PMC-FLASH2-#
PCI-FLASH2-#

User Manual

Preliminary

General Standards Corporation
8302A Whitesburg Drive
Huntsville, AL 35802
Phone: (256) 880-8787
Fax: (256) 880-8788
URL: www.generalstandards.com
E-mail: support@generalstandards.com

PREFACE

General Standards Corporation
Preliminary Revised May 1998.

Additional copies of this manual or other **General Standards Co.** literature may be obtained from:

General Standards Corporation
8302A Whitesburg Dr.
Huntsville, Alabama 35802
Tele: (256) 880-8787
FAX: (256) 880-8788
URL: www.generalstandards.com

The information in this document is subject to change without notice.

General Standards Corp. makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Although extensive editing and reviews are performed before release to ECO control, **General Standards Corp.** assumes no responsibility for any errors that may exist in this document. No commitment is made to update or keep current the information contained in this document.

General Standards Corp. does not assume any liability arising out of the application or use of any product or circuit described herein, nor is any license conveyed under any patent rights or any rights of others.

General Standards Corp. assumes no responsibility for any consequences resulting from omissions or errors in this manual, or from the use of information contained herein.

General Standards Corp. reserves the right to make any changes, without notice, to this product to improve reliability, performance, function, or design.

This user's manual provides information on register level programming of the PMC/PCI-FLAS2-# board.

RELATED PUBLICATIONS

Draft Standard Physical and Environmental Layers for PCI Mezzanine Cards: PMC

Sponsored by the
Bus Architecture Standards Committee
of the IEEE Computer Society

P1386.1/Draft 2.0
April 4,1995
Sponsor Ballot Draft

Questions or comments regarding this draft, please contact either the chair or draft editor of this proposed standard:

Wayne Fisher
PMC Chair
2001 Logic Drive
San Jose, CA 95124-3456 USA
Ph: 408-369-6250
Fx: 408-371-3382
Em:wfisher@fci.com

Dave Moore
PMC Draft Editor
Digital Equipment Corporation
146 Main Street MLO11-4/U32
Maynard, MA 01754-2571 USA
Ph: 408-493-2257
Fax: 408-493-0652
Em: moore@eng.pko.dec.com

PMC/PCI-Flash2-# History

1. Project started 6/4/97. Memory size is not determine yet; however, it is anticipated that it will have 32Mb of memory with .5Mb being flash, and 31.5Mb being battery back-up SRAM. 1st delivery date is to be between Aug. 27, 1997 and Sept. 24, 1997. See Attached for fax on authority to proceed with design. It also contains a list of applicable quality codes.
2. Unanswered questions:
3. Should it allow DMAs to and from board, there can be no DMAs to flash memory.
4. Can this board contain flash only (It is the desire of Lee Gavitt to only use flash, no SRAM. This will allow for a larger amount of memory since a battery is not necessary for flash devices.
5. The minimum speed requirements are as follows:
6. Writes Don't really care as of yet.
7. Read Customer requires a minimum of 8Mb/3 Sec. An attempt will be made to accommodate 10Mb/3 Sec. This breaks down to .3495 Accesses/1us.
8. 10/11/97, Board has been tested and is working fine. It is now ready for release. The memory size implemented is up to 64 Mbytes, in 8 Mbyte increments. The board also contains a small mezzanine to accommodate some custom interface, i.e. simple cable interfaces, more memory, timers, counters, etc.
9. The DMA registers on this board have been tested as far as being accessible; however, DMA transfers have not been tested as of yet. 10/11/97.
10. Updating PMC/PCI-Flash2-# Documentation. (# Is a wildcard for memory Size).
11. The DMAs to and from this board has been test and does appear to work. Worked on some of the grammar, fixed the error that showed 0x15554 and should have showed 0x15554. Since this manual has been shipped prior to these changes, the Rev mark of the manual has been updated to Rev A.
12. 5/11/98: corrected formatting and grammar, updated table of contents, put correct revisions in footer.

TABLE of CONTENTS

Chapter 1: THE PMC/PCI-FLASH2-# MEMORY CARD

1.1. Overview

The FLASH memory card's primary usage is for non-volatile storage. When the system is powered down this card will retain its data. It consists of 512Kbytes of non-volatile static RAM and up to 64Mbytes of FLASH memory. The base address for both sets of memory are located at the same address; therefore, there are 512K bytes of SRAM that are lying on top of the first 512Kbytes of Flash memory. This makes for 512K bytes of available static RAM and up to 63.5Mbytes of available FLASH memory. See Figure 1.1 below.

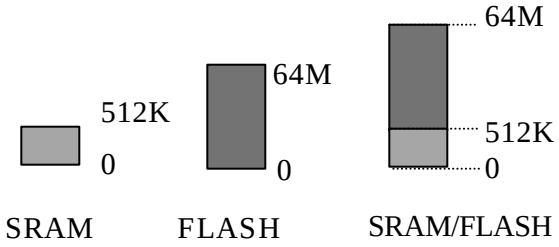

Figure 1.1: General Layout of Memory

1.2. Setup

Setup – This memory card complies with the concept of plug-n-play. Meaning that at the time of system reset, some systems will attempt to set up the card to meet the system's configuration needs. In doing this, the CPU will attempt to map the amount of memory requested by the card and return the base address of where the memory card is located. If the system is not plug-n-play compatible, it will be necessary to provide this information to the board via software.

1.3. Memory Map

Memory Map – The Mapping of this board is relatively simple. In the first 512k bytes is where the SRAM is located and the rest is Flash. They are listed as Bank 0 SRAM and then Bank 0-7 Flash. Bank 0 of the Flash starts at offset 0x800000 and continues up to offset 0x7FFFFF. Each Bank, starting with bank 1, is located in increments of 0x800000. Each bank of flash, except bank 0, is divided into 32 different sectors. Bank 0 is divided into just 30 different sectors. Each flash is also grouped by means of sector groups. Each flash, including bank 0, is divided into 8 sector groups. See Memory Map (Table 3.3, page 13).

1.4. SRAM

The SRAM on this board operates just like any other memory location. This memory can be written to and read from at any time by simply performing a read or write. The specialty of this SRAM is its non-volatility abilities. That is when the board looses power and then power is restored at a later time. The contents of the memory will still be intact. This is accomplished via some loss of power detection circuitry and a ten-year life time battery.

1.5. FLASH

The flash on this board, like the SRAM, is non-volatile. Unlike the SRAM, the flash contains various modes that it operates in. Before accessing the FLASH, it must be placed into the mode of operation desired via an ORDERED sequence of reads and writes to specific addresses with specific data. Ordered is stressed, because some systems will cache up accesses and burst them to the board in an undesired order; hence, the operation will be unsuccessful. This problem can be avoided by performing non-cacheable access or by inserting delays between accesses when performing certain activities. The various modes are as follows:

- Reset/Read
- Auto select
- Program
- Chip Erase
- Sector Erase
- Erase Suspend
- Erase Resume.

Note: Due to the nature of a flash, the ability to erase and write to the flash memory area of this board is limited to a minimum of approximately 99,000 write/erase cycles. After this many erase/write cycles have been performed, the ability to erase/write the flash may no longer function.

Chapter 2: Programming The PMC/PCI-Flash2-#

2.1 Setup

2.1.1 PMC/PCI-FLASH2-# Base Address

This board contains a list of PCI-Configuration registers that conform to the plug-n-play specification of PCI 2.1. These registers are located at various locations, depending on the system. Consult the documentation or contact Technical Support for the system in question for assistance in locating and accessing the PCI-Configuration registers.

2.1.2 Identify the PCI-Configuration Registers for this board

After being able to access the PCI configuration registers on the system it becomes necessary to search the registers to find the set that belongs to this board. Comparing the value at offset 0x00 within these registers can identify the PCI-Configuration registers for this board. Offset 0x00 of the PCI-Configuration registers for this board is read only and will return a value of 0x906010B5. After locating this list of registers, the base address for the SRAM and flash can be found at offset 0x18. This register, hereafter called Local Space Address 0, has read/write capabilities. This is the register that may be setup by the CPU in a plug-n-play compatible system and must be initialized by the software in a non plug-n-play system. The value located within this register is the PCI base address for the SRAM and the flash.

2.2 SRAM - Read/Write

The SRAM on this board starts at Location **Local Address Space 0.** While the 1st byte of SRAM begins at offset 0x0 of the **Local Address Space 0** the last byte is located at offset 0x0007FFFF. To Read or Write to the SRAM, all that must be done is to add the desired offset into the SRAM to the PCI Base Address for **Local Address Space 0** and perform the desired action. There are no special cycles to perform to achieve reads and writes to and from the SRAM and there is no worry about caching or the order of the accesses.

2.3 FLASH – Read/Write/Control

The FLASH card contains up to 64Mbytes of FLASH memory. Like the SRAM, data written to this memory will be retained even after the system has been powered down; however, reading or writing the flash is not as straight forward as the SRAM. There are a few areas of concern when accessing the flash. An explanation and detailed description of each of these are explained below, as well as some example algorithms. There are a few important issues to note before continuing.

- The Flash command code chart below may be used as a reference for keeping track of what is necessary to issue a command for each mode desired.
- In most of the operations for accessing the flash, the order in which the addresses are accessed is important as well as the data written to the flash for each access.
- The offset from **Local Address Space 0** of each bank is as follows (referred to as *flash bank offset*) :

0) Flash Bank 0 0x00080000
1) Flash Bank 1 0x00800000
2) Flash Bank 2 0x01000000
3) Flash Bank 3 0x01800000
4) Flash Bank 4 0x02000000
5) Flash Bank 5 0x02800000
6) Flash Bank 6 0x03000000

7) Flash Bank 7 0x03800000

- Each operation should start and end with a reset/read command
- Note: Once a bit location has been set to '0', it cannot be set to a '1' until after it has been erased.
- There are 4 flash parts per bank, each 1 byte wide, thus creating a 32-bit bank. It is possible to program only 1 flash part in a bank. For example, it is possible to programming only byte 0 of *flash bank offset 3* with byte accesses while ignoring bytes 1-3. This will result in byte 0 of each 32-bit word in bank 3 to have been programmed; however, bytes 1-3 of each 32-bit word in bank 3 will not have been programmed. The same holds true for 16 bit accesses; therefore, with the exceptions going to extreme circumstances, it is suggested – though not require – that 32-bit accesses be used to program the flash at all times to guarantee symmetry across the entire bank.
- The following is an explanation of the examples below:

*(*Flash bank offset* + Some_Addr_In_Bytes) = 0xAAAAAAAA;

This means to write a value of 0xAAAAAAAA to the flash bank offset (See list above) with a desired offset value in bytes. The added offset is compiler dependent. For example, when using a Borland C compiler and an unsigned long pointer as the flash bank offset, then the value listed here in Some_Addr_In_Bytes should be divided by 4. If it were using an unsigned int pointer then it would be divided by 2 but when used as an unsigned char (byte) pointer then it would not be divided at all.

Variable = *(*Flash bank offset* + Some_Addr_In_Bytes);

This means to read from the location flash bank offset (See list above) with an addition offset of Some_Addr_In_Bytes (See explanation of address above).

2.4 FLASH Command Codes

The Flash parts have 8 Command codes and 7 modes of operation. The modes of operation are as follows: Reset/Read, Auto select, Program, Chip Erase, Sector Erase, Erase Suspend, and Erase Resume. Each mode along with the cycles necessary to execute the desired mode is listed in the Command code chart below. This chart should be observed carefully when accessing the flash memory.

2.4.1 Reset/Read Command(1)

```
*Flash bank offset = 0xF0F0F0F0;   /* Reset/Read (1) */

Start Write Loop for N words to Read
        ReadValue = *(Flash bank offset + Offset in bank to Read);  /* Byte Read */
End Loop
```

2.4.2 Reset/Read Command (2)

```
Start Write Loop for N words to Read
        *(Flash bank offset + 0x15554) = 0xAAAAAAAA;  /* Reset/Read (2) */
        *(Flash bank offset + 0x0AAA8) = 0x55555555;   /* 2nd Cycle */
        *(Flash bank offset + 0x15554) = 0xF0F0F0F0;  /* 3rd  Cycle */
        ReadValue = *(Flash bank offset + Offset in bank to Read); /* 4th  Cycle */
End Loop
*Flash bank offset = 0XF0F0F0F0;   /* Reset/Read */
```

2.4.3 Auto-select Command

The Auto-select command is used to obtain the Part ID of the flash parts. There are two values present for each bank: the manufacturer ID (0x01010101) and the device code (0xADADADAD). This ability is used to indicate the presence of the flash parts.

**Flash bank offset* = 0XF0F0F0F0; /* Reset/Read */

*(*Flash bank offset* + 0x15554) = 0xAAAAAAAA; /* Autoselect */
*(*Flash bank offset* + 0x0AAA8) = 0x55555555; /* 2nd Cycle */
*(*Flash bank offset* + 0x15554) = 0x90909090; /* 3rd Cycle */

Man_ID = **Flash bank offset;*
Dev_ID = *(*Flash bank offset* + 0x0004);

**Flash bank offset* = 0XF0F0F0F0; /* Reset/Read */

2.4.4 Byte Program Command

**Flash bank offset* = 0xF0F0F0F0; /* Reset/Read */

Start Write Loop for N words to Program
*(*Flash bank offset* + 0x15554) = 0xAAAAAAAA; /* Byte Program */
*(*Flash bank offset* + 0x0AAA8) = 0x55555555; /* 2nd Cycle */
*(*Flash bank offset* + 0x15554) = 0xA0A0A0A0; /* 3rd Cycle */

*(*Flash bank offset* + Offset into Bank to write) = Data To Write;
End Loop

**Flash bank offset* = 0XF0F0F0F0; /* Reset/Read */

Note: Once a bit location has been set to '0', then it cannot be set to a '1' until after it is erased.

2.4.5 Chip Erase Command

*(*Flash bank offset* + 0x15554) = 0xAAAAAAAA; /* Chip Erase */
*(*Flash bank offset* + 0x0AAA8) = 0x55555555; /* 2nd Cycle */
*(*Flash bank offset* + 0x15554) = 0x80808080; /* 3rd Cycle */
*(*Flash bank offset* + 0x15554) = 0xAAAAAAAA; /* 4th Erase */
*(*Flash bank offset* + 0x0AAA8) = 0x55555555; /* 5th Cycle */
*(*Flash bank offset* + 0x15554) = 0x10101010; /* 6th Cycle */

Start Loop to know when Erase is complete
Do nothing.
End Loop if (**Flash bank offset* = 0xFFFFFFFF) Multiple times; /* Check for Erasure Complete */
Else continue loop;

2.4.6 Sector Erase Command

*(*Flash bank offset* + 0x15554) = 0xAAAAAAAA; /* Chip Erase */
*(*Flash bank offset* + 0x0AAA8) = 0x55555555; /* 2nd Cycle */
*(*Flash bank offset* + 0x15554) = 0x80808080; /* 3rd Cycle */
*(*Flash bank offset* + 0x15554) = 0xAAAAAAAA; /* 4th Erase */
*(*Flash bank offset* + 0x0AAA8) = 0x55555555; /* 5th Cycle */
*(*Flash bank offset* + Sector Address To Erase) = 0x30303030; /* 6th Cycle */

2.4.7 Erase Suspend Command

**Flash bank offset* = 0xB0B0B0B0;

2.4.8 Erase Resume Command

**Flash bank offset* = 0x30303030;

Chapter 3: Tables of Reference

Table 3.1 PCI-Configuration Registers

Device ID		Vendor ID		Offset 0x00
Status		Command		Offset 0x04
Class Code			Revision ID	Offset 0x08
BIST	Header Type	Latency Timer	Cache Line Size	Offset 0x0C
PCI Base Address for Memory Mapped Runtime Registers				Offset 0x10
PCI Base Address for I/O Mapped Runtime Registers				Offset 0x14
PCI Base Address for Local Address Space 0				Offset 0x18
Reserved				Offset 0x1C
Reserved				Offset 0x20
Reserved				Offset 0x24
Reserved				Offset 0x28
Reserved				Offset 0x2C
PCI Base Address to Local Expansion ROM				Offset 0x30
Reserved				Offset 0x34
Reserved				Offset 0x38
Max Latency	Min Grant	Interrupt Pin	Interrupt Line	Offset 0x3C

Table 3.2 PCI Base Address for Local Space 0

Bits	Description	Accessibility	Default
0	Memory Space Indicator	Read	1
2:1	Location of Register: 00 Locate any where in 32 bit Memory Space 01 Locate Below 1Mbyte memory space 10 Locate anywhere in 64 bit memory space 11 Reserved	Read	0x00
3	Prefetchable 1 indicates no side effects on prefetched reads. 0 indicates side effects on prefetched reads	Read	1
31:4	Memory base address for access to local address space	Read/Write	System Dependent

Table 3.3 Memory Map of SRAM and Flash

Bank Name	A25	A24	A23	A22-A19	Range	Size
Bank 7 Flash	1	1	1	XXXX	Offset 0x03800000 – 0x03FFFFFF	8 Mbytes
Bank 6 Flash	1	1	0	XXXX	Offset 0x03000000 – 0x037FFFFF	8 Mbytes
Bank 5 Flash	1	0	1	XXXX	Offset 0x02800000 – 0x02FFFFFF	8 Mbytes
Bank 4 Flash	1	0	0	XXXX	Offset 0x02000000 – 0x027FFFFF	8 Mbytes
Bank 3 Flash	0	1	1	XXXX	Offset 0x01800000 – 0x01FFFFFF	8 Mbytes
Bank 2 Flash	0	1	0	XXXX	Offset 0x01000000 – 0x017FFFFF	8 Mbytes
Bank 1 Flash	0	0	1	XXXX	Offset 0x00800000 – 0x00FFFFFF	8 Mbytes
Bank 0 Flash	0	0	0	0001 – 1111	Offset 0x00080000 – 0x007FFFFF	7.5 Mbytes
Bank 0 SRAM	0	0	0	0000	Offset 0x0 (Local Address Space 0) – 0x0007FFFF	512 Kbytes

Table 3.4 Memory Map of Flash Sector Groups

	Address Bits used to Specify Sector Groups			
Sector Group	A22	A21	A20	Sectors
SGA 0	0	0	0	SA0-SA3 (SA2-SA3 for Bank 0)
SGA 1	0	0	1	SA4-SA7
SGA 2	0	1	0	SA8-SA11
SGA 3	0	1	1	SA12-SA15
SGA 4	1	0	0	SA16-SA19
SGA 5	1	0	1	SA20-SA23
SGA 6	1	1	0	SA24-SA27
SGA 7	1	1	1	SA28-SA31

Table 3.5 Memory Map of Flash Sector Groups

	Address Bits used to Specify Sectors					
Sector	A22	A21	A20	A19	A18	Address Range
SA0	0	0	0	0	0	0x000000 – 0x03FFFF (Not Available on Bank 0)
SA1	0	0	0	0	1	0x040000 – 0x07FFFF (Not Available on Bank 0)
SA2	0	0	0	1	0	0x080000 – 0x0BFFFF
SA3	0	0	0	1	1	0x0C0000 – 0x0FFFFF
SA4	0	0	1	0	0	0x100000 – 0x13FFFF
SA5	0	0	1	0	1	0x140000 – 0x17FFFF
SA6	0	0	1	1	0	0x180000 – 0x1BFFFF
SA7	0	0	1	1	1	0x1C0000 – 0x1FFFFF
SA8	0	1	0	0	0	0x200000 – 0x23FFFF
SA9	0	1	0	0	1	0x240000 – 0x27FFFF
SA10	0	1	0	1	0	0x280000 – 0x2BFFFF
SA11	0	1	0	1	1	0x2C0000 – 0x2FFFFF
SA12	0	1	1	0	0	0x300000 – 0x33FFFF
SA13	0	1	1	0	1	0x340000 – 0x37FFFF
SA14	0	1	1	1	0	0x380000 – 0x3BFFFF
SA15	0	1	1	1	1	0x3C0000 – 0x3FFFFF
SA16	1	0	0	0	0	0x400000 – 0x43FFFF
SA17	1	0	0	0	1	0x440000 – 0x47FFFF
SA18	1	0	0	1	0	0x480000 – 0x4BFFFF
SA19	1	0	0	1	1	0x4C0000 – 0x4FFFFF
SA20	1	0	1	0	0	0x500000 – 0x53FFFF
SA21	1	0	1	0	1	0x540000 – 0x57FFFF
SA22	1	0	1	1	0	0x580000 – 0x5BFFFF
SA23	1	0	1	1	1	0x5C0000 – 0x5FFFFF
SA24	1	1	0	0	0	0x600000 – 0x63FFFF
SA25	1	1	0	0	1	0x640000 – 0x67FFFF
SA26	1	1	0	1	0	0x680000 – 0x6BFFFF
SA27	1	1	0	1	1	0x6C0000 – 0x6FFFFF
SA28	1	1	1	0	0	0x700000 – 0x73FFFF
SA29	1	1	1	0	1	0x740000 – 0x77FFFF
SA30	1	1	1	1	0	0x780000 – 0x7BFFFF
SA31	1	1	1	1	1	0x7C0000 – 0x7FFFFF

Table 3.6 Flash Command Codes

Command	First Bus Write Cycle		Second Bus Write Cycle		Third Bus Write Cycle		Fourth Bus Read/Write Cycle		Fifth Bus Write Cycle		Sixth Bus Write Cycle	
	Addr	Data In Byte to Program	Addr	Data In Byte to Program	Addr	Data In Byte to Program	Addr	Data In Byte to Program	Addr	Data In Byte to Program	Addr	Data In Byte to Program
Reset/Read	XXXXH	0xF0	---	---	---	---	---	---	---	---	---	---
Reset/Read	0x15554	0xAA	0x0AAA8	0x55	0x15554	0xF0	RA	RD	---	---	---	---
Auto Select	0x15554	0xAA	0x0AAA8	0x55	0x15554	0x90	---	---	---	---	---	---
Byte Program	0x15554	0xAA	0x0AAA8	0x55	0x15554	0xA0	PA	Data	---	---	---	---
Chip Erase	0x15554	0xAA	0x0AAA8	0x55	0x15554	0x80	0x15554	0xAA	0x0AAA8	0x55	0x15554	0x10
Sector Erase	0x15554	0xAA	0x0AAA8	0x55	0x15554	0x80	0x15554	0xAA	0x0AAA8	0x55	SA	0x30
Erase Suspend	XXXXH	0xB0	---	---	---	---	---	---	---	---	---	---
Erase Resume	XXXXH	0x30	---	---	---	---	---	---	---	---	---	---

Notes: 1. RA = Address of the memory location to be read
PA = Address of the memory location to be programmed.
SA = Address of the sector to be erased. The combination of A22, A21, A20, A19, and A18 will uniquely select any sector.

2. RD = Data read from location RA during read operation.
PD = Data to be programmed at location PA.

3. Read and Write program functions to non-erasing sectors are allowed in the Erase Suspend mode.