PRM-Q

PROGRAMMING INFORMATION

The PRM-Q is a "Q-Bus Plus" DMA interface for the Priam Winchester disk drives used with the Priam "Smart Interface". It provides a direct, easy-to-program connection between the Q-Bus and the Smart Interface. The PRM-Q can operate one Smart Interface which can control four drives.

The PRM-Q provides direct programmed I/O access to registers in the disk controller, it handles multi-sector DMA transfers through the full 22-bit address range of the Q-Bus Plus, allows for partial sector transfers and handles interrupts for byte-count-zero, bus error and disk controller attention conditions. In addition, the PRM-Q contains an on-board PROM for easy bootstrapping.

PRIAM SMART INTERFACE DISK CONTROLLER

The PRIAM "Smart Interface" contains eight read-only registers and eight write-only registers. The PRM-Q accesses the registers as two groups of four read-only/write-only register pairs. A select bit in the DMA control/status register determines which group of four is to be accessed. The registers consist of a command/status register (CSR), a data buffer (DBF), six parameter registers and six result registers. The protocol required by the controller to cause it to perform an operation is as follows:

- Load the parameter registers
- Load the command into the CSR
- Wait for attention flag
- Read status from CSR and result register 0
- Send completion acknowledge command to CSR

PRIAM SMART INTERFACE DISK CONTROLLER-continued

The transfer must be completed within four seconds of the first data request or the controller will detect a transfer timeout error.

The disk controller can transfer multiple sectors as a result of one command. A maximum of approximately 64KB (127 sectors) can be transferred this way. The disk controller always transfers data in multiples of whole sectors. It cannot transfer partial sectors.

PRM-Q REGISTERS

The PRM-Q responds to eight register (full word) addresses in the I/O page. Four of these registers correspond to the disk controller registers. One register is not used. The other three registers provide control over the DMA and interrupt functions of the PRM-Q interface, and select which group of four disk controller registers is to be accessed. The registers are:

-Smart Interface Registers (Read Only)-

Relative Address	Low Group (SEL = 0)	High Group (SEL = 1)
0	Result 5	Result 1
2	Result 4	Result 0
4	Result 3	Disk Data
6	Result 2	Controller Status

-Smart Interface Registers (Write Only)-

Relative Address	Low Group (SEL = 0)	High Group (SEL = 1)
0	Parameter 5	Parameter 1
2	Parameter 4	Parameter 0
4	Parameter 3	Disk Data
6	Parameter 2	Controller Command

PRM-Q REGISTERS-continued

-PRM-Q Interface Registers-

Relative Address	Function
10	Not Used
12	Interface Control/Status (ICS)
14	Interface Memory Address (IMA)
16	Interface Byte Count (IBC)

The ICS register contains the following control functions:

Control Functions

- Enable external interrupt (disk controller attention)
- Enable internal interrupt (byte-count-zero or bus error)
- DMA transfer direction
- Disable IMA incrementing
- Reset disk controller
- Select disk controller register group
- High six bits of the 22-bit memory address

The "Disable IMA incrementing" control bit is used to allow partial sector transfers into program memory while accommodating the disk controller's requirement for full sector transfers. For example, when reading a partial sector the memory address is set to point into the desired data transfer area, the byte count register is set for the desired number of bytes, internal interrupt is enabled and the read operation in initiated. When byte-count-zero occurs the memory address is reset to point to an unused byte in memory, the "Disable IMA incrementing" bit is set and IBC is cleared. The bytes remaining in the sector are then all transferred into the unused byte. A similar operation can be followed to write partial sectors with zero fill.

PRM-Q REGISTERS-continued

The disk controller reset bit must be cleared or the disk controller will be held in a reset condition. The disk controller register select bit (SEL) determines which of two register groups is currently selected for DMA and programmed I/O transfers. The high six bits of memory address act as the high order extension of the 16-bit IMA register. When the IMA is incremented from 177777 to 0, the carry out causes this six-bit extension to be incremented.

The ICS register provides the following status information:

Status Bits

- External interrupt request flag
- Internal interrupt request flag
- Controller ready
- Bus error

The IMA register provides the low order 16-bits of the 22-bit memory address. the IMA is incremented before the first transfer so it must be loaded initially with one less than the memory address of the first data byte to be transferred. Note that this subtraction operation must be applied to the full 22-bit address value. For example, to transfer into location 0 of memory, the IMA must be loaded with 177777 and the high six bits in the ICS must be loaded 77.

The IBC register allows for transfers of up to 65KB of data. It is an up-counter and it stops counting when it reaches the value 177777. This is the byte-count-zero condition which asserts the internal interrupt request. The IBC must be loaded initially with the one's-complement of the number of bytes to be transferred.

PRIAM HANDLER

USER'S MANUAL

RT11 V4.Ø Device Handler

for the

PRM-Q

by

PERITEK CORPORATION

Peritek Corporation
5550 Redwood Road
Oakland, California 94619
(415) 531-6500
TWX: 910-366-2029

Version: 1.Ø
Date: 31-MAR-81

PRIAM RT11 HANDLER

TABLE OF CONTENTS

Chapter 1
INTRODUCTION

I.1 Handler Features

The Priam Winchester handler is a device handler for RT11 which works with any of the Priam Winchester disk drives. The handler can be easily adapted to work with the various drives which are available now and which will become available in the future.

The handler utilizes the full storage capacity of the drives. Defect mapping (if used) is transparent to the handler. A disk formatting program is provided which interleaves logical sectors to maximize block transfer rate on consecutive block transfers.

I.2 Hardware Environment

The Priam handler works with the PRM-Q interface manufactured by Peritek Corporation. The PRM-Q is connected to Priam's "Smart Interface" disk controller. Up to four drives may be connected to the disk controller.

The interrupt service routine runs at interrupt level 5. Therefore, if the PRM-Q is used with an LSI-11/23 CPU which has multi-level interrupts, the interrupt request level of the PRM-Q should be changed to BIRQ5. (See PRM-Q User's Manual for details). This is not a mandatory change.

I.3 Software Environment

This handler works with RT11 Version 4. It will work with the SJ, FB and XM monitors. The handler performs exactly like a standard RT11 handler and its use is completely transparent to the applications programmer.

The sysgen options supported by the Priam handler are memory management and error logging.

The device name used is 'WP' and the device code is octal 370.

I.4 Logical Unit Configuration

Each physical drive can be treated as a single logical unit or it can be divided into a number of smaller logical units. RT11 allows a device handler to have up to eight logical units. Each logical unit is limited to 32MB in size.

Chapter II
HANDLER DISTRIBUTION KIT

II.1 Summary

The Priam handler is distributed on RXØ1 compatible single sided, single density diskette. Source files for the handler are included so the user can generate his own handler if desired.

II.2 Contents of Diskette

The distribution package contains the following files:

WP.MAC	-	Handler source file
WPCND.MAC	-	Handler conditional file
WP.COM	-	Command file used to generate handler
WP.OBJ	-	Handler object file
WP.LST	-	Handler listing file
WP.SYG	-	Executable handler
WPFMT.SAV	-	Disk formatting program
ERRTXT.MAC	-	Register names for error log report writer
WPDWN.SAV	-	Turns off power to all drives
WPDWN.MAC	-	Source

II.3 Restrictions

The executable handler contained in the distribution kit will work with the standard monitors (RT11SJ, RT11FB, RT11BL) as provided in the V4 release kit. It will not work with user gen'd systems with any of the following options enabled: extended memory, error logging, timer support. The executable handler will only work with hardware (interface and drives) as specified in the handler conditional file. List the handler conditional file to see if it matches your hardware configuration.

If the executable handler is not compatible with either your operating system or your hardware then a new handler will have to be generated. See Chapter IV for instructions on how to do this.

II.4 Getting On-Line

There are several steps which must be taken to get the Priam handler up and running as the system device.

a) Format the disk drive(s) in 512 byte format using the program WPFMT. This is described in Section III.3.

b) Generate the appropriate handler to suit the hardware/software environment and the desired logical unit configuration. This is described in Chapter IV.

c) Copy the generated handler to the system device on the existing system. Use the commands:

```
.COPY/SYS  WP.SYG  SY:WP.SYS
.INSTALL   WP
```

It is now possible to use the target disk as a non-system device on the existing system. Initialize the directories on the target disk and verify that files can be written to and read back from the drive.

d) Copy the system files and other important files to the target disk. Be sure to include the SWAP file, monitors, handlers, utility programs, etc.

e) Execute the boot copy operation as follows:

```
.COPY/BOOT  WP:RT11xx  WP:
```

Where xx represents SJ, FB, or XM. The drive is now a bootable system device.

f) Bring up the new system by executing a soft boot using the monitor command:

```
.BOOT       WP:
```

g) Alternately, bring up the system by executing the hardware boot. In micro-ODT type:

```
@777000G
```

If the address of the PRM-Q card is non-standard be sure to use the correct address.

Note that a soft boot can be performed on any logical unit but the hardware boot always uses drive 0, logical unit 0.

II.5 Powering Down

The program WPDWN.SAV outputs the command "Sequence Power Down" to all drives and then halts. It is advisable to execute this program before turning off power to the Smart Interface and drives. The program source is included in case the user wants to eliminate the halt or change the controller address.

Chapter III
DRIVE PREPARATION

III.1 Physical Connection

The PRM-Q User's Manual provides information on how to option the PRM-Q, how to install it, connect it to the Smart Interface, and how to verify that it is working properly.

III.2 Drive Options

The important drive options to select are the drive number and number of sectors per track. The Priam handler addresses drives in numerical order. If only one drive is connected it must respond to drive address Ø. If two drives are being used they must respond to drive addresses Ø and 1, and so forth.

The drives must be set up to use 512 byte sectors. Choose the appropriate number of sectors per track to give 512 byte sectors.

Write protect must be disabled.

III.3 Drive Formatting

The drives, as shipped by Priam, may not be in the proper 512 byte format for use by RT11. A program included on the diskette handles formatting for any of the Priam drives. It does not use defect mapping.

There are three purposes for formatting a disk. They are:

1. Set up the disk with 512 byte sectors for use by RT11.
2. Interleave logical sectors for maximum data transfer rate on multi-block transfers.
3. Flag all sectors containing media defects so they cannot be read or written by RT11.

III.3a Operating Procedures

The formatting program is WPFMT.SAV. It is an interactive program which requires the user to specify the controller address, drive number and format desired. The dialogue is shown below with sample answers for a standard PRM-Q and a 335Ø drive:

Priam Formatting Program (31-MAR-81)

Enter. . . .
Controller address (octal) = 177100
Drive number to be formatted = 0
Starting cylinder = 0
Ending cylinder = 560
Total bytes per sector = ~~574~~ 560
Interleave factor = 4
Logical units per drive = 4

All values entered are decimal numbers except for the controller address which is octal. Starting and ending cylinders specify the cylinder range to be formatted. Suggested values to use are shown in the table below:

Drive Model	Maximum Cylinder	Interleave Factor 11/2	Interleave Factor 11/23	Total Bytes/Sector
1070	189	4	3	648
2050	524	4	3	582
3450	524	4	3	582
3350	560	5	4	~~574~~ 560
6650	1120	5	4	~~574~~ 560
15450	1120	5	4	~~574~~ 560

The total bytes per sector parameter indicates the physical size of a sector including header, filler, data, CRC and gap. This number is used in calculating the sector location of a media defect.

Sectors are interleaved by pairs (the Smart Interface always tries to transfer two 512 byte sectors at a time on multiblock transfers). An interleave factor of zero results in consecutive sectors while a factor of one results in logical consecutive sector pairs which are separated by one physical sector pair. Note that a smaller interleave factor can be used with the LSI-11/23 processor since DMA latency is less.

The logical units per drive specified should correspond to the number of logical units per drive which will be used by the RT11 handler (see Section IV.1). This value is used to determine the location of defective sectors in terms of RT11 logical unit and block numbers.

Once the parameters have been entered the program then reads the drive type and drive parameters from the Smart Interface and displays them on the console device as follows:

Drive Information

Type	Model	Heads	Cyls	S/Trk	S/Drive	B/S
1	3350	3	561	34	57222	512

If an error occurs when reading the drive type or parameters or if the sector size is not 512 bytes, then an error message is printed and the program aborts.

If the drive type and parameters are read successfully and if the sector size is 512 bytes then the program will request the name of a file or device to receive information on the formatting process, including the drive information and the logical-to-physical sector map.

Output file = LP:

If the null string (return only) is entered then no output is generated. If a device or file name is entered, the output is directed to that file or device.

The Program then requests:

Type 'GO' to proceed:

When the user types GO (followed by return) the program begins execution. While formatting, the program produces no output to the console except on error conditions. Formatting can take a long time (up to 25 minutes on large drives) so rest assured that the program is working and wait for it to finish.

III.3b Formatting Process

The program formats the disk in five steps, executing all five operations on a given track before proceeding to the next track. Formatting begins on the starting cylinder specified and continues to the last cylinder on the drive.

The operations used in formatting the drive are as follows:

a) Format Disk
   Writes headers for 512 byte sectors.

b) Write Data
   Writes 512 bytes of data to every sector on the track.

c) Read Skip Defect Record
   Determines if there are any defects on the track.

d) Interleave Logical Sectors
   Rewrites the headers in interleaved order.

e) Verify
   Reads back the entire disk and lists any sectors with faulty CRC's on ID or data.

If an error is encountered the program types out a short character string identifying the operation in progress, followed by seven bytes in octal. These values represent the status register, the transaction status (result 0), and result registers 1 through 5. The failing operation is repeated. If three errors occur on the same track, another error message is printed out specifying the head and cylinder number in decimal, and the program proceeds to the next track.

If defects are encountered on a track the verify operation is bypassed since it will always result in error messages.

III.3c Program Output

The output which goes to the user specified device or file includes the input parameters, the drive information, and a sector map which shows the physical to logical sector correspondence. If the number of sectors per track is even the map is the same for all tracks on the device. If the number of sectors per track is odd, the map is different for each track on a cylinder but all cylinders are exactly the same. A sample sector map for a 3350 drive with 34 sectors per track and interleave factor = 4 is shown below:

Priam Formatting Program (31-MAR-81)

Controller address = 177100
Drive number = 0
Starting cylinder = 0
Ending cylinder = 560
Total bytes per sector = ~~574~~ 560
Interleave factor = 4
Logical units per drive = 4

Drive Information

Type	Model	Heads	Cyls	S/Trk	S/Drive	B/S
1	3350	3	561	34	57222	512

Sector Map

PHYSICAL SECTOR

Head	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	1	14	15	28	29	8	9	22	23	2	3	16	17	30	31	10	11	24	25
1	0	1	14	15	28	29	8	9	22	23	2	3	16	17	30	31	10	11	24	25
2	0	1	14	15	28	29	8	9	22	23	2	3	16	17	30	31	10	11	24	25

PHYSICAL SECTOR

Head	20	21	22	23	24	25	26	27	28	29	30	31	32	33
0	4	5	18	19	32	33	12	13	26	27	6	7	20	21
1	4	5	18	19	32	33	12	13	26	27	6	7	20	21
2	4	5	18	19	32	33	12	13	26	27	6	7	20	21

III.4 Media Defects

WPFMT does not use defect mapping because it is inefficient and it reduces the total number of cylinders available for use. RT11's bad block capability is a much more satisfactory method to take care of defects. When the skip defect record indicates media defects on a given track, the formatting program generates error messages to inform the user where the defects are. These error messages take the form:

Bad sector at unit 1 block 6933.

or:

Bad sectors at unit Ø blocks 1Ø2. to 135.

The formatting program writes invalid sector ID's on sectors containing defects so it is impossible for RT11 to read or write unreliable sectors. This ensures that errors will occur if a program ever tries to write into a defective area, not just at some future time when a program attempts to read back the data.

The user should create ".BAD" files at the locations specified as bad sectors so no RT11 program will ever attempt to use those areas.

Chapter IV
GENERATING THE HANDLER

IV.1 Handler Conditionals

The WPCND.MAC file defines several parameters which affect the assembly of the handler. By changing these parameter values the user can customize the handler for his particular application. The standard conditional file is shown below:

```
;       WPCND.MAC                  18-FEB-81
;       PRIAM HANDLER CONDITIONAL ASSEMBLY OPTIONS (RT11 V4.0)
;       FOR THE PERITEK CORPORATION PRM-Q

;PRM-Q INTERFACE DEFINITION
WPCSR= 177100               ;STANDARD DEVICE BASE ADDRESS
WPVEC= 374                  ;STANDARD INTERRUPT VECTOR

;DISK GEOMETRY DEFINITIONS
WPHEAD= 3                   ;NUMBER OF HEADS
WPCYLS= 561.                ;NUMBER OF CYLINDERS TO USE
WPSPT= 34.                  ;NUMBER OF SECTORS PER TRACK

;LOGICAL UNIT CONFIGURATION
NDRVS=  2                   ;NUMBER OF DRIVES ON THIS CONTROLLER
NLUN=   4                   ;NUMBER OF LOGICAL UNITS PER DRIVE
                ;NOTE: NDRVS*NLUN MUST BE LESS THAN OR EQUAL TO 8

;HANDLER OPTIONS
...EIS= 0                   ;EIS AVAILABLE? (=1 IF EIS IS AVAILABLE)
```

The PRM-Q interface definitions do not have to be changed unless the user has changed the jumper selectable addresses on the PRM-Q board. If the addresses have been changed, be sure that the definitions correspond to the actual addresses on the board.

The disk geometry definitions depend on the type of drive being used. Sample values to use for different drive models are:

Drive:	1070	3450	3350A	3350B	6650
Cylinders:	190	525	561	561	1121
Heads:	4	5	3	3	3
Sectors:	22	23	34	35	35

If the user implements defect mapping on his drives, the number of cylinders to use will have to be reduced.

The logical unit configuration must be determined by the user, based on his requirements. The number of drives can range from 1 to 4. The number of logical units per drive can range from 1 to 8. However, the total number of logical units (number of logical units per drive times number of drives) must be less than or equal to eight. Also remember that no logical unit can contain more than 65K blocks (32MB).

The handler option is simple. If the ...EIS parameter is set to a non-zero value the DIV instruction will be used in place of a software divide loop. This shortens the handler by a few words and speeds up the entry section slightly.

IV.2 Sysgen Options

There are three sysgen options which affect device handlers: extended memory, error logging, and timer support. If any of these options is being used the system conditional file (SYCND.MAC) created when the system was gen'd will have to be included in the assembly of the handler.

Copy the SYCND file to the disk being used for the WP handler assembly. Then edit the command file WP.COM. Change the line:

```
MACRO/OBJ:WP/ALLOC:10/LIST:WP/CRO:S/SHOW:MEB WPCND+WP
```

to read:

```
MACRO/OBJ:WP/ALLOC:10/LIST:WP/CRO:S/SHOW:MEB SYCND+WPCND+WP
```

If error logging is used then the file ERRTXT.MAC supplied with the WP distribution kit will have to be used in place of the standard file which came with the V4 release kit. ERRTXT is assembled and then linked with ERROUT as described in the RT11 V4.0 Software Support manual, page 7-37.

IV.3 Generating The Handler

Once the procedures specified in the previous two sections have been completed, the handler can be generated by executing the indirect command file WP.COM. The files generated by this operation are:

WP.OBJ, WP.LST, WP.SYG, WP.MAP

If memory management was enabled these files should be renamed:

WPX.OBJ, WPX.LST, WPX.SYG, WPX.MAP