PMDK PC-Side Basic Function Reference

(Version 1.1)

ICP DAS CO., LTD.

Warranty

All products manufactured by ICPDAS Inc. are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICPDAS Inc. assumes no liability for damages consequent to the use of this product. ICPDAS Inc. reserves the right to change this manual at any time without notice. The information furnished by ICPDAS Inc. is believed to be accurate and reliable. However, no responsibility is assumed by ICPDAS Inc. for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Trademark

The names used for identification only maybe registered trademarks of their respective companies.

License

The user can use, modify and backup this software on a single machine. The user may not reproduce, transfer or distribute this software, or any copy, in whole or in part.

Contents

CHAPTER 1

1. Introduction

This software is dedicated to PMDK control card. It includes Microsoft® Windows 2000, Windows XP and Windows 7 WDM (Windows Driver Model) driver and basic ANSI C application functions.

One unique Card ID is referred by each function in this library. This Card ID is configured by a 4-bit on-board Dip-Switch, and is used for identifying multiple PMDK cards in your system. In other words, users no longer have to worry about the order of PMDK cards detected by the Windows; the only thing users must take care is to make sure each card has a unique Card ID.

ICP DAS provides this library for the customers to develop applications based on Microsoft® Visual Studio (Visual C++).

This document explains how to use the functions in the library. The description includes function declarations, parameters and return codes. Functions are classified and explained in Chapter 2.

CHAPTER 2

2. System functions

2.1 Card ID Configuration

The functions in this chapter provide the interface to Operating System. When PMDK libaray is loaded, it will automatically scan all PMDK cards and create device Handles. Those device Handles and their corresponding Card-IDs, which are configured by on-board Dip-Switch, form a fixed mapping table inside the system. Almost all functions provided here need a device Handle. The pmdk_open() function allows users to get the device Handle before they use other functions.

Note: Make sure that each PMDK card installed in the same PC must have a unique Card ID.

2.1.1 pmdk_open

short pmdk_open(U16 bCardID, HANDLE *pHandle)

Description:

This function opens the device node of PMDK based on the provided Card ID. If this function returns successfully, the process that calls this function owns the device until pmdk_close() is called. Each PMDK has to be opened before be accessed by the other functions.

Parameters:

bCardID：The specific Card ID that is configured with the on-board Dip-Switch.

pHandle：PMDK will use this Handle value as the incoming value of other functions.

Return Code:

Please refer to Appendix A.

2.1.2 pmdk_close

short pmdk_close(HANDLE hCard)

Description:

This function closes the device node of PMDK based on the assigned Handle. After calling this function, the resource of PMDK card will be released and allow other process can open it again.

Parameters:

hCard: The Handle of the specific PMDK when opened.

Return Code:

Please refer to Appendix A.

2.1.3 pmdk_dsp_reset

short pmdk_dsp_reset(HANDLE hCard)

Description:

This function resets the internal DSP and re-run the firmware. After calling this function, all configurations will return to the initial values. This function terminates the current motion, too. When DSP is reset, it requires a little time to reload the program stored in Flash. Users programs on PC can use a while loop and function **pmdk_board_status_get()** to check if PMDK is ready or not. PMDK can accept commands only after it is ready.

Parameters:

hCard: The Handle of the specific PMDK when opened.

Return Code:

Please refer to Appendix A.

2.1.4 pmdk_dip_switch_get

short pmdk_dip_switch_get (HANDLE hCard, U16* pCardID)

Description:

The function can get specific Card ID according to the Handle.

Parameters:

hCard: The Handle of the specific PMDK when opened.

pCardID: A pointer variable pointing to the assigned Card ID.

Return Code:

Please refer to Appendix A.

2.1.5 pmdk_int_event_get

short pmdk_int_event_get(HANDLE hCard, HANDLE* phEvent)

Description:

This function can get the interrupt event number generated by a specified PMDK. An application program can wait for this event then start a specific procedure.

Parameters:

hCard: The Handle of the specific PMDK when opened.

phEvent: A pointer variable points to the event number.

Return Code:

Please refer to Appendix A.

2.1.6 pmdk_int_code_get

short pmdk_int_code_get (HANDLE hCard, I16* pINTCode)

Description:

This function can get the interrupt code which is written on DPRAM by DSP. Users can define different values for different purposes. These codes can be used for communication between PC and PMDK.

Parameters:

hCard: The Handle of the specific PMDK when opened.

pINTCode: A pointer variable points to the interrupt code.

Return Code:

Please refer to Appendix A.

2.1.7 pmdk_board_status_get

short pmdk_board_status_get (HANDLE hCard, U16* pStatus)

Description:

This function can get the READY status of PMDK.

Parameters:

hCard: The Handle of the specific PMDK when opened.

pStatus: A pointer variable points to the status value of PMDK.

Bit	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Definition	**READY**	-	-	-	-	-	-	-
Bit	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Definition	-	-	-	-	-	-	-	-

Return Code:

Please refer to Appendix A.

Example:

```
U16 state, PMDK_ready;
HANDLE hCard
PMDK_ready = 0;
do
{
    pmdk_board_status_get( hCard, &state );
    PMDK_ready = (state & 0x80)>>7;
    Sleep(10);
} while (PMDK_ready == 0);
```

2.1.8 pmdk_cpld_version_read

short pmdk_cpld_version_read (HANDLE hCard, U16* pPLDVersion)

Description:

This function can get the version number of CPLD.

Parameters:

hCard: The Handle of the specific PMDK when opened.

pPLDVersion: A pointer variable points to the CPLD version number of PMDK.

Return Code:

Please refer to Appendix A.

2.1.9 pmdk_cpld_offset_write

short pmdk_cpld_offset_write (HANDLE hCard, U32 dwOffset, U16 wData)

Description:

This function is used to write a 16-bit data to a register of CPLD.

Parameters:

hCard: The Handle of the specific PMDK when opened.

dwOffset: This is the offset address of the assigned CPLD register.

wData: Use this 16-bit data for this write operation.

Return Code:

Please refer to Appendix A.

2.1.10 pmdk_cpld_offset_read

short pmdk_cpld_offset_read (HANDLE hCard, U32 dwOffset, U16* pData)

Description:

This function is used to read a 16-bit data from a register of CPLD.

Parameters:

hCard: The Handle of the specific PMDK when opened.

dwOffset: This is the offset address of the assigned CPLD register.

pData: The pointer variable points to the read 16-bit data.

Return Code:

Please refer to Appendix A.

2.2 DPRAM Read/Write Functions

The size of dual-port Ram (DPRAM) is 0x800 (2048) bytes, and its cell addresses are defined in folllowing table. The basic unit is a WORD (two bytes) when accessing this memory; and the starting address must be an even-number offset address. A block contains 32 bytes (16 words). Sixty-four blocks are defined on this DPRAM as in the following table.

When using following read/write API to access DPRAM, the range of offset addresses can not cover two blocks. For example, the **dwOffset** value of **pmdk_dpram_offset_write_U32()** can not be 0x01E since this action will write 4 bytes and cross the boundary of block 0. On the contrary, the address can be 0x00E since all the memory addresses locate inside block 0 only. When using function **pmdk_dpram_entry_write()** to write data, the starting address is defined to be the starting address of a block (an integral multiple of 0x20), and the length must be an even number and less then or equal to 16 words (0x20 bytes).

Addresses (0x7FC-0x7FD) and (0x7FE - 0x7FF) have special behaviors. When PC writes data to (0x7FE - 0x7FF), it will generate an interrupt to DSP side. Simillarly, When DSP writes data to (0x7FC-0x7FD), it will generate an interrupt to PC side. Users should take advantage of these characteristics to do fast communication between PC and DSP.

Note: The Block restriction only applys on the PC side; DSP-side does not have to consider the block restriction.

Table 1. Blocks and offset addresses definition of DPRAM

Block 0	0x000 - 0x001 (2 bytes)	0x002 - 0x003 (2 bytes)	0x004 - 0x005 (2 bytes)	0x006 - 0x007 (2 bytes)	0x008 - 0x009 (2 bytes)	0x00A - 0x00B (2 bytes)	0x00C - 0x00D (2 bytes)	0x00E - 0x00F (2 bytes)
	0x010 - 0x011 (2 bytes)	0x012 - 0x013 (2 bytes)	0x014 - 0x015 (2 bytes)	0x016 - 0x017 (2 bytes)	0x018 - 0x019 (2 bytes)	0x01A - 0x01B (2 bytes)	0x01C - 0x01D (2 bytes)	0x01E - 0x01F (2 bytes)
Block 1	0x020 - 0x021 (2 bytes)	0x022 - 0x023 (2 bytes)	0x024 - 0x025 (2 bytes)	0x026 - 0x027 (2 bytes)	0x028 - 0x029 (2 bytes)	0x02A - 0x02B (2 bytes)	0x02C - 0x02D (2 bytes)	0x02E - 0x02F (2 bytes)
	0x030 - 0x031 (2 bytes)	0x032 - 0x033 (2 bytes)	0x034 - 0x035 (2 bytes)	0x036 - 0x037 (2 bytes)	0x038 - 0x039 (2 bytes)	0x03A - 0x03B (2 bytes)	0x03C - 0x03D (2 bytes)	0x03E - 0x03F (2 bytes)
...	...							
Block 63	0x7E0 - 0x7E1 (2 bytes)	0x7E2 - 0x7E3 (2 bytes)	0x7E4 - 0x7E5 (2 bytes)	0x7E6 - 0x7E7 (2 bytes)	0x7E8 - 0x7E9 (2 bytes)	0x7EA - 0x7EB (2 bytes)	0x7EC- 0x7ED (2 bytes)	0x7EE - 0x7EF (2 bytes)
	0x7F0 - 0x7F1 (2 bytes)	0x7F2 - 0x7F3 (2 bytes)	0x7F4 - 0x7F5 (2 bytes)	0x7F6 - 0x7F7 (2 bytes)	0x7F8 - 0x7F9 (2 bytes)	0x7FA - 0x7FB (2 bytes)	0x7FC- 0x7FD (2 bytes)	0x7FE - 0x7FF (2 bytes)

2.2.1 pmdk_dpram_offset_write

short pmdk_dpram_offset_write(HANDLE hCard, U32 dwOffset, U16 wData)

Description:

This function is used to write a 16-bit data to the assigned address on DPRAM.

Parameters:

hCard: The Handle of the specific PMDK when opened.

dwOffset: This is the offset address on DPRAM. It must be an even number.

wData: Write this 16-bit data to DPRAM.

Return Code:

Please refer to Appendix A.

2.2.2 pmdk_dpram_offset_read

short pmdk_dpram_offset_read(HANDLE hCard, U32 dwOffset, U16* pData)

Description:

This function is used to read a 16-bit data from DPRAM.

Parameters:

hCard: The Handle of the specific PMDK when opened.

dwOffset: This is the offset address on DPRAM. It must be an even number.

pData: The pointer variable points to the read 16-bit data.

Return Code:

Please refer to Appendix A.

2.2.3 pmdk_dpram_offset_write_U32

short pmdk_dpram_offset_write_U32(HANDLE hCard, U32 dwOffset, U32 dwData)

Description:

This function is used to write a 32-bit data to the assigned address on DPRAM.

Parameters:

hCard: The Handle of the specific PMDK when opened.

dwOffset: This is the offset address on DPRAM. It must be an even number. dwOffset can not be 0x01E since this action will write 4 bytes and cross the boundary of block 0. On the contrary, the address can be 0x00E since all the memory addresses locate inside block 0 only. Please refer to Table 1.

dwData: Write this 32-bit data to DPRAM.

2.2.4 pmdk_dpram_offset_read_U32

short pmdk_dpram_offset_read_U32(HANDLE hCard, U32 dwOffset, U32* pData)

Description:

This function is used to read a 32-bit data from DPRAM.

Parameters:

hCard: The Handle of the specific PMDK when opened.

dwOffset: This is the offset address on DPRAM. It must be an even number. dwOffset can not be 0x01E since this action will write 4 bytes and cross the boundary of block 0. On the contrary, the address can be 0x00E since all the memory addresses locate inside block 0 only. Please refer to Table 1.

pData: The pointer variable points to the read 32-bit data.

Return Code:

Please refer to Appendix A.

2.2.5 pmdk_dpram_entry_write

short pmdk_dpram_entry_write(HANDLE hCard, U32 dwOffset, U16 wDataLength, U16* pData)

Description:

This function can write an array of 16-bit data to DPRAM.

Parameters:

hCard: The Handle of the specific PMDK when opened.

dwOffset: This is the offset address on DPRAM. It must be an integral multiple of 0x20.

wDataLength: This value specifies the length of this array. It must be less than or equal to 16.

pData: The pointer variable points to the first starting address of this array.

Return Code:

Please refer to Appendix A.

2.2.6 pmdk_dpram_entry_read

short pmdk_dpram_entry_read(HANDLE hCard, U32 dwOffset, U16 wDataLength, U16* pData)

Description:

This function is used to read an array of 16-bit data from DPRAM.

Parameters:

hCard: The Handle of the specific PMDK when opened.

dwOffset: This is the offset address on DPRAM. It must be an integral multiple of 0x20.

wDataLength: This value specifies the length of this array. It must be less than or equal to 16.

pData: The pointer variable points to the first starting address of this array.

Return Code:

Please refer to Appendix A.

APPENDIX A

Return Codes (Error Codes)

Definitions of return codes:

PMDK_ERROR_SUCCESS	0
PMDK_ERROR_SCAN_INDEX	-1
PMDK_ERROR_CLOSE_DEVICE	-2
PMDK_ERROR_INVALID_HANDLE	-3
PMDK_ERROR_DEV_IO_CONTROL	-4
PMDK_ERROR_EVENT_CREATE	-5
PMDK_ERROR_INT_EVENT_ATTACH	-6
PMDK_ERROR_INT_EVENT_DETACH	-7
PMDK_ERROR_INVALID_EVENT_HANDLE	-8
PMDK_ERROR_INVALID_DATA_OFFSET	-9
PMDK_ERROR_INVALID_DATA_LENGTH	-10
PMDK_ERROR_DWORD_CROSS_PAGE	-11