APS Function Library V1.2
Build Date 2014.02.05

Support Products: PCI-8253/56, PCI-8392(H), DPAC-1000, DPAC-3000, PCI-7856, MNET-4XMO-(C), MNET-1XMO, HSL-DIO, PCI-8144, HSL-4XMO, PCI-8154/58/02, PCI-8158A

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

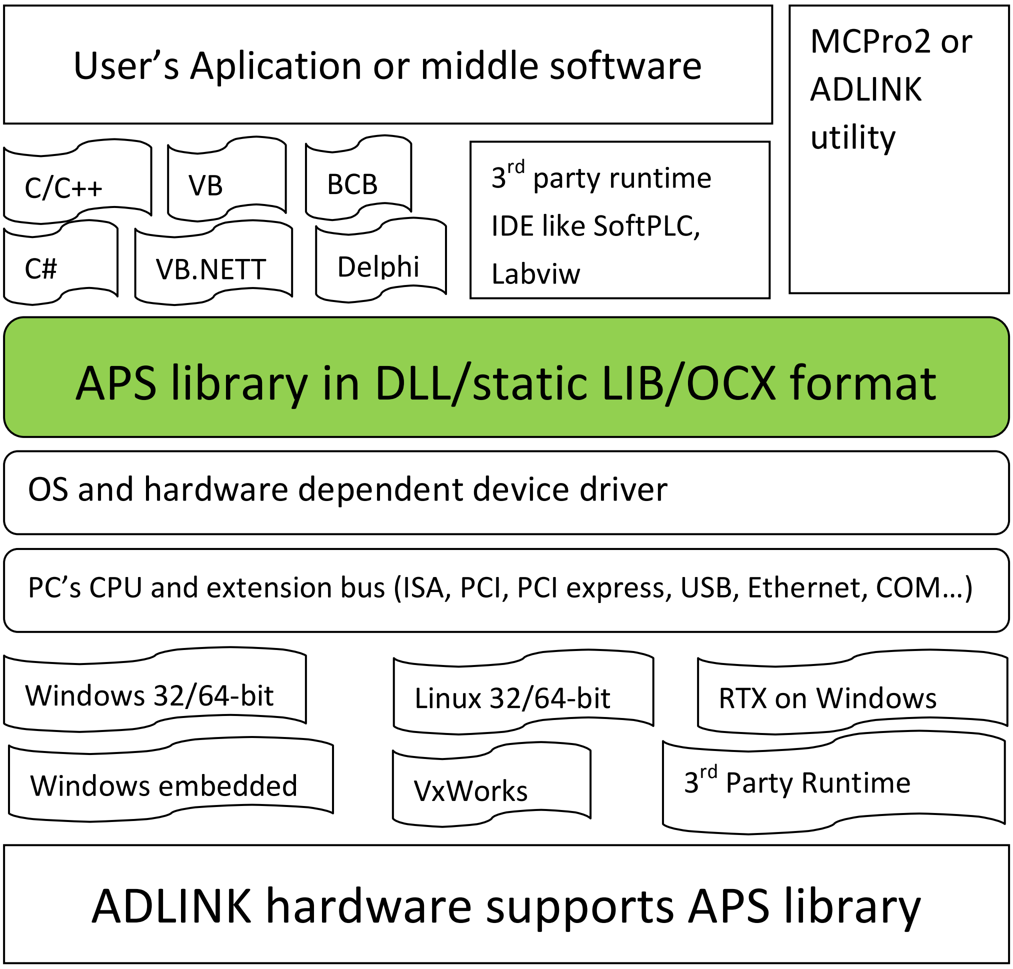
The first benefit is hardware independent. In the past, each product has its own software function set. Every time users want to add or remove different kinds of product even for the same purpose, they must re-program their software to fit it. Most of time, they must re-study new function usage. That’s 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

1. All functions List

Sec.	Function name	Descriptions	Page
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_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_set_security_key	Set security password	
	APS_check_security_key	Varify security password	
	APS_reset_security_key	Reset security password	
	APS_save_param_to_file	Save parameters to file	
	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	
	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_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	
7	**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	
8	**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	
9	**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.	
10	**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	
11	**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	
12	**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_move 2	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_start_feeder_move	Start point table move and feed points.	
	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.	
13	**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_axis no	Get first axis of the slave module	
	APS_get_field_bus_device_info	Get device information on a specified field bus	
14	**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	
15	**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_set_trigger_encoder_counter	Set trigger encoder counter	
	APS_get_trigger_encoder_counter	Get trigger encoder 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 status	
	APS_start_timer	Start timer	
16	**Manual Pulse Geneator functions**		
	APS_get_pulser_counter	Get pluse input counter	
	APS_set_pulser_counter	Set pluse input counter	
17	**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	
18	**NV RAM funciton**		
	APS_set_nv_ram	Set RAM data	
	APS_get_nv_ram	Get RAM data	
	APS_clear_nv_ram	Clear RAM data	
19	**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	
20	**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	
21	**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	
22	**Single-latch functions**		
	APS_manual_latch2	Manual latch for a axis	
	APS_get_latch_data2	Get latch data for a axis	
	Table definition		
	Board parameters definition table		
	Axis parameters definition table		
	Sampling parameters definition table		
	Sampling source definition table		
	The bit definition of motion IO status		
	Motion status definition table		
	Interrupt factor table		
	Field bus parameter definition		
	Gantry parameters definition table		
	Trigger parameter table		
	Device information table		

2. *List of functions for DPAC-1000*

3. *List of functions for DPAC-3000*

	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)	
11	**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	
13	**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	
16	Manual Pulse Geneator Input		
	APS_get_pulser_counter	Get pluse input counter	
	APS_set_pulser_counter	Set pluse input counter	
17	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	
18	Non-Volatile RAM funciton		
	APS_set_nv_ram	Set RAM data	
	APS_get_nv_ram	Get RAM data	
	APS_clear_nv_ram	Clear RAM data	
	Table definition		
	Board parameters definition table		
	Field bus parameter definition		
	Interrupt factor table		
	Device information table		
	DPAC displayIndex table		
	DPAC Buttonstatus table		
	Function Return Code		

4. *List of functions for PCI-8392(H)*

	Function name	Descriptions	Page
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_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	
7	**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	
8	**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	
9	**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)	
10	**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	
12	**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	
13	**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_axis no	Get first axis of the slave module	
	APS_get_field_bus_device_info	Get device information on a specified field bus	

14	**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	
	Table definition		
	Board parameters definition table		
	Axis parameters definition table		
	Sampling parameters definition table		
	Sampling source definition table		
	The bit definition of motion IO status		
	Motion status definition table		
	Interrupt factor table		
	Field bus parameter definition		
	Gantry parameters definition table		
	Device information table		
	Field bus slave parameter table		
	SSCNET servo monitor source table		
	Function Return Code		

5. List of all functions for PCI-8253/56

Sec.	Function name	Descriptions	Page
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_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	
7	**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	
8	**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	
9	**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)	
10	**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	
11	**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	
12	**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	
14	**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)	
15	**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.	
16	**Manual Pulse Geneator functions**		
	APS_get_pulser_counter	Get pluse input counter	
20	**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	
	Table definition		
	Board parameters definition table		
	Axis parameters definition table		
	Sampling parameters definition table		
	Sampling source definition table		
	The bit definition of motion IO status		
	Motion status definition table		
	Interrupt factor table		
	Gantry parameters definition table		
	Trigger parameter table		
	Device information table		
	VAO parameter table		
	Function Return Code		

6. *List of all functions for PCI-8144*

Sec.	Function name	Descriptions	Page
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_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	
9	**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)	
11	**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	**Non-Volatile RAM funciton**		
	APS_set_nv_ram	Set RAM data	
	APS_get_nv_ram	Get RAM data	
	APS_clear_nv_ram	Clear RAM data	
	Table definition		
	Axis parameters definition table		
	The bit definition of motion IO status		
	Motion status definition table		
	Interrupt factor table		
	Device information table		
	Function Return Code		

7. *List of all functions for PCI-7856*

Sec.	Function name	Descriptions	Page
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_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	
9	**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	**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_axis no	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	
18	**Non-Volatile RAM funciton**		
	APS_set_nv_ram	Set RAM data	
	APS_get_nv_ram	Get RAM data	

	APS_clear_nv_ram	Clear RAM data	
	Table definition		
	Board parameters definition table		
	Field bus parameter definition		
	Interrupt factor table		
	Device information table		
	Function Return Code		

8. List of all functions for MNET-4XMO

Sec.	Function name	Descriptions	Page
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	
8	**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	
9	**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.	
10	**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	**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_axis no	Get first axis of the slave module	
	APS_get_field_bus_device_info	Get device information on a specified field bus	
21	**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	
22	**Single-latch functions**		
	APS_manual_latch2	Manual latch for a axis	
	APS_get_latch_data2	Get latch data for a axis	
	Table definition		
	Axis parameters definition table		
	The bit definition of motion IO status		
	Motion status definition table		
	Interrupt factor table		
	Field bus parameter definition		
	Device information table		

	Function Return Code	

9. *List of all functions for MNET-4XMO-C*

Sec.	Function name	Descriptions	Page
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	
8	**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	
9	**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.	
10	**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	
12	**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	
13	**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_axis no	Get first axis of the slave module	
	APS_get_field_bus_device_info	Get device information on a specified field bus	
19	**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_c mp	Get current table comparing value	
	APS_get_field_bus_trigger_linear_c mp	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_re main_count	Get remaining counter of linear comparator	
	APS_get_field_bus_table_cmp_rem ain_count	Get remaining counter of table comparator	

	APS_get_field_bus_encoder	Get encoder counter	
	APS_set_field_bus_encoder	Set encoder counter	
21	**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	
22	**Single-latch functions**		
	APS_manual_latch2	Manual latch for a axis	
	APS_get_latch_data2	Get latch data for a axis	
	Table definition		
	Axis parameters definition table		
	The bit definition of motion IO status		
	Motion status definition table		
	Interrupt factor table		
	Field bus parameter definition		
	Trigger parameter table		
	Device information table		
	Function Return Code		

10. List of all functions for MNET-1XMO

Sec.	Function name	Descriptions	Page
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_home_escape	Leave home switch	
	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	
9	**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.	
13	**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	
22	**Single-latch functions**		
	APS_manual_latch2	Manual latch for a axis	
	APS_get_latch_data2	Get latch data for a axis	
	Table definition		
	Axis parameters definition table		
	The bit definition of motion IO status		
	Motion status definition table		
	Interrupt factor table		
	Field bus parameter definition		
	Function Return Code		

11. List of all functions for HSL-4XMO

Sec.	Function name	Descriptions	Page
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	
8	**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	
12	**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	
13	**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_axis no	Get first axis of the slave module	
	APS_get_field_bus_device_info	Get device information on a specified field bus	
19	**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	
	Table definition		
	Axis parameters definition table		
	The bit definition of motion IO status		
	Motion status definition table		
	Field bus parameter definition		
	Trigger parameter table		
	Device information table		
	Function Return Code		

12. *List of all functions for HSL-DIO*

Sec.	Function name	Descriptions	Page
9	**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.	
13	**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	
	Table definition		
	Field bus parameter definition		
	Function Return Code		

13. List of all functions for PCI-8154/58/02/58A

Sec.	Function name	Descriptions	Page
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_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_home_escape	Leave home switch	
8	**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	
9	**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)	
11	**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	
15	**Compare trigger**		
	APS_set_trigger_param	Set trigger parameters	
	APS_set_trigger_encoder_counter	Set trigger encoder counter	
	APS_get_trigger_encoder_counter	Get trigger encoder counter	
	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 trigger parameters	
	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	
22	**Single-latch functions**		
	APS_manual_latch2	Manual latch for a axis	
	APS_get_latch_data2	Get latch data for a axis	
	Table definition		
	Axis parameters definition table		
	The bit definition of motion IO status		
	Motion status definition table		
	Interrupt factor table		
	Device information table		
	Function Return Code		
	Trigger parameter table		

3. System and Initialization

1	APS_initial	Device initialization

Support Products: PCI-8253/56, PCI-8392 (H), DPAC-1000, DPAC-3000, PCI-8144, PCI-7856, PCI-8154/58/02

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.

Syntax:

C/C++:

I32 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 0: auto mode (default)

	1: fixed mode
Bit 4 Bit 5	Option of load system & axes parameters method. (PCI-8253/6 and PCI-8392(H)) only (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
Bit 6	Option to select system mode. (PCI-7856 Only) (0) – Polling mode. (Not support motion interrupt) (1) – Interrupt mode. (Support motion interrupt)
Others	Reserved

Return Values:

I32 Error code: Please refer to error code table.

Example:

```
I32 ret; // return value
I32 BoardID_InBits;
I32 Mode = 0; //By system assigned
ret = APS_initial( &BoardID_InBits, Mode);

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

See also:

APS_close(), APS_get_axis_info();

2	APS_close	Devices close

Support Products: PCI-8253/56, PCI-8392 (H), DPAC-1000, DPAC-3000, PCI-8144, PCI-7856, PCI-8154/58/02

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 APS_close()

Visual Basic:

APS_close() As Long

Parameters:

No parameter.

Return Values:

I32 Error code: Please refer to error code table.

Example:

```
I32 ret; // return value
I32 BoardID_InBits;
I32 Mode = 0; //By system assigned
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-7856, PCI-8154/58/02

Descriptions :

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

Syntax:

C/C++:

I32 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-7856, PCI-8154/58/02

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 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:

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-7856, MNET-4XMO-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

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-8392 SSCNET 3 is parallel type axis. MNET-4XMO-© 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 16. 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.

	PCI-8392 (ID=0), 8-axis	PCI-8253 (ID=1), 3-axis
Auto Mode	Axis ID ranges 0~7	Axis ID ranges 8~10
Fixed Mode	Axis ID ranges 0~15	Axis ID ranges 16~31

If the board ID is not continuous,

	PCI-8392 (ID=0), 8-axis	PCI-8253 (ID=2), 3-axis
Auto Mode	Axis ID ranges 0~7	Axis ID ranges 8~10
Fixed Mode	Axis ID ranges 0~15	Axis ID ranges 32~47

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-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

Syntax:

C/C++

I32 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-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.

Return Values:

I32 Error code: Please refer to error code table.

Example:

See also:

APS_start_field_bus(), APS_initial()

APS_set_board_param	Set board parameter

Support Products: PCI-8253/56, PCI-8392 (H), DPAC-1000, DPAC-3000, PCI-7856

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 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 error code table.

Example:

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-7856

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 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 error code table.

Example:

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-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

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 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 error code table.

Example:

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-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

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 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 error code table.

Example:

See also:

APS_set_axis_param();

APS_get_system_timer	Get system timer counter

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

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 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 error code table.

Example:

See also:

APS_get_device_info	Get device information

Support Products: PCI-8253/56, PCI-8392 (H), DPAC-1000, DPAC-3000, PCI-8144, PCI-7856, PCI-8154/58/02

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 devices information table.

Refer to device iformation table.

Syntax:

C/C++

```
I32 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 devices information table.

I32 *Info: Reference to devices information table.

Return Values:

I32 Error code: Please refer to error code table.

Example:

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

See also:

APS_save_parameter_to_flash	Save system parameters & axes parameters to flash

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

Descriptions:

This function is used to save system parameters and axes parameters to flash.

Syntax:

C/C++:

```
I32 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 error code table.

Example:

```
I32 ret;
ret = APS_save_parameter_to_flash ( 0 );
if( ret == ERR_NoError )
        // Load 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)

Descriptions:

Load system parameters and axes parameters from flash.

Syntax:

C/C++:

I32 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 error code table.

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)

Descriptions:

Load default setting to system parameters & axes parameters.

Syntax:

C/C++:

I32 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 error code table.

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

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 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 error code table.

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

Descriptions:

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

Syntax:

C/C++:

I32 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 error code table.

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

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 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 error code table.

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_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 pro 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()

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 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. Refer to error code table.

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_save_param_to_file	Save parameters to file

Support Products: PCI-7856, MNET-4XMO-©, 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.
Note: User can also use Motion creator pro utility to create or modify a XML files.

Syntax:

C/C++:

```
I32 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. Refer to error code table.

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()

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 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 error code table.

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 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 error code table.

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 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 error code table.

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 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 error code table.

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 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 error code table.

Example:

```
I32 Alarm_No;
I32 Alarm_Detail;

…//Alarm occurred!
APS_get_sscnet_servo_alarm(Axis_ID, &Alarm_No, &Alarm_Detail ); //Get alarm �peration�i
…//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 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 error code table.

Example:

```
I32 Alarm_No;
I32 Alarm_Detail;

…//Alarm occurred!
APS_get_sscnet_servo_alarm(Axis_ID, &Alarm_No, &Alarm_Detail ); //Get alarm �peration�i
…//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 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 error code table.

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 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 error code table.

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 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 error code table.

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 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 error code table.

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 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 error code table.

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 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 error code table.

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 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 error code table.

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 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 error code table.

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-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

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 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-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

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 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-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

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 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 error code table.

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-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

Descriptions:

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

Syntax:

C/C++:

I32 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 error code table.

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-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

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 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 error code table.

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-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

Descriptions:

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

Syntax:

C/C++:

```
I32 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 error code table.

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-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

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 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 error code table.

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-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

Descriptions:

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

Syntax:

C/C++:

I32 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 error code table.

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)

Descriptions:

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

Syntax:

C/C++:

I32 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 error code table.

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-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

Descriptions:

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

Syntax:

C/C++:

```
I32 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 error code table.

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-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

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.

Syntax:

C/C++:

I32 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 error code table.

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-©, MNET-1XMO, PCI-8154/58/02/58A

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 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 error code table.

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-©, MNET-1XMO, PCI-8154/58/02/58A

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 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 error code table.

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-©, MNET-1XMO, PCI-8154/58/02/58A

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 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 error code table.

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-©, MNET-1XMO, PCI-8154/58/02/58A

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 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 error code table.

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-©, MNET-1XMO, PCI-8154/58/02/58A

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 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 error code table.

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-©, MNET-1XMO, PCI-8154/58/02/58A

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 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 error code table.

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-©, MNET-1XMO, PCI-8154/58/02/58A

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 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 error code table.

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()

6. Single axis motion

APS_relative_move	Begin a relative distance move

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8144, MNET-4XMO-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

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.

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.

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.

Syntax:

C/C++:

I32 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.

Return Values:

I32 Error code: Please refer to error code table.

Example:

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-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

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.
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.
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.

Syntax:

C/C++:

I32 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

Return Values:

I32 Error code: Please refer to error code table.

Example:

See also:

APS_velocity_move	Begin a velocity move

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8144, MNET-4XMO-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

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.

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

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.

Syntax:

C/C++:

I32 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

Return Values:

I32 Error code: Please refer to error code table.

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_home_move	Begin a home move

Support Products: PCI-8253/56, PCI-8392(H), PCI-8144, MNET-4XMO-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

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 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 error code table.

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-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)
```

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

APS_home_escape	Leave home switch

Support Products: MNET-4XMO-©, MNET-1XMO, PCI-8154/58/02

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 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 error code table.

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();

APS_stop_move	Stop move

Support Products: PCI-8253/56, PCI-8392(H) , PCI-8144, MNET-4XMO-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

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 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: Refer to error code table.

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-©, MNET-1XMO, HSL-4XMO, PCI-8154/58/02

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 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: Refer to error code table.

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_speed_override	Change speed on the fly

Support Products : MNET-1XMO, MNET-4XMO, MNET-4XMO-C

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 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 error code table.

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

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 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 error code table.

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

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 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 error code table.

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_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 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: Refer to error code table.

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 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: Refer to error code table.

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 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: Refer to error code table.

Example:

See also:

APS_home_move()

7. 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 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: Refer to error code table.

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 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: Refer to error code table.

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 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: Refer to error code table.

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)

Descriptions:

This function is used to start / stop a jog move. Before start a jog move, you must enable the axis to jog mode.

Syntax:

C/C++:

I32 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: Refer to error code table.

Example:

```
// 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
```

See also:

APS_set_jog_param(); APS_get_jog_param(); APS_jog_mode_switch();

8. Interpolation

APS_absolute_linear_move	Begin a absolute position linear interpolation

Support Products: PCI-8253/56, PCI-8392(H) , MNET-4XMO-© , HSL-4XMO, PCI-8154/58/02

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.

Syntax:

C/C++:

I32 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.

I32 *Position_Array: Absolute position array. (unit: pulse)

I32 Max_Linear_Speed: Maximum linear interpolation speed (unit: pulse/sec)

Return Values:

I32 Error code: Refer to error code table.

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-© , HSL-4XMO, PCI-8154/58/02

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.

Syntax:

C/C++:

I32 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.

I32 *Distance_Array: Relative distance array. (unit: pulse)

I32 Max_Linear_Speed: Maximum linear interpolation speed (unit: pulse/sec)

Return Values:

I32 Error code: Refer to error code table.

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-© , HSL-4XMO, PCI-8154/58/02

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.

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 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: Refer to error code table.

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-© , HSL-4XMO, PCI-8154/58/02

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.

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 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: Refer to error code table.

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 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: Refer to error code table.

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 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: Refer to error code table.

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. Users must use motion status checking function or interrupt event waiting function to wait it done. The motion status "Helical interpolation signal (HIP)" of each axis performing a helical interpolation will be turn on when command is started and will be turned off at command is finished. 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 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: Refer to error code table.

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. Users must use motion status checking function or interrupt event waiting function to wait it done. The motion status "Helical interpolation signal (HIP)" 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 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 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: Refer to error code table.

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().

9. Interrupt

APS_int_enable	Interrupt main switch

Support Products: PCI-8253/56, PCI-8392(H), DPAC-1000, DPAC-3000, PCI-8144, PCI-7856, PCI-8154/58/02

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 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: Refer to error code table.

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-7856, PCI-8154/58/02

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 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. Refer to error code table.

[Enable = 0] : Disable the interrupt factor

Return I32 error code. Refer to error code table.

Example:

<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
```

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-7856, PCI-8154/58/02

Descriptions :

This function is used to get the setting of interrupt factor.

Syntax:

C/C++:

I32 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. Refer to error code table.

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-7856, PCI-8154/58/02

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 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. Refer to error code table.

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-7856, PCI-8154/58/02

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 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 WaitAll is FALSE(0), the return value indicates the array index of the object that satisfied the wait. If more than one event became ⍰peratio during the call, this is the array index of the ⍰peratio object with the smallest index value of all the ⍰peratio objects.

Negative value: I32 error code: Refer to error code table.

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, I32 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-8154/58/02

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 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. Refer to error code table.

[Enable = 0] : Disable the interrupt

Return I32 error code. Refer to error code table.

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-7856, PCI-8154/58/02

Descriptions :

This function is used to reset singled event to non-singled state.

Syntax:

C/C++:

I32 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. Refer to error code table.

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-7856, PCI-8154/58/02

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 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. Refer to error code table.

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-7856, PCI-8154/58/02

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_factor(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-7856, PCI-8154/58/02

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-7856.

Support Products : PCI-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 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. Refer to error code table.

[Enable = 0] : Disable the interrupt factor

Return I32 error code. Refer to error code table.

Example:

```
<Set axis 1000 INSTP (BIT 0) interrupt ON MotionNet field bus on PCI-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-7856

Support Products : PCI-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 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. Refer to error code table.

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-7856.

Support Products : PCI-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 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. Refer to error code table.

[Enable = 0] : Disable the error interrupt factor

Return I32 error code. Refer to error code table.

Example:

<Set axis 1000 EPEL (BIT 5) error interrupt ON MotionNet field bus on PCI-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_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_get_field_bus_int_factor_error	Get error interrupt factor status for MotionNet series on PCI-7856

Support Products : PCI-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 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. Refer to error code table.

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-7856.

Support Products : PCI-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 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 �peratio 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: Refer to error code table.

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-7856.

Support Products : PCI-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 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 operatio 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: Refer to error code table.

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-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-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 �peration 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. Refer to error code table.

Negative value: I32 error code: Refer to error code table.

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-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-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. Refer to error code table.

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();

10. Sampling

APS_set_sampling_param	Set sampling parameter.

Support Products: PCI-8253/56, PCI-8392(H), MNET-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), 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 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 parameter table for definition.

I32 Param_Dat: The corresponding parameter value of sampling number. Refer to the sampling table.

Return Values:

I32 error code. Refer to error code table.

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

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 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 parameter table.

I32 *ParaDat: Return sampling parameter value. Refer to the sampling parameter table.

Return Values:

I32 error code. Refer to error code table.

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

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** functions cannot be used at the same time.

Syntax:

C/C++:

I32 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. Refer to error code table.

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

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** functions cannot be used at the same time.

Syntax:

C/C++:

I32 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. Refer to error code table.

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

Descriptions:

This function is used to get asynchronous wait sampling dat count.

Use **APS_wait_trigger_sampling_async** to start a sampling operation, you need to get sampling count to check the operation is finish success or failed.

Syntax:

C/C++:

I32 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. Refer to error code table.

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

Descriptions:

This function is used to forced stop **APS_wait_trigger_sampling** and **APS_wait_trigger_sampling_asnyc** function.

Syntax:

C/C++:

I32 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. Refer to error code table.

Example:

Refer to **APS_wait_trigger_sampling_asnyc** example

See also:

APS_wait_trigger_sampling; APS_wait_trigger_sampling_async

11. DIO & AIO

APS_write_d_output	Set digital output value

Support Products: PCI-8253/56, DPAC-1000, DPAC-3000, PCI-8144, PCI-8154/58/02

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-8154 has 4, PCI-8158 has 8, PCI-8102 has 2) output channels, user can assign group number to be constant 0. The PCI-8102 has 16 output channels, user can assign group number to be constant 1.

Syntax:

C/C++:

I32 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.

I32 DO_Data: The digit output data (Data type is bit type).

Return Values:

I32 error code. Refer to error code table.

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-8154/58/02

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-8154 has 4, PCI-8158 has 8, PCI-8102 has 2) output channels, user can assign group number to be constant 0. The PCI-8102 has 16 output channels, user can assign group number to be constant 1.

Syntax:

C/C++:

I32 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.

I32 *DO_Data: The digit output data (Data type is bit type).

Return Values:

I32 error code. Refer to error code table.

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-8154/58/02

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-8154 has 4, PCI-8158 has 8, PCI-8102 has 4) input channels, user can assign group number to be constant 0. The PCI-8102 has 16 input channels, user can assign group number to be constant 1.

Syntax:

C/C++:

I32 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.

I32 *DI_Data: The returned digit input data

Return Values:

I32 error code. Refer to error code table.

Example:

See also:

APS_write_d_output()

APS_read_a_input_value	Read back analog input value by volt

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 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.

Syntax:

C/C++:

I32 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. Refer to error code table.

Example:

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 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. Refer to error code table.

Example:

See also:

APS_read_a_input_value()

APS_write_a_output_value	Set analog output value by volt

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 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.

Syntax:

C/C++:

I32 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. Refer to error code table.

Example:

```
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 )
}
```

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 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. Refer to error code table.

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()

12. 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 �peratio 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 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. Refer to error code table.

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 ©
```

}

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 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. Refer to error code table.

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 �peratio 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 APS_set_point_table_ex(I32 Axis_ID, I32 Index, POINT_DATA_EX *Point);

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. Refer to error code table.

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 ©
}
```

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 APS_get_point_table_ex(I32 Axis_ID, I32 Index, POINT_DATA_EX *Point);

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. Refer to error code table.

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 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. Refer to error code table.

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 ©
}
```

```
// 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 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. Refer to error code table.

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 ©
}
```

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 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. Refer to error code table.

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 ©
}
```

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 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. Refer to error code table.

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 ©
}
```

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 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. Refer to error code table.

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 ©
}
```

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 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. Refer to error code table.

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 ©
}
```

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 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. Refer to error code table.

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 ©
```

}

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 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. Refer to error code table.

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 ©
}
```

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 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. Refer to error code table.

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 ©
}
```

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 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. Refer to error code table.

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 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. Refer to error code table.

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 ©
}
```

See also:

APS_point_table_continuous_move2	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. Refer to error code table.

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 ©
}

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. Refer to error code table.

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 ©
}

// 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 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. Refer to error code table.

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 ©
}
```

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. Refer to error code table.

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 ©
```

}

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 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. Refer to error code table.

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. Refer to error code table.

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 refre 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. Refer to error code table.

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-8253/6 PCI-8392(H)

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 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 error code table.

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 );
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 APS_get_feeder_group(I32 GroupId, I32 *Dimension, I32 *Axis_ID_Array);

I32 APS_free_feeder_group(I32 GroupId);

I32 APS_reset_feeder_buffer(I32 GroupId);

I32 APS_set_feeder_point_2D (I32 GroupId, POINT_DATA_2D* PtArray, I32 Size, I32 LastFlag);

I32 APS_start_feeder_move(I32 GroupId);

I32 APS_get_feeder_running_index(I32 GroupId, I32 *Index);

I32 APS_get_feeder_feed_index(I32 GroutId, I32 *Index);

APS_get_feeder_group	Return the configuration in one feeder group

Support Products: **PCI-8253/6 PCI-8392(H)**

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 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 error code table.

Example:

Refer to the example of *APS_set_feeder_group*()

See also:

I32 APS_set_feeder_group(I32 GroupId, I32 Dimension, I32 *Axis_ID_Array);

I32 APS_free_feeder_group(I32 GroupId);

I32 APS_reset_feeder_buffer(I32 GroupId);

I32 APS_set_feeder_point_2D(I32 GroupId, POINT_DATA_2D* PtArray, I32 Size, I32 LastFlag);

I32 APS_start_feeder_move(I32 GroupId);

I32 APS_get_feeder_running_index(I32 GroupId, I32 *Index);

I32 APS_get_feeder_feed_index(I32 GroutId, I32 *Index);

APS_free_feeder_group	Free a feeder group and it's resources

Support Products: PCI-8253/6 PCI-8392(H)

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 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 error code table.

Example:

Refer to the example of *APS_set_feeder_group*()

See also:

I32 APS_set_feeder_group(I32 GroupId, I32 Dimension, I32 *Axis_ID_Array);

I32 APS_get_feeder_group(I32 GroupId, I32 *Dimension, I32 *Axis_ID_Array);

I32 APS_reset_feeder_buffer(I32 GroupId);

I32 APS_set_feeder_point_2D(I32 GroupId, POINT_DATA_2D* PtArray, I32 Size, I32 LastFlag);

I32 APS_start_feeder_move(I32 GroupId);

I32 APS_get_feeder_running_index(I32 GroupId, I32 *Index);

I32 APS_get_feeder_feed_index(I32 GroutId, I32 *Index);

APS_reset_feeder_buffer	Reset the feeder's point buffer

Support Products: PCI-8253/6 PCI-8392(H)

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 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 error code table.

Example:

Refer to the example of *APS_set_feeder_group*()

See also:

I32 APS_set_feeder_group(I32 GroupId, I32 Dimension, I32 *Axis_ID_Array);

I32 APS_get_feeder_group(I32 GroupId, I32 *Dimension, I32 *Axis_ID_Array);

I32 APS_free_feeder_group(I32 GroupId);

I32 APS_set_feeder_point_2D(I32 GroupId, POINT_DATA_2D* PtArray, I32 Size, I32 LastFlag);

I32 APS_start_feeder_move(I32 GroupId);

I32 APS_get_feeder_running_index(I32 GroupId, I32 *Index);

I32 APS_get_feeder_feed_index(I32 GroutId, I32 *Index);

APS_set_feeder_point_2D	Add a point into feeder’s buffer

Support Products: **PCI-8253/6 PCI-8392(H)**

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 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 error code table.

Example:

Refer to the example of *APS_set_feeder_group*()

See also:

I32 APS_set_feeder_group(I32 GroupId, I32 Dimension, I32 *Axis_ID_Array);

I32 APS_get_feeder_group(I32 GroupId, I32 *Dimension, I32 *Axis_ID_Array);

I32 APS_free_feeder_group(I32 GroupId);

I32 APS_reset_feeder_buffer(I32 GroupId);

I32 APS_start_feeder_move(I32 GroupId);

I32 APS_get_feeder_running_index(I32 GroupId, I32 *Index);

I32 APS_get_feeder_feed_index(I32 GroutId, I32 *Index);

APS_start_feeder_move	Start point table move and feed points.

Support Products: PCI-8253/6 PCI-8392(H)

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 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 error code table.

Example:

Refer to the example of *APS_set_feeder_group*()

See also:

I32 APS_set_feeder_group(I32 GroupId, I32 Dimension, I32 *Axis_ID_Array);

I32 APS_get_feeder_group(I32 GroupId, I32 *Dimension, I32 *Axis_ID_Array);

I32 APS_free_feeder_group(I32 GroupId);

I32 APS_reset_feeder_buffer(I32 GroupId);

I32 APS_set_feeder_point_2D (I32 GroupId, PNT_DATA_2D * PtArray, I32 Size, I32 LastFlag);

I32 APS_get_feeder_running_index(I32 GroupId, I32 *Index);

I32 APS_get_feeder_feed_index(I32 GroutId, I32 *Index);

APS_get_feeder_running_index	Get which point is in operation.

Support Products: **PCI-8253/6 PCI-8392(H)**

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 �peration� ‘s ram; *APS_get_feeder_running_index*() return buffer index which order by feeder ‘s buffer in host ‘s ram.

Syntax:

C/C++:

I32 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 error code table.

Example:

Refer to the example of *APS_set_feeder_group*()

See also:

I32 APS_set_feeder_group(I32 GroupId, I32 Dimension, I32 *Axis_ID_Array);

I32 APS_get_feeder_group(I32 GroupId, I32 *Dimension, I32 *Axis_ID_Array);

I32 APS_free_feeder_group(I32 GroupId);

I32 APS_reset_feeder_buffer(I32 GroupId);

I32 APS_set_feeder_point_2D (I32 GroupId, PNT_DATA_2D * PtArray, I32 Size, I32 LastFlag);

I32 APS_start_feeder_move(I32 GroupId);

I32 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-8253/6 PCI-8392(H)

Descriptions:

This function will return which the latest buffer index in feeder is loaded into controller.

Syntax:

C/C++:

I32 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 error code table.

Example:

Refer to the example of *APS_set_feeder_group*()

See also:

I32 APS_set_feeder_group(I32 GroupId, I32 Dimension, I32 *Axis_ID_Array);

I32 APS_get_feeder_group(I32 GroupId, I32 *Dimension, I32 *Axis_ID_Array);

I32 APS_free_feeder_group(I32 GroupId);

I32 APS_reset_feeder_buffer(I32 GroupId);

I32 APS_set_feeder_point_2D(I32 GroupId, PNT_DATA_2D * PtArray, I32 Size, I32 LastFlag);

I32 APS_start_feeder_move(I32 GroupId);

I32 APS_get_feeder_running_index(I32 GroupId, I32 *Index);

APS_set_feeder_ex_pause	Motion paused(stopped) and feeder paused

Support Products: PCI-8253/6

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 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 error code table.

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 APS_set_feeder_ex_pause(I32 GroupId);

I32 APS_set_feeder_ex_rollback(I32 GroupId, I32 Max_Speed);

I32 APS_set_feeder_ex_resume(I32 GroupId);

APS_set_feeder_ex_rollback	Move back to the starting position of paused index

Support Products: PCI-8253/6

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 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 error code table.

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 APS_set_feeder_ex_pause(I32 GroupId);

```
I32 APS_set_feeder_ex_resume( I32 GroupId );
```

APS_set_feeder_ex_resume	Resume the point-table move.

Support Products: PCI-8253/6

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 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 error code table.

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 APS_set_feeder_ex_pause(I32 GroupId);

I32 APS_set_feeder_ex_rollback(I32 GroupId, I32 Max_Speed);

13. Field bus functions

APS_set_field_bus_param	Set field bus related parameters

Support Products: PCI-8392H, DPAC-3000, PCI-7856, MNET-4XMO-©, 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 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 �peration�

I32 BUS_Param: Field bus parameter data. Refer to table definition.

Return Values:

I32 error code. Refer to error code table.

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-7856, MNET-4XMO-©, 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 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 �peration�

I32 *BUS_Param: Return field bus parameter data. Refer to table definition.

Return Values:

I32 error code. Refer to error code table.

Example:

See also:

APS_set_field_bus_param

APS_start_field_bus	Start the network of specified field bus

Support Products: PCI-8392H, DPAC-3000, PCI-7856, MNET-4XMO-©, MNET-1XMO, HSL-4XMO, HSL-DIO

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.

Syntax:

C/C++:

I32 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-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.

Return Values:

I32 error code. Refer to error code table.

Example:

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(⍰perati, busNum, startingAxisId);

// Field bus operation…

APS_stop_field_bus(⍰perati, busNum); //Stop field bus.

See also:

APS_stop_field_bus();

APS_stop_field_bus	Stop the network of specified field bus

Support Products: PCI-8392H, DPAC-3000, PCI-7856, MNET-4XMO-©, MNET-1XMO, HSL-4XMO, HSL-DIO

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 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-7856, HSL field bus is Bus_No 0 and MNET field bus is Bus_No 1

Return Values:

I32 error code. Refer to error code table.

Example:

```
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( ⍰perati, busNum, startingAxisId );
// Field bus operation…
APS_stop_field_bus(⍰perati, busNum ); //Stop field bus.
```

See also:

APS_start_field_bus();

APS_field_bus_d_set_output	Set field bus digital output

Support Products: PCI-8392H, DPAC-3000, PCI-7856, MNET-4XMO-© , 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 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. Refer to error code table.

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( ?perati, busNum, startingAxisId );
DO_Value = 0xF;
ret = APS_field_bus_d_set_output(?perati, 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-7856, MNET-4XMO-© , 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 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. Refer to error code table.

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(?perati, busNum, startingAxisId);

ret = APS_field_bus_d_get_output(?perati, 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-7856, MNET-4XMO-©, 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 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. Refer to error code table.

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( �perati, busNum, startingAxisId );
ret = APS_field_bus_d_get_input(�perati, 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-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 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. Refer to error code table.

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( 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(boardId, 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-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 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. Refer to error code table.

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( boardId, busNum, startingAxisId );

ret = APS_field_bus_d_get_output_ex(boardId, 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-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 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. Refer to error code table.

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( boardId, 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-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 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. Refer to error code table.

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-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 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. Refer to error code table.

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-7856

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 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.

Return Values:

I32 error code. Refer to error code table.

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-7856

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 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. Refer to error code table.

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-7856

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 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.

Return Values:

I32 error code. Refer to error code table.

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-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 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. Refer to error code table.

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( ⍰perati, busNum, startingAxisId );
ret = APS_set_field_bus_param (⍰perati, busNum, PRF_CHKERRCNT_LAYER, bus_param );
ret = APS_get_slave_connect_quality( ⍰perati, 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

Support Products: PCI-8392(H) , DPAC-3000, PCI-7856, MNET-4XMO-©, MNET-1XMO, HSL-4XMO, HSL-DIO

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.

Syntax:

C/C++:

I32 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.

Return Values:

I32 error code. Refer to error code table.

Example 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( □perati, busNum, startingAxisId );
ret = APS_get_slave_online_status ( □perati, busNum, moduleNum, & on_line );
//if on_line is 5, it means that first and third ids are online.
```

Example 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 ( □perati, busNum, moduleNo, & on_line );
```

See also:

APS_get_slave_connect_quality;

APS_get_field_bus_last_scan_info	Get fieldbus info after system scanning.

Support Products: PCI-8392(H) , DPAC-3000, PCI-7856, MNET-4XMO-©, MNET-1XMO, HSL-4XMO, HSL-DIO

Descriptions:

This function is used to get the fieldbus info after system scanning. Please refer to the **fieldbus Info table**.

Syntax:

C/C++:

I32 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:

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.

Return Values:

I32 error code. Refer to error code table.

Example:

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
}
```

See also:

APS_get_field_bus_master_type	Get master type of the fieldbus

Support Products: PCI-8392(H) , DPAC-3000, PCI-7856, MNET-4XMO-©, MNET-1XMO, HSL-4XMO, HSL-DIO

Descriptions:

This function is used to get the master type of the fieldbus.

Syntax:

C/C++:

```
I32 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. Refer to error code table.

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-7856, MNET-4XMO-©, MNET-1XMO, HSL-4XMO, HSL-DIO

Descriptions:

This function is used to get the slave type on the fieldbus.

Syntax:

C/C++:

I32 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. Refer to error code table.

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-7856, MNET-4XMO-©, MNET-1XMO, HSL-4XMO, HSL-DIO

Descriptions:

This function is used to get the slave name on the fieldbus.

Syntax:

C/C++:

I32 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. Refer to error code table.

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_axisno	Get first axis of the slave module

Support Products: PCI-8392(H) , DPAC-3000 , PCI-7856, MNET-4XMO-©, 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 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. Refer to error code table.

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-7856, MNET-4XMO-©, 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 devices information table.

Syntax:

C/C++

I32 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 devices information table.

I32 *Info: Reference to devices information table.

Return Values:

I32 Error code: Please refer to error code table.

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:

14. Gantry functions

APS_set_gantry_param	Set gantry function related parameter

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 definition table**.

Syntax:

C/C++:

I32 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 definition table**.

I32 ParaDat: Parameter data. Please refer to **Gantry parameters definition table**.

Return Values:

I32 error code. Refer to error code table.

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 definition table**.

Syntax:

C/C++:

I32 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 definition table**.

I32 *ParaDat: Return a parameter data. Please refer to **Gantry parameters definition table**.

Return Values:

I32 error code. Refer to error code table.

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 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. Refer to error code table.

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 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. Refer to error code table.

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 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. Refer to error code table.

Example:

```
I32 ret; //return error code.
I32 boardId = 0;
I32 GroupNum = 0;
I32 GentryError;

ret = APS_get_gantry_error(�perati, 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 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 error code table.

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 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 error code table.

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 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 error code table.

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

15. Compare trigger

APS_set_trigger_param	Set compare trigger related parameter

Support Products: PCI-8253/56 PCI-8154/58

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 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. Refer to error code table.

Example:

Refer to example of "APS_set_trigger_linear", "APS_set_trigger_table"

See also:

APS_get_trigger_param();

APS_get_trigger_param	Get compare trigger related parameter

Support Products: PCI-8253/56 PCI-8154/58

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 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. Refer to error code table.

Example:

See also:

APS_set_trigger_param();

APS_set_trigger_linear	Set linear comparing function

Support Products: PCI-8253/56 PCI-8154/58

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 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.

I32 StartPoint: Start linear trigger point.

I32 RepeatTimes: Trigger repeat times.

I32 Interval: Trigger interval.

For PCI-8253/56, Interval: 24bit unsigned value.

Return Values:

I32 error code. Refer to error code table.

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

Descriptions:

This function is used to configure the specified comparing table.

Syntax:

C/C++:

I32 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.

I32 *DataArr: Comparing data array.

I32 ArraySize The size of comparing data array. Please refer to product's specification.

Return Values:

I32 error code. Refer to error code table.

Example:

```
#define POINTS    1000
I32 ret;
I32 data[POINTS];
I32 i;
for( i = 0; i < POINTS; i++ )
    data[i] = 10 + © * 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-8154/58

Descriptions:

This function is used to forced output a trigger at specified trigger output channel.

Syntax:

C/C++:

```
I32 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.

Return Values:

I32 error code. Refer to error code table.

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

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 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. Refer to error code table.

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

Descriptions:

This function is used to get current comparing value in the specified table comparator.

Syntax:

C/C++:

I32 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.

I32 *CmpVal: Return the current comparing value in the comparator.

Return Values:

I32 error code. Refer to error code table.

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-8154/58

Descriptions:

This function is used to get current comparing value in the specified linear comparator.

Syntax:

C/C++:

I32 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.

I32 *CmpVal: Return the current comparing value in the comparator.

Return Values:

I32 error code. Refer to error code table.

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-8154/58

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 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.

I32 *TrgCnt: Return trigger counter value.

Return Values:

I32 error code. Refer to error code table.

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-8154/58

Descriptions:

This function is used to reset the triggered counter to zero.

Syntax:

C/C++:

I32 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.

Return Values:

I32 error code. Refer to error code table.

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_set_trigger_encoder_counter	Set trigger encoder counter

Support Products: PCI-8154/58

Descriptions:

This function is used to set the encoder(counter) value directly.

Syntax:

C/C++:

I32 FNTYPE APS_set_trigger_encoder_counter(I32 Board_ID, I32 TrgCh, I32 TrgCnt);

Visual Basic:

APS_set_trigger_encoder_counter (ByVal Board_ID As Long, ByVal TrgCh As Long, ByVal 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: The specified channel number. (0 – 1)

I32 TrgCnt: The encoder (counter) value. (- 2147483647~ 2147483647)

Return Values:

I32 error code. Refer to error code table.

Example:

I32 Board_ID = 0;

I32 TrgCh = 0;

I32 TrgCnt = 1000;

I32 ret = 0;

ret = APS_set_trigger_encoder_counter(Board_ID, TrgCh, TrgCnt);

See also:

APS_get_trigger_encoder_counter()

APS_get_trigger_encoder_counter	Get trigger encoder counter

Support Products: PCI-8154/58

Descriptions:

This function is used to get the encoder(counter) value directly.

Syntax:

C/C++:

I32 FNTYPE APS_get_trigger_encoder_counter(I32 Board_ID, I32 TrgCh, I32 *TrgCnt);

Visual Basic:

APS_get_trigger_encoder_counter (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: The specified channel number. (0 – 1)

I32 *TrgCnt: The encoder (counter) value.

Return Values:

I32 error code. Refer to error code table.

Example:

```
I32 Board_ID = 0;
I32 TrgCh = 0;
I32 *TrgCnt ;
I32 ret = 0;
ret = APS_get_trigger_encoder_counter(Board_ID, TrgCh, &TrgCnt );
```

See also:

APS_set_trigger_encoder_counter()

APS_enable_trigger_fifo_cmp	Enable trigger fifo comparator

Support Products: PCI-8154/58

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. Refer to error code table.

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-8154/58

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. Refer to error code table.

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-8154/58

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

Bit8=0: equal or greater than the preset level,

Bit8=1: below the preset level

Other bits be reserved

Return Values:

I32 error code. Refer to error code table.

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-8154/58

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 – 2097151)

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. Refer to error code table.

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 );
```

See also:

APS_start_timer	Start / Stop timer

Support Products: PCI-8154/58

Descriptions:

Start / stop timer

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. (0~1)

I32 Start: start/stop timer

0: stop timer

1: start timer

Return Values:

I32 error code. Refer to error code table.

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:

16. Manual Pulse Geneator functions

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 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. Refer to error code table.

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 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. Refer to error code table.

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

17. 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 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 error code table.

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 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 error code table.

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 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 error code table.

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 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 error code table.

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 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 error code table.

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

18. Non-Volatile RAM

APS_set_nv_ram	Set NVRAM data

Support Products: DPAC-1000, DPAC-3000, PCI-8144, PCI-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 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-7856)

I32 DataWidth: 0: RW_WIDTH_8; 1: RW_WIDTH_16; 2: RW_WIDTH_32(PCI-7856 Only)

I32 Offset: The Offset from 0x0000 to 0x75FF(DPAC).; The Offset from 0x0000 to 0x7FFF(PCI-7856)

I32 Data: DataWidth: 0 The Data from -128 to 127.; (DPAC,PCI-7856)

DataWidth: 1 The Data from -32768 to 32767.; (DPAC,PCI-7856)

DataWidth: 2 The Data from -2147483648 to 2147483647.; (PCI-7856 Only)

Return Values:

I32 error code. Refer to error code table.

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-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 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-7856)

I32 DataWidth: 0: RW_WIDTH_8; 1: RW_WIDTH_16; 2: RW_WIDTH_32(PCI-7856 Only)

I32 Offset: The Offset from 0x0000 to 0x75FF(DPAC).; The Offset from 0x0000 to 0x7FFF(PCI-7856)

I32 *Data: DataWidth: 0 The Data from -128 to 127.; (DPAC,PCI-7856)

DataWidth: 1 The Data from -32768 to 32767.; (DPAC,PCI-7856)

DataWidth: 2 The Data from -2147483648 to 2147483647.; (PCI-7856 Only)

Return Values:

I32 error code. Refer to error code table.

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-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 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-7856)

Return Values:

I32 error code. Refer to error code table.

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()

19. Field bus compare trigger

APS_set_field_bus_trigger_param	Set compare trigger related parameter

Support Products: MNET-4XMO-C, HSL-4XMO

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 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:

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.

Return Values:

I32 error code. Refer to error code table.

Example:

Refer to example of “APS_set_field_bus_trigger_linear”, “APS_set_field_bus_trigger_table”

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

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 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:

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.

Return Values:

I32 error code. Refer to error code table.

Example:

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

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.

Syntax:

C/C++:

I32 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:

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.

Return Values:

I32 error code. Refer to error code table.

Example:

I32 BoardId = 0;

I32 Bus_No = 1;

I32 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

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

Descriptions:

This function is used to configure the specified comparing table.

Syntax:

C/C++:

I32 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:

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.

Return Values:

I32 error code. Refer to error code table.

Example:

```
#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 + © * 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
```

See also:

APS_set_field_bus_trigger_linear;

APS_set_field_bus_trigger_manual	Manual output trigger

Support Products: MNET-4XMO-C

Descriptions:

This function is used to forced output a trigger at specified trigger output channel.

Syntax:

C/C++:

I32 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:

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.

Return Values:

I32 error code. Refer to error code table.

Example:

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

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

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 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:

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

Return Values:

I32 error code. Refer to error code table.

Example:

```
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.
//…
```

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

Descriptions:

This function is used to get current comparing value in the specified table comparator.

Syntax:

C/C++:

I32 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:

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.

Return Values:

I32 error code. Refer to error code table.

Example:

```
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.
}
```

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

Descriptions:

This function is used to get current comparing value in the specified linear comparator.

Syntax:

C/C++:

I32 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:

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.

Return Values:

I32 error code. Refer to error code table.

Example:

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.
}
```

See also:

APS_get_field_bus_trigger_table_cmp;

APS_get_field_bus_trigger_count	Get triggered count.

Support Products: MNET-4XMO-C

Descriptions:

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.

Syntax:

C/C++:

I32 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:

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.

Return Values:

I32 error code. Refer to error code table.

Example:

```
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.
```

```
}
```

See also:

APS_reset_field_bus_trigger_count

APS_reset_field_bus_trigger_count	Reset triggered count.

Support Products: MNET-4XMO-C

Descriptions:

This function is used to reset the triggered counter to zero.

Syntax:

C/C++:

I32 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:

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.

Return Values:

I32 error code. Refer to error code table.

Example:

```
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 );
…
```

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

Descriptions:

This function is used to get remaining counter of linear comparator.

Syntax:

C/C++:

I32 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:

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.

Return Values:

I32 error code. Refer to error code table.

Example:

```
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.
}
```

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

Descriptions:

This function is used to get remaining counter of table comparator.

Syntax:

C/C++:

I32 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:

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.

Return Values:

I32 error code. Refer to error code table.

Example:

```
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.
}
```

See also:

APS_get_field_bus_linear_cmp_remain_count;

APS_get_field_bus_encoder	Get encoder count.

Support Products: MNET-4XMO-C

Descriptions:

This function is used to get encoder count

Syntax:

C/C++:

I32 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:

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. Refer to error code table.

Example:

```
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.
}
```

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 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. Refer to error code table.

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;

20. VAO/PWM functions (Laser function)

APS_set_vao_param	Set parameter to 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 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 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. Refer to error code table.

Example:

```
I32 ret;

//Set output type of voltage mode to VAO table 0
ret = APS_set_vao_param(Board_ID, 0x00, 0);
```

See also:

APS_get_vao_param();

APS_get_vao_param	Get parameter of 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 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 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. Refer to error code table.

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

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:

Output type (0~3)	Output Range(范围)
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

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. Refer to error code table.

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

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 APS_set_vao_param_ex( I32 Board_ID, I32 Table_No, VAO_DATA* VaoData );
```

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 �peration�ion
	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 �peration�ion
	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;

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

Return Values:

I32 error code. Refer to error code table.

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

Descriptions:

This function is used to get parameters via VAO structure.

Refer to APS_set_vao_param_ex() for details.

Syntax:

C/C++:

I32 APS_get_vao_param_ex(I32 Board_ID, I32 Table_No, VAO_DATA* VaoData);

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 �peration�ion
    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 �peration�ion
    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. Refer to error code table.

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()

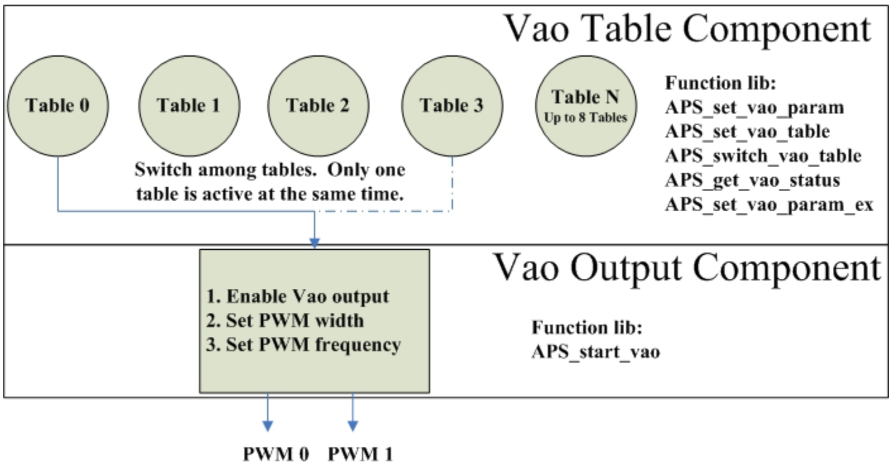
APS_switch_vao_table	Switch to specified VAO table

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 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. Refer to error code table.

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.

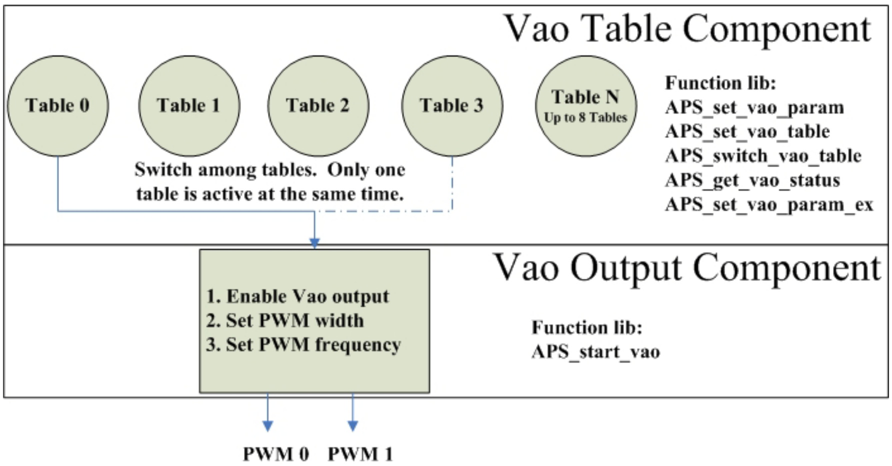
Support Products: PCI-8253/56

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 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:

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

Return Values:

I32 error code. Refer to error code table.

Example:

I32 ret;

ret = APS_start_vao(Board_ID, 0, 1); // Enable PWM channel 0 to output

See also:

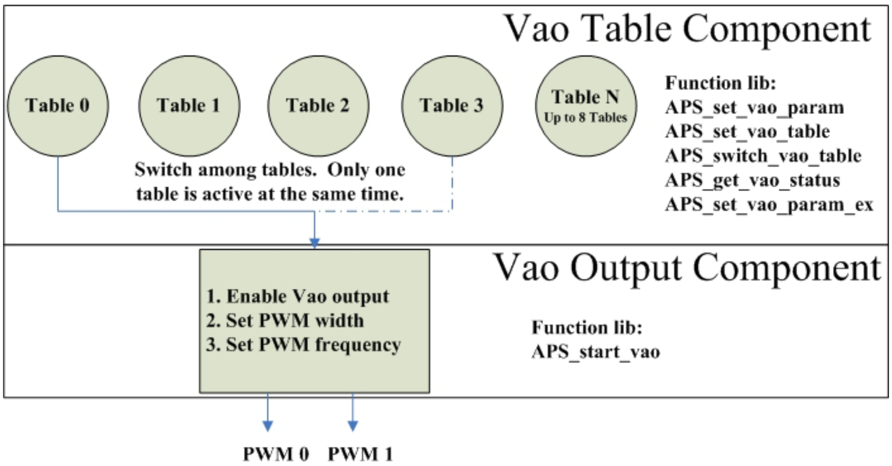
APS_set_vao_param();APS_get_vao_param(); APS_set_vao_table (); APS_switch_vao_table()

APS_get_vao_status	Get VAO status

Support Products: PCI-8253/56

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 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. Refer to error code table.

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

Descriptions:

This function is used to check table parameters of specidied VAO table.

Syntax:

C/C++:

I32 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 □peration□ion 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. Refer to error code table.

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

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 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. Refer to error code table.

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

Descriptions:

This function is used to set pulse width to specialized PWM channel.

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.

Syntax:

C/C++:

I32 APS_set_pwm_width(I32 Board_ID, I32 PWM _Ch, I32 Width);

Visual Basic:

I32 APS_set_pwm_width(ByVal Board_ID As Long , ByVal PWM_Ch As Long , ByVal Width 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 Width: Pulse width. Unit: ns. Range is from 40 to 335544340.

Return Values:

I32 error code. Refer to error code table.

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

Descriptions:

This function is used to set pulse frequency to specialized PWM channel.

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).

Syntax:

C/C++:

I32 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:

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.

Return Values:

I32 error code. Refer to error code table.

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

Descriptions:

This function is used to get pulse width from specialized PWM channel.

Syntax:

C/C++:

I32 APS_get_pwm_width(I32 Board_ID, I32 PWM _Ch, I32 *Width);

Visual Basic:

I32 APS_get_pwm_width(ByVal Board_ID As Long , ByVal PWM_Ch As Long , Width 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 Width: Pulse width. Unit: ns. Range is from 40 to 335544340.

Return Values:

I32 error code. Refer to error code table.

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

Descriptions:

This function is used to get pulse frequency from specialized PWM channel.

Syntax:

C/C++:

I32 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:

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.

Return Values:

I32 error code. Refer to error code table.

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()

21. Simultaneous move functions

APS_set_absolute_simultaneous_move	Setup a absolute simultaneous move

Support Products: MNET-4XMO-©

Descriptions:

The function is used to setup a 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: Refer to error code table.

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-©

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: Refer to error code table.

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-©

Descriptions:

The function is used to start a simultaneous operation for starting specified axes at the same time.

Syntax:

C/C++

I32 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 error code table.

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-©

Descriptions:

The function is used to stop a simultaneous operation for stopping specified axes at the same time.

Syntax:

C/C++

I32 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 error code table.

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

22. Single latch functions

APS_manual_latch2	Manual latch for a axis

Support Products: MNET-4XMO-©, MNET-1XMO, PCI-8154/58/02/58A

Descriptions:

The function is used to produce a manual latch signal.

Syntax:

C/C++:

I32 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: Refer to error code table.

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-©, MNET-1XMO, PCI-8154/58/02/58A

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 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

3: General-purpose counter

I32 *LatchData: Latch data

Return Values:

I32 Error code: Refer to error code table.

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()

23. Board Parameter Table

1. 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 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

2. *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	0

			count period is WDT0_UNIT. When timer counter reaches zero (time out), WDT_ACTION will happen.	
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

3. 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"

4. 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 DSP boots. DSP will reboot when power on or PCI bus reset.	0: default table 1: Flash ROM	0
110h	PRB_PWM0_MAP_DO	Enable the mapping between PWM0 & Do.	-1: Disable mapping Positive number: Enable mapping	-1

		Specify a Do channel to map PWM0. Select its mapping logic between PWM0 & Do.	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.	
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 & 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 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. 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”

5. *PCI-7856 board parameter table*

PCI-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

24. Axis Parameter Table

1. *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)	-100,00 0
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	0

			direction	
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	22,520,0 00
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	10,000,0 00
22h (34)	PRA_DEC	Deceleration rate	Unit: pulse/sec^2	10,000,0 00
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	10,000,0 00
44h (68)	PRA_JG_DEC	Deceleration rate	Unit: pulse/sec^2	10,000,0 00
45h (69)	PRA_JG_VM	Max. velocity	Unit: pulse/sec	1,000,00 0
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	Unit: pulse Value = 1 ~ 2147483647	200

		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.		
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.

$$\frac{1}{10} < \frac{PRA\_SSC\_GMR}{PRA\_SSC\_GDR} < 2000$$

2. PCI-8253/56 Axis parameter table.

PCI-8253/56 axis parameter table.				
NO.	Define	Description	Value	Note:
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	100,000,00 0
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	Unit: system cycle	0

		bit) The motion status NSTP bit will be turn on after specified delay cycle, when motion done condition is met.	time.	
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 ring counter value and enable ring counter mode	0
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 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).	Unit: Pulse	0
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

3. 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
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
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 �peration speed. Change this parameter will affect other motion parameters include	Unit: pulse/sec	409550

		PRA_ACC, PRA_VS		
10000h	PRA_CMD_CNT_EN	Enable soft command counter.	0: Disable 1: Enable	0
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

4. *MNET-4XMO-© Axis parameter table.*

MNET-4XMO-© axis parameter table.				
NO.	Define	Description	Value	Note:
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) 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 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.	0

			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. (*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) 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 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: 1.067 us (16-bit unsigned)	0
22C	PRA_VIBSUP_FT	Supress vibration – Forward Time	Unit: 1.067 us (16-bit unsigned)	0
22D	PRA_LATCH_DATA_SPD	Choose latch data of error position or current speed	0: Latch error position 1: Latch current speed	0

		for latch No.2		
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	57220458 9
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.

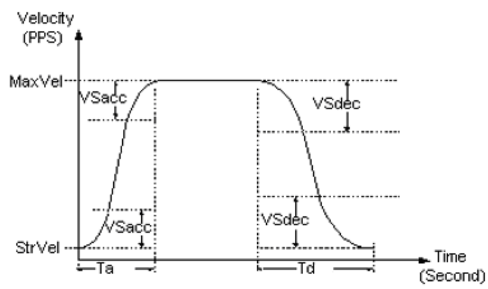
The formula is listed as below:

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.

5. MNET-1XMO Axis parameter table.

MNET-1XMO axis parameter table.				
NO.	Define	Description	Value	Note:
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 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	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(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 &	0

			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 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
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	Unit: pulse/sec^2	166666666

		speed. **(Read only)**		
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.

6. *HSL-4XMO Axis parameter table.*

HSL-4XMO axis parameter table.				
NO.	Define	Description	Value	Note:
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	0

			8) is reserved. If set, return error.	
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(1^{st} count) to 1111(16^{th} 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 &	0

			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. (*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: 1.067 us (16-bit unsigned)	0
22Ch	PRA_VIBSUP_FT	Supress vibration – Forward Time	Unit: 1.067 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(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	6553500
241h(577)	PRA_MAX_ACCDEC	Get max acceleration /deceleration which is	Unit: pulse/sec^2	245760000

		limited by fixed speed. **(Read only)**		
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.

7. *PCI-8154/58/02/58A Axis parameter table*

PCI-8154/58/02/58A axis parameter table				
NO.	Define	Description	Value	Note:
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) 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 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~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	Value = 1 ~ Motor	40,000

		Encoder resolution(112h) / Value (*1,)	Encoder resolution.	
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)	PRA_RESET_COUNTER	Reset counter’s value to zero when home moving be completed.	By bit setting: Bit0: Reset counter1	15

			(command position) 0: disable 1: enable Bit1: Reset counter2 (mechanical position) 0: disable 1: enable Bit2: Reset counter3 (deflection position) 0: disable 1: enable	
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	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	
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: 1.067 us (16-bit unsigned)	0
22C	PRA_VIBSUP_FT	Supress vibration – Forward Time	Unit: 1.067 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~22F	Reserved			
230h(560) (54/58 Only)	PRA_GPDO_SEL	Select DO/CMP Output mode	0: DO 1: CMP	0
231(561) (54/58 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) (02 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
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	24576000 0
242h(578)	PRA_MIN_ACCDEC	Get minimum acceleration/deceleration which is limited by fixed speed. **(Read only)**	Unit: pulse/sec^2	7500
250h	PRA_CONTI_MODE	Continuous Mode	0:Disable 1:Enable	0
251h	PRA_CONTI_BUFF	Continuous Buffer **(Read only)**	0:Empty 1:Full(8102) 1~3:Buffer(8154/58)	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.

25. Sampling parameters table

1. *Sampling parameter table for PCI-8392(H) and PCI-8253/56 and MNET-4XMO*

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)	0
11h	SAMP_PA_SRC_CH1	Sampling source of Channel 1	Refer to sampling source table.(*1)	0
12h	SAMP_PA_SRC_CH2	Sampling source of Channel 2	Refer to sampling source table.(*1)	0
13h	SAMP_PA_SRC_CH3	Sampling source of Channel 3	Refer to sampling source table.(*1)	0

.*1. 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 axis id, and the low two bytes is the type of sampling source.

2. *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	0: Not at max. velocity 1: At max. velocity	

		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_DEVIATIO N	Gantry deviation between master and slave encoder raw data	I32 value	
22h	Reserved			
23h	SAMP_ERROR_COUNT ER	Error counter data	I32 value	

(*1) Monitor data is according to monitor data source setting. Please refer to SSCNET servo monitor source table.

3. 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 (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	
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	

4. 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	

26. Motion IO status and motion status definitions

1. *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

2. *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

3. *MNET-4XMO-©/1XMO, HSL-4XMO, PCI-8154/58/02/58A motion IO status table*

MNET-4XMO-©/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

4. *PCI-8144 motion IO status table*

PCI-8144 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

5. *Motion IO status description table*

Motion IO status description table		
Bit	Define	Description
0	ALM	Servo alarm
1	PEL	Positive end limit
2	MEL	Negative 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 positive end limit
12	SMEL	Software negative end limit
13	TLC	Torque is limited by torque limit value. (When torque control is turned ON)
14	ABSL	Absolute position lost.
15		
16	PSD	Positive slow down signal input
17	MSD	Negative slow down signal input

6. 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

7. MNET-4XMO-©, HSL-4XMO, PCI-8154/58/02/58A Motion status definition table

7	6	5	4	3	2	1	0
SMV	HMV	NSTP	--	DEC	ACC	VM	CSTP
15	14	13	12	11	10	9	8
--	--	--	--	--	VS	CIP	LIP
23	22	21	20	19	18	17	16
--	MELS	PELS	--	ALMS	EMGS	--	ASTP
31	30	29	28	27	26	25	24
--	--	--	--	--	--	SMELS	SPELS

8. 1XMO Motion status definition table

7	6	5	4	3	2	1	0
SMV	HMV	NSTP	--	DEC	ACC	VM	CSTP
15	14	13	12	11	10	9	8
--	--	--	--	--	VS	--	--
23	22	21	20	19	18	17	16
--	MELS	PELS	--	ALMS	EMGS	--	ASTP
31	30	29	28	27	26	25	24
--	--	--	--	--	--	SMELS	SPELS

9. PCI-8144 Motion status definition table

7	6	5	4	3	2	1	0

--	**HMV**	--	**DIR**	**DEC**	**ACC**	**--**	**CSTP**
15	14	13	12	11	10	9	8
--	--	--	--	--	--	--	--
23	22	21	20	19	18	17	16
--	--	--	--	--	--	--	--
31	30	29	28	27	26	25	24
--	--	--	--	--	--	--	--

10. 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	HM	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	Pulsar mode
30	--	Reserved

27. Interrupt factor table

1. *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	Positive 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	

2. 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	Positive 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.

3. *Interrupt factor Item definition table for DPAC-1000*

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
	--	--	--	--	--	--	--	--

4. *Interrupt factor Item definition table for DPAC-3000*

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
	--	--	--	--	--	--	--	--

5. *PCI-7856 Interrupt Item Definition Table*

PCI-7856 Interrupt factor Item definition table

Interrupt factor Item definition table		
Item	Description	
0	CPLD Interrupt	

PCI-7856 CPLD Interrupt factor definition of Item 0

PCI-7856 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
	--	--	--	--	--	--	--	--

6. PCI-8144 Interrupt Item Definition Table

PCI-8144 Interrupt factor Item definition table

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 (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 0~5								
BitNo	7	6	5	4	3	2	1	0
	--	--	--	--	--	--	--	ICSTP

PCI-8144 Axes interrupt factors description table

PCI-8253/56 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-8392(H) System interrupt factors definition of item 16								
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-8392(H) 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-8392(H) 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-8392(H) System interrupt factors description			
NO.	Define	Interrupt condition description	Note
0	Din_R	Digital input Channl NO.n Rising edge interrupt	

7. MotionNet Interrupt Item Definition Table

MotionNet Axis Motion Interrupt factor definition(4XMO(-C))

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

4XMO© 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© 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 reset 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 reset 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© 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© 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)	

8. PCI-8154/58/02 Interrupt Item Definition Table

PCI-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-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-8154/58/02 Axes motion interrupt factors definition of Items

7	6	5	4	3	2	1	0
IDECE	IDECS	IACCE	IACCS	--	IRNM	IRNX	INSTP
15	14	13	12	11	10	9	8
IORGC	--	ICLRC	--	ICOMP4	--	ISMEL	ISPEL
23	22	21	20	19	18	17	16
--	--	--	--	--	--	--	ISD
31	30	29	28	27	26	25	24
--	--	--	--	--	--	--	--

PCI-8154/58/02 Axes motion interrupt factors description table

PCI-8154/58/02 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 reset 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)	

PCI-8154/58/02 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	Reserved	
3	General Comparator is on and axis is stopped	
4	Reserved	
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	Reserved	
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	

9. PCI-8158A Interrupt Item Definition Table

PCI-8158A 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	
16	Latch/Compare channel 0 interrupt	
…		
23	Latch/Compare channel 7 interrupt	

PCI-8158A Axes motion interrupt factors definition of Items

7	6	5	4	3	2	1	0
IDECE	IDECS	IACCE	IACCS	--	IRNM	IRNX	INSTP
15	14	13	12	11	10	9	8
IORGC	--	ICLRC	--	ICOMP4	--	ISMEL	ISPEL
23	22	21	20	19	18	17	16
--	--	--	--	--	--	--	ISD
31	30	29	28	27	26	25	24
--	--	--	--	--	--	--	--

PCI-8158A Axes motion interrupt factors description table

PCI-8158A 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 reset 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)	

PCI-8158A 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	Reserved	
3	General Comparator is on and axis is stopped	
4	Reserved	
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	Reserved	
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-8158A 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
--	--	LTCN	EZN	EBN	EAN	CMPEO	CMPL
23	22	21	20	19	18	17	16
--	--	--	--	--	--	--	--
31	30	29	28	27	26	25	24
--	--	--	--	--	--	--	--

PCI-8158A Latch/Compare interrupt factors description table

PCI-8158A Axes motion 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(Be equal to or greater than level)	
3	LTCFO	Latch fifo is in full overflow	
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(Be equal to or less than level)	
9	CMPEO	comparator fifo is in full overflow	
10	EAN	EA noise interrupt occurs	
11	EBN	EB noise interrupt occurs	
12	EZN	EZ noise interrupt occurs	
13	LTCN	LTC noise interrupt occurs	
14~31	Reserved	Reserved(Always set to 0)	

28. 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-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-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

29. Gantry parameters table

PCI-8253/56 Gantry parameters definition 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

PCI-8392(H) Gantry parameters definition 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

30. 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) 8:FCMP1 16: TMR	4
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	8

			4:FCMP0 8:FCMP1 (Default) 16: TMR	
0x14	TG_TRG0_PWD	TRG0 pulse width	$\text{Pulse Width} = (\text{N}+2) * 20\ \text{ns}$ 24 bit value. 0~ 16777215	0 (40 ns)
0x15	TG_TRG1_PWD	TRG1 pulse width	$\text{Pulse Width} = (\text{N}+2) * 20\ \text{ns}$ 24 bit value. 0~ 16777215	0 (40 ns)
0x16	TG_TRG2_PWD	TRG2 pulse width (*1)	$\text{Pulse Width} = (\text{N}+2) * 20\ \text{ns}$ 24 bit value. 0~ 16777215	0 (40 ns)
0x17	TG_TRG3_PWD	TRG3 pulse width (*1)	$\text{Pulse Width} = (\text{N}+2) * 20\ \text{ns}$ 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) Bit 2~31: Reserved (set 0)	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 (regardless of counting direction) 2: data = cmp counter (while counting up) 3: data = cmp counter (while counting down) 4: data > cmp counter	0

			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 Bit 2:CMP 2 Bit 3:CMP 3 Bit 4:CMP H Value: 0x00 ~ 0x1f	2
0x12	TG_TRG2_SRC	Trigger output 2 (TRG2) source	Bit 0:CMP 0 Bit 1:CMP 1 Bit 2:CMP 2	4

			Bit 3:CMP 3 Bit 4:CMP H Value: 0x00 ~ 0x1f	
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	$\text{Pulse Width} = (\text{N}+5) * 10\ \text{ns}$ Value: 0x05 ~ 0x7fffffff The value smaller than 0x05 is treated as 0x05.	5 (100 ns)
0x15	TG_TRG1_PWD	TRG1 pulse width	$\text{Pulse Width} = (\text{N}+5) * 10\ \text{ns}$ Value: 0x05 ~ 0x7fffffff The value smaller than 0x05 is treated as 0x05.	5 (100 ns)
0x16	TG_TRG2_PWD	TRG2 pulse width	$\text{Pulse Width} = (\text{N}+5) * 10\ \text{ns}$ Value: 0x05 ~ 0x7fffffff The value smaller than 0x05 is treated as 0x05.	5 (100 ns)
0x17	TG_TRG3_PWD	TRG3 pulse width	$\text{Pulse Width} = (\text{N}+5) * 10\ \text{ns}$ 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 (1) Bit 1~2: pulse (0) / toggle (1) / ByPass (2) / Disable (3) Bit 3~31: Reserved (set 0)	0
0x19	TG_TRG1_CFG	TRG 1 configuration	Bit 0: Pulse logic inverse. Not Inverse (0) / Inverse (1) Bit 1~2: pulse (0) / toggle	0

			(1) / ByPass (2) / Disable (3) Bit 3~31: Reserved (set 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, 0x03: 2XAB, 0x04: 4XAB.	0

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	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	
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 (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
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	0

			(while counting down) 4: data > cmp counter 5: data < cmp counter	
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 (1)	0
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 2: 1x AB-Phase 3: 2x AB-Phase	0

			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 1: Enable	0
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_ED	Set DI LATCH condition	0: DI falling edge to latch	0

	GE	(CH0)	1: DI Rising edge to latch	
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	0x4 (FIFO comparator)

			Bit 2: FIFO comparator 0: Disable 1: Enable Other bits reserved Note: FIFO comparator Only Support CH0	

*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	TRG4b	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.

31. 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

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-7856 Device information		
InfoNo	Information meaning	Format
0x00	Reserved	-
0x10	Driver version	Date format
0x20 0x30	CPLD version PCB version	16 Bits 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 0x40	Reserved DSP version	-- Date format	0x31	Reserved	--

PCI-8154/58/02 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-8158A 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	--

Format description:

1. Date format: 32 bit value

Value = YYMMDD; Y:year, M:month, D:day

Eg. Driver version = 80212. 2008/2/12 release.

2. PCB format: 4 bits value.

00 00b = PCB A1 version

00 00b

Sub version (1~4)

Main version (A~D)

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

3. 16 Bits Format: 16 bit value (1 ~ 255)

32. 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

33. DPAC displayIndex 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		
‘7.’	0x2C		
‘8.’	0X2D		
‘9.’	0X2E		
‘ ’	0X2F 0X20	0X20(ASCII’ ’)	

34. DPAC Push 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.

buttonstatus	B1 (Bit3)	B2 (Bit2)	B3 (Bit1)	B4 (Bit0)
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

35.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)		

36. 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_OU TPUT_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 point table is running.	0: No delay time. N: Delay time is N*DSP cycle. For PCI-8253, DSP cycle is 400 us. For PCI-8256, DSP cycle is	0

			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().

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.

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.
-1000	ERR_Win32Error	No such event number, or WIN32_API error, contact with ADLINK's FAE staff.