The invisible hand for more convenient world
Autonics sensors & controllers

Autonics
Sensors & Controllers

Motion Controller

PMC-4B-PCI

LIBRARY MANUAL

PMC-4B-PCI

Thank you very much for selecting Autonics products.
For your safety, please read the following before using.

Preface

Thank you for purchasing Autonics product.

Please familiarize yourself with the information contained in the **Safety Precautions** section before using this product.

This user manual contains information about the product and its proper use, and should be kept in a place where it will be easy to access.

User Manual Guide

- Please familiarize yourself with the information in this manual before using the product.
- This manual provides detailed information on the product's features. It does not offer any guarantee concerning matters beyond the scope of this manual.
- This manual may not be edited or reproduced in either part or whole without permission.
- A user manual is not provided as part of the product package. Visit (www.autonics.com) to download a copy.
- The manual's content may vary depending on changes to the product's software and other unforeseen developments within Autonics, and is subject to change without prior notice. Upgrade notice is provided through out homepage.
- We contrived to describe this manual more easily and correctly. However, if there are any corrections or questions, please notify us these on our homepage.

User Manual Symbols

Symbol	Description
 Note	Supplementary information for a particular feature.
 Warning	Failure to follow instructions can result in serious injury or death.
 Caution	Failure to follow instructions can lead to a minor injury or product damage.
 Ex.	An example of the concerned feature's use.
※1	Annotation mark.

Safety Precautions

- Following these safety precautions will ensure the safe and proper use of the product and help prevent accidents and minimize hazards.
- Safety precautions are categorized as Warnings and Cautions, as defined below:

Warning	**Warning**	Cases that may cause serious injury or fatal accident if instructions are not followed.
Caution	**Caution**	Cases that may cause minor injury or product damage if instructions are not followed.

Warning

- In case of using this unit with machinery (Ex: nuclear power control, medical equipment, ship, vehicle, train, airplane, combustion apparatus, safety device, crime/disaster prevention equipment, etc.) which may cause damages to human life or property, it is required to install fail-safe device.
  It may cause a fire, human injury or damage to property.
- Use this unit in the rated environment. Avoid using this unit where flammable or explosive gas or, high temperature and humidity, or vibration exists. It may cause fire, deterioration, malfunction, or damage to the product.
  It may cause a fire, human injury, or damage to property.
- Do not disassemble or modify this unit.
  It may cause a fire, human injury or damage to property.
- Do not cut off the power during operating.
  It may cause human injury, damage to property, or malfunction.
- Emergency stop should be available during operating.
  It may cause human injury, or damage to the product.
- Do not remove connector and jumper pin during operating.
  It may cause human injury, damage to property, or malfunction.
- Regard this product as industrial waste when discarding it.
  It may cause human injury, damage to property.
- Mount this unit on the PCI bus connector.
  It may cause damage to the product, electric shock, a fire, or human injury.
- When connecting this unit, refer to the connection diagram.
  It may cause electric shock, a fire or damage to the product.
- Install the limit switch.
  It may cause human injury or damage to property.
- Install the emergency stop switch.

Caution

- Do not connecte, inspect or repair this unit when it is power on.
  It may cause electric shock or malfunction.

- Do not disassemble the product. Please contact us if it is required.
  It may cause electric shock or a fire.
- Please observe the rated specification.
  It may shorten life cycle of the product or cause a fire.
- In cleaning the unit, do not use water or organic solvent. And use dry cloth.
  It may cause electric shock, a fire, or damage to the unit.
- Do not inflow dust or wire dregs into the unit.
  It may cause electric shock, a fire, or damage to the unit.
- After using this product and for storage, remove the I/O cable from the PC and pack this unit with wrapping paper for preventing static electricity. Keep this unit within the rated temperature and humidity.
- Turn OFF the power during installing or wiring.
  It may cause electric shock, damage to the product.
- Be sure not to short the each other cable during installing and wiring.
  It may cause electric shock or damage to the product.
- Do not wire to the unused terminal and be sure that not to short with the other terminals.
  It may cause electric shock or damage to the product.

Table of Contents

1 Initialization

1.1 open

MMC_INT16U **pmc4bpci_open**(int *id*, void (WINAPI **funcIntHandler*)(void));

(1) Descriptions

This function is for MMC board initialization. When interrupt is active, it is able to install callback function.
The below is initialization order of PMC-4B-PCI board and default values.

Selects 4-axis all → Reset → Register clear → Initialize

Range = 8,000,000, (magnification=1:1)

K= 101 (jerk speed=619kpps/sec²)

A= 1000 (acceleration=125kpps/sec)
D= 1000 (deceleration=125kpps/sec)
SV = 1000 (initial speed=1000pps)
V= 40000 (drive speed= 40000PPS)
P= 100000 (interpolation end point setting=100000)
LP = 0 (logical/position counter setting =0)

interpolation mode axis setting: 1-axis→X axis, 2-axis→Y axis, 3-axis→Z axis

(2) Factors

- id: Enters ID number. (range:0 to 15)
- funcIntHandler: When the set event at MMC card occurs, calls user function and transfers pointer of user function

(3) Return value

Success of driver initialization and library initialization returns PMC4BPCI_OK. When it is failed, it returns MMC_ERROR. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void WINAPI isr_sub0( void)
{
        // MMC Interrupt
}
void WINAPI isr_sub1( void)
{
        // MMC Interrupt
}
void WINAPI isr_sub2( void)
{
        // MMC Interrupt
}
void WINAPI isr_sub3( void)
{
```

```
		// MMC Interrupt
}
void WINAPI isr_sub4( void)
{
		// MMC Interrupt
}
void WINAPI isr_sub5( void)
{
		// MMC Interrupt
}
void WINAPI isr_sub6( void)
{
		// MMC Interrupt
}
void WINAPI isr_sub7( void)
{
		// MMC Interrupt
}
void WINAPI isr_sub8( void)
{
		// MMC Interrupt
}
void WINAPI isr_sub9( void)
{
		// MMC Interrupt
}
void WINAPI isr_sub10( void)
{
		// MMC Interrupt
}
void WINAPI isr_sub11( void)
{
		// MMC Interrupt
}
void WINAPI isr_sub12( void)
{
		// MMC Interrupt
}
void WINAPI isr_sub13( void)
{
		// MMC Interrupt
}
void WINAPI isr_sub14( void)
{
		// MMC Interrupt
}
void WINAPI isr_sub15( void)
{
		// MMC Interrupt
}

void main()
{
		MMC_INT16U stat;
		int i;

		// initialization function of MMC card
		// return value: When success of initialization, it returns MMC_OK.
		// factors:
```

```
		// id - id setting switch value of MMC card
		// funcIntHandler - When the set event at MMC card occurs, calls user function
		//                  transfers pointer of user function
		for(i=0;i<MAX_CARD;i++)
		{
			switch(i) {
			case  0: stat = pmc4bpci_open( 0, isr_sub0); break;
			case  1: stat = pmc4bpci_open( 1, isr_sub1); break;
			case  2: stat = pmc4bpci_open( 2, isr_sub2); break;
			case  3: stat = pmc4bpci_open( 3, isr_sub3); break;
			case  4: stat = pmc4bpci_open( 4, isr_sub4); break;
			case  5: stat = pmc4bpci_open( 5, isr_sub5); break;
			case  6: stat = pmc4bpci_open( 6, isr_sub6); break;
			case  7: stat = pmc4bpci_open( 7, isr_sub7); break;
			case  8: stat = pmc4bpci_open( 8, isr_sub8); break;
			case  9: stat = pmc4bpci_open( 9, isr_sub9); break;
			case 10: stat = pmc4bpci_open(10, isr_sub10); break;
			case 11: stat = pmc4bpci_open(11, isr_sub11); break;
			case 12: stat = pmc4bpci_open(12, isr_sub12); break;
			case 13: stat = pmc4bpci_open(13, isr_sub13); break;
			case 14: stat = pmc4bpci_open(14, isr_sub14); break;
			case 15: stat = pmc4bpci_open(15, isr_sub15); break;
			}
			if(stat==MMC_OK)
			{
printf("MESSAGE : Found and open 'PMC-4B-PCI(ID=%d)' driver\n", i);
			}
		}

		// closes open window driver
		pmc4bpci_close_all();

		printf("\nComplete OK!\n");
}
```

(5) Reference function

pmc4bpci_open, pmc4bpci_close_all

※For pmc4bpci_open function, interrupt call function does not support for Windows 7 64 bit.

1.2 reset

MMC_INT16U **pmc4bpci_reset**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

Initializes PMC-4B-PCI board.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When setting is success, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
rtn=pmc4bpci_set_reset(MMC_CARD_NO,PMC4BPCI_AXIS_X);
// ……

if(rtn!=MMC_OK)
{printf(“Fail!\n”);     }
else
{printf(“Complete OK!\n”);    }

pmc4bpci_close_all();

}
```

(5) Reference macro

PMC4BPCI_CMD_RST

2 Stop, End

2.1 close

MMC_INT16U **pmc4bpci_close**(MMC_INT16U *id*);

(1) Descriptions

It ends PMC-4B-PCI board.

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

It returns MMC_OK at driver end.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
        MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        } else
        {
                printf("Open card(%d)!\n", MMC_CARD_NO);
        }

        // ……
        rtn = pmc4bpci_close(MMC_CARD_NO);
        if(rtn!=MMC_OK)
                printf("ERROR : CardID=%d\n", MMC_CARD_NO);
        else
                printf("Close card(%d)!\n", MMC_CARD_NO);
        // ……
}
```

(5) Reference function

pmc4bpci_open

2.2 cloes all

MMC_INT16U pmc4bpci_close _all();

(1) Descriptions

It ends all PMC-4B-PCIboard.

(2) Return value

It must return MMC_OK at driver end.

(3) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
        MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        } else
        {
                printf("Open card(%d)!\n", MMC_CARD_NO);
        }

        // ……
        rtn = pmc4bpci_close_all();
        if(rtn!=MMC_OK)
printf("ERROR : Cannot find the open driver!\n", MMC_CARD_NO);
        else
                printf("All card closed!\n");
        // ……
}
```

(4) Reference function

pmc4bpci_open

2.3 dstop

MMC_INT16U **pmc4bpci_dstop**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It stops the axis drive by deceleration.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When setting is success, it returns PMC4BPCI_OK.
Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
MMC_INT16U    rtn;
            int OpenFlag;
            MMC_INT16U axis = PMC4BPCI_AXIS_X | PMC4BPCI_AXIS_Y |
                                        PMC4BPCI_AXIS_Z | PMC4BPCI_AXIS_U ;

            OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
            if(OpenFlag!=MMC_OK)
            {
                        printf("Initialize error!\n");
                        return;
            }

rtn = pmc4bpci_dstop(MMC_CARD_NO, axis);

if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_dstop

2.4 stop

MMC_INT16U **pmc4bpci_stop**(MMC_INT16U *id*,MMC_INT16U *axis*);

(1) Descriptions

It ends the axis drive immediately.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When setting is success, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
MMC_INT16U    rtn;
            int OpenFlag;
            MMC_INT16U axis = PMC4BPCI_AXIS_X | PMC4BPCI_AXIS_Y |
                                        PMC4BPCI_AXIS_Z | PMC4BPCI_AXIS_U ;

            OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
            if(OpenFlag!=MMC_OK)
            {
                        printf("Initialize error!\n");
                        return;
            }

rtn = pmc4bpci_stop(MMC_CARD_NO, axis);

if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_dstop

3 Register value read/write

3.1 outw

MMC_INT16U **pmc4bpci_outw**(MMC_INT16U *id*, MMC_INT16U *wOffset*, MMC_INT16U *wData*);

(1) Descriptions

It is I/O port input/output function. There is difference between input/output address for pmc4bpci and for NOVA compatibility. The above wOffset is relative address based on the base address of I/O address. wOffset address starts from 0 address.

pmc4bpci_outw writes wData(bData) at the address(wOffset) of pmc4bpci.
Address setting is word type function(pmc4bpci_outw()). Set the address as 2byte.
(Ex: pmc4bpci_outw(id, 0x00,0x8000))

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- wOffset: Relative address based on the base address
- wData: Data to write at I/O port

(3) Return value

When address input is out of the range, it returns PMC4BPCI_IOADDRESS_ERR. When setting is success, it returns PMC4BPCI_OK. Error check is available by checking global variable gError. Error variable gError value is same as Return value.

(4) Caution

When the function is to read I/O port data, it is able to return data or error. The decision whether error or data is depending on only globalvariable gError status for displaying error. For example, when returned value is same as PMC4BPCI_IOADDRESS_ERR and gError status is PMC4BPCI_OK, it decides as data.

(5) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
        MMC_INT16U    rtn, rv, i;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        // ……
```

```
        rtn = pmc4bpci_outw(MMC_CARD_NO, wr0, PMC4BPCI_CMD_RST); // reset
        if(MMC_ERROR()!=MMC_OK)
                printf("Out of write setting address range!\n");

        for(i=rr0;i<=rr7;i+=2) // word size, increases by 2byte
        {
                rv = pmc4bpci_inw(MMC_CARD_NO, i); // read status
                if(MMC_ERROR()!=MMC_OK)
                        printf("Out of read setting address range!\n");
                else
                        printf("id=%d, rr%d=0x%04x\n", MMC_CARD_NO, i/2, rv);
        }
        // ……

        pmc4bpci_close_all();
}
```

(6) Reference function

pmc4bpci_open, MMC_ERROR, pmc4bpci_close_all

3.2 inw

MMC_INT16U **pmc4bpci_inw**(MMC_INT16U *id*, MMC_INT16U *wOffset*);

(1) Descriptions

It is I/O port input/output function. There is difference between input/output address for pmc4bpci and for NOVA compatibility. The above wOffset is relative address based on the base address of I/O address. wOffset address starts from 0 address.

pmc4bpci_inw reads data at the address (wOffset) of pmc4bpci.
Address setting is word type function(pmc4bpci_inw()). Set the address as 2byte.
(Ex: pmc4bpci_inw(id, 0x00))

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- wOffset: Relative address based on the base address

(3) Return value

When address input is out of the range, it returns PMC4BPCI_IOADDRESS_ERR. When setting is success, it returns PMC4BPCI_OK. Error check is available by checking globalvariable gError. Error variable gError value is same as Return value.

(4) Caution

When the function is to read I/O port data, it is able to return data or error. The decision whether error or data is depending on only globalvariable gError status for displaying error. For example, when returned value is same as PMC4BPCI_IOADDRESS_ERR and gError status is PMC4BPCI_OK, it decides as data.

(5) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
        MMC_INT16U    rtn, rv, i;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        // ……
        rtn = pmc4bpci_outw(MMC_CARD_NO, wr0, PMC4BPCI_CMD_RST); // reset
        if(MMC_ERROR()!=MMC_OK)
                printf("Out of write setting address range!\n");

        for(i=rr0;i<=rr7;i+=2) // word size, increases by 2byte
        {
```



```
                    rv = pmc4bpci_inw(MMC_CARD_NO, i); // read status
                    if(MMC_ERROR()!=MMC_OK)
                        printf("Out of read setting address range!\n");
                    else
                        printf("id=%d, rr%d=0x%04x\n", MMC_CARD_NO, i/2, rv);
                }
                // ……

                pmc4bpci_close_all();
}
```

(6) Reference function

pmc4bpci_open, MMC_ERROR, pmc4bpci_close_all

4 Register value record

4.1 wwr1

MMC_INT16U **pmc4bpci_wwr1**(MMC_INT16U *id*, MMC_INT16U *axis*, MMC_INT16U *wdata*)

(1) Descriptions

At mode register(1) of the selected axis, it records deceleration/stop input signal during driving, valid/invalid setting bit status of IN0 to IN3 , and sets enable/disable of interrupt.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: Status value to the selected axis (for status value, refer to the user manual.)

Drive stop input signal
WR1_IN0_L: Logic level for input signal IN0 (When it is 0, it stops at low level. When it is 1, it stops by deceleration at high level.)
WR1_IN0_E: Valid/Invalid setting for input signal IN0 (0-Invalid, 1-Valid)
WR1_IN1_L: Logic level for input signal IN1 (When it is 0, it stops at low level. When it is 1, it stops by deceleration at high level.)
WR1_IN1_E: Valid/Invalid setting for input signal IN1 (0-Invalid, 1-Valid)
WR1_IN2_L: Logic level for input signal IN2 (When it is 0, it stops at low level. When it is 1, it stops by deceleration at high level.)
WR1_IN2_E: Valid/Invalid setting for input signal IN2 (0-Invalid, 1-Valid)
WR1_IN3_L: Logic level for input signal IN3 (When it is 0, it stops at low level. When it is 1, it stops by deceleration at high level.)
WR1_IN3_E: Valid/Invalid setting for input signal IN3 (0-Invalid, 1-Valid)

Interrupt occurnace conditions
WR1_INT_PULSE: Rising edge of drive pulse (occurance setting interrupt occurs at rising moment(rising edge) at pulse (positive logic of drive pulse)
WR1_INT_PGECM: Logical/Actual position pulse value ≥ COMP-register value
WR1_INT_PLCM: Logical/Actual position pulse value < COMP-register value
WR1_INT_PLCP: Logical/Actual position pulse value < COMP+register value
WR1_INT_PGEP: Logical/Actual position pulse value ≥ COMP+register value
WR1_INT_CEND: Ends pulse output at constant speed zone (accel/decel drive)
WR1_INT_CSTA: Starts pulse output at constant speed zone (accel/decel drive)
WR1_INT_DEND: Ends drive

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"
```



```
void WINAPI isr_sub( void)
{
        // MMC Interrupt
}

void main()
{
        #define MMC_CARD_NO     15
        MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, isr_sub);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }
        // ……
        rtn = pmc4bpci_wwr1(MMC_CARD_NO, PMC4BPCI_AXIS_X,
WR1_INT_PGECM |
WR1_INT_CEND);
        // ……

        if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

        pmc4bpci_close_all();
}
```

(5) Reference macro

WR1_IN0_L, WR1_IN0_E, WR1_IN1_L, WR1_IN1_E, WR1_IN2_L, WR1_IN2_E, WR1_IN3_L, WR1_IN3_E, WR1_INT_PULSE, WR1_INT_PGECM, WR1_INT_PLCM, WR1_INT_PLCP, WR1_INT_PGEP, WR1_INT_CEND, WR1_INT_CSTA, WR1_INT_DEND
IN0_SLOWSTOP, IN0_ENABLE, IN1_SLOWSTOP, IN1_ENABLE, IN2_SLOWSTOP, IN2_ENABLE, IN3_SLOWSTOP, IN3_ENABLE, INT_PULSE, INT_PGECM, INT_PLCM, INT_PLCP, INT_PGEP, INT_CEND, INT_CSTA, INT_DEND

4.2 wwr2

MMC_INT16U **pmc4bpci_wwr2**(MMC_INT16U *id*, MMC_INT16U *axis*, MMC_INT16U *wdata*);

(1) Descriptions

At mode register(2) of the selected axis, it sets mode of limit input signal, drive pulse, encoder input signal and valid/invalid of signal logic level for servo motor.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: Mode setting value(it is same as wr2 register contents.)

WR2_LMT_SLMTP: COMP+register as software limit setting
WR2_LMT_SLMTM: COMP-register as software limit setting
WR2_LMT_LMTMD: When hardware limit (nLMTP, nLMTM) is active,
stop method setting (0: immediate stop, 1: deceleration stop)
WR2_LMT_HLMTP: Logic level setting of +direction limit input signal(nLMTP)
(Active at 0: low, 1: high)
WR2_LMT_HLMTM: Logic level setting of -direction limit input signal(nLMTM)
(Active at 0: low, 1: high)
WR2_LMT_CMPSL: COMP+/- register comparison target (Compares 0: logical position, 1: actual position)
WR2_DRV_PLSMD: Output method of drive pulse (0: Individual 2-pulse method, 1: 1-pulse method)
WR2_DRV_PLS_L: Logic level setting of drive pulse (0: Positive logic, 1: Negative logic)
WR2_DRV_DIR_L: Logic level setting of direction output signal for drive pulse (0: low for + direction, high for - direction,1: high for + direction, low for - direction)
WR2_ENC_PINMD: Output method of encoder input signal(nECA/PPIN, nECB/PMIN) (0:2-phase 1:up/dn pulse input)
WR2_ENC_PIND0: Uses by combination of WR2_ENC_PIND1
WR2_ENC_PIND1: Division ratio setting of encoder 2-phase pulse input (D1D0 00=1/1, 01=1/2, 10=1/4, 11=invalid)
WR2_SRV_ALM_L: Logic level setting of nALARM input signal (Be active at 0: low, 1:high)
WR2_SRV_ALM_E: Valid/Invalid setting of input signal nALARM for servo motor alarm (0: invalid, 1: valid)
WR2_SRV_INP_L: Logic level setting of nINPOS input signal (Be active at 0: low, 1: high)
WR2_SRV_INP_E: Valid/Invalid setting for servo motor inposition input signal nINPOS (0: invalid, 1: valid)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"
```



```
void main()
{
        #define MMC_CARD_NO     15
        MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }


rtn = pmc4bpci_wwr2(MMC_CARD_NO, PMC4BPCI_AXIS_X, WR2_LMT_SLMTP |
WR2_LMT_SLMTM);

        if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

        pmc4bpci_close_all();
}
```

(5) Reference macro

WR2_LMT_SLMTP, WR2_LMT_SLMTM, WR2_LMT_LMTMD, WR2_LMT_HLMTP, WR2_LMT_HLMTM, WR2_LMT_CMPSL, WR2_DRV_PLSMD, WR2_DRV_PLS_L, WR2_DRV_DIR_L, WR2_ENC_PINMD, WR2_ENC_PIND0, WR2_ENC_PIND1, WR2_SRV_ALM_L, WR2_SRV_ALM_E, WR2_SRV_INP_L, WR2_SRV_INP_E

LMT_SLMTP, LMT_SLMTM, LMT_LMTMD, LMT_HLMTP, LMT_HLMTM, LMT_CMPSL, DRV_PLSMD, DRV_PLS_L, DRV_DIR_L, ENC_PINMD, ENC_PIND0, ENC_PIND1, SRV_ALM_L, SRV_ALM_E, SRV_INP_L, SRV_INP_E

4.3 wwr3

MMC_INT16U **pmc4bpci_wwr3**(MMC_INT16U *id*, MMC_INT16U *axis*, MMC_INT16U *wdata*);

(1) Descriptions

At mode register(3) of the selected axis, it sets manual deceleration, individual deceleration, S curve, external adjustment mode, and general output OUT4 to 7.

(2) Factors

- id: Enters ID number. (range:0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- *wdata*: Mode setting value

WR3_DRV_MANLD: Deceleration method of accel/decel constant speed drive (0: auto deceleration,1: manual deceleration)
WR3_DRV_DSNSE: Affected value during accel/decel drive deceleration
(0: affects acceleration value, 1: affectes deceleration value – uses only for manual method)
WR3_DRV_SACC: Linear accel/decel/S curve setting (0: linear accel/decel, 1: S curve)
WR3_DRV_EXOP0: Uses by combination of WR3_DRV_EXOP1
WR3_DRV_EXOP1: external input signal (D4D3 00=Invalid drive adjustment by external input signal, 01=Continuous drive mode, 10=Constant speed drive mode, 11=Invalid drive adjustment by external input signal)
WR3_OUT_OUTSL: Displays output signal nOUT4 to 7 as general or drive status
(Displays 0: general output, 1: drive status)
WR3_OUT_OUT4: Uses as output signal(0: low level, 1: high level)
WR3_OUT_OUT5: Uses as output signal(0: low level, 1: high level)
WR3_OUT_OUT6: Uses as output signal(0: low level, 1: high level)
WR3_OUT_OUT7: Uses as output signal(0: low level, 1: high level)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it retunrs MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"


void main()
{
        #define MMC_CARD_NO     15
        MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
```

```
        }

        rtn = pmc4bpci_wwr3(MMC_CARD_NO, PMC4BPCI_AXIS_X,
WR3_DRV_SACC);

        if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

        pmc4bpci_close_all();
}
```

(5) Reference macro

WR3_DRV_MANLD, WR3_DRV_DSNSE, WR3_DRV_SACC, WR3_DRV_EXOP0, WR3_DRV_EXOP1, WR3_OUT_OUTSL, WR3_OUT_OUT4, WR3_OUT_OUT5, WR3_OUT_OUT6, WR3_OUT_OUT7
DRV_MANLD, DRV_DSNSE, DRV_SACC, DRV_EXOP0, DRV_EXOP1, OUT_OUTSL, OUT_OUT4, OUT_OUT5, OUT_OUT6, OUT_OUT7

4.4 wwr4

MMC_INT16U **pmc4bpci_wwr4**(MMC_INT16U *id*, MMC_INT16U *axis*, MMC_INT16U *wdata*);

(1) Descriptions

Within data range of each axis X, Y, Z, U * (nOUT0 to 3)4bit = 16bit, it sets each axis data or 16bit output.

(2) Factors

- id: Enters ID number. (range:0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: Output value

WR4_OUT_XOUT0: Output setting of X axis output signal OUT0 (0: low, 1: high)
WR4_OUT_XOUT1: Output setting of X axis output signal OUT1 (0: low, 1: high)
WR4_OUT_XOUT2: Output setting of X axis output signal OUT2 (0: low, 1: high)
WR4_OUT_XOUT3: Output setting of X axis output signal OUT3 (0: low, 1: high)
WR4_OUT_YOUT0: Output setting of Y axis output signal OUT0 (0: low, 1: high)
WR4_OUT_YOUT1: Output setting of Y axis output signal OUT1 (0: low, 1: high)
WR4_OUT_YOUT2: Output setting of Y axis output signal OUT2 (0: low, 1: high)
WR4_OUT_YOUT3: Output setting of Y axis output signal OUT3 (0: low, 1: high)
WR4_OUT_ZOUT0: Output setting of Z axis output signal OUT0 (0: low, 1: high)
WR4_OUT_ZOUT1: Output setting of Z axis output signal OUT1 (0: low, 1: high)
WR4_OUT_ZOUT2: Output setting of Z axis output signal OUT2 (0: low, 1: high)
WR4_OUT_ZOUT3: Output setting of Z axis output signal OUT3 (0: low, 1: high)
WR4_OUT_UOUT0: Output setting of U axis output signal OUT0 (0: low, 1: high)
WR4_OUT_UOUT1: Output setting of U axis output signal OUT1 (0: low, 1: high)
WR4_OUT_UOUT2: Output setting of U axis output signal OUT2 (0: low, 1: high)
WR4_OUT_UOUT3: Output setting of U axis output signal OUT3 (0: low, 1: high)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"


void main()
{
        #define MMC_CARD_NO     15
        MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
```



```
                printf("Initialize error!\n");
                return;
        }
        rtn = pmc4bpci_wwr4(MMC_CARD_NO, PMC4BPCI_AXIS_X,
WR4_OUT_XOUT0 |
WR4_OUT_XOUT1);

        if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

        pmc4bpci_close_all();
}
```

(5) Reference macro

WR4_OUT_XOUT0, WR4_OUT_XOUT1, WR4_OUT_XOUT2, WR4_OUT_XOUT3,
WR4_OUT_YOUT0, WR4_OUT_YOUT1, WR4_OUT_YOUT2, WR4_OUT_YOUT3,
WR4_OUT_ZOUT0, WR4_OUT_ZOUT1, WR4_OUT_ZOUT2, WR4_OUT_ZOUT3,
WR4_OUT_UOUT0, WR4_OUT_UOUT1, WR4_OUT_UOUT2, WR4_OUT_UOUT3

4.5 wwr5

MMC_INT16U **pmc4bpci_wwr5**(MMC_INT16U *id*, MMC_INT16U *axis*, MMC_INT16U *wdata*);

(1) Descriptions

It sets interpolation mode.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: output value

WR5_INP_LSPD0: 0x0100 constant linear velocity mode setting of interpolation drive
WR5_INP_LSPD1: 0x0200 D1D0(00: invalid constant linear velocity, 01: 2-axis constant linear velocity, 10: disable to set, 11: 3-axis constant linear velocity)
WR5_INP_EXPLS: 0x0800 external send(0: not send, 1: step sending interpolation drive as external signal-EXPLSN-)
WR5_INP_CMPLS: 0x1000(0: not send, 1: step sending interpolation drive as command)
WR5_INP_CIINT: 0x4000
(interrupt occurance enable/disable setting (0: disable, 1: enable) in consecutive interpolation)
WR5_INP_BPINT: 0x8000 interrupt occurance enable/disable setting in bit pattern interpolation
(0: disable, 1: enable)
WR5_INP_AXIS1_X: Defines the axis to the 1-axis in interpolation (X axis)
WR5_INP_AXIS1_Y: Y axis
WR5_INP_AXIS1_Z: Z axis
WR5_INP_AXIS1_U: U axis
WR5_INP_AXIS2_X: Defines the axis to the 2-axis in interpolation (X axis)
WR5_INP_AXIS2_Y: Y axis
WR5_INP_AXIS2_Z: Z axis
WR5_INP_AXIS2_U: U axis
WR5_INP_AXIS3_X: Defines the axis to the 3-axis in interpolation (X axis)
WR5_INP_AXIS3_Y: Y axis
WR5_INP_AXIS3_Z: Z axis
WR5_INP_AXIS3_U: U axis
WR5_INP_LCMODE_INVALIDATE: Invalid mode for interpolation drive
WR5_INP_LCMODE_2AXIS: 2-axis constant linear velocity
WR5_INP_LCMODE_3AXIS: 3-axis constant linear velocity

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
```



```
{
        #define MMC_CARD_NO     15
        MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        // Selects interpolation axis (1-axis=X, 2-axis =Y, 3-axis =U)
        rtn = pmc4bpci_wwr5(MMC_CARD_NO, 0, WR5_INP_AXIS1_X |
WR5_INP_AXIS2_Y
| WR5_INP_AXIS3_U);

        if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

        pmc4bpci_close_all();
}
```

(5) Reference macro

WR5_INP_EXPLS, WR5_INP_CMPLS, WR5_INP_CIINT, WR5_INP_BPINT,
R5_INP_AXIS1_X, WR5_INP_AXIS1_Y, WR5_INP_AXIS1_Z, WR5_INP_AXIS1_U,
WR5_INP_AXIS2_X, WR5_INP_AXIS2_Y, WR5_INP_AXIS2_Z, WR5_INP_AXIS2_U,
WR5_INP_AXIS3_X, WR5_INP_AXIS3_Y, WR5_INP_AXIS3_Z, WR5_INP_AXIS3_U,
WR5_INP_LCMODE_INVALIDATE, WR5_INP_LCMODE_2AXIS,
WR5_INP_LCMODE_3AXIS, WR5_INP_AX10, WR5_INP_AX11, WR5_INP_AX20,
WR5_INP_AX21, WR5_INP_AX30, WR5_INP_AX31, WR5_INP_LSPD0,
WR5_INP_LSPD1, WR5_INP_EXPLS, WR5_INP_CMPLS, WR5_INP_CIINT,
WR5_INP_BPINT
INP_EXPLS, INP_CMPLS, INP_CIINT, INP_BPINT, INP_AXIS1_X, INP_AXIS1_Y,
INP_AXIS1_Z, INP_AXIS1_U, INP_AXIS2_X, INP_AXIS2_Y, INP_AXIS2_Z, INP_AXIS2_U,
INP_AXIS3_X, INP_AXIS3_Y, INP_AXIS3_Z, INP_AXIS3_U, INP_LCMODE_INVALIDATE,
INP_LCMODE_2AXIS, INP_LCMODE_3AXIS

4.6 write_data

MMC_INT16U **pmc4bpci_write_data**(MMC_INT16U *id*, MMC_INT16U *axis*, MMC_INT32U *wdata*);

(1) Descriptions

It records data at no. 6.7 write register(ww6, ww7).

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: Output data value

(3) Return value

This function is always successful and it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
 #include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
        MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        // ……
        // example of range setting
        rtn = pmc4bpci_write_data(MMC_CARD_NO, PMC4BPCI_AXIS_X, 8000000);
        rtn = pmc4bpci_outw(MMC_CARD_NO, wr0, PMC4BPCI_AXIS_X |
PMC4BPCI_CMD_R);

        // ……

        if(rtn!=MMC_OK)
{printf(“Fail!\n”);     }
else
{printf(“Complete OK!\n”);    }

        pmc4bpci_close_all();
}
```


(5) Reference macro

WR6_OUT_WD0, WR6_OUT_WD1, WR6_OUT_WD2, WR6_OUT_WD3, WR6_OUT_WD4, WR6_OUT_WD5, WR6_OUT_WD6, WR6_OUT_WD7, WR6_OUT_WD8, WR6_OUT_WD9, WR6_OUT_WD10, WR6_OUT_WD11, WR6_OUT_WD12, WR6_OUT_WD13, WR6_OUT_WD14, WR6_OUT_WD15 , WR7_OUT_WD0, WR7_OUT_WD1, WR7_OUT_WD2, WR7_OUT_WD3, WR7_OUT_WD4, WR7_OUT_WD5, WR7_OUT_WD6, WR7_OUT_WD7, WR7_OUT_WD8, WR7_OUT_WD9, WR7_OUT_WD10, WR7_OUT_WD11, WR7_OUT_WD12, WR7_OUT_WD13, WR7_OUT_WD14, WR7_OUT_WD15

5 Register value read

5.1 rr0 to rr5

MMC_INT16U **pmc4bpci_rr0(**MMC_INT16U *id*, MMC_INT16U *axis*);

MMC_INT16U **pmc4bpci_rr1(**MMC_INT16U *id*, MMC_INT16U *axis*);

MMC_INT16U **pmc4bpci_rr2(**MMC_INT16U *id*, MMC_INT16U *axis*);

MMC_INT16U **pmc4bpci_rr3(**MMC_INT16U *id*, MMC_INT16U *axis*);

MMC_INT16U **pmc4bpci_rr4(**MMC_INT16U *id*, MMC_INT16U *axis*);

MMC_INT16U **pmc4bpci_rr5(**MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It reads data at read register rr0 to 5.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.

(3) Return value

When data reading is success, it returns 16bit data. For each returned code information, refer to the “User Manual[7. Read/Write register]”. When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void WINAPI isr_sub( void)
{
        // MMC Interrupt
}

void main()
{
        #define MMC_CARD_NO     15
        MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, isr_sub);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        rtn = pmc4bpci_rr1(MMC_CARD_NO, PMC4BPCI_AXIS_X);
```



```
                        // ……
                        if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }
                        pmc4bpci_close_all();
}
```

6 Parameter setting

6.1 wait

MMC_INT16U **pmc4bpci_wait**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It waits while drive pulse of selected axis outputs.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the function ends after the set time-out time, it returns PMC4BPCI_TIMEOUT_ERR. When the setting is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
// linear   accel/decel fixed pulse driver
pmc4bpci_pls_move(MMC_CARD_NO, PMC4BPCI_AXIS_X, 20000, 2000, 100);

// Waiting driver end
pmc4bpci_wait(MMC_CARD_NO, PMC4BPCI_AXIS_X);
// ……

printf("Complete OK!\n");

pmc4bpci_close_all();

}
```


(5) Reference function

pmc4bpci_get_timeout, pmc4bpci_set_timeout, pmc4bpci_wait_timeout

6.2 wait time-out

MMC_INT16U **pmc4bpci_wait_timeout**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT32U *msec*);

(1) Descriptions

It waits while drive pulse of selected axis outputs. By the set time-out time(msec), when it waits over the set time, **pmc4bpci_wait_timeout**() function ends automatically. When time-out time(msec) is 0, it waits indefinitely. Inner time-out time is changed by factors msec.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- msec: Time-out time setting by msec(1/1000sec) unit.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the function ends after the set time-out time, it returns PMC4BPCI_TIMEOUT_ERR. When the setting is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
// linear   accel/decel fixed pulse driver
pmc4bpci_pls_move(MMC_CARD_NO, PMC4BPCI_AXIS_X, 20000, 2000, 100);

// Waits for time-out time (1000msec)
pmc4bpci_wait_timeout(MMC_CARD_NO, PMC4BPCI_AXIS_X, 1000);
// ……

printf("Complete OK!\n");

pmc4bpci_close_all();

}
```


(5) Reference function

pmc4bpci_set_timeout, pmc4bpci_get_timeout, pmc4bpci_wait

6.3 nextwait

MMC_INT16U **pmc4bpci_nextwait**(MMC_INT16U *id*, MMC_INT16U *msec*);

(1) Descriptions

It waits while parameter data/interpolation command for next node is writable status in consecutive interpolation drive.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- msec: Time-out time setting by msec(1/1000sec) unit.

(3) Return value

When the function ends after the set time-out time, it returns PMC4BPCI_TIMEOUT_ERR. When the setting is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

{
        // In interpolation, the specified axis to 1-axis, In interpolation, the specified axis to
       2-axis
        MMC_INT16U work_plane_axis[2] = {WR5_INP_AXIS1_X,
WR5_INP_AXIS2_Y};
        MMC_VERTEX p = {20000, -20000, };
        MMC_INT16U axis = PMC4BPCI_AXIS1 | PMC4BPCI_AXIS2 |
PMC4BPCI_AXIS3 |
                        PMC4BPCI_AXIS4 ;

// ……
// 2-axis   linear interpolation drive
pmc4bpci_pos_imove2(MMC_CARD_NO, work_plane_axis, &p);

// Waits until enable next command writing
pmc4bpci_nextwait(MMC_CARD_NO, 0);

p.x = 20000;
p.y = 0;
```



```
// 2-axis    linear interpolation drive
pmc4bpci_pos_imove2(MMC_CARD_NO, work_plane_axis, &p);
// ……

printf("\nComplete OK!\n");

}

 // closes open window driver
 pmc4bpci_close_all();
}
```

6.4 get_timeout

MMC_INT32U pmc4bpci_get_timeout();

(1) Descriptions

It returns the set time-out time.

(2) Return value

It returns the set inner time-out time (by 0.001sec. unit). This command is always successful and PMC4BPCI_OK is recorded at global error check variable gError.

(3) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
MMC_INT32U t = pmc4bpci_get_timeout();
// ……

 printf("\nComplete OK!\n");

 // closes open window driver
 pmc4bpci_close_all();

}
```

(4) Reference function

pmc4bpci_set_timeout, pmc4bpci_wait, pmc4bpci_wait_timeout

6.5 set_timeout

MMC_INT16U **pmc4bpci_set_timeout**(MMC_INT32U *msec*);

(1) Descriptions

To escape infinite loop which waits drive end, it sets inner time-out time. Time is msec(0.001 sec.) unit.
This method does not count system timer. Therefore, time-out time is inaccurate.

(2) Factors

- msec: Time-out time setting by msec(0.001sec.) unit.

(3) Return value

Inner time-out time setting must return PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
pmc4bpci_set_timeout(1000); // delay 1sec
// ……

printf("\nComplete OK!\n");

 // closes open window driver
 pmc4bpci_close_all();

}
```

(5) Reference function

pmc4bpci_get_timeout, pmc4bpci_wait, pmc4bpci_wait_timeout

6.6 bpwait

MMC_INT16U **pmc4bpci_bpwait**(MMC_INT16U *id*);

(1) Descriptions

It indicates stack counter(SC) value at bit pattern interpolation drive. Stack counter waits for 11(SC=3) days. If it waits over the set time by the set inner time-out time, pmc4bpci_bpwait() function ends automatically. When inner time-out time is 0, it waits indefinitely.

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

It checks that there is space of stack area for bit pattern interpolation. When bit pattern data of the next check result 16bit is writable, it returns PMC4BPCI_OK. When the set time is over by the set inner time-out time and function ends, it returns PMC4BPCI_TIMEOUT_ERR. Same value is recorded at gError as global error check variable.

(4) Example

```
 #include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
rtn = pmc4bpci_bpwait(MMC_CARD_NO);
// ……

if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_set_timeout, pmc4bpci_get_timeout

6.7 set_range

MMC_INT16U **pmc4bpci_set_range**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT32 *wdata*)

(1) Descriptions

It sets drive range. Magnification is decided automatically by the set range.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: Drive range setting (R)
  Setting range: 8,000,000/R (Setting range 8,000,000(1magnification) to 16,000(500magnification))

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"


void main()
{
        #define MMC_CARD_NO       15
        MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        // ……
        // example of range setting
        rtn = pmc4bpci_set_range(MMC_CARD_NO,
 PMC4BPCI_AXIS_X|PMC4BPCI_AXIS_Y|PMC4BPCI_AXIS_Z|PMC4BPCI_AXIS_U,
        8000000);

        if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

        pmc4bpci_close_all();
}
```

(5) Reference macro

PMC4BPCI_CMD_R, PMC4BPCI_RANGE

6.8 set_axis

MMC_INT16U **pmc4bpci_set_axis**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

Selects the axis.

(2) Factors

- id: Enters ID number. (range:0 to 15)
- axis: Macro selects the axis.

Macro and value for PMC-4B-PCI
X axis: PMC4BPCI_AXIS1, PMC4BPCI_AXIS_X, 0x0100
Y axis: PMC4BPCI_AXIS2, PMC4BPCI_AXIS_Y, 0x0200
Z axis: PMC4BPCI_AXIS3, PMC4BPCI_AXIS_Z, 0x0400
U axis: PMC4BPCI_AXIS4, PMC4BPCI_AXIS_U, 0x0800
The below macro definition uses same as NOVAcompatibility function.
X axis: AXIS_X, 0x01
Y axis: AXIS_Y, 0x02
Z axis: AXIS_Z, 0x04
U axis: AXIS_U, 0x08

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
rtn=pmc4bpci_set_axis(MMC_CARD_NO,
 PMC4BPCI_AXIS_X+PMC4BPCI_AXIS_Y);// Select X,Y axis
// ……

if(rtn!=MMC_OK)
{printf("Fail!\n");     }
```

```
else
{printf("Complete OK!\n");    }

pmc4bpci_close_all();

}
```

(5) Reference macro

PMC4BPCI_CMD_NOP

6.9 smove_stop

MMC_INT16U **pmc4bpci_smove_stop**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

After S speed curve drive of the axis ends, it sets linear accel/decel drive.

(2) Factors

- id: Enters ID number. (range:0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range,it returns MMC_INVALID_AXIS. When the setting is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
MMC_INT16U    rtn;
            int OpenFlag;
            MMC_INT16U axis = PMC4BPCI_AXIS_X | PMC4BPCI_AXIS_Y |
                                        PMC4BPCI_AXIS_Z | PMC4BPCI_AXIS_U ;

            OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
            if(OpenFlag!=MMC_OK)
            {
                        printf("Initialize error!\n");
                        return;
            }

rtn = pmc4bpci_smove_stop(MMC_CARD_NO, axis);

if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_pos_smove, pmc4bpci_pos_smove_wait, pmc4bpci_set_mod3

6.10 set_msgdispatch

MMC_INT16U **pmc4bpci_set_msgdispatch**(PMC4BPCI_MSG_DISPATCH *funcHandler,* MMC_INT32U *wParam*);

(1) Descriptions

It provides Hooking function to use window function at MMC library.

(2) Factors

- funcHandler: Function type pointer
- wParam: funcHandler function send factors

(3) Return value

This function is always successful and it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

// Initialized interrupt process function by pmc4bpci_open() function
MMC_VOID MyHookFunc(MMC_INT32U wParam)
{
 // Message Dispatch
}

void main()
{
MMC_INT16U    rtn;
int OpenFlag;

OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
if(OpenFlag!=MMC_OK)
{
 printf("Initialize error!\n");
 return;
}

rtn = pmc4bpci_set_msgdispatch( MyHookFunc , 0);
if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

pmc4bpci_close_all();
}
```


6.11 command

MMC_INT16U **pmc4bpci_command**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT16U *wCmd*);

(1) Descriptions

It commands to the selected axis. For more description of command code, refer to the user manual.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wCmd: Command value

Command code
PMC4BPCI_CMD_R: Range setting (8,000,000:magnification=1 to 16,000:magnification:500),4 byte
PMC4BPCI_CMD_K: Jerk speed setting (1 to 65535), 2 byte
PMC4BPCI_CMD_A: Acceleration setting (1 to 8000), 2 byte
PMC4BPCI_CMD_D: Deceleration setting (1 to 8000), 2 byte
PMC4BPCI_CMD_SV: Initial speed setting (1 to 8000), 2 byte
PMC4BPCI_CMD_V: Drive speed setting (1 to 8000), 2 byte
PMC4BPCI_CMD_P: Number of output pulses (0 to 268,435,455), 4 byte
Interpolation end point (-8,388,608 to 8,388,607), 4 byte
PMC4BPCI_CMD_DP: Manual deceleration point setting (0 to 2g), 4 byte
PMC4BPCI_CMD_C: Circular center coordinate setting (-8m to 8m), 4 byte
PMC4BPCI_CMD_LP: Logical position counter setting (-2g to 2g),4 byte
PMC4BPCI_CMD_EP: Actual position counter setting (-2g to 2g), 4 byte
PMC4BPCI_CMD_CP: COMP +register setting (-2g to 2g), 4 byte
PMC4BPCI_CMD_CM: COMP -register setting (-2g to 2g), 4 byte
PMC4BPCI_CMD_AO: Acceleration counter offset setting (0 to 65535), 2 byte
PMC4BPCI_CMD_AD: Decel/Accel setting (1 to 65535), 2 byte
PMC4BPCI_CMD_NOP: NOP(For axis exchange)
PMC4BPCI_CMD_HS_V: Home search speed setting (1 to 8,000), 2 byte
PMC4BPCI_CMD_HS_RUN: Home search run
PMC4BPCI_CMD_RST: Reset

Data read command
PMC4BPCI_READ_LP: Logical position counter reading (-2g to 2g),4 byte
PMC4BPCI_READ_EP: Actual position counter reading (-2g to 2g),4 byte
PMC4BPCI_READ_CV: Current drive speed reading (1 to 8000),2 byte
PMC4BPCI_READ_CA: Current accel/decel reading (1 to 8000),2 byte

Drive command
PMC4BPCI_DRV_PF: +direction constant speed drive
PMC4BPCI_DRV_MF: -direction constant speed drive
PMC4BPCI_DRV_PC: +direction consecutive drive
PMC4BPCI_DRV_MC: -direction consecutive drive
PMC4BPCI_DRV_DH: Drive start hold
PMC4BPCI_DRV_DF: Drive start free/end status clear
PMC4BPCI_DRV_DDS: Drive deceleration stop
PMC4BPCI_DRV_DS: Drive immediate stop

Interpolation command
PMC4BPCI_INP_2LD: 2-axis linear interpolation drive
PMC4BPCI_INP_3LD: 3-axis linear interpolation drive
PMC4BPCI_INP_CW: CW circular interpolation drive
PMC4BPCI_INP_CCW: CCW circular interpolation drive

PMC4BPCI_INP_2BP: 2-axis bit pattern interpolation drive
PMC4BPCI_INP_3BP: 3-axis bit pattern interpolation drive
PMC4BPCI_INP_BPE: BP register writable
PMC4BPCI_INP_BPD: BP register not writable
PMC4BPCI_INP_BPS: BP data stack
PMC4BPCI_INP_BPC: BP data clear
PMC4BPCI_INP_SS: Interpolation single step
PMC4BPCI_INP_DV1: Valid deceleration
PMC4BPCI_INP_DV2: Invalid deceleration
PMC4BPCI_INP_IC: Interpolation interrupt clear

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
        MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        // ……
        // example of range setting
        rtn = pmc4bpci_write_data(MMC_CARD_NO, PMC4BPCI_AXIS_X, 8000000);
        rtn = pmc4bpci_command(MMC_CARD_NO, PMC4BPCI_AXIS_X,
                                                PMC4BPCI_CMD_R);
//      Same result when using outw
//      rtn = pmc4bpci_outw(MMC_CARD_NO, wr0, PMC4BPCI_AXIS_X |
                                        PMC4BPCI_CMD_R);

        // example of range setting
//      pmc4bpci_set_range(MMC_CARD_NO, PMC4BPCI_AXIS_X, 8000000);

                if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }
                pmc4bpci_close_all();
}
```


(5) Reference macro

PMC4BPCI _CMD_R, PMC4BPCI _CMD_K, PMC4BPCI _CMD_A, PMC4BPCI _CMD_D,
PMC4BPCI _CMD_SV, PMC4BPCI _CMD_V, PMC4BPCI _CMD_P, PMC4BPCI_CMD_DP,
PMC4BPCI_CMD_C, PMC4BPCI_CMD_LP, PMC4BPCI_CMD_EP, PMC4BPCI_CMD_CP,
PMC4BPCI_CMD_CM, PMC4BPCI_CMD_AO, PMC4BPCI_CMD_NOP,
PMC4BPCI_CMD_RST,
PMC4BPCI_READ_LP, PMC4BPCI_READ_EP, PMC4BPCI_READ_CV,
PMC4BPCI_READ_CA,
PMC4BPCI_DRV_PF, PMC4BPCI_DRV_MF, PMC4BPCI_DRV_PC, PMC4BPCI_DRV_MC,
PMC4BPCI_DRV_DH, PMC4BPCI_DRV_DF, PMC4BPCI_DRV_DDS,
PMC4BPCI_DRV_DS,
PMC4BPCI_INP_2LD, PMC4BPCI_INP_3LD, PMC4BPCI_INP_CW,
PMC4BPCI_INP_CCW,
PMC4BPCI_INP_2BP, PMC4BPCI_INP_3BP, PMC4BPCI_INP_BPE, PMC4BPCI_INP_BPD,
PMC4BPCI_INP_BPS, PMC4BPCI_INP_BPC, PMC4BPCI_INP_SS, PMC4BPCI_INP_DV1,
PMC4BPCI_INP_DV2, PMC4BPCI_INP_IC

7 Initial setting

7.1 set_acac

MMC_INT16U **pmc4bpci_set_acac**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT16U *wdata*);

(1) Descriptions

It sets jerk speed.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: Jerk speed setting value(K), jerk speed=((62.5*10^6)/K)*magnification
  (Setting range: 1 to 65,535)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

rtn = pmc4bpci_set_acac(MMC_CARD_NO,
PMC4BPCI_AXIS_X|PMC4BPCI_AXIS_Y|PMC4BPCI_AXIS_Z|PMC4BPCI_AXIS_U, 100);

        if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

        pmc4bpci_close_all();
}
```


(5) Reference macro

PMC4BPCI_CMD_K

7.2 set_dcac

MMC_INT16U **pmc4bpci_set_dcac**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT16U *wdata*);

(1) Descriptions

It sets accel/decel.

(2) Factors

- id: Enters ID number. (range:0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: Accel/Decel setting value(L), Accel/Decel =((62.5*10^6)/L)*magnification
  (Setting range: 1 to 65,535)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

rtn = pmc4bpci_set_dcac(MMC_CARD_NO,
PMC4BPCI_AXIS_X|PMC4BPCI_AXIS_Y|PMC4BPCI_AXIS_Z|PMC4BPCI_AXIS_U, 100);

        if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

        pmc4bpci_close_all();
}
```

(5) Reference macro

PMC4BPCI_CMD_AD

7.3 set_acc

MMC_INT16U **pmc4bpci_set_acc**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT16 *wdata*);

(1) Descriptions

It sets acceleration.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC4BPCIP consists of 4-axis.
- wdata: Acceleration setting value (A), acceleration = A*125*magnification
  (Setting range: 1 to 8,000)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

rtn = pmc4bpci_set_acc(MMC_CARD_NO,
PMC4BPCI_AXIS_X|PMC4BPCI_AXIS_Y|PMC4BPCI_AXIS_Z|PMC4BPCI_AXIS_U, 100);

        if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

        pmc4bpci_close_all();
}
```

(5) Reference macro

PMC4BPCI_CMD_D

(6) Reference function

pmc4bpci_set_dec

7.4 set_dec

MMC_INT16U **pmc4bpci_set_dec**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT16 *wdata*);

(1) Descriptions

It sets deceleration.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: Deceleration setting value(D), deceleration = D*125*magnification
  (Setting range: 1 to 8,000)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

rtn = pmc4bpci_set_dec(MMC_CARD_NO,
PMC4BPCI_AXIS_X|PMC4BPCI_AXIS_Y|PMC4BPCI_AXIS_Z|PMC4BPCI_AXIS_U, 100);

        if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

        pmc4bpci_close_all();
}
```

(5) Reference macro

PMC4BPCI_CMD_A

(6) Reference function

pmc4bpci_set_acc

7.5 set_startv

MMC_INT16U **pmc4bpci_set_startv**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT16 *wdata*);

(1) Descriptions

It sets start speed of the selected axis.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: Start speed setting value(SV), Start speed = SV*magnification
  (Setting range: 1 to 8,000)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"


void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
rtn=pmc4bpci_set_startv(MMC_CARD_NO,PMC4BPCI_AXIS_X| PMC4BPCI_AXIS_Y,
                                                        1000);
// ……

        if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

        pmc4bpci_close_all();
}
```

(5) Reference macro

PMC4BPCI_CMD_SV

7.6 set_speed

MMC_INT16U **pmc4bpci_set_speed**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT16 *wdata*);

(1) Descriptions

It sets drive speed.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: Speed setting value(V), drive speed = V*magnification
  (Setting range 1 to 8,000)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
rtn=pmc4bpci_set_speed(MMC_CARD_NO, MC4BPCI_AXIS_X|PMC4BPCI_AXIS_Y,
1000);
// ……

        if(rtn!=MMC_OK)
{printf(“Fail!\n”);     }
else
{printf(“Complete OK!\n”);    }

        pmc4bpci_close_all();
}
```

(5) Reference macro

PMC4BPCI_CMD_V

7.7 set_pulse, set_endpoint

MMC_INT16U pmc4bpci_set_pulse(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT32 *wdata*);

MMC_INT16U pmc4bpci_set_endpoint(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT32 *wdata*);

(1) Descriptions

It sets the number of output pulses. It sets end point position at interpolation.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: Number of pulses/Interpolation end point position
  (Setting range: pulse 0 to 4,294,967,295, interpolation end point
  -2,147,483,646 to 2,147,483,646)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example1

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
rtn=pmc4bpci_set_pulse(MMC_CARD_NO, PMC4BPCI_AXIS_X, 100000);// 100000
pulses
pmc4bpci_command(MMC_CARD_NO, PMC4BPCI_AXIS_X, PMC4BPCI_DRV_MF);
//-direction constant speed drive
// ……

        if(rtn!=MMC_OK)
{printf(“Fail!\n”);     }
else
{printf(“Complete OK!\n”);    }
```

```
        pmc4bpci_close_all();
}
```

(5) Example2

```
    // Interpolation end point position setting
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
pmc4bpci_outw(MMC_CARD_NO, wr5, WR5_INP_LSPD0 | 0x0104);
// Selects constant linear velocity mode+interpolation axis
pmc4bpci_set_range(MMC_CARD_NO, PMC4BPCI_AXIS_X,4000000);// x2
pmc4bpci_set_range(MMC_CARD_NO, PMC4BPCI_AXIS_Y,5656000);
                                        // 4000000*1.414=5656000
pmc4bpci_set_startv(MMC_CARD_NO, PMC4BPCI_AXIS_X,500);      // x2 =1000pps
pmc4bpci_set_speed(MMC_CARD_NO, PMC4BPCI_AXIS_X,500);       // x2 =1000pps
pmc4bpci_set_center(MMC_CARD_NO, PMC4BPCI_AXIS_X,5000);
// Center axis X=5000
pmc4bpci_set_center(MMC_CARD_NO, PMC4BPCI_AXIS_Y,0);        // Center axis Y=0
pmc4bpci_set_endpoint(MMC_CARD_NO, PMC4BPCI_AXIS_X,5000);
// X end point
pmc4bpci_set_endpoint(MMC_CARD_NO, PMC4BPCI_AXIS_Y,-5000);
// Y end point
pmc4bpci_command(MMC_CARD_NO, 0x0,PMC4BPCI_INP_CW);
// CW circular interpolation drive start
// ……

                printf("Complete OK!\n");

                pmc4bpci_close_all();
}
```

(6) Reference macro

PMC4BPCI_INP_CCW, PMC4BPCI_INP_CW, PMC4BPCI_CMD_P

7.8 set_decpoint

MMC_INT16U **pmc4bpci_set_decpoint**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT32 *wdata*);

(1) Descriptions

It sets manual deceleration point.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: Manual deceleration point (Setting range: 0 to 4,294,967,295)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
rtn=pmc4bpci_set_decpoint(MMC_CARD_NO, PMC4BPCI_AXIS_X, 20000);
// ……

        if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

        pmc4bpci_close_all();
}
```

(5) Reference macro

PMC4BPCI_CMD_DP

7.9 set_center

MMC_INT16U **pmc4bpci_set_center**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT32 *wdata*);

(1) Descriptions

It sets circular center point.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: Circular center point (Based on current position)
  (Setting range: -2,147,483,648 to 2,147,483,647)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
rtn=pmc4bpci_set_center(MMC_CARD_NO, PMC4BPCI_AXIS_X, 20000);
rtn=pmc4bpci_set_center(MMC_CARD_NO, PMC4BPCI_AXIS_Y, -20000);
// ……

        if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

        pmc4bpci_close_all();
}
```

(5) Reference macro

PMC4BPCI_CMD_C

7.10 set_lpcounter

MMC_INT16U **pmc4bpci_set_lpcounter**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT32 *wdata*);

(1) Descriptions

It sets logical position counter value.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: Logical position counter value(Setting range: –2,147,483,648 to 2,147,483,647)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
pmc4bpci_set_lpcounter(MMC_CARD_NO, PMC4BPCI_AXIS_X|PMC4BPCI_AXIS_Y, 0);
// ……

        if(rtn!=MMC_OK)
{printf(“Fail!\n”);     }
else
{printf(“Complete OK!\n”);    }

        pmc4bpci_close_all();
}
```

(5) Reference macro

PMC4BPCI_CMD_LP

(6) Reference function

pmc4bpci_set_epcounter

7.11 set_epcounter

MMC_INT16U **pmc4bpci_set_epcounter**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT32 *wdata*);

(1) Descriptions

It sets actual position counter value.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: Actual position counter value(Setting range: –2,147,483,648 to 2,147,483,647)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
rtn=pmc4bpci_set_epcounter(MMC_CARD_NO,PMC4BPCI_AXIS_X|PMC4BPCI_AXIS_Y
,0);
// ……

        if(rtn!=MMC_OK)
{printf(“Fail!\n”);     }
else
{printf(“Complete OK!\n”);    }

        pmc4bpci_close_all();
}
```

(5) Reference macro

PMC4BPCI_CMD_EP

(6) Reference function

pmc4bpci_set_lpcounter

7.12 set_compplus

MMC_INT16U **pmc4bpci_set_compplus**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT32 *wdata*);

(1) Descriptions

It sets the range of COMP+ register.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: COMP + register setting value(Setting range: –2,147,483,648 to 2,147,483,647)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
rtn=pmc4bpci_set_compplus(MMC_CARD_NO, PMC4BPCI_AXIS_X, 10000);
// COMP+ register setting
// ……

if(rtn!=MMC_OK)
{printf(“Fail!\n”);     }
else
{printf(“Complete OK!\n”);    }

pmc4bpci_close_all();

}
```


(5) Reference macro

PMC4BPCI_CMD_CP, PMC4BPCI_CMD_CM

7.13 set_compminus

MMC_INT16U **pmc4bpci_set_compminus**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT32 *wdata*);

(1) Descriptions

It sets the range of COMP- register.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: COMP - register setting value (setting range: –2,147,483,648 to 2,147,483,647)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
rtn=pmc4bpci_set_compminus(MMC_CARD_NO, PMC4BPCI_AXIS_X, 10000);
// COMP- register setting
// ……

if(rtn!=MMC_OK)
{printf(“Fail!\n”);     }
else
{printf(“Complete OK!\n”);    }

pmc4bpci_close_all();

}
```


(5) Reference macro

PMC4BPCI_CMD_CP, PMC4BPCI_CMD_CM

7.14 set_accoffset

MMC_INT16U **pmc4bpci_set_accoffset**(MMC_INT16U *id*, MMC_INT16U *axis*, MMC_INT32 *wdata*);

(1) Descriptions

It sets acceleration counter offset. At reset, 8 is set.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- wdata: Acceleration counter offset value (Setting range: 0 to 65,535),

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
rtn= pmc4bpci_set_accoffset (MMC_CARD_NO, PMC4BPCI_AXIS_X, 8);
// Acceleration counter offset setting
// ……

if(rtn!=MMC_OK)
{printf(“Fail!\n”);     }
else
{printf(“Complete OK!\n”);    }

pmc4bpci_close_all();

}
```


(5) Reference macro

PMC4BPCI_CMD_AO

8 Calculation

8.1 calc_scale

MMC_INT32U **pmc4bpci_calc_scale**(MMC_INT32U *range*);

(1) Descriptions

Range magnification calculation. This function is able to use before calling pmc4bpci_open() function.

(2) Factors

Enters the range of drive range value. (range: 16,000 to 8,000,000)

(3) Return value

When it is out of min./max. value supported in PMC-4B-PCI board, it returns MMC_ILLEGAL_PARAMETER. When the execution is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U scale;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        // Input value is 800000 and return value is 10.
        scale = pmc4bpci_calc_scale(800000);

        printf("%d\n", scale);

        pmc4bpci_close_all();
}
```

(5) Reference macro

PMC4BPCI_RANGE

8.2 calc_acceleration

MMC_INT32U **pmc4bpci_calc_acceleration**(MMC_INT16U *acc*, MMC_INT32U *range*);

(1) Descriptions

Acceleration calculation. This function is able to use before calling pmc4bpci_open() function.

(2) Factors

- acc: Enters acceleration setting value. (range: 1 to 8,000)
- range: Enters drive range. (range: 16,000 to 8,000,000)

(3) Return value

When it is out of the range of acceleration(acc) and range value(range) supported in PMC-4B-PCI board, it returns MMC_ILLEGAL_PARAMETER. When the execution is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT32U ac;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        // acceleration=acc*125*(8000000/range)
        ac = pmc4bpci_calc_acceleration(100, 800000);

        printf("acceleration : %d\n", ac);

        pmc4bpci_close_all();
}
```

(5) Reference macro

PMC4BPCI_RANGE

8.3 calc_velocity

MMC_INT16U **pmc4bpci_calc_velocity**(MMC_INT16U *vel*, MMC_INT32U *range*);

(1) Descriptions

Velocity calculation. This function is able to use before calling pmc4bpci_open() function.

(2) Factors

- vel: Enters velocity value. (range: 1 to 8,000)
- range: Enters drive range. (range: 16,000 to 8,000,000)

(3) Return value

When it is out of the range of velocity and range value supported in PMC-4B-PCI board, it returns PMC4BPCI_ILLEGAL_PARAMETER. When the execution is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U vel;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        // Velocity=vel*(8000000/range)=1000 PPS
        vel = pmc4bpci_calc_velocity(100, 800000);

        printf("Velocity : %d\n", vel);

        pmc4bpci_close_all();
}
```

(5) Reference macro

PMC4BPCI_RANGE

8.4 calc_acac

MMC_INT16U **pmc4bpci_calc_acac**(MMC_INT16U *acac*, MMC_INT32U *range*);

(1) Descriptions

Jerk speed calculation. This function is able to use before calling pmc4bpci_open() function.

(2) Factors

- acac: Enters jerk speed value. (range: 1 to 65,535)
- range: Enters drive range. (range: 16,000 to 8,000,000)

(3) Return value

When it is out of the range of jerk speed(acac) and range value(range) supported in PMC-4B-PCI board, it returns PMC4BPCI_ILLEGAL_PARAMETER. When the execution is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT32U acac;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        // jerk speed=(62500000/acac) * (8000000/range) = 100000 PPS/SEC2
        acac = pmc4bpci_calc_acac(6250, 800000);

        printf("jerk speed : %d\n", acac);

        pmc4bpci_close_all();
}
```

(5) Reference macro

PMC4BPCI_RANGE

8.5 util_math_dist

MMC_DOUBLE **util_math_dist**(MMC_VERTEX* *p1*, MMC_VERTEX* *p2*);

(1) Descriptions

It obtains between the point(p1) and the point (p2).
This function is able to use before calling pmc4bpci_open() function.

(2) Factors

- p1: First point
- p2: Second point

```
typedef struct {
        double x;                   // X-coordinate
        double y;                   // Y-coordinate
        union {
                double z;           // Z-coordinate
                double k;           // Bulge value of poly line arc
        };
} MMC_VERTEX;
```

(3) Return value

It returns the distance between two points.

(4) Example

```
 #include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"
#define MMC_CARD_NO     15

void main()
{
        MMC_DOUBLE dist;
        int OpenFlag;

        // p1 X-coordinate : 0, Y-coordinate: 0
        // p2 X-coordinate : 0, Y-coordinate: 5
        MMC_VERTEX p1 = {0,0}, p2={0,5};

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        dist = util_math_dist(&p1, &p2);

        printf("p1 , p2 distance : %f\n", dist);
        pmc4bpci_close_all();
}
```


(5) Reference

MMC_VERTEX

8.6 util_math_angle

MMC_DOUBLE **util_math_angle**(MMC_VERTEX* *p1*, MMC_VERTEX* *p2*);

(1) Descriptions

It obtains the angle based on the point (p1) to the point (p2).
This function is able to use before calling pmc4bpci_open() function.

(2) Factors

- p1: First point
- p2: Second point

(3) Return value

It returns the angle between two points.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_DOUBLE dist;
        int OpenFlag;

        // p1 X-coordinate: 1, Y-coordinate: 4
        // p2 X-coordinate: 5, Y-coordinate: 2
        MMC_VERTEX p1 = {1,4}, p2={5,2};

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        dist = util_math_angle(&p1, &p2);

        printf("angle between p1 and p2 : %f\n", dist);

        pmc4bpci_close_all();
}
```

(5) Reference

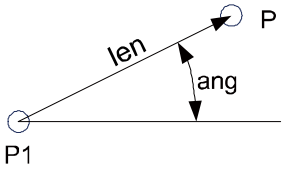
MMC_VERTEX, MMC_PI

8.7 util_math_polar

MMC_VOID **util_math_polar**(MMC_VERTEX* *p1*, MMC_DOUBLE *ang*, MMC_DOUBLE *len*, MMC_VERTEX* *p*);

(1) Descriptions

It obtains the coordinate which is based on the point (p1) to the ang direction with length (len) distance.
This function is able to use before calling pmc4bpci_open() function.



(2) Factors

- p1: Standard point
- ang: Direction(Radian)
- len: Away distance
- p: Return value(Point coordinate)

(3) Return value

It returns point coordinate. There is return value by function in itself.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15
#define MMC_PI 3.14159265358979323846

void main()
{
        int OpenFlag;

        MMC_VERTEX p1 = {0,0}, p;
        MMC_DOUBLE dist = 10.0;
        MMC_DOUBLE angle = MMC_PI * 30 / 180;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

       // Obtains the point based on the point (p1) to 30 degree direction with 10 distance.
        util_math_polar(&p1, angle, dist, &p);

        printf("X coordinate : %f, Y coordinate : %f\n", p.x, p.y);
        pmc4bpci_close_all();
```



```
}
```

(5) Reference

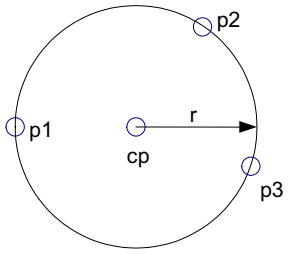
MMC_VERTEX, MMC_PI

8.8 util_math_circle_3p

MMC_VOID **util_math_circle_3p**(MMC_VERTEX* *p1*, MMC_VERTEX* *p2*, MMC_VERTEX* *p3*, MMC_VERTEX* *cp*, MMC_DOUBLE* *r*);

(1) Descriptions

It obtains center point (cp) of circle which passes three points (p1, p2, p3) and radius(r). This function is able to use before calling pmc4bpci_open() function.



(2) Factors

- p1: First point passing circle
- p2: Second point passing circle
- p3: Third point passing circle
- cp: Return value(Center point coordinate)
- r: Return value(Radius)

(3) Return value

It returns calculation result of center point coordinate(*cp*) and radius(*r*). There is return value by function in itself.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"
#define MMC_CARD_NO     15

void main()
{
        int OpenFlag;
        MMC_VERTEX p1 = {5,5}, p2 = {5,-5}, p3={10,0}, cp;
        MMC_DOUBLE r;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        util_math_circle_3p(&p1, &p2, &p3, &cp, &r);

        printf("center point X coordinate: %f, center point Ycoordinate: %f\n", cp.x, cp.y);
        printf("Radius distance at center point: %f\n", r);
```



```
            pmc4bpci_close_all();
}
```

(5) Reference

MMC_VERTEX, MMC_PI

9 Status reading

9.1 check_valid_id

MMC_INT16U **pmc4bpci_check_valid_id**(MMC_INT16 *id*);

(1) Descriptions

It checks that input id value is valid.

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

If Handle value relevanting id number exists, it returns MMC_OK. If it does not exist, it returns MMC_ERROR. If it is successful, MMC_OK value is set in globalvariable gError value. Otherwhise, MMC_ERROR value is set.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
        MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }
        // ……
        rtn = pmc4bpci_check_valid_id(MMC_CARD_NO);
        if(rtn==MMC_ERROR) {
                printf("ERROR: Invalid ID number\n");
        } else {
                printf("Found!\n");
        }
        // ……
        pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_open, pmc4bpci_close_all

9.2 get_logicalposition

MMC_INT32 **pmc4bpci_get_logicalposition**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It reads current logical position.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects one axis among 4 axes of PMC4BPCI.

(3) Return value

When it successes reading, it returns inner counted logical position by the chipset MCX314 of MMCboard(PMC-4B-PCI board). When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. Same value is recorded at gError as global error check variable. (When it successes reading, MMC_OK is recorded.)

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
long x=pmc4bpci_get_logicalposition(MMC_CARD_NO, PMC4BPCI_AXIS_X);
long y=pmc4bpci_get_logicalposition(MMC_CARD_NO, PMC4BPCI_AXIS_Y);
// ……

printf("Complete OK!\n");

pmc4bpci_close_all();

}
```

(5) Reference macro

PMC4BPCI_READ_LP

9.3 get_encoderposition

MMC_INT32 **pmc4bpci_get_encoderposition**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It reads actual position of current encoder.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects one axis among 4 axes of PMC4BPCI.

(3) Return value

When it successes reading, it returns counted actual position by externally connected encoder of PMC-4B-PCI board. When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. Same value is recorded at gError as global error check variable. (When it successes reading, MMC_OK is recorded.)

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
long x=pmc4bpci_get_encoderposition(MMC_CARD_NO, PMC4BPCI_AXIS_X);
long y=pmc4bpci_get_encoderposition(MMC_CARD_NO, PMC4BPCI_AXIS_Y);
// ……

printf("Complete OK!\n");

pmc4bpci_close_all();

}
```

(5) Reference macro

PMC4BPCI_READ_EP

9.4 get_currentvelocity

MMC_INT16U **pmc4bpci_get_currentvelocity**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It reads current velocity.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects one axis among 4 axes of PMC4BPCI.

(3) Return value

When it successes reading, it returns the set current velocity value by the chipset MCX314 of PMC-4B-PCI board. When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. Same value is recorded at gError as global error check variable. (When it successes reading, MMC_OK is recorded.)

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U velocity;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
velocity=pmc4bpci_get_currentvelocity(MMC_CARD_NO, PMC4BPCI_AXIS_X);
// ……

printf("Complete OK!\n");

pmc4bpci_close_all();

}
```

(5) Reference macro

PMC4BPCI_READ_CV

9.5 get_currentacc

MMC_INT16U **pmc4bpci_get_currentacc**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It reads current acceleration.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects one axis among 4 axes of PMC4BPCI.

(3) Return value

When it successes reading, it returns the set current acceleration value by the chipset MCX314 of PMC-4B-PCI. When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. Same value is recorded at gError as global error check variable. (When it successes reading, MMC_OK is recorded.)

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U acc;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
acc = pmc4bpci_get_currentacc (MMC_CARD_NO, PMC4BPCI_AXIS_X);
// ……

printf("Complete OK!\n");

pmc4bpci_close_all();

}
```

(5) Reference macro

PMC4BPCI_READ_CA

9.6 is_drive

MMC_INT16U **pmc4bpci_is_drive**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It displays drive status of each axis.
When return value is high level(1), it displays the drive pulse of selected axis is outputting.
When return value is low level(0), it displays the drive of selected axis is ending.
pmc4bpci_wait() function and **pmc4bpci_is_drive**() function check same bit of RR0 register.
However, pmc4bpci_wait() function waits only end signal (low level(0)) of drive pulse and **pmc4bpci_is_drive**() function does not wait end signal and returns only current status.
When inposition input signal for servo motor (nINPOS) is set as valid, drive pulse output signal(nINPOS) becomes active and returns to low level(0).
It is able to check by referring D0 to D3 bit of RR0 register not using the above function.
(D0=X axis, D1=Y axis, D2=Z axis, D3=U axis)

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When output status logic is high level(1), it returns MMC_HIGH_LEVEL. When output status logic is low level(0), it returns MMC_LOW_LEVEL. Related value for success and error is recorded at global error check variable gError.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

rtn = pmc4bpci_is_drive(MMC_CARD_NO,PMC4BPCI_AXIS_X+PMC4BPCI_AXIS_Y);

        if(rtn!=MMC_OK)
        {       printf("Not Move!\n");          }
        else
        {       printf("Move!\n");  }
                pmc4bpci_close_all();
```

}

(5) Reference function

pmc4bpci_wait

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL

9.7 is_idrive

MMC_INT16U **pmc4bpci_is_idrive**(MMC_INT16U *id*);

(1) Descriptions

It displays interpolation drive.
During interpolation driving, D8 bit of RR0 is high level(1). The other case, it is low level(0).

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

When interpolation drive status logic is high level, it returns MMC_HIGH_LEVEL. When interpolation drive status logic is low level, it returns MMC_LOW_LEVEL. Related value for success and error is recorded at global error check variable gError.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

rtn = pmc4bpci_is_idrive(MMC_CARD_NO);

        if(rtn!=MMC_OK)
        {       printf("Not Move!\n");          }
        else
        {       printf("Move!\n");  }

                pmc4bpci_close_all();
}
```

(5) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR0_I_DRV

9.8 flag_error

MMC_INT16U **pmc4bpci_flag_error**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It displays error occurance status of the specified axis.
When it checks the specified axis among D4 to D7 bit of RR0, and even one axis becomes high level(1), **pmc4bpci_flag_error**() returns error.(D4=X axis, D5=Y axis, D6=Z axis, D7=U axis)

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When interpolation drive status logic level is high, it returns MMC_HIGH_LEVEL. When it is low, it returns MMC_LOW_LEVEL. Related value for success and error is recorded at global error check variable gError.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
                        MMC_INT16U    rtn;
                        int OpenFlag;

                        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
                        if(OpenFlag!=MMC_OK)
                        {
                                printf("Initialize error!\n");
                                return;
                        }

rtn = pmc4bpci_flag_error(MMC_CARD_NO,
            PMC4BPCI_AXIS_X|PMC4BPCI_AXIS_Y);

                        if(rtn!=MMC_OK)
                        {       printf("Not Error!\n");         }
                        else
                        {       printf("Error!\n");   }

                        pmc4bpci_close_all();
}
```


(5) Reference function

pmc4bpci_is_error , Pmc4bpci_error_slimitplus, MMC_ERROR_slimitminus, MMC_ERROR_hlimitplus, MMC_ERROR_hlimitminus, MMC_ERROR_servoalarm, MMC_ERROR_emergency

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR0_X_ERR, RR0_Y_ERR, RR0_Z_ERR, RR0_U_ERR, RR2_SLMTP, RR2_SLMTM, RR2_HLMTP, RR2_HLMTM, RR2_ALARM, RR2_EMG

9.9 error

MMC_INT16U **pmc4bpci_error**();

(1) Descriptions

This function returns error status.

(2) Return value

MMC_OK	1
MMC_FALSE	0
MMC_IOADDRESS_ERR	6
MMC_OPEN_ERR	5
MMC_TIMEOUT_ERR	7
MMC_INVALID_AXIS	8
MMC_ILLEGAL_PARAMETER	9
MMC_ZERO_PARAMETER	10
MMC_ERROR	11
MMC_QUIT	12
MMC_INVALID_CARD	13

(3) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        level = pmc4bpci_error ();
        printf("error value : %d\n", level);
        pmc4bpci_close_all();
}
```

(4) Reference function

pmc4bpci_error_slimitplus, pmc4bpci_error_slimitminus,
pmc4bpci_error_hlimitplus, pmc4bpci_error_hlimitminus,
pmc4bpci_error_servoalarm, pmc4bpci_error_emergency

(5) Reference macro

MMC_OK, MMC_FALSE, MMC_IOADDRESS_ERR, MMC_OPEN_ERR,
MMC_TIMEOUT_ERR, MMC_INVALID_AXIS, MMC_ILLEGAL_PARAMETER,
MMC_ZERO_PARAMETER, MMC_ERROR, MMC_QUIT

9.10 flag_nextcommand

MMC_INT16U **pmc4bpci_flag_nextcommand**(MMC_INT16U *id*);

(1) Descriptions

It displays writable function the next data at consecutive interpolation.
When D9 (CNEXT) of RR0 is high level(1) in consecutive interpolation, parameter and interpolation command are writable for the next node.

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When the next data is writable in consecutive interpolation, it returns MMC_HIGH_LEVEL. When the next data is not writable, it returns MMC_LOW_LEVEL. It sets as MMC_OK at global error check variable gError.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        rtn = pmc4bpci_flag_nextcommand(MMC_CARD_NO);

        if(rtn!=MMC_OK)
        {       printf("Possible!\n");          }
        else
        {       printf("Impossible!\n");        }

                pmc4bpci_close_all();
}
```

(5) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR0_CNEXT

9.11 is_zone

MMC_INT16U **pmc4bpci_is_zone**(MMC_INT16U *id*);

(1) Descriptions

It displays current drive position in circular interpolation drive.
It displays current position as D10 to D12 bit of RR0 and the value means as below. The below table's each number is same as the position of circular interpolation drive as the below figure.

Table. Circular interpolation drive position

D12	D11	D10	Circular interpolation drive position
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

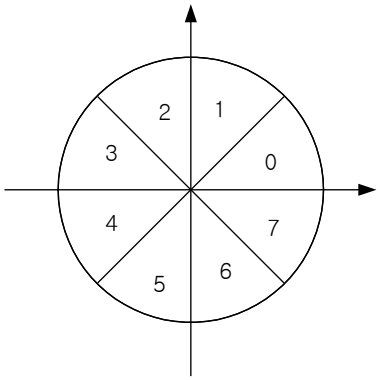


Figure. Circular interpolation drive position

(2) Factors

id: Enters ID number. (range: 0 to 15)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When the next data is writable in consecutive interpolation, it returns MMC_HIGH_LEVEL. When the next data is not writable, it returns MMC_LOW_LEVEL. It sets as PMC4BPCI_OK at global error check variable gError.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
```



```
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        rtn = pmc4bpci_is_zone(MMC_CARD_NO);

        printf(" circular interpolation current position : %d\n", rtn);

        pmc4bpci_close_all();
}
```

(5) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR0_CNEXT, RR0_ZONE0, RR0_ZONE1, RR0_ZONE2

9.12 is_bpstackcounter

MMC_INT16U **pmc4bpci_is_bpstackcounter**(MMC_INT16U *id*);

(1) Descriptions

It displays current stack counter(SC) position in bit pattern interpolation drive. SC position displays as D13, D14 bit of RR0 and the meansing is as below table.

Table. Stack Counter(SC) value

D14	D13	Stack Counter(SC) value
0	0	0
0	1	1
1	0	2
1	1	3

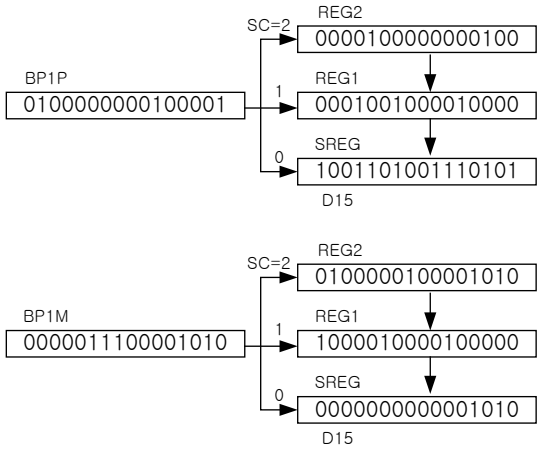


Figure. Register configuration of bit pattern interpolation

When recording bit pattern data to BP1P 16bit register, recorded register by SC value is changed.
As above example, when SC=2, BP1P register contents moves to REG2 register. When SC=1, BP1P register contents moves to REG1. When SC=0, BP1P register contents moves to SREG register. MCX314 chip of PMC4BPCI outputs drive pulse as SREG register value from lower bit in order. When SREG register's all contents outputs and SC value is bigger than 0, 16bit register contents moves to REG2 → REG1 → SREG register in order. BP1P means + direction, and BP1M means – direction.

(2) Factors

id: Enters ID number. (range: 0 to 15)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. It returns Stack Counter(SC) value in bit pattern interpolation. It sets as PMC4BPCI_OK at global error check variable gError.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
```



```
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        rtn = pmc4bpci_is_bpstackcounter(MMC_CARD_NO);

        printf("Stack Counter(SC) value : %d\n", rtn);

        pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_is_bpdrive

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR0_BPSC0, RR0_BPSC1

9.13 is_bpdrive

MMC_INT16U **pmc4bpci_is_bpdrive**(MMC_INT16U *id*);

(1) Descriptions

It check whether to writable bit pattern data in bit pattern interpolation drive.

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When stacks are full in bit pattern interpolation, it returns MMC_HIGH_LEVEL. When any one stack is empty, it returns MMC_LOW_LEVEL. It sets PMC4BPCI_OK at global error check variable gError.

(4) Example

```
 #include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        rtn = pmc4bpci_is_bpdrive (MMC_CARD_NO);

        if(rtn == MMC_OK)                       // if it cannot record data at bit
pattern
        { printf("bit pattern data not recordable!\n"); }
        else                                    // if it can record data at bit pattern
        { printf("bit pattern data recordable!\n"); }

        printf("Stack Counter(SC) value : %d\n", rtn);

        pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_is_bpstackcounter

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR0_BPSC0, RR0_BPSC1

9.14 error_servoalarm

MMC_INT16U **pmc4bpci_error_servoalarm**(MMC_INT16U *id*);

(1) Descriptions

It returns high level(1) when alarm signal for servo motor (nALARM) is active level at valid setting.
After checking whether to use error by pmc4bpci_is_error() function, use **pmc4bpci_error_servoalarm**() function. To use **pmc4bpci_error_servoalarm**() function not using pmc4bpci_is_error() function, select the axis at first. It is able to search by D4(ALARM) bit of RR2 register.

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When alarm signal for servo motor(nALARM) is active level at valid setting, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

Refer to pmc4bpci_error_slimitplus() function.

(5) Reference function

pmc4bpci_error_slimitplus, pmc4bpci_error_slimitminus,
pmc4bpci_error_hlimitplus, pmc4bpci_error_hlimitminus,
pmc4bpci_error_servoalarm, pmc4bpci_error_emergency

(6) Reference macro

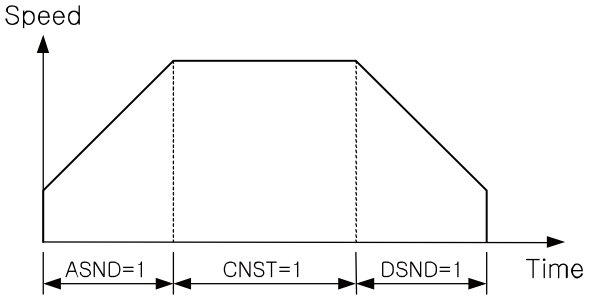
MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR2_SLMTP, RR2_SLMTM, RR2_HLMTP, RR2_HLMTM, RR2_ALARM, RR2_EMG

9.15 flag_aacc

MMC_INT16U **pmc4bpci_flag_aacc**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It determines whether it is acceleration in accel/decel drive.
When it is acceleration, it returns high level(1). It is able to determine as D2(ASND) bit of RR1 register.



(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. In accel/decel drive, when it is accelerating, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        level = pmc4bpci_flag_aacc(MMC_CARD_NO, PMC4BPCI_AXIS_X);

        if(level == MMC_OK)         // When it is accelerating in accel/decel drive
        { printf("Acceleration driving!\n"); }
```

```
pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_flag_aacc, pmc4bpci_flag_cons, pmc4bpci_flag_dacc

(6) Reference macro

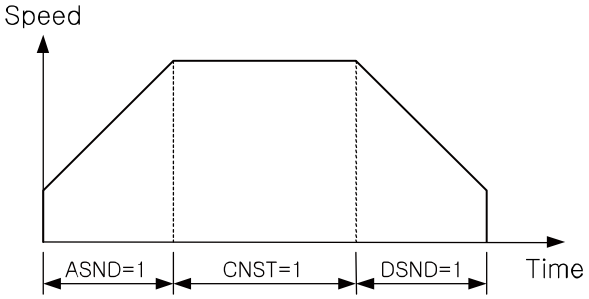
MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR1_ASND, RR1_CNST, RR1_DSND

9.16 flag_cons

MMC_INT16U **pmc4bpci_flag_cons**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It determines whether it is constant speed in accel/decel drive.
When it is constant speed, it returns high level(1). It is able to determine as D3(CNST) bit of RR1 register.



(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS.
In accel/decel drive, when it is constant speed, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

level = pmc4bpci_flag_cons(MMC_CARD_NO, PMC4BPCI_AXIS_X);

if(level == MMC_OK)         // In accel/decel drive, it is constant speeding
{ printf("constant speed driving!\n"); }
```



```
 pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_flag_aacc, pmc4bpci_flag_cons, pmc4bpci_flag_dacc

(6) Reference macro

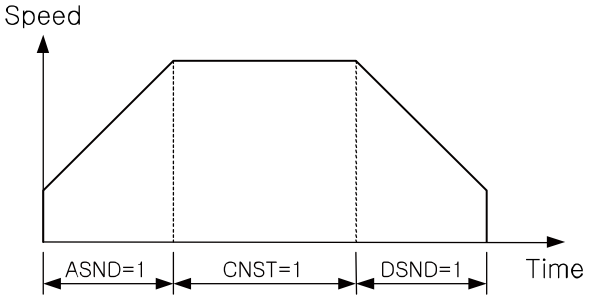
MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR1_ASND, RR1_CNST, RR1_DSND

9.17 pmc4bpci_flag_dacc

MMC_INT16U **pmc4bpci_flag_dacc**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It determines whether it is deceleration in accel/decel drive.
When it is deceleration, it returns high level(1). It is able to determine as D4(DSND) bit of RR1 register.



(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. In accel/decel drive, when it is deceleration, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

level = pmc4bpci_flag_dacc(MMC_CARD_NO, PMC4BPCI_AXIS_X);

if(level == MMC_OK)         // In accel/decel drive, it is decelcerating
{ printf("Decelerate driving!\n"); }
```



```
 pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_flag_aacc, pmc4bpci_flag_cons, pmc4bpci_flag_dacc

(6) Reference macro

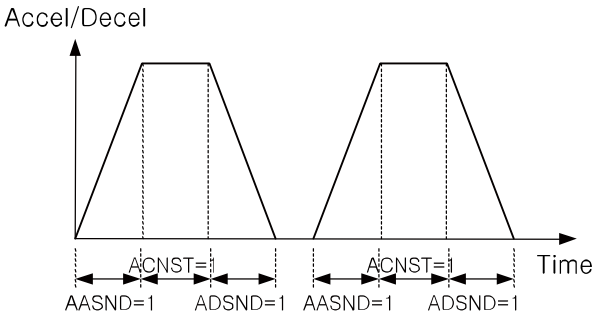
MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR1_ASND, RR1_CNST, RR1_DSND

9.18 flag_aacac

MMC_INT16U **pmc4bpci_flag_aacac**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It determines whether acceleration/deceleration increase in S curve drive.
When the value increases, it returns high level(1). It is able to determine as D5(AASND) bit of RR1 register.



(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. In S curve drive, when acceleration/deceleration value increase, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

level = pmc4bpci_flag_aacac(MMC_CARD_NO, PMC4BPCI_AXIS_X);

if(level == MMC_OK)         // In S curve drive, acceleration/deceleration is increasing
{ printf("acceleration/deceleration increase !\n"); }
```

```
 pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_flag_aacac, pmc4bpci_flag_acons, pmc4bpci_flag_dacac

(6) Reference macro

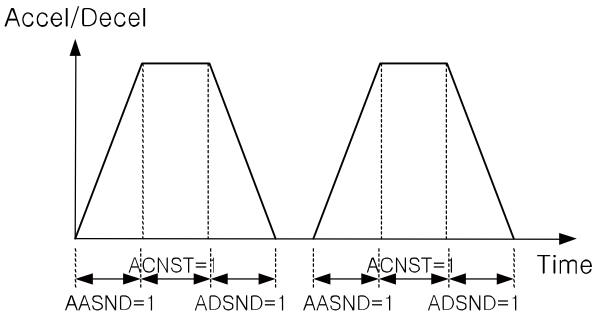
MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR1_AASND, RR1_ACNST, RR1_ADSND

9.19 flag_acons

MMC_INT16U **pmc4bpci_flag_acons**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It determines whether acceleration/deceleration increase in S curve drive.
When the value is constant, it returns high level(1). It is able to determine as D6(ACNST) bit of RR1 register.



(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. In S curve drive, when the acceleration/deceleration value is constant, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

level = pmc4bpci_flag_acons(MMC_CARD_NO, PMC4BPCI_AXIS_X);

if(level == MMC_OK)         // In S curve drive, when acceleration/deceleration is constant
{ printf("acceleration/deceleration constant!\n"); }
```

```
 pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_flag_aacac, pmc4bpci_flag_acons, pmc4bpci_flag_dacac

(6) Reference macro

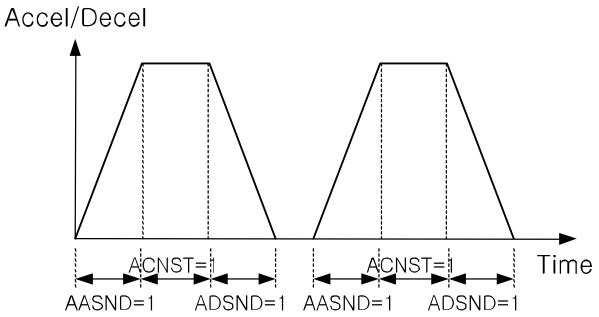
MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR1_AASND, RR1_ACNST, RR1_ADSND

9.20 flag_dacac

MMC_INT16U **pmc4bpci_flag_dacac**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It determines whether acceleration/deceleration increase in S curve drive.
When the value decreases, it returns high level(1). It is able to determine as D7(ADSND) bit of RR1 register.



(2) Factors

- id: Enters ID number. (range:0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. In S curve drive, when acceleration/deceleration value decrease, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

level = pmc4bpci_flag_dacac(MMC_CARD_NO, PMC4BPCI_AXIS_X);

if(level == MMC_OK)         // In S curve drive, when acceleration/deceleration decreases
{ printf("acceleration/deceleration decrease!\n"); }
```

```
pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_flag_aacac, pmc4bpci_flag_acons, pmc4bpci_flag_dacac

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL,
RR1_AASND, RR1_ACNST, RR1_ADSND

9.21 flag_compp

MMC_INT16U **pmc4bpci_flag_compp**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It displays the size between logical/actual position counter and COMP+register.
High level(1): Logical/Actual position counter ≥ COMP+ register
Low level(0): Logical/Actual position counter < COMP+ register
It is able to compare by reading D0(CMP+)bit of RR1 register.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When logical/actual position counter value is over COMP+ register value, it returns MMC_HIGH_LEVEL. When logical/actual position counter value is lower than COMP+ register value, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        level = pmc4bpci_flag_compp(MMC_CARD_NO, PMC4BPCI_AXIS_X);

        if(level == MMC_OK)         // Logical/Actual position counter  ≥  COMP+ register
        { printf("Logical/Actual position counter  ≥  COMP+\n"); }
        else                        // Logical/Actual position counter < COMP+ register
        { printf("Logical/Actual position counter < COMP+\n"); }
        pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_flag_compm

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR1_CMPP, RR1_CMPM

9.22 flag_compm

MMC_INT16U **pmc4bpci_flag_compm**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It displays the size between logical/actual position counter and COMP-register.
High level(1): Logical/Actual position counter <= COMP- register
Low level(0): Logical/Actual position counter > COMP- register
It is able to compare by reading D1(CMP-) bit of RR1 register.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When logical/actual position counter value is lower than COMP- register value, it returns MMC_HIGH_LEVEL. When logical/actual position counter value is bigger than COMP-register value, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        level = pmc4bpci_flag_compm(MMC_CARD_NO, PMC4BPCI_AXIS_X);

        if(level == MMC_OK)         // Logical/Actual position counter <= COMP- register
        { printf("Logical/Actual position counter <= COMP-\n"); }
        else                        // Logical/Actual position counter > COMP- register
        { printf("Logical/Actual position counter > COMP-\n"); }
 pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_flag_compp

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR1_CMPP, RR1_CMPM

10 Signal stop

10.1 flag_in0

MMC_INT16U **pmc4bpci_flag_in0**(MMC_INT16U *id*);

(1) Descriptions

When drive stops by external deceleration stop(nIN0), it returns high level(1). After checking that there is error or not by using pmc4bpci_is_stop() function, it uses **pmc4bpci_flag_in0**() function. To use **pmc4bpci_flag_in0**() function not using pmc4bpci_is_stop() function, select the axis at first. It is able to search by D8(IN0) bit of RR1 register.

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When drive generates external deceleration stop(nIN0) signal, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        level = pmc4bpci_is_stop(MMC_CARD_NO, PMC4BPCI_AXIS_X);

        if(MMC_ERROR()==MMC_OK)
        {
                if(level==MMC_OK)
                {
                        if(pmc4bpci_flag_in0(MMC_CARD_NO)==MMC_OK)
                        {       printf("in0 stop!\n");                  }
                        else if(pmc4bpci_flag_in1(MMC_CARD_NO)==MMC_OK)
                        {       printf("in1 stop!\n");                  }
                        else if(pmc4bpci_flag_in2(MMC_CARD_NO)==MMC_OK)
                        {       printf("in2 stop!\n");                  }
                        else if(pmc4bpci_flag_in3(MMC_CARD_NO)==MMC_OK)
```



```
                                {       printf("in3 stop!\n");          }
                                else
        if(pmc4bpci_flag_limitplus(MMC_CARD_NO)==MMC_OK)
                                {       printf("limitplus stop!\n");    }
                                else
        if(pmc4bpci_flag_limitminus(MMC_CARD_NO)==MMC_OK)
                                {       printf("limitminus stop!\n");   }
                                else
                                if(pmc4bpci_flag_servoalarm(MMC_CARD_NO)==MMC_OK)
                                {       printf("servoalarm stop!\n");   }
                                else
                                if(pmc4bpci_flag_emergency(MMC_CARD_NO)==MMC_OK)
                                {       printf("emergency stop!\n");    }
                        }
                }

                pmc4bpci_close_all();
        }
```

(5) Reference function

pmc4bpci_is_stop, pmc4bpci_flag_in0, pmc4bpci_flag_in1, pmc4bpci_flag_in2, pmc4bpci_flag_in3, pmc4bpci_flag_limitplus, pmc4bpci_flag_limitminus, pmc4bpci_flag_servoalarm, pmc4bpci_flag_emergency

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR1_IN0, RR1_IN1, RR1_IN2, RR1_IN3, RR1_LMTP, RR1_LMTM, RR1_ALARM, RR1_EMG

10.2 flag_in1

MMC_INT16U **pmc4bpci_flag_in1**(MMC_INT16U *id*);

(1) Descriptions

When drive stops by external deceleration stop(nIN1), it returns high level(1). After checking there is error or not by using pmc4bpci_is_stop() function, it uses **pmc4bpci_flag_in1**() function. To use **pmc4bpci_flag_in1**() function not using pmc4bpci_is_stop(), select the axis at first. It is able to search by D9(IN1) bit of RR1 register.

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When drive generates external deceleration stop(nIN1) signal, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

Refer to pmc4bpci_flag_in0() function.

(5) Reference function

pmc4bpci_is_stop, pmc4bpci_flag_in0, pmc4bpci_flag_in1, pmc4bpci_flag_in2, pmc4bpci_flag_in3, pmc4bpci_flag_limitplus, pmc4bpci_flag_limitminus, pmc4bpci_flag_servoalarm, pmc4bpci_flag_emergency

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR1_IN0, RR1_IN1, RR1_IN2, RR1_IN3, RR1_LMTP, RR1_LMTM, RR1_ALARM, RR1_EMG

10.3 flag_in2

MMC_INT16U **pmc4bpci_flag_in2**(MMC_INT16U *id*);

(1) Descriptions

When drive stops by external deceleration stop(nIN2), it returns high level(1). After checking that there is error or not by using pmc4bpci_is_stop() function, it uses **pmc4bpci_flag_in2**() function. To use **pmc4bpci_flag_in2**() function not using pmc4bpci_is_stop() function, select the axis at first. It is able to seach by D10(IN2) bit of RR1 register.

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When drive generates external deceleration stop(nIN2) signal, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

Refer to pmc4bpci_flag_in0() function.

(5) Reference function

pmc4bpci_is_stop, pmc4bpci_flag_in0, pmc4bpci_flag_in1, pmc4bpci_flag_in2, pmc4bpci_flag_in3, pmc4bpci_flag_limitplus, pmc4bpci_flag_limitminus, pmc4bpci_flag_servoalarm, pmc4bpci_flag_emergency

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR1_IN0, RR1_IN1, RR1_IN2, RR1_IN3, RR1_LMTP, RR1_LMTM, RR1_ALARM, RR1_EMG

10.4 flag_in3

MMC_INT16U **pmc4bpci_flag_in3**(MMC_INT16U *id*);

(1) Descriptions

When drive stops by external deceleration stop(nIN3), it returns high level(1). After checking that there is error or not by using pmc4bpci_is_stop() function, it uses **pmc4bpci_flag_in3**() function. To use **pmc4bpci_flag_in3**() function not using pmc4bpci_is_stop() function, select the axis at first. It is able to search by D11(IN3) bit of RR1 register.

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When drive generates external deceleration stop(nIN3) signal, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

Refer to pmc4bpci_flag_in0() function.

(5) Reference function

pmc4bpci_is_stop, pmc4bpci_flag_in0, pmc4bpci_flag_in1, pmc4bpci_flag_in2, pmc4bpci_flag_in3, pmc4bpci_flag_limitplus, pmc4bpci_flag_limitminus, pmc4bpci_flag_servoalarm, pmc4bpci_flag_emergency

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR1_IN0, RR1_IN1, RR1_IN2, RR1_IN3, RR1_LMTP, RR1_LMTM, RR1_ALARM, RR1_EMG

10.5 flag_limitplus

MMC_INT16U **pmc4bpci_flag_limitplus**(MMC_INT16U *id*);

(1) Descriptions

When drive stops by +direction limit signal(nLMTP), it returns high level(1). After checking that there is error or not by using pmc4bpci_is_stop() function, it uses **pmc4bpci_flag_limitplus**() function. To use **pmc4bpci_flag_limitplus**() function not using pmc4bpci_is_stop() function, select the axis at first. It is able to seach by D12(LMT+) bit of RR1 register.

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When drive generates +direction limit signal(nLMTP) signal, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

Refer to pmc4bpci_flag_in0() function.

(5) Reference function

pmc4bpci_is_stop, pmc4bpci_flag_in0, pmc4bpci_flag_in1, pmc4bpci_flag_in2, pmc4bpci_flag_in3, pmc4bpci_flag_limitplus, pmc4bpci_flag_limitminus, pmc4bpci_flag_servoalarm, pmc4bpci_flag_emergency

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR1_IN0, RR1_IN1, RR1_IN2, RR1_IN3, RR1_LMTP, RR1_LMTM, RR1_ALARM, RR1_EMG

10.6 flag_limitminus

MMC_INT16U **pmc4bpci_flag_limitminus**(MMC_INT16U *id*);

(1) Descriptions

When drive stops by -direction limit signal(nLMTM), it returns high level(1). After checking that there is error or not by using pmc4bpci_is_stop() function and it uses **pmc4bpci_flag_limitminus**() function. To use **pmc4bpci_flag_limitminus**() function not using pmc4bpci_is_stop() function, select the axis at first. It is able to search by D13(LMT-) bit of RR1 register.

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When drive generates -direction limit signal(nLMTM) signal, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

Refer to pmc4bpci_flag_in0() function.

(5) Reference function

pmc4bpci_is_stop, pmc4bpci_flag_in0, pmc4bpci_flag_in1, pmc4bpci_flag_in2, pmc4bpci_flag_in3, pmc4bpci_flag_limitplus, pmc4bpci_flag_limitminus, pmc4bpci_flag_servoalarm, pmc4bpci_flag_emergency

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR1_IN0, RR1_IN1, RR1_IN2, RR1_IN3, RR1_LMTP, RR1_LMTM, RR1_ALARM, RR1_EMG

10.7 flag_servoalarm

MMC_INT16U **pmc4bpci_flag_servoalarm**(MMC_INT16U *id*);

(1) Descriptions

When drive stops by alarm signal for servo motor(nALARM), it returns high level(1). After checking that there is error or not by using pmc4bpci_is_stop() function, it uses **pmc4bpci_flag_servoalarm**() function. To use **pmc4bpci_flag_servoalarm**() function not using pmc4bpci_is_stop() function, select the axis at first. It is able to seach by D14(ALARM) bit of RR1 register.

(2) Factors

id: Enters ID number. (range: 0 to 15)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When drive generates alarm signal for servo motor(nALARM) signal, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

Refer to pmc4bpci_flag_in0() function.

(5) Reference function

pmc4bpci_is_stop, pmc4bpci_flag_in0, pmc4bpci_flag_in1, pmc4bpci_flag_in2, pmc4bpci_flag_in3, pmc4bpci_flag_limitplus, pmc4bpci_flag_limitminus, pmc4bpci_flag_servoalarm, pmc4bpci_flag_emergency

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR1_IN0, RR1_IN1, RR1_IN2, RR1_IN3, RR1_LMTP, RR1_LMTM, RR1_ALARM, RR1_EMG

10.8 flag_emergency

MMC_INT16U **pmc4bpci_flag_emergency**(MMC_INT16U *id*);

(1) Descriptions

When drive stops by emergency stop signal(nEMG), it returns high level(1). After checking that there is error or not by using pmc4bpci_is_stop() function, it uses **pmc4bpci_flag_emergency**() function. To use **pmc4bpci_flag_emergency**() function not using pmc4bpci_is_stop() function, select the axis at first. It is able to search by D15(EMG) bit of RR1 register.

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When drive generates emergency stop signal(nEMG) signal, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

Refer to pmc4bpci_flag_in0() function.

(5) Reference function

pmc4bpci_is_stop, pmc4bpci_flag_in0, pmc4bpci_flag_in1, pmc4bpci_flag_in2, pmc4bpci_flag_in3, pmc4bpci_flag_limitplus, pmc4bpci_flag_limitminus, pmc4bpci_flag_servoalarm, pmc4bpci_flag_emergency

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR1_IN0, RR1_IN1, RR1_IN2, RR1_IN3, RR1_LMTP, RR1_LMTM, RR1_ALARM, RR1_EMG

10.9 is_error

MMC_INT16U **pmc4bpci_is_error**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It is error check function. Even any one of software limit(SLMT+/-), limit(HLMT+/-), servo motor alarm(ALARM), emergency stop signal(EMG) is set, it returns high level(1).

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. Even any one of software limit (SLMT+/-), limit (HLMT+/-), servo motor alarm(ALARM), emergency stop signal(EMG) is set, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        level = pmc4bpci_is_error(MMC_CARD_NO, PMC4BPCI_AXIS_X);

        if(level == MMC_OK)
        {       printf("error occurs!\n");              }

        pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_error_slimitplus, pmc4bpci_error_slimitminus,
pmc4bpci_error_hlimitplus, pmc4bpci_error_hlimitminus,
pmc4bpci_error_servoalarm, pmc4bpci_error_emergency

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR2_SLMTP, RR2_SLMTM, RR2_HLMTP, RR2_HLMTM, RR2_ALARM, RR2_EMG

10.10 error_slimitplus

MMC_INT16U **pmc4bpci_error_slimitplus**(MMC_INT16U *id*);

(1) Descriptions

When drive stops by +direction soft limit signal(nSLMTP), it returns high level(1). After checking that there is error or not by using pmc4bpci_is_error() function, it uses **pmc4bpci_error_slimitplus**() function. To use p**mc4bpci_error_slimitplus**() function not using pmc4bpci_is_error() function, select the axis at first. It is able to seach by D0(SLMT+) bit of RR2 register.

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When drive generates stop signal by +direction soft limit signal(nSLMTP), it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        level = pmc4bpci_is_error(MMC_CARD_NO, PMC4BPCI_AXIS_X);

        if(MMC_ERROR()==MMC_OK)
        {
                // If error factor occurs,
                if(level==MMC_OK)
                {
                        if(pmc4bpci_error_slimitplus(MMC_CARD_NO)==MMC_OK)
                        {       printf("+direction Soft Limit error!\n");                       }
                        else

if(pmc4bpci_error_slimitminus(MMC_CARD_NO)==MMC_OK)
                        {       printf("-direction Soft Limit error!\n");               }
                        else
```

```
                                    if(pmc4bpci_error_hlimitplus(MMC_CARD_NO)==MMC_OK)
                                    {       printf("+direction Limit signal error!\n");             }
                                    else

        if(pmc4bpci_error_hlimitminus(MMC_CARD_NO)==MMC_OK)
                                    {       printf("-direction Limit signal error!\n");             }
                                    else

        if(pmc4bpci_error_servoalarm(MMC_CARD_NO)==MMC_OK)
                                    {       printf("in0 stop!\n");                  }
                                    else

        if(pmc4bpci_error_emergency(MMC_CARD_NO)==MMC_OK)
                                    {       printf("emergencysignal error!\n");             }
                        }
                }

                pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_error_slimitplus, pmc4bpci_error_slimitminus,
pmc4bpci_error_hlimitplus, pmc4bpci_error_hlimitminus,
pmc4bpci_error_servoalarm, pmc4bpci_error_emergency

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL,
RR2_SLMTP, RR2_SLMTM, RR2_HLMTP, RR2_HLMTM, RR2_ALARM, RR2_EMG

10.11 error_slimitminus

MMC_INT16U **pmc4bpci_error_slimitminus**(MMC_INT16U *id*);

(1) Descriptions

If drive stops by -direction soft limit signal(nSLMTM), it returns high level(1). After checking that there is error or not by using pmc4bpci_is_error() function, it uses **pmc4bpci_error_slimitminus**() function. To use **pmc4bpci_error_slimitminus**() function not using pmc4bpci_is_error() function, select the axis at first. It is able to search by D1(SLMT-) bit of RR2 register.

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When drive generates stop signal by -direction soft limit signal(nSLMTM), it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

Refer to pmc4bpci_error_slimitplus() function.

(5) Reference function

pmc4bpci_error_slimitplus, pmc4bpci_error_slimitminus,
pmc4bpci_error_hlimitplus, pmc4bpci_error_hlimitminus,
pmc4bpci_error_servoalarm, pmc4bpci_error_emergency

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR2_SLMTP, RR2_SLMTM, RR2_HLMTP, RR2_HLMTM, RR2_ALARM, RR2_EMG

10.12 error_hlimitplus

MMC_INT16U **pmc4bpci_error_hlimitplus**(MMC_INT16U *id*);

(1) Descriptions

When drive stops by +direction limit signal(nLMTP), it returns high level(1). After checking that there is error or not by using pmc4bpci_is_error() function, it uses **pmc4bpci_error_hlimitplus**() function. To use **pmc4bpci_error_hlimitplus**() function not using pmc4bpci_is_error() function, select the axis at first. It is able to search by D2(HLMT+) bit of RR2 register.

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When drive generates stop signal by +direction limit signal(nLMTP), it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

Refer to pmc4bpci_error_slimitplus() function.

(5) Reference function

pmc4bpci_error_slimitplus, pmc4bpci_error_slimitminus,
pmc4bpci_error_hlimitplus, pmc4bpci_error_hlimitminus,
pmc4bpci_error_servoalarm, pmc4bpci_error_emergency

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR2_SLMTP, RR2_SLMTM, RR2_HLMTP, RR2_HLMTM, RR2_ALARM, RR2_EMG

10.13 error_hlimitminus

MMC_INT16U **pmc4bpci_error_hlimitminus**(MMC_INT16U *id*);

(1) Descriptions

When drive stops by -direction limit signal(nLMTM), it returns high level(1). After checking that there is error or not by using pmc4bpci_is_error() function, it uses **pmc4bpci_error_hlimitminus**() function. To use **pmc4bpci_error_hlimitminus**() function not using pmc4bpci_is_error() function, select the axis at first. It is able to search by D3(HLMT-) bit of RR2 register.

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When drive generates stop signal by -direction limit signal(nLMTM), it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

Refer to pmc4bpci_error_slimitplus() function.

(5) Reference function

pmc4bpci_error_slimitplus, pmc4bpci_error_slimitminus, pmc4bpci_error_hlimitplus, pmc4bpci_error_hlimitminus, pmc4bpci_error_servoalarm, pmc4bpci_error_emergency

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR2_SLMTP, RR2_SLMTM, RR2_HLMTP, RR2_HLMTM, RR2_ALARM, RR2_EMG

10.14 error_emergency

MMC_INT16U **pmc4bpci_error_emergency**(MMC_INT16U *id*);

(1) Descriptions

When emergency stop signal(nEMG) is low level(0), it returns high level(1). After checking that there is error or not by using pmc4bpci_is_error() function, it uses **pmc4bpci_error_emergency**() function. To use **pmc4bpci_error_emergency**() function, pmc4bpci_is_error() function select the axis at first. It is able to search D5(EMG) bit of RR2 register.

(2) Factors

- id: Enters ID number. (range: 0 to 15)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When emergency stop signal(nEMG) is low level(0), it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

Refer to pmc4bpci_error_slimitplus() function.

(5) Reference function

pmc4bpci_error_slimitplus, pmc4bpci_error_slimitminus,
pmc4bpci_error_hlimitplus, pmc4bpci_error_hlimitminus,
pmc4bpci_error_servoalarm, pmc4bpci_error_emergency

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR2_SLMTP, RR2_SLMTM, RR2_HLMTP, RR2_HLMTM, RR2_ALARM, RR2_EMG

10.15 is_stop

MMC_INT16U **pmc4bpci_is_stop**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

When drive stops by external deceleration stop(nIN0 to 3), +direction limit signal(nLMTP), -direction limit signal(nLMTM), alarm for servo motor signal(nALARM), emergency stop signal(nEMG), it returns high level(1). Even any one of becoming error factors in drive end status bit nLMTP, nLMTM, nALARM, nEMG becomes high level(1), nERR bit of RR0 becomes high level(1).

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When drive generates any one of external deceleration stop(nIN0 to 3), +direction limit signal(nLMTP), -direction limit signal(nLMTM), alarm signal for servo motor(nALARM), emergency stop signal(nEMG), etc, it returns MMC_HIGH_LEVEL. When drive generates none of stop signals, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

level = pmc4bpci_is_stop(MMC_CARD_NO, PMC4BPCI_AXIS_X);

if(level == MMC_OK)          // When generating stop
{ printf("Stop by external factors!\n"); }
pmc4bpci_close_all();
}
```


(5) Reference function

pmc4bpci_is_stop, pmc4bpci_flag_in0, pmc4bpci_flag_in1, pmc4bpci_flag_in2, pmc4bpci_flag_in3, pmc4bpci_flag_limitplus, pmc4bpci_flag_limitminus, pmc4bpci_flag_servoalarm, pmc4bpci_flag_emergency

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR1_IN0, RR1_IN1, RR1_IN2, RR1_IN3, RR1_LMTP, RR1_LMTM, RR1_ALARM, RR1_EMG

11 interrupt

11.1 is_interrupt

MMC_INT16U **pmc4bpci_is_interrupt**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

When interrupt occurs, it becomes high level(1).
Interrupt occurance factors are when pulse is rising (when drive pulse is positive logic),
when logical/actual position is bigger than COMP- register value,
when logical/actual position is lower than COMP- register value,
when logical/actual position is lower than COMP+ register value,
when logical/actual position is bigger than COMP+ register value,
when it ends pulse output to constant speed zone during accel/decel driving,
when it starts pulse output to constant speed zone in accel/decel drive.
If interrupt occurs by any interrupt occurance factors, interrupt status bits of RR3 register become high level(1), and interrupt output signal(INTN) becomes low level(0). When using **pmc4bpci_is_interrupt**() function, RR3 register contents are full of 0 and interrupt output signal returns to non active level.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When interrupt occurs, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        level = pmc4bpci_is_interrupt(MMC_CARD_NO, PMC4BPCI_AXIS_X);

        if(level == MMC_HIGH_LEVEL)
        {       printf("Interrupt occurs!\n");          }
```



```
        pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_interrupt_pulse, pmc4bpci_interrupt_pulseGEcompm, pmc4bpci_interrupt_pulseLcompm, pmc4bpci_interrupt_pulseLcompp, pmc4bpci_interrupt_pulseGEcompp, pmc4bpci_interrupt_cend, pmc4bpci_interrupt_cstart, pmc4bpci_interrupt_enddrive, pmc4bpci_get_interrupt

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR3_PULSE, RR3_PGECM, RR3_PLCM, RR3_PLCP, RR3_PGECP, RR3_CEND, RR3_CSTA, RR3_DEND

11.2 get_interrupt

MMC_INT16U **pmc4bpci_get_interrupt**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It returns interrupt status register value.
Interrupt occurance factors are when pulse is rising (when drive pulse is positive logic),
when logical/actual position is bigger than COMP- register value,
when logical/actual position is lower than COMP- register value,
when logical/actual position is lower than COMP+ register value,
when logical/actual position is bigger than COMP+ register value,
when it ends pulse output to constant speed zone during accel/decel driving,
when it starts pulse output to constant speed zone in accel/decel drive.
If interrupt occurs by any interrupt occurance factors, interrupt status bits of RR3 register become high level(1), and interrupt output signal(INTN) becomes low level(0). When using **pmc4bpci_is_interrupt**() function, RR3 register contents are full of 0 and interrupt output signal returns to non active level.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. For global error check variable gError, when it is successful, MMC_OK value is recorded. Otherwise, the related error is recorded.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    intval;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        intval = pmc4bpci_get_interrupt(MMC_CARD_NO, PMC4BPCI_AXIS_X);

        if(intval & RR3_PULSE)
        {       printf("Interrupt process by pulse\n");         }
        else if(intval & RR3_PGECM)
        {       printf("Drive pulse is rising edge(pulse rising moment at positive
logic)\n");}
```



```
            else if(intval & RR3_PLCM)
            {       printf("Logical/Actual position≥COMP- register\n");     }
            else if(intval & RR3_PLCP)
            {       printf("Logical /Actual position<COMP- register\n");                }
            else if(intval & RR3_PGECP)
            {       printf("Logical /Actual position<COMP+ register\n");                }
            else if(intval & RR3_CEND)
            {       printf("Logical /Actual position≥COMP+ register\n");    }
            else if(intval & RR3_CSTA)
            {       printf("Pulse output ends to constant speed zone in accel/decel drive\n");
 }
            else if(intval & RR3_DEND)
            {       printf("pulse output starts to constant speed zone in accel/decel drive
\n");       }

            pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_interrupt_pulse, pmc4bpci_interrupt_pulseGEcompm, pmc4bpci_interrupt_pulseLcompm, pmc4bpci_interrupt_pulseLcompp, pmc4bpci_interrupt_pulseGEcompp, pmc4bpci_interrupt_cend, pmc4bpci_interrupt_cstart, pmc4bpci_interrupt_enddrive, pmc4bpci_get_interrupt

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR3_PULSE, RR3_PGECM, RR3_PLCM, RR3_PLCP, RR3_PGECP, RR3_CEND, RR3_CSTA, RR3_DEND

11.3 interrupt_pulse

MMC_INT16U **pmc4bpci_interrupt_pulse**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It returns high level(1) when interrupt occurs (when drive pulse is set as positive logic, and the pulse is rising (rising edge)). When using the related interrupt status function, RR3 register contents are cleared and the related interrupt status function cannot be used over two times. It is able to determine as RR3 register D0(PULSE) bit.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When interrupt occurs, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example1

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        level = pmc4bpci_interrupt_pulse(MMC_CARD_NO, PMC4BPCI_AXIS_X);

        if(level == MMC_HIGH_LEVEL)
        {       printf("Interrupt by pulse\n");          }
        else
        {       printf("interrupt X\n");        }

        pmc4bpci_close_all();
}
```

(5) Example2

Refer to pmc4bpci_get_interrupt() function.

(6) Reference function

pmc4bpci_interrupt_pulse, pmc4bpci_interrupt_pulseGEcompm, pmc4bpci_interrupt_pulseLcompm, pmc4bpci_interrupt_pulseLcompp, pmc4bpci_interrupt_pulseGEcompp, pmc4bpci_interrupt_cend, pmc4bpci_interrupt_cstart, pmc4bpci_interrupt_enddrive, pmc4bpci_get_interrupt

(7) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR3_PULSE, RR3_PGECM, RR3_PLCM, RR3_PLCP, RR3_PGECP, RR3_CEND, RR3_CSTA, RR3_DEND

11.4 interrupt_pulseGEcompm

MMC_INT16U **pmc4bpci_interrupt_pulseGEcompm**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

When interrupt occurs(Logical/Actual position ≥ COMP- register), it returns high level(1). When using the related interrupt status function, RR3 register contents are cleared and the related interrupt status function cannot be used over two times. It is able to determine as RR3 register D1(P≥C-) bit.

(2) Factors

- id: Enters ID number. (range:0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When interrupt occurs, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example1

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        level = pmc4bpci_interrupt_ pulseGEcompm
 (MMC_CARD_NO, PMC4BPCI_AXIS_X);

        if(level == MMC_HIGH_LEVEL)
        {       printf("Logical/Actual position ≥ COMP- register interrupt\n");   }
        else
        {       printf("interrupt X\n");        }

        pmc4bpci_close_all();
}
```


(5) Example2

Refer to pmc4bpci_get_interrupt() function.

(6) Reference function

pmc4bpci_interrupt_pulse, pmc4bpci_interrupt_pulseGEcompm, pmc4bpci_interrupt_pulseLcompm, pmc4bpci_interrupt_pulseLcompp, pmc4bpci_interrupt_pulseGEcompp, pmc4bpci_interrupt_cend, pmc4bpci_interrupt_cstart, pmc4bpci_interrupt_enddrive, pmc4bpci_get_interrupt

(7) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR3_PULSE, RR3_PGECM, RR3_PLCM, RR3_PLCP, RR3_PGECP, RR3_CEND, RR3_CSTA, RR3_DEND

11.5 interrupt_pulseLcompm

MMC_INT16U **pmc4bpci_interrupt_pulseLcompm**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

When interrupt occurs (Logical/Actual position < COMP- register), it returns high level(1). When using the related interrupt status function, RR3 register contents are cleared and the related interrupt status function cannot be used over two times. It is able to determine as RR3 register D2(P<C-)bit.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When interrupt occurs, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example1

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        level = pmc4bpci_interrupt_ pulseLcompm
 (MMC_CARD_NO, PMC4BPCI_AXIS_X);

        if(level == MMC_HIGH_LEVEL)
        {       printf("Logical/Actual position < COMP- register interrupt\n");    }
        else
        {       printf("interrupt X\n");        }

        pmc4bpci_close_all();
}
```

(5) Example2

Refer to pmc4bpci_get_interrupt() function.

(6) Reference function

pmc4bpci_interrupt_pulse, pmc4bpci_interrupt_pulseGEcompm, pmc4bpci_interrupt_pulseLcompm, pmc4bpci_interrupt_pulseLcompp, pmc4bpci_interrupt_pulseGEcompp, pmc4bpci_interrupt_cend, pmc4bpci_interrupt_cstart, pmc4bpci_interrupt_enddrive, pmc4bpci_get_interrupt

(7) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR3_PULSE, RR3_PGECM, RR3_PLCM, RR3_PLCP, RR3_PGECP, RR3_CEND, RR3_CSTA, RR3_DEND

11.6 interrupt_pulseLcompp

MMC_INT16U **pmc4bpci_interrupt_pulseLcompp**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

When interrupt occurs (Logical/Actual position < COMP+ register), it returns high level(1). When using the related interrupt status function, RR3 register contents are cleared and the related interrupt status function cannot be used over two times. It is able to determine as RR3 register D3(P<C+)bit.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When interrupt occurs, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example1

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        level = pmc4bpci_interrupt_ pulseLcompp
 (MMC_CARD_NO, PMC4BPCI_AXIS_X);

        if(level == MMC_HIGH_LEVEL)
        {       printf("Logical/Actual position < COMP+ register interrupt\n");    }
        else
        {       printf("interrupt X\n");        }

        pmc4bpci_close_all();
}
```

(5) Example2

Refer to pmc4bpci_get_interrupt() function.

(6) Reference function

pmc4bpci_interrupt_pulse, pmc4bpci_interrupt_pulseGEcompm, pmc4bpci_interrupt_pulseLcompm, pmc4bpci_interrupt_pulseLcompp, pmc4bpci_interrupt_pulseGEcompp, pmc4bpci_interrupt_cend, pmc4bpci_interrupt_cstart, pmc4bpci_interrupt_enddrive, pmc4bpci_get_interrupt

(7) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR3_PULSE, RR3_PGECM, RR3_PLCM, RR3_PLCP, RR3_PGECP, RR3_CEND, RR3_CSTA, RR3_DEND

11.7 interrupt_pulseGEcompp

MMC_INT16U **pmc4bpci_interrupt_pulseGEcompp**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

When interrupt occurs(Logical/Actual position ≥ COMP+ register), it returns high level(1). When using the related interrupt status function, RR3 registercontents are cleared and the related interrupt status function cannot be used over two times. It is able to determine as RR3 register D4(P≥C+)bit.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When interrupt occurs, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example1

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        level = pmc4bpci_interrupt_ pulseGEcompp
 (MMC_CARD_NO, PMC4BPCI_AXIS_X);

        if(level == MMC_HIGH_LEVEL)
        {       printf("Logical/Actual position ≥ COMP+ register interrupt\n"); }
        else
        {       printf("interrupt X\n");        }

        pmc4bpci_close_all();
}
```

(5) Example2

Refer to pmc4bpci_get_interrupt() function.

(6) Reference function

pmc4bpci_interrupt_pulse, pmc4bpci_interrupt_pulseGEcompm,
pmc4bpci_interrupt_pulseLcompm, pmc4bpci_interrupt_pulseLcompp,
pmc4bpci_interrupt_pulseGEcompp, pmc4bpci_interrupt_cend, pmc4bpci_interrupt_cstart,
pmc4bpci_interrupt_enddrive, pmc4bpci_get_interrupt

(7) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR3_PULSE, RR3_PGECM, RR3_PLCM,
RR3_PLCP, RR3_PGECP, RR3_CEND, RR3_CSTA, RR3_DEND

11.8 interrupt_cend

MMC_INT16U **pmc4bpci_interrupt_cend**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

When interrupt occurs (pulse output ends to constant speed zone in accel/decel drive), it returns high level(1). When using the related interrupt status function, RR3 register contents are cleared and the related interrupt status function cannot be used over two times. It is able to determine as RR3 register D5(C-END) bit.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When interrupt occurs, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example1

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        level = pmc4bpci_interrupt_cend
 (MMC_CARD_NO, PMC4BPCI_AXIS_X);

        if(level == MMC_HIGH_LEVEL)
        {   printf("pulse output ends to constant speed zone in accel/decel drive\n");        }
        else
        {       printf("interrupt X\n");        }

        pmc4bpci_close_all();
}
```

(5) Example2

Refer to pmc4bpci_get_interrupt() function.

(6) Reference function

pmc4bpci_interrupt_pulse, pmc4bpci_interrupt_pulseGEcompm, pmc4bpci_interrupt_pulseLcompm, pmc4bpci_interrupt_pulseLcompp, pmc4bpci_interrupt_pulseGEcompp, pmc4bpci_interrupt_cend, pmc4bpci_interrupt_cstart, pmc4bpci_interrupt_enddrive, pmc4bpci_get_interrupt

(7) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR3_PULSE, RR3_PGECM, RR3_PLCM, RR3_PLCP, RR3_PGECP, RR3_CEND, RR3_CSTA, RR3_DEND

11.9 interrupt_cstart

MMC_INT16U **pmc4bpci_interrupt_cstart**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

When interrupt occurs (pulse output starts to constant speed zone in accel/decel drive), it returns high level(1). When using the related interrupt status function, RR3 register contents are cleared and the related interrupt status function cannot be used over two times. It is able to determine as RR3 register D6(C-STA) bit.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When interrupt occurs, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example1

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        level = pmc4bpci_interrupt_cstart(MMC_CARD_NO, PMC4BPCI_AXIS_X);

        if(level == MMC_HIGH_LEVEL)
        {   printf("pulse output starts to constant speed zone in accel/decel drive \n");       }
        else
        {           printf("interrupt X\n");         }

        pmc4bpci_close_all();
}
```

(5) Example2

Refer to pmc4bpci_get_interrupt() function.

(6) Reference function

pmc4bpci_interrupt_pulse, pmc4bpci_interrupt_pulseGEcompm,
pmc4bpci_interrupt_pulseLcompm, pmc4bpci_interrupt_pulseLcompp,
pmc4bpci_interrupt_pulseGEcompp, pmc4bpci_interrupt_cend, pmc4bpci_interrupt_cstart,
pmc4bpci_interrupt_enddrive, pmc4bpci_get_interrupt

(7) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR3_PULSE, RR3_PGECM, RR3_PLCM,
RR3_PLCP, RR3_PGECP, RR3_CEND, RR3_CSTA, RR3_DEND

11.10 interrupt_enddrive

MMC_INT16U **pmc4bpci_interrupt_enddrive**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

When interrupt occurs (drive ends), it returns high level(1). When using the related interrupt status function, RR3 registercontents are cleared and the related interrupt status function cannot be used over two times. It is able to determine as RR3 register D7(D-END) bit.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When interrupt occurs, it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. Same value is recorded at gError as global error check variable.

(4) Example1

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    level;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        level = pmc4bpci_interrupt_enddrive(MMC_CARD_NO, PMC4BPCI_AXIS_X);

        if(level == MMC_HIGH_LEVEL)
        {       printf("interrupt when drive ends\n");}
        else
        {       printf("interrupt X\n");        }

        pmc4bpci_close_all();
}
```

(5) Example2

Refer to pmc4bpci_get_interrupt() function.

(6) Reference function

pmc4bpci_interrupt_pulse, pmc4bpci_interrupt_pulseGEcompm, pmc4bpci_interrupt_pulseLcompm, pmc4bpci_interrupt_pulseLcompp, pmc4bpci_interrupt_pulseGEcompp, pmc4bpci_interrupt_cend, pmc4bpci_interrupt_cstart, pmc4bpci_interrupt_enddrive, pmc4bpci_get_interrupt

(7) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, RR3_PULSE, RR3_PGECM, RR3_PLCM, RR3_PLCP, RR3_PGECP, RR3_CEND, RR3_CSTA, RR3_DEND

12 Input signal status reading

12.1 input_status

MMC_INT16U **pmc4bpci_input_status**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It returns input signal status of the specified axis as 8 bit data.
It determines by dividing RR4 and RR5 register per 8 bit as 4 bytes. (It displays for Low byte of RR4 =X axis, High byte of RR4=Y axis, Low byte of RR5=Z axis, High byte of RR5=U axis.)

D7	D6	D5	D4	D3	D2	D1	D0
n-ALM	n-INP	n-EX-	n-EX+	n-IN3	n-IN2	n-IN1	n-IN0

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects only 1-axis.

(3) Return value

It returns the status of the specified axis as 8bit data, and MMC_OK is recorded at global error check variable gError. When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD(13). When it is out of axis range, it returns MMC_INVALID_AXIS(8). Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U    stat;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        stat = pmc4bpci_input_status(MMC_CARD_NO, PMC4BPCI_AXIS_X);

        // Low active normally high
        if(!(stat & n_IN0))
        {       printf("n_IN0 ON\n");       }
        else if(!(stat & n_IN1))
        {       printf("n_IN1 ON\n");       }
        else if(!(stat & n_IN2))
        {       printf("n_IN2 ON\n");       }
```



```
        else if(!(stat & n_IN3))
        {       printf("n_IN3 ON\n");       }
        else if(!(stat & n_EXPP))
        {       printf("n_EXPP ON\n");      }
        else if(!(stat & n_EXPM))
        {       printf("n_EXPM ON\n");      }
        else if(!(stat & n_INPOS))
        {       printf("n_INPOS ON\n");     }
        else if(!(stat & n_ALARM))
        {       printf("n_ALARM ON\n");     }

        pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_inputstatus_in0, pmc4bpci_inputstatus_in1, pmc4bpci_inputstatus_in2, pmc4bpci_inputstatus_in3, pmc4bpci_inputstatus_exp, pmc4bpci_inputstatus_exm, pmc4bpci_inputstatus_inpos, pmc4bpci_inputstatus_alarm

(6) Reference macro

n_IN0, n_IN1, n_IN2, n_IN3, n_EXPP, n_EXPM, n_INPOS, n_ALARM

12.2 inputstatus_in0 to in3, inputstatus_exp, inputstatus_exm, inputstatus_inpos, inputstatus_alarm

MC_INT16U **pmc4bpci_inputstatus_in0**(MMC_INT16U *id*, MMC_INT16U *axis*);

MC_INT16U **pmc4bpci_inputstatus_in1**(MMC_INT16U *id*, MMC_INT16U *axis*);

MC_INT16U **pmc4bpci_inputstatus_in2**(MMC_INT16U *id*, MMC_INT16U *axis*);

MC_INT16U **pmc4bpci_inputstatus_in3**(MMC_INT16U *id*, MMC_INT16U *axis*);

MC_INT16U **pmc4bpci_inputstatus_exp**(MMC_INT16U *id*, MMC_INT16U *axis*);

MC_INT16U **pmc4bpci_inputstatus_exm**(MMC_INT16U *id*, MMC_INT16U *axis*);

MC_INT16U **pmc4bpci_inputstatus_inpos**(MMC_INT16U *id*, MMC_INT16U *axis*);

MC_INT16U **pmc4bpci_inputstatus_alarm**(MMC_INT16U *id*, MMC_INT16U *axis*);

(1) Descriptions

It displays input signal status of the specified axis. It is divided RR4 and RR5 register per 8 bit as 4 bytes.

RR4	D7	D6	D5	D4	D3	D2	D1	D0
Low	X-ALM	X-INP	X-EX-	X-EX+	X-IN3	X-IN2	X-IN1	X-IN0
RR4	D15	D14	D13	D12	D11	D10	D9	D8
High	Y-ALM	Y-INP	Y-EX-	Y-EX+	Y-IN3	Y-IN2	Y-IN1	Y-IN0
RR5	D7	D6	D5	D4	D3	D2	D1	D0
Low	Z-ALM	Z-INP	Z-EX-	Z-EX+	Z-IN3	Z-IN2	Z-IN1	Z-IN0
RR5	D15	D14	D13	D12	D11	D10	D9	D8
High	U-ALM	U-INP	U-EX-	U-EX+	U-IN3	U-IN2	U-IN1	U-IN0

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects only 1-axis.

(3) Return value

When the status of the specified axis is high level(1), it returns MMC_HIGH_LEVEL. Otherwise, it returns MMC_LOW_LEVEL. When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void main()
{
        MMC_INT16U level;
        int OpenFlag;
```

```
        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

        level = pmc4bpci_inputstatus_in0(MMC_CARD_NO, PMC4BPCI_AXIS_X);

        // Signal entering to ICis always ON(low active), when entering IN0signal, it is Low
        if(!(level == MMC_HIGH_LEVEL))
        {       printf("IN0 signal ON\n");    }
        else if(level == MMC_HIGH_LEVEL)
        {       printf("IN0 signal off\n");     }

        pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_inputstatus_in0, pmc4bpci_inputstatus_in1, pmc4bpci_inputstatus_in2, pmc4bpci_inputstatus_in3, pmc4bpci_inputstatus_exp, pmc4bpci_inputstatus_exm, pmc4bpci_inputstatus_inpos, pmc4bpci_inputstatus_alarm

(6) Reference macro

MMC_HIGH_LEVEL, MMC_LOW_LEVEL, n_IN0, n_IN1, n_IN2, n_IN3, n_EXPP, n_EXPM, n_INPOS, n_ALARM

13 Operation

13.1 pls_move

MMC_INT16U **pmc4bpci_pls_move**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT32 *pls*,MMC_INT16 *vel*,MMC_INT16 *acc*);

(1) Descriptions

It operates linear accel/decel constant speed drive as the specified pulses. It does not wait until drive ends.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- pls: Pulse(range: –268,435,455 to 268,435,455)
- vel: Velocity(range: 1 to 8,000)
- acc: Accel/Decel (range: 1 to 8,000)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
rtn = pmc4bpci_pls_move(MMC_CARD_NO,PMC4BPCI_AXIS_X,10000,
4000, 200);
// ……

if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }
```



```
pmc4bpci_close_all();

}
```

(5) Reference function

pmc4bpci_pls_move, pmc4bpci_pls_move_wait

13.2 pls_move_wait

MMC_INT16U **pmc4bpci_pls_move_wait**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT32 *pls*,MMC_INT16 *vel*,MMC_INT16 *acc*);

(1) Descriptions

It operates linear accel/decel constant speed drive as the specified pulses. It waits until drive ends.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- pls: Pulse(range: –268,435,455 to 268,435,455)
- vel: Velocity(range: 1 to 8,000)
- acc: Accel/Decel (range: 1 to 8,000)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the function ends after the set inner time-out time, it returns MMC_TIMEOUT_ERR. When the result is successful, it returns MMC_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
rtn = pmc4bpci_pls_move_wait
 (MMC_CARD_NO,PMC4BPCI_AXIS_X,10000,4000, 200);
// ……

if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

pmc4bpci_close_all();
```



```
}
```

(5) Reference function

pmc4bpci_pls_move, pmc4bpci_pls_move_wait

13.3 pos_move

MMC_INT16U **pmc4bpci_pos_move**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT32 *pos*,MMC_INT16 *vel*,MMC_INT16 *acc*);

(1) Descriptions

It operates linear accel/decel constant speed drive to the specified position. It does not wait until drive ends.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- pos: Enable to set based on logical/actual position value (range: -2,147,483,648 to 2,147,483,647) within offset range(–268,435,455 to 268,435,455)
- vel: Velocity(range: 1 to 8,000)
- acc: Accel/Decel(range: 1 to 8,000)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
rtn = pmc4bpci_ pos_move
 (MMC_CARD_NO,PMC4BPCI_AXIS_X,10000,4000, 200);
// ……

if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }
```



```
pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_pos_move, pmc4bpci_pos_move_wait

13.4 pos_move_wait

MMC_INT16U **pmc4bpci_pos_move_wait**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT32 *pos*, MMC_INT16 *vel*,MMC_INT16 *acc*);

(1) Descriptions

It operates linear accel/decel constant speed drive to the specified position. It waits until drive ends.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- pos: Enable to set based on logical/actual position value (range: -2,147,483,648 to 2,147,483,647) within offset range (–268,435,455 to 268,435,455)
- vel: Velocity(range: 1 to 8,000)
- acc: Accel/Decel (range: 1 to 8,000)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the function ends after the set inner time-out time, it returns MMC_TIMEOUT_ERR. When the result is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
rtn = pmc4bpci_pos_move_wait
 (MMC_CARD_NO,PMC4BPCI_AXIS_X,10000,4000, 200);
// ……

if(rtn!=MMC_OK)
{printf("Fail!\n");    }
else
{printf("Complete OK!\n");   }
```



```
pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_pos_move, pmc4bpci_pos_move_wait

13.5 cmove

MMC_INT16U **pmc4bpci_cmove**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT16 *vel*);

(1) Descriptions

It sets consecutive drive of the specified axis as the velocity (vel). It stops by pmc4bpci_stop() or pmc4bpci_dstop(). It stops by external stop signal IN0 to 3.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- vel: Velocity(range: 1 to 8,000)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
rtn = pmc4bpci_cmove(MMC_CARD_NO,PMC4BPCI_AXIS_X, 2000);
// ……

if(rtn!=MMC_OK)
{printf("Fail!\n");    }
else
{printf("Complete OK!\n");    }

pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_stop, pmc4bpci_dstop

13.6 pls_smove

MMC_INT16U **pmc4bpci_pls_smove**(MMC_INT16U *id*, MMC_INT16U *axis*, MMC_INT32 *pls*, MMC_INT16 *vel*, MMC_INT16 *acc*, MMC_INT16 *acac,* MMC_INT16 *dec,* MMC_INT16 *dcac*);

(1) Descriptions

It operates S curve drive of the specified axis with the set pulses.
It does not wait until drive ends.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- pls: Pulse(range: –268,435,455 to 268,435,455)
- vel: Velocity(range: 1 to 8,000)
- acc: Accel/Decel (range: 1 to 8,000)
- acac: Jerk speed(range: 1 to 65,535)
- dec: Deceleration(range: 1 to 8,000)
- dcac: Deceleration increment rate (range: 1 to 65,535)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }
// ……
rtn = pmc4bpci_pls_smove(MMC_CARD_NO, PMC4BPCI_AXIS_X, 10000,
4000, 1000, 300, 1000, 300);

// ……
if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }
pmc4bpci_close_all();
```



```
}
```

(5) Reference function

pmc4bpci_pls_smove_wait

13.7 pls_smove_wait

MMC_INT16U **pmc4bpci_pls_smove_wait**(MMC_INT16U *id*, MMC_INT16U *axis*, MMC_INT32 *pls*, MMC_INT16 *vel*, MMC_INT16 *acc*, MMC_INT16 *acac,* MMC_INT16 *dec,* MMC_INT16 *dcac*);

(1) Descriptions

It operates S curve drive of the specified axis with the set pulses.
It waits until drive ends.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- pls: Pulse(range: –268,435,455 to 268,435,455)
- vel: Velocity(range: 1 to 8,000)
- acc: Accel/Decel(range: 1 to 8,000)
- acac: Jerk speed(range: 1 to 65,535)
- dec: Deceleration(range: 1 to 8,000)
- dcac: Deceleration increment rate (range: 1 to 65,535)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }
// ……
rtn = pmc4bpci_pls_smove_wait(MMC_CARD_NO, PMC4BPCI_AXIS_X,
10000, 4000, 1000, 300, 1000, 300);

// ……
if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }
```

```
pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_pls_smove

13.8 pos_smove

MMC_INT16U **pmc4bpci_pos_smove**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT32 *pos*,MMC_INT16 *vel*,MMC_INT16 *acc*, MMC_INT16 *acac,* MMC_INT16 *dec,* MMC_INT16 *dcac*);

(1) Descriptions

It operates S curve drive to the specified position. It does not wait until drive ends.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- pos: Enable to set based on logical/actual position value (range: -2,147,483,648 to 2,147,483,647) within offset range (–268,435,455 to 268,435,455)
- vel: Velocity(range: 1 to 8,000)
- acc: Accel/Decel(range: 1 to 8,000)
- acac:Jerk speed(range: 1 to 65,535)
- dec: Deceleration(range: 1 to 8,000)
- dcac: Deceleration increment rate(range: 1 to 65,535)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }
// ……
rtn = pmc4bpci_pos_smove(MMC_CARD_NO, PMC4BPCI_AXIS_X, 10000,
                        2000, 200, 200, 200, 200);
if(rtn!=MMC_OK)
{printf("Fail!\n");    }
else
{printf("Complete OK!\n");    }
pmc4bpci_close_all();
```



```
}
```

(5) Reference function

pmc4bpci_pos_smove_wait, pmc4bpci_smove_stop

13.9 pos_smove_wait

MMC_INT16U **pmc4bpci_pos_smove_wait**(MMC_INT16U *id*, MMC_INT16U *axis*,MMC_INT32 *pos*,MMC_INT16 *vel*,MMC_INT16 *acc*, MMC_INT16 *acac,* MMC_INT16 *dec,* MMC_INT16 *dcac*);

(1) Descriptions

It operates S curve drive to the specified position. It waits until drive ends.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- pos: Enable to set based on logical/actual position value (range: -2,147,483,648 to 2,147,483,647) within offset range (–268,435,455 to 268,435,455)
- vel: Velocity(range: 1 to 8,000)
- acc: Accel/Decel (range: 1 to 8,000)
- acac: Jerk speed(range: 1 to 65,535)
- dec: Deceleration(range: 1 to 8,000)
- dcac: Deceleration increment rate (range: 1 to 65,535)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the function ends after the set inner time-out time, it returns PMC4BPCI_TIMEOUT_ERR. When the result is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }
// ……
rtn = pmc4bpci_pos_smove_wait(MMC_CARD_NO, PMC4BPCI_AXIS_X,
10000, 2000, 200, 200, 200, 200);
if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }
pmc4bpci_close_all();
```



```
}
```

(5) Reference function

pmc4bpci_pos_smove, pmc4bpci_smove_stop

13.10 pos_imove2

MMC_INT16U **pmc4bpci_pos_imove2**(MMC_INT16U *id*, MMC_INT16U **axis*,MMC_VERTEX **p*)

(1) Descriptions

It operates 2-axis linear interpolation drive. (absoulte position movement)

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. Enters by one axis to interpolation.
  Selects one among INP_AXIS1_X, INP_AXIS1_Y, INP_AXIS1_Z for 1-axis.
  Selects one among INP_AXIS2_X, INP_AXIS2_Y, INP_AXIS2_Z for 2-axis.
- p: Enters target pulses same as the axis order.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void WINAPI isr_sub( void)
{
        // MMC Interrupt
}

void main()
{
        MMC_INT16U stat;

        stat = pmc4bpci_open(MMC_CARD_NO, isr_sub);
        if(stat!=MMC_OK)
        {
                printf("\nERROR : Can't open 'PMC-4B-PCI(ID=%d)' driver\n",
                                                                MMC_CARD_NO);

        // closes open window driver
        pmc4bpci_close_all();
        return;
        }

        // 2-axis linear interpolation drive
        // ex) Linear interpolation by X axis +2000, Z axis -200
        // axis     = [WR5_INP_AXIS1_X, WR5_INP_AXIS2_Y]
        // p        = [2000, -2000]
        {
```



```
        // the specified axis to 1-axis in interpolation, the specified axis to 2-axis in
interpolation
        MMC_INT16U work_plane_axis[2] = {WR5_INP_AXIS1_X,
WR5_INP_AXIS2_Y};
        MMC_VERTEX p = {20000, -20000, };
        MMC_INT16U axis = PMC4BPCI_AXIS1 | PMC4BPCI_AXIS2 |
PMC4BPCI_AXIS3 | PMC4BPCI_AXIS4 ;

        // initialization
        pmc4bpci_set_accoffset(MMC_CARD_NO, axis, 8);// AO = 8(SV at reset = 8)
        pmc4bpci_set_range(MMC_CARD_NO, axis, 8000000); // (magnification = 1       )
        pmc4bpci_set_acac(MMC_CARD_NO, axis,101);//K=101(jerk
speed=619kpps/sec²)
        pmc4bpci_set_acc(MMC_CARD_NO, axis,1000);//
A=1000(acceleration=125kpps/sec)
        pmc4bpci_set_dec(MMC_CARD_NO, axis,1000);//
D=1000(deceleration=125kpps/sec)
        pmc4bpci_set_startv(MMC_CARD_NO, axis,1000);// SV=1000(initial
speed=1000pps)
        pmc4bpci_set_speed(MMC_CARD_NO, axis,4000);//V=4000(drive speed4000PPS)
        pmc4bpci_set_pulse(MMC_CARD_NO, axis,100000);//P=100000(interpolation
end point setting)
        pmc4bpci_set_lpcounter(MMC_CARD_NO, axis,0);// LP= 0(logical/position
counter setting)
        // 2-axis   linear interpolation drive
        pmc4bpci_pos_imove2(MMC_CARD_NO, work_plane_axis, &p);

        // Stops 2000msec until drive stops
        pmc4bpci_wait_timeout(MMC_CARD_NO, axis, 2000);

        printf("\nComplete OK!\n");
        }

        // closes open window driver
        pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_pos_iarc, pmc4bpci_pos_iarca, pmc4bpci_pos_imove3,

(6) Reference macro

INP_AXIS1_X, INP_AXIS1_Y, INP_AXIS1_Z, INP_AXIS2_X, INP_AXIS2_Y, INP_AXIS2_Z, INP_AXIS3_X, INP_AXIS3_Y, INP_AXIS3_Z

13.11 pls_imove2

MMC_INT16U **pmc4bpci_pls_imove2**(MMC_INT16U *id*, MMC_INT16U **axis*,MMC_VERTEX **p*)

(1) Descriptions

It operates 2-axis linear interpolation drive. (relative position movement)

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. Enters by one axis to interpolation.
  Selects one among INP_AXIS1_X, INP_AXIS1_Y, INP_AXIS1_Z for 1-axis.
  Selects one among INP_AXIS2_X, INP_AXIS2_Y, INP_AXIS2_Z for 2-axis.
- p: Enters target pulses same as the axis order.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void WINAPI isr_sub( void)
{
        // MMC Interrupt
}

void main()
{
        MMC_INT16U stat;

        stat = pmc4bpci_open(MMC_CARD_NO, isr_sub);
        if(stat!=MMC_OK)
        {
                printf("\nERROR : Can't open 'PMC-4B-PCI(ID=%d)' driver\n",
                                                                MMC_CARD_NO);

        // closes open window driver
        pmc4bpci_close_all();
        return;
        }

        // 2-axis    linear interpolation drive
        // Ex) Linear interpolation by X axis +2000, Z axis -200
        // axis     = [WR5_INP_AXIS1_X, WR5_INP_AXIS2_Y]
        // p        = [2000, -2000]
        {
```



```
        // the specified axis to 1-axis in interpolation, the specified axis to 2-axis in
interpolation
        MMC_INT16U work_plane_axis[2] = {WR5_INP_AXIS1_X,
WR5_INP_AXIS2_Y};
        MMC_VERTEX p = {20000, -20000, };
        MMC_INT16U axis = PMC4BPCI_AXIS1 | PMC4BPCI_AXIS2 |
PMC4BPCI_AXIS3 | PMC4BPCI_AXIS4 ;

        // initialization
        pmc4bpci_set_accoffset(MMC_CARD_NO, axis, 8);// AO = 8(SV at reset = 8)
        pmc4bpci_set_range(MMC_CARD_NO, axis, 8000000); // (magnification = 1       )
        pmc4bpci_set_acac(MMC_CARD_NO, axis,101);//K=101(jerk
speed=619kpps/sec²)
        pmc4bpci_set_acc(MMC_CARD_NO, axis,1000);//
A=1000(acceleration=125kpps/sec)
        pmc4bpci_set_dec(MMC_CARD_NO, axis,1000);//
D=1000(deceleration=125kpps/sec)
        pmc4bpci_set_startv(MMC_CARD_NO, axis,1000);// SV=1000(initial
speed=1000pps)
        pmc4bpci_set_speed(MMC_CARD_NO, axis,4000);//V=4000(drive speed4000PPS)
        pmc4bpci_set_pulse(MMC_CARD_NO, axis,100000);//P=100000(interpolation
end point setting)
        pmc4bpci_set_lpcounter(MMC_CARD_NO, axis,0);// LP= 0(logical/position
counter setting)
        // 2-axis linear interpolation drive
        pmc4bpci_pls_imove2(MMC_CARD_NO, work_plane_axis, &p);

        // waits 2000msec until drive stops
        pmc4bpci_wait_timeout(MMC_CARD_NO, axis, 2000);

        printf("\nComplete OK!\n");
        }

        // closes open window driver
        pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_pos_iarc, pmc4bpci_pos_iarca, pmc4bpci_pos_imove3,

(6) Reference macro

INP_AXIS1_X, INP_AXIS1_Y, INP_AXIS1_Z, INP_AXIS2_X, INP_AXIS2_Y, INP_AXIS2_Z, INP_AXIS3_X, INP_AXIS3_Y, INP_AXIS3_Z

13.12 pos_imove3

MMC_INT16U **pmc4bpci_pos_imove3**(MMC_INT16U *id*, MMC_INT16U **axis*,MMC_VERTEX **p*);

(1) Descriptions

It operates 3-axis linear interpolation drive. (absoulte position movement)

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. Enters by one axis to interpolation.
  Selects one among INP_AXIS1_X, INP_AXIS1_Y, INP_AXIS1_Z for 1-axis.
  Selects one among INP_AXIS2_X, INP_AXIS2_Y, INP_AXIS2_Z for 2-axis.
  Selects one among INP_AXIS3_X, INP_AXIS3_Y, INP_AXIS3_Z for 3-axis.
- p: Enters target pulses same as the axis order.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../../include/pmc4bpci.h"

#define MMC_CARD_NO     15


void WINAPI isr_sub( void)
{
        // MMC Interrupt
}

void main()
{
        MMC_INT16U stat;

        stat = pmc4bpci_open(MMC_CARD_NO, isr_sub);
        if(stat!=MMC_OK)
        {
printf("\nERROR : Can't open 'PMC-4B-PCI(ID=%d)' driver\n",
                                        MMC_CARD_NO);

                // closes open window driver
                pmc4bpci_close_all();
                return;
        }

        // 3-axis linear interpolation drive
        // ex) linear interpolation by X axis +2000, Y axis -200, Z axis 1000
```



```
        // work_plane_axis          = [WR5_INP_AXIS1_X, WR5_INP_AXIS2_Y,
WR5_INP_AXIS3_Z]
        // p        = [2000, -200, 1000]
        {
                // X axis to 1-axis in interpolation, Y axis to 2-axis in interpolation, Z axis
to 3-axis in interpolation
                MMC_INT16U work_plane_axis[3] = {WR5_INP_AXIS1_X,
                                WR5_INP_AXIS2_Y, WR5_INP_AXIS3_Z};
                MMC_VERTEX p = {20000, -20000, 10000 };
                MMC_INT16U axis = PMC4BPCI_AXIS1 | PMC4BPCI_AXIS2 |
                                        PMC4BPCI_AXIS3 | PMC4BPCI_AXIS4 ;

                // initialization
                pmc4bpci_set_accoffset(MMC_CARD_NO, axis, 8);// AO = 8
(SV at reset = 8)
                pmc4bpci_set_range(MMC_CARD_NO, axis,PMC4BPCI_RANGE);
// R = 8000000      (magnification = 1)
                pmc4bpci_set_acac(MMC_CARD_NO, axis,101);
                                        // K = 101(jerk speed = 619kpps/sec²)
                pmc4bpci_set_acc(MMC_CARD_NO, axis,1000);
                                        // A = 1000(acceleration = 125kpps/sec)
                pmc4bpci_set_dec(MMC_CARD_NO, axis,1000);
// D = 1000(deceleration=125kpps/sec)
                pmc4bpci_set_startv(MMC_CARD_NO, axis,1000);
// SV= 1000         (initial speed= 1000pps)
                pmc4bpci_set_speed(MMC_CARD_NO, axis,4000);
                                        // V = 4000(drive speed = 4000PPS)
                pmc4bpci_set_pulse(MMC_CARD_NO, axis,100000);
// P = 100000       (interpolation end point setting = 100000)
                pmc4bpci_set_lpcounter(MMC_CARD_NO, axis,0);
// LP= 0(logical/position counter setting = 0)

                // 3-axis linear interpolation drive
                pmc4bpci_pos_imove3(MMC_CARD_NO, work_plane_axis, &p);

                // waits until drive stops
                pmc4bpci_wait(MMC_CARD_NO, axis);

                printf("\nComplete OK!\n");
        }

        // closes open window driver
        pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_pos_iarc, pmc4bpci_pos_iarca, pmc4bpci_pos_imove2

(6) Reference macro

INP_AXIS1_X, INP_AXIS1_Y, INP_AXIS1_Z, INP_AXIS2_X, INP_AXIS2_Y, INP_AXIS2_Z, INP_AXIS3_X, INP_AXIS3_Y, INP_AXIS3_Z

13.13 pls_imove3

MMC_INT16U **pmc4bpci_pls_imove3**(MMC_INT16U *id*, MMC_INT16U **axis*,MMC_VERTEX **p*);

(1) Descriptions

It operates 3-axis linear interpolation drive. (relative position moveoment)

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. Enters by one axis to interpolation.
  Selects one among INP_AXIS1_X, INP_AXIS1_Y, INP_AXIS1_Z for 1-axis.
  Selects one among INP_AXIS2_X, INP_AXIS2_Y, INP_AXIS2_Z for 2-axis.
  Selects one among INP_AXIS3_X, INP_AXIS3_Y, INP_AXIS3_Z for 3-axis.
- p: Enters target pulses same as the axis order.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../../include/pmc4bpci.h"

#define MMC_CARD_NO     15


void WINAPI isr_sub( void)
{
        // MMC Interrupt
}

void main()
{
        MMC_INT16U stat;

        stat = pmc4bpci_open(MMC_CARD_NO, isr_sub);
        if(stat!=MMC_OK)
        {
printf("\nERROR : Can't open 'PMC-4B-PCI(ID=%d)' driver\n",
                                        MMC_CARD_NO);

                // closes open window driver
                pmc4bpci_close_all();
                return;
        }

        // 3-axis   linear interpolation drive
        // ex) X axis +2000, Y axis -200, Z axis linear interpolation as 1000
```



```
        // work_plane_axis          = [WR5_INP_AXIS1_X, WR5_INP_AXIS2_Y,
WR5_INP_AXIS3_Z]
        // p        = [2000, -200, 1000]
        {
                // X axis to 1-axis in interpolation, Y axis to 2-axis in interpolation, Z axis
to 3-axis in interpolation
                MMC_INT16U work_plane_axis[3] = {WR5_INP_AXIS1_X,
                                WR5_INP_AXIS2_Y, WR5_INP_AXIS3_Z};
                MMC_VERTEX p = {20000, -20000, 10000 };
                MMC_INT16U axis = PMC4BPCI_AXIS1 | PMC4BPCI_AXIS2 |
                                        PMC4BPCI_AXIS3 | PMC4BPCI_AXIS4 ;

                // initialization
                pmc4bpci_set_accoffset(MMC_CARD_NO, axis, 8);// AO = 8
(SV at reset = 8)
                pmc4bpci_set_range(MMC_CARD_NO, axis,PMC4BPCI_RANGE);
// R = 8000000      (magnification = 1)
                pmc4bpci_set_acac(MMC_CARD_NO, axis,101);
                                        // K = 101(jerk speed = 619kpps/sec²)
                pmc4bpci_set_acc(MMC_CARD_NO, axis,1000);
                                        // A = 1000(acceleration = 125kpps/sec)
                pmc4bpci_set_dec(MMC_CARD_NO, axis,1000);
// D = 1000(deceleration=125kpps/sec)
                pmc4bpci_set_startv(MMC_CARD_NO, axis,1000);
// SV= 1000         (initial speed= 1000pps)
                pmc4bpci_set_speed(MMC_CARD_NO, axis,4000);
                                        // V = 4000(drive speed = 4000PPS)
                pmc4bpci_set_pulse(MMC_CARD_NO, axis,100000);
// P = 100000       (interpolation end point setting = 100000)
                pmc4bpci_set_lpcounter(MMC_CARD_NO, axis,0);
// LP= 0(logical/position counter setting = 0)

                // 3-axis linear interpolation drive
                pmc4bpci_pls_imove3(MMC_CARD_NO, work_plane_axis, &p);

                // waits until drive stops
                pmc4bpci_wait(MMC_CARD_NO, axis);

                printf("\nComplete OK!\n");
        }

        // closes open window driver
        pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_pos_iarc, pmc4bpci_pos_iarca, pmc4bpci_pos_imove2

(6) Reference macro

INP_AXIS1_X, INP_AXIS1_Y, INP_AXIS1_Z, INP_AXIS2_X, INP_AXIS2_Y, INP_AXIS2_Z, INP_AXIS3_X, INP_AXIS3_Y, INP_AXIS3_Z

13.14 pos_iarc

MMC_INT16U **pmc4bpci_pos_iarc**(MMC_INT16U *id*, MMC_INT16U **axis*, MMC_VERTEX **cp*,MMC_VERTEX **ep*, MMC_INT16 *dir*);

(1) Descriptions

It operates 2-axis circular interpolation drive. The selected axis (axis) must be 2-axis. It draws an arc from the current position as arc start to end point(ep) of horizontal position with cp point as arc center point to dir direction. When dir direction is +, it an arc CCW. When it is -, it draws an arc to CW.

(2) Factors

- id: Enters ID number. (range:0 to 15)
- axis: Selects the axis. Enters by one axis to interpolation.
  Selects one among INP_AXIS1_X, INP_AXIS1_Y, INP_AXIS1_Z for 1-axis.
  Selects one among INP_AXIS2_X, INP_AXIS2_Y, INP_AXIS2_Z for 2-axis.
- cp: Sets the center point by relative position based on current position.
- ep: Sets the target point by relative position based on current position.
- dir: Enters rotation direction of the circle. (If dir>0, it is CCW. Otherwise, it returns CW)

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void WINAPI isr_sub( void)
{
        // MMC Interrupt
}

void main()
{
        MMC_INT16U stat;

        stat = pmc4bpci_open(MMC_CARD_NO, isr_sub);
        if(stat!=MMC_OK)
        {
                printf("\nERROR : Can't open 'PMC-4B-PCI(ID=%d)' driver\n",
                                                                MMC_CARD_NO);

                // closes open window driver
                pmc4bpci_close_all();
                return;
        }
```

```
            // 2-axis circular interpolation drive
            // ex) R=20000, X axis center 0, Y axis center -20000, CCW
            // axis     = [X, Y, 0]
            // cp       = [-20000, 0, 0]
            // ep     = [-40000, 0, 0]
            // dir      = 1 (dir≥0 then ccw else cw)
            {
            // the specified axis to 1-axis in interpolation, the specified axis to 2-axis in
interpolation
            MMC_INT16U work_plane_axis[2] = {WR5_INP_AXIS1_X,
WR5_INP_AXIS2_Y};
            MMC_VERTEX cp = {-20000, 0, };
            MMC_VERTEX ep = {-40000, 0, };
            MMC_INT16U axis = PMC4BPCI_AXIS1 | PMC4BPCI_AXIS2 |
PMC4BPCI_AXIS3 |
                                                PMC4BPCI_AXIS4 ;

                        // 2-axis     circular interpolation drive
                        pmc4bpci_pos_iarc(MMC_CARD_NO, work_plane_axis, &cp, &ep, 1);

                        // waits until drive stops
                        pmc4bpci_wait(MMC_CARD_NO, axis);

                        printf("\nComplete OK!\n");
            }

            // closes open window driver
            pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_pos_iarca

(6) Reference macro

INP_AXIS1_X, INP_AXIS1_Y, INP_AXIS1_Z, INP_AXIS2_X, INP_AXIS2_Y, INP_AXIS2_Z, INP_AXIS3_X, INP_AXIS3_Y, INP_AXIS3_Z

13.15 pos_iarca

MMC_INT16U **pmc4bpci_pos_iarca**(MMC_INT16U *id*, MMC_INT16U **axis*, MMC_VERTEX **cp*,MMC_DOUBLE *ang*);

(1) Descriptions

It operates 2-axis circular interpolation drive. The selected axis (axis) must be 2-axis. It rotates as ang based on the current position and angle. When ang value is +(plus), it rotates to CCW. When it is -(minus), it rotates to CW.

The selected axis (axis) must be 2-axis. It draws an arc from the current position as arc start to end point(ep) of horizontal position with cp point as arc center point to dir direction. When dir direction is +, it an arc CCW. When it is -, it draws an arc to CW.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. Enters by one axis to interpolation.
  Selects one among INP_AXIS1_X, INP_AXIS1_Y, INP_AXIS1_Z for 1-axis.
  Selects one among INP_AXIS2_X, INP_AXIS2_Y, INP_AXIS2_Z for 2-axis.
- cp: Sets the center point as relative coordinate based on the current position (0,0).
- ang: Sets rotation angle as relative angle based on the current angle.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../../include/pmc4bpci.h"

#define MMC_CARD_NO     15

void WINAPI isr_sub( void)
{
        // MMC Interrupt
}

void main()
{
        MMC_INT16U stat;

        stat = pmc4bpci_open(MMC_CARD_NO, isr_sub);
        if(stat!=MMC_OK)
        {
                printf("\nERROR : Can't open 'PMC-4B-PCI(ID=%d)' driver\n",
                                                        MMC_CARD_NO);

                // closes open window driver
```

```
                pmc4bpci_close_all();
                return;
        }

        // 2-axis circular interpolation drive
        // ex) R=20000, X axis center 0, Y axis center -20000, CCW
        // axis     = [X, Y, 0]
        // cp       = [-20000, 0, 0]
        {
        // the specified axis to 1-axis in interpolation, the specified axis to 2-axis in
interpolation
        MMC_INT16U work_plane_axis[2] = {WR5_INP_AXIS1_X,
WR5_INP_AXIS2_Y};
        MMC_VERTEX cp = {-20000, 0, };
        MMC_INT16U axis = PMC4BPCI_AXIS1 | PMC4BPCI_AXIS2 |
PMC4BPCI_AXIS3 |
                                                PMC4BPCI_AXIS4 ;

        // 2-axis     circular interpolation drive
        pmc4bpci_pos_iarca(MMC_CARD_NO, work_plane_axis, &cp, 30.0);

        // waits until drive stops
        pmc4bpci_wait(MMC_CARD_NO, axis);

        printf("\nComplete OK!\n");
        }

        // closes open window driver
        pmc4bpci_close_all();
}
```

(5) Reference function

pmc4bpci_pos_iarca

(6) Reference macro

INP_AXIS1_X, INP_AXIS1_Y, INP_AXIS1_Z, INP_AXIS2_X, INP_AXIS2_Y, INP_AXIS2_Z, INP_AXIS3_X, INP_AXIS3_Y, INP_AXIS3_Z

14 Home search

14.1 homesearch_sw

MMC_INT16U **pmc4bpci_homesearch_sw**(MMC_INT16U *id*, MMC_INT16U *axis*, MMC_INT16 *vel*);

(1) Descriptions

It operates home search of the selected axis (searches home based on logical position). It does not use linear interpolation. However, it does not move to the shortest distance because it uses fixed pulse drive.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Selects the axis. PMC-4B-PCI consists of 4-axis.
- vel: Sets home search speed of the specified axis.

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>
#include "../include/pmc4bpci.h"

void main()
{
        #define MMC_CARD_NO     15
MMC_INT16U    rtn;
        int OpenFlag;

        OpenFlag = pmc4bpci_open(MMC_CARD_NO, NULL);
        if(OpenFlag!=MMC_OK)
        {
                printf("Initialize error!\n");
                return;
        }

// ……
rtn = pmc4bpci_homesearch_sw(MMC_CARD_NO,
 PMC4BPCI_AXIS_X|PMC4BPCI_AXIS_Y, 2000);
// ……

if(rtn!=MMC_OK)
{printf("Fail!\n");     }
else
{printf("Complete OK!\n");    }

pmc4bpci_close_all();
}
```


14.2 homesearch

MMC_INT16U **pmc4bpci_homesearch**(MMC_INT16U *id*, MMC_INT16U *axis*, stHomeSearch* *param*);

(1) Descriptions

It operates home search of the selected axis. Software limit should be set as disable and move speed should be faster than home search speed. For more information about home search, refer to the “5.5 Auto home search output of user manual manual”.

(2) Factors

- id: Enters ID number. (range: 0 to 15)
- axis: Select one axis among 4 axes of PMC4BPCI.
- param: Home search setting structure of the specified axis

(3) Return value

When it is out of range for the supported ID at PMC-4B-PCI board, it returns MMC_INVALID_CARD. When it is out of axis range, it returns MMC_INVALID_AXIS. When the setting is successful, it returns PMC4BPCI_OK. Same value is recorded at gError as global error check variable.

(4) Example

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>
#include <conio.h>

#include "../../include/pmc4bpci.h"

#define MMC_CARD_NO     15


void WINAPI isr_sub( void)
{
                        // MMC Interrupt
}

void main()
{
            MMC_INT16U stat;
            MMC_INT32U t = 0L;
            stHomeSearch hs = {
                        0x5D00,             // WR6
                        0x495F,             // WR7
                        0x00000DAC,         // 3500pulse
                        0x0032,             // home-search speed
                        0x07d0              // move speed
            };

            stat = pmc4bpci_open(MMC_CARD_NO, isr_sub);
            if(stat!=MMC_OK)
            {
printf("\nERROR : Can't open 'PMC-4B-PCI(ID=%d)' driver\n", MMC_CARD_NO);

                        // closes open window driver
                        pmc4bpci_close_all();
```



```
                return;
        }
        printf("\nMOVE...\n", MMC_CARD_NO);

        {
                // X, Y axis home search function
                MMC_INT16U axis = PMC4BPCI_AXIS_X | PMC4BPCI_AXIS_Y;
                MMC_INT32 x, y;
                MMC_INT16U xstat, ystat;

                // After setting default, enable to individual setting as belows.
                hs.move_speed = 5000;

                pmc4bpci_homesearch(MMC_CARD_NO, axis, &hs);

                // waits until drive stops
                while(pmc4bpci_inw(MMC_CARD_NO, rr0) & (PMC4BPCI_AXIS_X>>8))
                {
                // Must read status value by one axis
                // Inner counter position
                x = pmc4bpci_get_logicalposition(MMC_CARD_NO,
PMC4BPCI_AXIS_X);
                y = pmc4bpci_get_logicalposition(MMC_CARD_NO,
PMC4BPCI_AXIS_Y);
                xstat = pmc4bpci_rr2(MMC_CARD_NO, PMC4BPCI_AXIS_X);
// home search status value
                ystat = pmc4bpci_rr2(MMC_CARD_NO, PMC4BPCI_AXIS_Y);
// home search status value
                printf("\rStat=0x%04X,0x%04X : Pos=%d,%d    ", xstat, ystat, x, y);
                }

                printf("\nComplete OK!\n");
        }

        // closes open window driver
        pmc4bpci_close_all();
}
```

15 Appendix

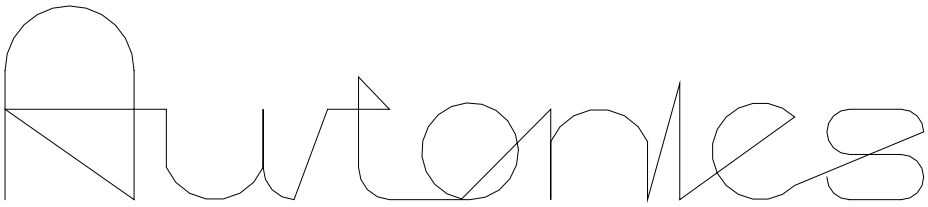
15.1 DXF creation for 'Autonics' logo consecutive interpolation

You can use DXF(Data eXchange Format) type file in MMClibrary of the drawn figure using polyline in AutoCAD.

Refer to the below operation order.

- Operation order

1st Draw Figure1. using polyline command in AutoCAD.



FigureA.1 'Autonics' logo which consists of polyline in AutoCAD

2nd Create DXF type file by using "DXFOUT" command in AutoCAD.

Command : DXFOUT

3rd Edit list A.1 or list A, 2 which are created by 2nd as list A.3 type in text editor. (List A.1 or list A.2 are part of output file.)

List A.1 Part of polyline in AutoCAD 2000

`...	10	0.0	42
2	125.0	42	2.414213562373097
ENTITIES	20	22.90376554843131	10
0	25.0	10	611.9095991350035
LWPOLYLINE	42	353.8812881779981	20
5	0.6666666666666666	20	10.98349570550448
2B	10	0.569709262042181	10
330	200.0	3	711.9095991350035
1F	20	10	20
100	25.0	422.8930948405079	52.5
AcDbEntity	10	20	42
8	200.0	69.99999999999997	0.4142135623730951
0	20	10	10
100	69.99999999999997	422.8930948405079	694.4095991350035
AcDbPolyline	10	20	20
90	200.0	0.0	70.0
38	20	10	10
70	25.0	422.8930948405079	654.4095991350035
0	42	20	20
43	0.4016633628195203	44.99999999999997	70.0
0.0	10	42	42
10	223.9465474202539	-0.666666666666666	0.9999999999999998
0.0	20	10	10
20	0.0	654.4095991350035	20
0.0	10	20	

10 0.0 20 100.0 42 -1.0 10 100.0 20 100.0 10 99.99999999999998 20 0.0 10 0.0 20 70.0 10 99.99999999999998 20 69.99999999999997 10 125.0 20 69.99999999999997	250.0 20 69.99999999999997 10 297.8930948405079 20 69.99999999999997 10 273.9465474202539 20 94.99999999999998 10 273.9465474202539 20 25.0 42 0.4016633628195203 10 297.8930948405079 20 0.0 10 360.3930948405079 20	497.8930948405079 20 44.99999999999997 10 497.8930948405079 20 0.0 10 522.8930948405079 20 89.99999999999998 10 522.8930948405079 20 79.99999999999997 10 522.8930948405079 20 69.99999999999997 10 522.8930948405079 20 0.0 10 611.9095991350035 20 64.01650429449552	35.0 10 694.4095991350035 20 35.0 42 -0.9999999999999998 10 694.4095991350035 20 0.0 10 654.4095991350035 20 0.0 42 -0.4142135623730951 10 636.9095991350035 20 17.5 0 ENDSEC ...

List A.2 Part of polyline in AutoCAD r12

... 0 SECTION 2 ENTITIES 0 POLYLINE 5 2B 8 0 66 1 10 0.0 20 0.0 30 0.0 0 VERTEX 5 93 8 0 10 0.0 20	0.0 0 VERTEX 5 9C 8 0 10 200.0 20 69.999999999999972 30 0.0 0 VERTEX 5 9D 8 0 10 200.0 20 25.0 30 0.0 42 0.4016633628195203 0	5 A6 8 0 10 422.8930948405079 20 69.99999999999997 2 30 0.0 0 VERTEX 5 A7 8 0 10 422.8930948405079 20 0.0 30 0.0 0 VERTEX 5 A8 8	611.90959913500353 20 10.983495705504479 30 0.0 0 VERTEX 5 B1 8 0 10 711.90959913500353 20 52.5 30 0.0 42 0.4142135623730951 0 VERTEX 5 B2 8 0 10 694.40959913500353 20

0.0 30 0.0 0 VERTEX 5 94 8 0 10 0.0 20 100.0 30 0.0 42 -1.0 0 VERTEX 5 95 8 0 10 100.0 20 100.0 30 0.0 0 VERTEX 5 96 8 0 10 99.999999999999986 20 0.0 30 0.0 0 VERTEX 5 97 8 0 10 0.0 20 70.0 30 0.0 0 VERTEX 5 98 8	VERTEX 5 9E 8 0 10 223.94654742025389 20 0.0 30 0.0 0 VERTEX 5 9F 8 0 10 250.0 20 69.999999999999972 30 0.0 0 VERTEX 5 A0 8 0 10 297.8930948405079 20 69.999999999999972 30 0.0 0 VERTEX 5 A1 8 0 10 273.94654742025392 20 94.999999999999986 30 0.0 0 VERTEX 5 A2 8 0 10 273.94654742025392 20 25.0 30	0 10 422.8930948405079 20 44.99999999999997 2 30 0.0 42 - 0.666666666666666 6 0 VERTEX 5 A9 8 0 10 497.8930948405079 20 44.99999999999997 2 30 0.0 0 VERTEX 5 AA 8 0 10 497.8930948405079 20 0.0 30 0.0 0 VERTEX 5 AB 8 0 10 522.8930948405079 6 20 89.99999999999998 6 30 0.0 0 VERTEX 5 AC 8 0 10	70.0 30 0.0 0 VERTEX 5 B3 8 0 10 654.40959913500353 20 70.0 30 0.0 42 0.9999999999999998 0 VERTEX 5 B4 8 0 10 654.40959913500353 20 35.0 30 0.0 0 VERTEX 5 B5 8 0 10 694.40959913500353 20 35.0 30 0.0 42 - 0.9999999999999998 0 VERTEX 5 B6 8 0 10 694.40959913500353 20 0.0 30 0.0 0 VERTEX


```
0
 10
99.999999999999986
 20
69.999999999999972
 30
0.0
  0
VERTEX
  5
99
  8
0
 10
125.0
 20
69.999999999999972
 30
0.0
  0
VERTEX
  5
9A
  8
0
 10
125.0
 20
25.0
 30
0.0
 42
0.6666666666666666
  0
VERTEX
  5
9B
  8
0
 10
200.0
 20
25.0
 30
0.0
 42
0.4016633628195203
  0
VERTEX
  5
A3
  8
0
 10
297.8930948405079
 20
0.0
 30
0.0
  0
VERTEX
  5
A4
  8
0
 10
360.3930948405079
 20
0.0
 30
0.0
 42
22.903765548431309
  0
VERTEX
  5
A5
  8
0
 10
353.8812881779981
 20
0.5697092620421813
 30
0.0
  0
VERTEX
  5
A6
  8
0
 10
522.8930948405079
6
 20
79.99999999999997
2
 30
0.0
  0
VERTEX
  5
AD
  8
0
 10
522.8930948405079
6
 20
69.99999999999997
2
 30
0.0
  0
VERTEX
  5
AE
  8
0
 10
522.8930948405079
6
 20
0.0
 30
0.0
  0
VERTEX
  5
AF
  8
0
 10
611.9095991350035
3
 20
64.01650429449551
9
 30
0.0
 42
2.414213562373097
1
  0
VERTEX
  5
B0
  8
0
 10
654.40959913500353
 20
0.0
 30
0.0
 42
-
0.4142135623730951
  0
VERTEX
  5
B8
  8
0
 10
636.90959913500353
 20
17.5
 30
0.0
  0
SEQEND
...
```

List A.3 Edit as C++ type

```
……
MMC_VERTEX Autonics_p[] = {
        // A
        {0.0, 0.0, 0},
        {0.0,100.0,-1.0},
        {100.0,100.0,0},
        {99.99999999999998,0.0,0},
        {0.0,70.0,0},
        {99.99999999999998,69.99999999999997,0},
        // u
        {125.0,69.99999999999997,0},
        {125.0,25.0,0.6666666666666666},
        {200.0,25.0,0},
        {200.0,69.99999999999997,0},
        {200.0,25.0,0.4016633628195203},
        {223.9465474202539,0.0,0},
        // t
        {250.0,69.99999999999997,0},
        {297.8930948405079,69.99999999999997,0},
        {273.9465474202539,94.99999999999998,0},
        {273.9465474202539,25.0,0.4016633628195203},
        {297.8930948405079,0.0,0},
        // o
        {360.3930948405079,0.0,22.90376554843131},
        {353.8812881779981,0.5697092620421813,0},
        // n
        {422.8930948405079,69.99999999999997,0},
        {422.8930948405079,0.0,0},
        {422.8930948405079,44.99999999999997,-
0.6666666666666666},
        {497.8930948405079,44.99999999999997,0},
        {497.8930948405079,0.0,0},
        // i
        {522.8930948405079,89.99999999999998,0},
        {522.8930948405079,79.99999999999997,0},
        {522.8930948405079,69.99999999999997,0},
        {522.8930948405079,0.0,0},
        // c
```



```
{611.9095991350035,64.01650429449552,2.414213562373097},
        {611.9095991350035,10.98349570550448,0},
        // s
        {711.9095991350035,52.5,0.4142135623730951},
        {694.4095991350035,70.0,0},
        {654.4095991350035,70.0,0.9999999999999998},
        {654.4095991350035,35.0,0},
        {694.4095991350035,35.0,-0.9999999999999998},
        {694.4095991350035,0.0,0},
        {654.4095991350035,0.0,-0.4142135623730951},
        {636.9095991350035,17.5,0},
        // move to origin point
        {0.0, 0.0, 0}
};
  ……
```

For DXF output, be sure that draw continuous one polyline for finding the right order. Drawing order of AutoCAD is according to the drawing order. However, the figure order may be changed by add/edit operation in the middle and above all characters should be connected by the polyline. Otherwise, it cannot find the right character order.

Polyline DXF type

Table A.1 DXF Format

Recognized code	Descriptions
0	Command/Name
100	Command/Name (AutoCAD 2000)
10	Xcoordinate
20	Ycoordinate
30	Zcoordinate
42	Extrude value of arc

Polyline of AutoCAD r12 version and that of AutoCAD 2000 version are different of DXFoutput type. Above list.1 is AutoCAD 2000 version. List.2 is AutoCAD r12 version.

15.2 Macro definition

(1) MMC variable type

- Definition: MMC_INT8
  Original form: typedef signed char MMC_INT8;
  Descriptions: 8bit character type with sign. This definition part is compatible with the created source from WinDriver.
- Definition: MMC_INT8U
  Original form: typedef unsigned char MMC_INT8U;
  Descriptions: 8bit character type without sign. This definition part is compatible with the created source from WinDriver.
- Definition: MMC_INT16
  Original form: typedef signed short int MMC_INT16;
  Descriptions: 16bit whole number type with sign. This definition part is compatible with the created source from WinDriver.
- Definition: MMC_INT16U
  Original form: typedef unsigned long MMC_INT16U;
  Descriptions: 32bit whole number type without sign. This definition part is compatible with the created source from WinDriver.
- Definition: MMC_INT32
  Original form: typedef long MMC_INT32;
  Descriptions: 32bit whole number type with sign.
- Definition: MMC_INT32U
  Original form: typedef unsigned long MMC_INT32U;
  Descriptions: 32bit whole number type without sign.
- Definition: MMC_FLOAT
  Original form: typedef float MMC_FLOAT;
  Descriptions: 4byte real type
- Definition: MMC_DOUBLE
  Original form: typedef double MMC_DOUBLE;
  Descriptions: 8byte real type
- Definition: MMC_VOID
  Original form: typedef void MMC_VOID;
  Descriptions: void type
- Definition: MMC_VERTEX
  Original form: typedef struct {

```
double x;      // X-coordinate
double y;      // Y coordinate
union {
z;   // Z coordinate
k;   // Extrude of arc in polyline
}
} MMC_VERTEX;
```

  Descriptions: It is compatible with Vertex information of polyline.

(2) PMC4BPCI register definition

- Definition: wr0 to wr7, rr0 to rr7
  Descriptions: It has offset value of I/O address for using PMC4BPCI.
- wr0 to wr7(0x00 to 0x0e), rr0 to rr7(0x00 to 0x0e)
- Definition: bp1p, bp1m, bp2p, bp2m, bp3p, bp3m
  Descriptions: It is offset value of I/O address for using PMC4BPCI bit pattern.

(3) PMC-4B-PCI board ID setting

- function: pmc4bpci_set_current_id(MMC_INT16U *id*);
  Descriptions: It changes the to-be-operated board referring the set ID by DIP S/W of PMC-4B-PCI board.

Ex: pmc4bpci_set_current_id(3);

(4) Axis designation

Example: Specify 1-axis, 2-axis at the same time.
PMC4BPCI_AXIS1 | PMC4BPCI_AXIS2
or PMC4BPCI_AXIS1 + PMC4BPCI_AXIS2
or PMC4BPCI_AXIS_X | PMC4BPCI_AXIS_Y
or PMC4BPCI_AXIS_X + PMC4BPCI_AXIS_Y
or 0x0300
pmc4bpci_set_axis(PMC4BPCI_AXIS1 | PMC4BPCI_AXIS2);
Refer to the bit set of the actual mcx314 chip. Refer to the below macro definition.

```
#define   PMC4BPCI_AXIS_NUM     4                   // Number of axes

#define   PMC4BPCI_AXIS1                0x0100
#define   PMC4BPCI_AXIS2                0x0200
#define   PMC4BPCI_AXIS3                0x0400
#define   PMC4BPCI_AXIS4                0x0800
#define   PMC4BPCI_AXIS_X               PMC4BPCI_AXIS1
#define   PMC4BPCI_AXIS_Y               PMC4BPCI_AXIS2
#define   PMC4BPCI_AXIS_Z               PMC4BPCI_AXIS3
#define   PMC4BPCI_AXIS_U               PMC4BPCI_AXIS4
```

(5) Interpolation drive axis designation

Example: MMC_INT16U _ax[2]={INP_AXIS1_X, INP_AXIS2_Y};

```
#define INP_AXIS1_X             WR5_INP_AXIS1_X     // Specified the 1-axis
definition
#define INP_AXIS1_Y             WR5_INP_AXIS1_Y     //
#define INP_AXIS1_Z             WR5_INP_AXIS1_Z     //
#define INP_AXIS1_U             WR5_INP_AXIS1_U     //
#define INP_AXIS2_X             WR5_INP_AXIS2_X     // Specified the 2-axis
definition
#define INP_AXIS2_Y             WR5_INP_AXIS2_Y     //
#define INP_AXIS2_Z             WR5_INP_AXIS2_Z     //
#define INP_AXIS2_U             WR5_INP_AXIS2_U     //
#define INP_AXIS3_X             WR5_INP_AXIS3_X     // Specified the 3-axis
definition
#define INP_AXIS3_Y             WR5_INP_AXIS3_Y     //
#define INP_AXIS3_Z             WR5_INP_AXIS3_Z     //
#define INP_AXIS3_U             WR5_INP_AXIS3_U     //
```

(6) Drive range

Drive range of PMC4BPCI is 8,000,000.

```
#define PMC4BPCI_RANGE          8000000
```

(7) Command code

```
#define PMC4BPCI_CMD_R          0x00    // Range setting
#define PMC4BPCI_CMD_K          0x01    // Jerk speed setting (1 to 65535),2
#define PMC4BPCI_CMD_A          0x02    // Acceleration setting (1 to 8000),2
#define PMC4BPCI_CMD_D          0x03    // Deceleration setting (1 to 8000),2
#define PMC4BPCI_CMD_SV         0x04    // Initial speed setting (1 to 8000),2
#define PMC4BPCI_CMD_V          0x05    // Drive speed setting (1 to 8000),2
#define PMC4BPCI_CMD_P          0x06    // Number of output pulses(0 to
268,435,455),4

                                        // Interpolation end point(-8,388,608 to
8,388,607),4
#define PMC4BPCI_CMD_DP         0x07    // Manual deceleration point setting(0 to 2g),4
#define PMC4BPCI_CMD_C          0x08    // Circular center coordinate setting(-8m to
8m),4
#define PMC4BPCI_CMD_LP         0x09    // Logical position counter setting(-2g to 2g),4
#define PMC4BPCI_CMD_EP         0x0a    // Actual position counter setting (-2g to 2g),4
#define PMC4BPCI_CMD_CP         0x0b    // COMP +register setting (-2g to 2g),4
#define PMC4BPCI_CMD_CM         0x0c    // COMP -register setting (-2g to 2g),4
#define PMC4BPCI_CMD_AO         0x0d    // Accel counter offset setting (0 to 65535),2
#define PMC4BPCI_CMD_NOP        0x0f    // NOP(for replacing axis)
#define PMC4BPCI_CMD_RST        0x8000  // Reset
```

(8) Data read command definition

```
#define PMC4BPCI_READ_LP        0x10    // Logical position counter reading (-2g to
2g),4
#define PMC4BPCI_READ_EP        0x11    // Actual position counter reading (-2g to 2g),4
#define PMC4BPCI_READ_CV        0x12    // Current drive speed reading (1 to 8000),2
#define PMC4BPCI_READ_CA        0x13    // Current pressure speed reading (1 to
8000),2
```

(9) Drive command definition

```
#define PMC4BPCI_DRV_PF         0x20    // +direction constant speed drive
#define PMC4BPCI_DRV_MF         0x21    // -direction constant speed drive
#define PMC4BPCI_DRV_PC         0x22    // +direction consecutive drive
#define PMC4BPCI_DRV_MC         0x23    // -direction consecutive drive
#define PMC4BPCI_DRV_DH         0x24    // drive starts hold
#define PMC4BPCI_DRV_DF         0x25    // Drive starts free/ends status clear
#define PMC4BPCI_DRV_DDS        0x26    // Drive deceleration stop
#define PMC4BPCI_DRV_DS         0x27    // drive immediate stop
```

(10) Interpolation command definition

```
#define PMC4BPCI_INP_2LD        0x30    // 2-axis linear interpolation drive
#define PMC4BPCI_INP_3LD        0x31    // 3-axis linear interpolation drive
#define PMC4BPCI_INP_CW         0x32    // CW circular interpolation drive
#define PMC4BPCI_INP_CCW        0x33    // CCW circular interpolation drive
#define PMC4BPCI_INP_2BP        0x34    // 2-axis bit pattern interpolation drive
#define PMC4BPCI_INP_3BP        0x35    // 3-axis bit pattern interpolation drive
#define PMC4BPCI_INP_BPE        0x36    // BP register writable
#define PMC4BPCI_INP_BPD        0x37    // BP register not writable
#define PMC4BPCI_INP_BPS        0x38    // BP data stack
#define PMC4BPCI_INP_BPC        0x39    // BP data clear
#define PMC4BPCI_INP_SS         0x3a    // interpolation single step
#define PMC4BPCI_INP_DV1        0x3b    // Valid deceleration
#define PMC4BPCI_INP_DV2        0x3c    // Valid deceleration
#define PMC4BPCI_INP_IC         0x3d    // interpolation interrupt clear
```

(11) Write register bit definition

- WR1 (Mode register 1)

```
#define WR1_IN0_L          0x0001  // Logic level for input signal IN0
#define WR1_IN0_E          0x0002  // Valid/Invalid setting for input signal IN0
#define WR1_IN1_L          0x0004
#define WR1_IN1_E          0x0008
#define WR1_IN2_L          0x0010
#define WR1_IN2_E          0x0020
#define WR1_IN3_L          0x0040
#define WR1_IN3_E          0x0080
```

- Interrupt occurance conditions setting bit

```
#define WR1_INT_PULSE      0x0100  // Rising edge of drive pulse
#define WR1_INT_PGECM      0x0200  // Logical/Actual position pulse value ≥
COMP-register
#define WR1_INT_PLCM       0x0400  // Logical/Actual position pulse value <
COMP-register
#define WR1_INT_PLCP       0x0800  // Logical/Actual position pulse value <
COMP+register
#define WR1_INT_PGEP       0x1000  // Logical/Actual position pulse value ≥
COMP+register
#define WR1_INT_CEND       0x2000  // When pulse output ends in constant speed
zone
#define WR1_INT_CSTA       0x4000  // When pulse output starts in constant speed
zone
#define WR1_INT_DEND       0x8000  // When drive ends
```

- WR2 (Mode register 2)

```
// Sets COMP+register as software limit
#define WR2_LMT_SLMTP      0x0001
// Sets COMP-register as software limit
#define WR2_LMT_SLMTM      0x0002
// When hardware limit(nLMTP,nLMTM) is active, sets stop method (0: immediate stop,1:
deceleration stop)
#define WR2_LMT_LMTMD      0x0004
// Logic level setting of +direction limit input signal(nLMTP) (Be active at 0:low,1:high)
#define WR2_LMT_HLMTP      0x0008
// Logic level setting of -direction limit input signal(nLMTM) (Be active at 0:low,1:high)
#define WR2_LMT_HLMTM      0x0010
// Comparision target of COMP+/- register (Compares 0:logical position,1:actual position)
#define WR2_LMT_CMPSL      0x0020
// Output method of drive pulse (0:Individual 2-pulse method,1:1-pulse method)
#define WR2_DRV_PLSMD      0x0040
// Logic level setting of drive pulse (0:Positive logic,1: Negative logic)
#define WR2_DRV_PLS_L 0x0080
// Logic level setting of direction output signal in drive pulse
// (0: low for + direction, high for -direction,1:high for + direction, low for -direction)
#define WR2_DRV_DIR_L 0x0100
// Output method of encoder input signal(nECA/PPIN,nECB/PMIN) (0:2-phase
```

pulse,1:up/dn pulse input)
#define WR2_ENC_PINMD 0x0200
// Sets division ratio of encoder 2-phase pulse input (D1D0 00=1/1, 01=1/2
#define WR2_ENC_PIND0 0x0400
// Sets division ratio of encoder 2-phase pulse input (10=1/4, 11=invalid)
#define WR2_ENC_PIND1 0x0800
// Sets logic level of nALARM input signal (Active at 0: low, 1: high)
#define WR2_SRV_ALM_L 0x1000
// Sets valid/invalid of input signal nALARM for servo motor alarm (0: invalid,1: valid)
#define WR2_SRV_ALM_E 0x2000
// Sets logic level of nINPOS input signal (Active at 0: low, 1: high)
#define WR2_SRV_INP_L 0x4000
// Sets valid/invalid of servo motor inposition input signal nINPOS (0: invalid,1: valid)
#define WR2_SRV_INP_E 0x8000

- WR3 (Mode register 3)
// Deceleration method of accel/decel constant speed drive (0: auto deceleration, 1: manual deceleration)
#define WR3_DRV_MANLD 0x0001
// Effective kinds in accel/decel drive deceleration
// (0: affects to acceleration value, 1: affects to deceleration value – uses only for manual method -)
#define WR3_DRV_DSNSE 0x0002
// Linear accel/decel/S curve setting(0: linear accel/decel, 1: S curve)
#define WR3_DRV_SACC 0x0004
// external input signal (D4D3 00= invalid drive adjustment by external input signal, 01=continuous drive mode
#define WR3_DRV_EXOP0 0x0008
// external input signal 10=fised pulse drive mode, 11=invalid drive adjustment by external input signal
#define WR3_DRV_EXOP1 0x0010
// Displays output signal nOUT4 to 7 as general/drive status (Displays 0: general output, 1: drive status)
#define WR3_OUT_OUTSL 0x0080
// Uses as output signal(0: uses low level, 1: uses high level)
#define WR3_OUT_OUT4 0x0100
#define WR3_OUT_OUT5 0x0200
#define WR3_OUT_OUT6 0x0400
#define WR3_OUT_OUT7 0x0800

- WR4 (output register)
// Sets output of general output signalnOUT0 to 3 (0: low level, 1: high level)
#define WR4_OUT_XOUT0 0x0001
#define WR4_OUT_XOUT1 0x0002
#define WR4_OUT_XOUT2 0x0004
#define WR4_OUT_XOUT3 0x0008
#define WR4_OUT_YOUT0 0x0010
#define WR4_OUT_YOUT1 0x0020
#define WR4_OUT_YOUT2 0x0040
#define WR4_OUT_YOUT3 0x0080


```
#define WR4_OUT_ZOUT0          0x0100
#define WR4_OUT_ZOUT1          0x0200
#define WR4_OUT_ZOUT2          0x0400
#define WR4_OUT_ZOUT3          0x0800
#define WR4_OUT_UOUT0          0x1000
#define WR4_OUT_UOUT1          0x2000
#define WR4_OUT_UOUT2          0x4000
#define WR4_OUT_UOUT3          0x8000
```

- WR5 (interpolation mode register)

```
// Specifies the 1-axis to operate interpolation drive
#define WR5_INP_AX10                    0x0001
// AX10AX11 00:X, 01:Y, 10:Z, 11:U
#define WR5_INP_AX11                    0x0002
// Specifies the 2-axis to operate interpolation drive
#define WR5_INP_AX20                    0x0004
#define WR5_INP_AX21                    0x0008
// Specifies the 3-axis to operate interpolation drive (when using only 2-axis interpolation, uses the some value)
#define WR5_INP_AX30                    0x0010
#define WR5_INP_AX31                    0x0020
// constant linear velocity mode setting of interpolation drive
#define WR5_INP_LSPD0                   0x0100
// D1D0   00: invalid constant linear velocity, 01: 2-axis constant linear velocity, 10: disable to set, 11: 3-axis constant linear velocity
#define WR5_INP_LSPD1                   0x0200
// External send (0: not send, 1: step send interpolation drive to external signal-EXPLSN-)
#define WR5_INP_EXPLS                   0x0800
// (0: not send, 1: step send interpolation drive to command)
#define   WR5_INP_CMPLS                         0x1000
// interrupt occurance enable/disable setting (0: disable, 1: enable) in consecutive interpolation
#define WR5_INP_CIINT                   0x4000
// interrupt occurance enable/disable setting (0: disable, 1: enable) in bit pattern interpolation
#define WR5_INP_BPINT                   0x8000
// Definition of the specified axis to the 1-axis in interpolation
#define WR5_INP_AXIS1_X                         0x0000
#define WR5_INP_AXIS1_Y                         WR5_INP_AX10
#define WR5_INP_AXIS1_Z                         WR5_INP_AX11
#define WR5_INP_AXIS1_U                         (WR5_INP_AX10|WR5_INP_AX11)
// Definition of the specified axis to the the 2-axis in interpolation
#define WR5_INP_AXIS2_X                         0x0000
#define WR5_INP_AXIS2_Y                         WR5_INP_AX20
#define WR5_INP_AXIS2_Z                         WR5_INP_AX21
#define WR5_INP_AXIS2_U                         (WR5_INP_AX20|WR5_INP_AX21)
// Definition of the specified axis to the the 3-axis in interpolation
#define WR5_INP_AXIS3_X                         0x0000
#define WR5_INP_AXIS3_Y                         WR5_INP_AX30
```

```
#define WR5_INP_AXIS3_Z                 WR5_INP_AX31
#define WR5_INP_AXIS3_U                 (WR5_INP_AX30|WR5_INP_AX31)
// invalid interpolation drive mode
#define WR5_INP_LCMODE_INVALIDATE   0x0000
// 2-axis   constant linear velocity
#define WR5_INP_LCMODE_2AXIS        WR5_INP_LSPD0
// 3-axis   constant linear velocity
#define WR5_INP_LCMODE_3AXIS        (WR5_INP_LSPD0|WR5_INP_LSPD1)
```

- WR6 (write data register 1)

```
#define WR6_OUT_WD0         0x0001  // General output register
#define WR6_OUT_WD1         0x0002
#define WR6_OUT_WD2         0x0004
#define WR6_OUT_WD3         0x0008
#define WR6_OUT_WD4         0x0010
#define WR6_OUT_WD5         0x0020
#define WR6_OUT_WD6         0x0040
#define WR6_OUT_WD7         0x0080
#define WR6_OUT_WD8         0x0100
#define WR6_OUT_WD9         0x0200
#define WR6_OUT_WD10        0x0400
#define WR6_OUT_WD11        0x0800
#define WR6_OUT_WD12        0x1000
#define WR6_OUT_WD13        0x2000
#define WR6_OUT_WD14        0x4000
#define WR6_OUT_WD15        0x8000
```

- WR7 (write data register 2)

```
#define WR7_OUT_WD0         0x0001  // General output register
#define WR7_OUT_WD1         0x0002
#define WR7_OUT_WD2         0x0004
#define WR7_OUT_WD3         0x0008
#define WR7_OUT_WD4         0x0010
#define WR7_OUT_WD5         0x0020
#define WR7_OUT_WD6         0x0040
#define WR7_OUT_WD7         0x0080
#define WR7_OUT_WD8         0x0100
#define WR7_OUT_WD9         0x0200
#define WR7_OUT_WD10        0x0400
#define WR7_OUT_WD11        0x0800
#define WR7_OUT_WD12        0x1000
#define WR7_OUT_WD13        0x2000
#define WR7_OUT_WD14        0x4000
#define WR7_OUT_WD15        0x8000
#define WR6_OUT_WD16        WR7_OUT_WD0 // General output register
#define WR6_OUT_WD17        WR7_OUT_WD1
#define WR6_OUT_WD18        WR7_OUT_WD2
#define WR6_OUT_WD19        WR7_OUT_WD3
#define WR6_OUT_WD20        WR7_OUT_WD4
#define WR6_OUT_WD21        WR7_OUT_WD5
#define WR6_OUT_WD22        WR7_OUT_WD6
```



```
#define WR6_OUT_WD23        WR7_OUT_WD7
#define WR6_OUT_WD24        WR7_OUT_WD8
#define WR6_OUT_WD25        WR7_OUT_WD9
#define WR6_OUT_WD26        WR7_OUT_WD10
#define WR6_OUT_WD27        WR7_OUT_WD11
#define WR6_OUT_WD28        WR7_OUT_WD12
#define WR6_OUT_WD29        WR7_OUT_WD13
#define WR6_OUT_WD30        WR7_OUT_WD14
#define WR6_OUT_WD31        WR7_OUT_WD15
```

(12) read register bit Definition

- RR0 (main status register)

```
// Displays status of X axis drive (1: during drive pulse output, 0: during drive end)
#define RR0_X_DRV           0x0001
#define RR0_Y_DRV           0x0002  // Y
#define RR0_Z_DRV           0x0004  // Z
#define RR0_U_DRV           0x0008  // U
// Displays X axis error occurance status (1: sets when any one of error bits for RR1, RR2
register is set.)
#define RR0_X_ERR           0x0010
#define RR0_Y_ERR           0x0020  // Y
#define RR0_Z_ERR           0x0040  // Z
#define RR0_U_ERR           0x0080  // U
// Displays interpolation drive status(1: during interpolation drive pulse output)
#define RR0_I_DRV           0x0100
// (1: writable parameter data/interpolation command for next node in consecutive
interpolation drive)
#define RR0_CNEXT           0x0200
// Displays high limit of current drive in circular interpolation drive
#define RR0_ZONE0           0x0400
// E2E1E0 000:0, 001:1, 010:2, 011:3, 100:4, 101:5, 110:6, 111:7
#define RR0_ZONE1           0x0800
#define RR0_ZONE2           0x1000
// Displays stack counter(SC) in bit pattern interpolation drive
#define RR0_BPSC0           0x2000
// C1C0 00:0, 01:1, 10:2, 11:3
#define RR0_BPSC1           0x4000
```

- RR1 (status register 1)

```
// (1: Logical/Actual position counter≥COMP+ register, 0:else)
#define RR1_CMPP            0x0001
// (1: Logical/Actual position counter<COMP- register, 0:else)
#define RR1_COMM            0x0002
#define RR1_ASND            0x0004  // (1: Acceleration in accel/decel drive)
#define RR1_CNST            0x0008  // (1: Constant speed in accel/decel drive)
#define RR1_DSND            0x0010  // (1: Deceleration in accel/decel drive)
// (1: When acceleration/deceleration increases in S curve drive)
#define RR1_AASND           0x0020
// (1: Acceleration/deceleration is constant in S curve drive)
#define RR1_ACNST           0x0040
// (1: Acceleration/deceleration decreases in S curve drive)
#define RR1_ADSND           0x0080
// (1:When drive stops by external deceleration stop signalnIN0)
#define RR1_IN0             0x0100
#define RR1_IN1             0x0200  // nIN1
```

```
#define RR1_IN2                 0x0400  // nIN2
#define RR1_IN3                 0x0800  // nIN3
// (1:When drive stops by +direction limit signal(nLMTP))
#define RR1_LMTP                0x1000
// (1:drive stops by -direction limit signal(nLMTM))
#define RR1_LMTM                0x2000
// (1: drive stops by alarm signal for servo motor (nALARM))
#define RR1_ALARM               0x4000
// (1: drive stops by emergency stop signal(EMGN))
#define RR1_EMG                         0x8000
```

- RR2 (status register 2)

```
// (1: bigger than COMP+ register value)
#define RR2_SLMTP               0x0001
// (1: bigger than COMP- register)
#define RR2_SLMTM               0x0002
// (1: when +direction limit signal(nLMTP) is active level)
#define RR2_HLMTP               0x0004
// (1: when -direction limit signal(nLMTM) is active level)
#define RR2_HLMTM               0x0008
// (1: alarm signal for servo motor(nALARM) is active level in valid setting)
#define RR2_ALARM               0x0010
// (1: when emergency stop signal(EMGN) is low level)
#define RR2_EMG                         0x0020
```

- RR3 (status register 3)

```
// (1: when drive pulse is rising edge(the moment when pulse is rising at positive logic))
#define RR3_PULSE               0x0001
// (1: Logical/Actual position≥COMP- register)
#define RR3_PGECM               0x0002
// (1: Logical/Actual position<COMP- register)
#define RR3_PLCM                0x0004
// (1: Logical/Actual position<COMP+ register)
#define RR3_PLCP                0x0008
// (1: Logical/Actual position≥COMP+ register)
#define RR3_PGECP               0x0010
// (1: pulse output ends in accel/decel drive constant speed zone)
#define RR3_CEND                0x0020
// (1: pulse output starts to accel/decel drive constant speed zone)
#define RR3_CSTA                0x0040
// (1:drive ends)
#define RR3_DEND                0x0080
```

- RR4,5 (input register 1,2)

```
#define X_IN0                   0x0001  // Displays X axis input status
#define X_IN1                   0x0002  //
#define X_IN2                   0x0004  //
#define X_IN3                   0x0008  //
#define X_EXP                   0x0010  //
#define X_EXM                   0x0020  //
#define X_XINP                  0x0040  //
#define X_XALM                  0x0080  //
#define Y_IN0                   0x0100  // Displays Y axis input status
#define Y_IN1                   0x0200  //
#define Y_IN2                   0x0400  //
#define Y_IN3                   0x0800  //
#define Y_EXP                   0x1000  //
#define Y_EXM                   0x2000  //
#define Y_XINP                  0x4000  //
```



```
#define Y_XALM          0x8000  //
#define Z_IN0           0x0001  // Displays Z axis input status
#define Z_IN1           0x0002  //
#define Z_IN2           0x0004  //
#define Z_IN3           0x0008  //
#define Z_EXP           0x0010  //
#define Z_EXM           0x0020  //
#define Z_XINP          0x0040  //
#define Z_XALM          0x0080  //
#define U_IN0           0x0100  // Displays U axis input status
#define U_IN1           0x0200  //
#define U_IN2           0x0400  //
#define U_IN3           0x0800  //
#define U_EXP           0x1000  //
#define U_EXM           0x2000  //
#define U_XINP          0x4000  //
#define U_XALM          0x8000  //
#define n_IN0           X_IN0   // Displays the axis input status
#define n_IN1           X_IN1   //
#define n_IN2           X_IN2   //
#define n_IN3           X_IN3   //
#define n_EXPP          X_EXP   //
#define n_EXPM          X_EXM   //
#define n_INPOS         X_XINP  //
#define n_ALARM         X_XALM//
```

(13) PMC4BPCI function return macro

```
// Result for fail
#define MMC_FALSE                       0
// Result for success
#define MMC_TRUE                        1
// When function result is successful
#define MMC_OK                          1
// When logic level is high
#define MMC_HIGH_LEVEL                  1
// When logic level is low
#define MMC_LOW_LEVEL                   0
// Fail of MMCDriver calling
#define MMC_OPEN_ERR                    5
// When it is over I/O address range of MMCDriver
#define MMC_IOADDRESS_ERR               6
// When it is over the wating time during MMC driving
#define MMC_TIMEOUT_ERR                 7
// When specifying axis is wrong
#define MMC_INVALID_AXIS                8
// When setting parameter is wrong
#define MMC_ILLEGAL_PARAMETER           9
// When commanding movement which does not move
#define MMC_ZERO_PARAMETER              10
// Error from other causes
#define MMC_ERROR                       11
// When generating end event
#define MMC_QUIT                        12
// CARD irrecognition
#define MMC_INVALID_CARD                13
```

(14) PMC4BPCI library selection

It defines macro function for loading compatible library with standard PMC4BPCI library to application program as below.
The below example is definition for using PMC4BPCI.

```
#define MMC_USING_PMC4BPCI_DRIVER
#include  "pmc4bpci.h"
```

(15) macro for number of axes definition

```
// Including PMC4BPCI.H
#define PMC4BPCI_AXIS_NUM       4

// Including MMC_AX.H
#define AXIS_NUM                        PMC4BPCI_AXIS_NUM
```

(16) NOVA compatibility definition

```
// Base address of PMC-4B-PCI board
// Including PMC4BPCI.H
// It does not apply to inner PMC4BPCI library.
#define    adr                  0x0320

// macro for axis designation
// Including PMC4BPCI.H
#define AXIS_X                  0x01
#define AXIS_Y                  0x02
#define AXIS_Z                  0x04
#define AXIS_U                  0x08
```

Autonics

Sensors & Controllers

www.autonics.com

Distributor

■ **Major products**

•Photoelectric sensors •Fiber optic sensors •Door sensors •Door side sensors •Area sensors
•Proximity sensors •Pressure sensors •Rotary encoders •Connectors/Sockets •Temperature controllers
•Temperature/Humidity transducers •SSR/Power controllers •Counters •Timers •Panel meters
•Tachometer/Pulse(Rate)meters •Display units •Sensor controllers •Switching mode power supplies
•Control switches/Lamps/Buzzers •I/O Terminal Blocks & Cables
•Stepper motors/drivers/motion controllers •Graphic/Logic panels
• Field network devices•Laser marking system(Fiber, CO_2, Nd:YAG) •Laser welding/soldering system

■ **Any proposal for a product improvement and development: Product@autonics.com**

◉ **Headquarters**
116, Ungbigongdan-gil, Yangsan-si, Gyeongsangnam-do, Korea

◉ **Overseas Sales Dept.**
#402-404, Bucheon Techno Park, 655, Pyeongcheon-ro, Wonmi-gu, Bucheon, Gyeonggi-do, Korea
Tel: 82-32-610-2730/ Fax: 82-32-329-0728 / E-mail: sales@autonics.com

■ **Brazil Autonics do Brasil Comercial Importadora Exportadora Ltda**
Tel : 55-11-2307-8480 / Fax: 55-11-2309-7784/ E-mail: vendas@autonics.com.br

■ **China Autonics electronic(Jiaxing) Corporation**
Tel: 86-573-8216-1900 / Fax: 86-573-8216-1917 / E-mail: china@autonics.com

■ **India Autonics Corporation - India Liaison Office**
Tel : 91-22-2781-4305 / Fax : 91-22-2781-4518 / E-mail: india@autonics.com

■ **Indonesia PT. Autonics Indonesia**
Tel: 62-21-658 66 740 / Fax: 62-21-658 66 741 / E-mail: indonesia@autonics.com

■ **Japan Autonics Japan Corporation**
Tel: 81-265-79-8570 / Fax: 81-265-79-2442 / E-mail: support@autonicsjp.co.jp

■ **Malaysia Mal-Autonics Sensor Sdn. Bhd.**
Tel : 60-3-7805-7190(Hunting) / Fax : 60-3-7805-7193 / E-mail: malaysia@autonics.com

■ **Mexico Autonics Mexico Sales Office**
Tel : 52-55-5207-0019 / Fax : 52-55-1663-0712 / E-mail: ventas@autonics.com

■ **Russia Autonics Corp. Russia Representative Office**
Tel/Fax : 7-495-660-10-88 / E-mail : russia@autonics.com

■ **Turkey Autonics Otomasyon Ticaret Ltd. Sti.**
Tel : 90-216-365-9117 / Fax : 90-216-365-9112 / E-mail : info@autonics.com.tr

■ **USA Autonics USA, Inc.**
Tel: 1-847-680-8160 / Fax: 1-847-680-8155 / E-mail: sales@autonicsusa.net

■ **Vietnam Autonics Vietnam Representative Office**
Tel : 84-8-3771-2662 / Fax: 84-8-3771-2663 / E-mail: vietnam@autonics.com

MTC-PMC4BL1-V1.0-1401EN