ArraySize, LatchPoint );



if(ArraySize){
                for(i=0; i<ArraySize; i++)
                {
                        printf("%f\n", LatchPoint[i].position);
                }
        }
```

See also:

APS_set_ltc_fifo_param(); APS_get_ltc_fifo_param(); APS_enable_ltc_fifo()

32. Ring counter functions

APS_set_ring_counter	Enable ring counter function

Support Products: PCI-8154/58

Descriptions:

This function is used to set ring counter limitation value and enable ring counter function. When enable the ring counter function, the command and feedback counters will be operated as a ring counter.

When the ring counter limitation value be set to zero, the ring counter function will be disabled. For example, in Figure 1, when the ring counter limitation value(I32 RingVal) is set to 4, the command and feedback counters will count up until counter's value that is equal to four, then the command and feedback counters will be reset to zero and repeat above behavior. Relatively, in Figure 2, when the ring counter limitation value(I32 RingVal) is set to four, the command and feedback counters will count down until counter's value that is equal to zero, then the command and feedback counters will be reset to four and repeat above behavior

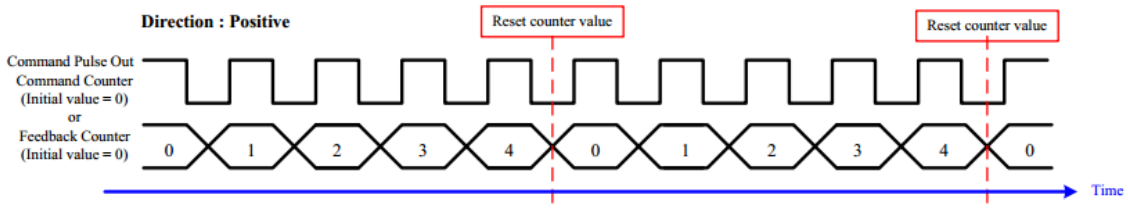


Figure 1

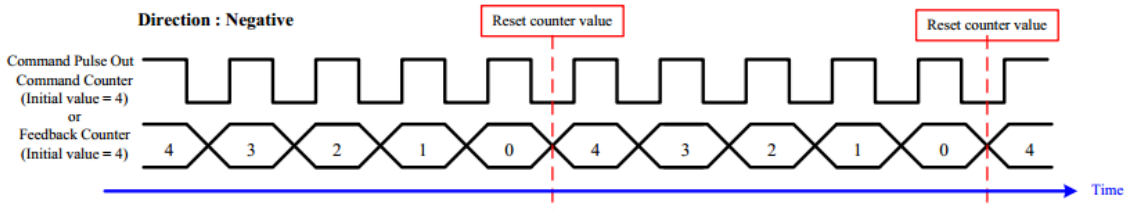


Figure 2

Syntax:

C/C++:

I32 FNTYPE APS_set_ring_counter(I32 AxisNo, I32 RingVal)

Visual Basic:

APS_set_ring_counter (ByVal AxisNo As Long, ByVal RingVal As Long) As Long

Parameters:

I32 AxisNo: The index of axis.

I32 RingVal: The limitation value of ring counter.(0 < RingVal< 134217727)

If RingVal equal to zero means disable ring counter function

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 AxisNo = 0; //Set axis ID
I32 RingVal     = 1000;       //Set the limitation value of ring counter
F64 Dist = 3000; //Set the relative distance to move (unit: pulse)
F64 MaxVel     = 1000;       //Set maximum velocity in units of pulse per second
APS_set_ring_counter(AxisNo, RingVal ); //Enable ring counter function
APS_relative_move(AxisNo, Dist, MaxVel); //Start relative move
……
APS_set_ring_counter(AxisNo, 0 ); //Disable ring counter function
```

See also:

APS_get_ring_counter ()

APS_get_ring_counter	Get limitation value of ring counter

Support Products: PCI-8154/58

Descriptions:

This function is used to get limitation value of ring counter.

Syntax:

C/C++:

I32 FNTYPE APS_get_ring_counter(I32 AxisNo, I32 *RingVal)

Visual Basic:

APS_get_ring_counter (ByVal AxisNo As Long, RingVal As Long) As Long

Parameters:

I32 AxisNo: The index of axis.

I32 *RingVal: Get the limitation value of ring counter.(0 < RingVal< 134217727)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 AxisNo = 0; //Set axis ID
I32 RingVal     = 1000;       //Set the limitation value of ring counter
F64 Dist = 3000; //Set the relative distance to move (unit: pulse)
F64 StrVel      = 0;          //Set starting velocity of a velocity profile in units of pulse per second
F64 MaxVel      = 1000;       //Set maximum velocity in units of pulse per second
APS_set_ring_counter(AxisNo, RingVal ); //Enable ring counter function
APS_relative_move(AxisNo, Dist, MaxVel); //Start relative move

APS_get_ring_counter(AxisNo, &RingVal ); //Get limitation value of ring counter
```

See also:

APS_set_ring_counter ()

33. Speed Profile Calculation

APS_relative_move_profile	Get relative speed profile

Support Products: PCI-C154(+)

Descriptions:

This function is used to get relative move speed profile. By this function, user can get the actual speed profile before motion. Therefore user needs to set speed pattern curve by aixs parameter PRA_CURVE(0x20) and start velocity by PRA_VS(0x23) for calculation profile.

Syntax:

C/C++:

I32 FNTYPE APS_relative_move_profile(I32 Axis_ID, I32 Distance, I32 Max_Speed, I32 *StrVel, I32 *MaxVel, F64 *Tacc, F64 *Tdec, F64 *Tconst)

Visual Basic:

APS_relative_move_profile(ByVal Axis_ID As Long, ByVal Distance As Long, ByVal Max_Speed As Long, StrVel As Long, MaxVel As Long, Tacc As Double, Tdec As Double, Tconst As Double) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Distance: Relative distance. Unit : pulse.

I32 Max_Speed: The maximum speed of this move profile. Unit: pulse/sec.

I32 * StrVel: Starting velocity. Unit: pulse/sec.

I32 * MaxVel: The maximum speed of this move profile. Unit: pulse/sec.

F64 * Tacc: Acceleration time by calculation. Unit: sec

F64 * Tdec: Deceleration time by calculation. Unit: sec

F64 * Tconst: Constant speed time(maximum speed). Unit: sec

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

I32 Axis_ID = 0;

I32 ret = 0;

I32 Distance = 100000;

```
I32 Max_Speed = 10000;
I32 StrVel =0,MaxVel=0;
F64 Tacc=0, Tdec=0, Tconst=0;
ret = APS_set_axis_param( Axis_ID, PRA_VS, 1000 ); // start velocity
ret = APS_set_axis_param( Axis_ID, PRA_CURVE, 0 ); // T curve
Ret = APS_relative_move_profile( Axis_ID, Distance, Max_Speed, &StrVel, &MaxVel, &Tacc,
&Tdec, &Tconst );
```

See also:

APS_absolute_move_profile	Get absolute speed profile

Support Products: PCI-C154(+)

Descriptions:

This function is used to get absolute move speed profile. By this function, user can get the actual speed profile before motion. Therefore user needs to set speed pattern curve by aixs parameter PRA_CURVE(0x20) and start velocity by PRA_VS(0x23) for calculation profile.

Syntax:

C/C++:

I32 FNTYPE APS_absolute_move_profile(I32 Axis_ID, I32 Position, I32 Max_Speed, I32 *StrVel, I32 *MaxVel, F64 *Tacc, F64 *Tdec, F64 *Tconst)

Visual Basic:

APS_relative_move_profile(ByVal Axis_ID As Long, ByVal position As Long, ByVal Max_Speed As Long, StrVel As Long, MaxVel As Long, Tacc As Double, Tdec As Double, Tconst As Double) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Position: Absolute command position. Unit is pulse.

I32 Max_Speed: The maximum speed of this move profile. Unit: pulse/sec.

I32 * StrVel: Starting velocity. Unit: pulse/sec.

I32 * MaxVel: The maximum speed of this move profile. Unit: pulse/sec.

F64 * Tacc: Acceleration time by calculation. Unit: sec

F64 * Tdec: Deceleration time by calculation. Unit: sec

F64 * Tconst: Constant speed time(maximum speed). Unit: sec

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Axis_ID = 0;
I32 ret = 0;
I32 Position = 100000;
I32 Max_Speed = 10000;
I32 StrVel =0,MaxVel=0;
F64 Tacc=0, Tdec=0, Tconst=0;
```

```
ret = APS_set_axis_param( Axis_ID, PRA_VS, 1000 ); // start velocity
ret = APS_set_axis_param( Axis_ID, PRA_CURVE, 1 ); // S curve
Ret = APS_absolute_move_profile( Axis_ID, Position, Max_Speed, &StrVel, &MaxVel, &Tacc,
&Tdec, &Tconst );
```

See also:

APS_check_motion_profile_emx	Get relative speed profile

Support Products: EMX-100

Descriptions:

This function is used to get relative move speed profile. By this function, user can get the actual
speed profile before motion. If calculated minimum distance is less than user expect, the controller will automatically calculate new dec, Vmax parameters that can refer Speed profile criteria.

.

Syntax:

C/C++:

I32 FNTYPE APS_check_motion_pfofile_emx(I32 Axis_ID, Speed_profile *profile_input, Speed_profile *profile_output, I32 *MinDis);

Visual Basic:

APS_check_motion_pfofile_emx(ByVal Axis_ID As Long, ByRef profile_input As Speed_profile, ByRef Param_Val As Speed_profile, ByRef MinDis As Long) As Long

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

Speed_profile profile_input: structure of input speed profile information.

Speed_profile profile_output: structure of output speed profile information.

I32 MinDis: The minimum distance by controller calculation user configuration input speed profile.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 Axis_ID = 0, MinDis = 0;
Speed_profile Inputprofile;
Speed_profile Outputprofile;
Inputprofile.Acc = 1000000;
Inputprofile.Dec = 1000000;
Inputprofile.Vmax = 1000000;
Inputprofile.VS = 1000;
Inputprofile.s_factor = 10;
```

ret = APS_check_motion_profile_emx(Axis_ID,&Inputprofile,&Outputprofile,&MinDis);

See also:

34. Backlash functions

APS_set_backlash_en	Enable/Disable backlash

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x

Descriptions:

This function is used to enable or disable backlash function. The backlash compensation is applied to that commanded axis movement by superimposition as the direction of axis movement is reversed. Users should configure the axis parameters PRA_BKL_DIST and PRA_BKL_CNSP prior to enable backlash. The former denotes the backlash compensation value, and the latter denotes the backlash compensation increment value every cycle. If this function is already enabled, it is not allowed to enable again. If user set *Enable* as 2, the output velocity will be limited by axis parameter PRA_VM (0x24). The motion status bit 27 is used to indicate backlash compensation is in operation or not.

The tables below show two backlash *Enable* mode examples: The backlash axis parameters are both 1000. The *Direction* denotes the motion status *DIR* bit, 1 for positive, and 0 for negative. And the *Command* and *Position* denote the command position and feedback position in user coordinate respectively. The *Encoder* denotes the encoder values of motor coordinate. It is supposed backlash is enabled after setting Servo ON, then run the point-to-point function step by step, and the backlash compensation is effected when the *Direction* is reversed.
The *Command* is return value of APS_get_command(f) .
The *Position* is return value of APS_get_position(f).
The *Encoder* is return value of APS_get_encoder(f), it is mapping to PDO actual position value for PCIe-833x series product.

Note:During backlash enable motion process, user can't dynamically switch Enable mode from 1 -> 2 or 2 -> 1. It is allowed Enable mode 1 -> 0 -> 2 or 2 -> 0 -> 1.

Table 1 An Example of setting *Enable* as 1: user coordinate position is NOT aligned

Step	Description	*Direction*	*Command*	*Position*	*Encoder*	Note
1	Initial condition	Positive	0	0	0	Servo ON
2	Forward 1000 pulse	Positive	1000	1000	1000	
3	Forward 1000 pulse	Positive	2000	2000	2000	
4	Backward 1000	Negative	1000	0	0	Backlash

	pulse					compensation
5	Backward 1000 pulse	Negative	0	-1000	-1000	
6	Backward 1000 pulse	Negative	-1000	-2000	-2000	
7	Forward 1000 pulse	Positive	0	0	0	Backlash compensation

Table 2 An Example of setting *Enable* as 2: user coordinate position is aligned

Step	Description	*Direction*	*Command*	*Position*	*Encoder*	Note
1	Initial condition	Positive	0	0	0	Servo ON
2	Forward 1000 pulse	Positive	1000	1000	1000	
3	Forward 1000 pulse	Positive	2000	2000	2000	
4	Backward 1000 pulse	Negative	1000	1000	0	Backlash compensation
5	Backward 1000 pulse	Negative	0	0	-1000	
6	Backward 1000 pulse	Negative	-1000	-1000	-2000	
7	Forward 1000 pulse	Positive	0	0	0	Backlash compensation

Syntax:

C/C++:

I32 FNTYPE APS_set_backlash_en (I32 Axis_ID, I32 Enable);

Visual Basic:

APS_set_backlash_en (ByVal Board_ID As Integer, ByVal Enable As Integer) As Integer

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 Enable: see description below

Enable	Description
0	Disable backlash compensation
1	a. Enable backlash compensation b. User coordinate position is NOT aligned c. The motion status bit 27 is used to indicate backlash compensation is in

	operation (=0) or not (=1).
2	a. Enable backlash compensation b. User coordinate position is aligned c. Output velocity is limited by axis parameter PRA_VM (0x24) d. The motion status bit 27 is used to indicate backlash compensation is in operation (=0) or not (=1).

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

APS_get_backlash_en	Check backlash is enabled/disabled

Support Products: PCI-8254/58 / AMP-204/8C, PCIe-833x

Descriptions:

This function is used to check backlash is enabled or disabled.

Syntax:

C/C++:

I32 FNTYPE APS_get_backlash_en(I32 Axis_ID, I32 *Enable);

Visual Basic:

APS_get_backlash_en (ByVal Board_ID As Integer, ByRef Enable As Integer) As Integer

Parameters:

I32 Axis_ID: The Axis ID from 0 to 65535.

I32 *Enable: see description below

Enable	Description
0	Disable backlash compensation
1	a. Enable backlash compensation b. User coordinate position is NOT aligned c. The motion status bit 27 is used to indicate backlash compensation is in operation (=0) or not (=1).
2	a. Enable backlash compensation b. User coordinate position is aligned c. Output velocity is limited by axis parameter PRA_VM (0x24) d. The motion status bit 27 is used to indicate backlash compensation is in operation (=0) or not (=1).

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

See also:

35. 2-D compensation

APS_set_2d_compensation_table	Create 2D compensation table

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to create 2D compensation table. Figure 1 shows the architecture of 2D compensation table. User can plan motion in user coordinate. When running linear interpolation function, the user coordinate target position will be modified by 2D compensation table. When motion is finished, user can get user coordinate command/position.

User need to give this function AxisID, total point number, start position, interval, and compensation data. It defines array index 0 for Axis X and array index 1 for Axis Y. It is not allowed to give zero or negative value to total point number and interval. A memory to save compensation data will be created internally based on given total point numbers. And this memory will be released when disabling table compensation function.

Figure 2 is an example of how to give function parameters. It gives start position (0 0), interval (10 10), and total point number (3 3). The dx_i and dy_j denote the compensation value for Axis X and Axis Y respectively, and the comp_data_x and comp_data_y are 1-D compensation data array.

Figure 3 shows the given comp. data should obey monotone principle, that is, $(x_i + dx_i) \leq (x_{i+1} + dx_{i+1})$ and $(y_j + dy_j) \leq (y_{j+1} + dy_{j+1})$. The dx_i and dy_j denote the compensation value and the x_i and y_j are the position of four points in compensation table. This function will return error code when giving improper comp. data.

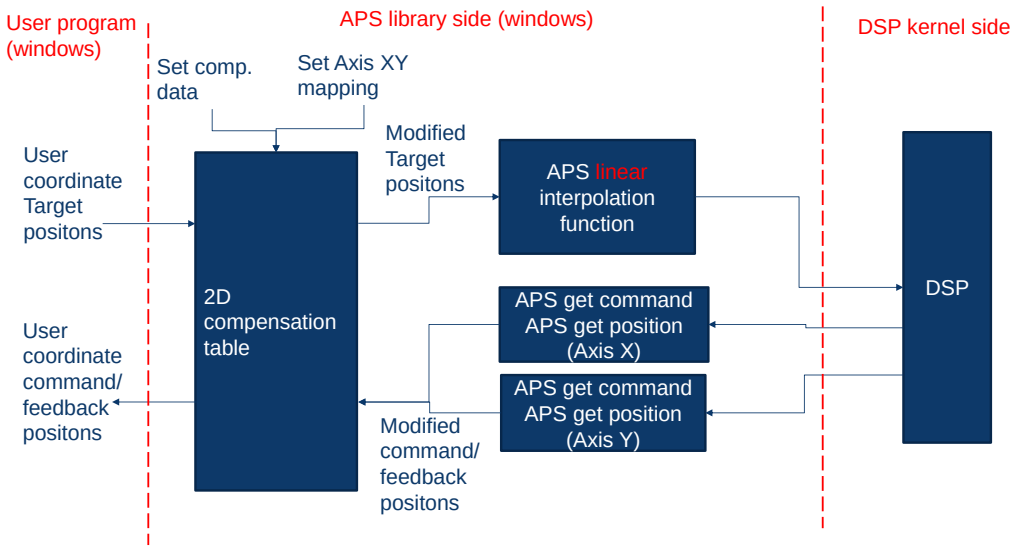
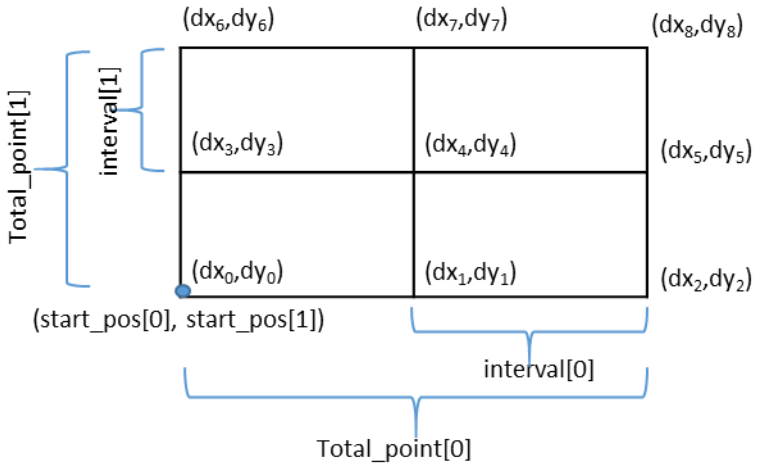
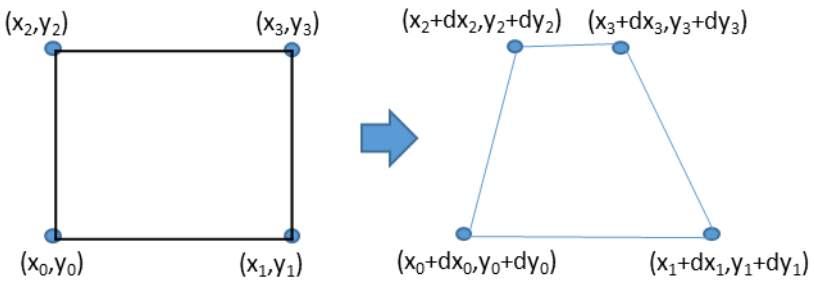


Figure 1 2-D compensation table schematic diagram

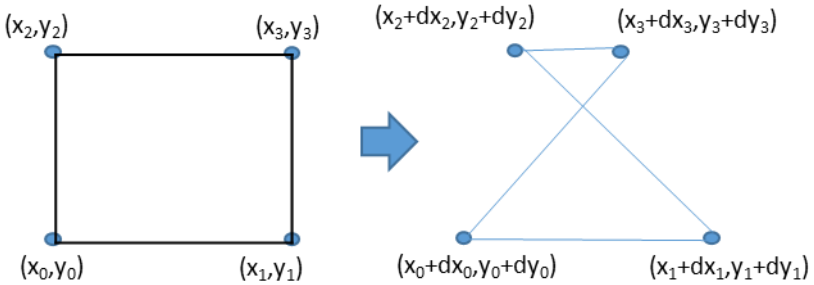


$$\text{comp\_data\_x}[i] = \text{dx}_i, \text{ and } i = 0 \sim 8$$
$$\text{comp\_data\_y}[j] = \text{dx}_j, \text{ and } j = 0 \sim 8$$

Figure 2 Example of 2-D compensation table configuration



(1) Proper comp. data



(2) Improper comp. data (line intersect)

Figure 3 Influence of choosing improper comp. data

Syntax:

C/C++:

I32 APS_set_2d_compensation_table(I32 *AxisIdArray, U32 CompType, U32 *TotalPointArray, F64 *StartPosArray, F64 *IntervalArray, F64 *CompDataArrayX, F64 *CompDataArrayY)

Visual Basic:

APS_set_2d_compensation_table (ByVal AxisIdArray() As Integer, ByVal CompType As UInteger, ByVal TotalPointArray() As UInteger, ByVal StartPosArray() As Double, ByVal IntervalArray() As Double, ByVal CompDataArrayX() As Double, ByRef CompDataArrayY() As Double) As Integer

Parameters:

I32 *AxisIdArray: Axis ID array; array size is 2; range from 0 to 65535.

U32 CompType: Error compensation method (reserved)

U32 *TotalPointArray: total point number array; array size is 2

F64 *StartPosArray: start position array; array size is 2

F64 *IntervalArray: interval array; array size is 2; (interval value > 0.0)

F64 *CompDataArrayX: Axis X compensation data array; array size > (TotalPointArray[0] * TotalPointArray[1])

F64 *CompDataArrayY: Axis X compensation data array; array size > (TotalPointArray[0] * TotalPointArray[1])

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

```
I32 ret = 0; // error code
I32 CompType = 0; // compensation method
U32 TotalPointArray[2] = {4,4}; // total point
F64 StartPosArray[2] = {-100,-100}; // start posiiton
F64 IntervalArray[2] = {100,100}; // interval
F64 CompDataArrayX[20]; // Axis X compensation data
F64 CompDataArrayY[20]; // Axis Y compensation data
F64 Position_Array[2] = {-100,-40}; // target position for linear interpolation
F64 Max_Linear_Speed = 10; // max speed for linear interpolation
F64 CommandX, CommandY, PositionX, PositionY; // user coordinate command position and feedback position
I32 Axis_ID_Array[2] = {0, 1}; // Axis X and Axis Y

// Example of compensation data for test
for(int i=1; i<16; i++)
{
	CompDataArrayX[i] = i;
	CompDataArrayY[i] = 15-i;
}
// Create 2D compensation table
ret = APS_set_2d_compensation_table( Axis_ID_Array, CompType, TotalPointArray, StartPosArray, IntervalArray, CompDataArrayX, CompDataArrayY);

// Enable 2D compensation table
ret = APS_start_2d_compensation(Axis_ID_Array[0], 1);

// Linear interpolation
ret = APS_absolute_linear_move_2d_compensation( Axis_ID_Array, Position_Array, Max_Linear_Speed );
```

```
// Get user coordinate command postion and feedback position
ret = APS_get_2d_compensation_command_position( Axis_ID_Array[0], &CommandX,
&CommandY, &PositionX, &PositionY );
```

APS_get_2d_compensation_table	Get 2D compensation table configuration

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get 2D compensation table configuration.

Syntax:

C/C++:

I32 APS_get_2d_compensation_table(I32 *AxisIdArray, U32 *CompType, U32 *TotalPointArray, F64 *StartPosArray, F64 *IntervalArray, F64 *CompDataArrayX, F64 *CompDataArrayY)

Visual Basic:

APS_get_2d_compensation_table (ByVal AxisIdArray() As Integer, ByRef CompType As UInteger, ByVal TotalPointArray() As UInteger, ByVal StartPosArray() As Double, ByVal IntervalArray() As Double, ByVal CompDataArrayX() As Double, ByRef CompDataArrayY() As Double) As Integer

Parameters:

I32 *AxisIdArray: Axis ID array; array size is 2; range from 0 to 65535.

U32 *CompType: Error compensation method (reserved)

U32 *TotalPointArray: total point number array; array size is 2

F64 *StartPosArray: start position array; array size is 2

F64 *IntervalArray: interval array; array size is 2; (interval value > 0.0)

F64 *CompDataArrayX: Axis X compensation data array; array size > (TotalPointArray[0] * TotalPointArray[1])

F64 *CompDataArrayY: Axis X compensation data array; array size > (TotalPointArray[0] * TotalPointArray[1])

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

APS_start_2d_compensation	Start or stop 2D compensation table

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x

Descriptions:

This function is used to start or stop 2D compensation table. It is noticed that the memory created for saving compensation data will be released when disabling 2D compensation function.

This function should be enabled before running *APS_absolute_linear_move_2d_compensation()* or *APS_get_2d_compensation_command_position()*.

Syntax:

C/C++:

I32 APS_start_2d_compensation(I32 Axis_ID, I32 Enable)

Visual Basic:

APS_start_2d_compensation (ByVal Axis_ID As Integer, ByVal Enable As Integer) As Integer

Parameters:

I32 Axis_ID: Axis ID of Axis X; range from 0 to 65535.

I32 Enable: Enable/disable 2D compensation table

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

APS_absolute_linear_move_2d_compensation	2D absolute linear interpolation

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to start an absolute linear interpolation positioning motion. The target position will be modified by 2D compensation table.

Syntax:

C/C++:

I32 APS_absolute_linear_move_2d_compensation(I32 *Axis_ID_Array, F64 *Position_Array, F64 Max_Linear_Speed)

Visual Basic:

APS_absolute_linear_move_2d_compensation (ByVal Axis_ID_Array() As Integer, ByVal Position_Array() As Double, ByVal Max_Linear_Speed As Double) As Integer

Parameters:

I32 *Axis_ID_Array: Axis ID array; array size is 2; from 0 to 65535.

F64 *Position_Array: Absolute position array; array size is 2 (unit: pulse)

F64 Max_Linear_Speed: Maximum linear interpolation speed (unit: pulse/sec)

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

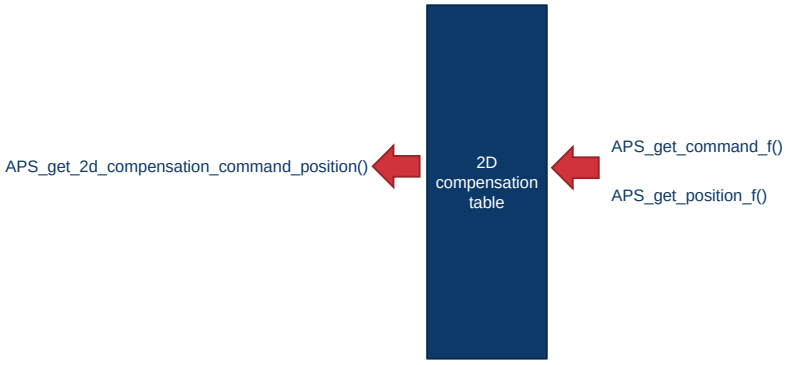
Example:

APS_get_2d_compensation_command_position	Get command and feedback position

Support Products: PCI-8254/58 / AMP-204/8C , PCIe-833x, ECAT-4XMO

Descriptions:

This function is used to get the command position and feedback position of one axis by double. The unit is pulse.The figure below explains the relationship between compensated position (*APS_get_command_f()andAPS_get_position_f()*)and user coordinate position (*APS_get_2d_compensation_command_position ()*).



Syntax:

C/C++:

I32 APS_get_2d_compensation_command_position(I32 Axis_ID, F64 *CommandX, F64 *CommandY, F64 *PositionX, F64 *PositionY)

Visual Basic:

APS_get_2d_compensation_command_position (ByVal Axis_ID As Integer, ByRef CommandX As Double, ByRef CommandY As Double, ByRef PositionX As Double, ByRef PositionY As Double) As Integer

Parameters:

I32 Axis_ID: Axis ID of Axis X; range from 0 to 65535.

F64 *CommandX: Axis X user coordinate command position. Unit in pulse.

F64 *CommandY: Axis Y user coordinate command position. Unit in pulse.

F64 *PositionX: Axis X user coordinate feedback position. Unit in pulse.

F64 *PositionY: Axis Y user coordinate feedback position. Unit in pulse.

Return Values:

I32 Error code: Please refer to APS Functions Return Code.

Example:

36. Table definition

A. Board Parameter table

DPAC-1000 board parameter table

DPAC-1000 board parameter table				
NO.	Define	Description	Parameter data meaning	Default
10h	PRB_WDT0_VALUE	WDT time out value.	0: Disable WDT N: 1~255 Start watch dog timer from N and down count. Down count period is WDT0_UNIT. When timer counter reaches zero (time out), WDT_ACTION will happen.	0
11h	PRB_WDT0_COUNTER	Restart WDT counting or get current WDT counter value	Set any value to restart WDT counter from WDT0_VALUE Get command to get current WDT counter.	0
12h	PRB_WDT0_UNIT	WDT counter down count period's unit	0: reserved 1:second 2: minute	0
13h	PRB_WDT0 _ACTION	WDT time out action	0: system reboot	0
20h	PRB_TMR0_BASE	Set TMR0 base unit clock	0~4095: TMR Value Timer period = (40 + (512 / 8.25) * TMR_value) us. Hardware interrupt will	0

			be genetated when each time out. To disable timer function, you must disable timer interrupt.	
21h	PRB_TMR0_VALUE	Get/Set timer0 value	32-bit unsign value. The counter increase one everytime when timer interrupt happens	0
30h	PRB_SYS_TMP_MONITOR	Get system temperature monitor data	8-bit signed value. The unit is degree of C	0
31h	PRB_CPU_TMP_MONITOR	Get CPU temperature monitor data	8-bit signed value. The unit is degree of C	0
32h	PRB_AUX_TMP_MONITOR	Get AUX temperature monitor data	8-bit signed value. The unit is degree of C	0
40h	PRB_UART_MULTIPLIER	Set UART Multiplier	0: x1 mode. If baud rate setting is 115200, the real baud rate is 115200. 1: x8 mode If baud rate setting is 115200, the real baud rate is 115200*8 = 921600.	0
90h	PRB_PSR_MODE	Set manual pulser generator (MPG) input mode	0: OUT/DIR 1: CW/CCW 2: 1x AB phase 3: 2x AB phase 4: 4x AB phase	4
91h	PRB_PSR_EA_LOGIC	Set EA signal logic	0: EA is not inverted 1: EA is inverted	0
92h	PRB_PSR_EB_LOGIC	Set EB signal logic	0: EB is not inverted 1: EB is inverted	0

10001h	PRB_DPAC_DISPLAY_MODE	DPAC Display mode	0: User Define Mode 1: Demo Mode	1
10002h	PRB_DPAC_DI_MODE	Set DI pin modes	0 : GPIO mode 1 : MPG input mode	0

DPAC-3000 board parameter table

DPAC-3000 board parameter table				
NO.	Define	Description	Parameter data meaning	Default
10h	PRB_WDT0_VALUE	WDT time out value.	0: Disable WDT N: 1~255 Start watch dog timer from N and down count. Down count period is WDT0_UNIT. When timer counter reaches zero (time out), WDT_ACTION will happen.	0
11h	PRB_WDT0_COUNTER	Restart WDT counting or get current WDT counter value	Set any value to restart WDT counter from WDT0_VALUE Get command to get current WDT counter.	0
12h	PRB_WDT0_UNIT	WDT counter down count period's unit	0: reserved 1:second 2: minute	0
13h	PRB_WDT0 _ACTION	WDT time out action	0: system reboot	0
20h	PRB_TMR0_BASE	Set TMR0 base unit clock	0~4095: TMR Value Timer period = (40 + (512 / 8.25) * TMR_value) us. Hardware interrupt will be genetated when each time out. To disable timer function, you must disable timer interrupt.	0

21h	PRB_TMR0_VALUE	Get/Set timer0 value	32-bit unsign value. The counter increase one everytime when timer interrupt happens	0
30h	PRB_SYS_TMP_MONITOR	Get system temperature monitor data	8-bit signed value. The unit is degree of C	0
31h	PRB_CPU_TMP_MONITOR	Get CPU temperature monitor data	8-bit signed value. The unit is degree of C	0
32h	PRB_AUX_TMP_MONITOR	Get AUX temperature monitor data	8-bit signed value. The unit is degree of C	0
40h	PRB_UART_MULTIPLIER	Set UART Multiplier	0: x1 mode. If baud rate setting is 115200, the real baud rate is 115200. 1: x8 mode If baud rate setting is 115200, the real baud rate is 115200*8 = 921600.	0
90h	PRB_PSR_MODE	Set manual pulser generator (MPG) input mode	0: OUT/DIR 1: CW/CCW 2: 1x AB phase 3: 2x AB phase 4: 4x AB phase	4
91h	PRB_PSR_EA_LOGIC	Set EA signal logic	0: EA is not inverted 1: EA is inverted	0
92h	PRB_PSR_EB_LOGIC	Set EB signal logic	0: EB is not inverted 1: EB is inverted	0
10001h	PRB_DPAC_DISPLAY_MODE	DPAC Display mode	0: User Define Mode 1: Demo Mode	1
10002h	PRB_DPAC_DI_MODE	Set DI pin modes	0 : GPIO mode 1 : MPG input mode	0

PCI-8392(H) board parameter table

PCI-8392(H) Board parameter table				
NO.	Define	Description	Parameter data meaning	Default
00h	PRB_EMG_LOGIC	EMG logic setting	0: Normal close 1: Normal open	0
10h	PRB_WDT0_VALUE	WDT time out value. (*1) Set 0 to disable watch dog. Set a positive value to enable watch dog function. When watch dog timer is enabled, wdt counter will count down per SSCNET cycle. Once WDT counter reaches zero (time out), SSCNET network will be stopped.	0: Disable WDT N(1~2147483647) (31 bits)	0
11h	PRB_WDT0_COUNTER	Restart WDT counting or get current WDT counter value. (*1)	Set any value to restart WDT counter from WDT0_VALUE Get command to get current WDT counter.	0
10000h	PRB_SSC_CYCLE_TIME	SSCNET 3 communication cycle time setting	0: 0.888ms 1: 0.444ms This value must be decided before start SSCNET communication	0

(*1) This parameter will not be saved to non-volatile memory (flash) when issue “APS_save_parameter_to_flash”

PCI-8253/56 board parameter table

PCI-8253/56 Board parameter table				
NO.	Define	Description	Parameter data meaning	Default
00h	PRB_EMG_LOGIC	EMG logic setting	0: Normal close 1: Normal open	0
10h	PRB_WDT0_VALUE	WDT time out value. (*1) Set 0 to disable watch dog. Set a positive value to enable watch dog function. When timer counter reaches zero (time out), servo signal will be turned off.	0: Disable WDT N: 1~2147483647 (31 bits) Start watch dog timer from N and down count. Down count period is cycle time.	0
11h	PRB_WDT0_COUNTER	Restart WDT counting or get current WDT counter value.(*1)	Set any value to restart WDT counter from WDT0_VALUE Get command to get current WDT counter.	0
80h	PRB_DENOMINATOR	Denominator	1~ 2147483647 Floating point type parameters will be divided by this value as its real value.	10,000
90h	PRB_PSR_MODE	Set manual pulser generator (MPG) input mode	0: OUT/DIR 1: CW/CCW 2: 1x AB phase 3: 2x AB phase 4: 4x AB phase	4
100h	PRB_BOOT_SETTING	The data source of axis and system parameters when	0: default table 1: Flash ROM	0

		DSP boots. DSP will reboot when power on or PCI bus reset.		
110h	PRB_PWM0_MAP_DO	Enable the mapping between PWM0 & Do. Specify a Do channel to map PWM0. Select its mapping logic between PWM0 & Do.	-1: Disable mapping Positive number: Enable mapping Bit0~7: Specify a Do channel. Bit8: Select logic. Set to 1: Turning on Do maps enabling PWM0. Turning off Do maps disabling PWM0. Set to 0: Turning on Do maps disabling PWM0. Turning off Do maps enabling PWM0.	-1
111h	PRB_PWM1_MAP_DO	Enable the mapping between PWM1 & Do. Specify a Do channel to map PWM1. Select its mapping logic between PWM1 & Do.	-1: Disable mapping Positive number: Enable mapping Bit0~7: Specify a Do channel. Bit8: Select logic. Set to 1: Turning on Do maps enabling PWM1. Turning off Do maps disabling PWM1. Set to 0: Turning on Do maps disabling PWM1. Turning off Do maps enabling PWM1.	-1
112h	PRB_PWM2_MAP_DO	Enable the mapping between PWM2 & Do. Specify a Do channel to map PWM2. Select its mapping logic between PWM2 &	-1: Disable mapping Positive number: Enable mapping Bit0~7: Specify a Do channel. Bit8: Select logic. Set to 1: Turning on Do maps enabling PWM2. Turning	-1

		Do.	off Do maps disabling PWM2. Set to 0: Turning on Do maps disabling PWM2. Turning off Do maps enabling PWM2.	
113h	PRB_PWM3_MAP_DO	Enable the mapping between PWM3 & Do. Specify a Do channel to map PWM3. Select its mapping logic between PWM3 & Do.	-1: Disable mapping Positive number: Enable mapping Bit0~7: Specify a Do channel. Bit8: Select logic. Set to 1: Turning on Do maps enabling PWM3. Turning off Do maps disabling PWM3. Set to 0: Turning on Do maps disabling PWM3. Turning off Do maps enabling PWM3.	-1

(*1) This parameter will not be saved to non-volatile memory (flash) when issue "APS_save_parameter_to_flash"

PCI(e)-7856 board parameter table

PCI(e)-7856 board parameter table				
NO.	Define	Description	Parameter data meaning	Default
20h	PRB_TMR0_BASE	Set TMR base unit clock	0~127: TMR Value Timer period = ((TMR_value + 2) * 0.1) ms. Hardware interrupt will be genetated when each time out. To disable timer function, you must disable timer interrupt.	0

EMX-100 board parameter table

EMX-100 Board parameter table				
NO.	Define	Description	Parameter data meaning	Default
51h(81)	PRB_DISCONNET_HANDLING	Set handling method for servo when network is disconnected	0: Stop 1: Servo off	0

PCI-8254/58 / AMP-204/8C board parameter table

PCI-8254/58 / AMP-204/8C Board parameter table				
NO.	Define	Description	Parameter data meaning	Default
00h	PRB_EMG_LOGIC	EMG Logic	0:Not inverse 1:Inverse	0
14h	PRB_DO_LOGIC	DO logic	0: no invert; 1: invert	0
15h	PRB_DI_LOGIC	DI logic	0: no invert; 1: invert	0
101h	PRS_EMG_MODE	EMG condition mode	0 (EMO): Servo off directly 1 (EMS): Emergency stop without servo off 2: Ignore all emergence handling but still update FPGA EMG status to motion IO status 3: Ignore all emergence handling and stop to update FPGA EMG status to motion IO status	0
110h	PRB_PWM0_MAP_DO	Enable the mapping between PWM0 & Do. Specify a Do channel to map PWM0. Select its mapping logic between PWM0 & Do. **NOTE: When disabling this parameter, the PWM output of VAO table may also be disabled.**	-1: Disable mapping Positive number: Enable mapping Bit0~7: Specify a Do channel. Bit8: Select logic. Set to 1: Turning on Do maps enabling PWM0. Turning off Do maps disabling PWM0. Set to 0: Turning on Do maps disabling PWM0. Turning off Do maps enabling PWM0.	-1
111h	PRB_PWM1_MAP_DO	Enable the	-1: Disable mapping	-1

		mapping between PWM1 & Do. Specify a Do channel to map PWM1. Select its mapping logic between PWM1 & Do. **NOTE: When disabling this parameter, the PWM output of VAO table may also be disabled.**	Positive number: Enable mapping Bit0~7: Specify a Do channel. Bit8: Select logic. Set to 1: Turning on Do maps enabling PWM1. Turning off Do maps disabling PWM1. Set to 0: Turning on Do maps disabling PWM1. Turning off Do maps enabling PWM1.	
112h	PRB_PWM2_MAP_DO	Enable the mapping between PWM2 & Do. Specify a Do channel to map PWM2. Select its mapping logic between PWM2 & Do. **NOTE: When disabling this parameter, the PWM output of VAO table may also be disabled.**	-1: Disable mapping Positive number: Enable mapping Bit0~7: Specify a Do channel. Bit8: Select logic. Set to 1: Turning on Do maps enabling PWM2. Turning off Do maps disabling PWM2. Set to 0: Turning on Do maps disabling PWM2. Turning off Do maps enabling PWM2.	-1
113h	PRB_PWM3_MAP_DO	Enable the mapping between PWM3 & Do. Specify a Do channel to map PWM3. Select its mapping logic between PWM3 & Do.	-1: Disable mapping Positive number: Enable mapping Bit0~7: Specify a Do channel. Bit8: Select logic. Set to 1: Turning on Do maps enabling PWM3. Turning off Do maps disabling PWM3.	-1

		NOTE: When disabling this parameter, the PWM output of VAO table may also be disabled.	Set to 0: Turning on Do maps disabling PWM3. Turning off Do maps enabling PWM3.	

PCIe-833x board parameter table

PCIe-833x Board parameter table				
NO.	Define	Description	Parameter data meaning	Default
00h	PRB_EMG_LOGIC	EMG Logic	0:Not inverse 1:Inverse	0
14h	PRB_DO_LOGIC	DO logic	0: no invert; 1: invert	0
15h	PRB_DI_LOGIC	DI logic	0: no invert; 1: invert	0
16h	PRB_IO_ACCESS_SEL	IO access selection	0: Mode 0 1: Reserved 2: Mode 2 (*Note 1)	0
17h	PRB_ECAT_SYNC_MODE	EtherCAT synchronization mode selection	0: DC mode (Default) 1: Free run	0
18h	PRB_OP_RETRY_COUNT	EtherCAT retry OP state count.		0
19h	PRB_ECAT_SERVO_ON_MODE	EtherCAT servo on mode selection	0: Standard mode(Default) 1: Fast mode, no check status word	0
1Ah	PRB_ECAT_SERVO_ON_NO_RESET_ALARM	EtherCAT servo on bypass reset alarm	0: No bypass reset alarm while servo on (Default) 1: Bypass reset alarm while servo on	0
20h	PRB_ECAT_SYNC_OFFSET	EtherCAT synchronization offset percentage value for DC mode.	Unit: Percenet Value range from 10 to 90.	66
101h	PRB_EMG_MODE	EMG condition mode	0 (EMO): Servo off directly 1 (EMS):	0

			Emergency stop without servo off	
104h	PRB_ECAT_RESTORE_OUTPUT	Keeps status for EtherCAT DIO/AIO slave device.	0:Don't keep the status. 1: Keep the status. (Default) (*Note 2)	1
0x105	PRB_DI_EMG_FILTER_ENABLE	Switch setting for on-board DI and EMG signal filter.	0: disable 1:enable	1
0x106	PRB_DI_EMG_FILTER_RANGE	Pulse-width setting for on-board DI and EMG signal filter. If the pulse-width of input signal is less than setting value of this parameter, the input signal will be cut-off.	0: 5 uSec. 1: 10 uSec. 2: 20 uSec. 3: 40 uSec. 4: 80 uSec 5: 160 uSec	0
0x107	PRB_PULSER_FILTER_RANGE	Pulse-width setting for on-board pulser signal filter. If the pulse-width of input signal is less than setting value of this parameter, the input signal will be cut-off. (The filter is always enable)	0: 5 uSec. 1: 10 uSec. 2: 20 uSec. 3: 40 uSec. 4: 80 uSec 5: 160 uSec	0
0x108	PRB_PULSER_FILTER_ENABLE	Pulser filter switch.	Pulser filter switch. 0: Disable 1: Enable	1
0x109	PRB_ECAT_AUTO_RECOVERY	EtherCAT auto recovery function	0: Disable 1: Enable	1

Note 1:

The **[TABLE 1]** as shown below that be used to identify the corresponding behavior and consumption time of APIs in each selection setting of the parameter **PRB_IO_ACCESS_SEL (0x16).**

[TABLE 1]

	PRB_IO_ACCESS_SEL(0x16) = 0	PRB_IO_ACCESS_SEL(0x16) = 2
APS_set_field_bus_d_channel_output	Mode: Synchronous	Mode: Synchronous
APS_get_field_bus_d_channel_output	Mode: Synchronous	Mode: Asynchronous
APS_get_field_bus_d_channel_input	Mode: Synchronous	Mode: Asynchronous
	PRB_IO_ACCESS_SEL(0x16) = 0	PRB_IO_ACCESS_SEL(0x16) = 2
APS_set_field_bus_d_port_output	Mode: Synchronous	Mode: Synchronous
APS_get_field_bus_d_port_input	Mode: Synchronous	Mode: Asynchronous
APS_get_field_bus_d_port_output	Mode: Synchronous	Mode: Asynchronous
	PRB_IO_ACCESS_SEL(0x16) = 0	PRB_IO_ACCESS_SEL(0x16) = 2
APS_get_field_bus_a_input	Mode: Synchronous	Mode: Synchronous
APS_set_field_bus_a_output	Mode: Synchronous	Mode: Synchronous
APS_get_field_bus_a_output	Mode: Synchronous	Mode: Synchronous

Note 2:

This parameter **PRB_ECAT_RESTORE_OUTPUT** doesn’t support EU-6000 DO modules by HW limitation.

B. Axis Parameter table

PCI-8392(H) Axis parameter table

PCI-8392(H) Axis parameter table				
NO. (Dec.)	Define	Description	Value	Default
00h (0)	PRA_EL_LOGIC	PEL/MEL input logic	0:Not inverse 1:Inverse	0
01h (1)	PRA_ORG_LOGIC	ORG input logic	0:Not inverse 1:Inverse	0
02h (2)	PRA_EL_MODE	The mode of stop when end limit ON	0: Deceleration stop 1: Stop immediately	0
03h (3)	PRA_MDN_CONDI	Motion done condition (Affective with motion stats NSTP bit)	0: Control command done (default) 1: Command done with INP 2: Command done with ZSP 3: Command done with INP & ZSP 4: Command done with soft INP	0
04h (4)~ 06h(6)	Reserved	Reserved	Reserved	
07h(7)	PRA_STP_DEC	Stop deceleration rate for APS_stop();	Unit: pulse/sec^2	100,000,000
08h(8)	PRA_SPEL_EN	Set Encorder event mode.	0: Disable 1: Encoder event 2: Soft-Limit (SPEL)	0
09h(9)	PRA_SMEL_EN	Set Encorder event mode.	0: Disable 1: Encoder event 2: Soft-Limit (SMEL)	0
0Ah(10)	PRA_EFB_POS0	SPEL / EFB position 0	Unit: pulse. (I32 value)	100,000
0Bh(11)	PRA_EFB_POS1	SMEL / EFB position 1	Unit: pulse. (I32 value)	-100,000
0Ch(12)	PRA_ EFB_CONDI0	Encoder compare condition. Feedback position >= or <= EFB pos0	0: Great equal (>=) 1: Less equal (<=)	0
0Dh(13)	PRA_ EFB_CONDI1	Encoder compare condition. Feedback position >= or <= EFB pos1	0: Great equal (>=) 1: Less equal	1

			(<=)	
0Eh(14)	PRA_EFB_SRC0	Encoder event pos0 comparing counter source.	0: Feedback position 1: Command position	0
0Fh(15)	PRA_EFB_SRC1	Encoder event pos0 comparing counter source.	0: Feedback position 1: Command position	0
10h (16)	PRA_HOME_MODE	Home mode setting	0: home mode 1	0
11h (17)	PRA_HOME_DIR	Homing direction	0: positive direction 1: negative direction	0
12h (18)	PRA_HOME_CURVE	Home move acceleration / Deceleration speed pattern	0: T-curve 1: S-curve	0
13h (19)	PRA_HOME_ACC	Home move acceleration/Deceleration rate	Unit: pulse/sec^2	22,520,000
14h (20)	PRA_HOME_VS	Homing start velocity	Unit: pulse/sec	0
15h (21)	PRA_HOME_VM	Homing maximum velocity	Unit: pulse/sec	225,200
16h (22)	Reserved (*1)			
17h (23)	Reserved (*1)			
18h (24)	PRA_HOME_EZC	Enable EZ signal alignment	0: Disable 1: Enable	0
19h (25)	PRA_HOME_VO	Homing leave home velocity	Unit: pulse/sec	112,600
1Ah-1Fh	Reserved			
20h (32)	PRA_CURVE	Acceleration / Deceleration speed pattern	0: T-Curve 1: S-Curve	0
21h (33)	PRA_ACC	Acceleration rate	Unit: pulse/sec^2	10,000,000
22h (34)	PRA_DEC	Deceleration rate	Unit: pulse/sec^2	10,000,000
23h (35)	PRA_VS	Start velocity	Unit: pulse/sec	0
24h (36)	Reserved			
25h (37)	PRA_VE	End velocity	Unit: pulse/sec	0
26h~2Fh	Reserved			
30h	Reserved			
31h	Reserved			
32h~3Fh	Reserved			
40h (64)	PRA_JG_MODE	Jog mode	0: Free mode 1: step mode	0
41h (65)	PRA_JG_DIR	Jog move direction	0: Positive direction 1:negative direction	0
42h (66)	PRA_JG_CURVE	Jog speed pattern	0: T-curve 1:S-curve	0
43h (67)	PRA_JG_ACC	Acceleration rate	Unit: pulse/sec^2	10,000,000
44h (68)	PRA_JG_DEC	Deceleration rate	Unit: pulse/sec^2	10,000,000
45h (69)	PRA_JG_VM	Max. velocity	Unit: pulse/sec	1,000,000

46h (70)	PRA_JG_STEP	Step offset	Unit: pulse (For step mode)	1,000
47h (71)	PRA_JG_DELAY	Delay time	Unit: ms (cycle time alignment) (For step mode)	500
50h(80)	PRA_MDN_DELAY	Motion done delay cycle (Affective with motion stats NSTP bit) The motion status NSTP bit will be turn on after specified delay cycle, when motion done condition is met.	Unit: system cycle time.	0
51h(81)	PRA_SINP_WDW	Soft INP window setting (Affective with motion I/O status INP bit). The motion I/O status INP bit will turn on when position into soft INP range and over INP stable cycle. The INP range define is as below INP range = (Target + window_setting) to (Target – window_setting). This function can be used by set PRA_MDM_CONDI parameter to command with soft INP.	Unit: pulse Value = 1 ~ 2147483647	200
52h(82)	PRA_SINP_STBL	Soft INP stable cycle (Affective with motion I/O status INP bit). The value decides how many cycles will turn on INP bit after position into soft INP range continuity.	Unit: system cycle time Value = 1 ~ 2147483647	100
82h(130)	PRA_MAX_E_LIMIT	Max encoder count. 2^{value} = encoder count limit(*5)	Unit : pulse 0 means rollover mode, other number means ring counter value and enable ring counter mode.	0
10000h	PRA_SSC_SERVO_PARAM_SRC	Select servo parameter source when start SSCNET	0: Do not update 1: Default value. 2: Flash memory	0
10001h	PRA_SSC_SERVO_ABS_POS_OPT	Enable absolute position system.	0: Disable. 1: Enable absolute position system	0
10002h	PRA_SSC_SERVO_ABS_CYC_CNT	Absolute cycle counter of servo driver	0 ~ 65535 (16 bit)	0
10003h	PRA_SSC_SERVO_ABS_RES_CNT	Absolute resolution counter of servo driver	0~262143 (18bit)	0

10004h	PRA_SSC_TORQUE_LIMIT_P	Positive torque limit value (0.1%) (*4)	0~32767	3,000
10005h	PRA_SSC_TORQUE_LIMIT_N	Negative torque limit value (0.1%) (*4)	0~32767	3,000
10006h	PRA_SSC_TORQUE_CTRL	Torque control enable (*3)	0: Disable, (Control with motor max. torque) 1: Enable, (Control with torque limit value)	0
10007h	PRA_SSC_RESOLUTION	E-gear factor 2^{Value} = resolution (*5)	Value = 12~18	18 (resolution = 262144)
10008h	PRA_SSC_GMR	E-gear factor molecular(*6)	Value = 1~1000000	1
10009h	PRA_SSC_GDR	E-gear factor denominator(*6)	Value = 1~1000000	1

(*1): Do not set any parameter data.

(*2): Reset to default value when start network.

(*3): Some SSCNET axis parameters will be rest to default value when you start SSCNET network.

(*4) 0.1% Set 1000 mean 100%

(*5): This parameter is valid after re-start SSCNET network.

(*6): This parameter is valid when PRA_SSC_RESOLUTION ==18 and after re-start SSCNET network.

$$\frac{1}{10} < \frac{PRA\_SSC\_GMR}{PRA\_SSC\_GDR} < 2000$$

PCI-8253/56 Axis parameter table

PCI-8253/56 axis parameter table.				
NO.	Define	Description	Value	Default
00h	PRA_EL_LOGIC	PEL/MEL input logic	0:Not inverse 1:Inverse	0
01h	PRA_ORG_LOGIC	ORG input logic	0:Not inverse 1:Inverse	0
02h	PRA_EL_MODE	The mode of stop when end limit ON	0: Deceleration stop 1: Stop immediately	1
03h	PRA_MDM_CONDI	Motion done condition (Affective with motion stats NSTP bit)	0: Control command done 1: Command done with INP 2: Command done with ZSP 3: Command done with INP & ZSP 4: Command done with soft INP	0
04h	PRA_ALM_LOGIC	Set ALM logic	0: Low active 1: High active	0
05h	PRA_ZSP_LOGIC	Set ZSP logic	0: Low active 1: High active	1
06h	PRA_EZ_LOGIC	Set EZ logic	0: Low active 1: High active	0
07h	PRA_STP_DEC	Stop deceleration rate for APS_stop();	Unit: pulse/sec^2	100,000,000
08h	PRA_SPEL_EN	Set Encorder event mode.	0: Disable 1: Encoder event 2: Soft-Limit (SPEL)	0
09h(9)	PRA_SMEL_EN	Set Encorder event mode.	0: Disable 1: Encoder event 2: Soft-Limit (SMEL)	0
0Ah(10)	PRA_EFB_POS0	SPEL / EFB position 0	Unit: pulse. (I32	100,000

			value)	
0Bh(11)	PRA_EFB_POS1	SMEL / EFB position 1	Unit: pulse. (I32 value)	-100,000
0Ch(12)	PRA_ EFB_CONDI0	Encoder compare condition. Feedback position >= or <= EFB pos0	0: Great equal (>=) 1: Less equal (<=)	0
0Dh(13)	PRA_ EFB_CONDI1	Encoder compare condition. Feedback position >= or <= EFB pos1	0: Great equal (>=) 1: Less equal (<=)	1
0Eh(14)	PRA_EFB_SRC0	Encoder event pos0 comparing counter source.	0: Feedback position 1: Command position	0
0Fh(15)	PRA_EFB_SRC1	Encoder event pos0 comparing counter source.	0: Feedback position 1: Command position	0
10h (16)	PRA_HOME_MODE	Home mode setting	0: home mode 1 (ORG) 1: home mode 2 (EL) 2: home mode 3 (EZ)	0
11h (17)	PRA_HOME_DIR	Homing direction	0: positive direction 1: negative direction	0
12h (18)	PRA_HOME_CURVE	Home move acceleration / Deceleration speed pattern	0: T-curve 1: S-curve	0
13h (19)	PRA_HOME_ACC	Home move acceleration/Deceleration rate	Unit: pulse/sec^2	22,520,000
14h (20)	PRA_HOME_VS	Homing start velocity	Unit pulse/sec	0
15h (21)	PRA_HOME_VM	Homing maximum velocity	Unit: pulse/sec	225,200
16h (22)	Reserved	(*1)		
17h (23)	Reserved	(*1)		
18h (24)	PRA_HOME_EZA	EZ alignment enable	0: Not enable 1: Enable	0
19h (25)	PRA_HOME_VO	Homing leave home velocity	Unit: pulse/sec	112,600
1Ah-1Fh	Reserved	(*1)		
20h(32)	PRA_CURVE	Acceleration / Deceleration speed pattern	0: T-Curve 1: S-Curve	0
21h(33)	PRA_ACC	Acceleration rate	Unit: pulse/sec2	10,000,000

22h(34)	PRA_DEC	Deceleration rate	Unit: pulse/sec2	10,000,000
23h(35)	PRA_VS	Start velocity	Unit: pulse/sec	0
24h(36)	Reserved	(*1)		
25h(37)	PRA_VE	End velocity	Unit: pulse/sec	0
30h(48)	Reserved	(*1)		
31h(49)	Reserved	(*1)		
32h(50)	PRA_PT_STP_DO_EN	Enable Do when point table stopping/pausing	0: Disable 1: Enable	0
33h(51)	PRA_PT_STP_DO	Set Do value when Point table stopping	0: Set to 0 1: Set to 1	0
34h(52)	PRA_PWM_OFF	Disable the specified PWM output when ASTP input signal is active.	0: Disable. No action when ASTP is active. 1: PWM_CH0 output will be disabled when ASTP is active. 2: PWM_CH1 output will be disabled when ASTP is active.	0
35h(53)	PRA_DO_OFF	Set Do value when ASTP input signal is active.	0: Disable. No action when ASTP is active. Bit0~3: select DO channel. Bit8: Set Do output value when ASTP is active. (*5)	0
40h(64)	PRA_JG_MODE	Jog mode	0: Free mode 1: step mode	0
41h(65)	PRA_JG_DIR	Jog move direction	0: Positive direction 1:negative direction	0
42h(66)	PRA_JG_CURVE	Jog speed pattern	0: T-curve 1:S-curve	0
43h(67)	PRA_JG_ACC	Acceleration rate	Unit: pulse/sec2	10,000,000

44h(68)	PRA_JG_DEC	Deceleration rate	Unit: pulse/sec2	10,000,000
45h(69)	PRA_JG_VM	Max. velocity	Unit: pulse/sec	10,000
46h(70)	PRA_JG_STEP	Step offset	Unit: pulse (For step mode)	1,000
47h(71)	PRA_JG_DELAY	Delay time	Unit: ms (cycle time alignment) (For step mode)	800
50h(80)	PRA_MDN_DELAY	Motion done delay cycle (Affective with motion stats NSTP bit) The motion status NSTP bit will be turn on after specified delay cycle, when motion done condition is met.	Unit: system cycle time.	0
51h(81)	PRA_SINP_WDW	Soft INP window setting (Affective with motion I/O status INP bit). The motion I/O status INP bit will turn on when position into soft INP range and over INP stable cycle. The INP range define is as below INP range = (Target + window_setting) to (Target – window_setting). This function can be used by set PRA_MDM_CONDI parameter to command with soft INP.	Unit: pulse Value = 1 ~ 2147483647	200
52h(82)	PRA_SINP_STBL	Soft INP stable cycle (Affective with motion I/O status INP bit). The value decides how many cycles will turn on INP bit after position into soft INP range continuity.	Unit: system cycle time Value = 1~ 2147483647	100
80h(128)	PRA_PLS_IPT_MODE	Pulse input mode	0: OUT/DIR 1: CW/CCW 2: 1x AB phase 3: 2x AB phase 4: 4x AB phase(default)	4
81h(129)	Reserved	(*1)		
82h(130)	PRA_MAX_E_LIMIT	Max encoder count	Unit : pulse 0 means rollover mode, other number means	0

			ring counter value and enable ring counter mode	
83h(131)	PRA_ENC_FILTER	Encoder filter	0 : Disable filter(Default) 1 : Enable filter(Neglect signal that smaller than 80ns)	0
84h(132)	PRA_EGEAR	E-Gear factor = Motor Encoder resolution(112h) / Value	Value = 1 ~ Motor Encoder resolution.	40,000
90h(144)	PRA_KP_GAIN	PID controller Kp gain (*2, *3)	Floating number	500
91h(145)	PRA_KI_GAIN	PID controller Ki gain(*2, *3)	Floating number	0
92h(146)	PRA_KD_GAIN	PID controller Kd gain (*2, *3)	Floating number	0
93h(147)	PRA_KFF_GAIN	Feed forward Kff gain (*2, *3)	Floating number	0
94h(148)	PRA_KVGTY_GAIN	Gantry Kgty gain (*2, *3)	Floating number	0
95h(149)	PRA_KPGTY_GAIN	Gantry Kpgty gain (*2, *3)	Floating number	0
96h(150)	PRA_IKP_GAIN	PID controller Kp gain in torque mode(*2, *3)	Floating number	10
97h(151)	PRA_IKI_GAIN	PID controller Ki gain in torque mode(*2, *3)	Floating number	0
98h(152)	PRA_IKD_GAIN	PID controller Kd gain in torque mod(*2, *3)	Floating number	0
99h(153)	PRA_IKFF_GAIN	Feed forward Kff gain in torque mode (*2, *3)	Floating number	0
100h(256)	PRA_M_INTERFACE	Motion interface	0: Analog motion	0
110h(272)	PRA_M_VOL_RANGE	Motor voltage input range (*3, *4)	Input value means ±(Value) volt	10
111h(273)	PRA_M_MAX_SPEED	Motor maximum speed (*3, *4)	Unit: RPS or mm / s	100 RPS

112h(274)	PRA_M_ENC_RES	Motor encoder resolution (*3, *4)	Unit: Pulse / rev or Pulse / mm	*40,000 Pulse / rev
120h(288)	PRA_V_OFFSET	Voltage offset (*2, *3)	Unit: volt	0
121h(289)	PRA_DZ_LOW	Dead zone lower side (*2, *3)	Unit: volt	0
122h(290)	PRA_DZ_UP	Dead zone upper side (*2, *3)	Unit: volt	0
123h(291)	PRA_SAT_LIMIT	Voltage saturation output limit (*2, *3)	Unit: volt	100000
124h(292)	PRA_ERR_C_LEVEL	Error counter check level	If set to 0, it means do not check error. Other value means error check then stop level	90000
125h(293)	PRA_V_INVERSE	Voltage ouput inverse	0: Not inverse, 1: Inverse	0
126h(294)	PRA_DZ_VAL	Assign dead band output value(*2, *3)	Unit: volt	0
127h(295)	PRA_IW_MAX	Integral windup upper limit value	Unit: Pulse(Value must input positive value)	45000
128h(296)	PRA_IW_MIN	Intergral windup lower limit value	Unit: Pulse(Value must input positive value)	45000
129h(297)	PRA_BKL_DIST	Use this parameter to define backlash length. If set to zero then backlash compensate function will be closed.	Unit: Pulse	0
12Ah(298)	PRA_BKL_CNSP	This parameter will define backlash compensate	Unit: Pulse	0

		consumption value. Because backlash compensate machine will consume pulse every cycle until backlash distance use up. And user must make sure initial state in motion direction before use backlash function (Direction initial state is negative, so user move positive will trigger backlash compensate machine ouput pulse).		
130h(304)	PRA_PSR_LINK	Connect pulser	0: Disable, 1: Enable	0
131h(305)	PRA_PSR_RATIO	Pulser ratio	Value = 1 ~ 2147483647	1
140h(320)	PRA_DA_TYPE	DAC output type	0: Differential output 1: Single output	0
141h(321)	PRA_CONTROL_MODE	Closed loop control mode (*3)	0: Velocity control loop 1: Torque control loop	0

*1: Do not set any parameter data.

*2: Change unit by setting system parameter 80h, if user want to change unit in program, remember re-set parameter after set system parameter 80h.

*3: Please give a correct value before use analog motion interface.

*4: This parameter is used to calculate a ratio that speed unit change to voltage unit

*5: Parameter value detail description

7	6	5	4	3	2	1	Bit : 0
-	-	-	-	1~8 :DO_CH0~ DO_CH7			
15	14	13	12	11	10	9	Bit : 8
-	-	-	-	-	-	-	ON/OFF

PCI-8144 Axis parameter table

PCI-8144 axis parameter table.				
NO.	Define	Description	Value	Default
00h	PRA_EL_LOGIC	Limit input logic	0: positive logic 1: negative logic	1
11h	PRA_HOME_DIR	Homing direction	0: positive direction 1: negative direction	0
15h	PRA_HOME_VM	Homing maximum velocity	Unit: pulse/sec	1000
1Ah	PRA_ORG_STP	Motion stop when ORG input is turned ON.	0: Disable 1: Enable	1
20h	PRA_CURVE	Acceleration / Deceleration speed pattern Change this parameter will affect motion parameters include PRA_ACC	0: T-curve 1: S-curve	0
21h	PRA_ACC	Acceleration rate / Deceleration rate. If ACC = 0, Axis feed as start velocity If ACC < 0, Axis feed as max velocity	Unit: pulse/s^2	99903
22h	Reserved			
23h	PRA_VS	Strart velocity	Unit: pulse/s	10
81h	PRA_PLS_OPT_MODE	Pulse output style (mode) selection. (logic)	0 = CW/CCW 1 = CW/CCW (logic inverse) 2 = OUT/DIR 3 = OUT/DIR (logic inverse)	0
212h	PRA_SD_EN	Enable slow down when SD input is turned ON.	0: Disable 1: Enable	0
240h	PRA_SPD_LIMIT	Posibile Maximum axis operation speed. Change this parameter will affect other motion parameters include PRA_ACC, PRA_VS	Unit: pulse/sec	409550
10000h	PRA_CMD_CNT_EN	Enable soft command counter.	0: Disable	0

			1: Enable	
10001h	PRA_MIO_SEN	Motion I/O: ORG, EL, STP input sensitivity setting.	0: High sensitivity 1: Low sensitivity	0
10002h	PRA_START_STA	Start(Trigger) motion via external input pin STA.	0: Disable 1: Enable	0
10003h	PRA_SPEED_CHN	(Set only) Set change speed command.	1: Change speed to start velocity 0: Change speed to max. speed.	0

AMP-104C Axis parameter table

AMP-104C axis parameter table.				
NO.	Define	Description	Value	Default
00h(0)	PRA_EL_LOGIC	Limit input logic	0: positive logic 1: negative logic	1
10h(16)	PRA_HOME_MODE	Home mode setting	0:Home mode 0 (ORG + SD) 1:Home mode 1 (ORG + high constant speed) 2: Home mode 2 (ORG+ high constant speed+ down counter) *(1)	0
11h(17)	PRA_HOME_DIR	Homing direction	0: positive direction 1: negative direction	0
15h(21)	PRA_HOME_VM	Homing maximum velocity	Unit: pulse/sec	1000
1Ah(26)	PRA_ORG_STP	Motion stop when ORG input is turned ON.	0: Disable 1: Enable	1
1Fh(31)	PRA_HOME_DOWN_COUNTER	Homing down counter.	Unit: pulse 0 ~ 16777215 *(2)	0
20h(32)	PRA_CURVE	Acceleration / Deceleration speed pattern Change this parameter will affect motion parameters include PRA_ACC	0: T-curve 1: S-curve	0
21h(33)	PRA_ACC	Acceleration rate / Deceleration rate. If ACC = 0, Axis feed as start velocity If ACC < 0, Axis feed as max velocity	Unit: pulse/s^2	99984
23h(35)	PRA_VS	Strart velocity	Unit: pulse/s	50
80h(128)	PRA_PLS_IPT_MODE	Pulse input mode	0: OUT/DIR 1: CW/CCW 2: 1x AB phase 3: 2x AB phase 4: 4x AB phase	1

81h(129)	PRA_PLS_OPT_MODE	Pulse output style (mode) selection. (logic)	0 = CW/CCW 1 = CW/CCW (logic inverse) 2 = OUT/DIR 3 = OUT/DIR (logic inverse)	2
212h(530)	PRA_SD_EN	Enable slow down when SD input is turned ON.	0: Disable 1: Enable	1
240h(576)	PRA_SPD_LIMIT	Posibile Maximum axis operation speed. Change this parameter will affect other motion parameters include PRA_ACC, PRA_VS. This configuration value is between 960 ~ 4914600.	Unit: pulse/sec	409550
10000h(65536)	PRA_CMD_CNT_EN	Enable soft command counter.	0: Disable 1: Enable	1
10001h(65537)	PRA_MIO_SEN	Motion I/O: ORG, EL, STP input sensitivity setting.	0: High sensitivity (no filter) 1: Low sensitivity (with fileter)	0
10002h(65538)	PRA_START_STA	Start(Trigger) motion via external input pin STA.	0: Disable 1: Enable	0
10003h(65539)	PRA_SPEED_CHN	(Set only) Set change speed command.	1: Change speed to start velocity 0: Change speed to max. speed.	0

*(1)

If user choose home mode 0,please make sure PRA_ACC > 0 and PRA_SD_EN enable . Otherwise when start home move that will return ERR_FunctionNotAvailable error(-13).

If user choose home mode 1,please make sure PRA_ACC < 0 and PRA_SD_EN disable. Otherwise when start home move that will return ERR_FunctionNotAvailable error(-13).

If user choose home mode 2,please make sure PRA_ACC < 0 ,PRA_SD_EN disable and PRA_HOME_DOWN_COUNTER > 0 . Otherwise when start home move that will return ERR_FunctionNotAvailable error(-13).

*(2)

This counuter will decrease (pulse number)when motor starts motion. Until counter equals zero the motor will stop motion. That is in order to prevent from endless operation that caused by breakage of origin switch.

MNET-4XMO-(C) Axis parameter table

MNET-4XMO-(C) axis parameter table.				
NO.	Define	Description	Value	Default
00h (0)	PRA_EL_LOGIC	PEL/MEL input logic	0:Not inverse 1:Inverse	0
01h (1)	PRA_ORG_LOGIC	ORG input logic	0:Active low 1:Active high	0
02h (2)	PRA_EL_MODE	The mode of stop when end limit ON	0: Deceleration stop 1: Stop immediately	0
03h (3)	PRA_MDN_CONDI	Motion done condition (Affective with motion stats NSTP bit)	0: Control command done (default) 1: Command done with INP	0
04h (4)	PRA_ALM_LOGIC	Set ALM Logic	0: Active low 1: Active high	0
05h(5)	Reserved			
06h(6)	PRA_EZ_LOGIC	Set EZ Logic	0: Falling edge 1: Rising edge	0
07h(7)	Reserved			
08h	PRA_SPEL_EN	Set Encorder event mode. (*3,)	0: Disable 1: Reserved 2: Soft-Limit (SPEL) Note: mode 1 is reserved. If set, return error.	0
09h(9)	PRA_SMEL_EN	Set Encorder event mode. (*3,)	0: Disable 1: Reserved 2: Soft-Limit (SMEL) Note: mode 1 is reserved. If set, return error.	0
0Ah(10)	PRA_EFB_POS0	SPEL / EFB position 0 (*3,)	Unit: pulse. (I32 value) (28-bit signed)	100,000
0Bh(11)	PRA_EFB_POS1	SMEL / EFB position 1 (*3,)	Unit: pulse. (I32 value) (28-bit signed)	-100,000
0Ch(12)	Reserved			
0Dh(13)	Reserved			
0Eh(14)	Reserved			
0Fh(15)	Reserved			
10h (16)	PRA_HOME_MODE	Home mode setting	Home search – 0(1st mode) to 12(13th mode) Home move – 20(1st mode) to 32(13th mode)	0

			Note: Home search (6 to 8) is reserved. If set, return error.	
11h (17)	PRA_HOME_DIR	Homing direction	0: positive direction 1: negative direction	0
12h (18)	Reserved			
13h (19)	Reserved			
14h (20)	Reserved			
15h (21)	PRA_HOME_VM	Homing maximum velocity	Unit: pulse/sec	10000
16h (22)	Reserved			
17h (23)	Reserved			
18h (24)	PRA_HOME_EZA	Specify the EZ count up value	0000(1st count) to 1111(16th count)	0
19h (25)	PRA_HOME_VO	Homing leave home velocity – Specify FA speed	Unit: pulse/sec	152
1Ah	PRA_HOME_OFFSET	Homing leave home distance – Specify ORG offset	Unit: pulse	100
1Bh-1Fh	Reserved			
20h (32)	PRA_CURVE	Acceleration / Deceleration speed pattern(*4)	0: T-Curve 1: S-Curve	0
21h (33)	PRA_ACC	Acceleration rate	Unit: pulse/sec^2	1000000
22h (34)	PRA_DEC	Deceleration rate	Unit: pulse/sec^2	1000000
23h (35)	PRA_VS	Start velocity	Unit: pulse/sec	152
24h~26h	Reserved			
28h	PRA_ACC_SR	S curve ratio in acceleration.(*4)	Unit: Milli %. Value = 1 ~ 100,000	100,000
29h	PRA_DEC_SR	S curve ratio in deceleration(*4)	Unit: Milli %. Value = 1 ~ 100,000	100,000
2Ah~50h	Reserved			
53h(83)	PRA_SERVO_LOGIC	SERVO output logic	0: Active low 1: Active high	0
80h(128)	PRA_PLS_IPT_MODE	Pulse input mode	0: A/B X1 1: A/B X2 2: A/B X4 3: CW/CCW	0
81h(129)	PRA_PLS_OPT_MODE	Pulse output mode	0: OUT/DIR (AL,H+) 1: OUT/DIR (AH,H+) 2: OUT/DIR (AL,L+) 3: OUT/DIR (AH,L+) 4: CW/CCW (AH) 5: CW/CCW (AL) 6: AB (Out Leading) 7: AB (Out Lagging)	0
84h(132)	PRA_EGEAR	E-Gear factor = Motor Encoder resolution(112h) / Value (*1,)	Value = 1 ~ Motor Encoder resolution.	40,000

112h(274)	PRA_M_ENC_RES	Motor encoder resolution (*1,)	Unit: Pulse / rev or Pulse / mm	40,000
200h(512)	PRA_PLS_IPT_LOGIC	Pulse input logic	0: don not reverse counting direction 1: reverse counting direction	0
201h(513)	PRA_FEEDBACK_SRC	Select feedback source	0: Ext. Encoder mode Ext. Encoder counter & Absolute mode reference to Encoder counter. 1: Stepper mode Ext. Command counter & Absolute mode reference to Command counter. 2: ACServo mode Ext. Encoder counter & Absolute mode reference to Command counter.	0
210h(528)	PRA_ALM_MODE	ALM mode setting	0: Immediate stop 1: Slow down then stop	0
211h(529)	PRA_INP_LOGIC	INP input logic	0: Active low 1: Active high	0
212h(530)	PRA_SD_EN	Enable SD. (*2)	0: Disable 1: Enable	0
213h(531)	PRA_SD_MODE	SD mode setting	0: Only slow down 1: Slow down and stop	0
214h(532)	PRA_SD_LOGIC	SD input logic	0: Active low 1: Active high	0
215h(533)	PRA_SD_LATCH	Latch SD input	0: Disable latch function 1: Enable latch function	0
216h(534)	PRA_ERC_MODE	ERC mode setting	0: disable 1: output ERC when stopped by EL, ALM, or EMG input 2: output ERC when complete home return 3: both 1 and 2	3
217h(535)	PRA_ERC_LOGIC	ERC output logic	0: Active low 1: Active high	0
218h(536)	PRA_ERC_LEN	Pulse width of ERC setting	0: 12 us 1: 102 us 2: 409 us 3: 1.6 ms 4: 13 ms 5: 52 ms 6: 104 ms 7: Level Output	3

219h(537)	Reserved			
21Ah(538)	Reserved			
21B(539)	PRA_PLS_IPT _FLT	EA/EB Filter Enable	0: Disable 1: Enable	1
21C	Reserved			
21D	PRA_LTC_LOGIC	LTC input logic	0: Falling edge 1: Rising edge	0
21E	PRA_IO_FILTER	Apply a filter to the PEL, MEL, SD, ORG, ALM, INP inputs. When a filter is applied, signal pulses shorter than 4 micro-second is ignored.	0: Don't apply a filter 1: Apply a filter	1
21F~220	Reserved			
221	PRA_ COMPENSATION _PULSE	A backlash or slip correction amount	0 to 4095	0
222	PRA_ COMPENSATION _MODE	Backlash or slip mode setting	0: Disable 1: Backlash correction 2: Slip correction	0
223	PRA_LTC_SRC	Select latch source	0: LTC pin 1: ORG pin 2: GCMP ON	0
224	PRA_LTC_DEST	Select latch target	0: Command counter 1: Position counter	0
225	PRA_LTC_DATA	Get latch data **(Read only)**	Pulse (28-bit signed)	0
226	PRA_GCMP_EN	Genernal comparator enable & set method	0: Disable Other: Enable 1:data = cmp counter (regardless of counting direction) 2: data=cmp counter (while counting up)	0

			3: data=cmp counter (while counting down) 4: data>cmp counter 5: data<cmp counter	
227	PRA_GCMP_POS	General comparator position	Pulse (28-bit signed)	0
228	PRA_GCMP_SRC	Select general comparator source	0: Command counter 1: Position counter	0
229	PRA_GCMP_ACTION	Select action when GCMP are met.	0: Do nothing 1: Immediate stop 2: Deceleration stop	0
22A	PRA_GCMP_STS	Check if GCMP is met **(Read only)**	0: Not meet 1: meet	0
22B	PRA_VIBSUP_RT	Supress vibration – Reverse Time	Unit: 1.6 us (16-bit unsigned)	0
22C	PRA_VIBSUP_FT	Supress vibration – Forward Time	Unit: 1.6 us (16-bit unsigned)	0
22D	PRA_LATCH_DATA_SPD	Choose latch data of error position or current speed for latch No.2	0: Latch error position 1: Latch current speed	0
22E~230	Reserved			
231	PRA_GPDI_SEL	Select gpio input – DI / LTC / SD. (*2)	0: DI 1: LTC 2: SD	0
232	Reserved			
233(563)	PRA_RDY_LOGIC	RDY input logic	0: Active high 1: Active low	0
234h~23Fh	Reserved			
240h(576)	PRA_SPD_LIMIT	Set Fixed Speed	Unit: pulse/sec	9999847
241h(577)	PRA_MAX_ACCDEC	Get max acceleration /deceleration which is limited by fixed speed. **(Read only)** (*5)	Unit: pulse/sec^2	572204589

242h(578)	PRA_MIN_ACCDEC	Get minimum acceleration/deceleration which is limited by fixed speed. **(Read only)** (*5)	Unit: pulse/sec^2	17462
260h(608)	PRA_SYNC_STOP_MODE	Set stop mode when stopping simultaneous move	0: Immediate stop 1: Deceleration stop	0

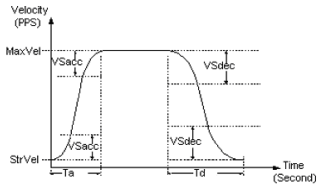
*1: This parameter is used to calculate a move ratio. It is only effective when PRA_FEEDBACK_SRC was set to 0 or 2.

*2: When PRA_GPDI_SEL set to DI/LTC, PRA_SD_EN automatically set to disable. Before PRA_SD_EN set to enable, be sure that PRA_GPDI_SEL set to SD mode.

*3: When positive or negative software limit is selected, command counter is used as the comparison counter. The comparison method is mentioned as follows:
(EFB position 0 < command counter) for positive software limit, (EFB position 1 > command counter) for negative software limit.

*4: If PRA_ACC_SR and PRA_DEC_SR are set to 100,000, it represents the curve profile is pure S curve. If PRA_ACC_SR or PRA_DEC_SR is not equal to 100,000, it represents the curve profile is S curve with linear range.

The formula is listed as below:



PRA_ACC_SR =
2Svacc / (MaxV – StrV) * 100,000 milli%

PRA_DEC_SR =
2Svacc / (MaxV – StrV) * 100,000 milli%

*5: According to (*4), when the curve profile is set to S curve with linear range, the PRA_MAX_ACCDEC and PRA_MIN_ACCDEC are always return 0. The PRA_MAX_ACCDEC and PRA_MIN_ACCDEC are only available in T and pure S curve mode.

MNET-1XMO Axis parameter table

MNET-1XMO axis parameter table.				
NO.	Define	Description	Value	Default
00h (0)	Reserved			
01h (1)	PRA_ORG_LOGIC	ORG input logic	0:Active low 1:Active high	0
02h (2)	PRA_EL_MODE	The mode of stop when end limit ON	0: Deceleration stop 1: Stop immediately	0
03h (3)	PRA_MDN_CONDI	Motion done condition (Affective with motion stats NSTP bit)	0: Control command done (default) 1: Command done with INP	0
04h (4)	PRA_ALM_LOGIC	Set ALM Logic	0: Active low 1: Active high	1
05h(5)	Reserved			
06h(6)	PRA_EZ_LOGIC	Set EZ Logic	0: Falling edge 1: Rising edge	0
07h(7)	Reserved			
08h	PRA_SPEL_EN	Set Encorder event mode. (*2,)	0: Disable 1: Reserved 2: Soft-Limit (SPEL)	0
09h(9)	PRA_SMEL_EN	Set Encorder event mode. (*2,)	0: Disable 1: Reserved 2: Soft-Limit (SMEL)	0
0Ah(10)	PRA_EFB_POS0	SPEL / EFB position 0 (*2,)	Unit: pulse. (I32 value) (28-bit signed)	100,000
0Bh(11)	PRA_EFB_POS1	SMEL / EFB position 1 (*2,)	Unit: pulse. (I32 value) (28-bit signed)	-100,000
0Ch(12)	Reserved			
0Dh(13)	Reserved			
0Eh(14)	Reserved			
0Fh(15)	Reserved			
10h (16)	PRA_HOME_MODE	Home mode setting	Home search – 0(1st mode) to 12(13th mode) Home move – 20(1st mode) to 32(13th mode) Note: Home search (6 to 8) is reserved. If set, return error.	0
11h (17)	PRA_HOME_DIR	Homing direction	0: positive direction 1: negative direction	0
12h (18)	Reserved			
13h (19)	Reserved			
14h (20)	Reserved			
15h (21)	PRA_HOME_VM	Homing maximum	Unit: pulse/sec	10000

		velocity		
16h (22)	Reserved			
17h (23)	Reserved			
18h (24)	PRA_HOME_EZA	Specify the EZ count up value	0000(1st count) to 1111(16th count)	0
19h (25)	PRA_HOME_VO	Homing leave home velocity – Specify FA speed	Unit: pulse/sec	0
1Ah	PRA_HOME_OFFSET	Homing leave home distance – Specify ORG offset	Unit: pulse	100
1Bh-1Fh	Reserved			
20h (32)	PRA_CURVE	Acceleration / Deceleration speed pattern	0: T-Curve 1: S-Curve	0
21h (33)	PRA_ACC	Acceleration rate	Unit: pulse/sec^2	1000000
22h (34)	PRA_DEC	Deceleration rate	Unit: pulse/sec^2	1000000
23h (35)	PRA_VS	Start velocity	Unit: pulse/sec	66
24h ~ 27h	Reserved			
28h	PRA_ACC_SR	S curve ratio in acceleration.(*4)	Unit: Milli %. Value = 1 ~ 100,000	100,000
29h	PRA_DEC_SR	S curve ratio in deceleration(*4)	Unit: Milli %. Value = 1 ~ 100,000	100,000
2Ah~50h	Reserved			
53h(83)	PRA_SERVO_LOGIC	SERVO output logic	0: Active low 1: Active high	0
80h(128)	PRA_PLS_IPT_MODE	Pulse input mode	0: A/B X1 1: A/B X2 2: A/B X4 3: CW/CCW	0
81h(129)	PRA_PLS_OPT_MODE	Pulse output mode	0: OUT/DIR (AL,H+) 1: OUT/DIR (AH,H+) 2: OUT/DIR (AL,L+) 3: OUT/DIR (AH,L+) 4: CW/CCW (AH) 5: CW/CCW (AL) 6: AB (Out Leading) 7: AB (Out Lagging)	0
84h(132)	PRA_EGEAR	E-Gear factor = Motor Encoder resolution(112h) / Value (*1,)	Value = 1 ~ Motor Encoder resolution.	40,000
112h(274)	PRA_M_ENC_RES	Motor encoder resolution (*1,)	Unit: Pulse / rev or Pulse / mm	40,000
200h(512)	PRA_PLS_IPT_LOGIC	Pulse input logic	0: don not reverse EA/EB counting 1: reverse EA/EB counting	0
201h(51	PRA_FEEDBACK_S	Select feedback	0: Ext. Encoder mode	0

3)	RC	source	Ext. Encoder counter & Absolute mode reference to Encoder counter. 1: Stepper mode Ext. Command counter & Absolute mode reference to Command counter. 2: ACServo mode Ext. Encoder counter & Absolute mode reference to Command counter.	
210h(528)	PRA_ALM_MODE	ALM mode setting	0: Immediate stop 1: Slow down then stop	0
211h(529)	PRA_INP_LOGIC	INP input logic	0: Active low 1: Active high	0
212h(530)	PRA_SD_EN	Enable SD	0: Disable 1: Enable	0
213h(531)	PRA_SD_MODE	SD mode setting	0: Only slow down 1: Slow down and stop	0
214h(532)	PRA_SD_LOGIC	SD input logic	0: Active low 1: Active high	0
215h(533)	PRA_SD_LATCH	Latch SD input	0: Disable latch function 1: Enable latch function	0
216h(534)	PRA_ERC_MODE	ERC mode setting	0: disable 1: output ERC when stopped by EL, ALM, or EMG input 2: output ERC when complete home return 3: both 1 and 2	3
217h(535)	PRA_ERC_LOGIC	ERC output logic	0: Active low 1: Active high	0
218h(536)	PRA_ERC_LEN	Pulse width of ERC setting	0: 12 us 1: 102 us 2: 409 us 3: 1.6 ms 4: 13 ms 5: 52 ms 6: 104 ms 7: Level Output	3
219h(537)	Reserved			
21Ah(538)	Reserved			
21B(539)	PRA_PLS_IPT _FLT	EA/EB Filter Enable	0: Enable 1: Disable	1

21C	Reserved			
21D	PRA_LTC_LOGIC	LTC input logic	0: Falling edge 1: Rising edge	0
21E	PRA_IO_FILTER	Apply a filter to the PEL, MEL, SD, ORG, ALM, INP inputs. When a filter is applied, signal pulses shorter than 4 micro-second is ignored.	0: Don't apply a filter 1: Apply a filter	1
21F~220	Reserved			
221	PRA_COMPENSATION_PULSE	A backlash correction amount	0 to 4095	0
222	PRA_ COMPENSATION _MODE	Backlash mode setting	0: Disable 1: Backlash correction	0
223	PRA_LTC_SRC	Select latch source	0: LTC pin 1: ORG pin 2: GCMP ON	0
224	PRA_LTC_DEST	Select latch target	0: Command counter 1: Position counter	0
225	PRA_LTC_DATA	Get latch data (Read only)	Pulse (28-bit signed)	0
226	PRA_GCMP_EN	Genernal comparator enable & set method	0: Disable Other: Enable 1:data = cmp counter (regardless of counting direction) 2: data=cmp counter (while counting up) 3: data=cmp counter (while counting down) 4: data>cmp counter 5: data<cmp counter	0
227	PRA_GCMP_POS	General comparator position	Pulse (28-bit signed)	0
228	PRA_GCMP_SRC	Select general comparator source	0: Command counter 1: Position counter	0
229	PRA_GCMP_ACTION	Select action when GCMP are met.	0: Do nothing 1: Immediate stop	0

			2: Deceleration stop	
22A	PRA_GCMP_STS	Check if GCMP is met **(Read only)**	0: Not meet 1: meet	0
22B	PRA_VIBSUP_RT	Supress vibration – Reverse Time	Unit: 1.6 us (16-bit unsigned)	0
22C	PRA_VIBSUP_FT	Supress vibration – Forward Time	Unit: 1.6 us (16-bit unsigned)	0
22D	PRA_LATCH_DATA_SPD	Choose latch data of error position or current speed for latch No.2	0: Latch error position 1: Latch current speed	0
22F ~23F	Reserved			
240h(576)	PRA_SPD_LIMIT	Set Fixed Speed	Unit: pulse/sec	6666666
241h(577)	PRA_MAX_ACCDEC	Get max acceleration/deceleration which is limited by fixed speed. **(Read only)**	Unit: pulse/sec^2	166666666
242h(578)	PRA_MIN_ACCDEC	Get minimum acceleration/deceleration which is limited by fixed speed. **(Read only)**	Unit: pulse/sec^2	5086

*1: This parameter is used to calculate a move ratio. It is only effective when PRA_FEEDBACK_SRC was set to 0 or 2.

*2: When positive or negative software limit is selected, command counter is used as the comparison counter. The comparison method is mentioned as follows:
(EFB position 0 < command counter) for positive software limit, (EFB position 1 > command counter) for negative software limit.

HSL-4XMO Axis parameter table

HSL-4XMO axis parameter table.				
NO.	Define	Description	Value	Default
00h (0)	PRA_EL_LOGIC	PEL/MEL input logic	0:Not inverse 1:Inverse	0
01h (1)	PRA_ORG_LOGIC	ORG input logic	0:Active low 1:Active high	0
02h (2)	PRA_EL_MODE	The mode of stop when end limit ON	0: Deceleration stop 1: Stop immediately	0
03h (3)	Reserved			
04h (4)	PRA_ALM_LOGIC	Set ALM Logic	0: Active low 1: Active high	0
05h(5)	Reserved			
06h(6)	PRA_EZ_LOGIC	Set EZ Logic	0: Falling edge 1: Rising edge	0
07h(7)	Reserved			
08h	PRA_SPEL_EN	Set Encorder event mode.	0: Disable 1: Reserved 2: Soft-Limit (SPEL)	0
09h(9)	PRA_SMEL_EN	Set Encorder event mode.	0: Disable 1: Reserved 2: Soft-Limit (SMEL)	0
0Ah(10)	PRA_EFB_POS0	SPEL / EFB position 0	Unit: pulse. (I32 value) Range: -10^8 ~ 10^8	100,000
0Bh(11)	PRA_EFB_POS1	SMEL / EFB position 1	Unit: pulse. (I32 value) Range: -10^8 ~ 10^8	-100,000
0Ch(12)	Reserved			
0Dh(13)	Reserved			
0Eh(14)	Reserved			
0Fh(15)	Reserved			
10h (16)	PRA_HOME_MODE	Home mode setting	Home search – 0(1st mode) to 12(13th mode) Home move – 20(1st mode) to 32(13th mode) Note: Home search (6 to 8) is reserved. If set, return error.	0
11h (17)	Reserved			
12h (18)	Reserved			
13h (19)	Reserved			
14h (20)	Reserved			
15h (21)	PRA_HOME_VM	Homing maximum velocity	Unit: pulse/sec	10000
16h (22)	Reserved			

17h (23)	Reserved			
18h (24)	PRA_HOME_EZA	Specify the EZ count up value	0000(1st count) to 1111(16th count)	0
19h (25)	PRA_HOME_VO	Homing leave home velocity – Specify FA speed	Unit: pulse/sec	100
1Ah	PRA_HOME_OFFSET	Homing leave home distance – Specify ORG offset	Unit: pulse	100
1Bh-1Fh	Reserved			
20h (32)	PRA_CURVE	Acceleration / Deceleration speed pattern	0: T-Curve 1: S-Curve	0
21h (33)	PRA_ACC	Acceleration rate	Unit: pulse/sec^2	1000000
22h (34)	PRA_DEC	Deceleration rate	Unit: pulse/sec^2	1000000
23h (35)	PRA_VS	Start velocity	Unit: pulse/sec	100
24h~50h	Reserved			
53h(83)	PRA_SERVO_LOGIC	SERVO output logic	0: Active low 1: Active high	0
80h(128)	PRA_PLS_IPT_MODE	Pulse input mode	0: A/B X1 1: A/B X2 2: A/B X4 3: CW/CCW	0
81h(129)	PRA_PLS_OPT_MODE	Pulse output mode	0: OUT/DIR (AL,H+) 1: OUT/DIR (AH,H+) 2: OUT/DIR (AL,L+) 3: OUT/DIR (AH,L+) 4: CW/CCW (AH) 5: CW/CCW (AL)	0
84h(132)	PRA_EGEAR	E-Gear factor = Motor Encoder resolution(112h) / Value (*1,)	Value = 1 ~ Motor Encoder resolution.	40,000
112h(274)	PRA_M_ENC_RES	Motor encoder resolution (*1,)	Unit: Pulse / rev or Pulse / mm	40,000
200h(512)	PRA_PLS_IPT_LOGIC	Pulse input logic	0: don not reverse EA/EB counting 1: reverse EA/EB counting	0
201h(513)	PRA_FEEDBACK_SRC	Select feedback source	0: Encoder counter & Absolute mode reference to Encoder counter. 1: Command counter & Absolute mode reference to Command counter. 2: Encoder counter & Absolute mode reference to Command counter.	0
210h(52	PRA_ALM_MODE	ALM mode setting	0: Immediate stop	0

8)			1: Slow down then stop	
211h(529)	PRA_INP_LOGIC	INP input logic	0: Active low 1: Active high	0
212h(530)	PRA_SD_EN	Enable SD. (*2)	0: Disable 1: Enable	0
213h(531)	PRA_SD_MODE	SD mode setting	0: Only slow down 1: Slow down and stop	0
214h(532)	PRA_SD_LOGIC	SD input logic	0: Active low 1: Active high	0
215h(533)	PRA_SD_LATCH	Latch SD input	0: Disable latch function 1: Enable latch function	0
216h(534)	PRA_ERC_MODE	ERC mode setting	0: disable 1: output ERC when stopped by EL, ALM, or EMG input 2: output ERC when complete home return 3: both 1 and 2	3
217h(535)	PRA_ERC_LOGIC	ERC output logic	0: Active low 1: Active high	0
218h(536)	PRA_ERC_LEN	Pulse width of ERC setting	0: 12 us 1: 102 us 2: 409 us 3: 1.6 ms 4: 13 ms 5: 52 ms 6: 104 ms 7: Level Output	3
219h(537)	Reserved			
21Ah(538)	Reserved			
21Bh(539)	Reserved			
21Ch	Reserved			
21Dh	PRA_LTC_LOGIC	LTC input logic	0: Falling edge 1: Rising edge	0
21Eh~220h	Reserved			
221h	PRA_ COMPENSATION _PULSE	A backlash or slip correction amount	0 to 4095	0
222h	PRA_ COMPENSATION _MODE	Backlash or slip mode setting	0: Disable 1: Backlash correction 2: Slip correction	0

223h	PRA_LTC_SRC	Select latch source	0: LTC pin 1: ORG pin 2: GCMP ON	0
224h	PRA_LTC_DEST	Select latch target	0: Command counter 1: Position counter	0
225h	PRA_LTC_DATA	Get latch data **(Read only)**	Pulse (28-bit signed)	0
226h	PRA_GCMP_EN	Genernal comparator enable & set method	0: Disable Other: Enable 1:data = cmp counter (regardless of counting direction) 2: data=cmp counter (while counting up) 3: data=cmp counter (while counting down) 4: data>cmp counter 5: data<cmp counter	0
227h	PRA_GCMP_POS	Set/Get general comparator position	Pulse (28-bit signed)	0
228h	PRA_GCMP_SRC	Select general comparator source	0: Command counter 1: Position counter	0
229h	PRA_GCMP_ACTION	Select action when GCMP are met.	0: Do nothing 1: Immediate stop 2: Deceleration stop	0
22Ah	PRA_GCMP_STS	Check if GCMP is met **(Read only)**	0: Not meet 1: meet	0
22Bh	PRA_VIBSUP_RT	Supress vibration – Reverse Time	Unit: 1.6 us (16-bit unsigned)	0
22Ch	PRA_VIBSUP_FT	Supress vibration – Forward Time	Unit: 1.6 us (16-bit unsigned)	0
22Dh~22Fh	Reserved			
230h	Reserved			
231h	PRA_GPDI_SEL	Select gpio input – DI / LTC / SD. (*2)	0: LTC 1: SD	0
232h	Reserved			
233h(56	PRA_RDY_LOGIC	RDY input logic	0: Active high	0

3)			1: Active low	
234h~ 23Fh	Reserved			
240h(576)	PRA_SPD_LIMIT	Set Fixed Speed	Unit: pulse/sec	6553500
241h(577)	PRA_MAX_ACCDEC	Get max acceleration /deceleration which is limited by fixed speed. **(Read only)**	Unit: pulse/sec^2	245760000
242h(578)	PRA_MIN_ACCDEC	Get minimum acceleration/deceleration which is limited by fixed speed. **(Read only)**	Unit: pulse/sec^2	7500

*1: This parameter is used to calculate a move ratio. It is only effective when PRA_FEEDBACK_SRC set to 0.

*2: When PRA_GPDI_SEL set to DI/LTC, PRA_SD_EN automatically set to disable. Before PRA_SD_EN set to enable, be sure that PRA_GPDI_SEL set to SD mode.

PCI(e)-8154/8158, PCI-8102/PCI-C154(+) Axis parameter table

PCI(e)-8154/8158, PCI-8102/PCI-C154(+) axis parameter table				
NO.	Define	Description	Value	Default
00h (0)	PRA_EL_LOGIC	PEL/MEL input logic	0:Not inverse 1:Inverse	0
01h (1)	PRA_ORG_LOGIC	ORG input logic	0:Active low 1:Active high	0
02h (2)	PRA_EL_MODE	The mode of stop when end limit(include soft-limit) ON. Deceleration profile is given according to PRA_DEC	0: Deceleration stop 1: Stop immediately	0
03h (3)	PRA_MDN_CONDI	Motion done condition (Affective with motion stats NSTP bit)	0: Control command done (default) 1: Command done with INP	0
04h (4)	PRA_ALM_LOGIC	Set ALM Logic	0: Active low 1: Active high	0
05h(5)	Reserved			
06h(6)	PRA_EZ_LOGIC	Set EZ Logic	0: Falling edge 1: Rising edge	0
07h(7)	Reserved			
08h	PRA_SPEL_EN	Set Encorder event mode. (*3,)	0: Disable 1: Reserved 2: Soft-Limit (SPEL) Note: mode 1 is reserved. If set, return error.	0
09h(9)	PRA_SMEL_EN	Set Encorder event mode. (*3,)	0: Disable 1: Reserved 2: Soft-Limit (SMEL) Note: mode 1 is reserved. If set, return error.	0
0Ah(10)	PRA_EFB_POS0	SPEL / EFB position 0 (*3,)	Unit: pulse. (I32 value) (28-bit signed)	100,000
0Bh(11)	PRA_EFB_POS1	SMEL / EFB position 1 (*3,)	Unit: pulse. (I32 value) (28-bit signed)	-100,000
0Ch(12)	Reserved			
0Dh(13)	Reserved			
0Eh(14)	Reserved			
0Fh(15)	Reserved			
10h (16)	PRA_HOME_MODE	Home mode setting	Home search – 0(1st mode) to 12(13th mode)	0

			Home move – 20(1st mode) to 32(13th mode) Note: Home search (6 to 8) is reserved. If set, return error.	
11h (17)	PRA_HOME_DIR	Homing direction	0: positive direction 1: negative direction	0
12h (18)	Reserved			
13h (19)	Reserved			
14h (20)	Reserved			
15h (21)	PRA_HOME_VM	Homing maximum velocity	Unit: pulse/sec	10000
16h (22)	Reserved			
17h (23)	Reserved			
18h (24)	PRA_HOME_EZA	Specify the EZ count up value	0000(1st count) to 1111(16th count)	0
19h (25)	PRA_HOME_VO	Homing leave home velocity – Specify FA speed	Unit: pulse/sec	100
1Ah	PRA_HOME_OFFSET	Homing leave home distance – Specify ORG offset	Unit: pulse	100
1Bh-1Fh	Reserved			
20h (32)	PRA_CURVE	Acceleration / Deceleration speed pattern	[0.0~1.0] 0: T-Curve 1: S-Curve 0.0~1.0: SL-Curve (*7,)	0
21h (33)	PRA_ACC	Acceleration rate	Unit: pulse/sec^2(*7,)	1000000
22h (34)	PRA_DEC	Deceleration rate	Unit: pulse/sec^2(*7,)	1000000
23h (35)	PRA_VS	Start velocity	Unit: pulse/sec(*7,)	100
24h~27h	Reserved			
28h	Reserved			
29h	Reserved			
2Ah ~ 50h	Reserved			
53h(83)	PRA_SERVO_LOGIC	SERVO output logic	0: Active low 1: Active high	0
80h(128)	PRA_PLS_IPT_MODE	Pulse input mode	0: A/B X1 1: A/B X2 2: A/B X4 3: CW/CCW	0
81h(129)	PRA_PLS_OPT_MODE	Pulse output mode	0: OUT/DIR (AL,H+) 1: OUT/DIR (AH,H+) 2: OUT/DIR (AL,L+) 3: OUT/DIR (AH,L+) 4: CW/CCW (AH) 5: CW/CCW (AL) 6: AB (Out Leading) 7: AB (Out Lagging)	0
84h(132)	PRA_EGEAR	E-Gear factor = Motor Encoder	Value = 1 ~ Motor Encoder resolution.	40,000

		resolution(112h) / Value (*1,)		
112h(274)	PRA_M_ENC_RES	Motor encoder resolution (*1,)	Unit: Pulse / rev or Pulse / mm	40,000
160h(352)	PRA_PSR_IPT_MODE	Setting of manual pulser input mode from PA and PB pins	ipt_mode=0, 1X AB phase type pulse input. ipt_mode=1, 2X AB phase type pulse input. ipt_mode=2, 4X AB phase type pulse input. ipt_mode=3, CW/CCW type pulse input.	0
161h(353)	Reserved			
162h(354)	PRA_PSR_IPT_DIR	Reverse the moving direction from pulse direction	Inverse =0, no inverse Inverse =1, Reverse moving direction	0
163h(355)	PRA_PSR_RATIO_VALUE	reserved		0
164h(356)	PRA_PSR_PDV	Set manual pulser ratio for actual output pulse rate. The formula for pulser output rate is (1) When PDV = 1~2047, PMG = 0~31 Output Pulse Count = Input Pulser Count x (PMG + 1) x PDV / 2048 (2) When PDV = 0, PMG = 0~31 Output Pulse Count = Input Pulser Count x (PMG + 1)	PDV = 0~2047	0
165h(357)	PRA_PSR_PMG	Refer to description of PRA_PSR_PDV	PMG = 0~31	0
166h(358)	PRA_PSR_HOME	Specified home move	HomeType =0,	0

	_TYPE	type	Command Origin.(that means axis stops when command counter becomes ‘0’) HomeType =1, Feedback Origin.(that means axis stops when feedback counter becomes ‘0’)	
167h(359)	PRA_PSR_HOME_SPD	The maximum speed in pulser home move.	Unit: pulse/sec	1000
168h~169h	Reserved			
200h(512)	PRA_PLS_IPT_LOGIC	Pulse input logic	0: don not reverse counting direction 1: reverse counting direction	0
201h(513)	PRA_FEEDBACK_SRC	Select feedback source	0: Ext. Encoder mode Ext. Encoder counter & Absolute mode reference to Encoder counter. 1: Stepper mode Ext. Command counter & Absolute mode reference to Command counter. 2: ACServo mode Ext. Encoder counter & Absolute mode reference to Command counter.	0
	Reserved			
210h(528)	PRA_ALM_MODE	ALM mode setting	0: Immediate stop 1: Slow down then stop	0
211h(529)	PRA_INP_LOGIC	INP input logic	0: Active low 1: Active high	0
212h(530)	PRA_SD_EN	Enable SD. (*2)	0: Disable 1: Enable	0
213h(531)	PRA_SD_MODE	SD mode setting	0: Only slow down 1: Slow down and stop	0
214h(532)	PRA_SD_LOGIC	SD input logic	0: Active low 1: Active high	0
215h(533)	PRA_SD_LATCH	Latch SD input	0: Disable latch function	0

			1: Enable latch function	
216h(534)	PRA_ERC_MODE	ERC mode setting	0: disable 1: output ERC when stopped by EL, ALM, or EMG input 2: output ERC when complete home return 3: both 1 and 2	3
217h(535)	PRA_ERC_LOGIC	ERC output logic	0: Active low 1: Active high	0
218h(536)	PRA_ERC_LEN	Pulse width of ERC setting	0: 12 us 1: 102 us 2: 409 us 3: 1.6 ms 4: 13 ms 5: 52 ms 6: 104 ms 7: Level Output	3
219h(537)	PRA_RESET_COUNTER	Reset counter's value to zero when home moving be completed.	By bit setting: Bit0: Reset counter1 (command position) 0: disable 1: enable Bit1: Reset counter2 (mechanical position) 0: disable 1: enable Bit2: Reset counter3 (deflection position) 0: disable 1: enable	15
21Ah(538)	Reserved			
21Bh(539)	PRA_PLS_IPT_FLT	EA/EB Filter Enable. When a filter is applied, pulse input less than 3 CLK signal cycle long is ignored.	0: Disable 1: Enable	1
21Ch(540)	Reserved			
21Dh(541)	PRA_LTC_LOGIC	LTC input logic For single latch	0: Falling edge 1: Rising edge	0

		functions use		
21Eh(542)	PRA_IO_FILTER	Apply a filter to the PEL, MEL, SD, ORG, ALM, INP inputs. (*6)	0: Don't apply a filter 1: Apply a filter	1
21F~220	Reserved			
221h(545)	PRA_ COMPENSATION _PULSE	A backlash or slip correction amount	0 to 4095	0
222h(546)	PRA_ COMPENSATION _MODE	suppre or slip mode setting	0: Disable 1: Backlash correction 2: Slip correction	0
223h(547)	PRA_LTC_SRC	Select latch source For single latch functions use	0: LTC pin 1: ORG pin 2: GCMP ON	0
224h(548)	PRA_LTC_DEST	Select latch target	0: Command counter 1: Position counter	0
225h(549)	PRA_LTC_DATA	Get latch data **(Read only)**	Pulse (28-bit signed)	0
226h(550)	PRA_GCMP_EN	Genernal comparator enable & set method	0: Disable Other: Enable 1:data = cmp counter (regardless of counting direction) 2: data=cmp counter (while counting up) 3: data=cmp counter (while counting down) 4: data>cmp counter 5: data<cmp counter	0
227h(551)	PRA_GCMP_POS	General comparator position data	Pulse (28-bit signed)	0
228h(552)	PRA_GCMP_SRC	Select general comparator source	0: Command counter 1: Position counter	0
229h(553)	PRA_GCMP_ACTION	Select action when GCMP are met.	0: Do nothing 1: Immediate stop 2: Deceleration stop	0

22Ah(554)	PRA_GCMP_STS	Check if GCMP is met **(Read only)**	0: Not meet 1: meet	0
22Bh(555)	PRA_VIBSUP_RT	Supress vibration – Reverse Time	Unit: 1.6 us (16-bit unsigned)	0
22C(556)	PRA_VIBSUP_FT	Supress vibration – Forward Time	Unit: 1.6 us (16-bit unsigned)	0
22Dh(557)	PRA_LATCH_DATA_SPD	Choose latch data of error position or current speed for APS_get_latch_data2 , LatchNum = 2	0: Latch error position 1: Latch current speed	0
22E~22F	Reserved			
230h(560) (8154/8158 Only)	PRA_GPDO_SEL	Select DO/CMP Output mode	0: DO 1: CMP	0
231(561) (8154/8158 /PCI-C154 (+) Only)	PRA_GPDI_SEL	Select gpio input – DI / LTC / SD / PCS / CLR / EMG. (*2)	0: DI (Active-Low) 1: LTC 2: SD 3: PCS 4: CLR 5: EMG	0
231(561) (8102 Only)	PRA_GPDI_SEL	Select gpio input – CLR / LTC / SD / PCS.	0: CLR 1: LTC 2: SD 3: PCS	0
232h(562)	PRA_GPDI_LOGIC	Select gpio input logic	0: Active-Low 1: Active-High	0
233(563)	PRA_RDY_LOGIC	RDY input logic	0: Active high 1: Active low	0
234(564) (Only for PCI-C154(+))	PRA_DI_FILTER_WIDTH	DI filter width (*4)	0 : 2660 nS 1 :10340 nS 2 : 33380 nS 3 : 94820 nS 4 : 194660 nS 5 : 993380 nS	5
235(565) (Only for PCI-C154(+))	PRA_DI_FILTER_EN	DI filter enable	0: Disable 1: Enable	1
236h~ 23Fh	Reserved			

240h(576)	PRA_SPD_LIMIT	Set Fixed Speed	Unit: pulse/sec	6553500
241h(577)	PRA_MAX_ACCDEC	Get max acceleration /deceleration which is limited by fixed speed. **(Read only)**	Unit: pulse/sec^2	245760000
242h(578)	PRA_MIN_ACCDEC	Get minimum acceleration/deceleration which is limited by fixed speed. **(Read only)**	Unit: pulse/sec^2	7500
250h(592)	PRA_CONTI_MODE	Continuous Move Mode (*5)	0:Disable 1:Enable	0
251h(593)	PRA_CONTI_BUFF	Continuous Buffer **(Read only)(*5)**	0:Empty 1:Full(8102) 1~3:Buffer(8154/58/PCI-C154+)	0
270h(624)	PRA_TCMP_EN	Trigger comparator enable & set method	0: Disable Other: Enable 1: position data = cmp counter (regardless of counting direction) 2: position data = cmp counter (while counting up) 3: position data = cmp counter (while counting down) 4: position data > cmp counter 5: position data < cmp counter	0
271h(625)	PRA_TCMP_POS	Trigger comparator position data	Pulse (28-bit signed)	0
272h(626)	PRA_TCMP_SRC	Select trigger comparator source	0: Command counter 1: Position counter	0
273h(627)	PRA_TCMP_STS	Check if TCMP is met	0: Not meet	0

		(Read only)	1: meet	
274h(628)	PRA_TCMP_LOGIC	Set TCMP signal logic	0:Negative logic 1:Positive logic	1
275h(629)	PRA_TCMP_ACTION	Select action when TCMP are met.	0: Do nothing 1: Immediate stop 2: Deceleration stop	0
280h(640)	PRA_ECMP_EN	Error comparator enable & set method	0: Disable , Others Enable. 1:comparator postion data = Error counter value (regardless of counting direction), 4: comparator position data > Error counter value 5: comparator position data < Error counter value	0
281h(641)	PRA_ECMP_POS	Error comparator absolute position data.	Pulse : 0 ~ 32767	0
283h(643)	PRA_ECMP_ACTION	Select action when ECMP are met.	0: Do nothing 1: Immediate stop 2: Deceleration stop	0
284h(644)	PRA_ECMP_STS	Check if ECMP is met **(Read only)**	0: Not meet 1: meet *If meet,please reset error counter via axis parameter PRA_ERR_COUNTER, then change this meet status to 0 automaticlly.	0
290h(656)	PRA_ERR_COUNTER	Error counter value.	Pulse : -32768 ~ 32767 (16-bit)	0
2A0h(672)	PRA_PCS_EN	Enable PCS	0: Disable	0

			1: Positioning for the number of pulses stored in the PRMV, starting from the time at which the PCS input signal is turned ON	
2A1h(673)	PRA_PCS_LOGIC	PCS logic	0: Active low 1: Active high	0

*1: This parameter is used to calculate a move ratio. It is only effective when PRA_FEEDBACK_SRC was set to 0 or 2.

*2: When PRA_GPDI_SEL set to other mode such as DI/LTC, PRA_SD_EN automatically set to disable. Before PRA_SD_EN set to enable, be sure that PRA_GPDI_SEL set to SD mode.

*3: When positive or negative software limit is selected, command counter is used as the comparison counter. The comparison method is mentioned as follows:

(EFB position 0 < command counter) for positive software limit, (EFB position 1 > command counter) for negative software limit.

*4:When we set value 0 ,the DI filter width is 2660 ns(nano second).In other words the cut off frequency is $\frac{1}{10^{-9}\times 2660\times 2} = 187970\ Hz$

*5:Continuous Move Mode usage:

Continuous motion between different numbers of axes is not allowed.

```
I32 AxisNo = 0;// Axis ID
I32 Buffer = 0; //Using buffer number
I32 ContinuousMoveCount = 0; // Continuous motion executive number
ret = APS_set_axis_param(AxisNo, PRA_ACC, 100000); //Set acc
ret = APS_set_axis_param(AxisNo, PRA_DEC, 100000); //Set dec
ret = APS_set_axis_param(AxisNo, PRA_VS, 100000); //Set start velocity
ret = APS_set_axis_param(AxisNo, PRA_CONTI_MODE, 1); //enable continuous move
while(ISR==1)
    {
        ret = APS_get_axis_param(AxisNo,PRA_CONTI_BUFF,& Buffer);//get using buffer number
        if(Buffer<3)
        {
            ret = APS_relative_move(AxisNo,ContinuousMoveCount,100000);
            if (ContinuousMoveCount==1000){
                ISR=0;
                ContinuousMoveCount = 0;
                while(Buffer){
                    ret = APS_get_axis_param(AxisNo,PRA_CONTI_BUFF,& Buffer);
                }
            }
            else{
                ContinuousMoveCount++;
            }
```

```
        }
    }
```

Note : Continuous motion between different numbers of axes is not allowed.

*6: When the filter is applied, signal pulses shorter than cases below is ignored.

PCIe-8154/58:

(1) SD= 1.667ms

(2) EL=ORG= 1.79us

(3) ALM=INP= 1.43us

PCI-C154+:

(1) SD= 1.667ms

(2) EL=ORG= 16us

(3) ALM=INP= 1.43us

*7: Those parameter can set/get parameter by F64 type.

EMX-100 Axis parameter table

EMX-100 axis parameter table					
NO.	Define	Description	Value	Default	Type
00h (0)	PRA_EL_LOGIC	PEL/MEL input logic	0: not invert 1: invert	0	I32
01h (1)	PRA_ORG_LOGIC	ORG input logic	0: not invert 1: invert	0	I32
02h (2)	PRA_EL_MODE	Stop mode when EL turns on.	0: Deceleration stop 1: Stop immediately (*1)	0	I32
04h (4)	PRA_ALM_LOGIC	Set ALM Logic	0: not invert 1: invert	0	I32
06h(6)	PRA_EZ_LOGIC	Set EZ Logic	0: not invert 1: invert	0	I32
0Ah (10)	PRA_SPEL_POS	Soft-end-limit for positive end	Unit: pulse	100000	I32
0Bh (11)	PRA_SMEL_POS	Soft-end-limit for negative end	Unit: pulse	-100000	I32
10h (16)	PRA_HOME_MODE	Home mode setting	0: home mode 0 1: home mode 1 2: home mode 2 16:home mode 16	0	I32
11h (17)	PRA_HOME_DIR	Homing direction	0: positive direction 1: negative direction	0	I32
13h (19)	PRA_HOME_ACC	Home move acceleration / Deceleration rate (*2)	range 1 ~ 500,000,000 (Unit: pulse/sec2)	10000	I32
14h (20)	PRA_HOME_VS	Homing start velocity (*2)	range 1 ~ 4,000,000 (Unit: pulse/sec)	1	I32
15h (21)	PRA_HOME_VM	Homing maximum velocity	range 1 ~	1000	I32

		(*2)	4,000,000 (Unit: pulse/sec)		
18h (24)	PRA_HOME_EZA	EZ alignment enable	0: Not enable 1: Enable	0	I32
19h (25)	PRA_HOME_VO	Homing leave home velocity **(*2)**	range 1 ~ 4,000,000 (Unit: pulse/sec)	200	I32
1D (29)	PRA_HOME_EZ_DIR	Homing EZ direction	0: positive direction 1: negative direction	0	I32
1Eh(30)	PRA_HOME_SEARCH_TARGET	Home move search target *It onlys supports home mode 1 and 2	0: ORG 1: EL	0	I32
20h (32)	PRA_SF	Move acceleration / deceleration speed pattern **(*2)**	[0 ~ 10] 0: T curve 10: S curve	0	I32
21h (33)	PRA_ACC	Acceleration rate **(*2)**	range 1~ 500,000,000 (Unit: pulse/sec2)	10000000	I32
22h (34)	PRA_DEC	Deceleration rate **(*2)**	range 1~ 500,000,000 (Unit: pulse/sec2)	10000000	I32
23h (35)	PRA_VS	Start velocity **(*2)**	range 1~ 4,000,000 (Unit: pulse/sec)	1	I32
41h (65)	PRA_JG_DIR	Jog move direction	0: Positive direction 1: Negative direction	0	I32
43h (67)	PRA_JG_ACC	Jog move acceleration **(*2)**	range 1~ 500,000,000 (Unit: pulse/sec2)	2000	I32
45h (69)	PRA_JG_VM	Jog move max velocity **(*2)**	range 1~ 4,000,000 (Unit:	1000	I32

			pulse/sec)		
4Ch(76)	PRA_JG_STOP	Jog stop mode **(*2)**	0: Deceleration stop 1: stop immedately	1	I32
53h(83)	PRA_SERVO_LOGIC	SERVO output logic	0: Active low 1: Active high	0	I32
80h (128)	PRA_PLS_IPT_MODE	Pulse input mode (encoder type)	0: AB pulse & 1 edge evaluation 1: AB pulse & 2 edge evaluation 2: AB pulse & 4 edge evaluation 3: pulse+ & pulse- (CW & CCW)	0	I32
81h (129)	PRA_PLS_OPT_MODE	Pulse output mode	0: pulse+ & pulse- (CW & CCW) 1: pulse & direction 2: AB phase & 4 edge evaluation 3: AB phase & 2 edge evaluation	0	I32
87h(135)	PRA_PLS_IPT_PIN_DIR	Set DIR of encoder	0: not inverse 1: inverse	0	I32
88h(136)	PRA_PLS_OPT_PIN_DIR	Set DIR signal when AB phase and CW/CCW	0: not inverse 1: inverse	0	I32
89h(137)	PRA_RST_OPT_CHG_DIO	Set RST output mode	0: Reset alarm Mode 1: Output high Mode 2: Output low Mode	0	I32
B0h(176)	PRA_SOFT_EL_EN	Software EL enable	0: Disable 1: Enable	0	I32

B1h(177)	PRA_SOFT_EL_SRC	Select to compare encoder or command position for software limit (EL)	0: command position 1: encoder	0	I32
B2h(178)	PRA_PLS_IPT_NEG_DRIVE	Set negative driving by inverting encoder EA and EB signals	0: Positive driving 1: Negative driving	0	I32
B3h(179)	PRA_PLS_OPT_NEG_DRIVE	Set negative driving by inverting pusle out signals	0: Positive driving 1: Negative driving	0	I32
B4h(180)	PRA_PLS_OPT_DIR	Set DIR signal to determine positive or negative motion when PRA_PLS_OPT_MODE is pulse & direction type	0: positive motion if DIR is low, and vice versa 1: positive motion if DIR is high , and vice versa	1	I32
B5h(181)	PRA_TRIG_VEL_PREVENTION_EN	Enable triangle velocity profile prevention	0: disable prevention 1: enable prevention	0	I32
201h(513)	PRA_FEEDBACK_SRC	Select feedback source	0 : command position 1 : encoder	0	I32
211h(529)	PRA_INP_LOGIC	INP input logic	0: not invert 1: invert	0	I32
233h(563)	PRA_RDY_LOGIC	RDY input logic	0: not invert 1: invert	0	I32

(*1)Deceleration stop rule:

(1-1) S-factor =0, deceleration starts with specified dec, when speed reaches Vmax, the motion stops.

(1-2) S-factor ≠ 0, there are 2 cases, depending on when the deceleration stop is commanded, as shown below:

(I). During the acceleration(red section as below figure): The acceleration starts(or continues) to decrese to zero, then the deceleration starts with specified dec and S factor, the motion stops when the speed reaches VS. (Note that in order to keep the speed curve smooth, the speed is not decreased immediatly.)

(II). After reaching the maximum speed (③ or blue section as below figure 1): The deceleration starts (or continues) with specified DC and S factor, and the motion stops when the speed reaches VS.

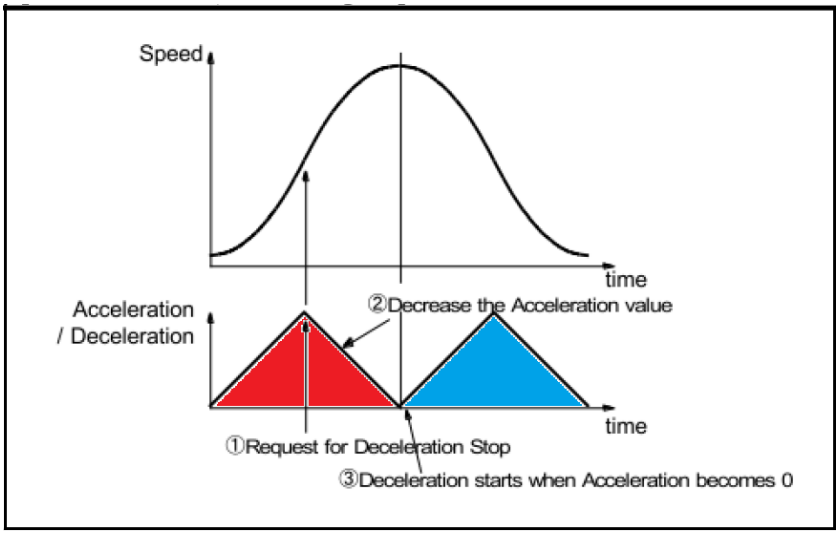


Figure 1

(*2)Speed profile criteria

S-factor	Dec	VS	DIS(pulse)
S = 0	Dec > Acc*Vmax / 8000000	VS ≧ AC½	DIS > 2*(DAC+DDE)
10 > S > 0	Any value by user configure	VS ≧ AC½	DIS > (DAC+DDE)
S = 10	Any value by user configure	VS≧ 0.1*AC*(Dec-VS)(-½)	DIS > (DAC+DDE)

DIS : Distance of motion

DAC : Distance of accelerating from VS to Vmax (red section of figure 3)

DDE : Distance of deceleration from Vmax to VS (blue section of figure 3)

Note : When distance can't meet criteria, controller will automatically start decelerating earlier and velocity will not reach Vmax. Here are two cases as below:

(a) When S = 0, as shown in Fig. 2, the acceleration stops earlier, a trapezoidal form is made so that DIS = 2 * (DAC+DDE).

(b) When S > 0, as shown in Fig. 3, to keep a smooth speed curve, the acceleration stops increasing when the number of output pulses is more than 1/12 of the total pulses. In this case, S will be set to 1, DEC will be set equal to ACC.

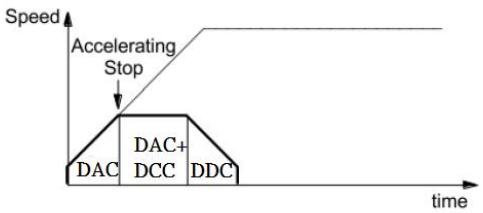


Figure 2

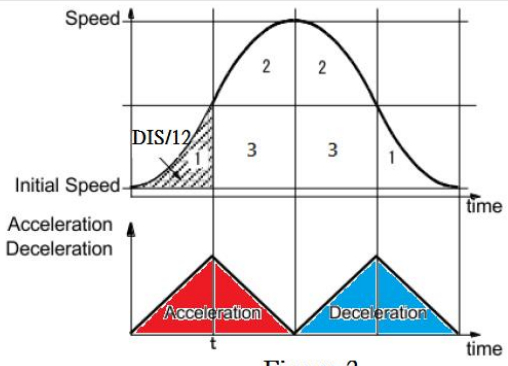


Figure 3

PCI-8254/58 / AMP-204/8C Axis parameter table

PCI-8254/58 / AMP-204/8C axis parameter table					
NO.	Define	Description	Value	Default	Type
00h (0)	PRA_EL_LOGIC	PEL/MEL input logic	0:Not inverse 1:Inverse	0	I32
01h (1)	PRA_ORG_LOGIC	ORG input logic	0:Not inverse 1:Inverse	0	I32
02h (2)	PRA_EL_MODE	Stop mode when EL turns on. Note: Deceleration profile is given according to PRA_SD_DEC.	0: Deceleration stop 1: Stop immediately	0	I32
03h (3)	PRA_MDM_CONDI	Motion done condition (Affective with motion stats NSTP bit)	0: Command done; 1: Command done with INP	0	I32
04h (4)	PRA_ALM_LOGIC	Set ALM logic	0: Low active 1: High active	0	I32
06h (6)	PRA_EZ_LOGIC	Set EZ logic	0: Low active 1: High active	0	I32
07h (7)	PRA_SD_DEC	Stop deceleration including EL stop, stop function and multi-stop().	Unit: pulse/sec2	100000000.0	F64
08h (8)	PRA_SPEL_EN	Soft PEL enable	0: Disable 1: Reserved 2: Soft-Limit (SPEL)	0	I32
09h (9)	PRA_SMEL_EN	Soft MEL enable	0: Disable	0	I32

			1: Reserved 2: Soft-Limit (SMEL)		
0Ah (10)	PRA_SPEL_POS	Soft-end-limit for positive end [F64]	Unit: pulse	100000.0	F64
0Bh (11)	PRA_SMEL_POS	Soft-end-limit for negative end [F64]	Unit: pulse	-100000.0	F64
10h (16)	PRA_HOME_MODE	Home mode setting	0: home mode 1 (ORG) 1: home mode 2 (EL) 2: home mode 3 (EZ) 3 ~ 6 : Reserved 7: home mode 7 (ORG, immediately stop) 8: home mode 8 (EL, immediately stop)	0	I32
11h (17)	PRA_HOME_DIR	Homing direction	0: positive direction 1: negative direction	0	I32
12h (18)	PRA_HOME_CURVE	Home move acceleration / deceleration speed pattern	[0.0 ~ 1.0] 0: T curve 1: S curve	0.5	F64
13h (19)	PRA_HOME_ACC	Home move acceleration / Deceleration rate	Unit: pulse/sec2	10000.0	F64
14h (20)	PRA_HOME_VS	Homing starting velocity	Unit: pulse/sec	0.0	F64
15h (21)	PRA_HOME_VM	Homing maximum velocity	Unit: pulse/sec	1000.0	F64
16h (22)	Reserved				
17h (23)	PRA_HOME_SHIFT	The distance shift from EZ, EL, ORG signal.	Unit: pulse	0.0	F64
18h (24)	PRA_HOME_EZA	EZ alignment enable	0: Not enable 1: Enable	0	I32
19h (25)	PRA_HOME_VO	Homing leave home velocity	Unit: pulse/sec	0.0	F64
1A (26)	Reserved				
1B (27)	PRA_HOME_POS	User defined position after homing.	Unit: pulse	0.0	F64

20h (32)	PRA_SF	move acceleration / deceleration speed pattern	[0.0 ~ 1.0] 0: T curve 1: S curve	0.0	F64
21h (33)	PRA_ACC	Acceleration rate	Unit: pulse/sec2	10000000.0	F64
22h (34)	PRA_DEC	Deceleration rate	Unit: pulse/sec2	10000000.0	F64
23h (35)	PRA_VS	Start velocity	Unit: pulse/sec	0.0	F64
24h (36)	PRA_VM	Maximum velocity	Unit: pulse/sec	1000.0	F64
25h (37)	PRA_VE	End velocity	Unit: pulse/sec	0.0	F64
2Ah (42)	PRA_PRE_EVENT_DIST	Pre-event distance	Unit: pulse	0.0	F64
2Bh (43)	PRA_POST_EVENT_DIST	Post-event distance	Unit: pulse	0.0	F64
32h(50)	PRA_PT_STP_DO_EN	Enable Do when point table stopping/pausing	0: Disable 1: Enable	0	I32
33h(51)	PRA_PT_STP_DO	Set Do value when Point table stopping	0: Set to 0 1: Set to 1	0	I32
34h(52)	PRA_PWM_OFF	Disable the specified PWM output when ASTP input signal is active.	0: Disable. No action when ASTP is active. 1: PWM_CH0 output will be disabled when ASTP is active. 2: PWM_CH1 output will be disabled when ASTP is active.	0	I32
35h(53)	PRA_DO_OFF	Set Do value when ASTP input signal is active.	0: Disable. No action when ASTP is active. Bit0~3: select DO channel. Bit8: Set Do output value when ASTP is active. (*1)	0	I32

40h (64)	PRA_JG_MODE	Jog mode	0:Continuous mode, 1:Step mode	0	I32
41h (65)	PRA_JG_DIR	Jog move direction	0: Positive direction 1: Negative direction	0	I32
42h (66)	PRA_JG_SF	Jog move acceleration / deceleration speed pattern	0 ~ 1	0.0	F64
43h (67)	PRA_JG_ACC	Jog move acceleration	value > 0	2000.0	F64
44h (68)	PRA_JG_DEC	Jog move deceleration	value > 0	2000.0	F64
45h (69)	PRA_JG_VM	Jog move max velocity	value > 0	1000.0	F64
46h (70)	PRA_JG_OFFSET	Jog offset, for step mode	value >= 0	1000.0	F64
47h (71)	PRA_JG_DELAY	Jog delay, for step mode, microsecnod	0 ~ 10,000,000 microsecnod	500000	I32
48h (72)	PRA_JG_MAP_DI_EN	(Enable Digital input map to jog command signal	1: Enable 0: Disable Bit 0: Active for PRA_JG_P_JOG_DI Bit 1: Active for PRA_JG_N_JOG_DI Bit 2: Active for PRA_JG_JOG_DI	0	I32
49h (73)	PRA_JG_P_JOG_DI	(I32) Mapping configuration for positive jog and digital input.	DI Channel 0 ~ 23	0	I32
4Ah (74)	PRA_JG_N_JOG_DI	(I32) Mapping configuration for negative jog and digital input.	DI Channel 0 ~ 23	1	I32
4Bh (75)	PRA_JG_JOG_DI	(I32) Mapping configuration for jog and digital input.	DI Channel 0 ~ 23	2	I32
51h (81)	PRA_SINP_WDW	Soft-INP window, unit (pulse count)	0: Disable 1~2147483647: Enable INP window	0	I32
52h (82)	PRA_SINP_STBT	Soft-INP stable time, unit	[0~10000] ms	0	I32

		(mill-second)			
60h (96)	PRA_GEAR_MASTER	Select gearing master: [0h~7h, 20h~27h] 0x00~0x07: Axis 0~7 command position deviation 0x20~0x27: Axis 0~7 feedback position deviation NOTE: 1. Setting master and slave in same axis is not allowed. For example, APS_set_axis_param(0 , 0x60, 0); 2. Relationship between master axis and slave axis can not become a loop. For example, APS_set_axis_param(0 , 0x60, 1); APS_set_axis_param(1 , 0x60, 0);	0x00~0x07: Axis 0~7 command position deviation 0x20~0x27: Axis 0~7 feedback position deviation	0x00	I32
61h (97)	PRA_GEAR_ENGAGE_RATE	Gear engage rate unit = 1 / sec, [>= 0.0] (*) When cycle time changed, this parameter must be reset. (Cycle time = 1ms)		1000.0	F64
62h (98)	PRA_GEAR_RATIO	Gear ratio	[-10000.0 ~ 10000.0]	1.0	F64
63h (99)	PRA_GANTRY_PROTECT_1	E-gear gantry mode protection level 1 [>= 0.0] | fbk_master – fbk_slave | >= value, sd-stop motion	0.0: disable gantry error check. >0.0: Enable gantry error check.	0.0	F64
64h (100)	PRA_GANTRY_PROTECT_2	E-gear gantry mode protection level 2 [>= 0.0] | fbk_master – fbk_slave | >= value, both servo-off	0.0: disable gantry error check. >0.0: Enable gantry error	0.0	F64

			check,		
65h(101)	PRA_EGEAR_MASTER	Select gearing master axis	Axisid: Specified a existed axis ID from 0 to 65535 virtual axisid: 0x7fffffff	0x7fffffff	I32
66h(102)	PRA_EGEAR_SOURCE	Select gearing master source	0: command position deviation 1: feedback position deviation	0	I32
80h (128)	PRA_PLS_IPT_MODE	Pulse input mode	0 :DEC_OUT_DIR_MODE0 1 :DEC_CW_CCW_MODE0 2 :DEC_1XAB 3 :DEC_2XAB 4 :DEC_4XAB 5 :DEC_OUT_DIR_MODE1 6:DEC_OUT_DIR_MODE2 7 :DEC_OUT_DIR_MODE3 8 :DEC_CW_CCW_MODE1	0	I32
81h (129)	PRA_PLS_OPT_MODE	Pulse output mode	0x00: OUT/DIR 0x01: CW/CCW 0x02: 4xA/B Phase	0	I32
83h (131)	PRA_ENCODER_FILTER	Encoder filter		0	I32
85h (133)	PRA_ENCODER_DIR	Encoder direction		0	I32
86h (134)	PRA_POS_UNIT_FACTOR			1	F64
88h (136)	PRA_MOVE_RATIO	Move ratio		1.0	F64
90h (144)	PRA_KP_GAIN(*2)	PID controller Kp gain		0	I32

91h (145)	PRA_KI_GAIN(*2)	PID controller Ki gain		0	I32
92h (146)	PRA_KD_GAIN(*2)	PID controller Kd gain		0	I32
93h (147)	PRA_KVFF_GAIN(*2)	Velocity feed-forward gain		0	I32
9Ah (154)	PRA_KAFF_GAIN(*2)	Acceleration feedforward gain		0	I32
9Bh(155)	PRA_KP_SHIFT(*2)	Proportional control result shift	>0: left shift; <0: right shift; =0: no shift; [31 ~ -31]	-5	I32
9Ch(156)	PRA_KI_SHIFT(*2)	Integral control result shift	>0: left shift; <0: right shift; =0: no shift; [31 ~ -31]	-15	I32
9Dh(157)	PRA_KD_SHIFT(*2)	Derivative control result shift	>0: left shift; <0: right shift; =0: no shift; [31 ~ -31]	0	I32
9Eh(158)	PRA_KVFF_SHIFT(*2)	Velocity feed-forward control result shift	>0: left shift; <0: right shift; =0: no shift; [31 ~ -31]	0	I32
9Fh(159)	PRA_KAFF_SHIFT(*2)	Acceleration feed-forward control result shift	>0: left shift; <0: right shift; =0: no shift; [31 ~ -31]	0	I32
A0h(160)	PRA_PID_SHIFT(*2)	PID control result shift	>0: left shift; <0: right shift; =0: no shift; [31 ~ -31]	-5	I32
120h (288)	PRA_SERVO_V_BIAS (*2)	Servo voltage offset		0	F64
123h (291)	PRA_SERVO_V_LIMIT (*2)	Voltage saturation output limit		10.0	F64

124h (292)	PRA_ERR_POS_LEVEL	Error counter check level		90000	F64
125h (293)	PRA_SERVO_V_INVERSE(*2)	Control voltage inverse,	1: inverse. 0:Not inverse	0	
129h(297)	PRA_BKL_DIST	Backlash compensation value		0.0	F64
12Ah(298)	PRA_BKL_CNSP	Backlash compensation increment value every cycle		0.0	F64
12Bh (299)	PRA_INTEGRAL_LIMIT (*2)	Integral limit		2147483647	I32
132h (306)	PRA_BIQUAD0_A1 (*2)	Biquad filter0 coefficient A1		0	F64
133h (307)	PRA_BIQUAD0_A2 (*2)	Biquad filter0 coefficient A2		0	F64
134h (308)	PRA_BIQUAD0_B0 (*2)	Biquad filter0 coefficient B0		1	F64
135h (309)	PRA_BIQUAD0_B1 (*2)	Biquad filter0 coefficient B1		0	F64
136h (310)	PRA_BIQUAD0_B2 (*2)	Biquad filter0 coefficient B2		0	F64
137h (311)	PRA_BIQUAD0_DIV(*2)	Biquad filter0 coefficient DIVDER		1	F64
138h (312)	PRA_BIQUAD1_A1 (*2)	Biquad filter1 coefficient A1		0	F64
139h (313)	PRA_BIQUAD1_A2 (*2)	Biquad filter1 coefficient A2		0	F64
13Ah (314)	PRA_BIQUAD1_B0 (*2)	Biquad filter1 coefficient B0		1	F64
13Bh (315)	PRA_BIQUAD1_B1 (*2)	Biquad filter1 coefficient B1		0	F64
13Ch (316)	PRA_BIQUAD1_B2 (*2)	Biquad filter1 coefficient B2		0	F64
13Dh (317)	PRA_BIQUAD1_DIV(*2)	Biquad filter1 coefficient DIVDER		1	F64

160h(352)	PRA_PSR_IPT_MODE	Pulser input mode	0: 1xAB; 1: 2xAB; 2: 4xAB 3: CW/CCW 4: OUT/DIR mode;	0	I32
161h(353)	PRA_PSR_IPT_LOGIC	Pulser EA and EB logic	0: EAinv = 0, EBinv = 0 1: EAinv = 0, EBinv = 1 2: EAinv = 1, EBinv = 0 3: EAinv = 1, EBinv = 1	0	I32
162h(354)	PRA_PSR_IPT_DIR	Pulser input direction	0: not inverse; 1: inverse	0	I32
163h(355)	PRA_PSR_RATIO_VALUE	Pulser ratio	Non zero value	1	F64
168h(360)	PRA_PSR_ACC	Pulser acceleration	Unit: pulse/sec2	1000000	F64
169h(361)	PRA_PSR_JERK	Pulser jerk	Unit: pulse/sec3	10000000000	F64
211h (529)	PRA_INP_LOGIC	Set inp logic	0: Low active 1: High active	0	I32

*1: Parameter value detail description

7	6	5	4	3	2	1	Bit : 0
-	-	-	-	1~8 :DO_CH0~ DO_CH7			
15	14	13	12	11	10	9	Bit : 8
-	-	-	-	-	-	-	ON/OFF

PCIe-833x Axis parameter table

PCIe-833x axis parameter table					
NO.	Define	Description	Value	Default	Type
02h (2)	PRA_EL_MODE	Stop mode when EL turns on. Note: Deceleration profile is given according to PRA_SD_DEC.	0: Deceleration stop 1: Stop immediately	0	I32
03h (3)	PRA_MDM_CONDI	Motion done condition (Affective with motion stats NSTP bit)	0: Command done; 1: Command done with INP	0	I32
07h (7)	PRA_SD_DEC	Stop deceleration including EL stop, stop function and multi-stop().	Unit: pulse/sec2	10000000.0	F64
08h (8)	PRA_SPEL_EN	Soft PEL enable	0: Disable 1: Reserved 2: Soft-Limit (SPEL)	0	I32
09h (9)	PRA_SMEL_EN	Soft MEL enable	0: Disable 1: Reserved 2: Soft-Limit (SMEL)	0	I32
0Ah (10)	PRA_SPEL_POS	Soft-end-limit for positive end [F64]	Unit: pulse	100000.0	F64
0Bh (11)	PRA_SMEL_POS	Soft-end-limit for negative end [F64]	Unit: pulse	-100000.0	F64
10h (16)	PRA_HOME_MODE	Home mode setting *If user uses EtherCAT home mode, please be sure to configure other four parameters PRA_HOME_ACC, PRA_HOME_VM, PRA_HOME_VO, PRA_HOME_SHIFT. Except	0: home mode 0 (ORG) 1: home mode 1 (EL) 2: home mode 2 (EZ) 3: home mode 3 (Torque) 4: home mode 4 (ORG, immediately stop) 5: home mode 5	0	I32

		these four parameters as above, other home parameters did not matter with the EtherCA servo home move.	(ORG+EZ, immediately stop) 6: home mode 6 (EL+EZ, immediately stop) 7: home mode 7 (ORG, immediately stop) 8: home mode 8 (EL, immediately stop) --- Here are EtherCAT servo home modes from 872 to 1127 872 : vendor specific 873 : vendor specific ~ 1000:No homing operation required 1001: Servo home mode 1 1002: Servo home mode 2 ~ 1035: Servo home mode 35 1036: Reserved ~ 1127: Reserved Please refer to *(1) for detail mapping table description.		
11h (17)	PRA_HOME_DIR	Homing direction	0: positive direction 1: negative direction	0	I32
12h (18)	PRA_HOME_CURVE	Home move acceleration / deceleration speed pattern	[0.0 ~ 1.0] 0: T curve 1: S curve	0.5	F64
13h (19)	PRA_HOME_ACC	Home move acceleration / Deceleration rate	Unit: pulse/sec2	10000.0	F64

		Note: If IF user uses EtherCAT home mode, the value type will change to U32.			
14h (20)	PRA_HOME_VS	Homing starting velocity	Unit: pulse/sec	0.0	F64
15h (21)	PRA_HOME_VM	Homing maximum velocity Note: If IF user uses EtherCAT home mode, the value will be mapping to CiA402 Home speed (0x6099) sub-index 1 and data type will change to U32.	Unit: pulse/sec	1000.0	F64
16h (22)	Reserved				
17h (23)	PRA_HOME_SHIFT	The distance shift from EZ, EL, ORG signal. Note: IF user uses EtherCAT home mode, this parameter usage is different from home mode 0 to 3. Please refer to *(2) description. Furthermore the value type will change to U32, please input U32 type value.	Unit: pulse	0.0	F64
18h (24)	PRA_HOME_EZA	In diffenert home mode have different meaning. Mode 0~3 : EZ alignment enable Mode 4~6 : EZ pulses counter	Mode 0~3 0: Not enable 1: Enable Mode 4~6 : Counter 0 ~ 15	0	I32
19h (25)	PRA_HOME_VO	Homing leave home velocity Note: If IF user uses EtherCAT home mode, the value will be mapping to CiA402 Home speed (0x6099) sub-index 2 and data type will change to U32.	Unit: pulse/sec	200	F64
1Ah(26)	PRA_HOME_OFFS	Homing leave ORG once	Unit: pulse	0	F64

	ET	distance			
1Bh(27)	PRA_HOME_POS	User defined position after homing.	Unit: pulse	0.0	F64
1Ch(28)	PRA_HOME_TORQUE	Torque-Limit value setting for home move. About the definition and unit in this parameter, please refer to CiA 402 Object-Dictionary (index: 0x6077) in EtherCAT servo driver user manual.	Range: 0~32767	10	I32
1Dh (29)	PRA_HOME_EZ_DIR	Select EZ searching direction in home mode 0(ORG)	0: Original EZ searching direction (default); 1: Opposite EZ searching direction	0	I32
20h (32)	PRA_SF	move acceleration / deceleration speed pattern	[0.0 ~ 1.0] 0: T curve 1: S curve	0.0	F64
21h (33)	PRA_ACC	Acceleration rate	Unit: pulse/sec2	10000000.0	F64
22h (34)	PRA_DEC	Deceleration rate	Unit: pulse/sec2	10000000.0	F64
23h (35)	PRA_VS	Start velocity	Unit: pulse/sec	0.0	F64
24h (36)	PRA_VM	Maximum velocity	Unit: pulse/sec	1000.0	F64
25h (37)	PRA_VE	End velocity	Unit: pulse/sec	0.0	F64
2Ah (42)	PRA_PRE_EVENT_DIST	Pre-event distance	Unit: pulse	0.0	F64
2Bh (43)	PRA_POST_EVENT_DIST	Post-event distance	Unit: pulse	0.0	F64
40h (64)	PRA_JG_MODE	Jog mode	0:Continuous mode, 1:Step mode	0	I32
41h (65)	PRA_JG_DIR	Jog move direction	0: Positive direction 1: Negative	0	I32

			direction		
42h (66)	PRA_JG_SF	Jog move acceleration / deceleration speed pattern	0 ~ 1	0.0	F64
43h (67)	PRA_JG_ACC	Jog move acceleration	value > 0	2000.0	F64
44h (68)	PRA_JG_DEC	Jog move deceleration	value > 0	2000.0	F64
45h (69)	PRA_JG_VM	Jog move max velocity	value > 0	1000.0	F64
46h (70)	PRA_JG_OFFSET	Jog offset, for step mode	value >= 0	1000.0	F64
47h (71)	PRA_JG_DELAY	Jog delay, for step mode, microsecnod	0 ~ 10,000,000 microsecnod	500000	I32
48h (72)	PRA_JG_MAP_DI_EN	(Enable Digital input map to jog command signal	1: Enable 0: Disable Bit 0: Active for PRA_JG_P_JOG_DI Bit 1: Active for PRA_JG_N_JOG_DI Bit 2: Active for PRA_JG_JOG_DI	0	I32
49h (73)	PRA_JG_P_JOG_DI	(I32) Mapping configuration for positive jog and digital input.	DI Channel 0 ~ 3	0	I32
4Ah (74)	PRA_JG_N_JOG_DI	(I32) Mapping configuration for negative jog and digital input.	DI Channel 0 ~ 3	1	I32
4Bh (75)	PRA_JG_JOG_DI	(I32) Mapping configuration for jog and digital input.	DI Channel 0 ~ 3	2	I32
51h (81)	PRA_SINP_WDW	Soft-INP window, unit (pulse count)	0: Disable 1~2147483647: Enable INP window	0	I32
52h (82)	PRA_SINP_STBL	Soft-INP stable time, unit (milli-second)	[0~10000] ms	0	I32
61h (97)	PRA_GEAR_ENGAGE_RATE	Gear engage rate unit = 1 / sec, [>= 0.0] (*) When cycle time changed,		1000.0	F64

		this parameter must be reset. (Cycle time = 1ms)			
62h (98)	PRA_GEAR_RATIO	Gear ratio	[-10000.0 ~ 10000.0]	1.0	F64
63h (99)	PRA_GANTRY_PROTECT_1	E-gear gantry mode protection level 1 [>= 0.0] | fbk_master – fbk_slave | >= value, sd-stop motion	0.0: disable gantry error check. >0.0: Enable gantry error check.	0.0	F64
64h (100)	PRA_GANTRY_PROTECT_2	E-gear gantry mode protection level 2 [>= 0.0] | fbk_master – fbk_slave | >= value, both servo-off	0.0: disable gantry error check. >0.0: Enable gantry error check,	0.0	F64
65h(101)	PRA_EGEAR_MASTER	Select gearing master axis	Axisid: Specified a existed axis ID from 0 to 65535 virtual axisid: 0x7fffffff	0x7fffffff	I32
66h(102)	PRA_EGEAR_SOURCE	Select gearing master source	0: command position deviation 1: feedback position deviation	0	I32
86h (134)	PRA_POS_UNIT_FACTOR	1) The user expected command multiplied by PRA_POS_UNIT_FACTOR equals actual command pulse output. 2) The actual encoder pulse divided by PRA_POS_UNIT_FACTOR equals feedback pulse.		1	F64

88h (136)	PRA_MOVE_RATIO	Move ratio		1.0	F64
124h (292)	PRA_ERR_C_LEVEL	Error counter check level		0	F64
129h(297)	PRA_BKL_DIST	Total distance to compensate backlash		0.0	F64
12Ah(298)	PRA_BKL_CNSP	This value will compensate backlash every cycle until being equal to total distance		0.0	F64
144h(324)	PRA_CMD_PSF	Command Reshaping Path Smooth Factor	20~1000: cut-off frequency (rad/sec) 0: disable	0	F64
160h(352)	PRA_PSR_IPT_MODE	Pulser input mode	0: 1xAB; 1: 2xAB; 2: 4xAB 3: CW/CCW 4: OUT/DIR	0	I32
161h(353)	PRA_PSR_IPT_LOGIC	Pulser EA and EB logic	0: EAinv = 0, EBinv = 0 1: EAinv = 0, EBinv = 1 2: EAinv = 1, EBinv = 0 3: EAinv = 1, EBinv = 1	0	I32
162h(354)	PRA_PSR_IPT_DIR	Pulser input direction	0: not inverse; 1: inverse	0	I32
163h(355)	PRA_PSR_RATIO_VALUE	Pulser ratio	Non zero value	1	F64
168h(360)	PRA_PSR_ACC	Pulser acceleration	Unit: pulse/sec2	1000000	F64
169h(361)	PRA_PSR_JERK	Pulser jerk	Unit: pulse/sec3	1000000000	F64

*(1) APS & CiA402 home mode mapping table:

DATA DESCRIPTION — CiA 402 Object 6098 h : Homing method

APS mode value	CiA402 Value	Description
872 ... 999	-128 .. -1	manufacturer specific
1000	0	No homing operation required
1001 .. 1035	1..35	Methods 1 to 35 (see the functional description)
1036 .. 1127	36 .. 127	reserved

ADLINK provides user interface to use EtherCAT CiA402 standard 35 types servo home but there are some conditions could happen.

- CiA 402 defines 35 types of home mode, but vendor's servo drive only supports less than 35 types. That could make users can't access all of them.
- Some home mode behavior defined by CiA 402 is different from vendor's. Because vendor maybe does not follow all CiA402 home behaviors.
  In words, users should refer to vendor's specfication guideline before using EtherCAT servo home mode.

*(2) Home shift of EtherCAT servo home mode:

If user use EtherCA servo home mode, the parameter PRA_HOME_SHIFT (0x17) will be mapped to CiA402 object 0x607C *Home Offset.* So this is different from home mode 0 ~ 3. Furthermore, the same home offset 0x607C, there could be also different define of these vendors. Take YASKAWA and PANASONIC for examples as below talbe (2-1). Set these two servo drives EtherCAT home mode (0x6098) 1033(CiA home mode 33 search EZ) and home offset (0x607C) 10000. Before home starting, the actual position (0x6064) of them is in 3000. When finish homing, the he actual position (0x6064) is different. The YASKAWA is the same with CiA402 define, but PANASONIC has it's own difne as below chart (2-2).

Vendor	Home offset	0x6064 (Before home)	0x6064 (Home finish)
YASKAWA	10000	3000	13000
PANASONIC	10000	3000	10000

(2-1)

PANASONIC

・原点位置检出后，此位置作为基准初始化下述的对象(预置)。
6062h(Position demand value) = 6064h(Position actual value) = 607Ch(Home offset)
6063h(Position actual internal value) = 60FCh(Position demand internal value) = 0

(2-2)

ECAT-4XMO Axis parameter table

ECAT-4XMO axis parameter table					
NO.	Define	Description	Value	Default	Type
02h (2)	PRA_EL_MODE	Stop mode when EL turns on. Note: Deceleration profile is given according to PRA_SD_DEC.	0: Deceleration stop 1: Stop immediately	0	I32
03h (3)	PRA_MDM_CONDI	Motion done condition (Affective with motion stats NSTP bit)	0: Command done; 1: Command done with INP	0	I32
07h (7)	PRA_SD_DEC	Stop deceleration including EL stop, stop function and multi-stop().	Unit: pulse/sec2	10000000.0	F64
08h (8)	PRA_SPEL_EN	Soft PEL enable	0: Disable 1: Reserved 2: Soft-Limit (SPEL)	0	I32
09h (9)	PRA_SMEL_EN	Soft MEL enable	0: Disable 1: Reserved 2: Soft-Limit (SMEL)	0	I32
0Ah (10)	PRA_SPEL_POS	Soft-end-limit for positive end [F64]	Unit: pulse	100000.0	F64
0Bh (11)	PRA_SMEL_POS	Soft-end-limit for negative end [F64]	Unit: pulse	-100000.0	F64
10h (16)	PRA_HOME_MODE	Home mode setting	0: home mode 0 (ORG) 1: home mode 1 (EL) 2: home mode 2 (EZ) 3: home mode 3 (Torque) 4: home mode 4 (ORG, immediately stop) 5: home mode 5 (ORG+EZ, immediately	0	I32

			stop) 6: home mode 6 (EL+EZ, immediately stop) 7: home mode 7 (ORG, immediately stop) 8: home mode 8 (EL, immediately stop)		
11h (17)	PRA_HOME_DIR	Homing direction	0: positive direction 1: negative direction	0	I32
12h (18)	PRA_HOME_CURVE	Home move acceleration / deceleration speed pattern	[0.0 ~ 1.0] 0: T curve 1: S curve	0.5	F64
13h (19)	PRA_HOME_ACC	Home move acceleration / Deceleration rate Note: If IF user uses EtherCAT home mode, the value type will change to U32.	Unit: pulse/sec2	10000.0	F64
14h (20)	PRA_HOME_VS	Homing starting velocity	Unit: pulse/sec	0.0	F64
15h (21)	PRA_HOME_VM	Homing maximum velocity Note: If IF user uses EtherCAT home mode, the value will be mapping to CiA402 Home speed (0x6099) sub-index 1 and data type will change to U32.	Unit: pulse/sec	1000.0	F64
16h (22)	Reserved				
17h (23)	PRA_HOME_SHIFT	The distance shift from EZ, EL, ORG signal. Note: IF user uses EtherCAT home mode, this parameter usage is different from home mode 0 to 3. Please refer to *(2) description. Furthermore the	Unit: pulse	0.0	F64

		value type will change to U32, please input U32 type value.			
18h (24)	PRA_HOME_EZA	In diffenert home mode have different meaning. Mode 0~3 : EZ alignment enable Mode 4~6 : EZ pulses counter	Mode 0~3 0: Not enable 1: Enable Mode 4~6 : Counter 0 ~ 15	0	I32
19h (25)	PRA_HOME_VO	Homing leave home velocity Note: If IF user uses EtherCAT home mode, the value will be mapping to CiA402 Home speed (0x6099) sub-index 2 and data type will change to U32.	Unit: pulse/sec	200	F64
1Ah(26)	PRA_HOME_OFFSET	Homing leave ORG once distance	Unit: pulse	0	F64
1Bh(27)	PRA_HOME_POS	User defined position after homing.	Unit: pulse	0.0	F64
1Ch(28)	PRA_HOME_TORQUE	Torque-Limit value setting for home move. About the definition and unit in this parameter, please refer to CiA 402 Object-Dictionary (index: 0x6077) in EtherCAT servo driver user manual.	Range: 0~32767	10	I32
1Dh (29)	PRA_HOME_EZ_DIR	Select EZ searching direction in home mode 0(ORG)	0: Original EZ searching direction (default); 1: Opposite EZ searching direction	0	I32
20h (32)	PRA_SF	move acceleration / deceleration speed pattern	[0.0 ~ 1.0] 0: T curve 1: S curve	0.0	F64
21h (33)	PRA_ACC	Acceleration rate	Unit: pulse/sec2	10000000	F64

				.0	
22h (34)	PRA_DEC	Deceleration rate	Unit: pulse/sec2	10000000.0	F64
23h (35)	PRA_VS	Start velocity	Unit: pulse/sec	0.0	F64
24h (36)	PRA_VM	Maximum velocity	Unit: pulse/sec	1000.0	F64
25h (37)	PRA_VE	End velocity	Unit: pulse/sec	0.0	F64
2Ah (42)	PRA_PRE_EVENT_DIST	Pre-event distance	Unit: pulse	0.0	F64
2Bh (43)	PRA_POST_EVENT_DIST	Post-event distance	Unit: pulse	0.0	F64
40h (64)	PRA_JG_MODE	Jog mode	0:Continuous mode, 1:Step mode	0	I32
41h (65)	PRA_JG_DIR	Jog move direction	0: Positive direction 1: Negative direction	0	I32
42h (66)	PRA_JG_SF	Jog move acceleration / deceleration speed pattern	0 ~ 1	0.0	F64
43h (67)	PRA_JG_ACC	Jog move acceleration	value > 0	2000.0	F64
44h (68)	PRA_JG_DEC	Jog move deceleration	value > 0	2000.0	F64
45h (69)	PRA_JG_VM	Jog move max velocity	value > 0	1000.0	F64
46h (70)	PRA_JG_OFFSET	Jog offset, for step mode	value >= 0	1000.0	F64
47h (71)	PRA_JG_DELAY	Jog delay, for step mode, microsecnod	0 ~ 10,000,000 microsecnod	500000	I32
48h (72)	PRA_JG_MAP_DI_EN	(Enable Digital input map to jog command signal	1: Enable 0: Disable Bit 0: Active for PRA_JG_P_JOG_DI Bit 1: Active for PRA_JG_N_JOG_DI Bit 2: Active for PRA_JG_JOG_DI	0	I32
49h (73)	PRA_JG_P_JOG_D	(I32) Mapping configuration	DI Channel	0	I32

	I	for positive jog and digital input.	0 ~ 3		
4Ah (74)	PRA_JG_N_JOG_DI	(I32) Mapping configuration for negative jog and digital input.	DI Channel 0 ~ 3	1	I32
4Bh (75)	PRA_JG_JOG_DI	(I32) Mapping configuration for jog and digital input.	DI Channel 0 ~ 3	2	I32
51h (81)	PRA_SINP_WDW	Soft-INP window, unit (pulse count)	0: Disable 1~2147483647: Enable INP window	0	I32
52h (82)	PRA_SINP_STBL	Soft-INP stable time, unit (milli-second)	[0~10000] ms	0	I32
61h (97)	PRA_GEAR_ENGAGE_RATE	Gear engage rate unit = 1 / sec, [>= 0.0] (*) When cycle time changed, this parameter must be reset. (Cycle time = 1ms)		1000.0	F64
62h (98)	PRA_GEAR_RATIO	Gear ratio	[-10000.0 ~ 10000.0]	1.0	F64
63h (99)	PRA_GANTRY_PROTECT_1	E-gear gantry mode protection level 1 [>= 0.0] | fbk_master – fbk_slave | >= value, sd-stop motion	0.0: disable gantry error check. >0.0: Enable gantry error check.	0.0	F64
64h (100)	PRA_GANTRY_PROTECT_2	E-gear gantry mode protection level 2 [>= 0.0] | fbk_master – fbk_slave | >= value, both servo-off	0.0: disable gantry error check. >0.0: Enable gantry error check,	0.0	F64
65h(101)	PRA_EGEAR_MASTER	Select gearing master axis	Axisid: Specified a existed axis ID from 0 to 65535 virtual axisid: 0x7fffffff	0x7fffffff	I32

66h(102)	PRA_EGEAR_SOURCE	Select gearing master source	0: command position deviation 1: feedback position deviation	0	I32
86h (134)	PRA_POS_UNIT_FACTOR	1) The user expected command multiplied by PRA_POS_UNIT_FACTOR equals actual command pulse output. 2) The actual encoder pulse divided by PRA_POS_UNIT_FACTOR equals feedback pulse.		1	F64
88h (136)	PRA_MOVE_RATIO	Move ratio		1.0	F64
124h (292)	PRA_ERR_C_LEVEL	Error counter check level		0	F64
129h(297)	PRA_BKL_DIST	Total distance to compensate backlash		0.0	F64
12Ah(298)	PRA_BKL_CNSP	This value will compensate backlash every cycle until being equal to total distance		0.0	F64
144h(324)	PRA_CMD_PSF	Command Reshaping Path Smooth Factor	20~1000: cut-off frequency (rad/sec) 0: disable	0	F64
160h(352)	PRA_PSR_IPT_MODE	Pulser input mode	0: 1xAB; 1: 2xAB; 2: 4xAB 3: CW/CCW 4: OUT/DIR	0	I32
161h(353)	PRA_PSR_IPT_LOGIC	Pulser EA and EB logic	0: EAinv = 0, EBinv = 0 1: EAinv = 0, EBinv = 1	0	I32

			2: EAinv = 1, EBinv = 0 3: EAinv = 1, EBinv = 1		
162h(354)	PRA_PSR_IPT_DIR	Pulser input direction	0: not inverse; 1: inverse	0	I32
163h(355)	PRA_PSR_RATIO_VALUE	Pulser ratio	Non zero value	1	F64
168h(360)	PRA_PSR_ACC	Pulser acceleration	Unit: pulse/sec2	1000000	F64
169h(361)	PRA_PSR_JERK	Pulser jerk	Unit: pulse/sec3	1000000000	F64

*(1) APS & CiA402 home mode mapping table:

DATA DESCRIPTION — CiA 402 Object 6098 h : Homing method

APS mode value	CiA402 Value	Description
872 ... 999	-128 .. -1	manufacturer specific
1000	0	No homing operation required
1001 .. 1035	1..35	Methods 1 to 35 (see the functional description)
1036 .. 1127	36 .. 127	reserved

ADLINK provides user interface to use EtherCAT CiA402 standard 35 types servo home but there are some conditions could happen.

- CiA 402 defines 35 types of home mode, but vendor's servo drive only supports less than 35 types. That could make users can't access all of them.
- Some home mode behavior defined by CiA 402 is different from vendor's. Because vendor maybe does not follow all CiA402 home behaviors.
  In words, users should refer to vendor's specfication guideline before using EtherCAT servo home mode.

*(2) Home shift of EtherCAT servo home mode:

If user use EtherCA servo home mode, the parameter PRA_HOME_SHIFT (0x17) will be mapped to CiA402 object 0x607C *Home Offset.* So this is different from home mode 0 ~ 3. Furthermore, the same home offset 0x607C, there could be also different define of these vendors. Take YASKAWA and PANASONIC for examples as below talbe (2-1). Set these two servo drives EtherCAT home mode (0x6098) 1033(CiA home mode 33 search EZ) and home offset (0x607C) 10000. Before home starting, the actual position (0x6064) of them is in 3000. When finish homing, the he actual position (0x6064) is different. The YASKAWA is the same with CiA402 define, but PANASONIC has it's own difne as below chart (2-2).

Vendor	Home offset	0x6064 (Before home)	0x6064 (Home finish)
YASKAWA	10000	3000	13000
PANASONIC	10000	3000	10000

(2-1)

PANASONIC

・原点位置检出后，此位置作为基准初始化下述的对象(预置)。
6062h(Position demand value) = 6064h(Position actual value) = 607Ch(Home offset)
6063h(Position actual internal value) = 60FCh(Position demand internal value) = 0

(2-2)

C. Sampling parameter table

Sampling parameter table for PCI-8392(H) and PCI-8253/56 and MNET-4XMO and PCI-8254/58 / AMP-204/8C and PCIe-833x,

Sampling parameter table				
Para NO.	Define	Description	Parameter data value.	Default
00h	SAMP_PA_RATE	Sampling rate(cycle), (depended on cycle time) For 8392 and 8253/6	1~ 65535(times of cycle)	1
		Sampling rate(ms), (depended on OS Timer) For MNET-4XMO	1~5	1
02h	SAMP_PA_EDGE	Edge triggered	0:Rising edge, 1:faling edge	0
03h	SAMP_PA_LEVEL	Triggered level	(I32) -2147483648 to 2147483647	0
05h	SAMP_PA_TRIGCH	Trigger channel	0 ~ 3 (Ch0~Ch3)	0
10h	SAMP_PA_SRC_CH0	Sampling source of Channel 0	Refer to sampling source table.(*1, *2)	0
11h	SAMP_PA_SRC_CH1	Sampling source of Channel 1	Refer to sampling source table.(*1, *2)	0
12h	SAMP_PA_SRC_CH2	Sampling source of Channel 2	Refer to sampling source table.(*1, *2)	0
13h	SAMP_PA_SRC_CH3	Sampling source of Channel 3	Refer to sampling source table.(*1, *2)	0

(*1), In PCI-8392, PCI-8253/56 and MNET-4XMO, this parameter must also involve the information of axis id. Four bytes data is needed for this parameter . The first two bytes is the information of including axis id / channel id / Vao id, and the low two bytes is the type of sampling source. EX: Axis id = 150 (96h), choose source is SAMP_FBK_POS (01h). Then set parameter value is 0x00960001.

(*2), In PCI-8254/58, AMP-204/8C, PCIe-833x , the parameter must also involve the information of axis id. Two bytes data is needed for this parameter. The first byte is the type of sampling source, and the second byte is the information of including axis id / channel id / Vao id. EX: Axis id = 150 (96h), choose source is SAMP_FBK_POS (01h). Then set parameter value is 0x9601.

D. Sampling source table

Sampling source table for PCI-8392(H)

PCI-8392(H) sampling source table				
Source	Symbol Define	Description	Value range	Note
00h	SAMP_COM_POS	Command position (pulse)	I32 value	
01h	SAMP_FBK_POS	Feedback position (pulse)	I32 value	
02h	SAMP_CMD_VEL	Command velocity (pps)	I32 value	
03h	SAMP_FBK_VEL	Feedback velocity (pps)	I32 value	
04h	SAMP_MIO	motion IO status (Same as Get motion IO function)	I32 value (bit format)	
05h	SAMP_MSTS	Motion status (Same as Get motion status function)	I32 value (bit format)	
06h	SAMP_MSTS_ACC	Motion status at acceleration (Command velocity)	0: Not at acceleration 1: At acceleration	
07h	SAMP_MSTS_MV	Motion status at max velocity (Command velocity)	0: Not at max. velocity 1: At max. velocity	
08h	SAMP_MSTS_DEC	Motion status at deceleration (Command velocity)	0: Not at deceleration 1: At deceleration	
09h	SAMP_MSTS_CSTP	Motion status command stop (CSTP)	0: CSTP status ON 1: CSTP status OFF	
0Ah	SAMP_MSTS_NSTP	Motion status normal stop (NSTP)	0: NSTP status ON 1: NSTP status OFF	
0Bh	SAMP_MIO_INP	Motion status in position (INP)	0: INP status ON 1: INP status OFF	
0Ch	SAMP_MIO_ZERO	Motion status zero (ZERO)	0: ZERO status ON 1: ZERO status OFF	
0Dh	SAMP_MIO_ORG	Motion status ORG status	0: OGR status ON 1: OGR status OFF	
10h	SAMP_SSC_MON_0	SSCNET servo monitor 0	I32 value	(*1)
11h	SAMP_SSC_MON_1	SSCNET servo monitor 1	I32 value	(*1)
12h	SAMP_SSC_MON_2	SSCNET servo monitor 2	I32 value	(*1)
13h	SAMP_SSC_MON_3	SSCNET servo monitor 3	I32 value	(*1)

20h	Reserved			
21h	SAMP_GTY_DEVIATION	Gantry deviation between master and slave encoder raw data	I32 value	
22h	Reserved			
23h	SAMP_ERROR_COUNTER	Error counter data	I32 value	

(*1) Monitor data is according to monitor data source setting. Please refer to SSCNET servo monitor source table.

PCI-8253/56 sampling source table

PCI-8253/56 sampling source table				
Source	Symbol Define	Description	Value range	Note
00h	SAMP_COM_POS	Command position (pulse)	I32 value	
01h	SAMP_FBK_POS	Feedback position (pulse)	I32 value	
02h	SAMP_CMD_VEL	Command velocity (pps)	I32 value	
03h	SAMP_FBK_VEL	Feedback velocity (pps)	I32 value	
04h	SAMP_MIO	motion IO status (Same as Get motion IO function)	I32 value (bit format)	
05h	SAMP_MSTS	Motion status (Same as Get motion status function)	I32 value (bit format)	
06h	SAMP_MSTS_ACC	Motion status at acceleration (Command velocity)	0: Not at acceleration 1: At acceleration	
07h	SAMP_MSTS_MV	Motion status at max velocity (Command velocity)	0: Not at max. velocity 1: At max. velocity	
08h	SAMP_MSTS_DEC	Motion status at deceleration (Command velocity)	0: Not at deceleration 1: At deceleration	
09h	SAMP_MSTS_CSTP	Motion status command stop	0: CSTP status ON	

		(CSTP)	1: CSTP status OFF	
0Ah	SAMP_MSTS_NSTP	Motion status normal stop (NSTP)	0: NSTP status ON 1: NSTP status OFF	
0Bh	SAMP_MIO_INP	Motion status in position (INP)	0: INP status ON 1: INP status OFF	
0Ch	SAMP_MIO_ZERO	Motion status zero (ZERO)	0: ZERO status ON 1: ZERO status OFF	
0Dh	SAMP_MIO_ORG	Motion status ORG status	0: OGR status ON 1: OGR status OFF	
20h	SAMP_CONTROL_VOL	Control voltage	I32 value	
21h	SAMP_GTY_DEVIATION	Gantry deviation between master and slave encoder raw data	I32 value	
22h	SAMP_ENCODER_RAW	Encoder raw data	I32 value	
23h	SAMP_ERROR_COUNTER	Error counter data	I32 value	

MNET-4XMO sampling source table

MNET-4XMO sampling source table				
Source	Symbol Define	Description	Value range	Note
00h	SAMP_COM_POS	Command position (pulse)	I32 value	
01h	SAMP_FBK_POS	Feedback position (pulse)	I32 value	
02h	SAMP_CMD_VEL	Command velocity (pps)	I32 value	

PCI-8254/58 / AMP-204/8C sampling source table

PCI-8254/58 / AMP-204/8C sampling source table					
Source	Symbol Define	Description	Value range	type	Referred id
0x00	SAMP_COM_POS	command position	I32 value	I32	Axis id
0x01	SAMP_FBK_POS	feedback position	I32 value	I32	Axis id
0x02	SAMP_CMD_VEL	command velocity	I32 value	I32	Axis id
0x03	SAMP_FBK_VEL	feedback velocity	I32 value	I32	Axis id
0x04	SAMP_MIO	motion IO	I32 value (bit format)	I32	Axis id
0x05	SAMP_MSTS	motion status	I32 value (bit format)	I32	Axis id
0x06	SAMP_MSTS_ACC	motion status acc	0: Not at acceleration 1: At acceleration	I32	Axis id
0x07	SAMP_MSTS_MV	motion status at max velocity	0: Not at max. velocity 1: At max. velocity	I32	Axis id
0x08	SAMP_MSTS_DEC	motion status at dec	0: Not at deceleration 1: At deceleration	I32	Axis id
0x09	SAMP_MSTS_CSTP	motion status CSTP	0: CSTP status ON 1: CSTP status OFF	I32	Axis id
0x0A	SAMP_MSTS_MDN	motion status MDN	0: NSTP status ON 1: NSTP status OFF	I32	Axis id
0x0B	SAMP_MIO_INP	motion status INP	0: INP status	I32	Axis id

			ON 1: INP status OFF		
0x0D	SAMP_MIO_ORG	motion status OGR	0: OGR status ON 1: OGR status OFF	I32	Axis id
0x20	SAMP_CONTROL_VOL	Control command voltage	I32 value	I32	Axis id
0x21	SAMP_GTY_DEVIATION	Gantry deviation	I32 value	I32	Axis id
0x22	SAMP_ENCODER_RAW	Encoder raw data	I32 value	I32	Axis id
0x23	SAMP_ERROR_POS	Error position	I32 value	I32	Axis id
0x24	SAMP_PTBUFF_RUN_INDEX	Point table running index	I32 value	I32	Table id 0~1
0x10	SAMP_COM_POS_F64	Command position	F64 value	F64	Axis id
0x11	SAMP_FBK_POS_F64	Feedback position	F64 value	F64	Axis id
0x12	SAMP_CMD_VEL_F64	Command velocity	F64 value	F64	Axis id
0x13	SAMP_FBK_VEL_F64	Feedback velocity	F64 value	F64	Axis id
0x14	SAMP_CONTROL_VOL_F64	Control command voltage	F64 value	F64	Axis id
0x15	SAMP_ERR_POS_F64	Error position	F64 value	F64	Axis id
0x18	SAMP_PWM_FREQUENCY_F64	PWM frequency (Hz)	F64 value	F64	PWM CH id
0x19	SAMP_PWM_DUTY_CYCLE_F64	PWM duty cycle (%)	F64 value	F64	PWM CH id
0x1A	SAMP_PWM_WIDTH_F64	PWM width (ns)	F64 value	F64	PWM CH id
0x1B	SAMP_VAO_COMP_VEL_F64	Composed velocity for Laser	F64 value	F64	VAO id

		power control (pps)			
0x1C	SAMP_PTBUFF_COMP_VEL_F64	Composed velocity of point table	F64 value	F64	Table id 0~1
0x1D	SAMP_PTBUFF_COMP_ACC_F64	Composed acceleration of point table	F64 value	F64	Table id 0~1

PCIe-833x sampling source table

PCIe-833x sampling source table					
Source	Symbol Define	Description	Value range	type	Referred id
0x00	SAMP_COM_POS	command position	I32 value	I32	Axis id
0x01	SAMP_FBK_POS	feedback position	I32 value	I32	Axis id
0x02	SAMP_CMD_VEL	command velocity	I32 value	I32	Axis id
0x03	SAMP_FBK_VEL	feedback velocity	I32 value	I32	Axis id
0x04	SAMP_MIO	motion IO	I32 value (bit format)	I32	Axis id
0x05	SAMP_MSTS	motion status	I32 value (bit format)	I32	Axis id
0x06	SAMP_MSTS_ACC	motion status acc	0: Not at acceleration 1: At acceleration	I32	Axis id
0x07	SAMP_MSTS_MV	motion status at max velocity	0: Not at max. velocity 1: At max. velocity	I32	Axis id
0x08	SAMP_MSTS_DEC	motion status at dec	0: Not at deceleration 1: At deceleration	I32	Axis id

0x09	SAMP_MSTS_CSTP	motion status CSTP	0: CSTP status ON 1: CSTP status OFF	I32	Axis id
0x0A	SAMP_MSTS_MDN	motion status MDN	0: NSTP status ON 1: NSTP status OFF	I32	Axis id
0x0B	SAMP_MIO_INP	motion status INP	0: INP status ON 1: INP status OFF	I32	Axis id
0x0D	SAMP_MIO_ORG	motion status OGR	0: OGR status ON 1: OGR status OFF	I32	Axis id
0x21	SAMP_GTY_DEVIATION	Gantry deviation	I32 value	I32	Axis id
0x22	SAMP_ENCODER_RAW	Encoder raw data	I32 value	I32	Axis id
0x23	SAMP_ERROR_POS	Error position	I32 value	I32	Axis id
0x24	SAMP_PTBUFF_RUN_INDEX	Point table running index	I32 value	I32	Table id 0~1
0x10	SAMP_COM_POS_F64	Command position	F64 value	F64	Axis id
0x11	SAMP_FBK_POS_F64	Feedback position	F64 value	F64	Axis id
0x12	SAMP_CMD_VEL_F64	Command velocity	F64 value	F64	Axis id
0x13	SAMP_FBK_VEL_F64	Feedback velocity	F64 value	F64	Axis id
0x15	SAMP_ERR_POS_F64	Error position	F64 value	F64	Axis id
0x1C	SAMP_PTBUFF_COMP_VEL_F64	Composed velocity of point table	F64 value	F64	Table id 0~1

0x1D	SAMP_PTBUFF_COMP_ACC_F64	Composed acceleration of point table	F64 value	F64	Table id 0~1

ECAT-4XMO sampling source table

ECAT-4XMO sampling source table					
Source	Symbol Define	Description	Value range	type	Referred id
0x00	SAMP_COM_POS	command position	I32 value	I32	Axis id
0x01	SAMP_FBK_POS	feedback position	I32 value	I32	Axis id
0x02	SAMP_CMD_VEL	command velocity	I32 value	I32	Axis id
0x03	SAMP_FBK_VEL	feedback velocity	I32 value	I32	Axis id
0x04	SAMP_MIO	motion IO	I32 value (bit format)	I32	Axis id
0x05	SAMP_MSTS	motion status	I32 value (bit format)	I32	Axis id
0x06	SAMP_MSTS_ACC	motion status acc	0: Not at acceleration 1: At acceleration	I32	Axis id
0x07	SAMP_MSTS_MV	motion status at max velocity	0: Not at max. velocity 1: At max. velocity	I32	Axis id
0x08	SAMP_MSTS_DEC	motion status at dec	0: Not at deceleration 1: At deceleration	I32	Axis id
0x09	SAMP_MSTS_CSTP	motion status CSTP	0: CSTP status ON 1: CSTP status OFF	I32	Axis id
0x0A	SAMP_MSTS_MDN	motion status MDN	0: NSTP status ON 1: NSTP	I32	Axis id

			status OFF		
0x0B	SAMP_MIO_INP	motion status INP	0: INP status ON 1: INP status OFF	I32	Axis id
0x0D	SAMP_MIO_ORG	motion status OGR	0: OGR status ON 1: OGR status OFF	I32	Axis id
0x21	SAMP_GTY_DEVIATION	Gantry deviation	I32 value	I32	Axis id
0x22	SAMP_ENCODER_RAW	Encoder raw data	I32 value	I32	Axis id
0x23	SAMP_ERROR_POS	Error position	I32 value	I32	Axis id
0x24	SAMP_PTBUFF_RUN_INDEX	Point table running index	I32 value	I32	Table id 0~1
0x10	SAMP_COM_POS_F64	Command position	F64 value	F64	Axis id
0x11	SAMP_FBK_POS_F64	Feedback position	F64 value	F64	Axis id
0x12	SAMP_CMD_VEL_F64	Command velocity	F64 value	F64	Axis id
0x13	SAMP_FBK_VEL_F64	Feedback velocity	F64 value	F64	Axis id
0x15	SAMP_ERR_POS_F64	Error position	F64 value	F64	Axis id
0x1C	SAMP_PTBUFF_COMP_VEL_F64	Composed velocity of point table	F64 value	F64	Table id 0~1
0x1D	SAMP_PTBUFF_COMP_ACC_F64	Composed acceleration of point table	F64 value	F64	Table id 0~1

E. Motion IO status and motion status definitions

PCI-8392(H) motion IO status table

PCI-8392(H) motion IO status table								
Bit No	7	6	5	4	3	2	1	0
	SVON	INP	EZ	EMG	ORG	MEL	PEL	ALM
Bit No	15	14	13	12	11	10	9	8
		ABSL	TLC	SMEL	SPEL	ZSP	WARN	RDY

PCI-8253/56 motion IO status table

PCI-8253/56 motion IO status table								
Bit No	7	6	5	4	3	2	1	0
	SVON	INP	EZ	EMG	ORG	MEL	PEL	ALM
Bit No	15	14	13	12	11	10	9	8
				SMEL	SPEL	ZSP	WARN	RDY

MNET-4XMO-(C)/1XMO,HSL-4XMO,PCI(e)-8154/8158,

PCI-8102/PCI-C154(+) motion IO status table

MNET-4XMO-(C)/1MXO, HSL-4XMO motion IO status table								
Bit No	7	6	5	4	3	2	1	0
	SVON	INP	EZ	EMG	ORG	MEL	PEL	ALM

Bit No	15	14	13	12	11	10	9	8
								RDY

PCI-8144 & AMP-104C motion IO status table

PCI-8144 & AMP-104C motion IO status table								
Bit No	7	6	5	4	3	2	1	0
	--	--	--	EMG(STP)	ORG	MEL	PEL	--
Bit No	15	14	13	12	11	10	9	8
	STA	--	--	--	--	--	--	--
Bit No	23	22	21	20	19	18	17	16
	--	--	--	--	--	--	MSD	PSD

Motion IO status description table

Motion IO status description table		
Bit	Define	Description
0	ALM	Servo alarm
1	PEL	Plus end limit
2	MEL	Minus end limit
3	ORG	Original position sensor(home sensor)
4	EMG	EMG sensor
5	EZ	EZ passed
6	INP	In position
7	SVON	Servo ON
8	RDY	Ready
9	WARN	Warning
10	ZSP	Zero speed, The zero speed output range setting, please refer to the manual of servo driver.
11	SPEL	Software plus end limit

12	SMEL	Software minus end limit
13	TLC	Torque is limited by torque limit value. (When torque control is turned ON)
14	ABSL	Absolute position lost
15	STA	External start signal
16	PSD	Positive slow down signal input
17	MSD	Negative slow down signal input

EMX-100 motion IO status table

EMX-100 motion IO status table								
Bit No	7	6	5	4	3	2	1	0
	SVON	INP	EZ	EMG	ORG	MEL	PEL	ALM
Bit No	15	14	13	12	11	10	9	8
	--	--	--	--	--	--	--	RDY
Bit No	23	22	21	20	19	18	17	16
	--	--	--	--	--	--	--	--
Bit No	31	30	29	28	27	26	25	24
	--	--	--	--	--	--	--	--

EMX-100 Motion IO status description table

Motion IO status description table		
Bit	Define	Description
0	ALM	Servo alarm
1	PEL	Plus end limit
2	MEL	Minus end limit
3	ORG	Original position sensor(home sensor)
4	EMG	EMG sensor
5	EZ	EZ passed

6	INP	In position
7	SVON	Servo ON
8	RDY	Servo Ready
25~31	Reserved	Reserved, always be 0

PCI-8254/58 / AMP-204/8C motion IO status table

PCI-8254/58 motion IO status table								
Bit No	7	6	5	4	3	2	1	0
	SVON	INP	EZ	EMG	ORG	MEL	PEL	ALM
Bit No	15	14	13	12	11	10	9	8
				SMEL	SPEL	SCL		

PCI-8254/58 / AMP-204/8C Motion IO status description table

Motion IO status description table		
Bit	Define	Description
0	ALM	Servo alarm
1	PEL	Plus end limit
2	MEL	Minus end limit
3	ORG	Original position sensor(home sensor)
4	EMG	EMG sensor
5	EZ	EZ passed
6	INP	In position
7	SVON	Servo ON
8~9	Reserved	Reserved, always be 0
10	SCL	Software circular limit
11	SPEL	Software plus end limit
12	SMEL	Software minus end limit
13~	Reserved	Reserved, always be 0

PCIe-833x motion IO status table

PCIe-833x motion IO status table								
Bit No	7	6	5	4	3	2	1	0
	SVON	INP	EZ	EMG	ORG	MEL	PEL	ALM
Bit No	15	14	13	12	11	10	9	8
				SMEL	SPEL	SCL		RDY
Bit No	23	22	21	20	19	18	17	16
Bit No	31	30	29	28	27	26	25	24
								OP

PCIe-833x Motion IO status description table

Motion IO status description table		
Bit	Define	Description
0	ALM	Servo alarm
1	PEL	Plus end limit
2	MEL	Minus end limit
3	ORG	Original position sensor(home sensor)
4	EMG	EMG sensor
5	EZ	EZ passed
6	INP	In position
7	SVON	Servo ON
8	RDY	Ready
9	Reserved	Reserved, always be 0
10	SCL	Software circular limit

11	SPEL	Software plus end limit
12	SMEL	Software minus end limit
13~23	Reserved	Reserved, always be 0
24	OP	0: EtherCAT slave is offline 1: EtherCAT slave is online
25~31	Reserved	Reserved, always be 0

F. Motion status definition table

PCI-8392(H), 8253/56 Motion status definition table

Motion status definition table								
BitNo	7	6	5	4	3	2	1	0
	SMV	HMV	NSTP	DIR	DEC	ACC	VM	CSTP
BitNo	15	14	13	12	11	10	9	8
	JOG	SLV	PPS	PDW	PMV	VS	CIP	LIP
BitNo	23	22	21	20	19	18	17	16
	ECES	MELS	PELS	WANS	ALMS	EMGS	SVONS	ASTP
BitNo	31	30	29	28	27	26	25	24
	--	--	PAPB	GTM	GDCES	STPOA	SMELS	SPELS

MNET-4XMO-(C), HSL-4XMO, PCI(e)-8154/8158, PCI-8102 /PCI-C154(+) Motion status definition table

Motion status definition table								
BitNo	7	6	5	4	3	2	1	0
	SMV	HMV	NSTP	--	DEC	ACC	VM	CSTP
BitNo	15	14	13	12	11	10	9	8
	--	--	--	--	--	VS	CIP	LIP
BitNo	23	22	21	20	19	18	17	16
	--	MELS	PELS	--	ALMS	EMGS	--	ASTP
BitNo	31	30	29	28	27	26	25	24
	--	--	PAPB(PCI-C154+ only)	--	--	--	SMELS	SPELS

1XMO Motion status definition table

Motion status definition table								
BitNo	7	6	5	4	3	2	1	0
	SMV	HMV	NSTP	--	DEC	ACC	VM	CSTP
BitNo	15	14	13	12	11	10	9	8
	--	--	--	--	--	VS	--	--
BitNo	23	22	21	20	19	18	17	16
	--	MELS	PELS	--	ALMS	EMGS	--	ASTP
BitNo	31	30	29	28	27	26	25	24
	--	--	--	--	--	--	SMELS	SPELS

PCI-8144 & AMP-104C Motion status definition table

Motion status definition table								
BitNo	7	6	5	4	3	2	1	0
	--	HMV	--	DIR	DEC	ACC	--	CSTP
BitNo	15	14	13	12	11	10	9	8
	--	--	--	--	--	--	--	--
BitNo	23	22	21	20	19	18	17	16
	--	--	--	--	--	--	--	--
BitNo	31	30	29	28	27	26	25	24
	--	--	--	--	--	--	--	--

Motion Status Description Table

Motion Status Description Table		
Bit	Define	Description
0	CSTP	Command stopped
1	VM	At maximum velocity
2	ACC	At acceleration
3	DEC	At deceleration

4	DIR	Move direction. 1:Positive direction, 0:Negative direction
5	NSTP	Normal stop(Motion done)
6	HMV	In homing
7	SMV	Single axis move(relative, absolute, velocity move)
8	LIP	Linear interpolation
9	CIP	Circular interpolation
10	VS	At start velocity
11	PMV	Point table move
12	PDW	Point table dwell move
13	PPS	Point table pause state
14	SLV	Slave axis move
15	JOG	Jog move
16	ASTP	Abnormal stop
17	SVONS	Servo off stopped
18	EMGS	EMG / SEMG stopped
19	ALMS	Alarm stop
20	WANS	Warn stopped
21	PELS	PEL stopped
22	MELS	MEL stopped
23	ECES	Error counter check level reaches and stopped
24	SPELS	SPEL stopped
25	SMELS	SMEL stopped
26	STPOA	Stop by others axes
27	GDCES	Gantry deviation error level reaches and stopped
28	GTM	Gantry mode
29	PAPB	Wait for PA/PB Input
30	--	Reserved

EMX-100 Motion status definition table

Motion status definition table								
BitNo	7	6	5	4	3	2	1	0
	--	--	MDN	--	--	--	--	--
BitNo	15	14	13	12	11	10	9	8
	--	--	--	--	--	--	--	--
BitNo	23	22	21	20	19	18	17	16

	--	MELS	PELS	--	ALMS	EMGS	--	--
BitNo	31	30	29	28	27	26	25	24
	--	ORGS	HMES	EZS	--	--	SMELS	SPELS

EMX-100 Motion Status Description Table

Motion Status Description Table		
Bit	Define	Description
…	Reserved	Reserved, always be 0
5	MDN	Motion is stopped; 0: In motion, 1: motion done (It could be abnormal stop)
…	Reserved	Reserved, always be 0
18	EMGS	EMG stopped
19	ALMS	Alarm stop
…	Reserved	Reserved, always be 0
21	PELS	PEL stopped
22	MELS	MEL stopped
…	Reserved	Reserved, always be 0
24	SPELS	SPEL stopped
25	SMELS	SMEL stopped
…	Reserved	Reserved, always be 0
28	EZS	EZ stopped
29	HMES	Home error stopped. If PRA_HOME_EZA enables with HOME_VO velocity before searching EZ signal the EZ signal already triggered.
30	ORGS	ORG stopped

PCI-8254/58 / AMP-204/8C Motion status definition table

Motion status definition table								
BitNo	7	6	5	4	3	2	1	0
	--	HMV	MDN	DIR	DEC	ACC	VM	CSTP
BitNo	15	14	13	12	11	10	9	8
	JOG	--	--	--	PTB	WAIT	--	--
BitNo	23	22	21	20	19	18	17	16
	--	--	--	--	POSTD	PRED	BLD	ASTP
BitNo	31	30	29	28	27	26	25	24
	--	--	--	GER	BACKLASH	--	--	--

PCI-8254/58 / AMP-204/8C Motion Status Description Table

Motion Status Description Table		
Bit	Define	Description
0	CSTP	Command stopped (But it could be in motion)
1	VM	In maximum velocity
2	ACC	In acceleration
3	DEC	In deceleration
4	DIR	Move direction. 1:Positive direction, 0:Negative direction
5	MDN	Motion done. 0: In motion, 1: Motion done (It could be abnormal stop)
6	HMV	In homing
…	Reserved	Reserved, always be 0
10	WAIT	Axis is in waiting state. (Wait move trigger)
11	PTB	Axis is in point buffer moving. (When this bit on, MDN and ASTP will be cleared)
…	Reserved	Reserved, always be 0
15	JOG	In jogging
16	ASTP	0: Stop normally, 1: abnormal stop, When axis in motion, this bit will be clear.
17	BLD	Axis (Axes) in blending moving
18	PRED	Pre-distance event, 1: event arrived. The event will be clear when axis start moving
19	POSTD	Post-distance event. 1: event arrived. The event will be clear when axis start moving
…	Reserved	Reserved, always be 0
27	BACKLASH	0: In operation; 1: IDLE
28	GER	1: In geared (This axis as slave axis and it follow a master specified in axis parameter.)
29-31	Reserved	Reserved, always be 0

PCIe-833x Motion status definition table

Motion status definition table								
BitNo	7	6	5	4	3	2	1	0
	--	HMV	MDN	DIR	DEC	ACC	VM	CSTP
BitNo	15	14	13	12	11	10	9	8
	JOG	--	--	--	PTB	WAIT	--	--
BitNo	23	22	21	20	19	18	17	16
	--	--	--	--	POSTD	PRED	BLD	ASTP
BitNo	31	30	29	28	27	26	25	24
	--	GRY	PSR	GER	BACKLASH	--	--	--

PCIe-833x Motion Status Description Table

Motion Status Description Table		
Bit	Define	Description
0	CSTP	Command stopped (But it could be in motion)
1	VM	In maximum velocity
2	ACC	In acceleration
3	DEC	In deceleration
4	DIR	Move direction. 1:Positive direction, 0:Negative direction
5	MDN	Motion done. 0: In motion, 1: Motion done (It could be abnormal stop)
6	HMV	In homing
…	Reserved	Reserved, always be 0
10	WAIT	Axis is in waiting state. (Wait move trigger)
11	PTB	Axis is in point buffer moving. (When this bit on, MDN and ASTP will be cleared)
…	Reserved	Reserved, always be 0
15	JOG	In jogging
16	ASTP	0: Stop normally, 1: abnormal stop, When axis in motion, this bit will be clear.
17	BLD	Axis (Axes) in blending moving (Only for interpolation move usage)
18	PRED	Pre-distance event, 1: event arrived. The event will be clear when axis start moving

19	POSTD	Post-distance event. 1: event arrived. The event will be clear when axis start moving
…	Reserved	Reserved, always be 0
27	BACKLASH	0: In operation; 1: IDLE
28	GER	1: In geared (This axis as slave axis and it follow a master specified in axis parameter.)
29	PSR	Pulser function status. 0: Disable, 1: Enable
30	GRY	1: When gantry mode is enabled, this axis is master and his motion status bit 30 (GRY) will be turned on. 0: When gantry mode is disable, turning this axis's motion status bit 30 (GRY) off will depends on his other slaves are in gantry mode or not.

Note

(1): IF user uses EtherCAT home mode, the motion status is avaliabe for MDN, HMV and ASTP.

(2): IF user uses EtherCAT home mode and error happened with process, the ASTP bit will be on.

G. Interrupt factor table

PCI-8392(H) Interrupt Item Definition Table

PCI-8392(H) Interrupt Item Definition Table		
Item	Description	
0	Axis 0 interrupt factors	
1	Axis 1 interrupt factors	
…	…	
15	Axis 15 interrupt factors	
16	System interrupt factors	

PCI-8392(H) Axes interrupt factors definition of Item 0~15

PCI-8392(H) Axes interrupt factors definition of Item 0~15								
BitNo	7	6	5	4	3	2	1	0
	IZERO	IWARN	IINP	IEZ	IORG	IMEL	IPEL	IALM
BitNo	15	14	13	12	11	10	9	8
	ISPEL	ITLC	IASTP	INSTP	IDEC	IACC	IVM	ICSTP
BitNo	23	22	21	20	19	18	17	16
	--	--	--	--	--	--	--	ISMEL
BitNo	31	30	29	28	27	26	25	24
	--	--	--	-	--	--	--	--

PCI-8392(H) Axes interrupt factors description table

PCI-8392(H) Axes interrupt factors description			
NO.	Define	Interrupt condition description	Note
0	IALM	Servo alarm signal turn ON	
1	IPEL	Plus end limit switch is turn ON	
2	IMEL	Minus (Negative) end limit switch turn ON	

3	IORG	Home switch turn ON	
4	IEZ / IEZP	EZ passed signal turn ON	(1)
5	IINP	In position signal turn ON	
6	IWARN	Servo warning ON	
7	IZSP	Zero speed	
8	ICSTP	Command stop	(2)
9	IVM	In maximum velocity	
10	IACC	In acceleration	
11	IDEC	In deceleration	
12	INSTP	Normal stop(Motion done)	(2)
13	IASTP	Abnormal stop	
14	ITLC	Torque limit control is turn ON	
15	ISPEL	SPEL turn ON	
16	ISMEL	SMEL turn ON	
17~	Reserved		

(1), In SSCNET system, When zero position signal(EZ) from servo driver is ON, EZP bit will ON even if EZ is turn OFF.

(2), INSTP: Axis is stopped normally. If axis is stopped abnormally such as emergency stop and Limit switch on stop etc, this interrupt factor will not be triggered. All motion action including home move which can be waited motion done by this interrupt factor.
Users can set normal stop (motion done) condition by set axis parameter function.
CSTP: Motion command is stopped, but the axis could be still in motion.

PCI-8392(H) System interrupt factors definition of item 16

PCI-8392(H) System interrupt factors definition of item 16								
BitNo	7	6	5	4	3	2	1	0
								ILNK
BitNo	15	14	13	12	11	10	9	8
BitNo	23	22	21	20	19	18	17	16
BitNo	31	30	29	28	27	26	25	24

PCI-8392(H) System interrupt factors description table

PCI-8392(H) System interrupt factors description

NO.	Define	Interrupt condition description	Note
0	ILNK	When SSCNET Link status 1->0	

PCI-8253/56 Interrupt Item Definition Table

PCI-8253/56 Interrupt Item Definition Table		
Item	Description	
0	Axis 0 interrupt factors	
1	Axis 1 interrupt factors	
…	…	
5	Axis 5 interrupt factors	

PCI-8253/56 Axes interrupt factors definition of Item 0~5

PCI-8253/56 Axes interrupt factors definition of Item 0~5								
BitNo	7	6	5	4	3	2	1	0
	IZERO	IWARN	IINP	IEZ	IORG	IMEL	IPEL	IALM
BitNo	15	14	13	12	11	10	9	8
	ISPEL	ITLC	IASTP	INSTP	IDEC	IACC	IVM	ICSTP
BitNo	23	22	21	20	19	18	17	16
	--	--	--	--	--	--	--	ISMEL
BitNo	31	30	29	28	27	26	25	24
	--	--	--	-	--	--	--	--

PCI-8253/56 Axes interrupt factors description table

PCI-8253/56 Axes interrupt factors description table			
NO.	Define	Interrupt condition description	Note
0	IALM	Servo alarm signal turn ON	
1	IPEL	Plus end limit switch is turn ON	
2	IMEL	Minus (Negative) end limit switch turn ON	
3	IORG	Home switch turn ON	
4	IEZ	EZ signal turn ON	

5	IINP	In position signal turn ON	
6	IWARN	Servo warning ON	
7	IZSP	Zero speed	
8	ICSTP	Command stop	(1)
9	IVM	In maximum velocity	
10	IACC	In acceleration	
11	IDEC	In deceleration	
12	INSTP	Normal stop(Motion done)	(1)
13	IASTP	Abnormal stop	
14	ITLC	Torque limit control is turn ON	
15	ISPEL	SPEL turn ON	
16	ISMEL	SMEL turn ON	
17~	Reserved		

(1), INSTP: Axis is stopped normally. If axis is stopped abnormally such as emergency stop and Limit switch on stop etc, this interrupt factor will not be triggered. All motion action including home move which can be waited motion done by this interrupt factor.
Users can set normal stop (motion done) condition by set axis parameter function.
CSTP: Motion command is stopped, but the axis could be still in motion.

DPAC-1000 Interrupt Item Definition Table

DPAC-1000 Interrupt factor Item definition table

Interrupt factor Item definition table		
Item	Description	
0	CPLD Interrupt	

DPAC-1000 CPLD Interrupt factor definition of Item 0

DPAC-1000 CPLD Interrupt factor definition of Item 0								
BitNo	7	6	5	4	3	2	1	0
	--	--	--	--	--	--	--	Timer
BitNo	15	14	13	12	11	10	9	8
	--	--	--	--	--	--	--	--
BitNo	23	22	21	20	19	18	17	16
	--	--	--	--	--	--	--	--
BitNo	31	30	29	28	27	26	25	24
	--	--	--	--	--	--	--	--

DPAC-3000 Interrupt Item Definition Table

DPAC-3000 Interrupt factor Item definition table

Interrupt factor Item definition table		
Item	Description	
0	CPLD Interrupt	
1	HSL Interrupt	

DPAC-3000 CPLD Interrupt factor definition of Item 0

DPAC-3000 CPLD Interrupt factor definition of Item 0								
BitNo	7	6	5	4	3	2	1	0
	--	--	--	--	--	--	--	Timer
BitNo	15	14	13	12	11	10	9	8
	--	--	--	--	--	--	--	--
BitNo	23	22	21	20	19	18	17	16
	--	--	--	--	--	--	--	--
BitNo	31	30	29	28	27	26	25	24
	--	--	--	--	--	--	--	--

DPAC-3000 HSL Interrupt factor definition of Item 1

DPAC-3000 HSL Interrupt factor definition of Item 1								
BitNo	7	6	5	4	3	2	1	0
	--	--	--	--	--	--	--	DI
BitNo	15	14	13	12	11	10	9	8
	--	--	--	--	--	--	--	--
	23	22	21	20	19	18	17	16
	--	--	--	--	--	--	--	----
	31	30	29	28	27	26	25	24

	--	--	--	--	--	--	--	--

PCI(e)-7856 Interrupt Item Definition Table

PCI(e)-7856 Interrupt factor Item definition table

Interrupt factor Item definition table		
Item	Description	
0	CPLD / FPGA Interrupt	

※ PCI-7856 using CPLD interface, PCIe-7856 using FPGA interface.

PCI(e)-7856 CPLD / FPGA Interrupt factor definition of Item 0

PCI(e)-7856 CPLD / FPGA Interrupt factor definition of Item 0								
BitNo	7	6	5	4	3	2	1	0
	--	--	--	--	--	--	--	Timer
BitNo	15	14	13	12	11	10	9	8
	--	--	--	--	--	--	--	--
BitNo	23	22	21	20	19	18	17	16
	--	--	--	--	--	--	--	--
BitNo	31	30	29	28	27	26	25	24
	--	--	--	--	--	--	--	--

PCI-8144 Interrupt Item Definition Table

PCI-8144 Interrupt factor Item definition table

PCI-8144Interrupt factor Item definition table		
Item	Description	
0	Axis0 Motion interrupt	
1	Axis1 Motion interrupt	
2	Axis2 Motion interrupt	
3	Axis3 Motion interrupt	
4	Digital input interrupt (Falling edge)	
5	Digital input interrupt (Rising edge)	

PCI-8144 Axes interrupt factors definition of Item 0~3

PCI-8144 Axes interrupt factors definition of Item 0~3								
BitNo	7	6	5	4	3	2	1	0
	--	--	--	--	--	--	--	ICSTP

PCI-8144 Axes interrupt factors description table

PCI-8144 Axes interrupt factors description table			
NO.	Define	Interrupt condition description	Note
0	ICSTP	Motion command output stop interrupt C	

PCI-8144 Digital interrupt factors definition of item 4

PCI-8144 System interrupt factors definition of item								
BitNo	7	6	5	4	3	2	1	0
	DI7_F	DI6_F	DI5_F	DI4_F	DI3_F	DI2_F	DI1_F	DI0_F

PCI-8144 Digital interrupt factors item 4 description table

PCI-8144 System interrupt factors description			
NO.	Define	Interrupt condition description	Note
0	Din_F	Digital input Channl NO.n falling edge interrupt	

PCI-8144 Digital interrupt factors definition of item 5

PCI-8144 System interrupt factors definition of item 16								
BitNo	7	6	5	4	3	2	1	0
	DI7_R	DI6_R	DI5_R	DI4_r	DI3_R	DI2_R	DI1_R	DI0_R

PCI-8144 Digital interrupt factors item 5 description table

PCI-8144 System interrupt factors description			
NO.	Define	Interrupt condition description	Note
0	Din_R	Digital input Channl NO.n Rising edge interrupt	

AMP-104C Interrupt Item Definition Table

AMP-104C Interrupt factor Item definition table

AMP-104C Interrupt factor Item definition table		
Item	Description	
0	Axis0 Motion interrupt	
1	Axis1 Motion interrupt	
2	Axis2 Motion interrupt	
3	Axis3 Motion interrupt	
4	Digital input interrupt channel 0 ~ 7 (Falling edge)	
5	Digital input interrupt channel 0 ~ 7 (Rising edge)	
6	Digital input interrupt channel 8 ~ 15 (Falling edge)	
7	Digital input interrupt channel 8 ~ 15 (Rising edge)	
8	TTL Digital input interrupt channel 0 ~ 3 (Falling edge)	
9	TTL Digital input interrupt channel 0 ~ 3 (Rising edge)	

AMP-104C Axes interrupt factors definition of Item 0~3

AMP-104C Axes interrupt factors definition of Item 0~3								
BitNo	7	6	5	4	3	2	1	0
	--	--	--	--	--	--	--	ICSTP

AMP-104C Axes interrupt factors description table

AMP-104C Axes interrupt factors description table			
NO.	Define	Interrupt condition description	Note
0	ICSTP	Motion command output stop interrupt C	

AMP-104C Digital interrupt channel 0 ~ 7 factors definition of item 4

AMP-104C System interrupt factors definition of item 4								
BitNo	7	6	5	4	3	2	1	0
	DI7_F	DI6_F	DI5_F	DI4_F	DI3_F	DI2_F	DI1_F	DI0_F

AMP-104C Digital interrupt channel 0 ~ 7 factors item 4 description table

AMP-104C System interrupt factors description			
NO.	Define	Interrupt condition description	Note
0	DIn_F	Digital input Channl NO.n falling edge interrupt	

AMP-104C Digital interrupt channel 0 ~ 7 factors definition of item 5

AMP-104C System interrupt factors definition of item 5								
BitNo	7	6	5	4	3	2	1	0
	DI7_R	DI6_R	DI5_R	DI4_R	DI3_R	DI2_R	DI1_R	DI0_R

AMP-104C Digital interrupt factors item 5 description table

AMP-104C System interrupt factors description			
NO.	Define	Interrupt condition description	Note
0	DIn_R	Digital input Channl NO.n Rising edge interrupt	

AMP-104C Digital interrupt channel 8 ~15 factors definition of item 6

AMP-104C System interrupt factors definition of item 6								
BitNo	7	6	5	4	3	2	1	0
	DI15_F	DI14_F	DI13_F	DI12_F	DI11_F	DI10_F	DI9_F	DI8_F

AMP-104C Digital interrupt channel 8 ~15 factors item 6 description table

AMP-104C System interrupt factors description			
NO.	Define	Interrupt condition description	Note
0	DIn_F	Digital input Channl NO.n falling edge interrupt	

AMP-104C Digital interrupt channel 8 ~15 factors definition of item 7

AMP-104C System interrupt factors definition of item 7								
BitNo	7	6	5	4	3	2	1	0
	DI15_R	DI14_R	DI13_R	DI12_R	DI11_R	DI10_R	DI9_R	DI8_R

AMP-104C Digital interrupt factors item 7 description table

AMP-104C System interrupt factors description			
NO.	Define	Interrupt condition description	Note
0	DIn_R	Digital input Channl NO.n Rising edge interrupt	

AMP-104C TTL Digital interrupt channel 0 ~ 3 factors definition of item 8

AMP-104C System interrupt factors definition of item 8								
BitNo	7	6	5	4	3	2	1	0
					DI3_F	DI2_F	DI1_F	DI0_F

AMP-104C TTL Digital interrupt channel 0 ~ 3 factors item 8 description table

AMP-104C System interrupt factors description			
NO.	Define	Interrupt condition description	Note
0	DIn_F	Digital input Channl NO.n falling edge interrupt	

AMP-104C TTL Digital interrupt channel 0 ~3 factors definition of item 9

AMP-104C System interrupt factors definition of item 9								
BitNo	7	6	5	4	3	2	1	0
					DI3_R	DI2_R	DI1_R	DI0_R

AMP-104C TTL Digital interrupt factors item 9 description table

AMP-104C System interrupt factors description			
NO.	Define	Interrupt condition description	Note
0	DIn_R	Digital input Channl NO.n Rising edge interrupt	

MotionNet Interrupt Item Definition Table

MotionNet Axis Motion Interrupt factor definition(4XMO(-C))

(MNET-4XMO/ MNET-4XMO-C)

4XMO(C) Axes motion interrupt factor definition								
BitNo	7	6	5	4	3	2	1	0
	IDECE	IDECS	IACCE	IACCS	(*)	(*)	(*)	INSTP
BitNo	15	14	13	12	11	10	9	8
	IORGC	(*)	ICLRC	(*)	ICOMP4	(*)	ISMEL	ISPEL
BitNo	23	22	21	20	19	18	17	16
	--	--	--	--	(*)	(*)	(*)	ISD
BitNo	31	30	29	28	27	26	25	24
	--	--	--	--	--	--	--	--

*: Reserved.

MotionNet Axes motion interrupt factors description table

(MNET-4XMO/ MNET-4XMO-C)

4XMO(C) Axes motion interrupt factors description table			
NO.	Define	Interrupt condition description	Note
0	INSTP	Normal stop	
1	Reserved	Reserved	
2	Reserved	Reserved	
3	Reserved	Reserved	
4	IACCS	Acceleration Start	
5	IACCE	Acceleration End	
6	IDECS	Deceleration Start	
7	IDECE	Deceleration End	
8	ISPEL	+soft limit	
9	ISMEL	-soft limit	
10	Reserved	Reserved	

11	ICOMP4	General comparator is ON	
12	Reserved	Reserved	
13	ICLRC	Counter is reset by CLR input	
14	Reserved	Reserved	
15	IORGC	Counter is latched by ORG input	
16	ISD	SD input turns on	
17	Reserved	Reserved	
18	Reserved	Reserved	
19	Reserved	Reserved	
20~	Reserved	Reserved(Always set to 0)	

MotionNet Axis Motion Interrupt factor definition(1XMO)

(MNET-1XMO)

1XMO Axes motion interrupt factor definition								
BitNo	7	6	5	4	3	2	1	0
	ICOMP	ISMEL	ISPEL	IDECE	IDECS	IACCE	IACCS	INSTP
BitNo	15	14	13	12	11	10	9	8
	--	--	--	(*)	ISD	IORGC	(*)	ICLRC
BitNo	23	22	21	20	19	18	17	16
	--	--	--	--	--	--	--	--
BitNo	31	30	29	28	27	26	25	24
	--	--	--	--	--	--	--	--

*: Reserved.

MotionNet Axes motion interrupt factors description table

(MNET-1XMO)

1XMO Axes motion interrupt factors description table			
NO.	Define	Interrupt condition description	Note
0	INSTP	Normal stop	
1	IACCS	Acceleration Start	

2	IACCE	Acceleration End	
3	IDECS	Deceleration Start	
4	IDECE	Deceleration End	
5	ISPEL	+soft limit	
6	ISMEL	-soft limit	
7	ICOMP	General comparator is ON	
8	ICLRC	Counter is reset by CLR input	
9	Reserved	Reserved	
10	IORGC	Counter is latched by ORG input	
11	ISD	SD input turns on	
12	Reserved	Reserved	
13~	Reserved	Reserved(Always set to 0)	

MotionNet Axis Error Interrupt factor definition(4XMO(-C))

(MNET-4XMO/ MNET-4XMO-C)

4XMO(C) Axes error interrupt factor definition								
BitNo	7	6	5	4	3	2	1	0
	EALM	EMEL	EPEL	(*)	EGCM	(*)	ENSL	EPSL
BitNo	15	14	13	12	11	10	9	8
	EPCO	EPBO	ESIP	(*)	(*)	ESD	EEMG	(*)
BitNo	23	22	21	20	19	18	17	16
	--	--	--	--	--	--	EPAB	EEAB
BitNo	31	30	29	28	27	26	25	24
	--	--	--	--	--	--	--	--

*: Reserved.

MotionNet Axes error interrupt factors description table

(MNET-4XMO/ MNET-4XMO-C)

Note that all default error factors are turned on.

4XMO(C) Axes error interrupt factors description table			
NO.	Define	Interrupt condition description	Note
0	EPSL	+Soft limit is ON and axis is stopped	

1	ENSL	-Soft limit is ON and axis is stopped	
2	Reserved		
3	EGCM	General comparator is ON and axis is stopped	
4	Reserved		
5	EPEL	+End limit is on and axis is stopped	
6	EMEL	-End limit is on and axis is stopped	
7	EALM	ALM is happened and axis is stopped	
8	Reserved		
9	EEMG	EMG is on and axis is stopped	
10	ESD	SD input is on and axis is slowed down to stop	
11	Reserved		
12	Reserved		
13	ESIP	Axis is stopped from other axis's error stop	
14	EPBO	Pulse input buffer overflow and stop	
15	EPCO	Interpolation counter overflow	
16	EEAB	Encoder input signal error but axis is not stopped	
17	EPAB	Pulse input signal error but axis is not stopped	
18~	Reserved	Reserved(Always set to 0)	

MotionNet Axis Error Interrupt factor definition(1XMO)

(MNET-1XMO)

1XMO Axes error interrupt factor definition								
BitNo	7	6	5	4	3	2	1	0
	EEMG	(*)	EALM	EMEL	EPEL	EGCM	ENSL	EPSL
BitNo	15	14	13	12	11	10	9	8
	--	EPAB	EEAB	ESOR	(*)	ESTN	EPBO	ESD
BitNo	23	22	21	20	19	18	17	16
	--	--	--	--	--	--	--	--
BitNo	31	30	29	28	27	26	25	24
	--	--	--	--	--	--	--	--

*: Reserved.

MotionNet Axes error interrupt factors description table

(MNET-1XMO)

Note that all default error factors are turned on.

1XMO Axes interrupt factors description table			
NO.	Define	Interrupt condition description	Note
0	EPSL	+Soft limit is ON and axis is stopped	
1	ENSL	-Soft limit is ON and axis is stopped	
2	EGCM	General comparator is ON and axis is stopped	
3	EPEL	+End limit is on and axis is stopped	
4	EMEL	-End limit is on and axis is stopped	
5	EALM	ALM is happened and axis is stopped	
6	Reserved		
7	EEMG	EMG is on and axis is stopped	
8	ESD	SD input is on and axis is slowed down to stop	
9	EPBO	Pulse input buffer overflow and stop	
10	ESTN	Stopped by a communication error	
11	Reserved	Reserved(Always set to 0)	
12	ESOR	Position override could not be executed	
13	EEAB	Encoder input signal error but axis is not stopped	
14	EPAB	Pulse input signal error but axis is not stopped	
15~	Reserved	Reserved(Always set to 0)	

PCI(e)-8154/8158, PCI-8102 Interrupt Item Definition Table

PCI(e)-8154 Interrupt factor Item definition table

Interrupt factor Item definition table		
Item	Description	
0	Axis0 Error interrupt	
1	Axis0 Motion interrupt	
…		
6	Axis3 Error interrupt	
7	Axis3 Motion interrupt	
…..		
9	DB-8150 interrupt	

PCI(e)-8158 Interrupt factor Item definition table

Interrupt factor Item definition table		
Item	Description	
0	Axis0 Error interrupt	
1	Axis0 Motion interrupt	
…		
14	Axis7 Error interrupt	
15	Axis7 Motion interrupt	
….		
17	DB-8150 interrupt	

PCI-8102 Interrupt factor Item definition table

Interrupt factor Item definition table		
Item	Description	
0	Axis0 Error interrupt	
1	Axis0 Motion interrupt	
2	Axis1 Error interrupt	
3	Axis1 Motion interrupt	
4	GPIO interrupt factors	

DB-8150 interrupt factors definition of Items

7	6	5	4	3	2	1	0
EZ1	EZ0	DI1	DI0	L1fin	L0fin	PWM1	PWM0
15	14	13	12	11	10	9	8
--	--	--	--	--	FIFO_full	FIFO_low	FIFO_empty
23	22	21	20	19	18	17	16
--	--	--	--	--	--	--	--
31	30	29	28	27	26	25	24
--	--	--	--	--	--	--	--

DB-8150 interrupt factors description table

DB-8150 interrupt factors description table			
NO.	Define	Interrupt condition description	Note
0	PWM0	PWM0 Trigger Out Event	
1	PWM1	PWM1 Trigger Out Event	
2	L0fin	LinearFunction0 Finish Event	
3	L1fin	LinearFunction1 Finish Event	
4	DI0	DI0 Edge Occur	
5	DI1	DI1 Edge Occur	
6	EZ0	EZ0 Edge Occur	
7	EZ1	EZ1 Edge Occur	
8	FIFO_empty	FIFO Empty event	
9	FIFO_low	FIFO Low event	
10	FIFO_full	FIFO Full event	
11~31	Reserved	Reserved	

PCI(e)-8154/8158, PCI-8102 Axes motion interrupt factors definition of Items

7	6	5	4	3	2	1	0
IDECE	IDECS	IACCE	IACCS	--	IWCOR2	INCBS	INSTP
15	14	13	12	11	10	9	8
IORGC	--	ICLRC	ICOMP5	ICOMP4	ICOMP3	ISMEL	ISPEL
23	22	21	20	19	18	17	16
--	--	--	--	ICSTA	--	--	ISD
31	30	29	28	27	26	25	24
--	--	--	--	--	--	--	--

PCI(e)-8154/8158, PCI-8102 Axes motion interrupt factors description table

PCI(e)-8154/8158, PCI-8102 Axes motion interrupt factors description table			
NO.	Define	Interrupt condition description	Note
0	INSTP	Normal stop	
1	INCBS	Next command in buffer starts	
2	IWCOR2	Command pre-register 2 is empty and new command	
3	Reserved	Reserved	
4	IACCS	Acceleration Start	
5	IACCE	Acceleration End	
6	IDECS	Deceleration Start	
7	IDECE	Deceleration End	
8	ISPEL	+soft limit	
9	ISMEL	-soft limit	
10	ICOMP3	Error comparator or comparator 3 is ON	
11	ICOMP4	General comparator is ON	
12	ICOMP5	Trigger comparator or comparator 5 is ON	
13	ICLRC	Counter is reset by CLR input	
14	Reserved	Reserved	
15	IORGC	Counter is latched by ORG input	
16	ISD	SD input turns on	
17	Reserved	Reserved	
18	Reserved	Reserved	
19	ICSTA	CSTA input or APS_start_simultaneous_move turn on	
20~	Reserved	Reserved(Always set to 0)	

PCI(e)-8154/8158, PCI-8102 Axes error interrupt definition of Items: (Return Code)

The error interrupt sources are non-maskable but the error number of situation could be get from APS_wait_error_int( )'s return code if it is not timeout.

Return Code	Interrupt condition description	Note
0	+Soft Limit is on and axis is stopped	
1	-Soft Limit is on and axis is stopped	
2	Error Comparator or comparator 3 is ON and axis is stopped	
3	General Comparator is on and axis is stopped	
4	Trigger Comparator or comparator 5 is ON and axis is	

	stopped	
5	+End Limit is on and axis is stopped	
6	-End Limit is on and axis is stopped	
7	ALM is happened and axis is stop	
8	CSTP is ON or APS_stop_simultaneous_move is on and axis is stopped	
9	CEMG is on and axis is stopped	
10	SD input is on and axis is slowed down to stop	
11	Reserved	
12	Interpolation operation error and stop	
13	Axis is stopped from other axis's error stop	
14	Pulse input buffer overflow and stop	
15	Interpolation counter overflow	
16	Encoder input signal error but axis is not stopped	
17	Pulse input signal error but axis is not stopped	
18~	Reserved	

PCI-8102 GPIO interrupt factors definition of Items

7	6	5	4	3	2	1	0
DI3 Raising	DI2 Raising	DI1 Raising	DIO Raising	DI3 Falling	DI2 Falling	DI1 Falling	DIO Falling
15	14	13	12	11	10	9	8
--	--	--	--	--	--	--	--
23	22	21	20	19	18	17	16
--	--	--	--	--	--	--	--
31	30	29	28	27	26	25	24
--	--	--	--	--	--	--	--

PCI-8102 GPIO interrupt factors description table

PCI-8102 GPIO interrupt factors description table			
NO.	Define	Interrupt condition description	Note
0	DIO Falling	DIO Falling Edge	
1	DI1 Falling	DI1 Falling Edge	
2	DI2 Falling	DI2 Falling Edge	
3	DI3 Falling	DI3 Falling Edge	

4	DIO Raising	DI0 Raising Edge	
5	DI1 Raising	DI1 Raising Edge	
6	DI2 Raising	DI2 Raising Edge	
7	DI3 Raising	DI3 Raising Edge	
8~	Reserved	Reserved	

PCI-C154(+) Interrupt Item Definition Table

PCI-C154(+) Interrupt factor Item definition table

Interrupt factor Item definition table		
Item	Description	
0	Axis0 Error interrupt	
1	Axis0 Motion interrupt	
…		
6	Axis3 Error interrupt	
7	Axis3 Motion interrupt	
8	Latch/Compare channel 0 interrupt	
…		
11	Latch/Compare channel 3 interrupt	

PCI-C154(+) Axes motion interrupt factors definition of Items

7	6	5	4	3	2	1	0
IDECE	IDECS	IACCE	IACCS	--	IWCOR2	INCBS	INSTP
15	14	13	12	11	10	9	8
IORGC	--	ICLRC	ICOMP5	ICOMP4	ICOMP3	ISMEL	ISPEL
23	22	21	20	19	18	17	16
--	--	--	--	ICSTA	--	--	ISD
31	30	29	28	27	26	25	24
--	--	--	--	--	--	--	--

PCI-C154(+) Axes motion interrupt factors description table

PCI-C154(+) Axes motion interrupt factors description table			
NO.	Define	Interrupt condition description	Note
0	INSTP	Normal stop	
1	INCBS	Next command in buffer starts	
2	IWCOR2	Command pre-register 2 is empty and new command to write.	
3	Reserved	Reserved	
4	IACCS	Acceleration Start	

5	IACCE	Acceleration End	
6	IDECS	Deceleration Start	
7	IDECE	Deceleration End	
8	ISPEL	+soft limit	
9	ISMEL	-soft limit	
10	ICOMP3	Error comparator is ON	
11	ICOMP4	General comparator is ON	
12	ICOMP5	Trigger comparator is ON	
13	ICLRC	Counter is reset by CLR input	
14	Reserved	Reserved	
15	IORGC	Counter is latched by ORG input	
16	ISD	SD input turns on	
17~18	Reserved	Reserved	
19	ICSTA	CSTA input or APS_start_simultaneous_move turn on	
20~31	Reserved	Reserved(Always set to 0)	

PCI-C154(+) Axes error interrupt definition of Items: (Return Code)

The error interrupt sources are non-maskable but the error number of situation could be get from APS_wait_error_int( )'s return code if it is not timeout.

Return Code	Interrupt condition description	Note
0	+Soft Limit is on and axis is stopped	
1	-Soft Limit is on and axis is stopped	
2	Error Comparator is on and axis is stopped	
3	General Comparator is on and axis is stopped	
4	Trigger Comparator is on and axis is stopped	
5	+End Limit is on and axis is stopped	
6	-End Limit is on and axis is stopped	
7	ALM is happened and axis is stop	
8	CSTP is ON or APS_stop_simultaneous_move is on and axis is stopped	
9	CEMG is on and axis is stopped	
10	SD input is on and axis is slowed down to stop	
11	Reserved	
12	Interpolation operation error and stop	
13	Axis is stopped from other axis's error stop	
14	Pulser input buffer overflow and stop	

15	Interpolation counter overflow	
16	Encoder input signal error but axis is not stopped	
17	Pulser input signal error but axis is not stopped	
18~	Reserved	

PCI-C154(+) Latch/Compare interrupt factors definition of Items

7	6	5	4	3	2	1	0
CMPE	CMPF	PWMO	LINF	LTCFO	LTCFL	LTCFE	LTCFF
15	14	13	12	11	10	9	8
--	--	--	--	--	--	CMPEO	CMPL
23	22	21	20	19	18	17	16
--	--	--	--	--	--	--	--
31	30	29	28	27	26	25	24
--	--	--	--	--	--	--	--

PCI-C154(+) Latch/Compare interrupt factors description table

PCI-C154+ Latch/Compare interrupt factors description table			
NO.	Define	Interrupt condition description	Note
0	LTCFF	Latch fifo is in full state	
1	LTCFE	Latch fifo is in empty state	
2	LTCFL	Latch fifo is above level state (Be equal to or greater than level)	
3	LTCFO	Latch fifo is in overflow state	
4	LINF	Linear comparator is finished.	
5	PWMO	PWM signal overlaps	
6	CMPF	Comparator is in full state	
7	CMPE	comparator fifo is in empty state	
8	CMPL	comparator fifo is below level state (Be equal to or less than level)	
9	CMPU	comparator fifo is in underflow state	
10~31	Reserved	Reserved(Always set to 0)	

PCI-8254/58 / AMP-204/8C Interrupt Item Definition Table

PCI-8254/58 / AMP-204/8C Interrupt factor Item definition table

Interrupt Item Definition Table		
Item No.	Description	
0~7	Axes interrupts For PCI-8254, Item is from 0 to 3. (4~7 is reserved.) For PCI-8254, Item is from 0 to 7.	
8	System interrupts	
9	DI – Rising edge interrupts	
10	DI- Falling edge interrupts	

PCI-8254/58 / AMP-204/8C Axes interrupt factors definition of Item0~7

PCI-8254/AMP-204C Axes interrupt factors definition of Item 0~3

PCI-8258/AMP-208C Axes interrupt factors definition of Item 0~7

Axes interrupt factors description table			
Factor No.	Define	Interrupt condition description	Note
0	IALM	Servo alarm signal turn ON	
1	IPEL	Plus end limit switch turn ON	
2	IMEL	Minus end limit switch turn ON	
3	IORG	Home switch turn ON	
4	IEZ	EZ passed signal turn ON	
5	IINP	In position	
6	IEMG	EMG signal turn ON	
7	Reserved	Reserved, always be 0	
8	ICSTP	Command stop	(*2)
9	IVM	In maximum velocity	
10	IACC	In acceleration	
11	IDEC	In deceleration	
12	IMDN	Motion done	(*1)
13	IASTP	Abnormal stop	
14	Reserved	Reserved, always be 0	
15	ISPEL	In positive soft limit	

16	ISMEL	In minus soft limit	
17	ISCL	In soft circular limit	
18	IPTB1	P(V)T buffer 1/4 empty (free size > 250)	
19	IPTB2	P(V)T buffer 1/2 empty (free size > 500)	
20	IPTB3	P(V)T buffer empty (free size = 1000)	
21~	Reserved	Reserved, always be 0	

(*1)IMDN: All motion action including home move which can be waited motion done by this interrupt factor.

Users can set normal stop (motion done) condition by set axis parameter function.

(*2)ICSTP: Motion command is stopped, but the axis could be still in motion.

PCI-8254/58 / AMP-204/8C System interrupt factors definition of Item 8

System interrupt factors description table			
Factor No.	Define	Interrupt condition description	Note
0	IEMG	Hardware emergency stop	
1	ILCF0	Hardware linear comparator 0 finish event	
2	ILCF1	Hardware linear comparator 1 finish event	
3	IFCF0	hardware FIFO comparator 0 finish event	
4	IFCF1	hardware FIFO comparator 1 finish event	
5	Reserved	Reserved	

PCI-8254/58 / AMP-204/8C DI-Rising edge interrupt factors definition of Item 9

DI interrupt factors description table			
Factor No.	Define	Interrupt condition description	Note
0 ~ 7	DI0~DI7	DI0 ~ 7 rising edge event	
8 ~ 23	TTL0~TTL15	TTL0 ~ 15 rising edge event	
24	Reserved	Reserved	

PCI-8254/58 / AMP-204/8C DI-Falling edge interrupt factors definition of Item 10

DI interrupt factors description table			
Factor No.	Define	Interrupt condition description	Note
0 ~ 7	DI0~DI7	DI0 ~ 7 falling edge event	
8 ~ 23	TTL0~TTL15	TTL0 ~ 15 falling edge event	
24	Reserved	Reserved	

PCIe-833x Interrupt Item Definition Table

PCIe-8332 Interrupt Item Definition Table

Interrupt Item Definition Table		
Item No.	Description	
0~15	Axes interrupts (16 axes)	
64	System interrupts	
65	DI – Rising edge interrupts	
66	DI- Falling edge interrupts	

PCIe-8334 Interrupt Item Definition Table

Interrupt Item Definition Table		
Item No.	Description	
0~31	Axes interrupts (32 axes)	
64	System interrupts	
65	DI – Rising edge interrupts	
66	DI- Falling edge interrupts	

PCIe-8338 Interrupt Item Definition Table

Interrupt Item Definition Table		
Item No.	Description	
0~63	Axes interrupts (64 axes)	
64	System interrupts	
65	DI – Rising edge interrupts	
66	DI- Falling edge interrupts	

PCIe-8332 Axes interrupt factors definition of Item 0~15

interrupt factors description table			
Factor No.	Define	Interrupt condition description	Note

0	IALM	Servo alarm signal turn ON	
1	IPEL	Plus end limit switch turn ON	
2	IMEL	Minus end limit switch turn ON	
3	IORG	Home switch turn ON	
4	Reserved	Reserved	
5	IINP	In position	
6	IEMG	EMG signal turn ON	
7	Reserved	Reserved, always be 0	
8	ICSTP	Command stop	(*2)
9	IVM	In maximum velocity	
10	IACC	In acceleration	
11	IDEC	In deceleration	
12	IMDN	Motion done	(*1)
13	IASTP	Abnormal stop	
14	Reserved	Reserved, always be 0	
15	ISPEL	In positive soft limit	
16	ISMEL	In minus soft limit	
17	ISCL	In soft circular limit	
18	IPTB1	P(V)T buffer 1/4 empty (free size > 25)	
19	IPTB2	P(V)T buffer 1/2 empty (free size > 50)	
20	IPTB3	P(V)T buffer empty (free size = 100)	
21~	Reserved	Reserved, always be 0	

(*1)IMDN: All motion action including home move which can be waited motion done by this interrupt factor.

Users can set normal stop (motion done) condition by set axis parameter function.

(*2)ICSTP: Motion command is stopped, but the axis could be still in motion.

PCIe-8334 Axes interrupt factors definition of Item 0~31

Axes interrupt factors description table			
Factor No.	Define	Interrupt condition description	Note
0	IALM	Servo alarm signal turn ON	
1	IPEL	Plus end limit switch turn ON	
2	IMEL	Minus end limit switch turn ON	
3	IORG	Home switch turn ON	
4	Reserved	Reserved	

5	IINP	In position	
6	IEMG	EMG signal turn ON	
7	Reserved	Reserved, always be 0	
8	ICSTP	Command stop	(*2)
9	IVM	In maximum velocity	
10	IACC	In acceleration	
11	IDEC	In deceleration	
12	IMDN	Motion done	(*1)
13	IASTP	Abnormal stop	
14	Reserved	Reserved, always be 0	
15	ISPEL	In positive soft limit	
16	ISMEL	In minus soft limit	
17	ISCL	In soft circular limit	
18	IPTB1	P(V)T buffer 1/4 empty (free size > 25)	
19	IPTB2	P(V)T buffer 1/2 empty (free size > 50)	
20	IPTB3	P(V)T buffer empty (free size = 100)	
21~	Reserved	Reserved, always be 0	

(*1)IMDN: All motion action including home move which can be waited motion done by this interrupt factor.

Users can set normal stop (motion done) condition by set axis parameter function.

(*2)ICSTP: Motion command is stopped, but the axis could be still in motion.

PCIe-8338 Axes interrupt factors definition of Item 0~63

PCIe-8338 Axes interrupt factors description table			
Factor No.	Define	Interrupt condition description	Note
0	IALM	Servo alarm signal turn ON	
1	IPEL	Plus end limit switch turn ON	
2	IMEL	Minus end limit switch turn ON	
3	IORG	Home switch turn ON	
4	Reserved	Reserved	
5	IINP	In position	
6	IEMG	EMG signal turn ON	
7	Reserved	Reserved, always be 0	
8	ICSTP	Command stop	(*2)
9	IVM	In maximum velocity	

10	IACC	In acceleration	
11	IDEC	In deceleration	
12	IMDN	Motion done	(*1)
13	IASTP	Abnormal stop	
14	Reserved	Reserved, always be 0	
15	ISPEL	In positive soft limit	
16	ISMEL	In minus soft limit	
17	ISCL	In soft circular limit	
18	IPTB1	P(V)T buffer 1/4 empty (free size > 25)	
19	IPTB2	P(V)T buffer 1/2 empty (free size > 50)	
20	IPTB3	P(V)T buffer empty (free size = 100)	
21~	Reserved	Reserved, always be 0	

(*1)IMDN: All motion action including home move which can be waited motion done by this interrupt factor.

Users can set normal stop (motion done) condition by set axis parameter function.

(*2)ICSTP: Motion command is stopped, but the axis could be still in motion.

PCIe-833x System interrupt factors definition of Item 64

System interrupt factors description table			
Factor No.	Define	Interrupt condition description	Note
0	IEMG	Hardware emergency stop	
1	IDISCONNECT	One of connecting slave disconnect.	(*1)

(*1)IDISCONNECT: Kernel supported after version “2020091701”.

PCIe-833x DI-Rising edge interrupt factors definition of Item 65

DI interrupt factors description table			
Factor No.	Define	Interrupt condition description	Note
0 ~ 3	DI0~DI3	DI0 ~ 3 rising edge event	

PCIe-833x DI-Falling edge interrupt factors definition of Item 66

DI interrupt factors description table			
Factor No.	Define	Interrupt condition description	Note
0 ~ 3	DI0~DI3	DI0 ~ 3 falling edge event	

H. Field bus parameter table

PCI-8392H HSL parameter table				
NO.	Define	Description	Value	Default
00h	PRF_COMMUNICATION_TYPE	FiledBus Communication Type	0:Half duplex 1:Full duplex	1
01h	PRF_TRANSFER_RATE	Network transfer rate.	1: 3 Mbps 2: 6 Mbps 3: 12 Mbps	2
02h	PRF_HUB_NUMBER	Total hub number.	0~7	0
03h	PRF_INITIAL_TYPE	Reset digital output to zero or not when connect the slave modules.	0: Reset digital output to zero. 1: Depend on slave state.	0
04h	PRF_CHKERRCNT_LAYER	Set the degree of checking error count	1~7	7

PCI(e)-7856 MNET parameter table				
NO.	Define	Description	Value	Default
00h	Reserved			
01h	PRF_TRANSFER_RATE	Network transfer rate.	0: 2.5Mbps 1: 5 Mbps 2: 10 Mbps 3: 20 Mbps	3
02h~	Reserved			

PCI(e)-7856 HSL parameter table				
NO.	Define	Description	Value	Default
00h	PRF_COMMUNICATION_TYPE	FiledBus Communication Type	0:Half duplex 1:Full duplex	1
01h	PRF_TRANSFER_RATE	Network transfer rate.	1: 3 Mbps 2: 6 Mbps 3: 12 Mbps	2
02h	PRF_HUB_NUMBER	Total hub number.	0~7	0
03h	PRF_INITIAL_TYPE	Reset digital output to zero or not when connect the slave modules.	0: Reset digital output to zero. 1: Depend on slave state.	0
04h	PRF_CHKERRCNT_LAYER	Set the degree of checking error count	1~7	7

DPAC-3000 MNET parameter table				
NO.	Define	Description	Value	Default
00h	Reserved			
01h	PRF_TRANSFER_RATE	Network transfer rate.	0: 2.5Mbps 1: 5 Mbps 2: 10 Mbps 3: 20 Mbps	3
02h~	Reserved			

DPAC-3000 HSL parameter table				
NO.	Define	Description	Value	Default
00h	PRF_COMMUNICATION_TYPE	FiledBus Communication Type	0:Half duplex 1:Full duplex	1
01h	PRF_TRANSFER_RATE	Network transfer rate.	1: 3 Mbps 2: 6 Mbps 3: 12 Mbps	2
02h	PRF_HUB_NUMBER	Total hub number.	0~7	0
03h	PRF_INITIAL_TYPE	Reset digital output to zero or not when connect the slave modules.	0: Reset digital output to zero. 1: Depend on slave state.	0
04h	PRF_CHKERRCNT_LAYER	Set the degree of checking error count	1~7	7

I. Gantry parameters table

PCI-8253/56 Gantry parameters table				
Para NO.	Define	Description	Parameter data value.	Default
00h	GANTRY_MODE	Enable/Disable gantry relation.	0: Disable 1: Enable	0
01h	GENTRY_DEVIATION	Set eviation protection. If deviation is over this setting, axis will be servo off.	Positive I32 value.	8,000
02h	GENTRY_DEVIATION_STP	Set deviation protection. If deviation is over this setting, axis will be stopped.	Positive I32 value.	5,000

PCI-8392(H) Gantry parameters table				
Para NO.	Define	Description	Parameter data value.	Default
00h	GANTRY_MODE	Enable/Disable gantry relation.	0: Disable 1: Enable	0
01h	GENTRY_DEVIATION	Set deviation protection. If deviation is over this setting, axis will be servo off.	Positive I32 value.	8,000
02h	GENTRY_DEVIATION_STP	Set deviation protection. If deviation is over this setting, axis will be stopped.	Positive I32 value.	5,000

J. Trigger parameter table

PCI-8253/56 Trigger parameter table

NO	Define	Description	Value	Default:
0x00	TG_LCMP0_SRC	Linear compare 0 (LCMP0) source	0 ~ 5: Encoder counter 0~5	0
0x01	TG_LCMP1_SRC	Linear compare 1 (LCMP1) source	0 ~ 5: Encoder counter 0~5	2
0x02	TG_TCMP0_SRC	Table compare 0 (TCMP0) source	0 ~ 5: Encoder counter 0~5	1
0x03	TG_TCMP1_SRC	Table compare 1 (TCMP1) source	0 ~ 5: Encoder counter 0~5	4
0x04	TG_LCMP0_EN	Linear compare 0 (LCMP0) enable	0: Disable, 1:Enable	0
0x05	TG_LCMP1_EN	Linear compare 1 (LCMP1) enable	0: Disable, 1:Enable	0
0x06	TG_TCMP0_EN	Table compare 0 (TCMP0) enable	0: Disable, 1:Enable	0
0x07	TG_TCMP1_EN	Table compare 1 (TCMP1) enable	0: Disable, 1:Enable	0
0x10	TG_TRG0_SRC	Trigger output 0 (TRG0) source	0:None 1:LCMP0 (Default) 2:LCMP1 4:FCMP0 8:FCMP1 16: TMR	1
0x11	TG_TRG1_SRC	Trigger output 1 (TRG1) source	0:None 1:LCMP0 2:LCMP1 4:FCMP0 (Default)	4

			8:FCMP1 16: TMR	
0x12	TG_TRG2_SRC	Trigger output 2 (TRG2) source (*1)	0:None 1:LCMP0 2:LCMP1 (Default) 4:FCMP0 8:FCMP1 16: TMR	2
0x13	TG_TRG3_SRC	Trigger output 3 (TRG3) source (*1)	0:None 1:LCMP0 2:LCMP1 4:FCMP0 8:FCMP1 (Default) 16: TMR	8
0x14	TG_TRG0_PWD	TRG0 pulse width	Pulse Width = (N+ 2) * 20 ns 24 bit value. 0~ 16777215	0 (40 ns)
0x15	TG_TRG1_PWD	TRG1 pulse width	Pulse Width = (N+ 2) * 20 ns 24 bit value. 0~ 16777215	0 (40 ns)
0x16	TG_TRG2_PWD	TRG2 pulse width (*1)	Pulse Width = (N+ 2) * 20 ns 24 bit value. 0~ 16777215	0 (40 ns)
0x17	TG_TRG3_PWD	TRG3 pulse width (*1)	Pulse Width = (N+ 2) * 20 ns 24 bit value. 0~ 16777215	0 (40 ns)
0x18	TG_TRG0_CFG	TRG 0 configuration	Bit 0: Pulse logic inverse. Bit 1: pulse (0) / toggle (1) Bit 2~31: Reserved (set 0)	0
0x19	TG_TRG1_CFG	TRG 1 configuration	Bit 0: Pulse logic inverse. Bit 1: pulse (0) / toggle (1)	0

			Bit 2~31: Reserved (set 0)	
0x1A	TG_TRG2_CFG	TRG 2 configuration (*1)	Bit 0: Pulse logic inverse. Bit 1: pulse (0) / toggle (1) Bit 2~31: Reserved (set 0)	0
0x1B	TG_TRG3_CFG	TRG 3 configuration (*1)	Bit 0: Pulse logic inverse. Bit 1: pulse (0) / toggle (1) Bit 2~31: Reserved (set 0)	0
0x20	TMR_ITV	Timer Interval	Timer Interval = (N+2) * 20 ns 28 bit value. 0~ 268435455	0 (40 ns)
0x21	TMR_EN	Timer enable	0: Disable, 1:Enable	0

*1: PCI-8256 only.

MNET-4XMO-C Trigger parameter table

NO	Define	Description	Value	Default
0x00	TG_CMP0_SRC	Compare 0 source	0: Command counter 1: Position counter	0
0x01	TG_CMP1_SRC	Compare 1 source	0: Command counter 1: Position counter	0
0x02	TG_CMP2_SRC	Compare 2 source	0: Command counter 1: Position counter	0
0x03	TG_CMP3_SRC	Compare 3 source	0: Comand counter 1: Position counter	0
0x04	TG_CMP0_EN	Compare 0 enable	0: Disable Other: Enable. 1:data = cmp counter (regardless of counting direction) 2: data = cmp counter (while counting up) 3: data = cmp counter (while counting down) 4: data > cmp counter 5: data < cmp counter	0
0x05	TG_CMP1_EN	Compare 1 enable	0: Disable Other: Enable. 1:data = cmp counter (regardless of counting direction) 2: data = cmp counter (while counting up) 3: data = cmp counter (while counting down) 4: data > cmp counter 5: data < cmp counter	0
0x06	TG_CMP2_EN	Compare 2 enable	0: Disable Other: Enable. 1:data = cmp counter	0

			(regardless of counting direction) 2: data = cmp counter (while counting up) 3: data = cmp counter (while counting down) 4: data > cmp counter 5: data < cmp counter	
0x07	TG_CMP3_EN	Compare 3 enable	0: Disable Other: Enable. 1:data = cmp counter (regardless of counting direction) 2: data = cmp counter (while counting up) 3: data = cmp counter (while counting down) 4: data > cmp counter 5: data < cmp counter	0
0x08	TG_CMP0_TYPE	Compare 0 type	0: Table, 1: Linear	0
0x09	TG_CMP1_TYPE	Compare 1 type	0: Table, 1: Linear	0
0x0A	TG_CMP2_TYPE	Compare 2 type	0: Table, 1: Linear	0
0x0B	TG_CMP3_TYPE	Compare 3 type	0: Table, 1: Linear	0
0x0C	TG_CMPH_EN	Compare H enable	0: Disable, 1:Enable	0
0x0D	TG_CMPH_DIR_EN	Compare H direction enable	0: Disable, 1:Enable	0
0x0E	TG_CMPH_DIR	Compare H direction	0: Positive direction, 1: Negative direction.	0
0x10	TG_TRG0_SRC	Trigger output 0 (TRG0) source	Bit 0:CMP 0 Bit 1:CMP 1 Bit 2:CMP 2 Bit 3:CMP 3 Bit 4:CMP H Value: 0x00 ~ 0x1f	1
0x11	TG_TRG1_SRC	Trigger output 1 (TRG1) source	Bit 0:CMP 0 Bit 1:CMP 1	2

			Bit 2:CMP 2 Bit 3:CMP 3 Bit 4:CMP H Value: 0x00 ~ 0x1f	
0x12	TG_TRG2_SRC	Trigger output 2 (TRG2) source	Bit 0:CMP 0 Bit 1:CMP 1 Bit 2:CMP 2 Bit 3:CMP 3 Bit 4:CMP H Value: 0x00 ~ 0x1f	4
0x13	TG_TRG3_SRC	Trigger output 3 (TRG3) source	Bit 0:CMP 0 Bit 1:CMP 1 Bit 2:CMP 2 Bit 3:CMP 3 Bit 4:CMP H Value: 0x00 ~ 0x1f	8
0x14	TG_TRG0_PWD	TRG0 pulse width	Pulse Width = (N+ 5) * 10 ns Value: 0x05 ~ 0x7fffffff The value smaller than 0x05 is treated as 0x05.	5 (100 ns)
0x15	TG_TRG1_PWD	TRG1 pulse width	Pulse Width = (N+ 5) * 10 ns Value: 0x05 ~ 0x7fffffff The value smaller than 0x05 is treated as 0x05.	5 (100 ns)
0x16	TG_TRG2_PWD	TRG2 pulse width	Pulse Width = (N+ 5) * 10 ns Value: 0x05 ~ 0x7fffffff The value smaller than 0x05 is treated as 0x05.	5 (100 ns)
0x17	TG_TRG3_PWD	TRG3 pulse width	Pulse Width = (N+ 5) * 10 ns Value: 0x05 ~ 0x7fffffff The value smaller than 0x05 is treated as 0x05.	5 (100 ns)
0x18	TG_TRG0_CFG	TRG 0 configuration	Bit 0: Pulse logic inverse. Not Inverse (0) / Inverse	0

			(1) Bit 1~2: pulse (0) / toggle (1) / ByPass (2) / Disable (3) Bit 3~31: Reserved (set 0)	
0x19	TG_TRG1_CFG	TRG 1 configuration	Bit 0: Pulse logic inverse. Not Inverse (0) / Inverse (1) Bit 1~2: pulse (0) / toggle (1) / ByPass (2) / Disable (3) Bit 3~31: Reserved (set 0)	0
0x1A	TG_TRG2_CFG	TRG 2 configuration	Bit 0: Pulse logic inverse. Not Inverse (0) / Inverse (1) Bit 1~2: pulse (0) / toggle (1) / ByPass (2) / Disable (3) Bit 3~31: Reserved (set 0)	0
0x1B	TG_TRG3_CFG	TRG 3 configuration	Bit 0: Pulse logic inverse. Not Inverse (0) / Inverse (1) Bit 1~2: pulse (0) / toggle (1) / ByPass (2) / Disable (3) Bit 3~31: Reserved (set 0)	0
0x20	TG_ENCH_CFG	Encoder H configuration	Bit 0: Filter Enable. 1: Enable, 0: Disable. Bit 1: Counter Direction Inverse. 0: Not Inverse, 1: Inverse. Bit 2~4: Decoder mode. 0x00: OUT/DIR, 0x01: CW/CCW, 0x02: 1XAB,	0

			0x03: 2XAB, 0x04: 4XAB.	

HSL-4XMO Trigger parameter table

NO	Define	Description	Value	Default:
0x00	TG_CMP0_SRC	Compare 0 source	0: Command counter 1: Position counter	0
0x01	TG_CMP1_SRC	Compare 1 source	0: Command counter 1: Position counter	0
0x02	TG_CMP2_SRC	Compare 2 source	0: Command counter 1: Position counter	0
0x03	TG_CMP3_SRC	Compare 3 source	0: Comand counter 1: Position counter	0
0x04	TG_CMP0_EN	Compare 0 enable	0: Disable Other: Enable. 1:data = cmp counter (regardless of counting direction) 2: data = cmp counter (while counting up) 3: data = cmp counter (while counting down) 4: data > cmp counter 5: data < cmp counter	0
0x05	TG_CMP1_EN	Compare 1 enable	0: Disable Other: Enable. 1:data = cmp counter (regardless of counting direction) 2: data = cmp counter (while counting up) 3: data = cmp counter (while counting down) 4: data > cmp counter 5: data < cmp counter	0
0x06	TG_CMP2_EN	Compare 2 enable	0: Disable Other: Enable. 1:data = cmp counter	0

			(regardless of counting direction) 2: data = cmp counter (while counting up) 3: data = cmp counter (while counting down) 4: data > cmp counter 5: data < cmp counter	
0x07	TG_CMP3_EN	Compare 3 enable	0: Disable Other: Enable. 1:data = cmp counter (regardless of counting direction) 2: data = cmp counter (while counting up) 3: data = cmp counter (while counting down) 4: data > cmp counter 5: data < cmp counter	0
0x08	Reserve			
0x09	Reserve			
0x0A	Reserve			
0x0B	Reserve			
0x0C	Reserve			
0x0D	Reserve			
0x0E	Reserve			
0x10	Reserve			
0x11	Reserve			
0x12	Reserve			
0x13	Reserve			
0x14	Reserve			
0x15	Reserve			
0x16	Reserve			
0x17	Reserve			
0x18	TG_TRG0_CFG	TRG 0 configuration	Not Inverse (0) / Inverse	0

			(1)	
0x19	TG_TRG1_CFG	TRG 1 configuration	Not Inverse (0) / Inverse (1)	0
0x1A	TG_TRG2_CFG	TRG 2 configuration	Not Inverse (0) / Inverse (1)	0
0x1B	TG_TRG3_CFG	TRG 3 configuration	Not Inverse (0) / Inverse (1)	0
0x21	TG_CMP0_DIR	Compare 0 direction	0: Positive direction, 1: Negative direction.	0
0x22	TG_CMP1_DIR	Compare 1 direction	0: Positive direction, 1: Negative direction.	0
0x23	TG_CMP2_DIR	Compare 2 direction	0: Positive direction, 1: Negative direction.	0
0x24	TG_CMP3_DIR	Compare 3 direction	0: Positive direction, 1: Negative direction.	0

DB-8150 Trigger parameter table

NO	Define	Description	Value	Default:
0x00	TG_PWM0_PULSE_WIDTH	Set PWM pulse width (CH0)	1~65535 Note: Pulse Width(nsec) = Parameter * 100 + 85	0x3E7 (999) (100usec)
0x01	TG_PWM1_PULSE_WIDTH	Set PWM pulse width (CH1)	1~65535 Note: Pulse Width(nsec) = Parameter * 100 + 85	0x3E7 (999) (100usec)
0x02	TG_PWM0_MODE	Select the pulse output or level switch output (CH0)	0: Pulse output 1: Level switch output (toggle output)	0
0x03	TG_PWM1_MODE	Select the pulse output or level switch output (CH1)	0: Pulse output 1: Level switch output (toggle output)	0
0x04	TG_TIMER0_INTERVAL	Set Timer interval (CH0)	0~1073741823 Note: Timer cycle time(nsec) = (interval + 5) * 25	0 (125nsec)
0x05	TG_TIMER1_INTERVAL	Set Timer interval (CH1)	0~1073741823 Note: Timer cycle time(nsec) = (interval + 5) * 25	0 (125nsec)
0x06	TG_ENC0_CNT_DIR	Set Encoder count direction (CH0)	0: Not inverse 1: Inverse	0
0x07	TG_ENC1_CNT_DIR	Set Encoder count direction (CH1)	0: Not inverse 1: Inverse	0
0x08	TG_IPT0_MODE	Set pulse input mode (CH0)	0: OUT/DIR 1: CW/CCW 2: 1x AB-Phase 3: 2x AB-Phase 4: 4x AB-Phase	0
0x09	TG_IPT1_MODE	Set pulse input mode (CH1)	0: OUT/DIR 1: CW/CCW	0

			2: 1x AB-Phase 3: 2x AB-Phase 4: 4x AB-Phase	
0x0A	TG_EZ0_CLEAR_EN	Enable EZ clear (CH0)	0: Disable 1: Enable	0
0x0B	TG_EZ1_CLEAR_EN	Enable EZ clear (CH1)	0: Disable 1: Enable	0
0x0C	TG_EZ0_CLEAR_LOGIC	Clear logic setting (CH0)	0: Falling edge 1: Rising edge	0
0x0D	TG_EZ1_CLEAR_LOGIC	Clear logic setting (CH1)	0: Falling edge 1: Rising edge	0
0x0E	TG_CNT0_SOURCE	Set counter's source (CH0)	0: Encoder0 (Carrier Board EA/B 0) 1: Encoder1 (Carrier Board EA/B 1) 2: Encoder2 (Daughter Board DEA/B 2) 3: Encoder3 (Daughter Board DEA/B 3) 4: Timer0 5: Timer1	0x2
0x0F	TG_CNT1_SOURCE	Set counter's source (CH1)	0: Encoder0 (Carrier Board EA/B 0) 1: Encoder1 (Carrier Board EA/B 1) 2: Encoder2 (Daughter Board DEA/B 2) 3: Encoder3 (Daughter Board DEA/B 3) 4: Timer0 5: Timer1	0x3
0x10	TG_FTR0_EN	Filter enable (CH0)	0: Disable 1: Enable	0
0x11	TG_FTR1_EN	Filter enable (CH1)	0: Disable	0

			1: Enable	
0x12	TG_DI_LATCH0_EN	Enable DI LATCH (CH0)	0: Disable 1: Enable	0
0x13	TG_DI_LATCH1_EN	Enable DI LATCH (CH1)	0: Disable 1: Enable	0
0x14	TG_DI_LATCH0_EDGE	Set DI LATCH condition (CH0)	0: DI falling edge to latch 1: DI Rising edge to latch	0
0x15	TG_DI_LATCH1_EDGE	Set DI LATCH condition (CH1)	0: DI falling edge to latch 1: DI Rising edge to latch	0
0x16	TG_DI_LATCH0_VALUE	Get DI Latch Value (CH0)		
0x17	TG_DI_LATCH1_VALUE	Get DI Latch Value (CH1)		
0x18	TG_TRGOUT_MAP	Set Trigger Out Mapping	0~65535 (Bit16~Bit31 reserved) *Note(1)	0x9
0x19	TG_TRGOUT_LOGIC	Set Trigger Out Logic	0~255 (Bit8~Bit31 reserved) *Note(2)	0
0x1A	TG_FIFO_LEVEL	Set/Get FIFO size Level	0: level=0 (empty) 1: level=1/4 2: level=1/2 (default) 3: level=3/4 Note: Only Support CH0	0
0x1B	TG_PWM0_SOURCE	Set PWM Source (CH0)	Bit 0: Timer 0: Disable 1: Enable Bit 1: Linear comparator 0: Disable 1: Enable Bit 2: FIFO comparator 0: Disable 1: Enable Other bits reserved Note: FIFO comparator Only Support CH0	0x4 (FIFO comparator)

0x1C	TG_PWM1_SOURCE	Set PWM Source (CH1)	Bit 0: Timer 0: Disable 1: Enable Bit 1: Linear comparator 0: Disable 1: Enable Bit 2: FIFO comparator 0: Disable 1: Enable Other bits reserved Note: FIFO comparator Only Support CH0	0x4 (FIFO comparator)

*Note(1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Function	TRG3b	TRG3a	TRG2b	TRG2a	TRG1b	TRG1a	TRG0b	TRG0a
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Function	TRG7b	TRG7a	TRG6b	TRG6a	TRG5b	TRG5a	ECAT-TRG4b	ECAT-TRG4a

The DB-8150 has 8 trigger output pins and 2 channel of PWM.

By this function, the trigger output pins can be mapped with 2 channel of PWM.

The symbol TRG0 ~ TRG7 representing pin0~pin7 of trigger output pins.

The "a" symbol represent PWM0.

The "b" symbol represent PWM1.

For example:

TRG0a=1 represent the PWM0 signal will be output by trigger output pin0.

TRG0a=0 represent the PWM0 signal will not be output by trigger output pin0.

if TRG0a and TRG0b are set to 1 at the same time,the pin0 will output signal by PWM0 and PWM1 making OR operator.

*Note(2)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Function	TRGInv7	TRGInv6	TRGInv5	TRGInv4	TRGInv3	TRGInv2	TRGInv1	TRGInv0

This parameter is used to set the logic of trigger output signal.

For example:

TRGInv0=1 represent the trigger output signal will be inversed by pin0.

TRGInv0=0 represent the trigger output signal will not be inversed by pin0.

PCI – C154(+) Trigger parameter table

NO	Define	Description	Value	Default:
0x00	TIG_ENC_IPT_MODE0	Encoder pulse input mode (CH0)	0: OUT/DIR (EA = Out, EB = Dir) 1: CW/CCW (EA = CW, EB = CCW) 2: 1x AB-Phase0 3: 2x AB-Phase 4: 4x AB-Phase	4
0x01	TIG_ENC_IPT_MODE1	Encoder pulse input mode (CH1)	0: OUT/DIR (EA = Out, EB = Dir) 1: CW/CCW (EA = CW, EB = CCW) 2: 1x AB-Phase0 3: 2x AB-Phase 4: 4x AB-Phase	4
0x02	TIG_ENC_IPT_MODE2	Encoder pulse input mode (CH2)	0: OUT/DIR (EA = Out, EB = Dir) 1: CW/CCW (EA = CW, EB = CCW) 2: 1x AB-Phase0 3: 2x AB-Phase 4: 4x AB-Phase	4
0x03	TIG_ENC_IPT_MODE3	Encoder pulse input mode (CH3)	0: OUT/DIR (EA = Out, EB = Dir) 1: CW/CCW (EA = CW, EB = CCW) 2: 1x AB-Phase0 3: 2x AB-Phase 4: 4x AB-Phase	4
0x08	TIG_ENC_EA_INV0	Invert EA encoder signal (CH0) Note(3)	1: Inverse 0: Not Inverse	0
0x09	TIG_ENC_EA_INV1	Invert EA encoder signal (CH1) Note(3)	1: Inverse 0: Not Inverse	0

0x0A	TIG_ENC_EA_INV2	Invert EA encoder signal (CH2) Note(3)	1: Inverse 0: Not Inverse	0
0x0B	TIG_ENC_EA_INV3	Invert EA encoder signal (CH3) Note(3)	1: Inverse 0: Not Inverse	0
0x10	TIG_ENC_EB_INV0	Invert EB encoder signal (CH0) Note(3)	1: Inverse 0: Not Inverse	0
0x11	TIG_ENC_EB_INV1	Invert EB encoder signal (CH1) Note(3)	1: Inverse 0: Not Inverse	0
0x12	TIG_ENC_EB_INV2	Invert EB encoder signal (CH2) Note(3)	1: Inverse 0: Not Inverse	0
0x13	TIG_ENC_EB_INV3	Invert EB encoder signal (CH3) Note(3)	1: Inverse 0: Not Inverse	0
0x28	TIG_ENC_SIGNAL_FILITER_EN0	CH0 encoder signal Low-Pass filter (Cutoff(3db) Frequency: 5MHz)	1: Enable 0: Disable	1
0x29	TIG_ENC_SIGNAL_FILITER_EN1	CH1 encoder signal Low-Pass filter (Cutoff(3db) Frequency: 5MHz)	1: Enable 0: Disable	1
0x2A	TIG_ENC_SIGNAL_FILITER_EN2	CH2 encoder signal Low-Pass filter (Cutoff(3db) Frequency: 5MHz)	1: Enable 0: Disable	1
0x2B	TIG_ENC_SIGNAL_FILITER_EN3	CH3 encoder signal Low-Pass filter (Cutoff(3db) Frequency: 5MHz)	1: Enable 0: Disable	1
0x30	TIG_TIMER8_DIR	Timer8 direction Note (1)	0: Positive count 1: Negative count	0
0x31	TIG_TIMER8_ITV	Timer8 Interval Note (1)	Timer8 Interval (sec) = Value * 30nS	0
0x32	TIG_CMP0_SRC	Compare CH0 source	0: Encoder counter 0 1: Timer 8 counter Note(1)	0
0x33	TIG_CMP1_SRC	Compare CH1 source	0: Encoder counter 1 1: Timer 8 counter	0

			Note(1)	
0x34	TIG_CMP2_SRC	Compare CH2 source	0: Encoder counter 2 1: Timer 8 counter Note(1)	0
0x35	TIG_CMP3_SRC	Compare CH3 source	0: Encoder counter 3 1: Timer 8 counter Note(1)	0
0x42	TIG_TRGOUT0_MAP	Trigger Output Pin (CH0) Mapping	0: Output_Pin_0 1: Output_Pin_1 2: Output_Pin_2 3: Output_Pin_3	0
0x43	TIG_TRGOUT1_MAP	Trigger Output Pin (CH1) Mapping	0: Output_Pin_0 1: Output_Pin_1 2: Output_Pin_2 3: Output_Pin_3	0
0x44	TIG_TRGOUT2_MAP	Trigger Output Pin (CH2) Mapping	0: Output_Pin_0 1: Output_Pin_1 2: Output_Pin_2 3: Output_Pin_3	0
0x45	TIG_TRGOUT3_MAP	Trigger Output Pin (CH3) Mapping	0: Output_Pin_0 1: Output_Pin_1 2: Output_Pin_2 3: Output_Pin_3	0
0x4A	TIG_TRGOUT0_LOGIC	Trigger Output Pin (CH0) Logic	0: Not inverse 1: Inverse	0
0x4B	TIG_TRGOUT1_LOGIC	Trigger Output Pin (CH1) Logic	0: Not inverse 1: Inverse	0
0x4C	TIG_TRGOUT2_LOGIC	Trigger Output Pin (CH2) Logic	0: Not inverse 1: Inverse	0
0x4D	TIG_TRGOUT3_LOGIC	Trigger Output Pin (CH3) Logic	0: Not inverse 1: Inverse	0
0x52	TIG_PWM0_PULSE_WIDTH	PWM pulse width (CH0)	PWM pulse width (sec) = Value * 480nS	2

			+ 60nS Value range: 1~8191	
0x53	TIG_PWM1_PULSE_WIDTH	PWM pulse width (CH1)	PWM pulse width (sec) = Value * 480nS + 60nS Value range: 1~8191	2
0x54	TIG_PWM2_PULSE_WIDTH	PWM pulse width (CH2)	PWM pulse width (sec) = Value * 480nS + 60nS Value range: 1~8191	2
0x55	TIG_PWM3_PULSE_WIDTH	PWM pulse width (CH3)	PWM pulse width (sec) = Value * 480nS + 60nS Value range: 1~8191	2
0x5A	TIG_PWM0_MODE	Select the pulse output or level switch output (CH0)	0: Pulse output 1: Level switch output (toggle output)	0
0x5B	TIG_PWM1_MODE	Select the pulse output or level switch output (CH1)	0: Pulse output 1: Level switch output (toggle output)	0
0x5C	TIG_PWM2_MODE	Select the pulse output or level switch output (CH2)	0: Pulse output 1: Level switch output (toggle output)	0
0x5D	TIG_PWM3_MODE	Select the pulse output or level switch output (CH3)	0: Pulse output 1: Level switch output (toggle output)	0
0x62	TIG_TIMER0_ITV	Timer0 Interval (Write Only) **Note(2)**	Timer0 Interval Value(sec) = Value * 30nS	0
0x63	TIG_TIMER1_ITV	Timer1 Interval (Write Only) **Note(2)**	Timer1 Interval Value(sec) = Value * 30nS	0
0x64	TIG_TIMER2_ITV	Timer2 Interval (Write Only) **Note(2)**	Timer2 Interval Value(sec)	0

			= Value * 30nS	
0x65	TIG_TIMER3_ITV	Timer3 Interval (Write Only) **Note(2)**	Timer3 Interval Value(sec) = Value * 30nS	0
0x6A	TIG_FIFO_LEVEL0	FIFO Comparator (CH0) Level	FIFO Comparator (CH0) Level Value (0~1023)	0
0x6B	TIG_FIFO_LEVEL1	FIFO Comparator (CH1) Level	FIFO Comparator (CH1) Level Value (0~1023)	0
0x6C	TIG_FIFO_LEVEL2	FIFO Comparator (CH2) Level	FIFO Comparator (CH2) Level Value (0~1023)	0
0x6D	TIG_FIFO_LEVEL3	FIFO Comparator (CH3) Level	FIFO Comparator (CH3) Level Value (0~1023)	0
0x72	TIG_OUTPUT_EN0	Trigger output pin enable (CH0)	0: Disable 1: Enable	1
0x73	TIG_OUTPUT_EN1	Trigger output pin enable (CH1)	0: Disable 1: Enable	1
0x74	TIG_OUTPUT_EN2	Trigger output pin enable (CH2)	0: Disable 1: Enable	1
0x75	TIG_OUTPUT_EN3	Trigger output pin enable (CH3)	0: Disable 1: Enable	1

Note (1) The timer 8 is used to simulate for encoder. It is used to be comparator source for 4 channels.

Note (2) Timer 0 ~ 3 are used to generate trigger signal periodically.

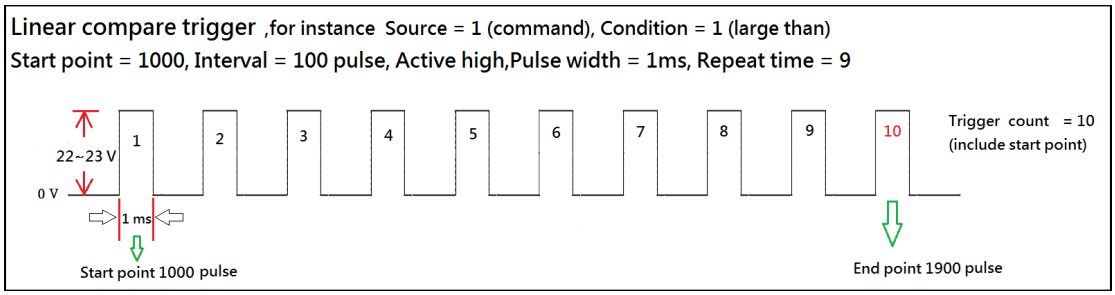
Note (3) When user change EA/EB logic once, the encoder counter will count once.

EMX-100 Trigger parameter table

NO	Define	Description	Value	Default:
0x00	TGR0_CMP_SRC	Set axis 0 compare source using encoder or command position	0: command position 1: encoder	1
0x01	TGR1_CMP_SRC	Set axis 1 compare source using encoder or command position	0: command position 1: encoder	1
0x02	TGR0_CMP_COND	Set axis 0 compare condition	0:lager than & equal (>=) 1:large than (>) 2:equal (=) 3:less than (<)	2
0x03	TGR1_CMP_COND	Set axis 1 compare condition	0:lager than & equal (>=) 1:large than (>) 2:equal (=) 3:less than (<)	2
0x04	TGR0_CMP_VALUE	Set axis 0 single comparator value or linear comparator start point by compare trigger mode	Integer value	100
0x05	TGR1_CMP_VALUE	Set axis 1 single comparator value or linear comparator start point by compare trigger mode	Integer value	100
0x06	TGR0_PULSE_WIDTH	Set axis 0 output pulse width	0: 64 usec, 1: 256 usec, 2: 1 MSec	0
0x07	TGR1_PULSE_WIDTH	Set axis 1 output pulse width	0: 64 usec, 1: 256 usec,	0

			2: 1 MSec	
0x08	TGR0_PULSE_LOGIC	Set axis 0 pulse output logic	0: Active low 1: Active high	0
0x09	TGR1_PULSE_LOGIC	Set axis 1 pulse output logic	0: Active low 1: Active high	0
0x0A	TGR0_CMP_EN	Enable axis 0 compare trigger function; The output signal is generated via TRG1 pin when compare condition is TRUE	0: disable 1: enable	0
0x0B	TGR1_CMP_EN	Enable axis 1 compare trigger function; The output signal is generated via TRG2 pin when compare condition is TRUE	0: disable 1: enable	0
0x0C	TGR0_CMP_MODE	Set axis 0 compare trigger mode	0:single compare 1:linear comapre	0
0x0D	TGR1_CMP_MODE	Set axis 1 compare trigger mode	0:single compare 1:linear comapre	0
0x0E	TGR0_LCMP_INTER	Set axis 0 linear compare tigger inverval (pulse)	Integer value	0
0x0F	TGR1_LCMP_INTER	Set axis 1 linear compare tigger inverval (pulse)	Integer value	0
0x10	TGR0_LCMP_RETIME	Set axis 0 linear compare tigger repeat times	Integer value	0
0x11	TGR1_LCMP_RETIME	Set axis 1 linear compare tigger repeat times	Integer value	0

Note 1: Here is a simple example as below. The real trigger count includes start point. Trigger count = Repeat time + 1



Note 2: When finish all triggering point, user must configure and enable again for functioning it.

Note 3: The maximum compare trigger frequency supports 500 Hz, if user configure maximum speed 4M pps,the interval should be large than 8000(4000000/500) pulse

PCI-8254/58 / AMP-204/8C Trigger parameter table

NO	Define	Description	Value	Default:
0x00	TGR_LCMP0_SRC	Linear compare 0 (LCMP0) source	0 ~ 7: Encoder counter 0~7 8: Timer 0 counter 9: Disable	9
0x01	TGR_LCMP1_SRC	Linear compare 1 (LCMP1) source	0 ~ 7: Encoder counter 0~7 8: Timer 0 counter 9: Disable	9
0x02	TGR_TCMP0_SRC	Table compare 0 (TCMP0) source	0 ~ 7: Encoder counter 0~7 8: Timer 0 counter 9: Disable	9
0x03	TGR_TCMP1_SRC	Table compare 1 (TCMP1) source	0 ~ 7: Encoder counter 0~7 8: Timer 0 counter 9: Disable	9
0x04	TGR_TCMP0_DIR	Table compare 0 (TCMP0) direction	0: Negative direction 1: Positive direction 2: Bi-direction(No direction)	0
0x05	TGR_TCMP1_DIR	Table compare 1 (TCMP1) direction	0: Negative direction 1: Positive direction 2: Bi-direction(No	0

			direction)	
0x06	TGR_TRG_EN	TRG 0 ~ 3 enable by bit NOTE: This parameter is also controlled by board parameter “PWMx map DO” and “VAO table” functions.	Bit x: (0: disable, 1: enable) Bit 0:TRG0 enable Bit 1:TRG1 enable Bit 2:TRG2 enable Bit 3:TRG3 enable	0
0x10	TGR_TRG0_SRC	Trigger output 0 (TRG0) source Note: OR multi-sources, then output to TRG0)	Bit x:(1: On, 0: Off) Bit 0:Manual0 Bit 1:Reserved Bit 2:FCMP0 Bit 3:FCMP1 Bit 4:LCMP0 Bit 5:LCMP1 Bit 6:MCMP Bit 7:FCMP2 Bit 8:FCMP3 Bit 9 LCMP2 Bit 10 LCMP3	0
0x11	TGR_TRG1_SRC	Trigger output 1 (TRG1) source Note: OR multi-sources, then output to TRG0)	Bit x:(1: On, 0: Off) Bit 0: Manual0 Bit 1:Reserved Bit 2:FCMP0 Bit 3:FCMP1 Bit 4:LCMP0 Bit 5:LCMP1 Bit 6:MCMP Bit 7:FCMP2 Bit 8:FCMP3 Bit 9 LCMP2 Bit 10 LCMP3	0
0x12	TGR_TRG2_SRC	Trigger output 2 (TRG2) source Note: OR multi-sources, then output to TRG0)	Bit x:(1: On, 0: Off) Bit 0: Manual0 Bit 1:Reserved Bit 2:FCMP0 Bit 3:FCMP1 Bit 4:LCMP0	0

			Bit 5:LCMP1 Bit 6:MCMP Bit 7:FCMP2 Bit 8:FCMP3 Bit 9 LCMP2 Bit 10 LCMP3	
0x13	TGR_TRG3_SRC	Trigger output 3 (TRG3) source Note: OR multi-sources, then output to TRG0)	Bit x:(1: On, 0: Off) Bit 0:Manual0 Bit 1:Reserved Bit 2:FCMP0 Bit 3:FCMP1 Bit 4:LCMP0 Bit 5:LCMP1 Bit 6:MCMP Bit 7:FCMP2 Bit 8:FCMP3 Bit 9 LCMP2 Bit 10 LCMP3	0
0x14	TGR_TRG0_PWD	TRG0 pulse width	Pulse Width = (N-1)*20ns N = 2 ~ 0xffffff	11
0x15	TGR_TRG1_PWD	TRG1 pulse width	Pulse Width = (N-1)*20ns N = 2 ~ 0xffffff	11
0x16	TGR_TRG2_PWD	TRG2 pulse width	Pulse Width = (N-1)*20ns N = 2 ~ 0xffffff	11
0x17	TGR_TRG3_PWD	TRG3 pulse width	Pulse Width = (N-1)*20ns N = 2 ~ 0xffffff	11
0x18	TGR_TRG0_LOGIC	TRG 0 logic	0: Not inverse 1:Inverse	0
0x19	TGR_TRG1_LOGIC	TRG 1 logic	0: Not inverse 1:Inverse	0
0x1A	TGR_TRG2_LOGIC	TRG 2 logic	0: Not inverse 1:Inverse	0
0x1B	TGR_TRG3_LOGIC	TRG 3 logic	0: Not inverse 1:Inverse	0

0x1C	TGR_TRG0_TGL	TRG 0 toggle mode	0: Pulse out 1:Toggle out	0
0x1D	TGR_TRG1_TGL	TRG 1 toggle mode	0: Pulse out 1:Toggle out	0
0x1E	TGR_TRG2_TGL	TRG 2 toggle mode	0: Pulse out 1:Toggle out	0
0x1F	TGR_TRG3_TGL	TRG 3 toggle mode	0: Pulse out 1:Toggle out	0
0x20	TIMR_ITV	Timer Interval	Timer Interval = N * 100 ns N = 1~ 53687091	1(100 ns)
0x21	TIMR_DIR	Timer direction	0: Positive count 1: Negative count	0
0x22	TIMR_RING_EN	Enable timer counter to be as Ring counter	0: Disable (0x7fffffff+1➔0x80000000) (0x80000001-1➔0x80000000) 1: Enable (0x7fffffff+1➔0x00000000) (0x00000000-1➔0x7fffffff)	0
0x23	TIMR_EN	Timer enable	0: Disable, 1:Enable	0
0x30	TGR_MCMP0_SRC	Multi-axis comparator 0 (MCMP0) source	0 ~ 7: Encoder counter 0~7 8: Timer 0 counter 9: Disable	9
0x31	TGR_MCMP1_SRC	Multi-axis comparator 1 (MCMP1) source	0 ~ 7: Encoder counter 0~7 8: Timer 0 counter 9: Disable	9
0x32	TGR_MCMP2_SRC	Multi-axis comparator 2 (MCMP2) source	0 ~ 7: Encoder counter 0~7 8: Timer 0 counter 9: Disable	9
0x33	TGR_MCMP3_SRC	Multi-axis comparator 3 (MCMP3) source	0 ~ 7: Encoder counter 0~7	9

			8: Timer 0 counter 9: Disable	
0x34	TGR_MCMP_MODE	Mode to decide when multi-axis comparator can trigger PWM	0: original mode: when motor reaches boundary defined by window 1: precision mode: when motor is inside boundary defined by window and the position deviation relative to compared point is shortest	0
0x35	TGR_TRG0_TOGGLE_MODE	Enable this mode to allow user to set the status of the trigger out pin. NOTE: This mode will affect other FPGA modules such that Compare trigger.	0: disable 1: enable	0
0x36	TGR_TRG1_TOGGLE_MODE	Same as above	0: disable 1: enable	0
0x37	TGR_TRG2_TOGGLE_MODE	Same as above	0: disable 1: enable	0
0x38	TGR_TRG3_TOGGLE_MODE	Same as above	0: disable 1: enable	0
0x39	TGR_TRG0_TOGGLE_STATUS	Write and read the Toggle output status	0: output low 1: output high	0
0x3A	TGR_TRG1_TOGGLE_STATUS	Same as above	0: output low 1: output high	0
0x3B	TGR_TRG2_TOGGLE_STATUS	Same as above	0: output low 1: output high	0
0x3C	TGR_TRG3_TOGGLE_STATUS	Same as above	0: output low 1: output high	0
0x40	TGR_TCMP2_SRC	Table compare 2 (TCMP2) source	0 ~ 7: Encoder counter 0~7 8: Timer 0 counter 9: Disable	9

0x41	TGR_TCMP3_SRC	Table compare 3 (TCMP3) source	0 ~ 7: Encoder counter 0~7 8: Timer 0 counter 9: Disable	9
0x42	TGR_TCMP2_DIR	Table compare 2 (TCMP2) direction	0: Negative direction 1: Positive direction 2: Bi-direction(No direction)	0
0x43	TGR_TCMP3_DIR	Table compare 3 (TCMP3) direction	0: Negative direction 1: Positive direction 2: Bi-direction(No direction)	0
0x44	TGR_LCMP2_SRC	Linear compare 2 (LCMP2) source	0 ~ 7: Encoder counter 0~7 8: Timer 0 counter 9: Disable	9
0x45	TGR_LCMP3_SRC	Linear compare 3 (LCMP3) source	0 ~ 7: Encoder counter 0~7 8: Timer 0 counter 9: Disable	9

ECAT-4XMO , ECAT-TRG4 Trigger parameter table

NO	Define	Description	Value	Default:
0x00	TGR_LCMP0_SRC	Linear compare 0 (LCMP0) source	0 ~ 3: Encoder counter 0~3 4: Timer 0 counter 5: Disable	5
0x01	TGR_LCMP1_SRC	Linear compare 1 (LCMP1) source	0 ~ 3: Encoder counter 0~3 4: Timer 0 counter 5: Disable	5
0x02	TGR_TCMP0_SRC	Table compare 0 (TCMP0) source	0 ~ 3: Encoder counter 0~3 4: Timer 0 counter 5: Disable	5
0x03	TGR_TCMP1_SRC	Table compare 1 (TCMP1) source	0 ~ 3: Encoder counter 0~3 4: Timer 0 counter 5: Disable	5
0x04	TGR_TCMP0_DIR	Table compare 0 (TCMP0) direction	0: Negative direction 1: Positive direction 2: Bi-direction(No direction)	1
0x05	TGR_TCMP1_DIR	Table compare 1 (TCMP1) direction	0: Negative direction 1: Positive direction 2: Bi-direction(No direction)	1
0x06	TGR_TRG_EN	TRG 0 ~ 3 enable by bit	Bit x: (0: disable, 1: enable) Bit 0:TRG0 enable, PWM pulse out0 enable. Bit 1:TRG1 enable, PWM pulse out1 enable. Bit 2:TRG2 enable, PWM pulse out2 enable. Bit 3:TRG3 enable, PWM	0

			pulse out3 enable.	
0x07	TGR_TCMP0_REUSE	Table compare 0 (TCMP0) compare data reuse function enable.	0: Disable reuse function. 1: Enable reuse function. NOTE: Must make sure comparison is not running before using.	0
0x08	TGR_TCMP1_REUSE	Table compare 1(TCMP1) compare data reuse function enable.	0: Disable reuse function. 1: Enable reuse function. NOTE: Must make sure comparison is not running before using.	0
0x09	TGR_TCMP0_TRANSFER_DONE	Table compare 0(TCMP0) data transfer done status.	-1: Not in using. 0: Transfer not finish. 1: Transfer finish. Be attention: Transfer finish status is read clear status. Status will be “-1” -> “0”... data transfer… -> ”1” (By user program read) -> “-1”.	-1
0x0A	TGR_TCMP1_TRANSFER_DONE	Table compare 1(TCMP1) data transfer done status.	-1: Not in using. 0: Transfer not finish. 1: Transfer finish. Be attention: Transfer finish status is read clear status. Status will be “-1” -> “0”... data transfer… -> ”1” (By user program	-1

			read) -> “-1”.	
0x10	TGR_TRG0_SRC	Trigger output 0 (TRG0) source Note: OR multi-sources, then output to TRG0)	Bit x:(1: On, 0: Off) Bit 0:Manual0 Bit 1:Reserved Bit 2:TCMP0 Bit 3:TCMP1 Bit 4:LCMP0 Bit 5:LCMP1 Bit 6: Reserved Bit 7:TCMP2 Bit 8:TCMP3 Bit 9:LCMP2 Bit 10:LCMP3	0
0x11	TGR_TRG1_SRC	Trigger output 1 (TRG1) source Note: OR multi-sources, then output to TRG0)	Bit x:(1: On, 0: Off) Bit 0: Manual0 Bit 1:Reserved Bit 2:TCMP0 Bit 3:TCMP1 Bit 4:LCMP0 Bit 5:LCMP1 Bit 6: Reserved Bit 7:TCMP2 Bit 8:TCMP3 Bit 9:LCMP2 Bit 10:LCMP3	0
0x12	TGR_TRG2_SRC	Trigger output 2 (TRG2) source Note: OR multi-sources, then output to TRG0)	Bit x:(1: On, 0: Off) Bit 0: Manual0 Bit 1:Reserved Bit 2:TCMP0 Bit 3:TCMP1 Bit 4:LCMP0 Bit 5:LCMP1 Bit 6: Reserved Bit 7:TCMP2 Bit 8:TCMP3 Bit 9:LCMP2 Bit 10:LCMP3	0

0x13	TGR_TRG3_SRC	Trigger output 3 (TRG3) source Note: OR multi-sources, then output to TRG0)	Bit x:(1: On, 0: Off) Bit 0:Manual0 Bit 1:Reserved Bit 2:TCMP0 Bit 3:TCMP1 Bit 4:LCMP0 Bit 5:LCMP1 Bit 6: Reserved Bit 7:TCMP2 Bit 8:TCMP3 Bit 9:LCMP2 Bit 10:LCMP3	0
0x14	TGR_TRG0_PWD	TRG0 pulse width	Pulse Width = (N-1)*20ns N = 2 ~ 0xffffff	11
0x15	TGR_TRG1_PWD	TRG1 pulse width	Pulse Width = (N-1)*20ns N = 2 ~ 0xffffff	11
0x16	TGR_TRG2_PWD	TRG2 pulse width	Pulse Width = (N-1)*20ns N = 2 ~ 0xffffff	11
0x17	TGR_TRG3_PWD	TRG3 pulse width	Pulse Width = (N-1)*20ns N = 2 ~ 0xffffff	11
0x18	TGR_TRG0_LOGIC	TRG 0 logic	0: Not inverse 1: Inverse	0
0x19	TGR_TRG1_LOGIC	TRG 1 logic	0: Not inverse 1: Inverse	0
0x1A	TGR_TRG2_LOGIC	TRG 2 logic	0: Not inverse 1: Inverse	0
0x1B	TGR_TRG3_LOGIC	TRG 3 logic	0: Not inverse 1: Inverse	0
0x1C	TGR_TRG0_TGL	TRG 0 toggle mode	0: Pulse out 1: Toggle out	0
0x1D	TGR_TRG1_TGL	TRG 1 toggle mode	0: Pulse out 1: Toggle out	0
0x1E	TGR_TRG2_TGL	TRG 2 toggle mode	0: Pulse out 1: Toggle out	0

0x1F	TGR_TRG3_TGL	TRG 3 toggle mode	0: Pulse out 1: Toggle out	0
0x20	TIMR_ITV	Timer Interval	Timer Interval = N * 40 ns N = 1~ 107374182	1 (40 ns)
0x21	TIMR_DIR	Timer direction	0: Positive count 1: Negative count	0
0x22	TIMR_RING_EN	Enable timer counter to be as Ring counter	0: Disable (0x7fffffff+1➔0x80000000) (0x80000001-1➔0x80000000) 1: Enable (0x7fffffff+1➔0x00000000) (0x00000000-1➔0x7fffffff)	0
0x23	TIMR_EN	Timer enable	0: Disable, 1:Enable	0
0x40	TGR_TCMP2_SRC	Table compare 2 (TCMP2) source	0 ~ 3: Encoder counter 0~3 4: Timer 0 counter 5: Disable	5
0x41	TGR_TCMP3_SRC	Table compare 3 (TCMP3) source	0 ~ 3: Encoder counter 0~3 4: Timer 0 counter 5: Disable	5
0x42	TGR_TCMP2_DIR	Table compare 2 (TCMP2) direction	0: Negative direction 1: Positive direction 2: Bi-direction(No direction)	1
0x43	TGR_TCMP3_DIR	Table compare 3 (TCMP3) direction	0: Negative direction 1: Positive direction 2: Bi-direction(No direction)	1
0x44	TGR_LCMP2_SRC	Linear compare 2 (LCMP2) source	0 ~ 3: Encoder counter 0~3 4: Timer 0 counter 5: Disable	5
0x45	TGR_LCMP3_SRC	Linear compare 3	0 ~ 3: Encoder counter	5

		(LCMP3) source	0~3 4: Timer 0 counter 5: Disable	
0x46	TGR_TCMP2_REUSE	Table compare 2 (TCMP2) compare data reuse function enable.	0: Disable reuse function. 1: Enable reuse function. NOTE: Must make sure comparison is not running before using.	0
0x47	TGR_TCMP3_REUSE	Table compare 3(TCMP3) compare data reuse function enable.	0: Disable reuse function. 1: Enable reuse function. NOTE: Must make sure comparison is not running before using.	0
0x48	TGR_TCMP2_TRANSFER_DONE	Table compare 2(TCMP2) data transfer done status.	-1: Not in using. 0: Transfer not finish. 1: Transfer finish. Be attention: Transfer finish status is read clear status. Status will be “-1” -> “0”... data transfer… -> ”1” (By user program read) -> “-1”.	-1
0x49	TGR_TCMP3_TRANSFER_DONE	Table compare 2(TCMP3) data transfer done status.	-1: Not in using. 0: Transfer not finish. 1: Transfer finish. Be attention: Transfer finish status is read clear status. Status will be “-1” -> “0”... data transfer… -> ”1” (By user program read) -> “-1”.	-1
0x50	TGR_CMP_EXTENC0_SRC	Set external encoder 0 source Axis ID. Only could set on same 833x master card slave servo motor has	0 ~ 65535 : Axis ID	0

		encoder pulse output function (ECAT-ECAT-TRG4 only)		
0x51	TGR_CMP_EXTENC1_SRC	Set external encoder 1 source Axis ID. Only could set on same 833x master card slave servo motor has encoder pulse output function (ECAT-ECAT-TRG4 only)	0 ~ 65535 : Axis ID	0
0x52	TGR_CMP_EXTENC2_SRC	Set external encoder 2 source Axis ID. Only could set on same 833x master card slave servo motor has encoder pulse output function (ECAT-ECAT-TRG4 only)	0 ~ 65535 : Axis ID	0
0x53	TGR_CMP_EXTENC3_SRC	Set external encoder 3 source Axis ID. Only could set on same 833x master card slave servo motor has encoder pulse output function (ECAT-ECAT-TRG4 only)	0 ~ 65535 : Axis ID	0

K. Latch parameter table

PCI-C154(+) Latch parameter table

PCI-C154(+) Latch parameter table				
NO	Define	Description	Value	Default:
0x00	LTC_ENC_IPT_MODE	Encoder pulse input mode	0: OUT/DIR (EA = Out, EB = Dir) 1: CW/CCW (EA = CW, EB = CCW) 2: 1x AB-Phase0 3: 2x AB-Phase 4: 4x AB-Phase	4
0x01	LTC_ENC_EA_INV	Invert EA encoder signal	1: Inverse 0: Not Inverse	0
0x02	LTC_ENC_EB_INV	Invert EB encoder signal	1: Inverse 0: Not Inverse	0
0x05	LTC_ENC_SIGNAL_FLT_EN	Encoder signal Low-Pass filter (Cutoff(3db) Frequency: 10MHz)	1: Enable 0: Disable	1
0x06	LTC_FIFO_HIGH_LEVEL	Latch fifo high Level	0 ~ 255	0
0x07	LTC_SIGNAL_FLT_EN	Latch signal filter	1: Enable 0: Disable	1
0x08	LTC_SIGNAL_TRIG_LOGIC	Latch signal trigger logic	1: Rising active 0: Falling active	0

PCI-8254/58 / AMP-204/8C Latch parameter table

PCI-8254/58 / AMP-204/8C Latch parameter table				
NO	Define	Description	Value	Default:
0x10	LTC_IPT	Latch source	Source to trigger the position latch: bit 0~7: TTL0~TTL7 Digital input signals; bit 8~11: PWM pulse out	0
0x11	LTC_ENC	Latch encoder	Determine which encoder is latched when latch source is triggered; The range of encoder no. is 0~7	0
0x12	LTC_LOGIC	Latch logic	Support three kinds of logic modes to trigger the latch process: 0: Only RisingEdge 1: Only FallingEdge 2: Both RisingEdge and FallingEdge	0

AMP-104C Latch parameter table

AMP-104C Latch parameter table				
NO	Define	Description	Value	Default
0x10	LTC_IPT	Latch source	Enable or disable Source to trigger the position latch: bit 0~3:SCSI 68 pin DI0 ~ DI3 signals bit 4~7:TTL DI0 ~ DI3 signals bit set 1 -> enable bit set 0 -> disable For instance, value = 0x6 -> DI1 and DI2 eanble and select these as source, others disable.	0
0x11	LTC_ENC	Latch encoder	Determine which encoder is latched when latch source is triggered; The range of encoder no. is 0~3	0
0x12	LTC_LOGIC	Latch logic	Support three kinds of logic modes to trigger the latch process: 0: Only RisingEdge 1: Only FallingEdge 2: Both RisingEdge and FallingEdge	0

ECAT-4XMO, ECAT-TRG4 Latch parameter table

ECAT-4XMO, ECAT-TRG4 Latch parameter table				
NO	Define	Description	Value	Default:
0x10	LTC_IPT	Latch source	Source to trigger the position latch: bit 0~3: Axis 0~3 digital input “LTC”; bit 8~11: PWM pulse out 0~3, refer to TGR_TRG_EN setting	0
0x11	LTC_ENC	Latch encoder	Determine which encoder is latched when latch source is triggered; The range of encoder no. is 0~3	0
0x12	LTC_LOGIC	Latch logic	Support three kinds of logic modes to trigger the latch process: 0: Rising edge. 1: Falling edge. 2: Both rising and falling edge.	0
0x13	LTC_EN	Latch flow enable	Enable position latch process, refer to Field bus position latch functions . 0: Disable, 1: Enable *This parameter would not store at device.	0
0x14	LTC_FIFO_MODE	Latch single point	Bit 0~3: latch channel mode 0: Latch point using FIFO mode. 1: Latch point using single point mode. New point will replace old point. LTC_FIFO_MODE = 0xF -> latch 0~3 using single point mode	0
0x15	LTC_EXTENC_SRC	Latch external encoder source Axis ID. Only could set on	0 ~ 65535 : Axis ID	0

		same 833x master card slave servo motor has encoder pulse output function (ECAT-ECAT-TR G4 only)		

L. Device information table

PCI-8392 (H) Device information		
InfoNo	Information meaning	Format
0x00	Reserved	-
0x10	Driver version	Date
0x20	CPLD version	16 Bits
0x30	PCB version	PCB
0x40	DSP version	Date

PCI-8253/56 Device information					
InfoNo.	Info	Format	InfoNo.	Info	
0x00	Reserved	--	0x01	Reserved	--
0x10	Driver version	Date	0x11	Reserved	--
0x20	CPLD version	16 Bits	0x21	FPGA version	16 Bits
0x30	PCB version (Carrier)	PCB	0x31	PCB Ver.(DB)	PCB
0x40	DSP version	Date	0x41	Reserved	--

PCI-8144 Device information		
InfoNo	Information meaning	Format
0x00	Reserved	-
0x10	Driver version	Date format
0x20	CPLD version	16 Bits

AMP-104C Device information		
InfoNo	Information meaning	Format
0x00	Reserved	-
0x10	Driver version	Date format
0x20	FPGA version	32 Bits

DPAC-1000 Device information		
InfoNo	Information meaning	Format
0x00	Reserved	-
0x10	Driver version	Date
0x20	CPLD version	16 Bits
0x30	PCB version	PCB

DPAC-3000 Device information		
InfoNo	Information meaning	Format
0x00	Reserved	-
0x10	Driver version	Date
0x20	CPLD version	16 Bits
0x30	PCB version	PCB

PCI(e)-7856 Device information		
InfoNo	Information meaning	Format
0x00	Reserved	-
0x10	Driver version	Date format
0x20	CPLD version(PCI-7856) / FPGA version(PCIe-7856)	CPLD : 16 Bits / FPGA : Date format
0x30	PCB version	PCB

MNET-4XMO Device information					
InfoNo.	Info	Format	InfoNo.	Info	
0x00	Reserved	--	0x01	Reserved	--
0x10	Reserved	--	0x11	Reserved	--
0x20	CPLD version	16 Bits	0x21	Reserved	--
0x30	PCB version (Button)	PCB	0x31	PCB Ver.(Top)	PCB

MNET-4XMO-C Device information					
InfoNo.	Info	Format	InfoNo.	Info	
0x00	Reserved	--	0x01	Reserved	--
0x10	Reserved	--	0x11	Reserved	--
0x20	Reserved	--	0x21	FPGA version	16 Bits
0x30	PCB version (Button)	PCB	0x31	PCB Ver.(Top)	PCB

HSL-4XMO Device information					
InfoNo.	Info	Format	InfoNo.	Info	
0x00	Reserved	--	0x01	Reserved	--
0x10	Reserved	--	0x11	Reserved	--
0x20	CPLD version	16 Bits	0x21	Reserved	--
0x30	Reserved	--	0x31	Reserved	--
0x40	DSP version	Date format			

PCI(e)-8154/8158, PCI-8102 Device information					
InfoNo.	Info	Format	InfoNo.	Info	
0x00	Reserved	--	0x01	Reserved	--
0x10	Driver version	Date	0x11	Reserved	--
0x20	CPLD version(Carrier)	16 Bits	0x21	FPGA/CPLD Ver.(DB)	16 Bits
0x30	PCB version (Carrier)	PCB	0x31	PCB Ver.(DB)	PCB

PCI-C154(+) Device information					
InfoNo.	Info	Format	InfoNo.	Info	
0x00	Reserved	--	0x01	Reserved	--
0x10	Driver version	Date	0x11	Reserved	--
0x20	FPGA version(Carrier)	16 Bits	0x21	Reserved	--
0x30	PCB version (Carrier)	PCB	0x31	Reserved	--

EMX-100 Device information					
InfoNo.	Info	Format	InfoNo.	Info	Format
0x00	FMAC software version	Date format	0x01	FMAC middleware version	Date format
0x11	FMAC EXE version	Date format	0x100	FMAC library version	Date format

PCI-8254/58 / AMP-204/8C Device information					
InfoNo.	Info	Format	InfoNo.	Info	
0x00	Reserved	--	0x01	Reserved	--
0x10	Driver version	Date	0x11	Reserved	--
0x20	Reserved	--	0x21	FPGA version	32 Bits
0x30	PCB version (Carrier)	PCB	0x31	Reserved	--
0x40	DSP version	Date	0x41	Reserved	--

PCIe-833x Device information					
InfoNo.	Info	Format	InfoNo.	Info	
0x00	Reserved	--	0x01	Reserved	--
0x10	Driver version	Date	0x11	Reserved	--
0x20	Reserved	--	0x21	FPGA version	32 Bits
0x30	Product and PCB version (Carrier)	PRO,PCB	0x31	Reserved	--
0x40	Kernel version	Date	0x41	Reserved	--

Format description:

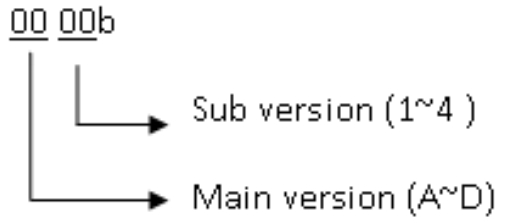
- **Date format:** 32 bit value

Value = YYMMDD; Y:year, M:month, D:day

Eg. Driver version = 80212. 2008/2/12 release.

- **Product format and PCB format:** 4 bits value.

00 00b = PCB A1 version

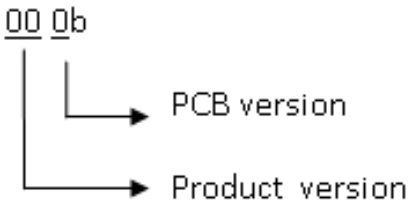


Dec.	Bin	Version	Dec.	Bin	Version
0	0000b	A1	8	1000b	C1
1	0001b	A2	9	1001b	C2
2	0010b	A3	10	1010b	C3
3	0011b	A4	11	1011b	C4
4	0100b	B1	12	1100b	D1
5	0101b	B2	13	1101b	D2
6	0110b	B3	14	1110b	D3
7	0111b	B4	15	1111b	D4

- **16 Bits Format:** 16 bit value (1 ~ 255)

For PCIe-833x:

- **Product format and PCB format (3 bits):**
  - PCB format: bit 0 value, 0b: A1, 1b: A2.
  - Product format: bit 1 ~ 2 value, 00b: PCIe-8338, 01b: PCIe-8332, 10b: PCIe-8334



Dec.	Bin	Version
0	000b	PCIe-8338, A1
1	001b	PCIe-8338, A2
2	010b	PCIe-8332, A1
3	011b	PCIe-8332, A2
4	100b	PCIe-8334, A1
5	101b	PCIe-8334, A2
6		
7		

For PCI-8254/58 / AMP-204/8C and PCIe-833x:

- **32 Bits Format:** 32 bit value

M. Field bus slave parameter table

HSL-DI16-UL				
CH NO.	PA NO.	Description	Value	Default :
-1	0x0000	Enable / Disable stretch (latch) function for all channels. **(Set only)**	0: Enable 1: Disable	1
-1	0x0001	Set stretch (latch) duration for all channels. **(Set only)**	0 ~ 127 (ms) 0: No stretch.	0
0 ~ 15	0x0000	Set / Get stretch (latch) function for each channel.	0: Enable 1: Disable	1
0 ~ 15	0x0001	Set / Get stretch (latch) duration for each channel.	0 ~ 127 (ms) 0: No stretch.	0

HSL-AI16AO2				
CH NO.	PA NO.	Description	Value	Default :
-1	0x0000	Set / Get analog input range.	0: +/- 10V 1: +/- 5V 2: +/- 2.5V 3: +/- 1.25	0
-1	0x0001	Set / Get last scan analog input channel	0 ~ 15	15
-1	0x0002	Enable / Disable analog input (AD converator) **(Set only)**	0: Disable 1: Enable	0

HSL-AO4				
CH NO.	PA NO.	Description	Value	Default:
-1	0x0000	Set / Get keep mode. (Value format :Bit format) Keep Enable means that ananlg output will be keeped when communication is broken. Set 0 enable all channels keep mode. Set 0xF disable all channels keep mode.	Bit ON: Disable Bit OFF: Enable Bit 0~3: Ch 0 ~ Ch3	0

N. DPAC display index table

APS_get_display_data() and APS_set_display_data() reference table.

For alphabet type, users can use one of three values to disaply it. For example, for letter ‘A’, users can set 0x0A, 0x41 or 0x61 to display it.

7-Segment LED results	* displayIndex displayIndex	displayIndex	displayIndex
‘0’	0x00	0X30(ASCII’0’)	
‘1’	0x01	0X31(ASCII’1’)	
‘2’	0x02	0X32(ASCII’2’)	
‘3’	0x03	0X33(ASCII’3’)	
‘4’	0x04	0X34(ASCII’4’)	
‘5’	0x05	0X35(ASCII’5’)	
‘6’	0x06	0X36(ASCII’6’)	
‘7’	0x07	0X37(ASCII’7’)	
‘8’	0x08	0X38(ASCII’8’)	
‘9’	0x09	0X39(ASCII’9’)	
‘A’	0x0A	0X41(ASCII’A)	0X61(ASCII’a’)
‘b’	0x0B	0X42(ASCII’B’)	0X62(ASCII’b’)
‘C’	0x0C	0X43(ASCII’C’)	0X63(ASCII’c’)
‘d’	0x0D	0X44(ASCII’D’)	0X64(ASCII’d’)
‘E’	0x0E	0X45(ASCII’E’)	0X65(ASCII’e’)
‘F’	0x0F	0X46(ASCII’F’)	0X66(ASCII’f’)
‘G’	0x10	0X47(ASCII’G’)	0X67(ASCII’g’)
‘H’	0x11	0X48(ASCII’H’)	0X68(ASCII’h’)
‘i’	0x12	0X49(ASCII’I’)	0X69(ASCII’i’)
‘j’	0x13	0X4A(ASCII’J’)	0X6A(ASCII’j’)
‘K’	0x14	0X4B(ASCII’K’)	0X6B(ASCII’k’)
‘L’	0x15	0X4C(ASCII’L’)	0X6C(ASCII’l’)
‘M’	0x16	0X4D(ASCII’M’)	0X6D(ASCII’m’)
‘n’	0x17	0X4E(ASCII’N’)	0X6E(ASCII’n’)
‘o’	0x18	0X4F(ASCII’O’)	0X6F(ASCII’o’)
‘p’	0x19	0X50(ASCII’P’)	0X70(ASCII’p’)
‘q’	0x1A	0X51(ASCII’Q’)	0X71(ASCII’q’)
‘r’	0x1B	0X52(ASCII’R’)	0X72(ASCII’r’)

‘S’	0x1C	0X53(ASCII’S’)	0X73(ASCII’s’)
‘t’	0x1D	0X54(ASCII’T’)	0X74(ASCII’t’)
‘U’	0x1E	0X55(ASCII’U’)	0X75(ASCII’u’)
‘v’	0x1F	0X56(ASCII’V’)	0X76(ASCII’v’)
‘W’	0x21	0X57(ASCII’W’)	0X77(ASCII’w’)
‘X’	0x22	0X58(ASCII’X’)	0X78(ASCII’x’)
‘Y’	0x23	0X59(ASCII’Y’)	0X79(ASCII’y’)
‘Z’	0x24	0X5A(ASCII’Z’)	0X7A(ASCII’z’)
‘0.’	0x25		
‘1.’	0x26		
‘2.’	0x27		
‘3.’	0x28		
‘4.’	0x29		
‘5.’	0x2A		
‘6.’	0x2B		
A‘7.’	0x2C		
‘8.’	0X2D		
‘9.’	0X2E		
‘ ’	0X2F 0X20	0X20(ASCII’ ’)	

O. DPAC button status table

ON in the table means pushed.

Example Steps – check B3 ON/OFF

1) Read button status
2) To get a new button status by ‘NOT’ button status
3) Maps B3 to Bit# by “Bit#=(4 - B#)’. We get Bit1.
4) Use Bit1 (0010b) to ‘AND’ new button status
5) If the result is zero, it means B3 is not pushed.
6) If the result is non-zero, it means B3 is pushed.

Button status	B1 (Bit3)	B1 (Bit3)	B1 (Bit3)	B1 (Bit3)
0x0F	OFF	OFF	OFF	OFF
0x0E	OFF	OFF	OFF	ON
0x0D	OFF	OFF	ON	OFF
0x0C	OFF	OFF	ON	ON
0x0B	OFF	ON	OFF	OFF
0x0A	OFF	ON	OFF	ON
0x09	OFF	ON	ON	OFF
0x08	OFF	ON	ON	ON
0x07	ON	OFF	OFF	OFF
0x06	ON	OFF	OFF	ON
0x05	ON	OFF	ON	OFF
0x04	ON	OFF	ON	ON
0x03	ON	ON	OFF	OFF
0x02	ON	ON	OFF	ON
0x01	ON	ON	ON	OFF
0x00	ON	ON	ON	ON

P. SSCNET servo monitor source table

Monitor Source NO.	Content	Units	Note (bytes)
0	Position feeback	Pulse	4
1	Position droop	Pulse	4
2	Speed feedback	0.01 r/min	4
3	Electrical current feedback (torque)	0.1%	2 Bytes
4	Instataneous with-in one revolution position	Pulse	4 Bytes
5	Origenal position with-in one revolution	Pulse	4 Bytes
6	ZCT	Pulse	4 Bytes
7	Instataneous position encoder pulse/rev counter.	rev	2 Bytes
8	Origenal position encoder pulse/rev counter.	rev	2 Bytes
9	Bus voltage	V	2 Bytes
10	Regenerative load factor	%	2 Bytes
11	Effective load ratio	%	2 Bytes
12	Ratio of load inertia monemt to servo motor inertia moment	Times	2 Bytes
13	Position loop gain	Rad/s	2 Bytes
14	Alarm/warning number		
15	Alarm details bit		
16	Parameter number		
17	Alarm status (AL10~AL1F)		
18	Alarm status (AL20~AL2F)		
19	Alarm status (AL30~AL3F)		
20	Alarm status (AL40~AL4F)		
21	Alarm status (AL50~AL5F)		
22	Alarm status (AL60~AL6F)		
23	Alarm status (AL70~AL7F)		
24	Alarm status (AL80~AL8F)		
25	Alarm status (AL90~AL9F)		

26	Alarm status (ALE0~ALEF)		

Q. VAO parameter table

PCI-8253/56 VAO parameter table				
NO	Define	Description	Value	Default:
0x00 + (2 * N) Note: N is TableNo, range is 0 ~ 7. (*3)	VAO_TABLE_OUTPUT_TYPE	Table output type(*1)	0: Voltage 1: PWM mode 2: PWM frequency mode with fixed width 3. PWM frequency mode with fixed duty cycle	1
0x01 + (2 * N) Note: N is TableNo, range is 0 ~ 7. (*3)	VAO_TABLE_INPUT _TYPE	Table input type	0: Feedback speed 1: Command speed	0
0x10 + N Note: N is TableNo, range is 0 ~ 7. (*3)	VAO_TABLE_PWM_Config	Configure PWM according to output type.	a. Mode 0 - Don't care b. Mode 1 - set a fixed frequency (1 ~ 25M Hz) c. Mode 2 - set a fixed Pulse Width (40 ~ 335544340 ns) d. Mode 3 – set a fixed duty cycle: N * 0.05 %. (N: 1 ~ 2000)	100
0x20 + N Note: N is TableNo, range is 0 ~ 7. (*3)	VAO_TABLE_SRC	Specify axisID for VAO table. (linear speed on multi- axes)(*2)	Bit0: Axis 0 On Bit1: Axis 1 On Bit2: Axis 2 On Bit3: Axis 3 On	0x01
0x30(*4)	Reserved	Reserved	Reserved	
0x40	VAO_DO_DELAY_TIME	Specify a delay time for Do output when	0: No delay time. N: Delay time is N*DSP cycle. For PCI-8253, DSP	0

		point table is running.	cycle is 400 us. For PCI-8256, DSP cycle is 500 us.	
0x50~	Reserved			

(*1): PCI-8253 don't support voltage mode.

(*2): PCI-8253 supports 3 axes. Bit 0, bit 1 and bit 2 are available.

(*3): Vao supports 8 tables. Each table has own parameter setting. For example, user could use 0x00 to set table output type to table0 and use 0x02 to set output type to table1. For another example, user could use 0x20 to specify axis id for table 0 and use 0x21 to specify axis id for table 1.

(*4): A parameter named VAO_TABLE_TARGET(0x30), used to set output channel, is taken off because of supporting multi-table design. By new design, user could set output channel by APS_start_vao(). Refer to APS_start_vao().

PCI-8254/58 / AMP-204/8C VAO parameter table				
NO	Define	Description	Value	Default:
0x00 + (2 * N) Note: N is TableNo, range is 0 ~ 7. (*1)	VAO_TABLE_OUTPUT_TYPE	Table output type(*1)	0: Voltage(reserved) 1: PWM mode 2: PWM frequency mode with fixed width 3. PWM frequency mode with fixed duty cycle	1
0x01 + (2 * N) Note: N is TableNo, range is 0 ~ 7. (*1)	VAO_TABLE_INPUT _TYPE	Table input type	0: Feedback speed 1: Command speed	0
0x10 + N Note: N is TableNo, range is 0 ~ 7. (*1)	VAO_TABLE_PWM_Config	Configure PWM according to output type.	a. Mode 0 - Don't care b. Mode 1 - set a fixed frequency (3 ~ 50M Hz) c. Mode 2 - set a fixed Pulse Width (20 ~ 335544300 ns) d. Mode 3 – set a fixed duty cycle: N * 0.05 %.	100

			(N: 1 ~ 2000)	
0x20 + N Note: N is TableNo, range is 0 ~ 7. (*1)	VAO_TABLE_SRC	Specify axisID for VAO table. (linear speed on multi- axes)	Bit0: Axis 0 On Bit1: Axis 1 On Bit2: Axis 2 On Bit3: Axis 3 On	0x01
0x30	Reserved	Reserved	Reserved	

(*1): Vao supports 8 tables. Each table has own parameter setting. For example, user could use 0x00 to set table output type to table0 and use 0x02 to set output type to table1. For another example, user could use 0x20 to specify axis id for table 0 and use 0x21 to specify axis id for table 1.

37. APS Functions Return Code

The following table provides a list of possible return value in APS library. If the return value is a negative value, it means there are some errors or warning occurred.We provide C/C++ standard header file, “ErrorCodeDef.h”, which define all errors return value.

A. APS Error Code Table

Code	Define	Error descriptions and items to check
0	ERR_NoError	Success, No error
-1	ERR_OSVersion	Operating system version error. The current operating system you used are not supported by this function.
-2	ERR_OpenDriverFailed	Open driver failed. Create driver interface failed. Check device driver is installed correctly. Check devices are installed correctly in your system.
-3	ERR_InsufficientMemory	System memory insufficiently. There is not enough memory in your system.
-4	ERR_DeviceNotInitial	The Device or the card is not be initialized. Check the card ID The device has been closed The device is not be initialized.
-5	ERR_NoDeviceFound	Devices not found Check device driver is installed correctly. Check devices are installed correctly in your system.
-6	ERR_CardIdDuplicate	Card ID duplicated. Check the card ID settings (SW jump) Check the parameter of initial function is correctly.
-7	ERR_DeviceAlreadyIntialed	The devices have already been initialed. 1. Check the close card function is work

		correctly.
-8	ERR_InterruptNotEnable	Interrupt events not be enabled. 1. Enable the hardware interrupt. 2. Check the interrupt factor is set correctly.
-9	ERR_TimeOut	Function timeout.
-10	ERR_ParametersInvaild	The value of the parameters is incorrect. Check the setting range of parameters. Compare the setting value of parameters with user manual.
-11	ERR_SetEEPROM	Hardware memory write error.
-12	ERR_GetEEPROM	Hardware memory read error.
-13	ERR_FunctionNotAvailable	The function is not available in current stage. The device is not support this function. System is in error state. 1. Check the function library. 2. Check the hardware connection (servo drive connection) 3. Reinitial(Reboot) the system.
-14	ERR_FirmwareError	Firmware process error. 1. Check the firmware version.
-15	ERR_CommandInProcess	The previous command is in process.
-16	ERR_AxisIdDuplicate	Axes' ID is duplicated.
-17	ERR_ModuleNotFound	Slave module not found.
-18	ERR_InsufficientModuleNo	System ModuleNo insufficiently
-19	ERR_HandShakeFailed	HandSake with the DSP out of time.
-20	ERR_FILE_FORMAT	Config file format error.(cannot be parsed)
-21	ERR_ParametersReadOnly	Function parameters read only.
-22	ERR_DistantNotEnough	Distant is not enough for motion.
-23	ERR_FunctionNotEnable	Function not yet enabled
-24	ERR_ServerAlreadyClose	Server already closed
-25	ERR_DllNotFound	Could’t find virtual DLL
-32	ERR_DllFuncFailed	Could’t find specified function on virtual DLL
-33	ERR_FeederAbnormalStop	Feeder abnormally stop
-40	ERR_DoubleOverflow	Double format parameter is overflow

-41	ERR_SlaveNumberErr	Slave ID number error.
-42	ERR_SlaveStatusErr	The status of EtherCAT slave is error
-43	ERR_OverRange	The input data over assigned range
-46	ERR_LatchFlowErr	Execute latch function error
-48	ERR_InServoOnState	Axis is in servo on state would cause error. (*1)
-901	ERR_INIT_ERROR	Initialization error
-902	ERR_NO_INIT	No initialization
-903	ERR_NO_SETUP	No input parameters setup
-904	ERR_INPUT_ERROR	Error input parameters
-905	ERR_STATUS_NOT_READY	PC status is not ready
-906	ERR_AXIS_BUSY	The state of specified axis is busy
-907	ERR_NETWORK_ERROR	Network connection has error
-908	ERR_NETWORK_TIME_OUT	Network connection is time-out
-909	ERR_CRC_FAIL	Checking CRC is failed
-910	ERR_PARAM_INVALID	Invalid parameter setting for EMX-100
-911	ERR_NO_SERVO_ON	Not in servo on status
-912	ERR_API_TIMEOUT	API is timeout (150ms)
-913	ERR_EXE_GET_STATUS_TIMEOUT	Asynch status update timeout
-915	ERR_LOAD_XML_MISMATCH	Mismatch configuration of Board ID and Axis ID in XML file(*2)
-1000	ERR_Win32Error	No such event number, or WIN32_API error, contact with ADLINK's FAE staff.
-1001	ERR_NoENIFile	Generating ADLINK_Config2.xml file (ENI file) is failed.
-1002	ERR_TimeOut_SetVoltageEnable	Set voltage enable time out with servo on process.
-1003	ERR_TimeOut_SetReadyToSwitch	Set ready to switch time out with servo on process.
-1004	ERR_TimeOut_SetShutdown	Set shut down time out with servo on process.
-1005	ERR_TimeOut_SetSwitchOn	Set switch on time out with servo on process.
-1006	ERR_TimeOut_SetOperationEnable	Set operation enable time out with servo on process.
-1007	ERR_RegistryPath	System registry path fails or no registry.
-1008	ERR_MasterNotOPState	Master is not in OP state.

-1009	ERR_SlaveNotOPState	Slave is not in OP state.
-1010	ERR_SlaveTotalAxisNumber	The scanned number of EtherCAT slaves' axes exceeds the number of maximum axes that PCIe-8334/8 can support.
-1011	ERR_MissESIFileOrMissENIPath	Miss ESI file or ENI path.
-1012	ERR_MissConfig_1_Xml	Miss Config_1 xml file.
-1013	ERR_CopyConfig_1_Xml_fail	Copy Config_1 xml file fail.
-1014	ERR_MissConfig_2_Xml	Miss Config_2 xml file.
-1015	ERR_CopyConfig_2_Xml_fail	Copy Config_2 xml file fail.
-1016	ERR_InvalidSlaveLocalAxis	Invalid slave local axis
-1017	ERR_InvalidECATHomeMode	Invalid home mode
-1018	ERR_FoEFileNameOverLimit	File name length over 32 bytes.
-1019	ERR_FoEFileVerifyError	File compare result failed.
-1020	ERR_FoEConflictAutoRecovery	Auto recovery function is enabled. Must “DISABLE” auto recovery function before using FoE download.
-1021	ERR_FoEFileSizeOverLimit	FoE download file is too large. The limitation is 10M bytes.

(*1): Now only used in PCI(e)-8154/58 APS_spiral_ce_xxx function. Due to 8154/58 limit, 4 th / 8 th axis operation will be a dummy motion and it can’t be used for any other puspose. This axis need to be set servo-off. If not, it will return ERR_InServoOnState

(*2) :If used two or more EMX-100 device.Please save Board ID and Axis ID in the same XML file.

B. DSP motion kernel error code

Code	Define	Error descriptions and items to check
-2001	MKERR_AXIS_INDEX	Axis range error
-2002	MKERR_CHANNEL_INDEX	Channel range error
-2003	MKERR_PARA_UNDEFINE	Parameter number undefine
-2004	MKERR_PARA_FAULT	Parameter data is wrong
-2005	MKERR_STATE_UNAVAILABLE	This state cannot do this thing. (command position only can be set when axis is in idle)
-2006	MKERR_CHECKSUM	Checksum error (Internal error),(Check code error)
-2007	MKERR_TIME_OUT	Timeout error
-2008	MKERR_MEM_TEST	Memory test error
-2009	MKERR_CTRL_CMD	Unknown command or this state cannot accept this command
-2010	MKERR_AXES_DIMENSION	The dimension of the axes is invalid.
-2011	MKERR_MBUF_FULL	Motion buffer is full
-2012	MKERR_NO_AVAILABLE_SPG	This function cannot be accept when axes are in blending, (all spgs are busy)
-2013	MKERR_BLEND_PERCENT	The transition parameter " percent " is out of range.
-2014	MKERR_TRANSITION_MODE	Transition mode is undefine or not support.
-2015	MKERR_COORD_TRANS_INDEX	Transform matrix column index > MAX_AXES (invaild)
-2016	MKERR_SINP_WIDTH	Soft-inp width invaild (>= 0)
-2017	MKERR_SINP_STABLE_TIME	Soft-inp stable time invaild (< 65535)
-2018	MKERR_GANTRY_MASTER	Gantry master axis invaild
-2019	MKERR_FCMP_SIZE	The size of compare date is over range.
-2020	MKERR_VS_INVALID	Start velocity invalid, (vs >= 0)
-2021	MKERR_VE_INVALID	End velocity invalid. (ve >= 0)
-2022	MKERR_VM_INVALID	Maximum velocity invalid. (vm > 0)
-2023	MKERR_ACC_INVALID	Acceleration invalid. (acc > 0)
-2024	MKERR_DEC_INVALID	Deceleration invalid. (dec > 0)
-2025	MKERR_S_INVALID	S invalid. (0 <= S <= 1)
-2026	MKERR_SD_DEC_INVALID	SD Dec invalid. (>0)

-2027	MKERR_AXES_OVERLAPPING	Axes number cannot overlapping.
-2028	MKERR_BLEND_DISTANCE	The transition parameter "ResidueDistance" cannot < 0.0.
-2029	MKERR_ERROR_POS_LEVEL	Error position check level must >= 0.0
-2030	MKERR_WAIT_MOVE	Wait mode not accept when axis in moving
-2031	MKERR_AX_DISABLE	Axis is disable (Servo off)
-2032	MKERR_AX_ERROR	Axis is in error state (AX_ERR_STOPPING/AX_ERR_STOPPED). you should reset the error state.
-2033	MKERR_AX_MOVING	Axis is in moving and command cannot accept (cannot overwrite), axes must same dimension, same axes
-2034	MKERR_PRE_EVENT_DIST	pre-event distance must >= 0
-2035	MKERR_POST_EVENT_DIST	post-event distance must >= 0
-2040	MKERR_ARC_PARA	This function cannot accept half arc or full arc parameter
-2041	MKERR_ARC_FINAL_R	The finalR cannot be negative value.
-2042	MKERR_ARC_NORMAL	Normal vector invaild. or arc parameters invaild.
-2050	MKERR_GANTRY_DEV_PROTECT	Gantry deviation protect value must >= 0
-2051	MKERR_INVAILD_IN_GANTRY	This command is not allow in gantry mode.
-2052	MKERR_INVAILD_GEAR_MASTER	Gear master is not define
-2053	MKERR_ENGAGE_RATE	
-2054	MKERR_GEAR_RATIO	
-2055	MKERR_GEAR_ENABLE_MODE	
-2056	MKERR_GEAR_LOOP	Cannot be a gear loop.
-2057	MKERR_JOG_OFFSET	The jog offset paremeter is wrong. must > 0
-2062	MKERR_DI_GROUP	DI group number is wrong.
-2063	MKERR_DI_CH	DI channel number is wrong.
-2070	MKERR_FILTER_COEFFICIENT	Filter coefficient is wrong
-2071	MKERR_FBK_VEL_COEFFICIENT	
-2080	MKERR_PTBUFF_AXIS_NOT_IDLE	Axis (Axes) are now in moving state, cannot start point buffer
-2081	MKERR_PTBUFF_DIMENSION	Invalid axes dimension setting
-2082	MKERR_PTBUFF_AXIS_IN_USE	Axes number are already in use by another PTB
-2083	MKERR_PTBUFF_NOT_ENABLE	PTBUFF not enable

-2084	MKERR_PTBUFF_FULL	PTBUFF is full.
-2085	MKERR_PTBUFF_CURVE_TYPE	Invalid motion curve type
-2086	MKERR_PTBUFF_DIMENSION_MISS	Miss mach line move dimension and PTBUFF dimension
-2087	MKERR_PTBUFF_CURVE_DIMENSION	Curve type and it's dimension miss match
-2088	MKERR_PTBUFF_ABNORMAL_STOP	Axes abnornmal stop, please check axis stop code.
-2089	MKERR_PTBUFF_M_QUEUE_EXCEPTION	Axes motion queue exception.
-2090	MKERR_PTBUFF_AXIS_INDEX_INVAILD	Target axes index is over PTBUFF dimension
-2091	MKERR_PTBUFF_ID	Invalid PTBUFF ID. Check the input parameter.
-2092	MKERR_PTBUFF_CTRL_COMMAND	Invalid PTBUF control command. Check the input parameter.
-2093	MKERR_PTBUFF_AXES_IN_MOTION	PTBUFF detected axes are in motion when start PTBUFF.
-2094	MKERR_PTBUFF_EXT_COMMANDS_NUM	PTBUFF extra-commands must < = 7
-2095	MKERR_PTBUFF_EXT_COMMAND_EMPTY	Execute extra command but command queue is empty.
-2096	MKERR_PTBUFF_EXT_COMMAND_FULL	Push extra command but command queue is full.
-2097	MKERR_PTBUFF_EXT_COMMNAD	Invalid extra command code.
-2098	MKERR_PTBUFF_NOT_STOPPED	Invalid command when point buffer is running.
-2099	MKERR_PTBUFF_DWELL_TIME	Dwell time must >= 0
-2100	MKERR_HS_UNKNOWN_CMD	Handshake data error. command unknown.
-2101	MKERR_HS_SIZE	Handshake data error. size or dimension invaild.
-2102	MKERR_HS_SUB_ERRORS	Sub functions have errors. please check sub return code
-2201	MKERR_DSP_SYSTEM_CONFIG	DSP initialize error. (Call adlink R&D staff)
-2208	MKERR_LOAD_CALIBRATION_DATA	Load calibration data failed.
-2209	MKERR_DPRAM_TEST_FAILED	DPRAM hardware test failed. (Call adlink staff)
-2210	MKERR_FPGA_VERSION	FPGA version outdated. (Call adlink staff)
-2211	MKERR_PSC_TIME_OUT	
-2212	MKERR_PLL_TIME_OUT	
-2213	MKERR_PSC_PARAM_ERR	

-2300	MKERR_MOTION_LOOP_TIMING	The timing of motion loop is out of range.
-2401	MKERR_WRITE_ROM	
-2402	MKERR_READ_ROM	
-2403	MKERR_REF_5V	
-2404	MKERR_SET_AI_OFFSET	
-2405	MKERR_SET_AI_GAIN	
-2406	MKERR_SET_AO_OFFSET	
-2407	MKERR_SET_AO_GAIN	
-2408	MKERR_NO_CALIB	No calibration data in EEPROM. (Not all gain/offset of AO/AI are calibrated)
-2409	MKERR_DATA_SIZE	
-2410	MKERR_BACKDOOR_PWD	Backdoor password wrong.
-2411	MKERR_ROM_PROG_SIZE	(Flash) Byte count of data to much or Offset is over range.
-2500	MKERR_OVER_QUEUE_SIZE	Queue size is less than input array
-2600	MKERR_FRP_STATE	State of frequency response process is invalid
-2601	MKERR_RCP_STATE	State of relay control process is invalid
-2700	MKERR_TASK_NUM	Task number is wrong
-2701	MKERR_UNKNOWN_PROGRAM_REGISTER	Unknow progarm register.
-2702	MKERR_CANNOT_SET_REG _WHEN_PG_NOT_STOP	Program is runing, you cannot issue this command
-2703	MKERR_OVER_STACK_SIZE	Over stack size range
-2800	MKERR_ACCESS_UNDEFINE	Access type is undefined
-2801	MKERR_ACCESS_DENIED	Parameter access is denied since it is in save/load process
-2802	MKERR_ERASE_SECTION	Erase section is failed due to time out
-2803	MKERR_LAST_MARK_INVALID	Load data error; ; last mark in flash is error
-2804	MKERR_CHK_PARITY	Load data error; parity bit is error
-2805	MKERR_OVER_DATA_TYPE	Load data error; over max data type define
-2806	MKERR_OVER_PA_TYPE	Load data error; over max parameter type define
-2807	MKERR_OVER_MAX_BOARD	Load data error; over max board parameter define
-2808	MKERR_OVER_MAX_AXIS	Load data error; over max axis parameter define
-2900	MKERR_WDT1_STATE	Watchdog timer 1 has been enabled

-2901	MKERR_WDT1_PERIOD	Max reset period of watchdog timer 1 is out of range
-2903	MKERR_TIME_INTERVA	Time interval between two points of PVT/PT motion is a negative value or zero. For example: t0 = 0.0sec, t1 = 0.5sec, t2 = 0.1sec

C. EtherCAT Master Error Code

Code	Define	Error descriptions and items to check
-4001	EC_INIT_MASTER_ERR	Initialized EtherCAT master error
-4011	EC_GET_SLV_NUM_ERR	Get total slave number error
-4012	EC_CONFIG_MASTER_ERR	Configure EtherCAT master error
-4013	EC_BUSCONFIG_MISMATCH	Topology information is not match with current data
-4014	EC_CONFIGDATA_READ_ERR	Read configuration data error
-4015	EC_ENI_NO_SAFEOP_OP_SUPPORT	Don't support safeop and op state
-4021	EC_CONFIG_DC_ERR	Configure DC parameter error
-4022	EC_DCM_MODE_NO_SUPPORT	The dcm mode is not support
-4023	EC_CONFIG_DCM_FEATURE_DISABLED	The dcm feature was disabled
-4024	EC_CONFIG_DCM_ERR	Configure DCM parameter error
-4031	EC_REG_CLIENT_ERR	Register client error
-4041	EC_SET_INIT_STATE_ERR	Set EtherCAT master to initial state error
-4042	EC_SET_PREOP_STATE_ERR	Set EtherCAT master to preop state error
-4043	EC_SET_SAFEOP_STATE_ERR	Set EtherCAT master to safeop state error
-4044	EC_SET_OP_STATE_ERR	Set EtherCAT master to op state error
-4051	EC_DE_INIT_MASTER_ERR	Deinitialized EtherCAT master error
-4061	EC_ENI_FOPEN_ERR	Can't open ENI information
-4062	EC_ENI_FREAD_ERR	Can't read ENI information
-4063	EC_GEN_EBI_BUSSCAN_ERR	Scan the EtherCAT bus error
-4081	EC_WRONG_PORT_NO	Input wrong EtherCAT master port number
-4091	EC_GET_SLAVE_INFO_ERR	Get slave information error
-4101	EC_COE_SDO_UPLOAD_ERR	Execute sdo upload input data error
-4102	EC_COE_SDO_HOME_MODE_ERR	Input ECAT servo home mode error or wrong data.
-4103	EC_COE_SDO_HOME_ACCDEC_ERR	Input ECAT servo home ACC/DEC error or wrong data.

-4104	EC_COE_SDO_HOME_VM_SWITCH_ERR	Input ECAT servo home limit switch error or wrong data.
-4105	EC_COE_SDO_HOME_VM_ZERO_ERR	Input ECAT servo home zero error or wrong data.
-4106	EC_COE_SDO_HOME_OFFSET_ERR	Input ECAT servo home offset error or wrong data.
-4107	EC_CONTROL_WORD_HOME_ERR	Input ECAT servo home control word error or wrong data.
-4108	EC_COE_SDO_STOP_ERR	Input ECAT servo home stop error or wrong data.
-4109	EC_CONTROL_WORD_STOP_ERR	Input ECAT servo home limit switch error or wrong data.
-4110	EC_SET_OP_MODE_HOME_ERR	Set OP mode error with ECAT home process.
-4201	EC_WRONG_SLAVE_NO	Input slave number is over the limit
-4202	EC_WRONG_MODULE_NO	Input module number is over the limit
-4203	EC_WRONG_AI_CHANNEL_NO	Input AI channel number is over the limit
-4204	EC_WRONG_AO_CHANNEL_NO	Input AO channel number is over the limit
-4205	EC_COE_SDO_DOWNLOAD_ERR	Execute sdo download input data error
-4301	EC_COE_OD_INIT_ERR	Create memory error when initialized object dictionary
-4302	EC_COE_GET_OD_NUM_ERR	Get object dictionary number error
-4303	EC_COE_GET_OD_NUM_LAST	The input object dictionary number is the last
-4304	EC_COE_GET_OD_DESC_ERR	Get object dictionary description error
-4305	EC_COE_GET_OD_DESC_ENTRY_ERR	Get object dictionary description entry error
-4306	EC_COE_GET_OD_STATUS_PEND	Get object dictionary is pending
-4405	EC_PDO_ACCESS_OFFSET_ERR	PDO offset is invalid.
-4501	EC_GET_SLAVE_ID_ERR	Get slave ID failed.
-4502	EC_SET_SLAVE_ID_ERR	Set slave ID failed.
-4503	EC_DUPLICATE_SLAVE_ID_ERR	The slave ID number duplicate occurrence
-4504	EC_GET_SLAVE_REGISTER_ERR	Get slave register error
-4505	EC_SET_SLAVE_REGISTER_ERR	Set slave register error

-4506	EC_GEN_ENIFILE_BUFFER_ERR	Generate ENI file error while start field bus.
-4507	EC_LOAD_ENIFILE_FLASH_ERR	Load ENI file error while loading from flash.
-4600	EC_FOE_FILE_NAME_NULL	FoE download file name is null.
-4601	EC_FOE_FILE_OPEN_FAIL	FoE file open failed
-4602	EC_FOE_FILE_MEMORY_ALLOCATE_FAIL	Memory allocate failed when creating memory for FoE file.
-4603	EC_FOE_FILE_COPY_DATA_FAIL	Copy FoE file content to buffer failed.
-4604	EC_FOE_NO_MBX_SUPPORT	Slave module without mailbox support
-4605	EC_FOE_FILE_DELETE_FAIL	FoE file delete failed.
-4606	EC_FOE_DOWNLOAD_TIME_OUT	FoE file download time out
-4607	EC_FOE_DOWNLOAD_FILE_NOT_VALID	Slave module not support FoE download. Without “Bootstrap” state.
-4608	EC_FOE_DOWNLOAD_PWD_NOT_VALID	FoE download password not valid.