DOMAIN ENGINEERING HANDBOOK

Order No. 002398

Revision 03

Software Release 8.1

APOLLO COMPUTER INC. 330 Billerica Road
Chelmsford, Massachusetts 01824

First printing: September, 1981
Last printing: February, 1985

This document was formatted using the FMT tool
distributed with the Apollo Computer system.

PREFACE

The Apollo DOMAIN® Engineering Handbook contains system information for the Apollo DOMAIN nodes and related peripherals. This manual is for Apollo personnel only.

Audience

This guide is for Apollo employees.

Structure of This Document

This manual contains 11 chapters, three appendices, and an index.

Chapter 1 describes principal control blocks associated with AEGIS internal data structures.

Chapter 2 describes the formats of file system data structures.

Chapter 3 describes miscellaneous information useful for programming in the DOMAIN environment.

Chapter 4 describes all system error (status) codes and messages.

Chapter 5 describes system debugging tools, including the mnemonic debugger, extensions to the HLL debugger and system dumps.

Chapter 6 describes characteristics of peripheral I/O devices, including the format of control registers, and I/O commands.

Chapter 7 describes the hardware architecture of DN300/320 nodes, including the basic processor, Memory Management Unit (MMU), display hardware, FPU, ring, and serial I/O.

Chapter 8 describes the hardware architecture of DN400/420/600 nodes, including the basic processor, Memory Management Unit (MMU), display hardware, PEB, ring/disk, and serial I/O.

Chapter 9 describes the hardware architecture of DN460/660 and DSP160 nodes, including the basic processor, Memory Management Unit (MMU), display hardware, ring/disk, and serial I/O.

Chapter 10 describes the hardware architecture of DN550 nodes, including the basic processor, Memory Management Unit (MMU), display hardware, FPU, ring, serial I/O, MULTIBUS, and VME.

Chapter 11 describes the hardware architecture of DSP80/A nodes, including the basic processor, Memory Management Unit (MMU), ring, serial I/O, and MULTIBUS.

Appendix A describes the ASCII character set.

Appendix B is a powers of two table.

Appendix C is a list of Field Activity Report codes.

Related Documents

For information about system calls, see the DOMAIN System Programmer's Reference Manual, Order Number 000529.

For information about peripheral driver routines, see GPIO User's Guide, Order Number 000959.

For Shell and DM commands see, DOMAIN System Command Reference Manual, Order Number 002547.

Conventions

Symbol	Meaning
+00	Offset, in hexadecimal
-	Unused bit
- - - -	Range of unused bits
....	Continuation
..ltame	Name for this field
N = 1 =>	If bit N is set, then ...
N..N	All bits labeled N
=	Equal to
=>	Implies
—>	Pointer to
->	Pointer to
^	Pointer to
set of 0..31	32 bits
1..32	32 values; 5 bits
<a>	Variable
[pa | va]	Physical address | Virtual address
UPPERCASE	Uppercase words => literal commands or keywords
lowercase	Lowercase words => command values that you must supply.
[ ]	Square brackets enclose optional items in formats and commands.
{ }	Braces enclose a list from which you must choose an item in formats and command descriptions.
|	A vertical bar separates items in a list of choices
< >	Angle brackets enclose the name of a key on the keyboard.

TABLE OF CONTENTS

ack or ACK — acknowledge
addr — address
attn — attention
attr — attribute
aux — auxiliary

char — character
clk — clock
cmd or CMD — command
config — configure
ctl — control
ctlr — controller
cyl — cylinder

DADDR — direct address
DAT — date
diag or DIAG — diagnostics
disp — display

exp — exponent
EXTN — extension

Hld — hold
hdr — header

Init — initialize
I/O — Input/Output
Ins — Instruction
Int — Interrupt

lpt — line printer
LSB — Least Significant Byte

Mb — megabyte
mem — memory
MSB — Most Significant Byte
msg — message

num — number

PA — physical address
PCB — printed circuit board
phys mem — physical memory
pkt — packet

rcv — receive
rcvr — receiver
ref — reference
reg — register(s)
R/W — Read/Write

s/b — should be
spec — specification(s)
sync — sychronization or sychronize

tx — transmit

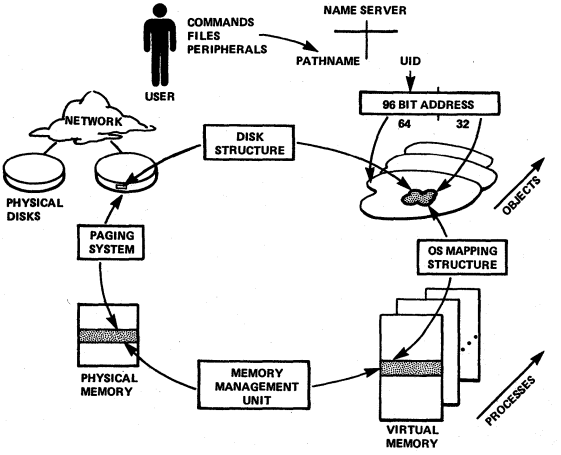
uaddr — unaddress
wrt — write

xfer — transfer
xmit — transmit

CHAPTER 1

AEGIS

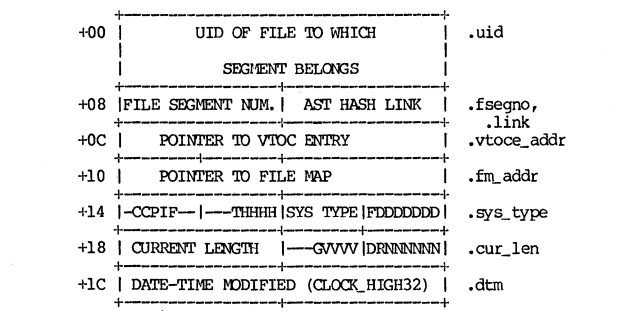
AEGIS SYSTEM RELATIONSHIPS



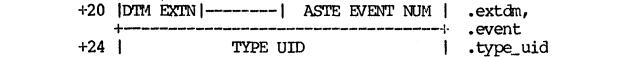
ACTIVE SEGMENT TABLE (AST)

AST: ARRAY[1..N] OF ASTE_T
N=448 or 1984 (DNx60)

(type "aste_t" in vm.ins.pas)



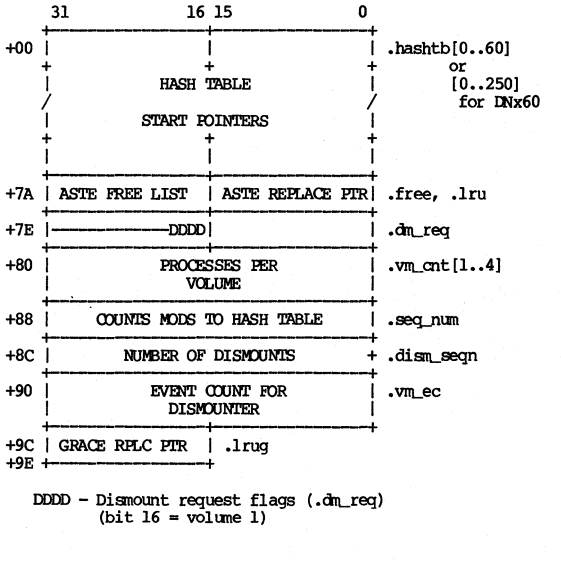
DNx60 also includes:



CC - Concurrency control from VTOC entry (.con_ctrl)
P - Permanent segment (.permanent)
I - Immutable segment (.immutable)
F - File trouble (file_trouble)
T - In transition (.in_trans)
H..H - ASTE hold count (.ehcnt)
F - File-map modified (.fm_mod)
D..D - Blocks added to seg since active (.blocks_delta)
G - Global transparent mode sw: no dtm (.gtms)
V..V - Index in DVT (.volx)
D - DTM needs updating (.dtm_fl)
R - Grace flag (.gracef)
N..N - Number of pages resident (.npr)

The AST is a wired system-wide table that describes the current state of active segments. Each ASTE has enough information to identify the file segment and, through the associated AST_PMAPS, the location of each page of the segment. All virtual segments mapped to the same file segment will use the same ASTE entry. The ASTE acts as a cache of the VTOC entry. The table is dynamically allocated at startup based on physical memory size.

ACTIVE SEGMENT TABLE HEADER



DDDD - Dismount request flags (.dm_req)
(bit 16 = volume 1)

AST_PAGE_MAPS

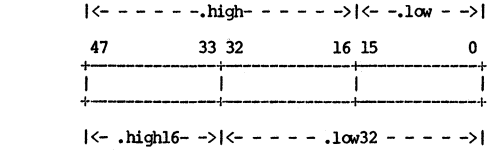
```
AST_PMAPS: ARRAY[1..448] OF PMAPE_T
PAGE MAP ENTRY (PMAPE):
(type "pmape_t" in vm.ins.pas)

If page is resident:
     +----------------+-----------------+
 +00 |WWTR|PHY PAGE NUM| DISK ADDR LOW   |
     +-------------------------------------+

If page is NOT resident:
     +----------------+-----------------+
 +00 |WWTR|-------N|   FULL DISK ADDRESS   |
     +-------------------------------------+
WW   - Wired count (.wired)
T    - Page is in transition (.in_trans)
R    - Page is resident in memory (.resident)
N    - Page is null (no copy on disk) (.null)
```

AST_PAGE_MAPS (DNx60)

```
AST_PMAPS: ARRAY[1..1984] OF PMAPE_T

If page is resident:
     +----------------+-----------------+
 +00 |VUMIXTWW|RNHHHHHH| PHYS PAGE NUM   |
     +-------------------------------------+

If page is NOT resident:
     +----------------+-----------------+
 +00 |VUMIXTWW|RN|   FULL DISK ADDRESS   |
     +-------------------------------------+
V    - Entry is valid (.valid)
U    - Page has been used, set by hardware (.used)
M    - Page has been modified, set by hardware (.pmod)
I    - Indirect entry (.indirect)
X    - Page may be written (.write)
H..H - Disk address high bits
Others as shown above.
```

As of SR7, page maps are not part of the AST. If a page is resident, some bits of the disk address are in the MMAP entry for the page (depending on the system). In DNx60 systems, these are the page tables used by the address translation hardware. AST_PMAPS are dynamically allocated one-for-one with ASTEs.

(type "clock_t" in base.ins.pas)



Low-order bit represents four microseconds.

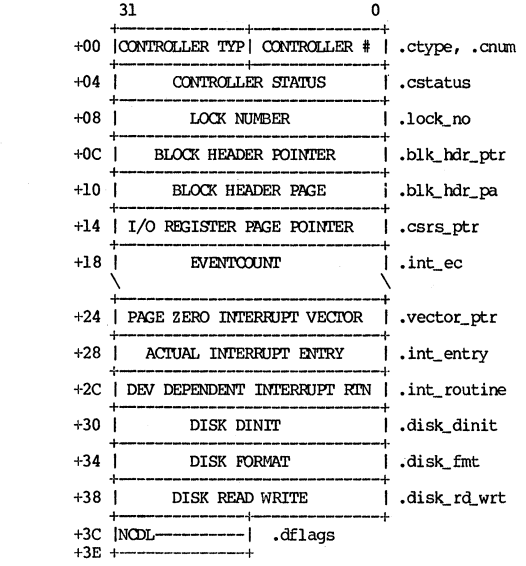
Clock.high is type clockh_t (base.ins.pas);
low-order bit represents approx. four milliseconds.

The hardware clock is only 16 bits. The upper 32 bits are maintained by the clock interrupt routine. There are three hardware clocks (not counting the calendar clock):

Clock 1: 4 microseconds (real-time clock)
Clock 2: 8 microseconds (process timer)
Clock 3: 128 microseconds (real-time intervals)

DISK CONTROLLER TABLE ENTRY

(type dcte_t in io.ins.pas)

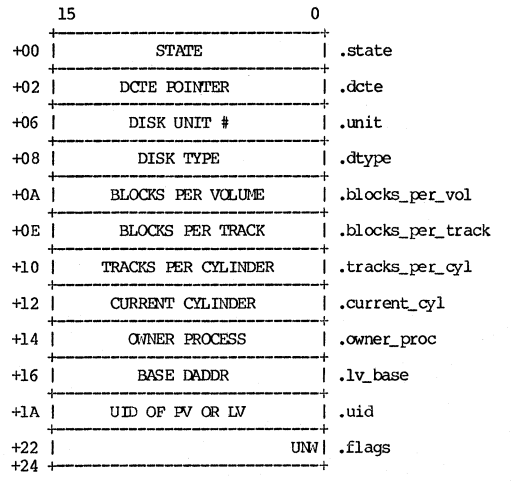


DISK FLAGS:

N - 1 = no headers on device
C - 1 = do checksumming for this controller
D - 1 = driver supports multi-read requests (temp)
L - 1 = liberty (smd) type winchester

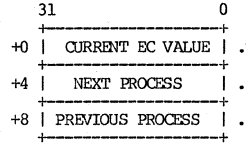
DISK VOLUME TABLE ENTRY

(type dvte_t in disk.pvt.pas)



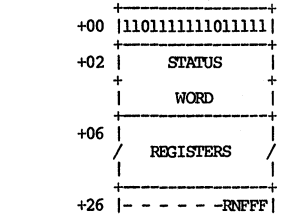
EVENT COUNT

(type eventcount_t of base.ins.pas)



FAULT DIAGNOSTIC RECORD

(type "fault_$diag_t" in fault.ins.pas)

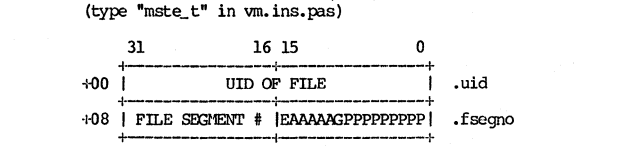


R - Read operation (.write_op)
N - Not instruction reference (.not_inst)
FFF - Instruction function code (.function_code):
001 User data
010 User program
101 Supervisor data
110 Supervisor program
111 Interrupt acknowledge

MAPPED SEGMENT TABLE (MST) (non-DNx60)

MST: ARRAY[0..26,0..319] OF MSTE_T (16Mb of VA)

MAPPED SEGMENT TABLE ENTRY (MSTE):
(type "mste_t" in vm.ins.pas)



E = 1 => File extension allowed (.ext_ok)
AAAAA - Access (.access):

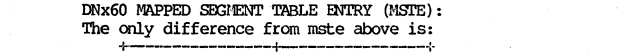
00000	Nil (access_$nil)
00010	Read (access_$r)
00011	Read-execute (access_$rx)
00110	Write-read (access_$wr)
00111	Write-read-execute (access_$wrx)
10010	Supervisor read (access_$sr)
10011	Supervisor read-execute (access_$srx)
10110	Supervisor write-read (access_$swr)
10111	Supervisor write-read-execute (access_$swrx)

G = 1 => Interrrupt on ref (.guard)
P - Probable ASTEX (.pastex)
R - Object is remote (.mste_remote)
C - Touch ahead count (.mste_touch_cnt)
S - Sequential access
Q - Next sequential page
M - Mste-node

MAPPED SEGMENT TABLE (MST) (DNx60)

MST: ARRAY[0..26,0..7680] OF MSTE_T (256Mb of VA)

DNx60 MAPPED SEGMENT TABLE ENTRY (MSTE):
The only difference from mste above is:



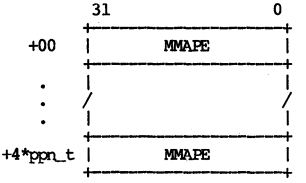
(larger .pastex, no access rights (see SMAPE)

The MST is a wired per-ASID table describing the process's address space. Each entry represents one virtual memory segment describing the object to which this segment is mapped.

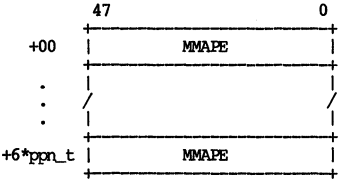
MEMORY MAP (MMAP)

The MMAP is a system-wide table describing the contents of physical memory. There is one entry per physical page of memory.

OTHER THAN DNx60

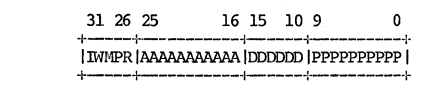


DNx60

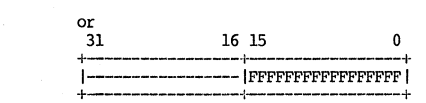


MEMORY MAP ENTRY (MMAPE)

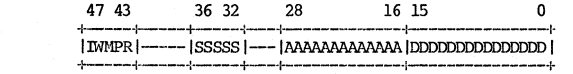
OTHER THAN DNx60 (type "mmape" in mmap.pvt.pas)



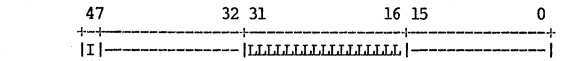
or



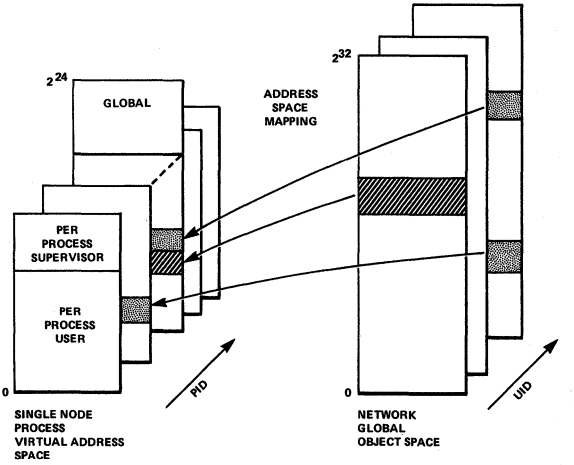
DNx60
(type "mmape" in mmap.pvt.pas)

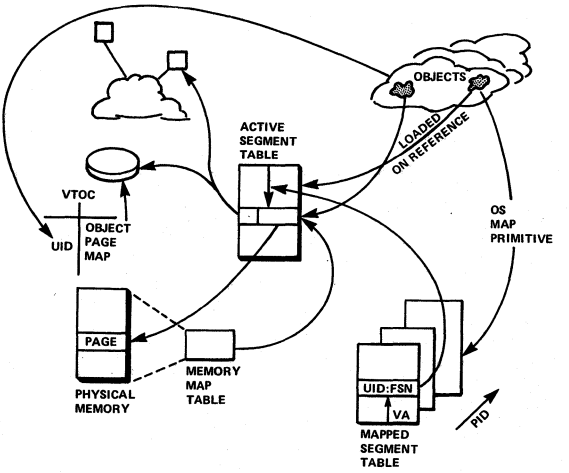


or



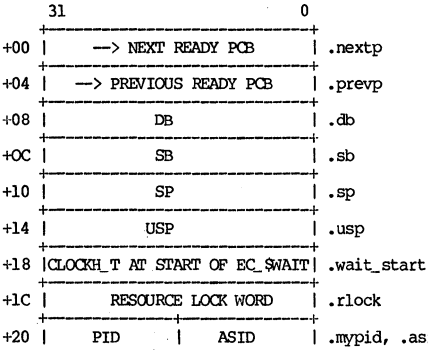
I = 1 => Entry in use (.inuse)
W = 1 => Wired
N = 1 => No copy on disk (.null)
M - Page modified but dtm shouldn't be updated (.rmod)
P = 1 => Purifier used (.usedp)
R = 1 => Replacer used (.usedr)
A..A - ASTE index (.astex)
D..D - Disk address high (.daddr_h)
P..P - PTT index (.pttx)
S..S - Segment page number (.mspgno)
L..L - Link to next free MMAP entry (.link)
F..F - Free entry


2^{24}
GLOBAL
PER
PROCESS
SUPERVISOR
PER
PROCESS
USER
0
PID
SINGLE NODE
PROCESS
VIRTUAL ADDRESS
SPACE
ADDRESS
SPACE
MAPPING
2^{32}
0
UID
NETWORK
GLOBAL
OBJECT SPACE


OBJECTS
LOADED ON REFERENCE
ACTIVE SEGMENT TABLE
VTOC
UID
OBJECT PAGE MAP
OS MAP PRIMITIVE
PAGE
PHYSICAL MEMORY
MEMORY MAP TABLE
UID:FSN
VA
PID
MAPPED SEGMENT TABLE

PROCESS CONTROL BLOCK (PCB)

(type "procl_t" in procl.put.asm)



State: T - Time slice end with rlock <> [] (tse_onb)
B - Bound (bound)
P - Suspension pending (susp_pending)
S - Suspended (suspended)
W - Waiting (waiting)

```
procl_$state_t = SET OF (waiting, suspended, susp_pending,
                         bound, tse_onb);
```

(State is type "procl_$state_t" in procl.ins.pas)

PROCESSES

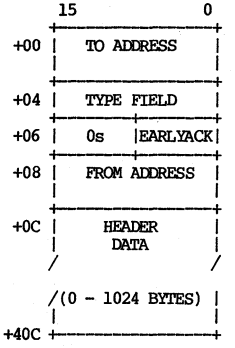
PID	ASID	Description
1	1	Initial user process and DM
2	0	Null process
3	0	Clock process
4	0	Page purifier
5	0	Network-receive server
6	0	Network-paging server
7	0	Network-general server
8	0	Terminal helper
9	2	First user process
10	3	Second user process
11	4	.
12	5	.
13	6	.
14	7	.
25	18	Last user process

NOTE: Processes with an ASID of 0 run entirely in global space.

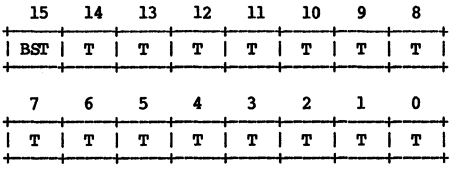
NOTE: If the debugger's Lights command has been given, there will be a Lights process with PID 9 or greater with a zero ASID.

RING PACKET FORMAT

Message Header



Type Field

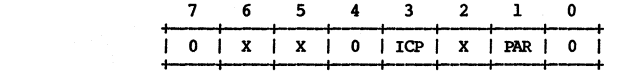


Bit 15 - BST (Broadcast)- This bit is set in a packet intended to be broadcast to all receivers. If it is set, the To Address field is ignored.

Bit 14 through 0 - T (Type)- This field determines whether a packet is to be received. Each 1 bit in the received Type field is compared to the corresponding bit in the controller Type register. If any bit selected in the Type register is a 1, the message is received. If all bits selected in the Type register are 0, the message is ignored. The APOLLO I implements bits 14 through 8 only; bits 7 through 0 never match.

Early Acknowledge Field

The Early Acknowledge field is inserted by the transmitter and modified by the receivers. For the purposes of the CRC calculation, the EARLY ACK field is treated like a byte of zeros. This allows receivers to modify the ACK field without having to recompute the CRC checksum.



Bit 7-0 - This bit is inserted to prevent the remaining bits in the late-acknowledge byte from being modified by the bit-stuffing protocol.

Bit 6-X - Don't care - this bit is not used.

Bit 5-X - Don't care - this bit is not used.

Bit 4-0 - This bit is inserted to prevent the remaining bits in the late-acknowledge byte from being modified by the bit-stuffing protocol.

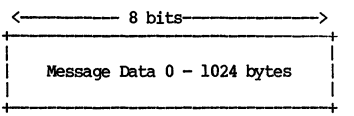
Bit 3-ICP - Intend to Copy - This bit is set by an addressed receiver if it was set up to copy a message and the type field matched. A NAK (negative acknowledge) condition is indicated when no receiver sets this bit.

Bit 2-X - Don't care - This bit is not used.

Bit 1-PAR - Parity - This parity bit is set so that there are an odd number of bits in the late acknowledge byte.

Bit 0-0 - This bit is inserted to prevent the remaining bits in the late-acknowledge byte from being modified by the bit-stuffing protocol.

Message Data

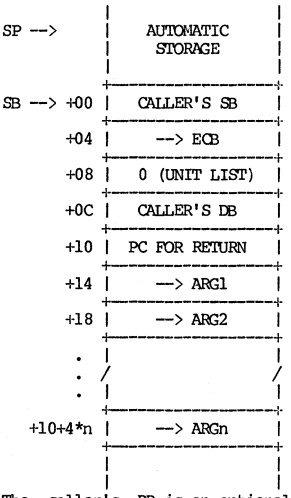


The Message Data field varies in length from 0 to 1024 bytes but must always be an even number of bytes(on the APOLLO I the controller always uses word DMA). The contents of the Message Data field is determined by the software; it is transmitted verbatim.

RESOURCE LOCK

```
network_$serv_lock,{ 00 1 }
mt_$lock,          { 01 2 }
ml_$free3,         { 02 4 }
ml_$free4,         { 03 8 }
ml_$free5,         { 04 10 }
file_$lock_lock,   { 05 20 }
ec2_$lock,         { 06 40 }
smd_$respond_lock, { 07 80 }
smd_$request_lock, { 08 100 }
disk_$mnt_lock,    { 09 200 }
term_$lock,        { 10 400 }
procl_$create_lock,{ 11 800 }
onb_$lock,         { 12 1000 faulted to CPUB }
bok_$lock,         { 13 2000 can be run on B }
vtuid_$lock,       { 14 4000 }
vtoc_$lock,        { 15 8000 }
bat_$lock,         { 16 10000 }
ast_$lock,         { 17 20000 }
pag_$lock,         { 18 40000 }
ml_$free6,         { 19 80000 }
flp_$lock,         { 20 100000 }
win_$lock,         { 21 200000 }
ring_$xmit_lock,   { 22 400000 }
ml_$free7,         { 23 800000 }
                   { the next two locks are highest}
time_$proc_lock,   { 24 1000000 clock process only }
time_$lock         { 25 2000000 clock process database }
```

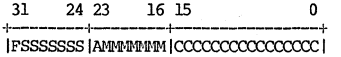
STACK FRAME



The caller's DB is an optional field. The "DB not saved" bit in ECB signals that it is not present.

STATUS WORD

(type "status_t" in base.ins.pas)



F = 1 => module couldn't handle error (.fail)
S..S - Subsystem identification (.subsys)
A = 1 => asynchronous fault; only set during delivery of fault (.async)
M..M - Module identification (.module)
C..C - Module-specific error code (.code)

See Chapter 4 - Error Codes and Messages.

SYSTEM BOOT FILES

FILES REQUIRED DURING BOOT

REQUESTING AGENT	FILE	
PROM	/SYSBOOT (records 2-B on track 0)	
(If DNx60)	/SAUn/WCS.UC	(microcode file)
	DCODE.UC	(instr. decode RAM contents)
	SPAD.UC	(scratchpad constants and temps)
	ULOAD	(program to load the above)
SYSBOOT	/SAUn/AEGIS	(AEGIS load file)
	/SAUn/SALVOL	(only if salvage required)
AEGIS	[os paging file]	(uncatalogued)
	//	(UIDs found and saved by NAME_$INIT)
	/	
	/COM	
	/SYS/NODE_DATA	
	/SYS/PEB_MICROCODE or PEB2_MICROCODE(1)	
	`NODE_DATA/SHELL(2)	(mapped by PROC2_$INIT)
SHELL	/SYS/APOLLO_LOGO(3)	
	`NODE_DATA/STARTUP_SHELL(3)	(cmd file to override dflts)
	/SYS/ENV	(SHELL tells him what to run)
ENV	/SYS/DM/DM	"GO" command or normal boot -OR-
	/SYS/BOOT	"SH" or boot from SIO line -OR-
	/SYS/SPM/SPM	"SPM" or normal boot on server node
DM	`NODE_DATA/DEV/SIO	
	/SYS/DM/FONTS	
	`NODE_DATA/STARTUP[.19L, .COLOR](3)	
	/SYS/BOOT	
BOOT	/REGISTRY/REGISTRY(4)	(+PPO,Account files pointed to)
	LOCAL_REGISTRY	
	LOCAL_SITE/?*	
	/COM/SH	

NOTES:
(1) PEB is disabled if microcode file not found.
(2) If booted from cartridge tape, the tape is first searched for BSCOM/RBAK_SHELL.
(3) Optional — system will manage without it.
(4) If no registries are available, you can login only as USER.NONE.NONE.

SYSTEM DIRECTORIES

```
/aux      AUX (UNIX) (optional)
/bscom    Boot shell command directory
/cc       C compiler (optional)
/com      Shell commands directory
/core     Core graphics (optional)
/dev      Peripheral I/O device definitions
/install  Installation scripts directory
/lib      System libraries directory
/sau1     Stand-alone utilities for DN4xx/600
/sau2     Stand-alone utilities for DN300/320
/sau3     Stand-alone utilities for DSP80,DSP80A
/sau4     Stand-alone utilities for DNx60,DSP160
/sau5     Stand-alone utilities for DN550
/sys
   /sys/alarm            Alarm server
   /sys/boot             Boot shell file
   /sys/dm               Display Manager programs
   /sys/dm/fonts         Display Manager font definitions
   /sys/env              Process 1 initializer
   /sys/hasp             Hasp support
   /sys/help             Help text files
   /sys/ins              User insert files
   /sys/mbx              Mailbox helper
   /sys/net              Netman (diskless node support)
   /sys/node_data[.nn]   Per node read/write data files
   /sys/sf               Store and forward files
   /sys/siologin         Sioline login support
   /sys/source           Selected source files
   /sys/spm              Server process manager
   /sys/stream_$sfcbs    Stream manager control blocks
   /sys/subsys           Protected system support
   /sys/sysdev           Stream device files
/sysboot   System boot file
/systest   On-line system tests directory
```

TRAP CODES

0	SVC — 0 arguments
1	SVC — 1 argument
2	SVC — 2 arguments
3	SVC — 3 arguments
4	SVC — 4 arguments
5	SVC — any number of arguments
6	SVC — from GPIO interrupt routines
7	Undefined (reflected to user space)
8	Undefined (reflected to user space)
9	Undefined (reflected to user space)
A	Undefined (reflected to user space)
B	Undefined (reflected to user space)
C	Undefined (reflected to user space)
D	Undefined (reflected to user space)
E	Software-generated fault (pfm_$error_trap)
F	Undefined (traps to PROM debugger)

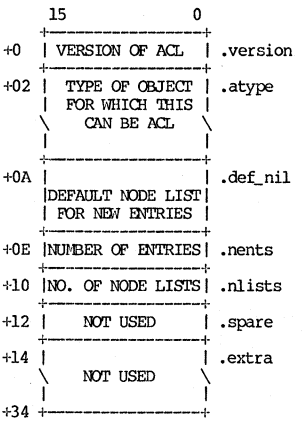
CHAPTER 2

FILE SYSTEM

ACLs STRUCTURE

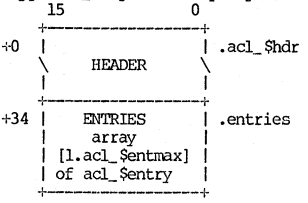
ACL Header Record

(type acl_$hdr in acls.pvt.pas)



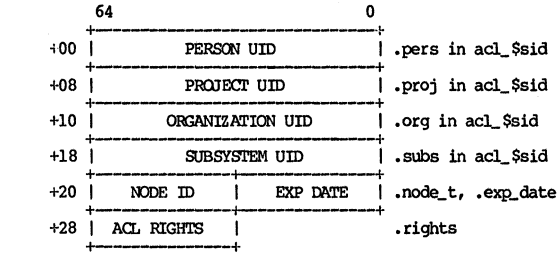
ACL Record

(type acl_$rep in acls.pvt.pas)



ACL Entry

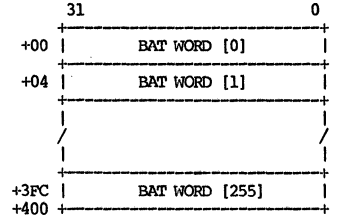
(type acl_$entry in acl.ins.pas)



BLOCK AVAILABILITY TABLE (BAT)

(type "bat_blk" in vol.ins.pas)

type bat_blk_t= array[0..255] of bat_lword_t

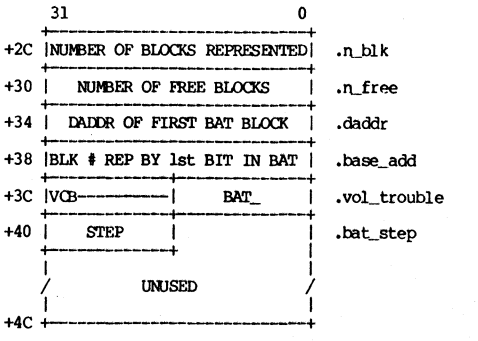


First BAT block pointed to by BAT header in logical volume label (pv_label). BAT resides in contiguous records.

Bit 0, BAT WORD[0] corresponds to the first block (bat_hdr.base_add) in the logical volume. BAT bit = 1 if block is available.

BLOCK AVAILABILITY TABLE HEADER

(type "bat_hdr_t" in vol.ins.pas)



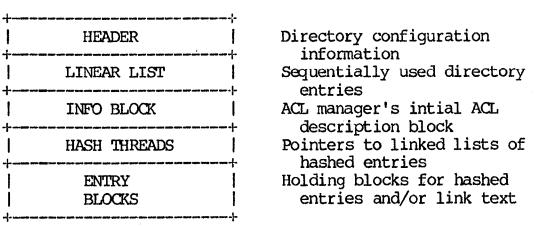
V - Volume trouble, set by OS if volume needs salvaging, cleared by SALVOL.
C - Volume CHUVOLed
B - Volume being CHUVOLed

BAT header lives in logical volume label.
Offsets given are from start of label.

DIRECTORY STRUCTURE

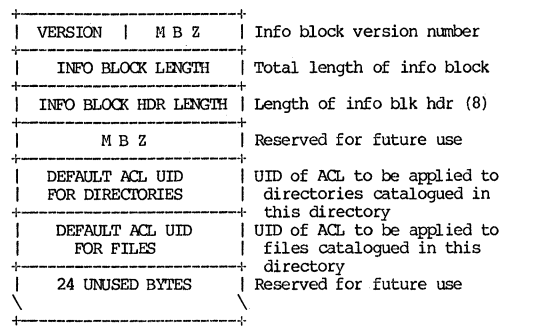
Directory Overview

(type dir_t in name.pvt.pas)



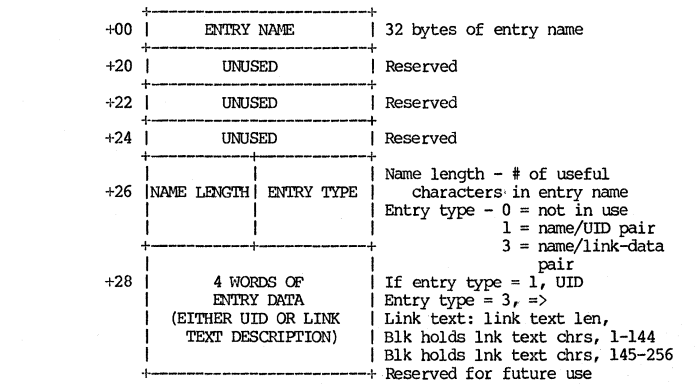
Directory Information Block

(type infoblk_hdr_t in name.pvt.pas)



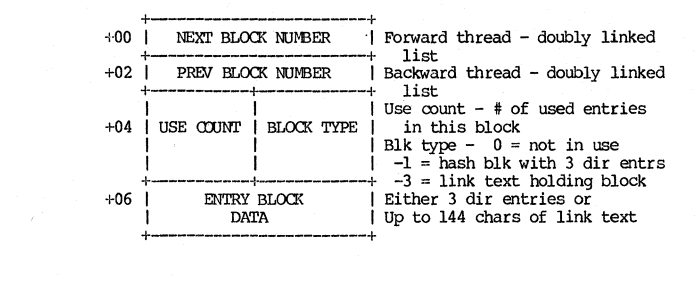
Directory Entry

(type dir_entry_t in name.pvt.pas)



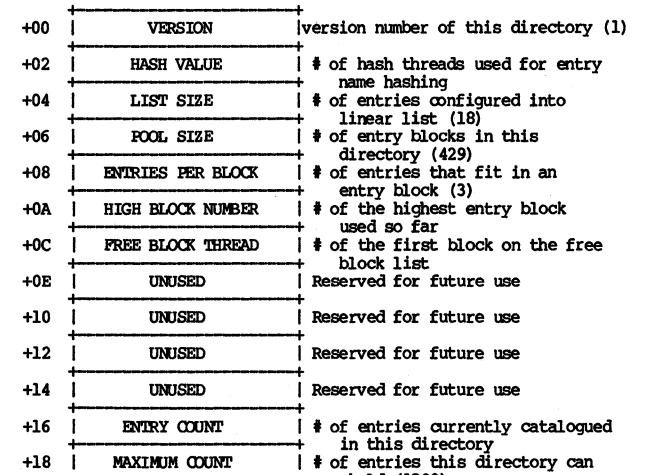
Directory Entry Block

(type entry_block_t in name.pvt.pas)
total length - 150 bytes



Directory Header

(First part of type "dir_t" in name.pvt.pas)



Notes on Directories

1. To add an entry to a directory:

   A. Look for an unused entry in the linear list. If you find one, use it and you're done.

   B. Hash the name you want to add:

```
- name is:
    name: array [1..32] of CHAR
- length is useful length of name
  sum: =0;
  For i : = 1 to length DO
     sum : = ord(name[i])+2*sum;
  HASH_VAL : = sum mod HASH_VALUE;
```

   C. Get the hash thread for the specified hash value and call that value the found block.

   D. If the found block number is 0 then we need a new entry block, so:

      a) See if there are any blocks threaded through the free block list and, if so, take one of those. Otherwise, bump the high block number and use that.

      b) Initialize the newly obtained block, add it to the end of the appropriate hash chain, add the new entry as the first entry in the new entry block, and you're done.

   E. If there is an unused entry in the found block, use it and you're done.

   F. Change the found block value to the number in the current found block's NEXT BLOCK field and go to step D.

2. The searching rule for a directory is:

   A. Look in the linear list.

   B. Hash the name you're searching for.

   C. Follow the hash thread for the specified hash value to the first entry block with that hash synonym.

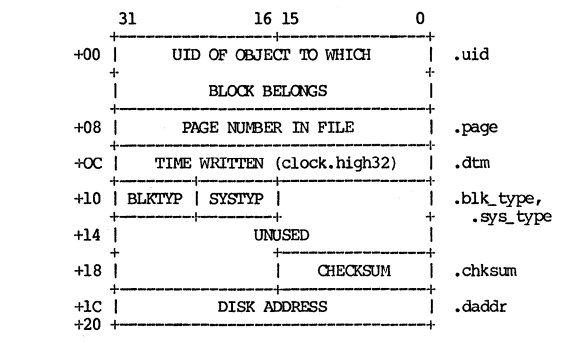
   D. Search all three of the entries in the found entry block.

E. Follow the "next block number" in the found entry block to get a NEW found entry block. If the next block number is zero, then return NOT FOUND.

F. Go to step D with the newly found block.

DISK BLOCK HEADER

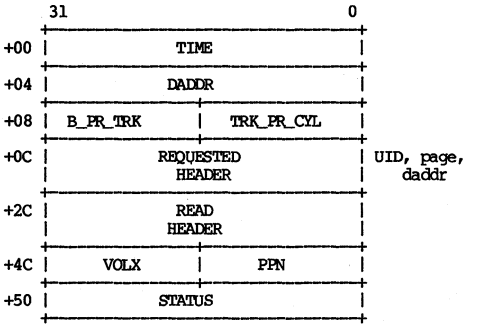
(type "blk_hdr_t" in base.ins.pas)



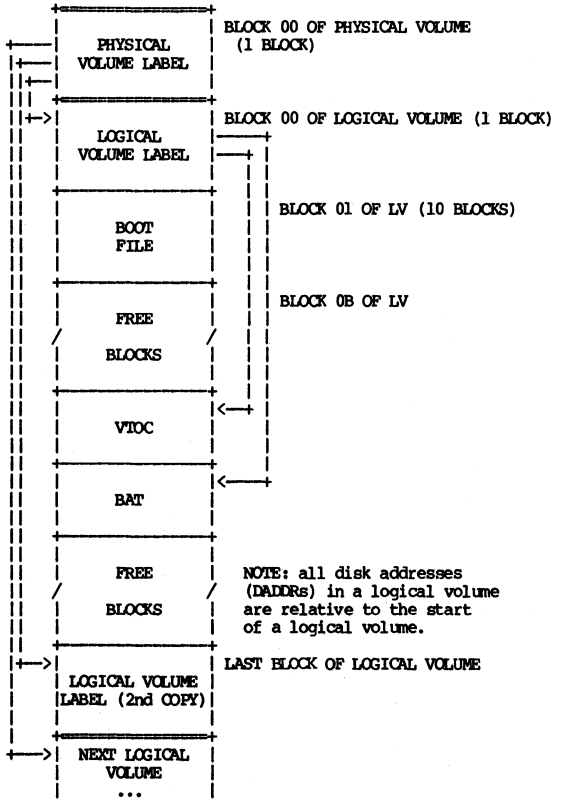
BLKTYP: 0 - Data block
1 - Level 1 index block in file map
2 - " 2 " " " " "
3 - " 3 " " " " "

SYSTYP: 0 - File
1 - Directory
2 - System directory

DISK_$ERROR_INFO

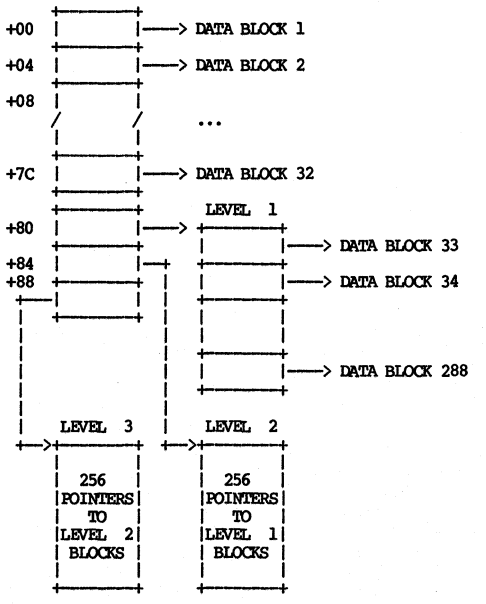


DISK/VOLUME FORMAT



Badspots may cause the VTOC to be non-contiguous and not adjacent to BAT. There may be dead space between logical volumes.

FILE MAP



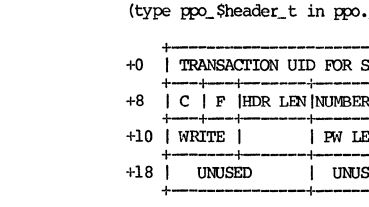
Maximum file size = (32+256+256**2+256**3)*1024 bytes
= 17,247,300,000 bytes.

File map resides in VTOCE and AST entries.

REGISTRY FORMAT

Header Record

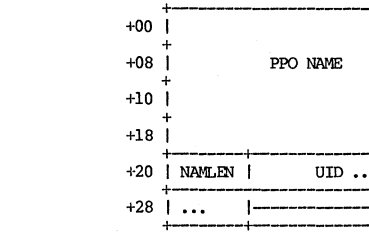
(type ppo_$header_t in ppo.ins.pas)



C = 1 => committed (.ppo_$committed)
F = 1 => local (.ppo_$local_flag)
NUMBER - number of entries (.ppo_$num_entries)
READ - oldest software that can read this (.ppo_$r_vers)
WRITE - oldest software that can write new (.ppo_$w_vers)
PW LEN - minimum password length (.ppo_$min_plen)

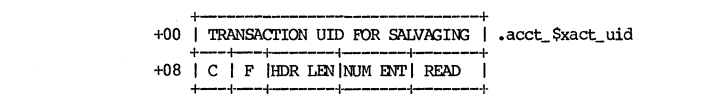
PPO Record

(type ppo_$record_t in ppo.ins.pas)



Account Header

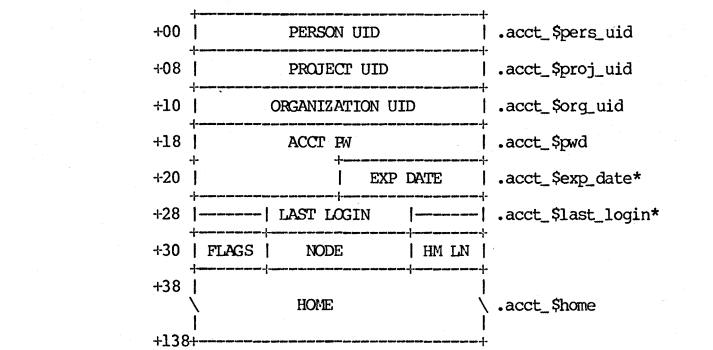
(type acct_$header in acct.ins.pas)



C = 1 => committed (.acct_$committed)
F = 1 => local (.acct_$local_flag)
HDR LEN - header length (.acct_$hdr_len)
NUM ENT - number of entries (.acct_$num_entries)
READ - oldest software that can read this (.acct_$r_vers)
WRITE - oldest software that can write new (.acct_$w_vers)
PW LEN - minimum password length (.acct_$min_plen)
REC LEN - record length (.acct_$rec_len)

Account Record

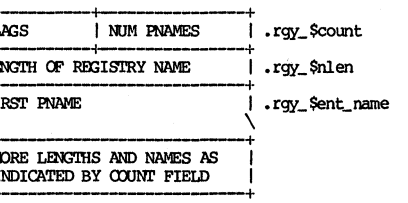
(type acct_$record_t in acct.ins.pas)



* local registries only
FLAGS - set of acct_$invalid (local registries only)
(.acct_$flags)
NODE - node type (local registries only) (.acct_$node)
HM LN - home length (.acct_$home_len)

Registry Record

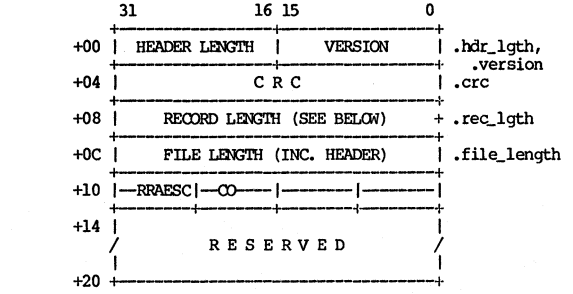
(type rgy_$registry_t in rgy.ins.pas)



First pname is path of original registry; it is used as lock.

STREAM FILE HEADER

(type "stream_$hdr_rec_t" in sbase.ins.pas)



CRC = -(integer sum of (1 + 3 through n longwords))
RR - Record type (.rec_type, stream_$rtype_t):
00 - var len w/counts (stream_$vl)
01 - fixed length (stream_$f2)
10 - no record structure (stream_$undef)

A = 0 => Binary, 1 => ASCII (.elb_flag)
E = 1 => No automatic type change (.explicit_type)
S = 1 => File may have holes (.sparse)
C = 1 => Carriage control (ASCII only) (.cc)

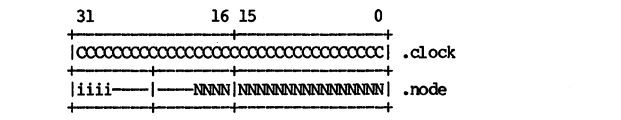
CO - Concurrency (.conc, stream_$fconc_t):
00 - N readers or 1 writer (stream_$n_or_1)
01 - N readers and 1 writer (stream_$n_and_1)
10 - N readers and N writers (steam_$n_and_n)

RECORD LENGTH: Record length for fixed
Maximum length for variable
0 for undefined record length

Stream file header is the first 32 bytes of a file to be accessed by the stream interface.

UNIQUE IDENTIFIER (UID)

(type "uid_t" in base.ins.pas)



C..C - Top 32 bits of clock (4 mSec units)

iiii - A counter if more than one UID is generated in one four-millisecond interval

N..N - Node ID

UID Hash Algorithm

X = the four words of the UID XORed together

INDEX = X mod TABLE_SIZE

where TABLE_SIZE is the size of the table into which the UID is being hashed (e.g., vtoc_hdr.vtoc_size).

UIDs — System

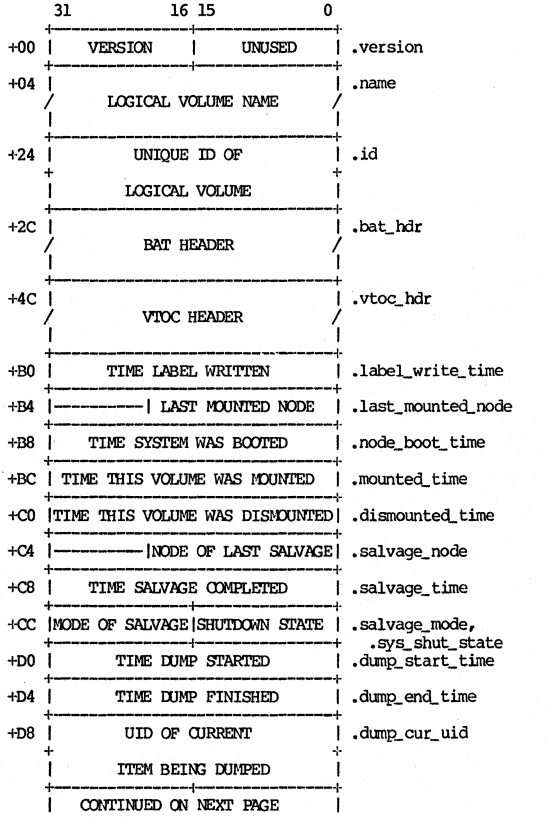
(from /os/nuc/uid_list.asm)

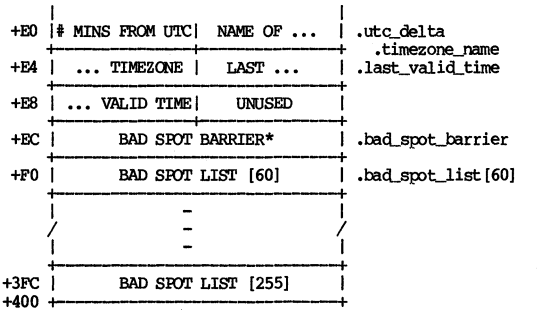
```
uid_$nil                00000000,0
acl_$nil                00000100,0

disk structure canned UIDs (000002xx series)

pv_label_$uid           00000200,0
lv_label_$uid           00000201,0
vtoc_$uid               00000202,0
bat_$uid                00000203,0

canned object type UIDs (000003xx series)

records_$uid            00000300,0
hdr_undef_$uid          00000301,0
object_file_$uid        00000302,0
UNDEF_$uid              00000304,0
pad_$uid                00000305,0
name_$canned_root_uid   00000308,0
input_pad_$uid          00000309,0
sio_$uid                0000030A,0
ddf_$uid                0000030B,0
mbx_$uid                0000030C,0
nulldev_$uid            0000030D,0
D3M_area_$uid           0000030E,0
D3M_sch_$uid            0000030F,0
pipe_$uid               00000310,0
uasc_$uid               00000311,0
directory_$uid          00000312,0
unix_directory_$uid     00000313,0
mt_$uid                 00000314,0
sysboot_$uid            00000315,0

canned objects UIDs (000004xx series)

display1_$uid           00000400,0
display2_$uid           00000401,0
special_seg_$uid        00000402,0
```

```
canned person. project, organization
      and subsystem UIDs (005xx series)

canned persons (0000050x series)

    acl_$sys_user_uid      00000500,0

canned projects (0000054x series)

    acl_$sys_proj_uid      00000540,0
    acl_$login_uid         00000541,0
    acl_$locksmith_uid     00000542,0

canned organizations (0000058x series)

   acl_$sys_org_uid        00000580,0

canned subsystems (000005Cx series)

   acl_$nil_subs_uid       000005C0,0

canned ACL type UIDs (000006xx series)

   acl_$file_acl           00000600,0
   acl_$dir_acl            00000601,0

canned ACL UIDs (file ACLs) (0001xxxx series)

   acl_$fnil               00010000,0
   acl_$fndwrx             0001800F,0
   acl_$file_nwrx          00018007,0

canned ACL UIDs (directory ACLs) (0002xxxx series)

   acl_$dnil               00020000,0
   acl_$dndcal             0002801F,0
   acl_$dir_ncal           0002800F,0
```

VOLUME LABEL -- LOGICAL

(type "lv_label_t" in vol.ins.pas)





salvage_mode: currently unused; always = 1

```
sys_shut_state:   lv_dismounted = 0
                  lv_mounted    = 1
                  lv_salvaged   = 2
```

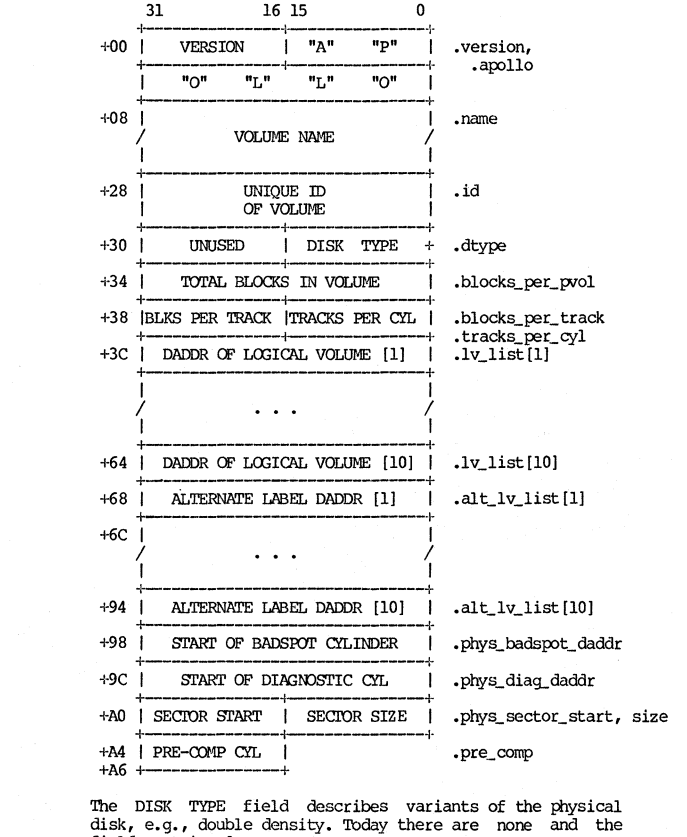
bad_spot_list allocated from end of list.

```
* FFFFFFFF -> no badspot list overflow
  FFFFFFFE -> +F0 is DADDR of overflow block
```

The LV label is the first block of a logical volume. The alternate LV label is a copy of the LV label and lives at or near the end of the logical volume.

VOLUME LABEL — PHYSICAL

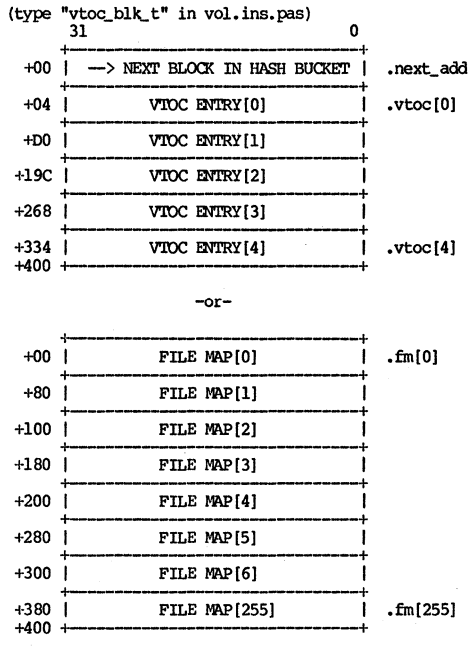
(type "pv_label_t" in vol.ins.pas)



The DISK TYPE field describes variants of the physical disk, e.g., double density. Today there are none and the field contains 0.

The PV label is the first block (cylinder 0, track 0, block 0) of a physical volume.

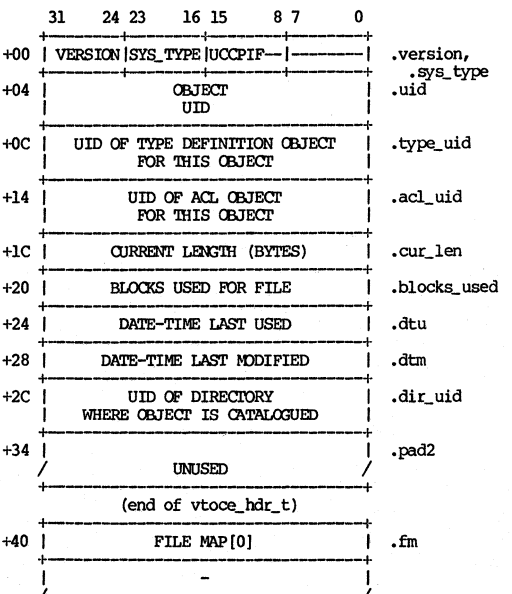
VTOC BLOCK



When the VTOC block contains a file map, the block is pointed to by vtoce.fm2[1-3] (see VTOC ENTRY).

VTOC ENTRY

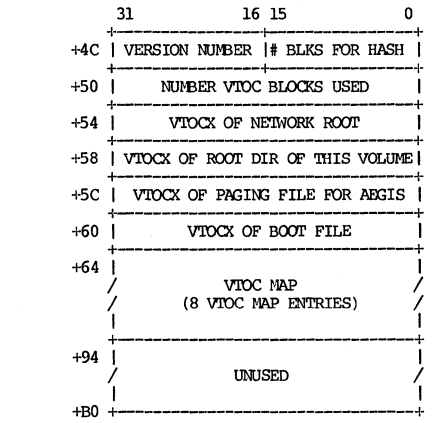
(types "vtoce_hdr_t" and "vtoce" in vol.ins.pas)



U - VTOC entry in use (.inuse)
CC - Concurrency control (.con_ctrl):
00 - None
01 - Shared
10 - Exclusive
P - Permanent (.permanent)
I - Immutable (.immutable)
F - File needs salvaging (.trouble)

VTOC HEADER

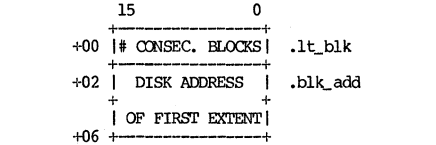
(type "vtoc_hdr_t" in vol.ins.pas)



The VTOC header lives in the logical volume label. Offsets given are from the start of the label.

VTOC MAP ENTRY

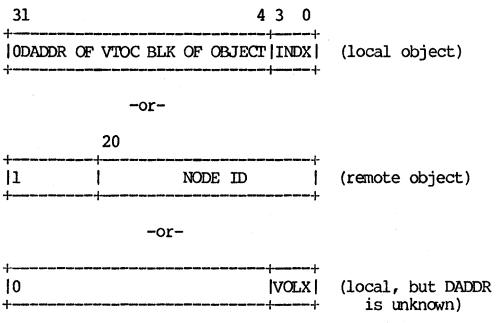
(type "vtoc_mape" in vol.ins.pas)



Each VTOC map entry describes one set of contiguous VTOC blocks ("extent"). VTOC extents are preallocated by INVOL to be near the middle of the logical volume and to avoid badspots.

VTOC_INDEX

(type "vtocx_t" in base.ins.pas)



INDX - Index of VTOC entry in VTOC block (0-4)
or File Map index (0-7)
VOLX - Logical volume number

CHAPTER 3

PROGRAMMING INFORMATION

ADDRESSING MODES

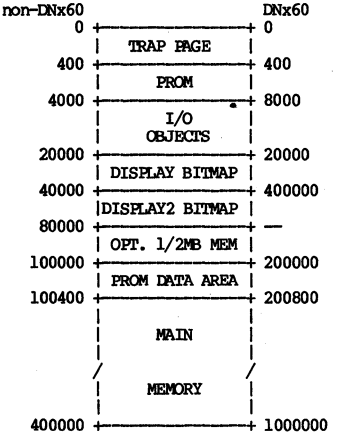
ADDRESSING MODES*

Effective Address Modes	Mode	Register	Addressing Categories				Assembler Syntax
			Data	Memory	Control	Alterable	
Dn	000	register number	X			X	Dn
An	001	register number				X	An
An@	010	register number	X	X	X	X	(An)
An@+	011	register number	X	X		X	(An)+
An@–	100	register number	X	X		X	–(An)
An@(d)	101	register number	X	X	X	X	d(An)
An@(d, ix)	110	register number	X	X	X	X	d(An, Ri)
xxx.W	111	000	X	X	X	X	xxx
xxx.L	111	001	X	X	X	X	xxxxxx
PC@(d)	111	010	X	X	X		PC relative
PC@(d, ix)	111	011	X	X	X		PC rel. + Ri
#xxx	111	100	X	X			#xxx

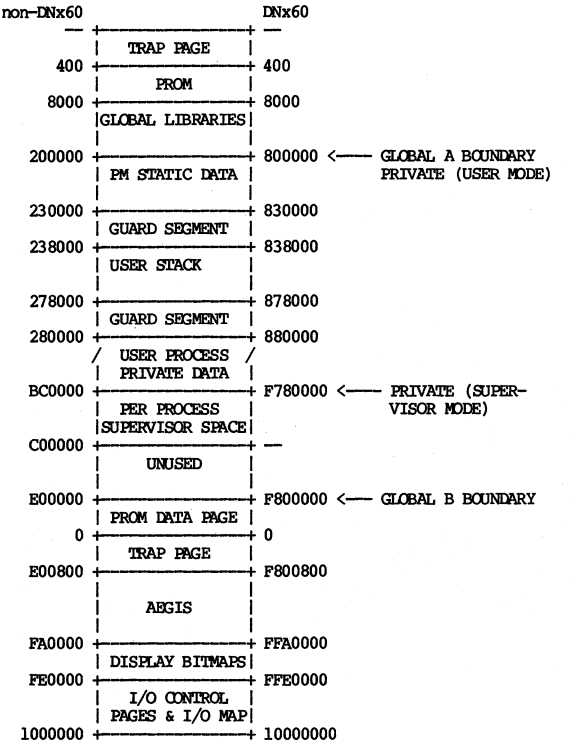
*Reprinted from MC68000, page B-1.

ADDRESS SPACE

Physical Address Space



Virtual Memory



CALLING SEQUENCE

Caller:

```
PEA       ARGn          PUSH ADDRESS OF LAST ARG
...
PEA       ARG1          PUSH ADDRESS OF FIRST ARG
MOVE.L    ECBADR,A0     GET ADDRESS OF ECB
JSR       (A0)          JUMP AND PUSH PC
ADD.W     #4*n,SP       POP ARG PTRS OFF STACK
```

Subroutine entry:

```
MOVE.L    DB,-(SP)      SAVE CALLER'S DATABASE REG
CLR.L     -(SP)         PUSH A RESERVED WORD
MOVE.L    A0,-(SP)      PUSH ADDRESS OF MY ECB
MOVE.L    6(A0),DB      LOAD MY DATABASE FROM ECB
LINK      SB,#autosize  LOAD MY STACK BASE
...                     & DEFINE AUTOMATIC STORAGE
MOVE.L    20(SB),WORK   GET ADDR FIRST ARG
...
```

Subroutine return:

```
UNLK      SB            RELOAD CALLER'S SB
ADD.W     #8,SP         POP LINKAGE STUFF
MOVE.L    (SP)+,DB      RELOAD CALLER'S DB
RTS                     RETURN TO CALLER
```

See also STACK FRAME, ECB.

NOTE: Registers other than DB, SB, SP are not preserved.

CONDITION CODES*

Operations	X	N	Z	V	C	Special Definition
ABCD	*	U	?	U	?	C = Decimal Carry $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
ADD, ADDI, ADDQ	*	*	*	?	?	$V = Sm \cdot Dm \cdot \overline{Rm} + \overline{Sm} \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot Dm + \overline{Rm} \cdot Dm + Sm \cdot \overline{Rm}$
ADDX	*	*	?	?	?	$V = Sm \cdot Dm \cdot \overline{Rm} + \overline{Sm} \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot Dm + \overline{Rm} \cdot Dm + Sm \cdot \overline{Rm}$ $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
AND, ANDI, EOR, EORI, MOVEQ, MOVE, OR, ORI, CLR, EXT, NOT, TAS, TST	–	*	*	0	0	
CHK	–	*	U	U	U	
SUB, SUBI SUBQ	*	*	*	?	?	$V = \overline{Sm} \cdot Dm \cdot \overline{Rm} + Sm \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot \overline{Dm} + Rm \cdot \overline{Dm} + Sm \cdot Rm$
SUBX	*	*	?	?	?	$V = \overline{Sm} \cdot Dm \cdot \overline{Rm} + Sm \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot \overline{Dm} + Rm \cdot \overline{Dm} + Sm \cdot Rm$ $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
CMP, CMPI, CMPM	–	*	*	?	?	$V = \overline{Sm} \cdot Dm \cdot \overline{Rm} + Sm \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot \overline{Dm} + Rm \cdot \overline{Dm} + Sm \cdot Rm$
DIVS, DIVU	–	*	*	?	0	V = Division Overflow
MULS, MULU	–	*	*	0	0	
SBCD, NBCD	*	U	?	U	?	C = Decimal Borrow $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
NEG	*	*	*	?	?	V = Dm · Rm, C = Dm + Rm
NEGX	*	*	?	?	?	V = Dm · Rm, C = Dm + Rm $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
BTST, BCHG, BSET, BCLR	–	–	?	–	–	$Z = \overline{Dn}$
ASL	*	*	*	?	?	$V = Dm \cdot (\overline{D_{m-1} + \ldots + D_{m-r}}) + \overline{Dm} \cdot (D_{m-1} + \ldots + D_{m-r})$ $C = D_{m-r+1}$
ASL (r = 0)	–	*	*	0	0	
LSL, ROXL	*	*	*	0	?	$C = D_{m-r+1}$
LSR (r = 0)	–	*	*	0	0	
ROXL (r = 0)	–	*	*	0	?	C = X
ROL	–	*	*	0	?	$C = D_{m-r+1}$
ROL (r = 0)	–	*	*	0	0	
ASR, LSR, ROXR	*	*	*	0	?	$C = D_{r-1}$
ASR, LSR (r = 0)	–	*	*	0	0	
ROXR (r = 0)	–	*	*	0	?	C = X
ROR	–	*	*	0	?	$C = D_{r-1}$
ROR (r = 0)	–	*	*	0	0	

– Not effected
U Undefined
? Other – see Special Definition

* General Case:
X = C
N = Rm
$Z = \overline{Rm} \cdot \ldots \cdot \overline{R0}$

Sm – Source operand most significant bit
Dm – Destination operand most significant bit
Rm – Result bit most significant bit
n – bit number
r – shift amount

*Reprinted from MC68000, page A-4.

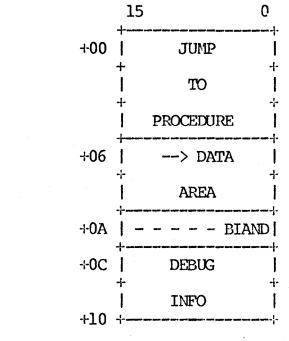
CONDITIONAL TESTS*

Mnemonic	Condition	Encoding	Test
T	true	0000	1
F	false	0001	0
HI	high	0010	$\overline{C} \cdot \overline{Z}$
LS	low or same	0011	$C + Z$
CC	carry clear	0100	$\overline{C}$
CS	carry set	0101	C
NE	not equal	0110	$\overline{Z}$
EQ	equal	0111	Z
VC	overflow clear	1000	$\overline{V}$
VS	overflow set	1001	V
PL	plus	1010	$\overline{N}$
MI	minus	1011	N
GE	greater or equal	1100	$N \cdot V + \overline{N} \cdot \overline{V}$
LT	less than	1101	$N \cdot \overline{V} + \overline{N} \cdot V$
GT	greater than	1110	$N \cdot V \cdot \overline{Z} + \overline{N} \cdot \overline{V} \cdot \overline{Z}$
LE	less or equal	1111	$Z + N \cdot \overline{V} + \overline{N} \cdot V$

*Reprinted from MC68000, page A-4.

ENTRY CONTROL BLOCK (ECB)

(type db_$ecb_rec in /us/ins/db.debug$.ins.pas)



B = 1 => Stop looking back in stack for inhbt routines
I = 1 => Inhibit async faults for routine
A = 1 => Apollo software
N = 1 => Debug info present
D = 1 => DB reg not saved

FILENAME SUFFIXES

SUFFIX	MEANING	RECOGNIZED BY
.ASM	Assembler source	Assembler (input)
.BAK	Backup file	Display Manager (output)
.BIN	Binary file	Compilers (output)
.BND	Binder	
.BS	Boot Shell command	Boot Shell
.C	C source	CC (input)
.DATA	Data file	
.FTN	FORTRAN source	FTN (input)
.HLP	Help text	HELP command (input)
.INS	Insert file	
.LST	Listing file	Compilers (output)
.MAP	Map file	Binder (output)
.PAS	Pascal source	PAS (input)
.RFC	Run file converter	RFC command (output), /sysboot (input)

D3M SUFFIX	MEANING
.DDL	Schema, subschema, aggregate schema DDL
.FMT	Output from the RDL
.LST	ASCII listing of the schema, subschema, aggregate schema compiler
.RPT	Output from the D3M/FORMATTER
.CMD	Executable D3M/DATAVIEW commands
.RDL	Source for report writer
.UWA.xxx	UWA definition generated by SSCH

AUX SUFFIX	MEANING
.C	C compiler
.O	Binary file from compiler
.H	C insert file

SCRIBE SUFFIX	MEANING
.MSS	Manuscript
.OTC	Outline
.LPT	Line printer
.AUX	Auxiliary
.ERR	Error listing

These conventions are not requirements; you can give a file any name you like, within the syntax rules. The operating system does not check a file's contents against its name. However, some programs assume that the names of input files end with a particular suffix. For example, the FORTRAN compiler requires that the names of its input files end in .FTN.

Symbol	Starting Point
//	Network root directory
/	Node entry directory
~	Naming directory
\	Parent directory
. or none	Working directory
`node_data	/sys/node_data[.nn]

Legal characters in names:

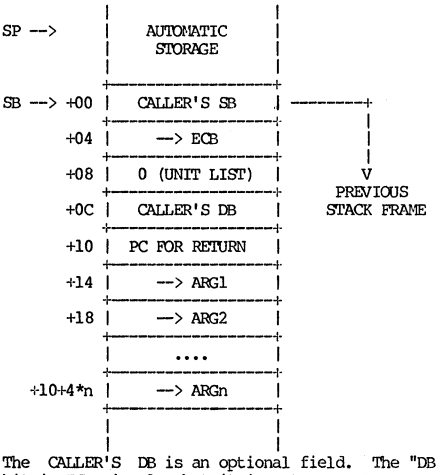
```
A-Z
a-z
0-9
$       (dollar sign)
_       (underscore)
.       (period)
```

Names cannot start with "_", ".", or digits.

Valid pathnames:

```
/PASCAL
\MISC/SAU_SOURCE
//US/INS/STREAMS.INS.FTN
~com
~link_name
```

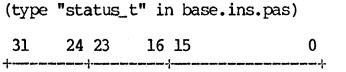
STACK FRAME



The CALLER'S DB is an optional field. The "DB not saved" bit in ECB signals that it is not present.

STATUS WORD

(type "status_t" in base.ins.pas)



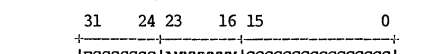
F = 1 => module couldn't handle error (fail bit)
S..S - Subsystem identification
A = 1 => asynchronous fault; only set during delivery of fault (.async)
M..M - Module identification
C..C - Module-specific error code

See Chapter 4, Error Codes and Messages.

CHAPTER 4

ERROR CODES AND MESSAGES

AEGIS ERROR CODES



F = 1 => module couldn't handle error
S..S - Subsystem identification
A = 1 => asynchronous fault; only set during delivery of fault
M..M - Module identification
C..C - Module-specific error code:

0 = OK status
negative = warning
positive = error

(type "status_$t" in base.ins.pas)

00000000 status_$ok

OS / BAT manager:
(00010001) attempt to free already-freed block
(00010002) disk is full
(00010003) attempt to free illegal disk address
(00010004) BAT not mounted
(00010005) disk needs salvaging

OS / VTOC manager:
(00020001) VTOC not mounted
(00020002) VTOC is bad
(00020003) no file map
(00020004) no UID
(00020005) not found
(00020006) UID not found
(00020007) duplicate UID
(00020008) uid mismatch

OS / AST manager:
(00030001) attempted reference to out-of-bounds address
(00030003) no replaceable aste's
(00030004) segment is not deactivatable
(00030005) write concurrency violation
(00030006) incompatible request
(00030007) reference count says unused
(00030008) segment not found in bst
(00030009) segment thread error in bst

OS / MST manager:
(00040001) object not found
(00040002) invalid length
(00040003) no space available
(00040004) reference to illegal address
(00040005) reference to out-of-bounds address
(00040006) no asid is available
(00040007) object is not mapped
(00040008) no rights
(00040009) insufficient rights
(0004000A) guard fault
(0004000B) wrong type - can't map system objects
(0004000C) ppn list overflow
(0004000D) uid mismatch
(0004000E) virtual memory resources exhausted
(0004000F) invalid va for install of io page
(00040010) invalid segment count

OS / PMAP manager:
(00050001) not allocated
(00050002) already allocated
(00050003) mismatch
(00050004) bad wire
(00050005) bad unwire
(00050006) bad assoc
(00050007) pages wired
(00050008) page null
(00050009) bad disk address
(0005000A) read concurrency violation
(0005000B) changed pmods
(0005000C) invalid pmape
(0005000D) attempt to map i/o page over real page
(0005000E) bst threads yielded invalid va

OS / MMAP manager:
(00060004) bad avail
(00060005) bad free
(00060006) bad unavail

OS / MMU manager:
(00070001) mmu miss
(00070002) va not in valid mmu manager range
(00070003) va does not have os_pmap

OS / disk manager:
(00080001) disk not ready
(00080002) disk controller busy
(00080003) disk controller time-out
(00080004) disk controller error
(00080005) disk equipment check
(00080006) floppy is not 2-sided
(00080007) disk write protected
(00080008) bad disk format
(00080009) disk data check
(0008000A) DMA overrun

(0008000B) volume in use
(0008000C) volume table full
(0008000D) volume not properly mounted or assigned
(0008000E) operation requires a physical volume
(0008000F) invalid volume index
(00080010) logical volume not found
(00080011) disk block header error
(00080012) invalid disk address
(00080013) disk buffer is not page aligned
(00080014) invalid logical volume index or list
(00080015) disk seek error
(00080016) drive timed out before operation completed
(00080017) bus error occurred during disk DMA transfer
(00080018) invalid unit number
(00080019) unknown status returned by hardware
(0008001A) invalid physical volume label
(0008001B) floppy door has been opened or storage module has been stopped
(0008001C) read after write failed
(0008001D) dma not at end of range
(0008001E) disk already mounted
(0008001F) software detected checksum error
(00080020) checksum error in read after write
(00080021) too many wired pages -- storage module manager
(00080022) disk driver logic error
(00080023) unknown error status from storage module controller
(00080024) unrecognized drive id
(00080025) memory parity error during disk read
(00080026) unrecognized interrupt from disktape controller

OS / eventcount manager:
(00090001) bad wait list on eventcount

OS / level 1 process manager:
(000A0001) illegal process id
(000A0002) illegal lock
(000A0003) process not suspended
(000A0004) process already suspended
(000A0005) process not bound
(000A0006) process already bound
(000A0007) bad atomic operation
(000A0008) no pcb is available
(000A0009) no stack space is available
(000A000A) process not suspendable

OS / terminal manager:
(000B0001) buffer too small
(000B0002) end of file entered from keyboard
(000B0003) invalid output length
(000B0004) invalid option passed to term_$control
(000B0005) input buffer overrun - characters lost
(000B0006) quit while waiting for input
(000B0007) invalid line number supplied

(000B0008) manual stop: type G<ret>G *+2<ret> to
continue
(000B0009) character framing error
(000B000A) character parity error
(000B000B) data carrier detect (dcd) changed
(000B000C) clear to send (cts) changed
(000B000D) requested line or operation not implemented
(000B000E) hangup fault

OS / DBUF manager:
(000C0001) bad ptr
(000C0002) bad free

OS / time manager:
(000D0001) no timer queue entry
(000D0002) entry to be cancelled not found
(000D0003) quit while waiting for event
(000D0004) bad timer interrupt
(000D0005) bad timer key
(000D0006) alarm fault

OS / naming server:
(000E0002) directory is full
(000E0003) name already exists
(000E0004) invalid pathname
(000E0005) invalid link
(000E0006) not a link
(000E0007) name not found
(000E000A) invalid link operation
(000E000B) invalid leaf
(000E000C) node is unavailable
(000E000D) bad directory
(000E000E) branch is not a directory
(000E000F) directory is not empty
(000E0010) name is not a file
(000E0011) illegal directory operation
(000E0012) bad type
(000E0013) no rights
(000E0014) insufficient rights
(000E0015) unable to delete system bootstrap (sysboot)
(000E0016) directory is in use (locked)

OS / file server:
(000F0001) object not found
(000F0002) object is remote
(000F0003) bad reply received from remote node
(000F0004) communications problem with remote node
(000F0005) object is not locked by this process
(000F0006) object is in use
(000F0007) illegal lock request
(000F0008) lock violation detected
(000F0009) local lock table is full
(000F000A) remote lock table is full
(000F000B) operation cannot be done from here
(000F000C) no more lock table entries
(000F000D) volume uid is unavailable

(000F000E) locking files is blocked for this volume
(000F000F) locking is already blocked for this volume
(000F0010) no rights
(000F0011) insufficient rights
(000F0012) wrong type - can't operate on system objects
(000F0013) objects are on different volumes

OS / I/O manager:
(00100001) dcte not found
(00100002) controller not in system

OS / network:
(00110004) transmit failed
(00110007) remote node failed to respond to request
(00110008) unable to route
(00110009) network hardware error
(0011000A) msg header too big
(0011000B) unexpected reply type
(0011000D) unknown request type
(0011000E) request denied by local node
(0011000F) request denied by remote node
(00110010) bad checksum
(00110011) too many transmit retries
(00110012) socket not open
(00110013) receive bus error
(00110014) transmit bus error
(00110015) bad asknode version number

OS / fault handler:
(00120001) odd address error
(00120002) illegal instruction
(00120003) integer divide by zero
(00120004) CHK instruction trapped - value out of range?
(00120005) arithmetic overflow
(00120006) privileged instruction violation
(00120007) invalid SVC code
(00120008) invalid SVC procedure name
(00120009) undefined TRAP instruction
(0012000A) unimplemented instruction
(0012000B) protection boundary violation
(0012000C) bus time-out
(0012000D) invalid user stack pointer
(0012000E) correctable memory error detected
(0012000F) uncorrectable memory error detected
(00120010) process quit
(00120011) access violation
(00120012) CPU B enabled with MMU valid bit reset
(00120013) null process running on CPU B
(00120014) OS-internal quit (with display return)
(00120015) single step completed
(00120016) invalid user-generated fault (subsystem code = 0)
(00120017) fault in user-space interrupt handler for pbu device
(00120018) process stop
(00120019) process BLAST

(0012001A) PEB cache parity error
(0012001B) PEB WCS parity error
(0012001C) unimplemented SVC
(0012001D) invalid stack format
(0012001E) memory parity error
(0012001F) process interrupt
(00120020) supervisor fault while resource lock(s) set
(00120021) spurious parity error
(00120022) floating point inexact result
(loss of significance)
(00120023) floating point divide by zero
(00120024) floating point underflow
(00120025) floating point operand error
(00120026) floating point overflow
(00120027) process suspend fault
(00120028) process suspend from keyboard
(00120029) process suspend due to background read
(0012002A) process suspend due to background write
(0012002B) process continue fault
(0012002C) fault(s) lost; process suspended or pfm_$enable
(0012002D) pfm_$inhibit mismatch

OS / display driver:
(00130001) invalid display unit number
(00130002) specified font not loaded
(00130003) internal font table full
(00130004) invalid use of display driver procedure
(00130005) font too large
(00130006) error unloading internal (hdmt) table
(00130007) invalid direction from SM
(00130008) unexpected BLT in use
(00130009) internal protocol violation
(0013000A) too many pages to be wired
(0013000B) unsupported font version #
(0013000C) invalid buffer size
(0013000D) error mapping display memory
(0013000E) error borrowing display from screen manager
(0013000F) unable to borrow - display in use
(00130010) display borrow request denied by screen
manager
(00130011) error returning display to screen manager
(00130012) can't return - display not borrowed
(00130013) can't borrow both displays simultaneously
(00130014) display already borrowed by this process
(00130015) invalid position argument
(00130016) invalid window limits argument
(00130017) invalid length argument
(00130018) invalid direction argument
(00130019) invalid scroll displacement argument
(0013001A) invalid blt mode register
(0013001B) invalid blt control register
(0013001C) invalid blt-done interrupt
(0013001D) invalid interrupt routine state
(0013001E) invalid screen coordinates in blt request
(0013001F) font associated with specified id not mapped
(00130020) display memory is already mapped

(00130021) display memory is not mapped
(00130022) quit while waiting
(00130023) invalid cursor number
(00130024) hidden display memory is full
(00130025) quit while waiting
(00130026) invalid eventcount key
(00130027) operation not implemented on color display
(00130028) non-conforming and main memory blts not implemented
(00130029) invalid DM window id
(0013002A) acquire denied because window is obscured
(0013002B) no more direct mode window ID's available
(0013002C) process not found
(0013002D) pad/stream operations not allowed while display acquired
(0013002E) display already acquired
(0013002F) display acquire timed out

OS / volume manager:
(0014FFFF) warning: disk is write protected
(00140001) entry directory problems on logical volume
(00140002) unable to dismount the boot volume
(00140003) logical volume is not mounted
(00140004) entry directory is not on specified logical volume
(00140005) physical volume replaced since mount

OS / calendar manager:
(00150001) invalid syntax for date or time specification
(00150002) date or time specification invalid
(00150003) an empty string was passed to a decode routine
(00150004) timezone specified is unknown
(00150005) invalid time-zone difference

OS / level 2 eventcount manager:
(00180001) internal table exhausted
(00180002) internal error
(00180003) process quit while waiting
(00180004) bad eventcount

OS / level 2 process manager:
(00190001) process not found
(00190002) not a level two process
(00190003) bad stack base
(00190004) request is for current process
(00190005) suspend request timed out
(00190006) process not suspended
(00190007) process already suspended
(00190008) child process terminated
(00190009) another fault is pending for this process
(0019000A) invalid process name
(0019000B) bad eventcount key
(0019000C) attempt to complete vfork on non-vforked process

OS / import/export manager:
(001A0001) entry directory is not catalogued in the namespace
(001A0002) files are locked on this volume
(001A0003) specified entry directory not on this volume
(001A0004) volume is not mounted

OS / startup/shutdown:
(001B0001) node ID mismatch
(001B0001) checksumming already enabled
(001B0001) no os paging file -- please run invol option 8

OS / vfmt:
(001C0001) unterminated control string
(001C0002) invalid control string
(001C0003) too few arguments supplied for read/decode
(001C0004) field width missing on "(" designator
(001C0005) encountered end of string where more text expected
(001C0006) encountered null token where numeric token expected
(001C0007) non-numeric character found where numeric was expected
(001C0008) sign encountered in unsigned field
(001C0009) value out of range in text string
(001C000A) character in text string does not match control string
(001C000B) terminator in text string does not match specified terminator

OS / circular buffer manager:
(001D0001) invalid block size requested
(001D0002) quit while waiting
(001D0003) buffer wrap-around error

OS / pbu manager:
(001E0001) ddf is larger than one page
(001E0002) ddf has wrong version
(001E0003) invalid unit number in ddf
(001E0004) invalid csr page address in ddf
(001E0005) csr page is in use
(001E0006) initialization routine not in library
(001E0007) cleanup routine not in library
(001E0008) interrupt library too large
(001E0009) interrupt routine not in library
(001E000A) pbu not present
(001E000B) too many pbu manager pages wired
(001E000C) invalid unit number
(001E000D) unit in use
(001E000E) unit not acquired
(001E000F) unit already acquired
(001E0010) bad parameter
(001E0011) no room in iomap
(001E0012) requested iomap in use
(001E0013) iomap already allocated
(001E0014) iomap not allocated

(001E0015) invalid iova
(001E0016) buffer too large
(001E0017) buffer page not wired
(001E0018) buffer not mapped
(001E0019) page already wired
(001E001A) page wired too many times
(001E001B) page not wired
(001E001C) reference to csr page caused bus timeout
(001E001D) trap 6 executed outside of interrupt routine
(001E001E) invalid trap 6 code
(001E001F) invalid usp at trap 6
(001E0020) invalid argument at trap 6
(001E0021) unexpected interrupt from pbu device
(001E0022) ddf has wrong file type
(001E0023) too many wired pages
(001E0024) csr not in device's csr page
(001E0025) controller already mapped
(001E0026) bad controller memory length
(001E0027) bad buffer address
(001E0028) interrupt library not found
(001E0029) device library not found
(001E002A) device is not a shared controller
(001E002B) device not mapped

OS / line printer module:
(001F0001) pna board not installed in system
(001F0002) invalid string length
(001F0003) invalid string termination
(001F0004) line printer not acquired
(001F0005) line printer already acquired
(001F0006) internal error
(001F0007) ppn list overflow - internal error
(001F0008) line printer not assigned
(001F0009) no line printer on system

OS / OS info supplier:
(00200001) array too small for complete table

OS / badspot manager:
(00210001) bad checksum in physical badspot block
(00210002) bad count in physical badspot block
(00210003) missing minus-one in physical badspot block
(00210004) badspot list too small
(00210005) no physical badspot blocks read or written
(00210006) physical badspot list partially read or written
(00210007) duplicate entry in badspot list
(00210008) no physical badspot information on disk
(00210009) bad daddr for lv label badspot extension block
(0021000A) too many extensions to lv badspot list

OS / magtape manager:
(0022FFFE) warning: tape not at load-point
(0022FFFF) warning: tape unit is offline
(00220001) invalid mt unit number

(00220002) invalid mode field
(00220003) invalid buffer length
(00220004) invalid parameter
(00220005) no PNA board installed in system
(00220006) magtape unit is not connected
(00220007) magtape not acquired
(00220008) magtape unit is not ready
(00220009) unit will not fit thru 25" hatch
(0022000A) magtape unit in use
(0022000B) magtape not initialized
(0022000C) magtape already acquired
(0022000D) invalid option
(0022000E) too many outstanding operations
(0022000F) invalid buffer address
(00220010) invalid count for erase or space operation
(00220011) tape drive is hung
(00220012) ppn list overflow - internal error
(00220013) config page in use - internal error
(00220014) release problems - internal error
(00220015) unexpected interrupt
(00220016) operation attempted before waiting
(00220017) wait attempted before go issued
(00220018) go command issued while not in batch mode
(00220019) header or buffer misalignment on chained r/w
(0022001A) user quit while in mt_$wait
(0022001B) timeout during wait or release
(0022001C) header buffer not on header page
(0022001D) no room from mt_$write - internal error
(0022001E) info array (passed to mt_$wait) too small
(0022001F) too many pages wired
(00220020) too many pbu devices in use
(00220021) buffer already wired
(00220022) buffer not wired

OS / ACL manager:
(00230001) no right to perform operation
(00230002) insufficient rights to perform operation
(00230003) exit_super called more often than enter_super
(00230004) wrong type - operation illegal on system objects
(00230005) entry already exists
(00230006) ACL is remote
(00230007) ACL is on different volume than object
(00230008) ACL protects wrong type of object
(00230009) insufficient address space to open ACL
(0023000A) entry was matched by a wild card
(0023000B) no entry - entry number too large
(0023000C) image buffer too small or incorrect size
(0023000D) ACL object not found
(0023000E) ACL would be unchangeable
(0023000F) object may not be readable by backup procedure
(00230010) no right to set subsystem data or subsystem manager

OS / PEB manager:
(00240001) fpu is hung
(00240002) PEB interrupt
(00240003) floating point overflow
(00240004) floating point underflow
(00240005) divide by zero
(00240006) floating point loss of significance
(00240007) floating point hardware error
(00240008) attempted use of unimplemented opcode
(00240009) wcs verify failed

OS / network logging manager:
(00250001) ppn list overflow

OS / color display manager:
(00260001) illegal caller
(00260002) too many wired pages
(00260003) virtual address not page aligned in color_$map
(00260004) pages unmapped out of order
(00260005) parameter value out of range
(00260006) color display not available
(00260007) instruction queue done wait timed out

OS / vme bus manager:
(00270001) undefined vme interrupt
(00270002) vme bus error

OS / cartridge tape manager:
(0028FFFC) warning: tape not at load-point
(0028FFFD) tape at load point
(0028FFFE) warning: tape unit is offline
(0028FFFF) tape power on/reset
(00280001) invalid ct unit number
(00280002) unit not acquired
(00280003) unit already acquired
(00280004) unit in use
(00280005) no tape controller on system
(00280006) invalid buffer length
(00280007) bad buffer alignment
(00280008) invalid buffer address
(00280009) unrecognized action type
(0028000A) invalid operation count
(0028000B) unit not ready
(0028000C) unexpected ct interrupt
(0028000D) quit during read or write
(0028000E) drive timeout
(0028000F) too many wired pages
(00280010) no cartridge in drive
(00280011) drive does not exist
(00280012) tape is write protected
(00280013) end of tape
(00280014) read/write abort
(00280015) read block error
(00280016) read filler error
(00280017) read no data

(00280018) read no data and end of tape
(00280019) read no data and load point
(0028001A) filemark detected
(0028001B) illegal drive command
(0028001C) marginal block detected
(0028001D) unrecognized drive status
(0028001E) dma not at end of range
(0028001F) dma underrun/overrun
(00280020) memory parity error during dma
(00280021) illegal controller command
(00280022) controller timeout
(00280023) controller diagnostic failed
(00280024) unrecognized controller status
(00280025) operation already in progress
(00280026) operation not in progress

BOOT ERRORS (PROM)

error: boot not found - The SYSBOOT read from records 2 through B did not have a good boot header.

disk init error <SC> <RCD> <UNIT> <W/F/S/C>
disk read error <SC> <RCD> <UNIT> <W/F/S/C>

SC = Status Code
RCD = Record Address
Unit = Disk Unit No.
W/F/S/C = Winchester/Floppy/SMD/Cartridge tape

Winchester Status Codes

1 - not responding
2 - not ready
11 - seek not complete
12 - CRC, timeout, buserr, overrun
13 - drive faults

Floppy Status Codes

1 - wrong no. of status bytes
2 - seek not complete
3 - equipment check
11 - insufficient status
12 - seek not complete
13 - equipment check
14 - bad seek
15 - insufficient status
16 - abnormal termination
17 - " "
18 - device not ready
19 - CRC error

DN550 Disk/Tape Status Codes:

11 - controller diagnostic failed
12 - controller timed out
13 - illegal controller command
15 - memory parity during dma
16 - dma overrun/underrun
17 - dma not at end of range
1E - disk still busy
1F - controller still busy
21 - seek did not complete
22 - write fault
23 - unit not present
24 - sector not found
25 - no index pulse
26 - drive not ready
27 - no track 0 on restore
28 - address mark not found
29 - ECC error in sector ID field

30 - illegal tape command
31 - filemark encountered
32 - read error - no data and BOM
33 - read error - no data and EOM
34 - read error - no data
35 - read error - filler block transfer
36 - read error - bad block transferred
37 - read or write abort
38 - end of media
39 - drive not present
3A - no cartridge in drive

FF - timeout waiting for controller done

error:
<test no.><detected at><object addr><data is><data sb>

Steady State

0	first instruction at "init"
1	memory passed tests at init
2	state saved
3	parity cleared in first 2 pages
4	sio's have been initialized
5	clr_disp called, returned
6	disp_init called, returned
7	display init got bus error
8	we're in service mode
9	character received from keyboard
A	character received from line 1
B	character received from line 2
C	just printed MD's banner msg
D	ptt enabled (map routine)
E	mmu initialized (map routine)
F	mmu enabled (map routine)

Digit 1	Digit 2	
0	———	Prom wait conditions
	2	boot: waiting for disk i/o to complete
	3	boot: waiting for network transmit to complete
	4	boot: waiting for network volunteer response
	5	boot: waiting for network receive to complete
	C	waiting for command input (service mode only)
1	———	DIAGNOSTIC 1: validate prom checksum
	1	prom checksum did not match calculated checksum
2	———	DIAGNOSTIC 2: test PFT
	1	verify failed with data = 0000... (first time)
	2	verify failed with data = FFFF...
	3	verify failed with data = AAAA...
	4	verify failed with data = 0000... (second time)
	5	verify failed for address uniqueness
3	———	DIAGNOSTIC 3: test PTT
	1	verify failed with data = 0000... (first time)
	2	verify failed with data = FFFF...
	3	verify failed with data = AAAA...
	4	verify failed with data = 0000... (second time)
	5	verify failed for address uniqueness
4	———	DIAGNOSTIC 4: test IOMAP
	1	verify failed with data = 0000... (first time)
	2	verify failed with data = FFFF...
	3	verify failed with data = AAAA...
	4	verify failed with data = 5555...
	5	verify failed with data = 0000... (second time)
	6	verify failed for address uniqueness
5	———	DIAGNOSTIC 5: test PFT, PTT IOMAP interaction
	1	writing PFT affected PTT (should be 0's and wasn't)
	2	writing IOMAP affected PTT (should be 0's and wasn't)
	3	writing PFT affected PTT (should be 1's and wasn't)

```
    4       writing IOMAP affected PIT (should be 1's and wasn't)
6 ------ DIAGNOSTIC 6: test physical/virtual memory
    1       verify failed during byte test with 0's
    2       verify failed during word test with 1's
    3       verify failed during long word test with 5's
    4       verify failed during long word test with A's
    5       verify failed during long word test with addresses
7 ------ DIAGNOSTIC 7: test virtual memory
    1       verify failed during long word test with addresses
8 ------ DIAGNOSTIC 8: retest memory in physical mode
    1       verify failed during long word test with addresses
9 ------ DIAGNOSTIC 9: timers, MULTIBUS, calendar
    1       timer  1 failed
    2       timer  2 failed
    3       timer  3 failed
    4       multibus map test failed pattern test
    5       Calendar hardware clock not incrementing seconds
    6       DMA continuity test for channel  0 failed
    7       DMA continuity test for channel  1 failed
    8       DMA continuity test for channel  2 failed
    9       DMA continuity test for channel  3 failed
    A       DMA test with all channels activated failed
4 ------
    7       Multibus Loopback Test failed, data non-compare
    8       Multibus Loopback Test failed, no Multibus timeout
9 ------ DIAGNOSTIC 10: test ring board
    B       ring board loopback test failed
4 ------ DIAGNOSTIC 11: test VME interface
    9       VME Register Test failed
    A       Open VME Bus Test failed
    B       Bus Arbiter Logic Test failed
    C       VME-to-BPORT Logic Test failed
A ------ DISK ERROR
    1       not responding
    2       not ready
    B       seek not complete
    C       CRC, timeout, buserr, overrun
    D       drive faults
B ------ NETWORK ERROR
C ------ SYSBOOT ERRORS
    1       bad command line (cmd not found, missing filename)
    2       unable to read pv label
    3       volume "n" not found
    4       unable to read lv label
    5       salvaging boot volume
    6       auto salvage failed
    7       unable to read root directory
    8       SAUn not found in root directory
    9       SAUn uid not found
    A       unable to restore SAUn directory
    B       <program> not found
    C       <program> uid not found
    D       <program> unreadable
    E       <program> not a file
    F       <program> has wrong machine type
D ------ NETBOOT ERRORS
```

```
    1       bad command line (cmd not found, missing filename)
    2       ring initialization failed
    3       unable to send load request
    4       no response to load request
    5       unexpected packet (get_reply)
    6       <program> not found
    7       bad pathname
    8       insufficient rights
    9       <program> has wrong machine type
    A       (other) non-zero status from netman
    B       unable to send uid request
    C       no response to uid request
E ------ RESERVED
F ------ AEGIS CRASH
    1       aegis crash
    F       normal shutdown complete
```

MNEMONIC DEBUGGER ERROR CODES (PROM)

Printed on system crash or other entry to MD. See also System Crash Analysis under Operational Procedures. For DNx60 CPIO, mnemomic debugger codes are printed just like non-DNx60, except the qualifier "(CPIO)" is also printed. For DNx60 CPU, the qualifier "(CPU)" is also printed to identify the environments.

```
A  <PC> <SR> <IR> <FA> <FC>   -  Address Error
   <PC> <Contents>
B  <PC> <SR> <IR> <FA> <FC>   -  Bus Error
   <PC> <Contents>
E                             -  Operational Error
F  <PC> <SR>                  -  Invalid stack format
   <PC> <Contents>
I  <PC> <SR>                  -  Unexpected Interrupt
   <PC> <Contents>
J                             -  Spurious Interrupt
O  <PC> <SR> <FW> <FA>        -  floating point trap
   <PC> <Contents>               (DNx60 CPU)
P                             -  Parity error (DN300 only)
S  <PC> <SR>                  -  Trap instruction or
   <PC> <Contents>               breakpoint
T  <PC> <SR>                  -  Trace trap
   <PC> <contents>
U  <PC> <SR>                  -  Unimp inst trap
   <PC> <Contents>
V  <PC> <SR> <FW> <FA>        -  access violation (DNx60 CPU)
   <PC> <Contents>
W  <PC> <SR> <FW> <FA>        -  region fault (DNx60 CPU)
   <PC> <Contents>
X  <PC> <SR> <FW> <FA>        -  segment fault (DNx60 CPU)
   <PC> <Contents>
Y  <PC> <SR> <FW> <FA>        -  page fault (DNx60 CPU)
   <PC> <Contents>
```

```
Z  <PC> <SR>                    -  Divide by zero trap
   <PC> <Contents>

     PC = Program Counter
     SR = Status Register
     IR = Instruction Register
     FA = Fault Address
     FC = Fault Code
     FW = Format Word identifies the trap vector

            System    User
          +--------+--------+
     SR:  |T-S--III|--XNZOC|
          +--------+--------+
           T -   Trace Mode        X - Extend
           S -   Supervisor Mode   N - Negative
           III - Interrupt Mask    Z - Zero
                 000 - enabled     O - Overflow
                 111 - disabled    C - Carry

           15             0
          +---------------+
     FC:  |----------RNFFF|
          +---------------+
           R   = 1 => Read operation
           N   = 1 => Not an instruction reference
           FFF - Function code:
                 001  User data
                 010  User program
                 101  Supervisor data
                 110  Supervisor program
                 111  Interrupt acknowledge
```

SYSBOOT ERROR CODES

```
boot error: unable to read pv_label
     ''     volume "N" not found
     ''     unable to read lv_label
     ''       ''  ''  ''  root_dir
     ''     SAU not found in root_dir
     ''     SAU uid not found
     ''     unable to restore SAU_dir
     ''     "FILENAME" not found
     ''        ''      uid not found
     ''        ''      unreadable
     ''        ''      not a file
     ''     missing file name
```

CHAPTER 5

SYSTEM DEBUGGING

BOOT SHELL COMMANDS

CF	[<pathname> | -E]	run/end command file
CHN	<pathname> <compname>	change name
CRD	<pathname>	create directory
CRF	<pathname>	create file
CRL	<pathname> <linkname>	add link
CTNODE	<leaf> <node_id>	add node to local copy of root
CTOB	<pathname> <uidhi> <uidlo>	catalog name with specified UID
DEBUG	<value>	enable/disable debug mode
DLF	<pathname>	delete file
DLL	<linkname>	drop linkname
DM		invoke the display manager
DMTVOL	{W | S | F} <lvno> [<pathname>]	dismount a logical volume
GLOB		list installed globals
GO		load as if in normal mode
H		prints this text
IN	<pathname> [-D] [-S | -NS]	invoke loader to install named file
LD	[<pathname>] [-A [-D]] [-U]	list directory
LI	<address>	set display lights address
LO	<pathname> [-D] [-S | -NS]	invoke the loader with the given file
MA	<pathname> [<l> <sz>] [-E]	map file
MTVOL	{W | S | F} <lvno> [<pathname>]	mount a logical volume
ND	<pathname>	set naming directory
REL	[-A]	release proc-mgr assigned storage
SH		load single-process shell
SHUT		shut down system
SPM		load DSP80 server-process manager
STCODE	<status-code>	print text definition of status code
TB		stack trace back
TI	{-ON | -OFF}	enable/disable timer
TR	<pathname> <sz>	truncate raw data file to given size (hex)
UCTNODE	<leaf>	drop node from local copy of root
UCTOB	<pathname>	uncatalog pathname from namespace
UMA	{<pathname> | <l> <sz>}	unmap file by name or address size
WD	<pathname>	set working directory

Key:

<l> := low address
<h> := high address
<s> := start address
<sz> := size
<type> := { nil, rec, hdru, obj, dev, pad, undef, uasc, mt, boot }

DEBUG COMMAND EXTENSIONS

The following commands and other items are available in the standard DEBUG, but are not advertised to the public.

New DEBUG Commands

- FPREGS — Display the floating-point registers of DNx60s.
- REGS — Display the registers of the target program.
- DB — Invoke the machine-level debugger DB (see below).

New Options to DEBUG Commands

- CDB — This option is useful to debug C code in the C library. It prevents DEBUG from running to the "main" program when DEBUG is invoked.
- DDD — This option is useful only when debugging DEBUG from another DEBUG. It must be given to the target DEBUG only. When it is given, the target DEBUG will return from the fault handler with the "continue fault handling" state, so that the master DEBUG can catch breakpoints and single steps.

Miscellaneous

More help is available on DEBUG commands in this file:

```
//us/latest/sysx/help/debug.hlp
```

Here is some useful information that IS PUBLIC, which people often forget how to do.

There are three debugger names that DEBUG looks for. If it finds them, it does some special things:

```
`cr             (macro name; made with MACRO command)
`max_array_dim (debugger variable; made with SET command)
`max_var_len   (debugger variable; made with SET command)
```

Note that "help examine" explains the `max_... names, and "help macro" explains the `cr name.

DB (MACHINE LEVEL DEBUGGER)

You enter the DB debugger by entering the DB command from within a shell. The formats of the internal DB commands follow.

```
dl                      invoke emt
ef                      enter fim
fp[.s|.d} [< >]         show any or all floating pt regs (0-f)
fpc [<value>]           set/show the floating pt control reg
fps [<value>]           set/show the floating pt status reg
help
in <path>               install library
lo <path>               load program
ma <path> [-ex] [<start>]    map a file (starting location)
pc                      display current pc
fa                      display last fault address
fc                      display last fault code
sh                      invoke new shell
ss                      single step
tb                      traceback current stack
```

crash analysis commands:

```
a7 <value>              set a7 in dump (dflt is from E003FC)
a{b|w|l}[e] <sym>       access via symbol name
am <path>               load Aegis Map
as [<asid>]             set/display current asid
aste <addr>|<astex>     print contents of aste
d460                    info unique to dn460/660 dumps
da [<clockh>]           display date (build_$time)
dcte [<index>]          display dcte (all if no index)
df <addr>               display diagnostic fault frame at addr
dp [<pid>]              display pcb (first ten if no pid entered)
dpt                     disable ptt (remove from address space)
dr                      display registers at crash
ds                      display disk statistics
dv <addr>               convert db address to virtual address
dvt                     print disk volume table
ept                     enable ptt (map into address space)
ff [<addr>]             try to find stack frame in addr - addr+1024
gd <unit>               get (pbu) dcte
ha <hi> <lo> | <addr> hash uid to ast
le                      list system error log
lvl <addr>              print lv label at addr
m                       enter mapped mode
mm <addr>|<ppn>         print mmap entry
mr                      print mem_rec
ms <args>               mapped search (just like md's 's')
mst [<asid>|<msteaddr>] print mst for an asid (0 for glb,
                        omit for curr)
mste <addr>             print mste for a virtual address
                        in current asid
p                       enter physical (normal) mode
```

```
pf <ppn>|<addr>     display pft entry
pt <pttx>           display ptt entry
pv <addr>           convert ppn to virtual address
pvl <addr>          print PV label at addr
r                   enter real mode
rl [anything]       display ready list [just check order]
st                  display status at crash
ts <pid or addr>    traceback stack
uid <hi> <lo> | <addr> interpret uid
vd <addr>           convert virtual address to db address
ve <addr>           print vtoce at <addr>
vm                  verify mmu
vp <addr>           convert virtual address to ppn
vv <addr> <data>    verify vmtest page
wh[p|d|e] <sym or addr> look up [proc|data|ecb] or address
                    in aegis map
```

Execute the DB debugger by entering the q command in response to the ! prompt as follows:

```
! q
```

Lights Program

Execute the Lights program to show network status from within the DB debugger as follows:

```
!LI 0FF9C12 (in DN400/420) {virtual address of
receive register}
!LI 0FF9C02 (in DN300/320)
or
!LI OFFF9C12 (in DN460)
```

Do not execute the Lights program from a color node! It will crash the node!

The transmit and receive status registers are displayed at the bottom of the screen. These registers are described in chapters 7, 8, and 9. To use the Lights program do the following:

- Exit from DB with the q command.
- Execute NETSTAT shell commands to send tokens across the network.
- Study the status bits in the receive status register.

To exit from the Lights program, execute the following commands:

```
$ DB        {to re-enter the DB environment}
! LI 0      {Provide an address of zero to the Lights program}
! q         {exit from the DB environment}
<ctrl> F    {remove Lights from the bottom of the screen}
```

Commands:

A <location>	Access location
B <location>	Breakpoint
C <start> <end> <target>	Copy Memory
CA <start>	CALL Subroutine
D <start> <end> <items/line>	Dump Memory
DI <type><unit> <log vol>	Define Disk
DL	Down-line Loader
EX <filename>	Load and Execute File
EY <filename>	Load and Execute File with trap after load
F <start> <end> <word>	Fill Memory
G <location>	Jump to Location
LD	Lists SAUn Directories
LO <filename>	Load File
M	Map Address Space
RE	Reset System
S <start> <end> <value> <mask>	Search Memory
SH <0-3>	Spindown Winchester
P	Unmap Address Space
V <start> <end> <target>	Verify Memory

A [<size_spec>] <location>[<base_spec>]

Accesses <location> and prints the address and contents according to <size_spec> and <base_spec>.

B [<location>]

Sets/clears the breakpoint at the location specified. Breakpoint is not inserted until the G command is given. Previous instruction is reinstalled on the breakpoint entry or vector entry.

C <start> <end> <target>

Copies memory defined by the bounds <start> to <end> onto memory starting at <target> through <target>+<end>-<start>.

CA <start>

Calls the subroutine that starts at <start>. All registers saved from the last entry except A0 are restored immediately prior to the call.

D [<size_spec>] <start> <end>

<items_per_line>[<base_spec>]

Dumps memory defined by the bounds <start> to <end> onto the terminal printing address followed by specified <items_per_line>. The default is one per line. The <size_spec> controls the item-size to be dumped: byte, word, long, instruction.

DI <W>|<F>|<N> [<nn>]|<S> <0-3> <1-10>

Disk defines the boot device: Winchester, Floppy, node (nn), storage module unit 0-3, and logical volume 1-10. Defaults are: W, 0, 1.

DL

Transfers control to the down-line loader.

EX <filename>

Execute restores the named file from the "SAU" directory of the boot device and transfers control to it. After the restore is completed, the LOW, HIGH, and START addresses are displayed.

EY <filename>

Just like EX except:

- Just before passing control to the program, the MD will trap, and you can patch the program. Type "G", "G*+2" to continue.
- If you are executing AEGIS, AEGIS will trap again after establishing the OS mapping and before calling OS_$INIT. AEGIS can then be patched or examined using the virtual addresses from AEGIS.MAP in the appropriate SAU directory. Type "G", "G*+2" to continue bringing up AEGIS.

NOTE: Older boot PROMs may not support the EY command.

F <start> <end> [<word>]

Fills memory defined by the bounds <start> to <end> with a word value <word>.

G [<location>]

Jumps to <location> after inserting a breakpoint (if any), restoring all registers and SR.

LD

List directory displays the contents of the "SAU" directory of the boot device.

LO <filename>

Load restores the named file from the "SAU" directory of the boot device. The LOW, HIGH, and START addresses are displayed.

M

Maps address space and enables MMU. Memory is rearranged as shown under Address Space in Chapters 7 through 11.

RE

Reset executes the RESET instruction. If entered while running on CPU B, a second Reset instruction is executed for CPU A. The debugger will initialize and wait for terminal input. This command also enables the POWER-OFF key.

S [<size_spec>] <start> <end> <value> [<mask>][<base_spec>]

Searches memory defined by the bounds <start> to <end> for <value> through an optional <mask>. If <mask> is not specified it defaults to $FFFFFFFF. The <size_spec> controls the item-size to be searched: byte, word, or long.

SH <0-3>

Shuts down the Winchester unit and acknowledges outstanding interrupts. This command also enables the POWER-OFF key.

P

Turns off the mapping. MMU is assumed at FFB400.

V [<size_spec>] <start> <end> <target>[<base_spec>]

Verifies equality of two memory areas defined by <start> to <end> and <target> to <target>+<end>-<start>. If a discrepancy is found, the address in the first area and the contents of each are printed in the appropriate format.

MNEMONIC DEBUGGER COMMANDS FOR THE DNx60

Commands valid in CPU only:

DC	Enable/disable/show data cache
GB	Go Back to CPIO environment, halt CPU
FP	Access Floating Point Registers

Commands valid in CPIO only:

Those marked with an asterisk require the micro exec to be loaded and/or running.

UC	Clock step CPU
*US	Micro step <cnt <uaddr>>
*UT	Micro trace <cnt <uaddr>>
UH	Halt the CPU and CPIO becomes master
UR	Run micro code <from uaddr>
UX	Reset micro machine
*UU	Micro trap to the micro executive
*UI	IPL micro machine at uaddr 0
UP	Display or set micro pc
UA	Load microcode file set
UL	Load one microcode file by name
UF	Set or clear microcode loaded flag
*MG	Start the CPU <at macro addr> and CPIO becomes slave
*MR	Start the CPU <at macro addr> and CPIO dies
*MS	Macro step the CPU <cnt <addr>>
*MT	Macro trace the CPU <cnt <addr>>
MX	Load and execute program using CPIO
GF	Go Forward to CPU with reset exception, halt CPIO
FR	Fill wcs with freeze micro instructions
*LP	Set loop mode for uexec commands
MM	Set CPIO in master mode
SM	Set CPIO in slave mode
DM	Set CPIO in dead mode
MO	Display CPIO master/slave mode
DS	Dump CPU state

Commands valid in both environments:

AS	Display current asid
B	Set or display breakpoint(s) in memory
CB	Clear breakpoint(s)
DR	Dump registers
EX	Load and Execute program using CPU
EY	Like EX except trap first
H	Help. List MD commands
IC	Enable/disable/show instruction cache
PV	Convert physical addr to virtual
RR	Access Region Registers
SS	Single step a program
VP	Convert virtual addr to physical
XE	XON/XOFF enable
XD	XON/XOFF disable

B [<location [number]>]

Sets a breakpoint at the location specified. You can set up to four breakpoints in the DNx60 MD.

CB [<location>]

Clears the breakpoint at <location>, or all the breakpoints set, if you omit the argument. An error message appears if you attempt to clear breakpoints that are not set.

DS

Dump the CPU state. Valid only from the CPIO environment. It is useful if micromachine freezes (UPCxxx).

EX <filename>

Loads the named file from the "SAUn" directory of the disk device used (specified by the DI command; is a Winchester disk if no DI command is given).

EY <filename>

Just like EX except:

- Just before passing control to the program, the MD will trap, and you can patch the program. Type "G", "G*+2" to continue.
- If you are executing AEGIS, AEGIS will trap again after establishing the OS mapping and before calling OS_$INIT. AEGIS can then be patched or examined using the virtual addresses from AEGIS.MAP in the appropriate SAU directory. Type "G", "G*+2" to continue bringing up AEGIS.

NOTE: Older boot PROMs may not support the EY command.

GB

Go back (return control) to the boot processor. This is valid only in the main processor environment.

GF

GF start the main processor with a reset exception. You must have loaded microcode before you execute GF.

LP

LP sets the loop bit on any microexecutive command issued until the machine is reset. With the loop bit set, the commands operate in loop mode.

MG [<addr>]

MG starts the main processor at the address you specify, or at the current main processor program counter (PC), if you do not give an argument. The boot processor goes into slave mode.

MR [<addr>]

Starts the main processor at the address you specify, or at the current main processor program counter (PC), if you do not give an argument. The boot processor halts.

MS [<Count> [<addr>]]

Macro steps once or for the number of counts, showing the program counter at the end of the steps. A macrostep executes one instruction from memory on the main processor.

MT [<Count> [<addr>]]

Macro traces for the number of counts you specify, or until you hit a key. Prints the program counter at each step.

MX <filename>

Loads and executes the program <filename> on the boot processor. <filename> is a program in the SAU4 directory on the volume defined as the disk device by the DI command.

PV <PA> <addr>

Converts the physical address to all virtual addresses by searching the page tables starting at current region registers or use region registers at ADDR. PA must be long-word aligned.

RR <region register number (0-31, decimal, 0-1F hex>

Accesses the region register you specify in the argument, and displays its contents.

UA

Loads the microcode loading program (ULOAD) and the entire set of microcode files needed to boot the operating system.

UC [<count>]

Executes a clock cycle, or the number of cycles you specify in the <count> field, on the main processor. The

boot processor enters slave mode.

UF [-OFF]

UF sets or clears the "microcode loaded" flag in the PROM. When the flag is set, the PROM does not load microcode when enabling the main processor.

UH

Halts (disables) the main processor and puts the boot processor into master mode. The boot processor now has bus mastership and control of the system.

UI

Resets the main processor and runs it from microlocation zero. This command starts the micro executive (if it is loaded).

UL <filename> -N

Reads one microcode data file from the SAU4 directory and the micro-loading program (ULOAD) from the boot device, and loads the microcode. By default, if the microcode file is a WCS file, UL fills the WCS with freezes. Use the -N option when you are loading two WCS files separately.

UP <uaddr>

Displays or sets the microcode program counter. If the main processor is running, you cannot read or set the microcode program counter.

UR [<uaddr>]

Runs the microcode on the main processor, starting at the micro address you specify. The boot processor enters slave mode. If you do not specify an address, UR uses the current microcode program counter.

US [<count> [<uaddr>]]

Micro steps once or for the number of counts you give, showing the micro program counter at the end of the steps. If you specify <uaddr>, the program starts at the micro address you specify.

UT [<count> [<uaddr>]]

Micro traces for the number of counts you specify or until you hit a key. The micro program counter is displayed at each step. If you specify a micro address, the trace starts at the micro address you specify.

UU

Causes a micro trap in the main processor, and displays the microcode program counter. You can use this command to stop the main processor.

UX

Resets the main processor and sets the micro program counter to zero.

VP <VA> <addr>

Convert virtual address to physical address using current region registers at ADDR. VA must be long-word aligned.

XD

Disables the X-On protocol used to communicate with dumb terminals.

XE

Enables the X-On protocol used to communicate with dumb terminals.

Command Formats

<command>[<size_spec>] [<parameter_list>][<base_spec>]

<command> ::= A|B|C|D|DL|F|G|S|V|<empty>

<size_spec> ::= :I|:B|:W|:L

<parameter_list> ::= <parameter> ... [up to four]

<parameter> ::= <num_exp>|Dn|An|CCR|SR|
(An)|<num_exp>(An)|<num_exp>(<index_spec>)|
<num_exp>(An,<index_spec>)

<num_exp> ::= <num>|*|<num_exp>+<num>|<num_exp>-<num>

<num> ::= <simple_number|$<simple_number>|
<base>$<simple_number>|-<num>|<quoted_string>

<base> ::= <simple_number>

<quoted_string> ::= '<letter> ... <letter>' [up to four]

<index_spec> ::= An.W|Dn.W|An.L|Dn.L

<base_spec> ::= :O|:D|:H|:A

Semantics

:I ::= instr-sized items, output in mnemonic format.

:B ::= byte-sized items, output in numeric format.

:W ::= word-sized items, output in numeric format.

:L ::= longword-sized items, output in numeric format.

Parameters are evaluated to a memory location or to an MD saved register, [e.g., Dn, An] or to a location computed from a saved register [num(An)]. Up to four parameters may be required. Unspecified parameters are set to zero.

:O ::= numbers and immediate constants printed in octal.

:D ::= numbers and immediate constants printed in decimal.

:H ::= numbers and immediate constants printed in hexadecimal.

:A ::= numbers and immediate constants printed in ASCII.

All numeric input defaults to hexadecimal. $num implies hexadecimal. <base>$num implies that base is <base> [8$777 is octal, 2$1001 is binary]. <base_spec> and <size_spec> may be specified anywhere in the command line as well as anywhere in A command input (except in quoted strings). All addresses and offsets are printed in

hexadecimal regardless of <base_spec>.

CRASH ANALYSIS

Most fatal errors recognized by AEGIS are reported by the crash_system routine, which prints a status code (see Chapter 4, Error Codes and Messages), the address of the ECB for the failing routine, and the process ID (PID) of the current process. AEGIS then executes a TRAP instruction, causing entry to the PROM mnemonic debugger with an "S" code (see MNEMONIC DEBUGGER ERROR CODES in Chapter 4). A dump can then be taken as described below.

If the system appears hung, make sure the NORMAL/SERVICE switch is in the SERVICE position and type CTRL/<RETURN>, to pass control to the mnemonic debugger (MD). If this fails, press the RESET switch. (You should first verify that the hang is not a temporary one caused by a network failure.)

Notes on DNx60 Crash Status

DNx60s are microcoded machines. They may crash as described above with a CRASH SYSTEM message, ending up in MD but still running the micromachine (the CPU - MD prompt is '>') or they may occasionally freeze, (i.e., the micromachine halts), ending up in MD but running the CPIO processor (MD prompt is '%'). When a freeze occurs, the micro PC is printed by CPIO:

```
UPC: xxx
```

You may still take a dump from CPIO exactly as described below. A freeze is a more radical failure than a normal crash and usually indicates a hardware failure in the micromachine.

SYSTEM DUMPS

AEGIS contains a memory dump routine that can be used to dump the state of physical memory and the MMU to a floppy diskette, a cartridge tape, or a file on another node. DB, which understands the format of the dump, can then be used to analyze the cause of the crash. On completion, the dump routine executes a TRAP instruction and returns to the Mnemonic Debugger (MD).

To use this routine, follow these steps:

1. To dump to a diskette, insert a diskette into the floppy diskette drive of the Disk Storage Option. The diskette must already be formatted (with INVOL) for 1231 blocks, and must be write-enabled. (To write-enable the diskette, cover the write-enable hole with a gummed label.) (Note: if dumping to a floppy, only the first 1Mb of memory is dumped.)

2. To dump to a cartridge tape, insert a cartridge tape into the tape drive and ensure that the tape is not write-protected (arrow in upper left should point away from "safe").

3. To dump to another node, make sure that the other node is running NETMAN and that its disk has enough space for the dump (30 blocks plus 1024 blocks for each megabyte of memory on the dumping node).

4. Execute the following MD commands.

MD Commands		Comments
>RE<RETURN> <RETURN>		Reset the machine
>DI F	or	To dump to a floppy
>DI C	or	To dump to a cartridge tape
>DI N nn		To dump to node nn
*** FOR OTHER THAN DNx60s:		***
>G 100C00	or	Start dump to floppy or tape
>G 100C04		Start the dump to another node
*** FOR DNx60s: ***		
>G 20A000	or	Start dump to floppy
>G 20A004		Start the dump to another node

5. When the dump routine is complete, it executes a TRAP $F instruction and returns to MD. You should then run SALVOL and reboot AEGIS.

/SYSTEST/SSR_UTIL

The following commands are available for system debugging in /SYSTEST/SSR_UTIL.

- ALL_STCODE — Displays a list of all the system status codes for this operating system revision.

- BCR — BINARY_CROSS_REFERENCE produces a cross reference from a list of object modules, whose names are read from standard input. Only the global symbols that can be referenced from outside the module are cross referenced.

  Usage: BCR

- CPFLP — A program used to copy a file from, or to, the node's floppy disk drive. To use the program, type: Usage: CPFLP [-READ file | -WRITE file | -FMT [file]]

- CPTAPE — A program used to copy a file from, or to, the node's cartridge tape drive. To use the program, type: Usage: CPTAPE [-READ file | -WRITE file | -FMT [file]]

- DISK_ERR — Displays information about the last recorded disk error.

  Usage: DISK_ERR

- DMPF — DUMP_FILE dumps a file of any type to STDOUT, interpreted in hexadecimal and ASCII. If hex_start_offset is given, it specifies the byte of the file at which to start dumping. If hex_end_offset is given, it specifies the byte of the file at which to stop dumping. The first byte of the file is at offset zero. By default DUMP_FILE dumps the entire input file.

  Usage: DMPF input_pathname [-From hex_start_offset]
  [-To hex_end_offset]

- DMPMBX — DMPMBX formats the contents of an MBX (mailbox) file. This program gathers information to document MBX bugs or problems. DMPMBX must be used from the same node as the server of the MBX file to be dumped. Normally, you use DMPMBX to display the contents of the MBX file just before the bug is run across and just after the bug is induced. In remote cases, both the server MBX file, and the client node's `node_data/sysmbx file should be dumped out.

  Usage: DMPMBX mbx_file [-FD] [-CHAN channel_list]

- FIXCOM — FIXCOM fixes comments in source files so that you can use the -COMCHK option when the file is compiled. For more information, see the file FIXCOM.HLP in this directory.

- JUMPER — JUMPER is an interactive program that graphically displays the locations and functions of jumpers on all system printed circuit boards. JUMPER, which is completely menu-driven, makes use of the function keys (F1 to F8). Help is available in the file /SYSTEST/SSR_UTIL/JUMPER.HLP.

- LSYSERR — LSYSERR writes the contents of a system error log to standard output. Help is available in the file LSYSERR.HLP in this directory.

- LTBL — Prints the debug line number table.

  Usage: LTBL (enter object module)

- OBJDMP — Dumps an object (.bin) file.

  Usage: OBJDMP infile [-L[IST] [outfile|-]]

- RINGLOG — Monitors network traffic in and out of a node.

  Usage: RINGLOG [-start | -stop | -read]

  -Start Activates ring network message logging.
  -Stop Deactivates ring network message logging (if active) and displays current contents of the log. (This is the default!)
  -Read Displays current contents of the log. (Different from STOP because STOP deactivates logging if it is active.)

- RWVOL — RWVOL reads one physical disk address at a time, and puts it into a buffer in memory. The program first asks you to specify the controller type, and then to specify whether you want to read (R) or write (W) the disk address. If you plan to write to the disk, first choose the "R" option to read the address first.

  When you choose the "R" option, the program asks you to enter the physical disk address (Daddr:) to be read; enter it in hex. The program next asks you for the memory address to be used for the buffer containing the data it reads from the disk. If you are reading only one record at a time, issue a <CR> to use a default location for the buffer, which RWVOL then displays. If you specify a start address, you must specify an end address when prompted for it; otherwise, type

<CR>.

After RWVOL displays the "Done" message, you can use DB (if you are running online), or the Mnemonic Debugger (if you are running off-line) to look at the contents of the record that you have read, in the location the program displayed for the buffer.

Usage: RWVOL

- XMT — Examines magtape. This program is interactive and will print out a list of possible commands that can be executed if help is typed once the program has been entered.

  Usage: XMT

CHAPTER 6

PERIPHERAL I/O

DEVICE ADDRESSES (PIO)

Refer to the ADDRESS SPACE section of the appropriate chapter for each model node.

DISK PARAMETERS

DTYPE	MODEL	CYLS	HEADS	BLK/TRK	TOTAL BLOCKS
Winchester Dtype Class 000 — 14" Ring/disk PRIAM interface					
001	PRIAM 3350	561	3	18	30294 (7656)
006	PRIAM 6650	1121	3	18	60534 (EC76)
007	PRIAM 15450	1121	7	18	141246 (227BE)
Winchester Dtype Class 100 — 8" ANSI Interface					
103	Micropolis 1203	580(1)	5	13(2)	37700 (9344)
104	PRIAM 3450	525	5	12	31500 (7B0C)
105	PRIAM 7050	1049	5	12	62940 (F5DC)
Winchester Dtype Class 200 — 8" Ring/disk SMD Interface					
201	NEC D2246	687	6	18	74196 (121D4)
202	NEC D2257	1024	8	18	147546 (24000)
Winchester Dtype Class 300 — 5 1/4" ST412 Interface					
301	Micropolis 50MB	830	6	8	39840 (9BA0)
302	Micro. 86MB	1024	8	8	65536 (10000)
303	Fujitsu 86MB	754	11	8	66352 (10330)
304	Maxtor 140MB	918	15	8	110160 (1AE50)
305	Maxtor 190MB	1224	15	8	146880 (23DC0)
306	Vertex 86MB	1166	7	8	65296 (FF10)
Floppy Controller (CTYPE=1)					
001	Floppy	77	2	8	1232 (4D0)
Intel Storage Module Controller (SMD I/F) (CTYPE=4)					
000	300MB SMD	823	19	18	281466 (44B7A)
Xylogics File Server Controller (SMD I/F) (CTYPE=4)					
001	CDC (pn 3863)	711	24	26	443664 (6C510)
002	NEC (pn 5100)	760	19	31	447640 (6D498)

(1) Software uses only 525 cylinders.
(2) Software uses only 12 sectors.

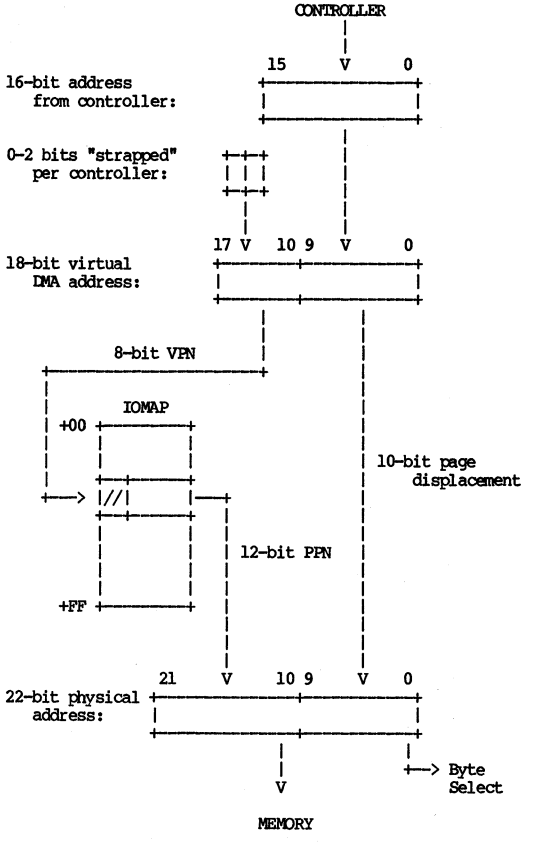
DTYPE	MODEL	SEEK TIMES (mSecs) T-to-T	AVG	MAX	RPM	AVG LATENCY	TRANSFER RATE(MBS)	AVG READ(*)
001	PRIAM 3350	8	45	85	3100	9.7	1.04	55.7
006	PRIAM 6650	8	45	85	3100	9.7	1.04	55.7
007	PRIAM 15450	8	45	75	3100	9.7	1.04	55.7
103	Micropolis 1203	12	42	85	3600	8.3	0.92	51.6
104	PRIAM 3450	8	42	75	3600	8.3	0.8	51.4
105	PRIAM 7050	8	42	75	3600	8.3	0.8	51.4
201	NEC D2246	7	25	50	3510	8.55	1.2	34.5
202	NEC D2257	5	20	40	3510	8.55	1.2	29.5
301	Micropolis 50MB	?	?	?	3600	8.3	.625	?
302	Micropolis 86MB	6	28	62	3600	8.3	.625	38.0
303	Fujitsu 86MB	5	30	65	3600	8.3	.625	40.0
304	Maxtor 140MB	5	30	52	3600	8.3	.625	40.0
305	Maxtor 190MB	5	30	52	3600	8.3	.625	40.0
306	Vertex 86MB	?	?	?	3600	8.3	.625	?
001	Floppy	?	?	?	300	10.0	?	?
000	300MB SMD	6	29	55	3600	8.3	1.2	38.18
001	CDC (pn 3863)	5	20	45	3600	8.3	1.8	28.9
002	NEC (pn 5100)	?	15	?	3070	9.8	1.8	25.4

(*) Average read = Avg. Seek time + Avg. Latency + Sector Time
(?) Exact specifications not available at time of printing.

		DADDR(hex)/CYL(dec) BAD-SPOT	DIAGNOSTIC	CYLS USED	BLKS USED(*)
001	PRIAM 3350	7620/560	75AE/559	556	30024 (7548)
006	PRIAM 6650	EC40/1120	EC0A/1119	1116	60264 (EB68)
007	PRIAM 15450	22740/1120	226C2/1119	1116	140616 (22548)
103	Micropolis 1203	9303/579	92C2/578	520	31200 (79E0)
104	PRIAM 3450	7AD0/524	7A94/523	520	31200 (79E0)
105	PRIAM 7050	F5A0/1048	F564/1047	1044	62640 (F4B0)
201	NEC D2246	12168/686	120FC/685	682	73656 (11FB8)
202	NEC D2257	23F70/1023	23EE0/1022	1019	146736 (23D30)
301	Micropolis 50MB	9B70/829	9B10/827	827	39696 (9B10)
302	Micropolis 86MB	FFC0/1023	FF40/1021	1021	65344 (FF40)
303	Fujitsu 86MB	102D8/753	10228/751	742	65296 (FF10)
304	Maxtor 140MB	1ADD8/917	1ACE8/915	915	109800 (1ACE8)
305	Maxtor 190MB	23D48/1223	23C58/1221	1221	146520 (23C58)
306	Vertex 86MB	FED8/1165	FE68/1163	1163	65128 (FE68)
001	Floppy	—	—	77	1232 (4D0)
000	300MB SMD	448CE/821	44A24/822	821	280782 (448CE)
001	CDC (pn 3863)	6C030/709	6C2A0/710	709	442416 (6C020)
002	NEC (pn 5100)	6CFFE/758	6D24B/759	751	442339 (6BFE3)

(*) Cyls, blocks used reflect loss of badspot and diagnostic cylinders and cylinders excluded for matching of secondary sources.

I/O MAP



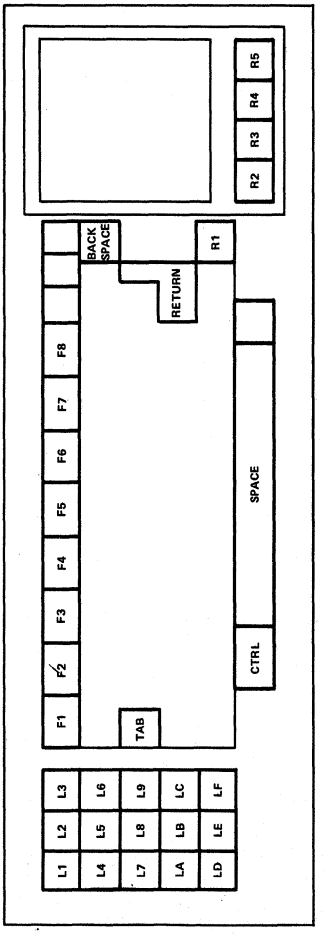
I/O MAP has 256 one-word entries from FFF800-FFF9FE.

I/O MAP ALLOCATION

DEVICE	I/O PAGES	I/O MAP ENTRY ADDRESSES	
Floppy	0-1	FFF800-FFF802	(2 pages)
Unused	2-3F	FFF804-FFF87E	(62 pages)
Winchester	40-41	FFF880-FFF882	(2 pages)
Ring Transmit	42-45	FFF884-FFF88A	(4 pages)
Ring Receive	46-49	FFF88C-FFF892	(4 pages)
Ring 2nd Rcv Chan	4A-4D	FFF894-FFF89A	(4 pages)
Unused	4E-5E	FFF89C-FFF8BC	(17 pages)
Color DMA	5F-7F	FFF8BE-FFF8FE	(33 pages)
Bit blit	80-9F	FFF900-FFF93E	(32 pages)
Unused	A0-BF	FFF940-FFF97E	(32 pages)
Multibus	C0-FF	FFF980-FFF9FE	(64 pages)

KEYBOARD

880 Keyboard - Map



880 Keyboard Chart - Physical

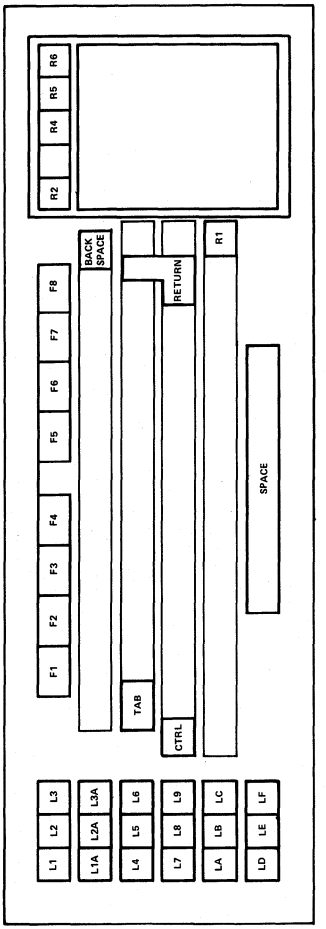
ASCII Codes — High Order Nibble (columns) / Low Order Nibble (rows)

Low Order Nibble	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Low Order Nibble
0	^`	^P	BLNK	0	@	P	`	p	R1	N0	R1U	N0U	F1	F1S	F1U	F1C	0
1	^A	^Q	!	1	A	Q	a	q	L1	N1	L1U	N1U	F2	F2S	F2U	F2C	1
2	^B	^R	"	2	B	R	b	r	L2	N2	L2U	N2U	F3	F3S	F3U	F3C	2
3	^C	^S	#	3	C	S	c	s	L3	N3	L3U	N3U	F4	F4S	F4U	F4C	3
4	^D	^T	$	4	D	T	d	t	L4	N4	L4U	N4U	F5	F5S	F5U	F5C	4
5	^E	^U	%	5	E	U	e	u	L5	N5	L5U	N5U	F6	F6S	F6U	F6C	5
6	^F	^V	&	6	F	V	f	v	L6	N6	L6U	N6U	F7	F7S	F7U	F7C	6
7	^G	^W	'	7	G	W	g	w	L7	N7	L7U	N7U	F8	F8S	F8U	F8C	7
8	^H	^X	(	8	H	X	h	x	L8	N8	L8U	N8U	\	\S	tpad	^\	8
9	^I	^Y	)	9	I	Y	i	y	L9	N9	L9U	N9U	|	|S		|C	9
A	^J	^Z	*	:	J	Z	j	z	LA	N.	LAU	N.U	TAB	TABS		TABC	A
B	^K	^[	+	;	K	{	k	[	LB	N=	LBU	N=U	CR	CRS		CRC	B
C	^L		,	<	L		l		LC	N+	LCU	N+U	/	?		^/	C
D	^M	^]	-	=	M	}	m	]	LD	N-	LDU	N-U	R2	R5	R2U	R5U	D
E	^N	^~	.	>	N	^	n	~	LE	N*	LEU	N*U	R3	BS	R3U		E
F	^O		/		O	_	o		LF	N/	LFU	N/U	R4		R4U		F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

880 Keyboard Chart - Translated (User Mode)

Low Order Nibble ASCII Codes	High Order Nibble 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Low Order Nibble ASCII Codes
0	^`	^P	SP	0	@	P	`	p		R1		R1U	F1	F1S	F1U	F1C	0
1	^A	^Q	!	1	A	Q	a	q	L1	R2	L1U	R2U	F2	F2S	F2U	F2C	1
2	^B	^R	"	2	B	R	b	r	L2	R3	L2U	R3U	F3	F3S	F3U	F3C	2
3	^C	^S	#	3	C	S	c	s	L3	R4	L3U	R4U	F4	F4S	F4U	F4C	3
4	^D	^T	$	4	D	T	d	t	L4	R5	L4U	R5U	F5	F5S	F5U	F5C	4
5	^E	^U	%	5	E	U	e	u	L5	BS	L5U		F6	F6S	F6U	F6C	5
6	^F	^V	&	6	F	V	f	v	L6	CR	L6U		F7	F7S	F7U	F7C	6
7	^G	^W	'	7	G	W	g	w	L7	TAB	L7U		F8	F8S	F8U	F8C	7
8	^H	^X	(	8	H	X	h	x	L8	STAB	L8U		N0	N8	N0U	N8U	8
9	^I	^Y	)	9	I	Y	i	y	L9	CTAB	L9U		N1	N9	N1U	N3U	9
A	^J	^Z	*	:	J	Z	j	z	LA		LAU		N2	N.	N2U	N.U	A
B	^K	^[	+	;	K	[	k	{	LB		LBU		N3	N=	N3U	N=U	B
C	^L	^\	,	<	L	\	l	|	LC		LCU		N4	N+	N4U	N+U	C
D	^M	^]	-	=	M	]	m	}	LD		LDU		N5	N-	N5U	N-U	D
E	^N	^~	.	>	N	^	n	~	LE		LEU		N6	N*	N6U	N*U	E
F	^O	^/	/	?	O	_	o	^|	LF		LFU		N7	N/	N7U	N/U	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

Low-Profile Keyboard - Map



Low-Profile Keyboard Chart - Physical

ASCII Codes

Low Order Nibble \ High Order Nibble	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	^SP	^P	SP	0	@	P	`	p	R1	R1S	R1U	L1A	F1	F1S	F1U	F1C	0
1	^A	^Q	!	1	A	Q	a	q	L1	L1S	L1U	L2A	F2	F2S	F2U	F2C	1
2	^B	^R	"	2	B	R	b	r	L2	L2S	L2U	L3A	F3	F3S	F3U	F3C	2
3	^C	^S	#	3	C	S	c	s	L3	L3S	L3U	R6	F4	F4S	F4U	F4C	3
4	^D	^T	$	4	D	T	d	t	L4	L4S	L4U	L1AS	F5	F5S	F5U	F5C	4
5	^E	^U	%	5	E	U	e	u	L5	L5S	L5U	L2AS	F6	F6S	F6U	F6C	5
6	^F	^V	&	6	F	V	f	v	L6	L6S	L6U	L3AS	F7	F7S	F7U	F7C	6
7	^G	^W	'	7	G	W	g	w	L7	L7S	L7U	R6S	F8	F8S	F8U	F8C	7
8	^H	^X	(	8	H	X	h	x	L8	L8S	L8U	L1AU	\		tpad	^\	8
9	^I	^Y	)	9	I	Y	i	y	L9	L9S	L9U	L2AU	|		R2S		9
A	^J	^Z	*	:	J	Z	j	z	LA	LAS	LAU	L3AU	TAB	TABS	R3S	TABC	A
B	^K	ESC	+	;	K	[	k	{	LB	LBS	LBU	R6U	CR	CRS	R4S	CRC	B
C	^L		,	<	L		l		LC	LCS	LCU		/	?	R5S	^/	C
D	^M	^]	-	=	M	}	m	]	LD	LDS	LDU		R2	R5	R2U	R5U	D
E	^N	^~	.	>	N	^	n	~	LE	LES	LEU		R3	BS	R3U		E
F	^O				O	__	o	DEL	LF	LFS	LFU		R4	mous	R4U		F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

Low-Profile Keyboard Chart - Translated (User Mode)

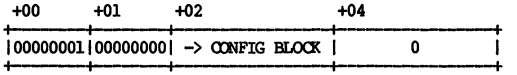
Low Order Nibble ASCII Codes	High Order Nibble 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	^SP	^P	SP	0	@	P	`	p		R1		R1U	F1	F1S	F1U	F1C	0
1	^A	^Q	!	1	A	Q	a	q	L1	R2	L1U	R2U	F2	F2S	F2U	F2C	1
2	^B	^R	"	2	B	R	b	r	L2	R3	L2U	R3U	F3	F3S	F3U	F3C	2
3	^C	^S	#	3	C	S	c	s	L3	R4	L3U	R4U	F4	F4S	F4U	F4C	3
4	^D	^T	$	4	D	T	d	t	L4	R5	L4U	R5U	F5	F5S	F5U	F5C	4
5	^E	^U	%	5	E	U	e	u	L5	BS	L5U	R2S	F6	F6S	F6U	F6C	5
6	^F	^V	&	6	F	V	f	v	L6	CR	L6U	R3S	F7	F7S	F7U	F7C	6
7	^G	^W	'	7	G	W	g	w	L7	TAB	L7U	R4S	F8	F8S	F8U	F8C	7
8	^H	^X	(	8	H	X	h	x	L8	STAB	L8U	R5S	R1S	L8S	L1A	L1AU	8
9	^I	^Y	)	9	I	Y	i	y	L9	CTAB	L9U		L1S	L9S	L2A	L2AU	9
A	^J	^Z	*	:	J	Z	j	z	LA		LAU		L2S	LAS	L3A	L3AU	A
B	^K	ESC	+	;	K	[	k	{	LB		LBU		L3S	LBS	R6	R6U	B
C	^L	^\	,	<	L	\	l	|	LC		LCU		L4S	LCS	L1AS		C
D	^M	^]	-	=	M	]	m	}	LD		LDU		L5S	LDS	L2AS		D
E	^N	^~	.	>	N	^	n	~	LE		LEU		L6S	LES	L3AS		E
F	^O	^?	/	?	O	_	o	DEL	LF		LFU		L7S	LFS	R6S		F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

MAGTAPE CONTROLLER

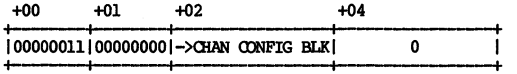
Controller control page: FE8000
Interrupt vector number: B3 ($2CC in page 0)
MULTIBUS interrupt level: 3 (See MIC)
I/O MAP entries: FFF980-FFF9FE (64 pages)

FE80AA	CHANNEL ATTENTION (do something useful)
FE80AB	CONTROLLER RESET

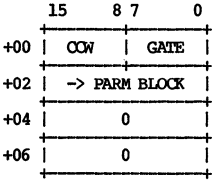
System Configuration Pointer (at xxxFF6)



System Configuration Block



Channel Control Block



CCW: Set to $11 for normal operations. Set to $09 to clear active interrupt.
GATE: Set to $FF before starting an operation. Set to 00 by controller on completion.

To initiate an operation

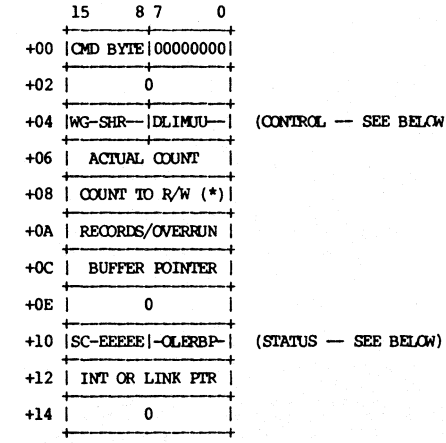
(Set up parameter block)

```
MOVE.W  #$11FF,GATE      CLOSE GATE
MOVE.B  #0,$FE80AA       WAKE UP CONTROLLER
```

To acknowledge tape interrupt:

```
MOVE.W  #$09FF,GATE         CLOSE GATE
MOVE.B  #$20,CMD_BYTE       DO-NOTHING COMMAND
MOVE.W  NO_I_BIT,CONTROL    ENSURE INTERRUPT BIT OFF
MOVE.B  #00,$FE80AA         WAKE UP CONTROLLER
TST.B   GATE+1              WAIT FOR ACK TO FINISH
BNE     *-4
```

Parameter Block



* Manufacturer recommends 4K-8K.

COMMAND BYTE

00	Initialize	3C	Edit (rewrite prior record)
20	No operation	40	Write filemark
28	Return status	44	Skip to filemark
2C	Read	48	Space "n" records
30	Write	70	Space "n" or to filemark
34	Rewind	4C	Erase fixed (3.5" * "n")
38	Rewind/Unload	50	Erase from here to EOT

CONTROL WORD (appropriate value)

W	8000	Bus width (0 => 8 bits, 1 => 16)	(1)
G	4000	Grab bus before tape movement	(0)
S	1000	Operate in streaming mode	(?)
H	0800	Select high speed (100ips)	(0)
R	0400	Reverse direction for operation	(?)
D	0080	Grab bus during DMA transfers	(0)
L	0040	Link (=> ignore I and M bits)	(?)
I	0020	Interrupt when done	(1)
M	0010	1 => use mailbox interrupts	(0)
UU	000C	Unit select (00 through 11)	(00)

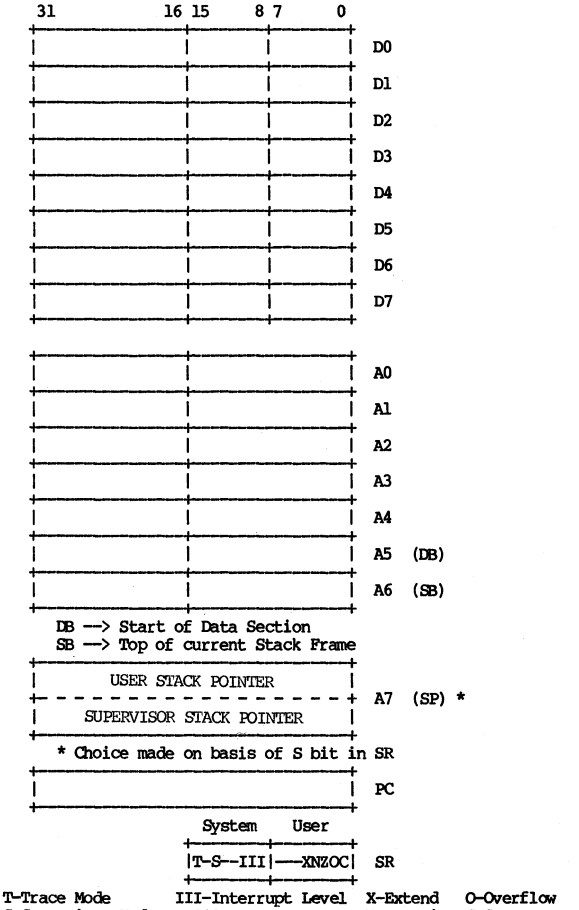
STATUS WORD (normal state at completion)

S	8000	Execution (of parm block) started	(1)
C	4000	Execution completed OK	(1)
E..E	1F00	Error code:	(00)
		00 No unrecoverable error	
		01 Timed out waiting for Data Busy false	
		02 Timed out waiting for Data Busy false, Formatter Busy false and Ready true	
		03 Timed out waiting for Ready false	
		04 Timed out waiting for Ready true	
		05 Timed out waiting for Data Busy true	
		06 Memory time-out during system memory reference	
		07 Blank tape encountered	
		08 Error in micro-diagnostic	
		09 Unexpected end of tape	
		0A Tape read/write error	
		0B Tape overrun	
		0D Read parity error	
		0E Checksum error	
		0F Tape timeout (read on blank tape or reading larger block than written)	
		10 Tape not ready	
		11 Write attempted on protected tape	
		13 Diagnostic mode jumper not installed	
		14 Illegal attempt to link	
		15 Filemark encountered during read operation	
		16 Parameter error (byte count zero or too large)	
		18 Hardware error	
		19 Read or write terminated by OS or disk	
O	0040	On-line	(1)
L	0020	Load point	(?)
E	0010	End of tape	(?)
R	0008	Ready	(1)
B	0004	Formatter busy	(0)
P	0002	Tape is write protected	(?)

MULTIBUS DEVICES

INTERRUPT LEVEL	PRODUCT CODE	PRODUCT DESCRIPTION	PART NO	CSR-ADDR (PAGE)	MEMORY MAPPED ADDRESSES
0	——	Reserved for customers	——	0-3C00	0000-7C00
1	COM_ETH	Interlan Ethernet Ctlr	003613	80-8F	
1	——	FPS array proc.	——	7400	
2	COM-X25	TTTN X25 Ctlr	002858		7000-7FFF
3	MSD1600	Rimfire Transport Ctlr	001012	7800	
4	MSD-300	Intel SMD Ctlr	001380	6C00	
4	MSD-500	Xylogics FSD Ctlr	003864		
5	SFT-VER	IKON 10071-5 Mbus Versatec (V80) Ctlr		400	
6	—	Printronix Parallel Ptr.		7C00	

REGISTER SET



TIMERS

Write CR1 or CR3	[8800	FFAC00]
Rd Status, Write CR2	[8802	FFAC02]
Read/Write Timer 1, High	[8804	FFAC04]
Read/Write Timer 1, Low	[8806	FFAC06]
Read/Write Timer 2, High	[8808	FFAC08]
Read/Write Timer 2, Low	[880A	FFAC0A]
Read/Write Timer 3, High	[880C	FFAC0C]
Read/Write Timer 3, Low	[880E	FFAC0E]
Calendar Control	[8880	FFAC80]
Calendar Data Write	[8882	FFAC82]
Calendar Data Read	[8884	FFAC84]

TOUCHPAD

The touchpad sends approximately 30 data points per second through the same SIO port (zero) as the keyboard, at a speed of 1200 baud. Each data point has four bytes, as follows:

escape code	low 8 bits of X	low 4: high 4 bits : bits of Y : of X	high 8 bits of Y
E8			
		4-7 0-3	
byte 0	byte 1	byte 2	byte 3

The range of X and Y coordinates is approximately 30 to 1100.

CHAPTER 7

DN300,DN320

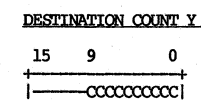
ADDRESS SPACE

physical		virtual
100400	traps	0
400	PROM	400->7FFF (one-to-one)
100800	phys mem	100800->FFFFF
700000	PTT	700000->7FFFFF
100000	MD STK,DATA	E00000
20000	displ_mem	FC0000->FDFFFF
B000	FPU ctl	FF7000
B400	FPU cmd	FF7400
B800	FPU cs	FF7800
9400	disp 1	FF9800
9800	ring 2	FF9C00
9000	DMA ctl	FFA000
9C00	FLP,WIN,CAL	FFA800
8800	timers	FFAC00
8400	sios	FFB000
8000	mmu	FFB400
4000	pft	FFB800->FFF7FF

BLT REGISTERS

(Each has an address used for reading and a separate address used for writing.)

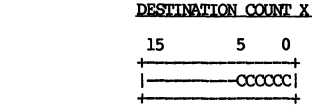
DESTINATION COUNT Y REGISTER [9414 | 0FF9814]



CCCCCC = -1 - ABS(WDSY-WDEY)

= two's complement for number of lines in height of destination block.

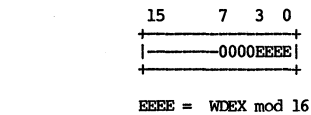
DESTINATION COUNT X REGISTER [9416 | 0FF9816]



CCCCCC = -1 - ABS(WDSX/16-WDEX/16)

= two's complement for number of 16-bit aligned words involved in X coordinate.

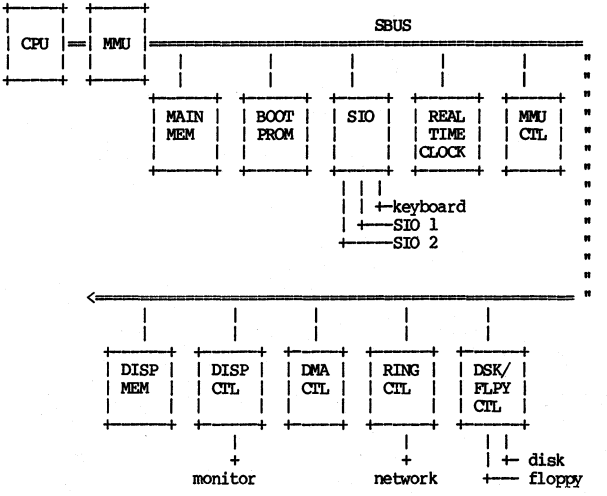
DESTINATION END BIT REGISTER [941C | 0FF981C]



EEEE = WDEX mod 16

= bit number in word of last bit of X.

CONFIGURATION



DISK (FLOPPY/WINCHESTER) CONTROLLER

Address: [9C00 | 0FFA800]

	WRITE	READ
+00	ANSI COMMAND	ATTENTION STATUS
+02	ANSI PARM OUT	ANSI PARM IN
+04		DRIVE NO. OF STATUS
+06	SECTOR	CONTROLLER STAT-HIGH
+07		CONTROLLER STAT-LOW
+08	CYLINDER-HIGH	
+09	CYLINDER-LOW	
+0A	HEAD	
+0C	INTERRUPT CONTROL	
+0E	CONTROLLER COMMAND	
+10		FLOPPY STATUS
+12	FLOPPY WRITE DATA	FLOPPY READ DATA
+14	FLOPPY CONTROL	
+20	CALENDAR CONTROL	
+22	CALENDAR WRITE DATA	
+24		CALENDAR READ DATA

CONTROLLER STATUS [9C06 | 0FFA806]

			Reset by
8000	15	Controller busy	Self-clearing
4000	14	Drive busy (from bus)	Self-clearing
2000	13	Drive attention (from bus)	Ctlr, if status avail enable
1000	12	Status available interrupt	Read attn status reg
0800	11	End of operation interrupt	Write to ctlr cmd reg
0400	10	Floppy interrupting	Read floppy status reg
0080	07	Time-out	Write to ctlr cmd reg
0040	06	Overrun	Write to ctlr cmd reg
0020	05	CRC error	Write to ctlr cmd reg
0010	04	Controller bus parity error	Write to ctlr cmd reg
0008	03	Illegal configuration	Write to ctlr cmd reg
0004	02	Status time-out	Read attention status register
0002	01	Parity error during DMA	Write to controller command register

PARAMETER OUT COMMANDS

		Parameter	
40	Attention control	Bit 7 = 0 => enable attention 1 => disable attention	
41	Write control	Bit 7 = 0 => write protect 1 => write enable	
42	Load cyl addr high	MSB of cylinder address	
43	Load cyl addr low	LSB of cylinder address	
44	Select head	Head number	Mandatory
50	Load attribute number	Attribute number	Optional
51	Load attribute	Attribute	
53	Read control	Bits 7,6 = 0x - nominal strobe 10 - strobe early 11 - strobe late	
54	Offset control	Bits 7,6 = 0x - no offset 10 - offset forward 11 - offset reverse	
55	Spin control	Bit 7 = 0 - spin down 1 - spin up	
56	Load sect/trk high	MSB of sectors/track	
57	Load sect/trk medium	MedSB of sectors/track	
58	Load sect/trk low	LSB of sectors/track	
59	Load bytes/sect high	MSB of bytes/sector	
5A	Load bytes/sect medium	MedSB of bytes/sector	
5B	Load bytes/sect low	LSB of bytes/sector	
6B	Load read permit high	MSB of read enable on cyl >=	
6C	Load read permit low	LSB of read enable cyl	
6D	Load write permit high	MSB of write enable on cyl >=	
6E	Load write permit low	LSB of write enable cyl	
6F	Load test byte	Test byte	

PARAMETER IN COMMANDS

00	Report illegal command		General status	
01	Clear fault		General status	
02	Clear attention		General status	
03	Seek	*	General status	
04	Rezero	*	General status	
0D	Report sense byte 2		Sense byte 2	
0E	Report sense byte 1		Sense byte 1	
0F	Report general status		General status	Mandatory
10	Report drive attribute		Drive attribute	Optional
11	Set attention	*	General status	
14	Selective reset	*	General status	
15	Seek to landing zone	*	General status	
16	Reformat track	*	General status	
29	Report cyl addr high		MSB of cylinder address	
2A	Report cyl addr low		LSB of cylinder address	
2B	Report read permit high		MSB of cylinder address	
2C	Report read permit high		LSB of cylinder address	
2D	Report wrt permit high		MSB of cylinder address	
2E	Report wrt permit high		LSB of cylinder address	
2F	Report test byte		Test byte	

* Time-dependent command; attention set on completion.

ATTENTION STATUS [9000 | 0FFA800]

			Cleared by
7	80	Normal completion *	Clear attention command
6	40	Busy	Self clearing
5	20	Read sense byte 2	See sense byte 2
4	10	Read sense byte 1	See sense byte 1
3	08	Illegal parameter *	Clear fault command
2	04	Illegal command *	Clear fault command
1	02	Control bus error *	Clear fault command
0	01	Not ready	Self clearing

* Zero to one transition sets attention.

SENSE BYTE 1

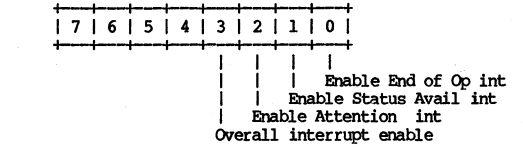
				mand/opt
7	80	Vendor unique errors	*	O
6	40	Other errors	*	O
5	20	Command reject	*	M
4	10	Speed error	*	O
3	08	R/W permit violation	*	O
2	04	Power fault	*	O
1	02	Read/write fault	*	M
0	01	Seek error	*	M

SENSE BYTE 1

				mand/opt
7	80	Vendor unique attns	*	O
6	40	In write-protected area		M
5	20	Attr table modified	*	O
4	10	Dev rsrved to alt port		O
3	08	Forced release	*	O
2	04	Dev rsrved to this port		O
1	02	Ready transition	*	M
0	01	Initial state	*	M

* Zero to one transition sets attention.

INTERRUPT CONTROL [9C0C | 0FFA80C]

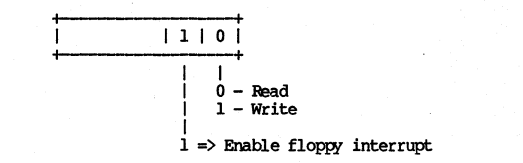


CONTROLLER COMMANDS [9C0E | 0FFA80E }

00 - No-op
01 - Read record
02 - Write record
03 - Format track
04 - Seek
05 - Execute ANSI command sequence
06 - Execute drive select sequence
07 - Execute attention in sequence
08 - Select head

Any command clears the controller status register.

FLOPPY CONTROL [9C14 | 0FFA814]



DISPLAY CONTROL AND STATUS REGISTER (DCSR)

DISPLAY REGISTERS [9400 | 0FF9800]

	WRITE	READ
+00	DISPLAY CONTROL	DISPLAY STATUS
+02	DEB	
+04	WSSY	
+06	WSSX	
+08	DCY	
+0A	DCX	
+0C	WSDY	
+0E	WSDX	

DISPLAY CONTROL REGISTER [9400 | 0FF9800]

8000 - GO (Start BLT operation)	15
0020 - Interrupt at end of frame	5
0010 - Interrupt at end of BLT operation	4
0008 - Increment Y coordinate	3
0004 - Increment X coordinate	2
0002 - Fill mode BLT operation	1
0001 - Enable display (blank if reset)	0

DISPLAY STATUS REGISTER [9400 | 0FF9800]

8000 - BLT operation in progress	15
0080 - End of frame interrupt	7
0002 - Reserved	1
0001 - Reserved	0

DMA CONTROLLER

DMAC page at [9000 | 0FFA000]

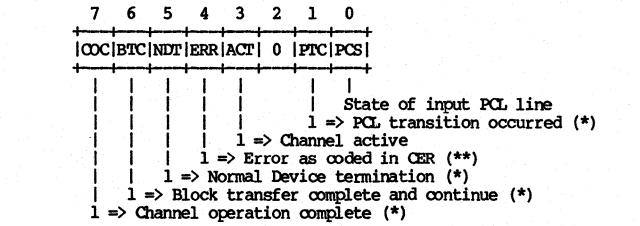
DMA controller is a Motorola M68450.

9000-903F - ring receive header
9040-907F - ring receive data
9080-90BF - ring transmit
9000-90FF - winchester/floppy

Register summary (for each channel):

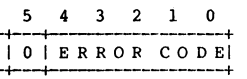
+00	Channel status register	(CSR)	R/W
+01	Channel error register	(CER)	R
+04	Device control register	(DCR)	R/W
+05	Operation control register	(OCR)	R/W
+06	Sequence control register	(SCR)	R/W
+07	Channel control register	(CCR)	R/W
+0A	Memory transfer counter	(MTC)	R/W
+0C	Memory address register	(MAR)	R/W
+14	Device address register (not used)		
+1A	Base transfer counter	(BTC)	R/W
+1C	Base address register	(BAR)	R/W
+25	Normal interrupt vector (not used)		
+27	Error interrupt vector (not used)		
+29	Memory function code register	(MFCR)	R/W
+2D	Channel priority register	(CPR)	R/W
+31	Device function code register (not used)		
+39	Base function code register	(BFCR)	R/W
+FF	General Control Register (not used)		

CHANNEL STATUS REGISTER (CSR) [9000 | 0FFA000]



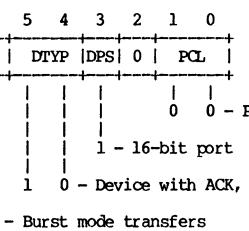
(*) Bit cleared by writing a one bit to CSR.
(**) Ditto; clearing also clears CER.

CHANNEL ERROR REGISTER (CER) [9001 | 0FFA001]

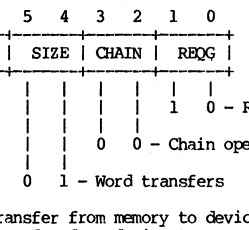


00 - No error
01 - Configuration error
02 - Operation timing error
03 - (undefined, reserved)
05 - Address error: memory address or memory counter
06 - Address error: device address
07 - Address error: base address or base counter
09 - Bus error: memory address or memory counter
0A - Bus error: device address
0B - Bus error: base address or base counter
0D - Count error: memory address or memory counter
0E - Count error: device address
0F - Count error: base address or base counter
10 - External abort
11 - Software abort

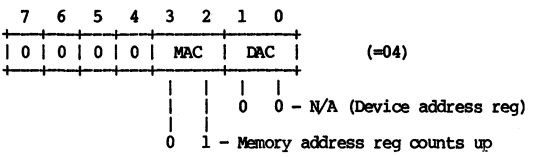
DEVICE CONTROL REGISTER (DCR) [9004 | 0FFA004]



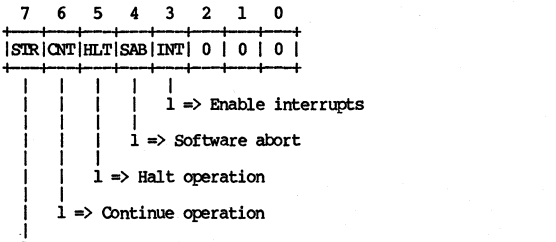
OPERATION CONTROL REGISTER (OCR) [9005 | 0FFA005]



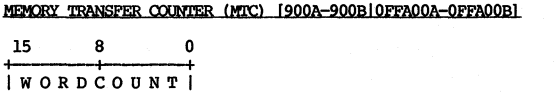
SEQUENCE CONTROL REGISTER (SCR) [9006 | 0FFA006]



CHANNEL CONTROL REGISTER (CCR) [9007 | 0FFA007]

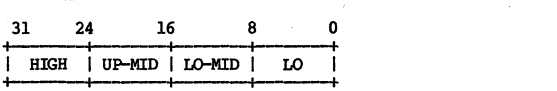


MEMORY TRANSFER COUNTER (MTC) [900A-900B|0FFA00A-0FFA00B]



(E.g., 512 to transfer a page.)

MEMORY ADDRESS REGISTER (MAR) [900C-900F|0FFA00C-0FFA00F]



Load with MOVEP.L A0,9018.

DEVICE ADDRESS REGISTER (DAR) [9014-9017|0FFA014-0FFA017]

Not used.

BASE TRANSFER COUNTER (BTC) [901A-901B|0FFA01A-0FFA01B]

Same as Memory Transfer Counter.

BASE ADDRESS REGISTER (BAR) [901C-901F | 0FFA01C-0FFA01F]

Same as Memory Address Register.

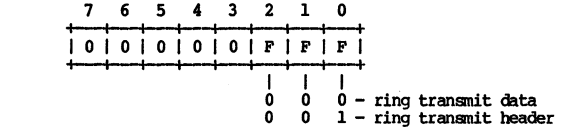
NORMAL INTERRUPT VECTOR REGISTER [9025 | 0FFA025]

Not used.

ERROR INTERRUPT VECTOR REGISTER [9027 | 0FFA027]

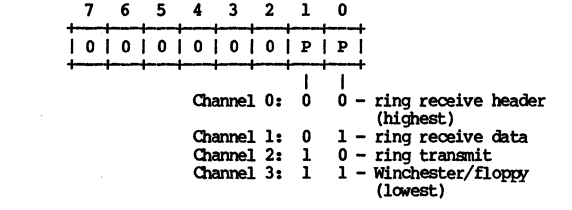
Not used.

MEMORY FUNCTION CODE REGISTER (MFCR) [9029 | 0FFA029]



Function code is not used on other channels.

CHANNEL PRIORITY REGISTER (CPR) [902D | 0FFA02D]



DEVICE FUNCTION CODE REGISTER (DFCR) [9031 | 0FFA031]

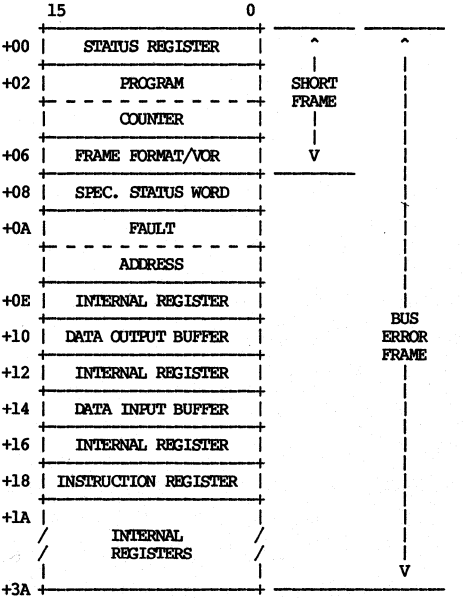
Not used.

BASE FUNCTION CODE REGISTER (BFCR) [9039 | 0FFA039]

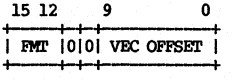
Not used.

FAULT FRAME

BUS/ADDRESS ERROR STACK FRAME FORMAT

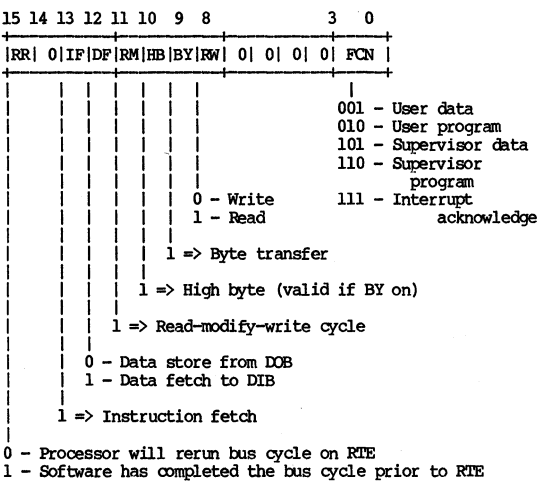


FRAME FORMAT/VECTOR OFFSET WORD



0000 - four-word format: SR, PC, VOR
1000 - 29-word 68010 format.

SPECIAL STATUS WORD



FAULT TYPES

Group	Exception	Processing
0	Reset Bus Error Address Error	Current instruction is aborted.
1	Trace Interrupt Illegal Instruction Privilege Instruction	Exception occurs before next instruction.
2	TRAP, TRAPV CHK Zero Divide	Processed by normal instruction execution.

Group 0 exceptions have the highest priority.

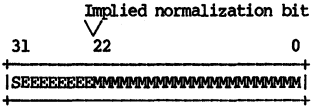
FAULT VECTORS

Exception vector at [100400 | 0]
(+ => DN300 only)

	Vector	Address	Assignment	
	00	000	Reset: Initial SSP	
		004	Reset: Initial PC	
	02	008	Bus Error	
	03	00C	Address Error	
	04	010	Illegal Instruction	
	05	014	Zero Divide	
	06	018	CHK Instruction	
	07	01C	TRAPV Instruction	
	08	020	Privilege Violation	
	09	024	Trace	
	0A	028	Unimplemented instruction	
	0B	02C	Unimplemented instruction	
	0C-0D	030	(Unassigned, reserved)	
+	0E	038	Invalid Stack Format	
	0F-17	03C	(Unassigned, reserved)	
	18	060	Spurious Interrupt	
+	19-1F	064	Level 1-7 Auto Vector	interrupt level
+	19	064	SIO (receive and transmit)	1
+	1A	068	Keyboard input	2
+	1B	06C	Ring	3
+	1C	070	Display	4
+	1D	074	Disk/floppy	5
+	1E	078	Timers 1,2,3	6
+	1F	07C	Parity error	7
	20-2F	080	TRAP Instruction Vectors	
	30-3F	0C0	(Unassigned, reserved)	
	40-8F	100	User Interrupt Vectors - unused	
	A0-AF	280	User Interrupt Vectors - unused	
	B8-C3	2E0	User Interrupt Vectors - unused	
	C4-CD	310	Unused	
	F8-FA	3E0	Unused	
	FC-FD	3F0	Unused	

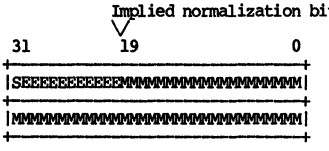
FLOATING-POINT FORMAT

Single-Precision Floating-Point Format



S - Sign; S = 1 => Negative
E..E - Exponent plus 127
M..M - Mantissa

Double-Precision Floating-Point Format



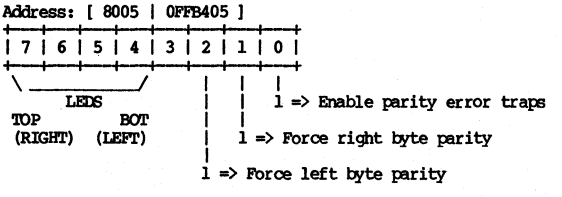
S - Sign; S = 1 => Negative
E..E - Exponent plus 1023
M..M - Mantissa

FPU

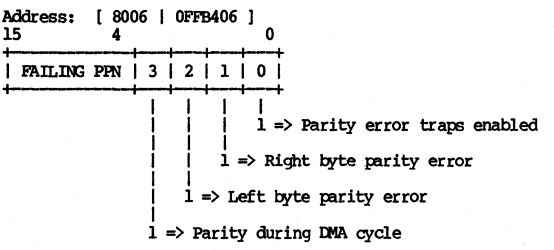
The FPU is essentially the same as PEB without the cache (and different microcode). Refer to the PEB section of Chapter 8, DN400,DN420,DN600.

MEMORY CONTROL/STATUS REGISTERS (MCSR)

Memory Control Register



Memory Status Register

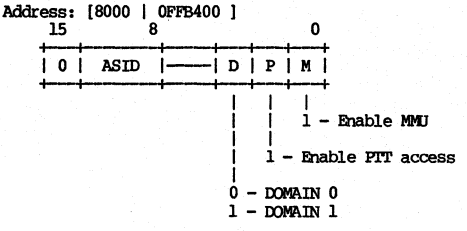


Writing MSR clears the parity error condition.

MEMORY MANAGEMENT UNIT (MMU)

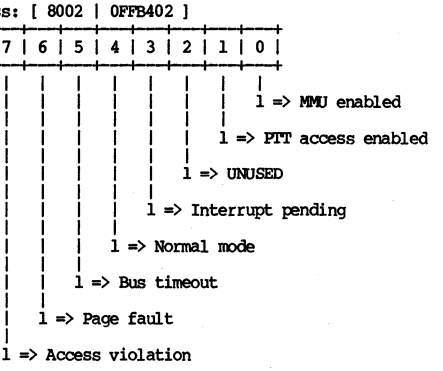
PID/PRIV/POWER [8000 | FFB400]
MMU Status [8002 | FFB402]

PID/PRIV Register



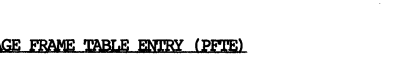
MMU Status Register

Address: [8002 | 0FFB402]



PAGE FRAME TABLE ENTRY (PFTE)

(Type "pfte" in mmpft.pvt.pas)



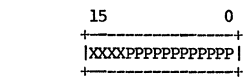
A..A - Address space ID (0-127) (.elsid)
S - Supervisor domain (.elccess)
D - DOMAIN (0 or 1)
W - Write access
R - Read access
X - Execute access
P..P - Excess virtual page number (.xsvpn)
E - End of chain (.eoc)
M - Page modified (.bbmod)
U - Page referenced (.used)
G - Page is global (.global)
L..L - PFT hash thread (.link)

PFT at [4000 | FFB800] through [8000 | FFF800].

There is one entry per physical page of memory.

PAGE TRANSLATION TABLE ENTRY (PTTE)

(type "ppn_t" in base.ins.pas)



P..P - Physical page number (PPN)

XXXX - Junk - ignore

Page Translation Table at [n/a | 700000]
through [n/a | 800000]

One PTTE every 1024 bytes in table.

PEB

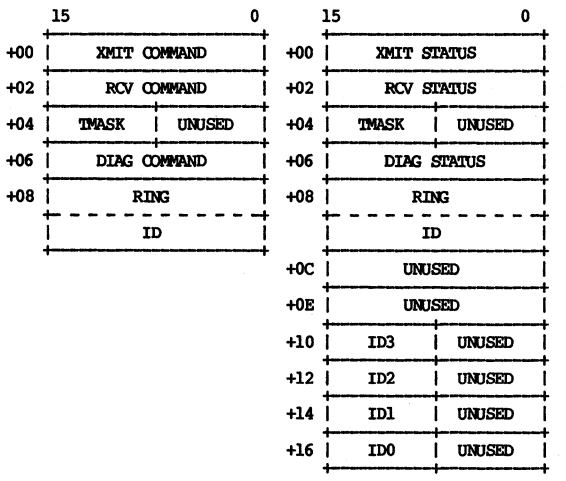
Refer to FPU section.

PROM ENTRY POINTS

```
100:    dc.w    2,0             2 => swallow, no aux info
104:    ac      getc            returns char in dl
108     ac      putc            prints char in dl
10C     ac      init_dsk        initialize disk
110     ac      read_dsk        read a record from disk
114     ac      reload_font     reload font
118     ac      pollc           returns char in dl, else -1 in dl.w
11C     ac      quiet_ret       quiet return to prom
```

RING REGISTERS

RING page at [9800 | 0FF9C00]



TRANSMIT COMMAND [9800 | 0FF9C00]

4000 transmit interrupt enable
2000 transmit enable (start the transmit)
1000 force transmit

NOTES:
1. To start a transmit normally, use 6000.
2. To force transmit, use 7000.
3. To stop a transmit that has already started, clear the transmit enable bit.
4. Writing anything to this register clears the transmit interrupt.

RECEIVE COMMAND [9802 | 0FF9C02]

4000 enable interrupt
2000 enable receive (start the receive)

NOTES:
1. To start a normal receive, use 6000.
2. To stop a receive that has already started, clear the receive enable bit.

TRANSMIT STATUS [9800 | 0FF9C00]

8000 interrupt pending
4000 interrupt enabled
2000 busy
1000 disconnected
0800 bi-phase error
0400 elastic store before error
0200 no return (a complete packet frame never arrived)
0100 crc error
0080 ack parity error (0=no error, 1=error detected)
0040 external error (error during DMA, e.g. parity, bus-error)
0020 protocol error (the packet hdr with FROM ID never came back)
0010 icopy (somebody Intended to COPY -- was willing to receive)
0008 ack byte errbit (somebody (anybody!) set the "error detected" bit)
0004 copy (somebody did COPY the packet)
0002 wack
0001 underrun (DMA didn't keep up with xmit data rate)

NOTES:
1. A successful transmit will have a transmit status of 0014.
2. A WACK will have a transmit status of 0012.

RECEIVE STATUS [9802 | 0FF9C02]

8000 interrupt pending
4000 interrupt enabled
2000 busy
1000 disconnected
0800 bi-phase error
0400 elastic store before error
0200 timeout (The hdr of a message was seen, but it never ended)
0100 crc error
0080 ack parity error (0=no error, 1=error detected)
0040 external error (error during DMA, e.g. parity, bus-error)
0020 DMA end of range
0010 icopy (somebody before me Intended to COPY)
0008 ack byte errbit (somebody before me set the "error detected" bit)
0004 copy (somebody before me did COPY the packet)
0002 wack (somebody before me WACKed the packet)
0001 overrun (DMA didn't keep up with receive data rate)

DIAGNOSTIC STATUS [9806 | 0FF9C06]

8000 interrupt pending (bad_pkt_cnt_overflow interrupt)
4000 interrupt enabled (bad_pkt_cnt_overflow interrupt)
2000 connected to the network
1000 sticky bi-phase error (error seen since bit was cleared)
0800 delay on (the delay is enabled)
0400 sticky good_seen (good packet seen since bit was cleared)
0200 sticky elastic store before error (error seen since bit was cleared)
01FF bad packet count (nine-bit counter for first detecting of errors)

NOTES:

1. Counter is the number of times this node found an error in a packet going by (regardless of packet target node ID), found the error bit in the ack byte clear, and so was the first to set the error bit to a one.

2. The bad_pkt_cnt interrupt occurs when the counter counts from 255 to 256 (i.e., it first uses its highest-order bit).

3. The counter sticks at 511 if more than 511 errors are seen.

4. Writing anything to the diagnostic command register (word) (see below) clears the interrupt and all sticky bits.

DIAGNOSTIC COMMAND [9806 | 0FF9C06]

8000 DMA test (loop transmit DMA to receive DMA)
4000 enable interrupt (bad_pkt_cnt overflow interrupt)
2000 connect (to the network)
1000 disconnect (from the network)
0800 delay off (disable the delay)
0400 delay on (enable the delay)
0100 snoop (accept all packets but only set ack byte for packets actually addressed to me)

NOTE:
Writing anything to the register (word) clears interrupt and all sticky bits in the diagnostic status register.

TMASK [9804 | 0FF9C04]

80 broadcast
40 hardware diagnostic
20 thank you
10 please
08 paging
04 user
02 software diagnostic
01 xtype3

NOTE:
Except for BROADCAST, these bits are software defined.

SERIAL I/O INTERFACE

SIO page at [8400 | 0FFB000]

SIO lines are implemented with a Signetics SC2681 DUART. The display keyboard interface implemented with a Motorola MC6850.

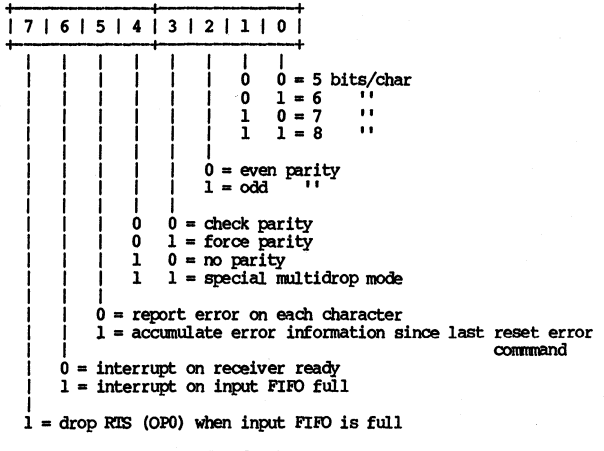
When both SIO lines are being used, it is possible to have incompatible baud rates due to limitations of the SC2681 chip. One SIO line can't have a baud rate from Group A while the other SIO line is set from Group B:

Group A	Group B
50	75
7200	150
	2000
	19.2K

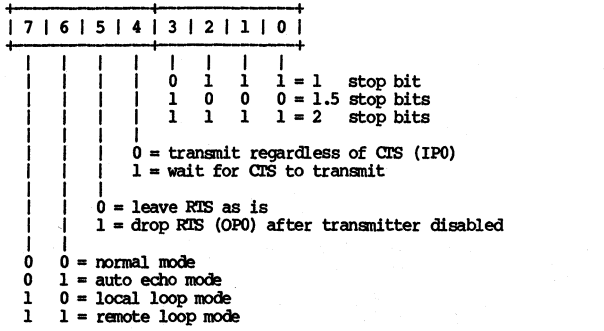
Register summary:

		READ	WRITE
8400	0FFB000	Mode Reg A (MRA)	Mode Reg A (MRA)
8402	0FFB002	Status Reg A (SRA	Clock Select Reg A (CSRA)
8404	0FFB004	—	Command Reg A (CRA)
8406	0FFB006	Rcv Hold Reg A (RHRA)	Transmit Hold Reg A (THRA)
8408	0FFB008	Input Port Change Reg (IPCR)	Aux Control Reg (ACR)
840A	0FFB00A	Interrupt Status Reg (ISR)	Interrupt Mask Reg (IMR)
8410	0FFB010	Mode Reg B (MRB)	Mode Register B (MRB)
8412	0FFB012	Status Reg B (SRB)	Clock Select Reg B (CSRB)
8414	0FFB014	—	Command Reg B (CRB)
8416	0FFB016	Rcv Hold Reg B (RHRB)	Transmit Hold Reg B (THRB)
841A	0FFB01A	Input Port Register (IPR)	Output Port Config Reg (OPCR)
841C	0FFB01C	—	Set Output Port Reg (OPR)
841E	0FFB01E	—	Reset Output Port Reg (OPR)
8420	0FFB020	Display Keyboard Status/Command Register	
8422	0FFB022	Display Keyboard Data I/O Register	

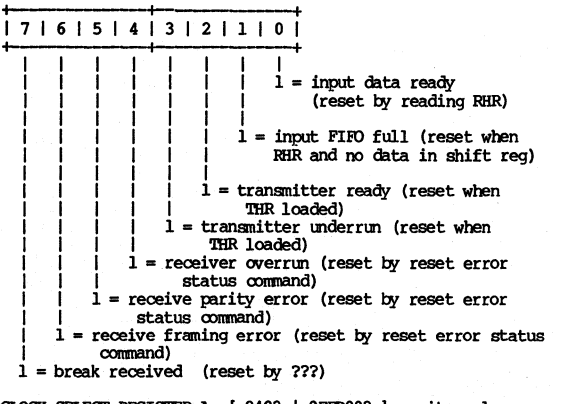
MODE REGISTER A, first access [8400 | 0FFB000]



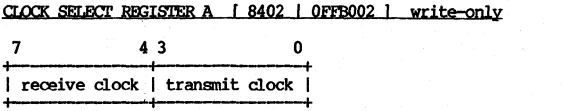
MODE REGISTER A, second and subsequent accesses
(until the mode register pointer is reset)



STATUS REGISTER A [8402 | 0FFB002] read-only

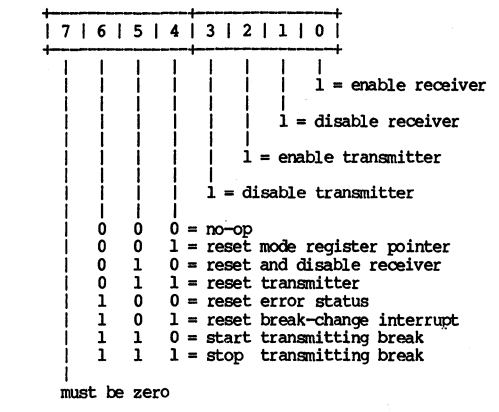


CLOCK SELECT REGISTER A [8402 | 0FFB002] write-only

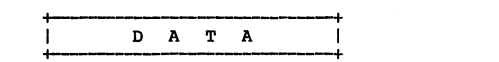


	ACR[7]=0	ACR[7]=1		ACR[7]=0	ACR[7]=1
0 -	50	75	8 -	2400	2400
1 -	110	110	9 -	4800	4800
2 -	134.5	134.5	A -	7200	1800(??)
3 -	200	150	B -	9600	9600
4 -	300	300	C -	38.4K	19.2K
5 -	600	600	D -	Timer	Timer
6 -	1200	1200	E -	IP4-16X	IP4-16X
7 -	1050	2000	F -	IP4-1X	IP4-1X

COMMAND REGISTER A [8404 | 0FFB004] write-only

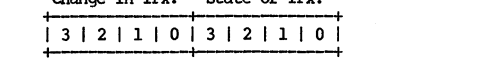


RECEIVE/TRANSMIT HOLDING REGISTER A [8406 | 0FFB006]



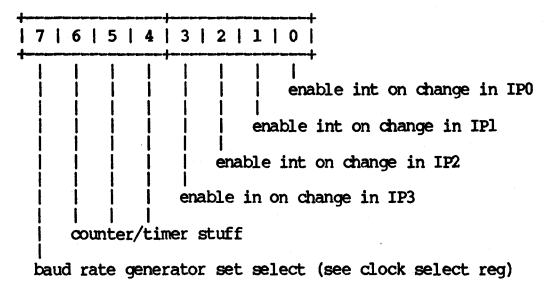
READ = top byte in input FIFO
WRITE = byte of data to transmit

INPUT PORT CHANGE REGISTER [8408 | 0FFB008] read-only

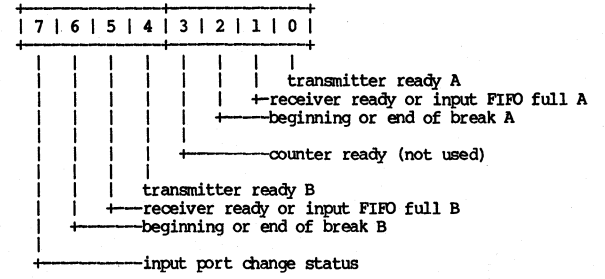


IP0 - Clear To Send (CTS) - A
IP1 - Clear To Send (CTS) - B
IP2 - Data Carrier Detect (DCD) - A
IP3 - Data Carrier Detect (DCD) - B

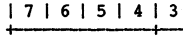
AUXILIARY CONTROL REGISTER [8408 | 0FFB008] write-only



INTERRUPT STATUS REGISTER [840A | 0FFB00A] read-only



INTERRUPT MASK REGISTER [840A | 0FFB00A] write-only



Each bit enables the corresponding bit
in the Interrupt Status Register.

MODE REGISTER B [8410 | 0FFB010]

See MODE REGISTER A.

STATUS/CLOCK SELECT REGISTER B [8412 | 0FFB012]

See STATUS/CLOCK SELECT REGISTER A.

COMMAND REGISTER B [8414 | 0FFB014]

See COMMAND REGISTER A.

RECEIVE/TRANSMIT HOLDING REGISTER B [8416 | 0FFB016]

See RECEIVE/TRANSMIT HOLDING REGISTER A.

INPUT PORT REGISTER [841A | 0FFB01A] read-only

IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0

IP0 - Clear to Send (CTS) - A
IP1 - Clear to Send (CTS) - B
IP2 - Data carrier Detect (DCD) - A
IP3 - Data carrier Detect (DCD) - B
IP4-IP7 - Undefined

OUTPUT PORT CONFIGURATION REGISTER [841A | 0FFB01A]

write-only

7	6	5	4	3	2	1	0

Load with zero. This selects:

OP0 = Ready to Send (RTS) A
OP1 = Ready to Send (RTS) B
OP2 = Data Terminal Ready (DTR) A
OP3 = Data Terminal Ready (DTR) B
OP4 - OP6 - Unused
OP7 = Speaker control

SET OUTPUT PORT REGISTER (OPR) [841C | 0FFB01C]

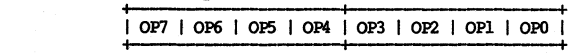
write-only

OP7	OP6	OP5	OP4	OP3	OP2	OP1	OP0

OP0 - Ready To Send (RTS) A
OP1 - Ready To Send (RTS) B
OP2 - Data Terminal Ready (DTR) A
OP3 - Data Terminal Ready (DTR) B
OP4-OP6 - Unused
OP7 - Turn off speaker

RESET OUTPUT PORT REGISTER (OPR) [841E | 0FFB01E]

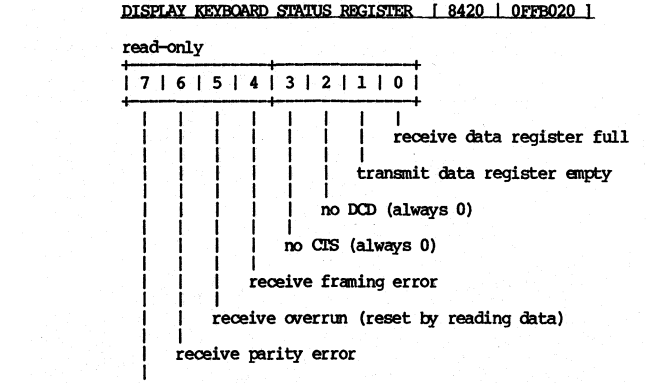
write-only



OP0 - Ready To Send (RTS) A
OP1 - Ready To Send (RTS) B
OP2 - Data Terminal Ready (DTR) A
OP3 - Data Terminal Ready (DTR) B
OP4-OP6 - Unused
OP7 - Turn on speaker

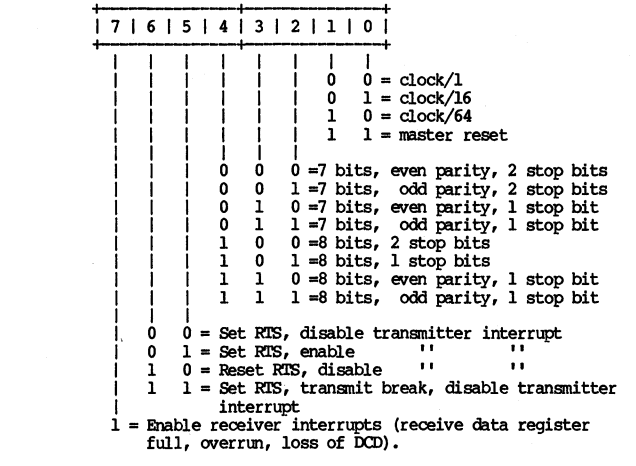
DISPLAY KEYBOARD STATUS REGISTER [8420 | 0FFB020]

read-only

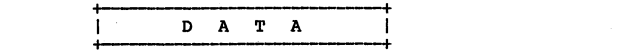


DISPLAY KEYBOARD COMMAND REGISTER [8420 | 0FFB020]

write-only



DISPLAY KEYBOARD DATA REGISTER [8422 | 0FFB022]



READ = empties receive data register
WRITE = loads transmit data register

CHAPTER 8

DN400,DN420,DN600

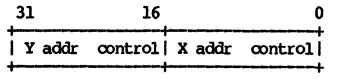
ADDRESS SPACE

physical		virtual
100400	traps	0
400	prom	400 (one-to-one)
80000	phys mem	80000->FFFFF
100800	phys mem	100800
100000	debug data	E00000
40000	disp2_mem	FA0000->FBFFFF
20000	disp1_mem	FC0000->FDFFFF
10000	multibus	FE0000->FEFFFF
E000	Color	FF6000
E400	Color	FF6400
E800	Color	FF6800
B000	PEB Ctl	FF7000
B400	FPU Cmd	FF7400
C000	FPU CS	FF7800
C400	Cache W 0	FF7C00
C800	Cache W 1	FF8000
FC00	Memory Ctl	FF9000
F400	disp 2	FF9400
F000	disp 1	FF9800
BC00	ring 2	FF9C00
B800	ring 1	FFA000
8C00	floppy	FFA800
8800	timers	FFAC00
8400	sios	FFB000
8000	mmu	FFB400
4000	pft	FFB800->FFF7FF
	io map	FFF800->FFF9FF

BLT REGISTERS

DN4xx

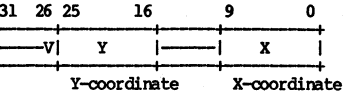
Source and Destination Control Register:



	Increment	Decrement
Y address control:	0202	0606
X address control:	0020	0060

		OS Virtual Address
(RCS)	Read source control reg	$FF988C
(WCS)	Write source control reg	$FF9884
(RCD)	Read destination control reg	$FF9888
(WCD)	Write destination control reg	$FF9880

Source/Destination Start/End Registers:

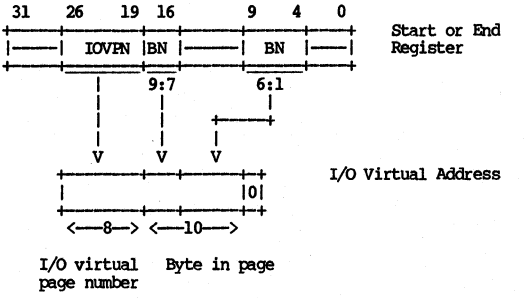


Y-coordinate X-coordinate (Low-order four bits are write only)

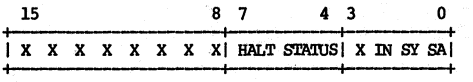
V - Used in BLTs to or from main memory

		OS Virtual Address
(RSS)	Read source start reg	$FF989C
(WSS)	Write source start reg	$FF98AC
(RES)	Read source end reg	$FF9894
(WES)	Write source end reg	$FF98B4
(RSD)	Read destination start reg	$FF9898
(WSD)	Write destination start reg	$FF98A8
(RED)	Read destination end reg	$FF9890
(WED)	Write destination end reg	$FF98B0

When start or end registers are used to specify locations in main memory, the correspondence is as follows:



DN6xx



IN = 1 Color display is executing an instruction queue starting at the address held in the instruction queue start location.

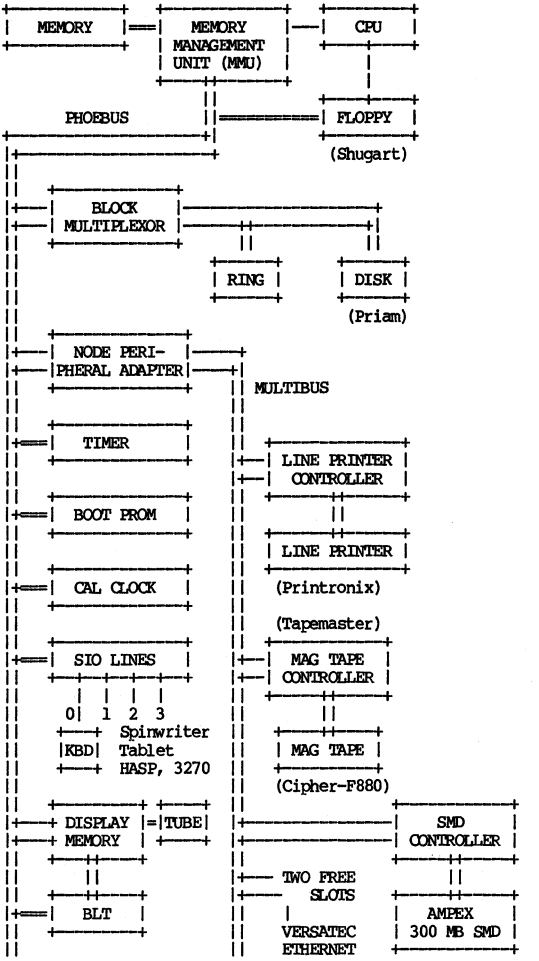
SY = 1 Color display is in system mode and will execute system functions.

SA Most significant address bit of all I/O addresses to be used by the color display for DMA.

HALT STATUS

0 0 0 0	Normal completion
0 0 0 1	Illegal opcode
0 0 1 0	Unimplemented instruction
0 0 1 1	BLT to main memory out of image memory region
1 1 1 1	Not halted

CONFIGURATION



DISPLAY BOARD JUMPERS

DN4xx

15 GREEN CRT:

W01-DOWN	W02-DOWN	W03-UP	W04-UP	W05-UP
W06-UP	W07-TOP-LEFT*	W08-UP	W09-UP	W10-UP
W11-UP	W12-UP	W13-LEFT	W14-RIGHT	W15-LEFT
W16-TOP-LEFT	W17-UP	W18-UP	W19-UP	W20-UP
W21-UP	W22-UP	W23-UP	W24-UP	W31-UP

15 BLACK & WHITE CRT:

W01-DOWN	W02-DOWN	W03-UP	W04-UP	W05-UP
W06-DOWN	W07-*	W08-UP	W09-UP	10-DOWN
W11-DOWN	W12-DOWN	W13-LEFT	W14-LEFT	W15-LEFT
W16-*	W17-DOWN	W18-UP	W19-DOWN	W20-DOWN
W21-UP	W22-OUT	W23-DOWN	W24-DOWN	W31-UP

19 BLACK & WHITE CRT

W01-DOWN	W02-UP	W03-UP	W04-DOWN	W05-DOWN
W06-DOWN	WO7- *	W08-DOWN	W09-DOWN	W10-DOWN
W11-DOWN	W12-DOWN	W13-RIGHT	W14-LEFT	W15-RIGHT
W16-*	W17-DOWN	W18-UP	W19-DOWN	W20-DOWN
W21-DOWN	W22-DOWN	W23-DOWN	W24-DOWN	W31-DOWN

* W07 AND W16 SHOULD BE SET BY SPECIFICATIONS ON CLOTH TAG, MOUNTED ON PCB.

DISPLAY BOARD 2		DISPLAY BOARD 1	
W25-DOWN	W26-DOWN	W25-RIGHT	W26-UP
W27-DOWN	W28-DOWN	W27-RIGHT	W28-UP
W29-DOWN	W30-DOWN	W29-UP	W30-UP

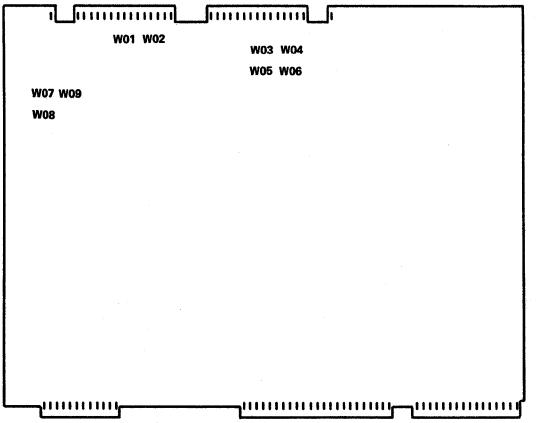
DN6xx

ARRAY BOARD JUMPER PLACEMENT:

ARRAY BOARD JUMPER PLACEMENT:

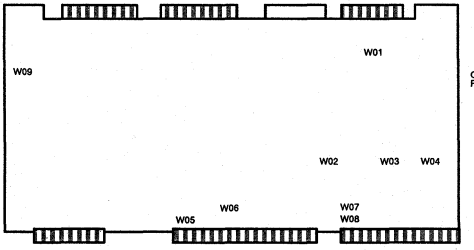
BOARD ORIENTATION: EJECTORS FACING UPWARDS

JUMP	W01	W02	W03	W04	W05	W06	W07	W08	W09
LOCT	~A08	~A09	~A11	~A12	~A11	~A12	C02-02	C02-08	C02-18
APR 1	LEFT	LEFT	UP	UP	UP	UP	UP	UP	OUT
APR 2	RIGHT	RIGHT	DOWN	DOWN	DOWN	DOWN	DOWN	UP	IN



CONTROL BOARD JUMPER PLACEMENT:

Control Board Jumper Placement:
Board orientation: ejectors facing upwards



Order of Boards:
Facing Front of Node

Control | Array 1 | Array 2

Jumper Orientation	W01	W02	W03	W04	W05	W06	W07	W08	W09
< Rev 12	Up	Down	Down	Down	Right	Up	Up	Down	None
>= Rev 12 800 Line	Down	Down	Down	Down	Right	Up	Up	Down	Down
1024 Line	Down	Down	Down	Down	Right	Up	Up	Down	Up

DISPLAY CONTROL AND STATUS REGISTER (DCSR)

DN4xx

[F000 | FF9800]

15	14	13	12	11	10	9	8
GO	x	x	x	x	x	x	NIL

7	6	5	4	3	2	1	0
FROMMM	TO MM	I EOF	IDONE	NONCONF	DECR	FILL	ON

R/W	BIT	NAME	
R/W	15	GO	- When set, this bit initiates a BLT operation. When cleared, this bit will abort any BLT in progress. When read, this bit will be set so long as a BLT operation is in progress.
W	8	NIL	- When set, enables non-interlaced mode. In this mode, only the odd lines are displayed at 60 frames per second.
W	7	FROM MM	- When set, enables BLTs from main memory as specified by the source box. This bit must be set in advance of the write operation that initiates a BLT. It should be cleared by a separate write operation after the BLT completes.
W	6	TO MM	- When set, enables BLTs to main memory as specified by the destination box.
W	5	IEOF	- When set, enables a 60 Hz interrupt request after the last line in the current field. This request must be acknowledged by reading from address DCSR + 2.
W	4	IDONE	- When set, enables interrupt request when BLT is done.
W	3	NON CONF	- When set, enables non-conforming BLT mode.

W	2	DECR	- Must be set when doing a BLT that decrements the X coordinate.
W	1	FILL	- When set, BLT fill mode is enabled.
W	0	ON	- When set, display is on; when reset, display is blanked.

DN6xx

CONTROL REGISTER [E000 | FF6000] (System)
[E400 | FF6400] (User)

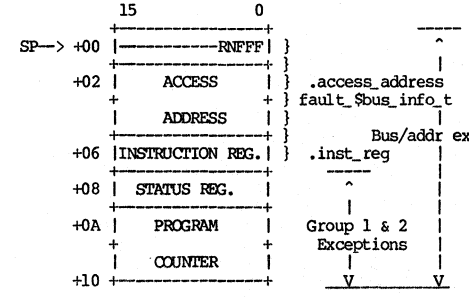
BIT	NAME	FUNCTION
15	Reserved	Set to zero
14	Reserved	Set to zero
13	Unused	
12	Unused	
11	Unused	
10	Unused	
9 - 4	WCSADH<5:0>+	During reads and writes to the control store through the control store window, these bits are used as the upper part of the control store address.
3	VIDENB+	This bit is used to enable or disable the video to the monitor.
2	RESET-	When this bit is asserted the micro machine is reset,the control store may be accessed through its window and no micro code is executed. The display will continue to be refreshed, no changes will be made to the display memory, and no software visible state will change.
1	CLKRUN+	When this bit is set the micro machine clocks will run.
0	CLKSTEP+	When the CLKRUN+ bit is not set the micro-machine clocks can be made to go through one clock cycle by toggling this bit low to high.

STATUS REGISTER

BIT	NAME	FUNCTION
15	DONE-	This bit is asserted by microcode to indicate that the micro-machine is ready to handle reads or writes to its command pages.
14	VBLANK-	When this bit is asserted, the display is currently in the vertical blanking period.
13	INST_DONE-	This bit is asserted by microcode to indicate that it is done executing an instruction queue.
12	CLKON+	This bit indicates that the micro-machine clocks are running.
11 - 0	NXTAD<11:0>-	These bits are the state of the next address bus at the end of the last micro cycle. This is needed in order to know the micro address when micro-code is being clock stepped.

FAULT FRAME

(type "fault_$bus_info_t" in fault.ins.pas)



R - Read operation (.write_op)
N - Not instruction reference (.not_inst)
FFF - Instruction function code (.function_code):
001 User data
010 User program
101 Supervisor data
110 Supervisor program
111 Interrupt acknowledge

FAULT TYPES

GROUP	EXCEPTION	PROCESSING
0	Reset Bus Error Address Error	Current instruction is aborted.
1	Trace Interrupt Illegal Ins. Privilege Ins.	Exception occurs before next instruction.
2	TRAP, TRAPV CHK Zero Divide	Processed by normal instruction execution.

Group 0 exceptions are highest priority.

FAULT VECTORS

Exception vector at [100400 | 0]

Vector	Address	Assignment
00	000	Reset: Initial SSP
	004	Reset: Initial PC
02	008	Bus Error
03	00C	Address Error
04	010	Illegal Instruction
05	014	Zero Divide
06	018	CHK Instruction
07	01C	TRAPV Instruction
08	020	Privilege Violation
09	024	Trace
0A	028	Unimplemented instruction
0B	02C	Unimplemented instruction
0C-0D	030	(Unassigned, reserved)
0F-17	03C	(Unassigned, reserved)
18	060	Spurious Interrupt
20-2F	080	TRAP Instruction Vectors
30-3F	0C0	(Unassigned, reserved)
40-8F	100	User Interrupt Vectors - unused
90-9F	240	Ring/disk board
A0-AF	280	User Interrupt Vectors - unused
B0	2C0	INT0/ -
B1	2C4	INT1/ -
B2	2C8	INT2/ -
B3	2CC	INT3/ - Tape Controller
B4	2D0	INT4/ - Storage Module
B5	2D4	INT5/ -
B6	2D8	INT6/ - Line Printer
B7	2DC	INT7/ - Parallel Input
B8-C3	2E0	User Interrupt Vectors -- unused
C4-CD	310	Unused
CE-CF	338	Color
D0-EF	340	SIO lines (even vectors only; odd vectors unused)
F0	3C0	P - ECCC (Automatic vectors)
F1	3C4	O -
F2	3C8	N - Display #2
F3	3CC	M - Floppy
F4	3D0	L - Display #1 (BLT)
F5	3D4	K -
F6	3D8	J -
F7	3DC	I -
F8-FA	3E0	Unused
FB	3EC	Timers 1,2,3
FC-FD	3F0	Unused
FE	3F8	CPU B-to-A
FF	3FC	ECCU

FLOATING-POINT FORMAT

Refer to the FLOATING-POINT FORMAT section of CHAPTER 7, DN300,DN320 for this information.

FLOPPY CONTROLLER

Floppy Status	[8C00 | FFA800]
Floppy I/O	[8C00 | FFA800]
Floppy DMA	[8C80 | FFA880]

Floppy Status Registers

Main Status Register

D7	D6	D5	D4	D3	D2	D1	D0

D7 - {RQM} Data register ready to send/receive data from Processor.
D6 - {DIO} I/O Direction, 1 = Data Reg to Processor, 0 = Processor to Data Reg
D5 - {EXM} Execution mode for non-DMA transfers
D4 - {CB} BUSY, a Read or Write command is in process
D3 - {D3B} FDD 3 is in Seek Mode
D2 - {D2B} FDD 2 is in Seek Mode
D1 - {D1B} FDD 1 is in Seek Mode
D0 - {D0B} FDD 0 is in Seek Mode

STATUS REGISTER 0

D7	D6	D5	D4	D3	D2	D1	D0

D7 - D6
 0 0 - Normal Termination of Command
 0 1 - Abnormal Term {command started, not completed}
 1 0 - Invalid Command{ command issued never started }
 1 1 - Abnormal Termination{ FDD went not ready during command execution }
D5 - Set when Seek Completed
D4 - Set if FDD issues Fault Signal or Track 0 Signal fails to occur after 77 Step Pulses.
D3 - Set if FDD Not Ready
D2 - Head Address
D1 - Unit Select 1 Status
D0 - Unit Select 0 Status

STATUS REGISTER 1

D7	D6	D5	D4	D3	D2	D1	D0

D7 - End of Cylinder
D6 - 0
D5 - Data Error
D4 - Overrun
D3 - 0
D2 - No Data
D1 - Not Writable
D0 - Missing Address Mark

STATUS REGISTER 2

D7	D6	D5	D4	D3	D2	D1	D0

D7 - 0
D6 - Control Mark
D5 - Data Error in Data Field
D4 - Wrong Cylinder
D3 - Scan Equal Hit
D2 - Scan NOT Satisfied
D1 - Bad Cylinder
D0 - Missing Address Mark in Address Field

STATUS REGISTER 3

D7	D6	D5	D4	D3	D2	D1	D0

D7 - Fault from FDD
D6 - Write Protected
D5 - Ready from FDD
D4 - Track 0
D3 - Two Sided
D2 - Head Address
D1 - Unit Select 1 Status
D0 - Unit Select 0 Status

I/O MAP

```
 15              0
+----------------+
|----PPPPPPPPPPPP|
+----------------+
```

P..P - Physical page number (PPN)
(type "ppn_t" in base.ins.pas)

I/O MAP at [-- | FFF800].
I/O MAP for DN460/660 at [9000 | FFFF800].

I/O map supplies physical addresses for non-CPU memory references (ring/disk, PNA, display, floppy).

See also I/O MAP in Peripheral I/O section, Chapter 6.

MEMORY CONTROL/STATUS REGISTERS (MCSR)

Board 1: [FC02 | FF9002]
Board 2: [FD02 | FF9102]
Board 3: [FE02 | FF9202]
Board 4: [FF02 | FF9302]

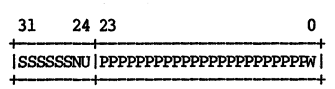
MCSR Control (Write-Only)

```
 15              0
+----------------+
|------------DCU|
+----------------+
```

D - Lock check bits on write (diag mode only)
C - Enable ECCC interrupts (through 3C0)
U - Enable ECCU interrupts (through 3FC)

Any write to this register acknowledges the interrupt.

MCSR Status Register (Read-Only)



S..S - Syndrome (valid only for ECCC error):

D0	data bit 00	70	data bit 08	2C	data bit 15
C8	" 01	68	" 09	F8	check bit 0
C4	" 02	58	" 10	F4	" 1
B0	" 03	54	" 11	EC	" 2
A8	" 04	4C	" 12	DC	" 3
A4	" 05	38	" 13	BC	" 4
94	" 06	34	" 14	7C	" 5
8C	" 07				

N - No error (0 => error detected, status valid)
U - Uncorrectable error (ECCU)
P..P - High-order 23 bits of physical address
(low-order bit is always 0)
W - Error during read-modify-write operation

MEMORY BOARD JUMPERS

Board #	JP3	JP4	Absolute Address	Mapped Address
1	in	in	fc02	ff9002
2	out	in	fd02	ff9102
3	in	out	fe02	ff9202
4	out	out	ff02	ff9302

Each board type has an address range that it can span specified by three jumpers; JP13, JP14, and JP15.

Memory Board Address Ranges

256Kb Board #	JP13	JP14	JP15	Address Range
1	in	out	out	100000-13ffff
2	out	out	out	140000-17ffff
3	in	out	in	180000-1bffff
4	out	out	in	1c0000-1fffff
3*	in	in	in	80000-bffff
4*	out	in	in	c0000-fffff

* Use these values if there are two 512KB boards or one 1Mb board in the system.

512KB Board #	JP13	JP14	JP15	Address Range
1	in	out	out	100000-17ffff
2	in	out	in	180000-1fffff
3	out	in	out	200000-27ffff
4	in	in	in	080000-0fffff

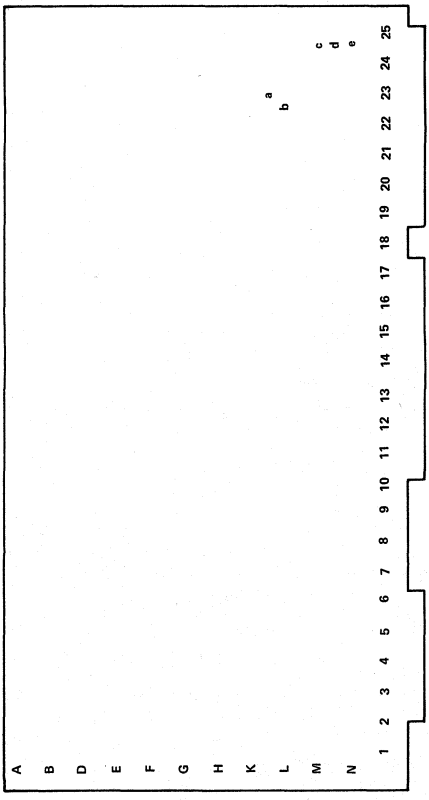
1MB Board #	JP13	JP14	JP15	Address Range
1	in	out	out	100000-1fffff
2	out	in	out	200000-2fffff
3	out	out	out	300000-3fffff
4	*** not allowed ***			

The memory test program checks that the tables above are satisfied.

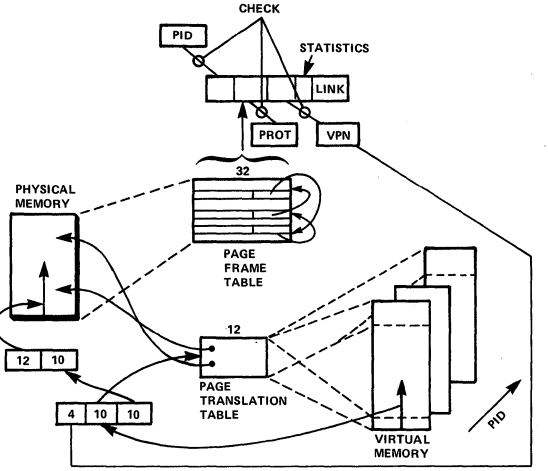
Size	256K	512K	1M	256K adr	512K adr	1M adr
1.0 M	0	0	1	–	–	1:100
2.0 M*	0	0	2	–	–	1:100 2:200
3.0 M*	0	0	3	–	–	1:100 2:200 3:300
0.5 M	0	1	0	–	1:100	–
1.5 M	0	1	1	–	1:100	2:200
2.5 M*	0	1	2	–	1:100	2:200 3:300
3.5 M*	0	1	3	–	4:080	1:100 2:200 3:300
1.0 M	0	2	0	–	1:100 2:180	–
2.0 M*	0	2	1	–	1:100 2:180	3:300
3.0 M*	0	2	2	–	1:100 4:080	2:200 3:300
1.5 M	0	3	0	–	1:100 2:180 3:180	–
2.5 M*	0	3	1	–	1:100 2:180 4:080	3:300
1.25 M	1	0	1	1:100	–	2:200
2.25 M	1	0	2	1:100	–	2:200 3:300
0.75 M	1	1	0	1:100	2:180	–
1.75 M	1	1	1	1:100	2:180	3:300
2.75 M	1	1	2	1:100	4:080	2:200 3:300
1.25 M	1	2	0	1:100	2:180 3:200	–
2.25 M	1	2	1	1:100	2:180 4:080	3:300
1.75 M	1	3	0	1:100	2:180 3:200 4:080	–
0.5 M	2	0	0	1:100 2:140	–	–
1.5 M	2	0	1	1:100 2:140	–	3:300
1.0 M	2	1	0	1:100 2:140	3:200	–
2.0 M	2	1	1	1:100 2:140	4:080	3:300
1.5 M	2	2	0	1:100 2:140	3:200 4:080	–
0.75 M	3	0	0	1:100 2:140 3:180	–	–
1.25 M	3	1	0	1:100 2:140 3:180	4:080	–
1.0 M	4	0	0	1:100 2:140 3:180	–	–

*Designates that the configuration requires a DOMAIN DN400 or DN420 cpu.

If you reconfigure memory boards, you must change the jumpers. Do not move any jumpers other than those attached at jumper points 3, 4, 13, 14, and 15.



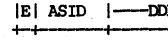
LABEL			COORDINATE LOCATION (APPROXIMATE)
a	=	JP3	L-23
b	=	JP4	L-23
c	=	JP13	M-25
d	=	JP14	M-25
e	=	JP15	N-25



PID/PRIV/POWER	[8000	FFB400]
CPU A Control	[8002	FFB402]
MMU Status, High	[8004	FFB404]
MMU Status, Low	[8006	FFB406]
Clear MMU Status	[8008	FFB408]
Bus Status	[800A	FFB40A]
Enable CPU B	[800E	FFB40E]

PID/PRIV Register

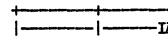
Address: [8000 | FFB400]



E - Enable power-off switch
DD - DOMAIN:
00 User domain 0 (least protected)
01 User domain 1
10 Supervisor domain 0
11 Supervisor domain 1 (most protected)
P - PT access enable
M - Enable MMU (virtual memory operations)

CPU A Control Register

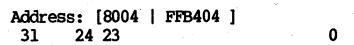
Address: [8002 | FFB402]



IA - Interrupt and/or abort:
00 CPU A will resume operation
01 CPU A will abort the attempted bus cycle
10 CPU A will receive level 6 interrupt

MMU Status Register

Address: [8004 | FFB404]



M - PFT miss (ASID and/or VPN not found)
A - Access fault (protection violation)
FFF - CPU function code:
001 User data reference
010 User procedure reference
101 Supervisor data reference
110 Supervisor procedure reference
111 Interrupt acknowledge
R - Read operation
V - MMU status register is valid
B - Miss occurred while CPU B running

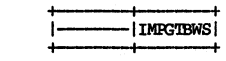
Clear MMU Status

Address: [8008 | FFB408]

When written, clears MMU status conditions.

Bus Status Register

Address: [800A | FFB40A]



I = 0 => unacknowledged interrupt pending
M - MMU enabled
P - PTT enabled
G - Big PFT (2K entries if not set, 4K if set)
T - Bus time-out (cleared by a read)
B - CPU B active
W - State of power-off switch (1 => ON position)
S - State of service switch (1 => Normal position)

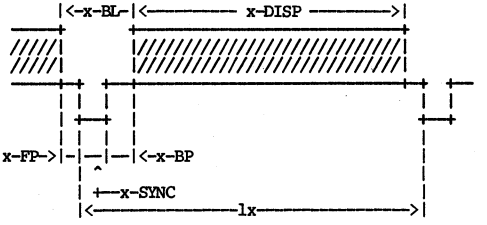
Enable CPU B Register

Address: [800E | FFB40E]

Any write will explicitly enable CPU B. Return to CPU A using Reset only.

MONITOR TIMING (DN6xx)

Axis	Item		Duration 800 Lines	Duration 1024 Lines
HORIZONTAL	1H	HORIZONTAL FREQUENCY	31.2 uSec @ 32.051 Hz	31.2 uSec @ 32.051 Hz
	H-FP	HORIZONTAL FRONT PORCH	0.7 uSec	0.7 uSec
	H-SYNC	HORIZONTAL SYNC	3.0 uSec	3.0 uSec
	H-BP	HORIZONTAL BACK PORCH	3.5 uSec	3.5 uSec
	H-BL	HORIZONTAL BLANKING	7.2 uSec	7.2 uSec
	H-DISP	HORIZONTAL DISPLAY AREA	24.0 uSec	24.0 uSec
VERTICAL	1V	VERTICAL FREQUENCY	13.2 mSec @ 75.70 Hz	16.7 mSec @ 60.0 Hz
	V-FP	VERTICAL FRONT PORCH	0.0	0.5 uSec
	V-SYNC	VERTICAL SYNC	93.6 uSec	93.6 uSec
	V-BP	VERTICAL BACK PORCH	600.0 uSec	616.0 uSec
	V-BL	VERTICAL BLANKING	693.6 uSec	710.0 uSec
	V-DISP	VERTICAL DISPLAY AREA	12.516 mSec	15.956 mSec



x = V or H

PAGE FRAME TABLE ENTRY (PFTE)

Refer to the PAGE FRAME TABLE ENTRY (PFTE) section of CHAPTER 7, DN300,DN320, for this information.

PAGE TRANSLATION TABLE ENTRY (PTTE)

Refer to the PAGE TRANSLATION TABLE ENTRY (PTTE) section of CHAPTER 7, DN300,DN320, for this information.

PEB

[B000 | FF7000]

CACHE Pages: [C400 | FF7C00]
[C800 | FF8000]

Definitions

peb_ctrl	FF7000	PEB control register
peb_status	FF7000	PEB status register (system space)
fpu_cmd	FF7400	FPU command register
fpu_status	FF77FC	FPU status register (user space)
fpu_cs_sa	FF7800	FPU control store SA
fpu_cs_ea	FF7C00	FPU control store EA
wcs_1k_size	1024*8	FPU control store bytes
wcs_4k_size	4096*8	FPU control store in bytes
fpu_defalth	00B2000E	FPU default microword, high 32
fpu_defaltl	500079E3	FPU default microword, low 32
wcserr_vec	F1<<2	WCS error interrupt vector
xcp_int_vec	2e0	FPU Exception Interrupt Vector
ld_fa	1*4+fpu_cmd	load FA SP
rd_fa	3*4+fpu_cmd	read FA SP
ld_fah	19*4+fpu_cmd	load FAH DP
ld_fal	20*4+fpu_cmd	load FAL DP
rd_fah	23*4+fpu_cmd	read FAH DP
rd_fal	24*4+fpu_cmd	read FAL DP
ld_fth	39*4+fpu_cmd	load FTL DP
dp_mul	40*4+fpu_cmd	load FTH and start a DP multiply to look at busy setting,clearing, reading
sp_div	17*4+fpu_cmd	load FTL sp div.
rd_exp	55*4+fpu_cmd	read FAM register as raw 32 bits
ld_exp	253*4+fpu_cmd	load FAM register as raw 32 bits and do NA<=EXP
xcp_reg	61*4+fpu_cmd	load FAM register as raw 32 bits
cache_sa	FF7C00	cache window start
cache_ea	FF8400	cache window end
memory_sa	120000	memory start
memory_ea	120800	memory end
tag_inval	800	invalid tag word
tag_valid_bit	11	

PEB Control Register Bits

peb_fpu_en	0001	FPU enable
peb_fpu_step	0002	FPU step
peb_fpu_reset	0004	FPU reset
peb_fpu_xie	0008	FPU interrupt enable
peb_fpu_csad	001F	upper control store address bits
peb_cache_a	0200	cache A (0050)
peb_cache_b	0400	cache B (0051)
peb_test_t	0800	test tag RAM (0052)
peb_test_d	1000	test data RAM (0053)
peb_cache_tpe	2000	cache tag parity enable (0054)
peb_cache_dpe	4000	cache data parity enable (0056)

Useful PEB Combinations

peb_cache_tta	peb_cache_a+peb_test_t	(0060)	test tag A
peb_cache_ttb	peb_cache_b+peb_test_t	(0061)	test tag B
peb_cache_tda	peb_cache_a+peb_test_d	(0062)	test data A
peb_cache_tdb	peb_cache_b+peb_test_d	(0063)	test data B
enab_tag_a	peb_cache_tta+peb_cache_tpe+peb_cache_dpe		(0065)
enab_tag_b	peb_cache_ttb+peb_cache_tpe+peb_cache_dpe		(0066)
enab_data_a	peb_cache_tda+peb_cache_dpe+peb_cache_tpe		(0067)
enab_data_b	peb_cache_tdb+peb_cache_dpe+peb_cache_tpe		(0068)
enab_cache_a	peb_cache_a+peb_cache_dpe+peb_cache_tpe		(0069)
enab_cache_b	peb_cache_b+peb_cache_dpe+peb_cache_tpe		(0070)
enab_cache	enab_cache_a+peb_cache_b (0071)		

PEB Status Register Bits

peb_fpu_cspe	0001	control store parity error
peb_cache_pe	0002	cache parity error
peb_fpu_xip	0004	FPU exception interrupt pending
peb_fpu_upc	07FF	FPU program counter bits
peb_fpu_busy	8000	FPU busy

PEB Commands

(from fpp.ins.pas)

FPP_$TSTX	Test S.P. or D.P. FAC for -,0,+; set cc's and D0.L then exit.
FPP_$SSTX	Same as FPP_$SSTA but exits when done (sets cc's).
FPP_$EXIT	Return to caller beginning with instruction in next word.
FPP_$SLV	FAC := (SP)+ (Single-Precision Load Value)
FPP_$SLA	FAC := ((SP)+) (Single-Precision Load using Address)
FPP_$SLC	FAC := Next four bytes in instruction stream (Constant)
FPP_$SSTA	((SP)+) := FAC (Single-Precision Store using Address)
FPP_$SAV	FAC := FAC + (SP)+ (Single-Precision Add Value)
FPP_$SAA	FAC := FAC + ((SP)+)
FPP_$SAC	FAC := FAC + Next four bytes (Constant)

```
FPP_$SSV      FAC := FAC - (SP)+  (Single-Precision Subtract
                                  Value)
FPP_$SSA      FAC := FAC - ((SP)+)
FPP_$SSC      FAC := FAC - Next four bytes (Constant)
FPP_$SISV     FAC := (SP)+ - FAC  (Single-Precision Inverse
                                  Subtract Value)
FPP_$SISA     FAC := ((SP)+) - FAC
FPP_$SISC     FAC := Next four bytes (Constant) - FAC
FPP_$SMV      FAC := FAC * (SP)+  (Single-Precision Multiply
                                  Value)
FPP_$SMA      FAC := FAC * ((SP)+)
FPP_$SMC      FAC := FAC * Next four bytes (Constant)
FPP_$SDV      FAC := FAC / (SP)+  (Single-Precision Divide Value)
FPP_$SDA      FAC := FAC / ((SP)+)
FPP_$SDC      FAC := FAC / Next four bytes (Constant)
FPP_$SIDV     FAC := (SP)+ / FAC  (Single-Precision Inverse Divide
                                  Value)
FPP_$SIDA     FAC := ((SP)+) / FAC
FPP_$SIDC     FAC := Next four bytes (Constant) / FAC
FPP_$SCVX     Compare FAC : (SP)+; set cc's & D0.L then exit
FPP_$SCAX     Compare FAC : ((SP)+); set cc's & D0.L then exit
FPP_$SCCX     Compare FAC : Next 4 bytes; set cc's & D0.L then
                            exit
FPP_$SLWV     FAC := Float[(SP)+]     (Float 16 bit integer)
FPP_$SLWA     FAC := Float[((SP)+)]   (Float 16 bit integer)
FPP_$SLLV     FAC := Float[(SP)+]     (Float 32 bit integer)
FPP_$SLLA     FAC := Float[((SP)+)]   (Float 32 bit integer)
FPP_$SSTWX    D0.W := Fix[FAC]; then exit with cc's set
FPP_$SSTLX    D0.L := Fix[FAC]; then exit with cc's set
FPP_$NEG      FAC := -FAC (Single- or Double-Precision)
FPP_$ABS      FAC := Abs[FAC]    (Absolute value Single- or Double-
                                  Precision)
FPP_$DLA      D.P. FAC := ((SP)+) (Double-Precision Load using
                                  Address)
FPP_$DLC      D.P. FAC := Next 8 bytes (Constant)
FPP_$DSTA     D.P. ((SP)+) := FAC (Double-Precision Store using
                                  Address)
FPP_$DSTX     D.P. Same as FPP_$DSTA but exits when done
               (sets cc's)
FPP_$DAA      D.P. FAC := FAC + ((SP)+)
FPP_$DAC      D.P. FAC := FAC + Next 8 bytes (Constant)
FPP_$DSA      D.P. FAC := FAC - ((SP)+)
FPP_$DSC      D.P. FAC := FAC - Next 8 bytes (Constant)
FPP_$DISA     D.P. FAC := ((SP)+) - FAC (Inverse Subtract)
FPP_$DISC     D.P. FAC := Next 8 bytes (Constant) - FAC
FPP_$DMA      D.P. FAC := FAC * ((SP)+)
FPP_$DMC      D.P. FAC := FAC * Next 8 bytes (Constant)
FPP_$DDA      D.P. FAC := FAC / ((SP)+)
FPP_$DDC      D.P. FAC := FAC / Next 8 bytes (Constant)
FPP_$DIDA     D.P. FAC := ((SP)+) / FAC (Inverse Divide)
FPP_$DIDC     D.P. FAC := Next 8 bytes (Constant) / FAC
FPP_$DCAX     D.P. Compare FAC : ((SP)+); set cc's and D0.L then
               exit
FPP_$DCCX     D.P. Compare FAC : Next 8 bytes; set cc's and D0.L
               then exit
FPP_$DLWV     D.P. FAC := Float[(SP)+]     (Float 16-bit integer)
```

```
FPP_$DLWA     D.P. FAC := Float[((SP)+)]  (Float 16-bit integer)
FPP_$DLLV     D.P. FAC := Float[(SP)+]    (Float 32-bit integer)
FPP_$DLLA     D.P. FAC := Float[((SP)+)]  (Float 32-bit integer)
FPP_$DSIWX    D.P. D0.W := Fix[FAC]; then exit
FPP_$DSILX    D.P. D0.L := Fix[FAC]; then exit
FPP_$SCNV     Convert D.P. FAC to Single-Precision
FPP_$DCNV     Convert Single-Precision FAC to D.P.
FPP_$SSQR     Take square root of FAC
FPP_$DSQR     Take square root of DFAC
FPP_$SEXP     EXP(<FAC>)
FPP_$DEXP     DEXP(<DFAC>)
FPP_$SLOG     ALOG(<FAC>)
FPP_$DLOG     DLOG(<DFAC>)
FPP_$SSIN     SIN(<FAC>)
FPP_$DSIN     DSIN(<DFAC>)
FPP_$SCOS     COS(<FAC>)
FPP_$DCOS     DCOS(<DFAC>)
FPP_$STAN     TAN(<FAC>)
FPP_$DTAN     DTAN(<DFAC>)
FPP_$SATAN    ATAN(<FAC>)
FPP_$DATEN    DATAN(<DFAC>)}
FPP_$SATAN2A  ATAN2(<FAC>,((sp+))
FPP_$SATAN2V  ATAN2(<FAC>,(sp+)
FPP_$SATAN2C  ATAN2(<FAC>,<CONST>)
FPP_$DATAN2A  DATAN2(<DFAC>,((sp+))
FPP_$DATAN2C  DATAN2(<DFAC>,<CONST>)
FPP_$E$21V    E$21(<FAC>,(sp)+)
FPP_$E$22A    E$22(<FAC>,((sp)+))
FPP_$E$22V    E$22(<FAC>,(sp)+)
FPP_$E$22C    E$22(<FAC>,<CONST>)
FPP_$E$61V    E$61(<DFAC>,(sp)+)
FPP_$E$62A    E$62(<DFAC>,((sp)+))
FPP_$E$62V    E$62(<DFAC>,(sp)+)
FPP_$E$62C    E$62(<DFAC>,<CONST>)
FPP_$E$66A    E$66(<DFAC>,((sp)+))
FPP_$E$66C    E$66(<DFAC>,<CONST>)
FPP_$STRUNC   FAC := Int[FAC]
FPP_$SNINT    FAC := Nint[FAC] (Nearest integer)
FPP_$DTRUNC   D.P. FAC := Int[FAC]
FPP_$DNINT    D.P. FAC := Nint[FAC] (Nearest integer)
FPP_$SMINV    FAC := Min[FAC, (SP)+]
FPP_$SMINA    FAC := Min[FAC, ((SP)+)]
FPP_$SMINC    FAC := Min[FAC, Next 4 bytes]
FPP_$SMAXV    FAC := Max[FAC, (SP)+]
FPP_$SMAXA    FAC := Max[FAC, ((SP)+)]
FPP_$SMAXC    FAC := Max[FAC, Next 4 bytes]
FPP_$DMINA    D.P. FAC := Min[FAC, ((SP)+)]
FPP_$DMINC    D.P. FAC := Min[FAC, Next 8 bytes]
FPP_$DMAXA    D.P. FAC := Max[FAC, ((SP)+)]
FPP_$DMAXC    D.P. FAC := Max[FAC, Next 8 bytes]
FPP_$CLA      Complex FAC := ((SP)+)
FPP_$CLC      Complex FAC := Next 8 Bytes
FPP_$CAA      Complex FAC := FAC + ((SP)+)
FPP_$CAC      Complex FAC := FAC + Next 8 bytes
FPP_$CSA      Complex FAC := FAC - ((SP)+)
FPP_$CSC      Complex FAC := FAC - Next 8 bytes
```

```
FPP_$CISA     Complex FAC := ((SP)+) - FAC
FPP_$CISC     Complex FAC := Next 8 bytes - FAC
FPP_$CMA      Complex FAC := FAC * ((SP)+)
FPP_$CMC      Complex FAC := FAC * Next 8 bytes
FPP_$CDA      Complex FAC := FAC / ((SP)+)
FPP_$CDC      Complex FAC := FAC / Next 8 bytes
FPP_$CIDA     Complex FAC := ((SP)+) / FAC
FPP_$CIDC     Complex FAC := Next 8 bytes / FAC
FPP_$CSTA     Complex Store FAC through (SP)+
FPP_$CSTX     Complex Store FAC through (SP)+ and exit
FPP_$CSWAP    Complex Exchange Real and Imaginary parts of FAC
FPP_$CCNV     Convert Single-Precision to Complex (set imagFAC := 0)
FPP_$CCONJ    Complex Conjugate (imagFAC := -imagFAC)
```

RING/DISK

address: Controller #2 [BC00 | FF9C00]
Controller #1 [B800 | FFA000]

(today's Controllers are #2's)

Address	Write function	Read function
00BC00	Cylinder MSByte	Cylinder MSByte
00BC01		-
00BC02	Cylinder LSByte	Cylinder LSByte
00BC03		-
00BC04	Disk Command Register	Disk Status Register
00BC05		-
00BC06	(reserved hole)	-
00BC07		-
00BC08	Head Number	-
00BC09		-
00BC0A	Sector	Node ID 1
00BC0B		
00BC0C	Controller Command	Node ID 2
00BC0D		
00BC0E	(reserved hole)	Node ID 3
00BC0F		-
00BC10	Network Type Mask	Disk status MSB
00BC11		Disk status LSB
00BC12	Network Receive Command	Network rcv stat MSB
00BC13		Network rcv stat LSB
00BC14	Network Transmit Command	Network tx status MSB
00BC15		Network tx status LSB
00BC16		
00BC17		-
00BC18	Disk Interrupt ACK	-
00BC19		-
00BC1A	Network Trans Interrupt ACK	-
00BC1B		-
00BC1C	Network Rec Interrupt ACK	-
00BC1D		-
00BC1E		-
00BC1F		-

Address	Write function	Read function
00BC40	Disk DMA Address 0	Disk DMA Address 0
00BC41		-
00BC42	Disk DMA Count 0	Disk DMA Count 0
00BC43		-
00BC44	Disk DMA Address 1	Disk DMA Address 1
00BC45		-
00BC46	Disk DMA Count 1	Disk DMA Count 1
00BC47		-
00BC48	Transmit DMA Address	Transmit DMA Address 0
00BC49		-
00BC4A	Transmit DMA Count 0	Transmit DMA Count 0
00BC4B		-
00BC4C	Transmit DMA Address	Transmit DMA Address 1
00BC4D		-
00BC4E	Transmit DMA Count 1	Transmit DMA Count 1
00BC4F		-
00BC50	DMA Command	DMA Status
00BC51		-
00BC52	DMA Request	-
00BC53		-
00BC54	DMA Single mask	-
00BC55		-
00BC56	DMA Mode	-
00BC57		-
00BC58	DMA Clear Byte Pointer	DMA Temporary
00BC59		-
00BC5A	DMA Master Clear	-
00BC5B		-
00BC5C	-	-
00BC5D		-
00BC5E	DMA All Masks	-
00BC5F		-

Address	Write function	Read function
00BC60	Receive 0 DMA Address	Receive 0 DMA Address 0
00BC61		-
00BC62	Receive 0 DMA Count 0	Receive 0 DMA Count 0
00BC63		-
00BC64	Receive 0 DMA Address	Receive 0 DMA Address 1
00BC65		-
00BC66	Receive 0 DMA Count 1	Receive 0 DMA Count 1
00BC67		-
00BC68	Receive 1 DMA Address	Receive 1 DMA Address 0
00BC69		-
00BC6A	Receive 1 DMA Count 0	Receive 1 DMA Count 0
00BC6B		-
00BC6C	Receive 1 DMA Address	Receive 1 DMA Address 1
00BC6D		-
00BC6E	Receive 1 DMA Count 1	Receive 1 DMA Count 1
00BC6F		-
00BC70	DMA Command	DMA Status
00BC71		-
00BC72	DMA Request	-
00BC73		-
00BC74	DMA Single mask	-
00BC75		-
00BC76	DMA Mode	-
00BC77		-
00BC78	DMA Clear Byte Pointer	DMA Temporary
00BC79		-
00BC7A	DMA Master Clear	-
00BC7B		-
00BC7C	-	-
00BC7D		-
00BC7E	DMA All Masks	-
00BC7F		-

Cylinder Address Register

MSB address: [BC00 | FF9C00]

15	14	13	12	11	10	9	8
0	0	0	0	0	C10	C09	C08

LSB address: [BC02 | FF9C02]

15	14	13	12	11	10	9	8
C07	C06	C05	C04	C03	C02	C01	C00

PRIAM Command Register

address: [BC04 | FF9C04]

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0

Sequence Up =		0	0	1
Sequence Down =		0	1	0
Restore =		0	1	1
Seek =		1	0	0
Fault Reset =		1	0	1

PRIAM Status Register

address: [BC04 | FF9C04]

15	14	13	12	11	10	9	8
CMDRJT	WRITPT	DRVFLT	BUSY	CYL 0	SEEKFT	SEEKOK	READY

READY - The drive is up to speed, servo is locked on track, and the unit is in a state to read or write.

SEEKOK- SEEK COMPLETE - This bit indicates the seek has completed successfully.

SEEKFT- SEEK FAULT - A fault was detected during a seek operation.

CYL 0 - Head on cylinder 0.

BUSY - The drive is in the process of executing a command and will not accept any other commands.

DRVFLT- DRIVE FAULT - A fault was detected during a write operation or a drive unsafe condition was detected.

WRITPT- WRITE PROTECT - The head selected is write protected. Write protection is set by switches in the drive or when the drive isn't sequenced up.

CMDRJT- Control or register load command received while drive is not ready, or improper command received.

Head/Drive Select
address: [BC08 | FF9C08]

15	14	13	12	11	10	9	8
SEL3\	SEL2\	SEL1\	SEL0\	x	H2\	H1\	H0\

SEL3-SEL0 select the DRIVE:

Drive #0 =	1	1	1	0
1 =	1	1	0	1
2 =	1	0	1	1
3 =	0	1	1	1

H2-H0 select the HEAD:

Head #0 =	1	1	1
1 =	1	1	0
2 =	1	0	1

Sector Select
address: [BC0A | FF9C0A]

15	14	13	12	11	10	9	8
x	x	x	S4	S3	S2	S1	S0

S4-S0 select the SECTOR(0-17):

Sector #0 =	0	0	0	0	0
1 =	0	0	0	0	1
2 =	0	0	0	1	0
3 =	0	0	0	1	1

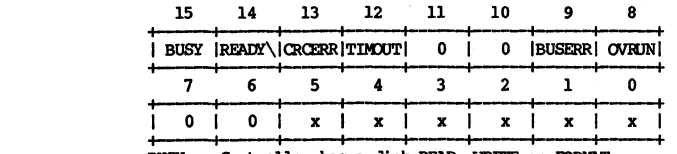
etc.

Disk Controller Command
address: [BC0C | FF9C0C]

15	14	13	12	11	10	9	8
READ	WRITE	FORMAT	x	x	x	x	INTE

Disk Controller Status

address: [BC10 | FF9C10]



BUSY - Controller has a disk READ, WRITE or FORMAT request pending or in progress.

READY - PRIAM ready bit, indicates that the PRIAM is sequenced up, on cylinder, and ready to accept READS, WRITES, or FORMATS.

CRCERR- CRC on the last disk operation was in error. This bit is undefined except for disk READs.

TIMOUT- The last disk operation didn't finish before three revolutions of the disk. During a READ or WRITE operation it is an indication that the drive has been positioned to the wrong cylinder, the sector number is too large, surface is unformatted, or there was a CRC error on the disk header.

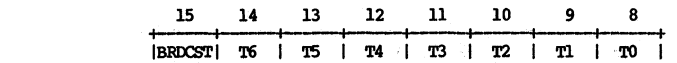
BUSERR- A BUS error occurred during the last DMA transfer. The DMA address register points past the address with the error.

OVRUN - DMA overrun occurred during last disk operation.

Status	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Controller busy	1	x	x	x	0	0	x	x	0	0	x	x	x	x	x	x
PRIAM not ready	x	1	x	x	0	0	x	x	0	0	x	x	x	x	x	x
Time-out sector error	0	0	x	1	0	0	0	0	0	0	x	x	x	x	x	x
DMA bus error	0	0	x	0	0	0	1	x	0	0	x	x	x	x	x	x
DMA overrun	0	0	x	0	0	0	0	1	0	0	x	x	x	x	x	x
CRC error	0	0	1	0	0	0	0	0	0	0	x	x	x	x	x	x
Disk operation OK	0	0	0	0	0	0	0	0	0	0	x	x	x	x	x	x

Packet Type

address: [BC10 | FF9C10]



BRDCST - Accept broadcast messages

T6-T0 - Type mask; accept message if any bits in type field of message are set in corresponding bit of type mask.

Receive Command Register

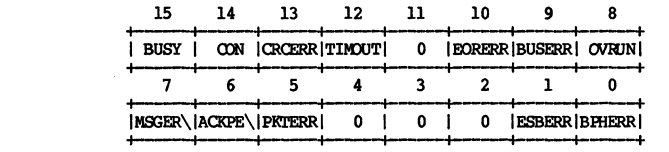
address: [BC12 | FF9C12]

15	14	13	12	11	10	9	8
REC	STOP	CON	DIS	x	x	x	INTE

REC - Enable receive. This informs the controller that the registers in the controller are set up to receive a message.
STOP - Abort a previously posted REC. This will be successful if the controller has not already seen a message begin.
CON - Connect to the ring.
DIS - Disconnect from the ring. This also happens on reset.
INTE - Enable interrupts to happen after a receive.

Receive Status Register

address: [BC12 | FF9C12]



BUSY - The controller is currently observing a message or a request is still pending.

CON - The controller is currently connected to the ring.

CRCERR - The last message received had a CRC error.

TIMOUT - The last message received started but didn't finish in 2**12 byte times.

EORERR - End of range error. One or both of the message fields was bigger than the DMA channel was set up for.

BUSERR - A BUS error occurred during the DMA transfer. The DMA address register is pointing one location past the point of the error.

OVRUN - A DMA overrun occurred during the last receive.

MSGER - No message error occurred during the last receive. A message error is any error that can be detected by the microcode of the controller Generally it checks for the packet protocol.

ACKPE - Acknowledge parity OK. No parity error was discovered in the acknowledge bytes in the last receive.

PKTERR - Either the transmitter or another receiver had an error in the packet. If the transmitter had an error in the transmission of the packet, one of the following errors occurred in the transmitter:

ESBERR - Elastic Store Buffer Error
BPHERR - Bi-Phase Error
OVRUN - DMA Overrun
BUSERR - DMA Bus Error
ACKPAR - Ack Byte Parity Error
SFTABT - Software Abort
NCOPY - No Receiver Enabled to Copy this Message
MSGERR - Message Error

If the receiver had an error in the reception of the packet, one of the following errors occurred in the receiver:

ESBERR - Elastic Store Buffer Error
BPHERR - Bi-Phase Error
OVRUN - DMA Overrun
BUSERR - DMA Bus Error
EORERR - End of Range Error in DMA Channel
CRCERR - CRC Error in Packet
ACKPAR - Ack Byte Parity Error

ESBERR - An error occurred in the modem's elastic store buffer.

BPHERR - An error occurred in the modem's decode of the Bi-phase encoded data.

Status	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Controller Busy	1	x	x	x	0	x	x	x	x	x	x	0	0	0	x	x
Connected to Ring	x	1	x	x	0	x	x	x	x	x	x	0	0	0	x	x
DMA Bus Error	0	x	x	x	0	x	1	x	x	x	x	0	0	0	x	x
DMA Overrun	0	x	x	x	0	x	0	1	x	x	x	0	0	0	x	x
Bi-Phase Error	0	1	x	x	0	x	0	0	x	x	x	0	0	0	x	1
Elastic Buffer Error	0	1	x	x	0	x	0	0	x	x	x	0	0	0	1	0
Time-out Error	0	1	x	1	0	x	0	0	x	x	x	0	0	0	0	0
Sync Protocol Error	0	1	x	0	0	x	0	0	0	x	x	0	0	0	0	0
Ack Byte Parity Error	0	1	x	0	0	x	0	0	1	0	x	0	0	0	0	0
CRC Error	0	1	1	0	0	x	0	0	1	1	x	0	0	0	0	0
End of Range Error	0	1	0	0	0	1	0	0	1	1	x	0	0	0	0	0
Packet Error	0	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0
Received OK	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Transmit Command Register

address: [BC14 | FF9C14]

15	14	13	12	11	10	9	8
TMT	STOP	FTRANS	CH1DIS	x	DELAY	NDELAY	INTE

TMT - Enable transmit. This bit informs the controller that all the registers are set up for a message to be transmitted when the next token is received on the network.

STOP - Stop a previously posted TMT. This will abort a request for a transmit. It will abort a packet if currently being transmitted.

FTRANS - Force transmit. Allow the TMT to start even though no token has been seen. This bit must be accompanied by the TMT bit.

CH1DIS - Disable the second transmit DMA channel from putting out any data. This will make for a zero length data field in the message.

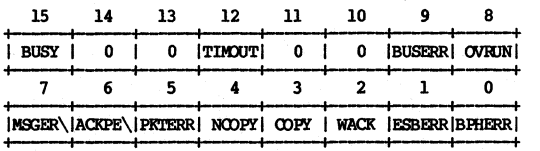
DELAY - Enable an additional seven-bit delay into the length of the network. This may be required to support the recirculation of the token, which is nine bits.

NDELAY - Disable the seven-bit delay.

INTE - Enable an interrupt to be generated at the completion of the transmit.

Transmit Status Register

address: [BC14 | FF9C14]



BUSY - The controller is currently transmitting a message or the request is still pending.

TIMOUT - The last message transmitted started but didn't finish in 2**12 byte times.

BUSERR - A BUS error occurred during the DMA transfer. The DMA address register is pointing one location past the point of the error.

OVRUN - A DMA overrun occurred during the last receive. A message error is any error that can be detected by the microcode of the controller. It is generally the packet protocol that is checked for.

MSGER\ - No message error occurred during the last receive. A message error is any error that can be detected by the microcode of the controller. Generally it checks for the packet protocol.

ACKPE\ - Ack parity OK. No parity error was discovered in the ack bytes in the last transmit.

PKTERR - Either the transmitter or the receiver had an error in the packet.

If the transmitter had an error in the transmission of the packet, one of the following errors occurred in the transmitter:

ESBERR - Elastic Store Buffer Error
BPHERR - Bi-Phase Error
OVRUN - DMA Overrun
BUSERR - DMA Bus Error
ACKPAR - Ack Byte Parity Error
SFTABT - Software Abort
NCOPY - No Receiver was Enabled to Copy this Message
MSGERR - Message Error

If the receiver had an error in the reception of the packet, one of the following errors occurred in the receiver:

ESBERR - Elastic Store Buffer Error
BPHERR - Bi-Phase Error
OVRUN - DMA Overrun
BUSERR - DMA Bus Error
EORERR - End of Range Error in DMA Channel
CRCERR - CRC Error in Packet
ACKPAR - Ack Byte Parity Error

NCOPY - No receiver observed his node address in this packet and/or was enabled to copy this message.
MSGCPY - The receiver successfully copied the message.
WACK - A receiver observed his node ID, but wasn't enabled to copy this message.
ESBERR - An error occurred in the modem's elastic store buffer.
BPHERR - An error occurred in the modem's decode of the Bi-phase encoded data.

Status	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Controller busy	1	0	0	x	0	0	x	x	x	x	x	x	x	x	x	x
DMA Bus Error	0	0	0	x	0	0	1	x	x	x	x	x	x	x	x	x
DMA Overrun	0	0	0	x	0	0	0	1	x	x	x	x	x	x	x	x
Bi-Phase Error	0	0	0	x	0	0	0	0	x	x	x	x	x	x	x	1
Elastic Buffer Error	0	0	0	x	0	0	0	0	x	x	x	x	x	x	1	0
Time-out Error	0	0	0	1	0	0	0	0	x	x	x	x	x	x	0	0
Sync Protocol Error	0	0	0	0	0	0	0	0	0	x	x	x	x	x	0	0
Ack Byte Parity Error	0	0	0	0	0	0	0	0	1	0	x	x	x	x	0	0
Negative Ack	0	0	0	0	0	0	0	0	1	1	x	1	0	0	0	0
Wack	0	0	0	0	0	0	0	0	1	1	x	x	x	1	0	0
Packet Error	0	0	0	0	0	0	0	0	1	1	1	0	x	x	0	0
Message Copied	0	0	0	0	0	0	0	0	1	1	0	0	1	x	0	0

Node ID Register
address: [BC0A | FF9C0A]
[BC0C | FF9C0C]
[BC0E | FF9C0E]

15	14	13	12	11	10	9	8
ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0

DMA Control/Status Registers

DMA Address address: [base + 0]
['' + 4]
['' + 8]
['' + C]

15	14	13	12	11	10	9	8
A08	A07	A06	A05	A04	A03	A02	A01
IVA16	IVA15	IVA14	IVA13	IVA12	IVA11	IVA10	A09

Notice that the address is shifted right by one.

DMA Count address: [base + 2]
['' + A]
['' + E]

15	14	13	12	11	10	9	8
C07	C06	C05	C04	C03	C02	C01	C00
C15	C14	C13	C12	C11	C10	C09	C08

C(15-00) is the desired count in words minus one.

DMA Command Register

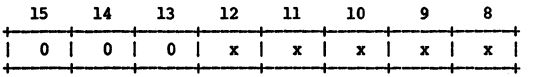
address: [base + 10]

15	14	13	12	11	10	9	8
0	1	1	0	0	x	0	0

BIT 7 - DACK Sense Active High
6 - DREQ Sense Active Low
5 - Extended Write
4 - Rotating Priority
3 - Compressed Timing
2 - Controller Disable
1 - Channel 0 Address Hold Enable
0 - Memory to Memory Enable

DMA Mode Register

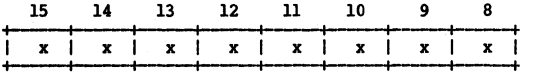
address: [base + 16]



BIT 7,6 - 00 - Demand Mode
01 - Single Mode
10 - Block Mode
11 - Cascade Mode
5 - Address Decrement
4 - Autoinitialize
3,2 - 00 - Verify transfer
01 - Write transfer
10 - Read transfer
11 - undefined
1,0 - Channel Select

DMA Request Register

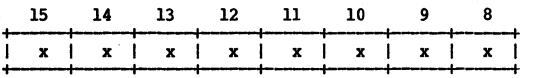
address: [base + 12]



BIT 2 - Request Bit
1,0 - Channel Select

DMA Mask Register

address: [base + 14]



BIT 2 - Set Mask Bit
1,0 - Channel Select

DMA ALL Mask Register

address: [base + 1E]

15	14	13	12	11	10	9	8
x	x	x	x	x	x	x	x

BIT 3 - Set Channel 3 Mask Bit
2 - Set Channel 2 Mask Bit
1 - Set Channel 1 Mask Bit
0 - Set Channel 0 Mask Bit

DMA Status Register

address: [base + 10]

15	14	13	12	11	10	9	8

BIT 7 - Channel 3 Request
6 - Channel 2 Request
5 - Channel 1 Request
4 - Channel 0 Request
3 - Channel 3 has Reached TC
2 - Channel 2 has Reached TC
1 - Channel 1 has Reached TC
0 - Channel 0 has Reached TC

```
Data Input/Output, Line 0 [ 8400 | FFB000 ]
Control/Status,    Line 0 [ 8402 | FFB002 ]
Data Input/Output, Line 1 [ 8404 | FFB004 ]
Control/Status,    Line 1 [ 8406 | FFB006 ]
Data Input/Output, Line 2 [ 8408 | FFB008 ]
Control/Status,    Line 2 [ 840A | FFB00A ]
Data Input/Output, Line 3 [ 840C | FFB00C ]
Control/Status,    Line 3 [ 840E | FFB00E ]
Bit-Rate Generator,Line 0 [ 8482 | FFB082 ]
Bit-Rate Generator,Line 1 [ 8486 | FFB086 ]
Bit-Rate Generator,Line 2 [ 848A | FFB08A ]
Bit-Rate Generator,Line 3 [ 848E | FFB08E ]
```

Data 03	02	01	00	Bit-Rate
0	0	0	0	50
0	0	0	1	75
0	0	1	0	110
0	0	1	1	134.5
0	1	0	0	150
0	1	0	1	300
0	1	1	0	600
0	1	1	1	1200
1	0	0	0	2400
1	0	0	1	2000
1	0	1	0	2400
1	0	1	1	3600
1	1	0	0	4800
1	1	0	1	7200
1	1	1	0	9600
1	1	1	1	19200

Display Speaker

SIO line 0 is wired to the Display Speaker.

- asserting RTS will emit a constant sound
- toggling DTR with RTS set emits different tones

SIO Write Control/Status Registers

WRITE REGISTER 0

D7	D6	D5	D4	D3	D2	D1	D0

0	0	0	Reg 0 Select
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 "

0	0	0	NULL Code
0	0	1	Send Abort
0	1	0	Reset External Status Int
0	1	1	Channel Reset
1	0	0	Enable Interrupt on next Rx Char
1	0	1	Reset Tx Interrupt Pending
1	1	0	Error Reset
1	1	1	Return from Interrupt (CH-A only)

0	0	NULL Code
0	1	Reset Rx CRC Checker
1	0	Reset Tx CRC Generator
1	1	Reset Tx Underrun/EOM Latch

WRITE REGISTER 1

D7	D6	D5	D4	D3	D2	D1	D0

D7 - Wait/Ready Enable
D6 - Wait/Ready Function
D5 - Wait/Ready on R/T
D4 - D3

0	0	Rx Interrupt Disable
0	1	Rx Interrupt on 1st Character
1	0	Int. on all Rx Chars (Parity affects vector)
1	1	" " " " " " does not " ")

D2 - Status affects vector (CH-B only)
D1 - Tx Interrupt only
D0 - EXT Interrupt Enable

WRITE REGISTER 2 (CHANNEL B ONLY)

D7	D6	D5	D4	D3	D2	D1	D0

D7 - D0 = Interrupt Vectors V7 - V0

WRITE REGISTER 3

D7	D6	D5	D4	D3	D2	D1	D0

D7 - D6
 0 0 Rx 5 Bits/Character
 0 1 Rx 7 Bits/Character
 1 0 Rx 6 Bits/Character
 1 1 Rx 8 Bits/Character
D5 - Auto Enables
D4 - Enter Hunt Phase
D3 - Rx CRC Enable
D2 - Address Search Mode(SDLC)
D1 - SYNC Character Load Inhibit
D0 - Rx Enable

WRITE REGISTER 4

D7	D6	D5	D4	D3	D2	D1	D0

D7 - D6
 0 0 X1 Clock Mode
 0 1 X16 Clock Mode
 1 0 X32 Clock Mode
 1 1 X64 Clock Mode
D5 - D4
 0 0 8 Bit Programmed SYNC
 0 1 16 Bit Programmed SYNC
 1 0 SDLC mode(01111110 flag pattern)
 1 1 External SYNC Mode
D3 - D2
 0 0 SYNC Modes Enable
 0 1 1 Stop Bit/Character
 1 0 1.5 Stop Bits/Character
 1 1 2 Stop Bits/Character
D1 - 0 = Odd Parity, 1 = Even Parity
D0 - Enable Parity

WRITE REGISTER 5

D7	D6	D5	D4	D3	D2	D1	D0

D7 - DTR
D6 - D5
 0 0 Tx 5 Bits(or less) Character
 0 1 Tx 7 Bits/Character
 1 0 Tx 6 Bits/Character
 1 1 Tx 8 Bits/Character
D4 - Send BREAK
D3 - Tx Enable
D2 - SDLC/CRC-16
D1 - RTS
D0 - Tx CRC Enable

WRITE REGISTER 6

D7	D6	D5	D4	D3	D2	D1	D0

D7-D0 = SYNC Bits 7-0

WRITE REGISTER 7

D7	D6	D5	D4	D3	D2	D1	D0

D7-D0 = SYNC Bits 15-8

SIO Read Control/Status Registers

READ REGISTER 0

D7	D6	D5	D4	D3	D2	D1	D0

D7 - BREAK/ABORT *
D6 - Tx Underrun/EOM *
D5 - CTS *
D4 - SYNC/Hunt *
D3 - DCD *
D2 - Tx Buffer Empty
D1 - Interrupt Pending(CH-A only)
D0 - Rx Character Available
* Used with External/Status Interrupt Mode

READ REGISTER 1

D7	D6	D5	D4	D3	D2	D1	D0

D7 - End of Frame(SDLC)
D6 - CRC/Framing Error
D5 - Rx Overrun Error
D4 - Parity Error

D3	D2	D1	I Field Prev Byte	I Field 2nd Prev Byte *
1	0	0	0	3
0	1	0	0	4
1	1	0	0	5
0	0	1	0	6
1	0	1	0	7
0	1	1	0	8
1	1	1	1	8
0	0	0	2	8

D0 - All Sent
* Residue data for 8 Rx bits/character programmed.

READ REGISTER 2

D7	D6	D5	D4	D3	D2	D1	D0

D7-D0 = Interrupt Vectors V7-V0

CHAPTER 9

DN460,DN660,DSP160

ADDRESS SPACE

physical		virtual
200800	traps	0
400	prom	0400-> 7FFF
200800	phys mem	0000400->FF7FFFF
200000	debug data	F800000->F8007FF
40000	disp2_mem	FFA0000->FFBFFFF
20000	displ_mem	FFC0000->FFDFFFF
10000	multibus	FFE0000->FFEFFFF
E000	color	FFF6000
E800	color	FFF6800
B000	sequencer	(not mapped)
C000	wcs	(not mapped)
B400	decoder i/o	(not mapped)
D000	i_cache0 i/o	(not mapped)
D400	i_cachel i/o	(not mapped)
D800	i_cache2 i/o	(not mapped)
DC00	i_cache3 i/o	(not mapped)
sftw	chksum buffer	FFF8800
sftw	zeroing buffer	FFF8C00
FC00	memory ctl	FFF9000
F400	disp 2	FFF9400
F000	disp 1	FFF9800
BC00	ring 2	FFF9C00
B800	ring 1	FFFA000
8C00	floppy	FFFA800
8800	timers	FFFAC00
8400	sios	FFFB000
8000	control panel	FFFB400
9000	io map	FFFF800

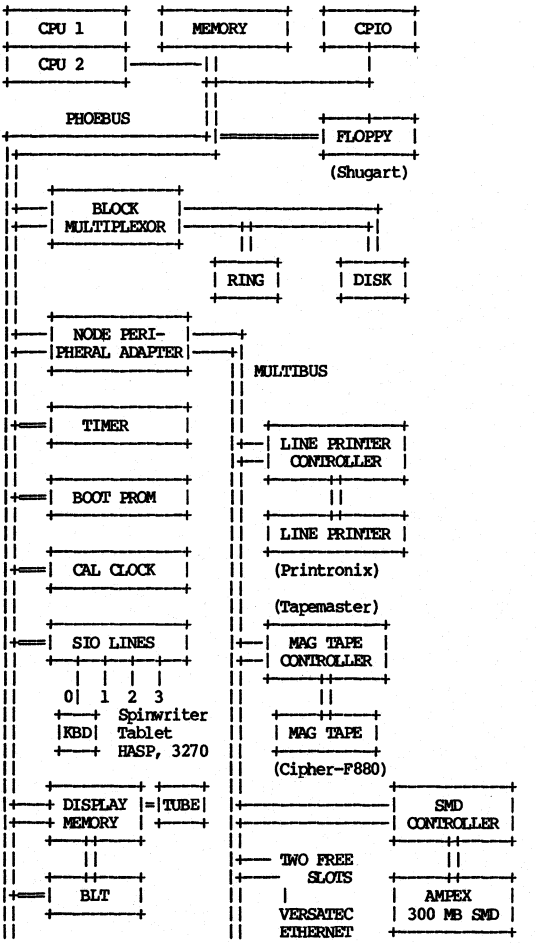
BLT REGISTERS

Refer to the BLT REGISTERS section of CHAPTER 8, DN400,DN420,DN600, for this information.

CACHE

MD commands (DC, IC) are used to turn data and instruction caches on and off. Machine must be in physical mode to toggle the instruction cache. Data cache is not controllable from CPIO. Sysboot (when running in CPU) turns both caches on.

CONFIGURATION



CPU CONTROL REGISTERS

The DNx60 is a micro-coded machine with a forward-mapped address translation mechanism (using in-memory page tables). There is no PIT or PFT. Some of the MMU Control Register functions described above are in the Control Panel Registers, while much of the control of the CPU is achieved through the MOVEC instruction. The following control register definitions are of interest:

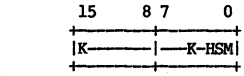
```
* 120-13F  - Current Region Registers (0-31)
*     11F  - Current ASID
      11E  - Current Floating Point ASID
      11D  - Purge Translation Buffer.  Values defined:
             1 => purge VA in Control Register 11C
             2 => purge entire TB
             4 => purge half of TB containing VA in 11C
      11C  - VA used in TB purge
*     119  - Address Translation and Data Cache
             Bit 0 : Enable/Disable Address Translation
             Bit 1 : Enable/Disable Data Cache
             Bit 7 : Domain Bit
      113  - Read Hardware and Microcode Revision Levels
             (Requires Micro Exec)
```

* These registers/functions are directly supported by the PROM.

CONTROL PANEL (CPIO) REGISTERS

CPIO Control Register

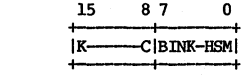
Address: [8000 | FFFB400]



K - Enables power-down switch (in two places)
H - Halts CPIO
S - CPIO becomes bus slave
M - CPIO becomes NOT bus master

CPIO Status Register

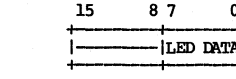
Address: [8002 | FFFB402]



K - Power-down switch enabled (in two places)
C - 0 => This is CPIO running
1 => This is CPU running
B - Bus timeout
I - Interrupt pending
N - Normal mode, 0 => Service Mode
H - CPIO is halted
S - CPIO is slave
M - CPIO is NOT master

LED Register

Address: [8004 | FFFB404]



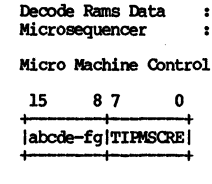
Micro-Machine Control Registers

None of these registers have mapped addresses.

These are used for loading microcode:

WCS Control Register	: C003	WCS Data/Address	: C005
Decode Rams Control	: B400	Decode Rams Address	: B402
Decode Rams Data	: B404	Decode Rams Shift	: B406
Microsequencer	: B000	CPIO/Microexec Port	: B003

Micro Machine Control and Status [B004]:



Left byte is read-only status:

a - Micro machine is frozen
b - Micro machine is in trap routine
c - Micro machine traps are enabled
d - Micro machine is fetching
e - 0 => micro machine is running
    1 => hit freeze on last instruction
f - CPIO data available in port for Micro Exec
g - Micro Exec data available in port for CPIO

Right byte may be written for control or read for status:

T - Cause Trap to Micro Exec
I - Enable Instruction Cache
P - Enable Pipeline Register
M - Macro step the micro machine
S - Micro step the micro machine
C - Clock step the micro machine
R - Start clocks (runs at address in pipeline)
E - Enable micro machine (0 => reset)

DISPLAY BOARD JUMPERS

Refer to the DISPLAY BOARD JUMPERS section of CHAPTER 8, DN400,DN420,DN600, for this information.

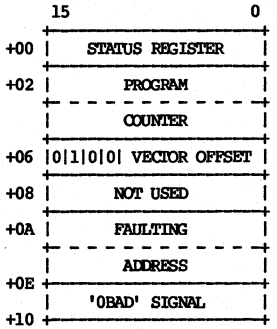
DISPLAY CONTROL AND STATUS REGISTER (DCSR)

Refer to the DISPLAY CONTROL AND STATUS REGISTER (DCSR) section of CHAPTER 8, DN400,DN420,DN600, for this information.

FAULT FRAME

LONG FAULT FRAME FORMAT

CPIO fault frames are the same as DN300 (68010). CPU short fault frames are also the same as DN300. Long fault frame used for bus error, address error, access violation, and region, segment and page faults is:



FAULT TYPES

Group	Exception	Processing
0	Reset Bus Error Address Error	Current instruction is aborted.
1	Trace Interrupt Illegal Ins. Privilege Ins.	Exception occurs before next instruction.
2	TRAP, TRAPV CHK Zero Divide	Processed by normal instruction execution.

Group 0 exceptions have the highest priority.

FAULT VECTORS

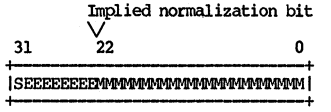
Exception vectors at [200800 | 0]

Vector	Address	Assignment
00	000	Reset: Initial SSP
	004	Reset: Initial PC
02	008	Bus Error
03	00C	Address Error
04	010	Illegal Instruction
05	014	Zero Divide
06	018	CHK Instruction
07	01C	TRAPV Instruction
08	020	Privilege Violation
09	024	Trace
0A	028	Unimplemented instruction
0B	02C	Unimplemented instruction
0C-0D	030	(Unassigned, reserved)
0E	038	Invalid Stack Format
0F-17	03C	(Unassigned, reserved)
18	060	Spurious Interrupt
19-1F	064	(Unassigned, reserved)
20-2F	080	TRAP Instruction Vectors
30	0C0	(Unassigned, reserved)
31	0C4	Floating Point Inexact Result
32	0C8	Floating Point Divide by Zero
33	0CC	Floating Point Underflow
34	0D0	Floating Point Operand Error
35	0D4	Floating Point Overflow
36-3F	0D8	(Unassigned, reserved)
40-7F	100	User Interrupt Vectors - unused
80	200	Region Fault
81	204	Segment Fault
82	208	Page Fault
83	20C	Access Violation
84	210	Floating Point ASID Trap
85-8F	214	(Unassigned, reserved)
90-9F	240	Ring/disk board
A0-AF	280	User Interrupt Vectors - unused
B0-B2	2C0	INT0/ INT2 -
B3	2CC	INT3/ - Tape Controller
B4	2D0	INT4/ - Storage Module
B5	2D4	INT5/ -
B6	2D8	INT6/ - Line Printer
B7	2DC	INT7/ - Parallel Input
B8-C3	2E0	User Interrupt Vectors -- unused
C4-CD	310	Unused
CE-CF	338	Color
D0-EF	340	Unused
F0	3C0	P - ECCC (Automatic vectors)
F1	3C4	O -
F2	3C8	N - Display #2
F3	3CC	M - Floppy

F4	3D0	L - Display #1 (BLT)
F5-F7	3D4	K, J, I -
F8	3E0	Unused
F9-FA	3E4	Sio Lines (2 lines/Vector)
FB	3EC	Timers 1,2,3
FC-FE	3F0	Unused
FF	3FC	ECCU

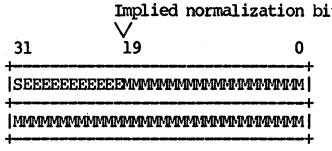
FLOATING-POINT FORMAT

Single-Precision Floating-Point Format



S - Sign; S = 1 => Negative
E..E - Exponent plus 127
M..M - Mantissa

Double-Precision Floating-Point Format



S - Sign; S = 1 => Negative
E..E - Exponent plus 1023
M..M - Mantissa

FLOPPY CONTROLLER

Refer to the FLOPPY CONTROLLER section of CHAPTER 8, DN400,DN420,DN600, for this information.

MEMORY CONTROL/STATUS REGISTERS (MCSR)

There are 64 possible memory control register addresses, beginning at [FC02 | FFF9002]. Within each range of $100 addresses, only one board may respond, but owing to the variety of boards supported ('old', 'new', interleaved, non-interleaved), a board may respond at any one of 8 addresses within that range (at intervals of $20). Also boards do not have to respond at contiguous ranges. There can be at most 4 memory boards.

Refer to Chapter 8, DN400,Dn420,DN600, for control and status format.

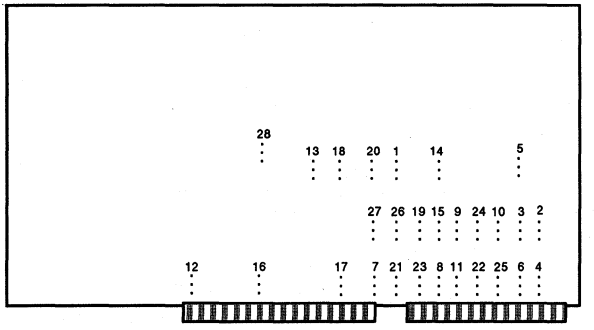
MEMORY BOARD JUMPERS

	NON-INTERLEAVED ADDRESS SPACE						INTERLEAVED ADDRESS SPACE							
	1-Mb				2-Mb		1-Mb				2-Mb			
	1	2	3	4	2	4	2	2	4	4	0	0	2	2
	0	0	0	0	0	0	0	0	0	0	8	8	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	2	0	2	0	2	0	2
	|	|	|	|	|	|	|	|	|	|	|	|	|	|
	1	2	3	4	3	5	3	3	5	5	3	3	5	5
	f	f	f	f	f	f	f	f	f	f	f	f	f	f
	f	f	f	f	f	f	f	f	f	f	f	f	f	f
	f	f	f	f	f	f	f	f	f	f	f	f	f	f
	f	f	f	f	f	f	f	f	f	f	f	f	f	f
	f	f	f	f	f	f	d	f	d	f	d	f	d	f
wp 1	u	u	u	u	u	u	u	u	u	u	u	u	u	u
wp 2	u	u	u	u	u	u	u	u	u	u	u	u	u	u
wp 3	u	u	u	d	u	u	u	u	d	d	u	u	u	u
wp 4	u	d	d	u	u	d	d	d	u	u	u	u	d	d
wp 5	d	u	d	u	d	u	u	d	u	d	u	d	u	d
wp 6	u	u	u	u	u	u	u	u	u	u	u	u	u	u
wp 7	u	u	u	u	u	u	d	d	d	d	d	d	d	d
wp 8	u	u	d	d	u	d	u	d	d	u	u	u	u	d
wp 9	u	d	u	d	d	d	d	u	d	u	d	u	d	u
wp 10	u	u	u	u	d	d	u	u	u	u	d	d	d	d
wp 11	u	u	u	u	u	u	u	u	u	u	u	u	d	d
wp 12	d	d	d	d	d	d	d	d	d	d	d	d	d	d
wp 13	d	d	d	d	u	u	d	d	d	d	u	u	u	u
wp 14	d	d	d	d	d	d	u	d	u	d	u	d	u	d
wp 15	d	d	d	d	d	d	d	u	d	u	d	u	d	u
wp 16	u	u	u	u	u	u	d	d	d	d	d	d	d	d
wp 17	u	u	u	u	u	u	d	d	d	d	d	d	d	d
wp 18	u	u	u	u	u	u	u	u	u	u	u	u	u	u
wp 19	u	u	u	u	d	d	u	u	u	u	d	d	d	d
wp 20	u	u	u	u	d	d	u	u	u	u	d	d	d	d
wp 21	u	u	u	u	d	d	u	u	u	u	d	d	d	d
wp 22	u	u	u	u	d	d	u	u	u	u	d	d	u	u
wp 23					Rev.01							Rev.01		
	u	u	u	u	o	o	u	u	u	u	o	o	o	o
					Rev.02							Rev.02		
					d	d					d	d	d	d
wp 24	u	u	u	u	d	d	u	u	u	u	d	d	d	d
wp 25	u	u	u	u	u	u	u	u	u	u	u	u	u	u
wp 26	u	u	u	u	u	u	u	u	u	u	u	u	u	u
wp 27	d	d	d	d	d	d	d	d	d	d	d	d	d	d
wp 28	u	u	u	u	u	u	d	d	d	d	d	d	d	d

u=UP d=DOWN o=OUT

MEMORY BOARD

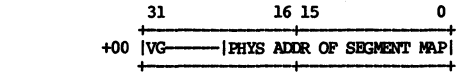
ADM60



REGION REGISTER ARRAYS

RARS: ARRAY[0..26,0..31] OF RAR_T

REGION REGISTER (RAR_T):
(type "rar_t" in vm.ins.pas)



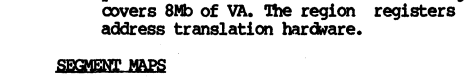
V - Region is valid (.valid)
G - Region is global (.global)

RARS is a dynamically-allocated per-asid table whose entries contain the region register values for that process. There are 32 hardware region registers, each covers 8Mb of VA. The region registers are used by the address translation hardware.

SEGMENT MAPS

SMAPS: ARRAY[0..26,0..7680] OF SMAPE_T

SEGMENT MAP ENTRY (SMAPE):
(type "smape_t" in vm.ins.pas)



V - Entry is valid (.valid)
A - Access rights as given in 16Mb MSTE

The SMAP is a dynamically-allocated per-ASID table whose entries match one-for-one with a process's mstes. It is used by the address translation hardware and is organized in 256 segment units, so that one page of smapes represents one REGION (8Mb of VA) of one address space.

See also the PAGE MAPS and MST sections of Chapter 1, AEGIS.

MONITOR TIMING (DN660)

Refer to the MONITOR TIMING section of CHAPTER 8, DN400,DN420,DN600, for this information.

RING/DISK

Refer to the RING/DISK section of CHAPTER 8, DN400,DN420,DN600, for this information.

SERIAL I/O INTERFACE

SIO page at [8400 | FFFB000]

DNx60 uses two SC2681 chips providing four SIO lines, with control at FFFB000 and FFFB010. The first line (make B at FFFB000) is used for the display keyboard.

Due to limitations of the SC2681 chip, when both SIO lines of a chip are being used, it is possible to have incompatible baud rates. One SIO line can't have a baud rate from Group A while the other SIO line is set from Group B:

Group A	Group B
50	75
7200	150
	2000
	19.2K

Lines 0 (the keyboard, which runs at 1200 baud) and 1 are paired on one chip, and lines 2 and 3 are on the other chip.

Refer to the SERIAL I/O INTERFACE section of CHAPTER 7, DN300,DN320, for more information.

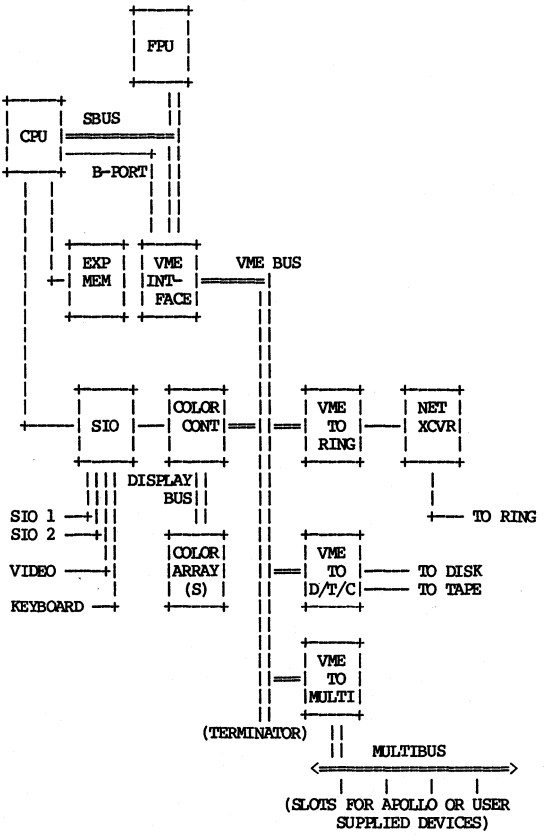
CHAPTER 10

DN550

ADDRESS SPACE

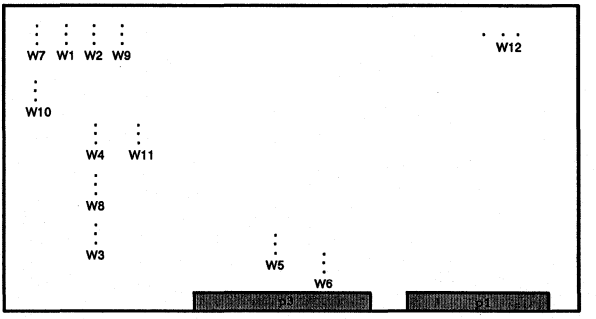
physical		virtual
400	prom	400-3FFF (one-to-one)
4000	pft	FFB800-FFF7FF
8000	mmu	FFB400
8400	sios	FFB000
8800	timers	FFAC00
9800	ring	FF9C00
9C00	disk, tape, cal	FFA800
A400	pbu ctl	FF7C00
B000	fpu ctl	FF7000
B400	fpu cmd	FF7400
B800	fpu cs	FF7800
BC00	VME control	FF9400
E000	color_sup	FF6000
E400	color_user	FF6400
E800	color_wcs	FF6800
F000	displ_sup	FF9800
F400	displ_user	FFA000
F800	displ_wcs	FFA400
10000	iomap	FF5000-FF5FFF
14000	prom2	-
20000	displ_mem	FC0000-FDFFFF
40000	color_mem	FA0000-FBFFFF
70000	pbu i/o ref	FE0000-FE7FFF
80000	pbu lst half	-
100000	mem: md data	E00000
100400	mem: traps	0
100800	mem	100800
380000	pbu 2nd half	-
700000	ptt	700000-7FFFFF

CONFIGURATION



CPU BOARD JUMPERS

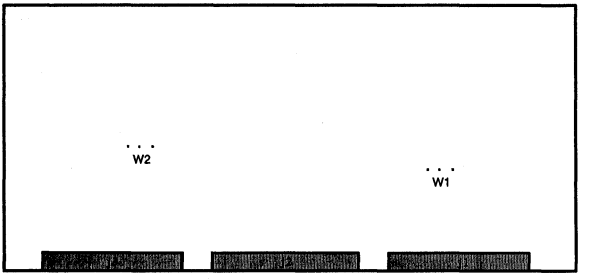
DN550 CPU Board (APNs 4141, 4145)



W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12
Up	Up	Up	Up	Up	Up	Down	Down	Down	Down	Up	Left

DISPLAY BOARD JUMPERS

ARRAY BOARD

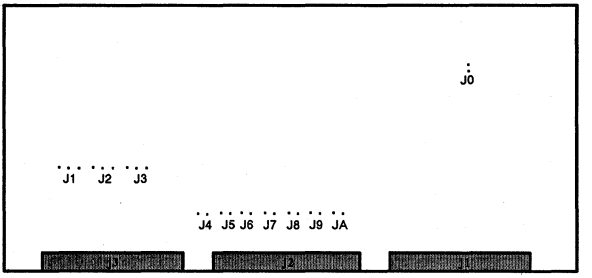


W1 (Timing Delay)
Right = Normal, Left = 3 nSec delay

W2 (Board Select)
Right = Board 1, Left = Board 2

CONTROL BOARD

DN550 Color Controller (APN 3954)



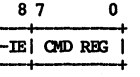
DISPLAY REGISTERS [F000 | 0FF9800]

```
displ_sup   [ F000 | 0FF9800 ]   (also second color display)
displ_user  [ F400 | 0FFA000 ]
displ_wcs   [ F800 | 0FFA400 ]

color_sup   [ E000 | 0FF6000 ]   (also second
color_user  [ E400 | 0FF6400 ]      black and white display)
color_wcs   [ E800 | 0FF6800 ]
```

Display controller will be a 600 with microcode changes.

The VME interface is presently undefined.

DISK/TAPE/CALENDAR CONTROLLER [9C00 | 0FFA800]



00-0F - Disk commands:

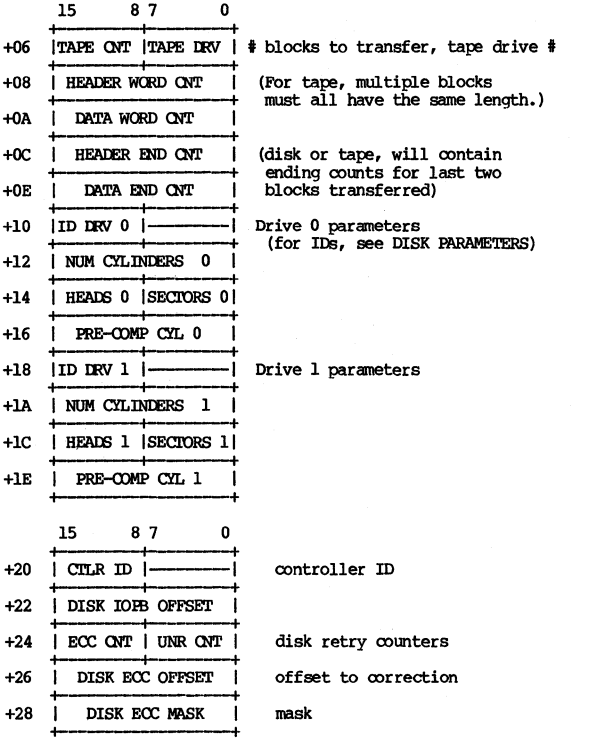
```
  00  Nop (ignored by controller)
  01  Seek
  02  Read/write sector(s)
  03  Format track
  04  Read sector ID(s) (into I/O PBs)
  05  Restore (to cylinder zero)
  06  Force ECC error(s)
  07-0F  Reserved (illegal disk commands)
```

10-1F - Tape commands:

```
  10  Select tape drive 0
  11  Select tape drive 1
  12  Select tape drive 2
  13  Select tape drive 3
  14  Read status
  15  Rewind to BOT
  16  Erase entire tape
  17  Initialize (retension) tape
  18  Write block(s)
  19  Write <count> filemarks
  1A  Read block(s)
  1B  Space forward <count> blocks
  1C  Space reverse <count> blocks (not implemented)
  1D  Space forward <count> filemarks
  1E  Space reverse <count> filemarks (not implemented)
  1F  Reset drive
```

20-2F - Controller commands:

```
  20  Reset controller
  21  Execute controller diagnostics
  22  Calibrate memory timing loop
  23  Calibrate BX timing loop
  24  Enable force underrun/overrun (on next command)
  25  Disable  "        "
 26-2F Reserved (illegal commands)

30-FF - Reserved (illegal commands)

I = 1 => controller interrupting (reset to get next interrupt)
E = 1 => enable controller interrupt

      15                    0
     +------------------------+
+02  |   CONTROLLER STATUS    |
     +------------------------+
     15  8000 - controller busy
     14  4000 - disk busy
     13  2000 - tape busy
     12  1000 -
     11  0800 - disk op complete
     10  0400 - disk status valid
      9  0200 - tape op complete
      8  0100 - tape status valid
      7  0080 - DMA not at end of range
      6  0040 - DMA overrun/underrun
      5  0020 - memory parity during DMA
      4  0010 -
      3  0008 - illegal controller command
      2  0004 - controller timeout
      1  0002 - controller diagnostic failed
      0  0001 -
      15      8 7       0
     +----------+---------+
+04   | DSK CNT |DSK STAT |  # sectors to transfer, disk status
     +----------+---------+
     01 Seek did not complete
     02 Write fault
     03 Unit not present
     04 Sector not found
     05 No index pulse
     06 Drive not ready
     07 No track zero
     08 Addr mark not found
     09 ECC error in ID field
     0A Correctable ECC error in data field
     0B Uncorrectable ECC error in data field
     0C Recovered ECC error (see dsk_ecc_cnt)
     0D Recovered overrun/underrun (see dsk_unr_cnt)
```


15 8 7 0
+06 TAPE CNT TAPE DRV # blocks to transfer, tape drive #
+08 HEADER WORD CNT (For tape, multiple blocks must all have the same length.)
+0A DATA WORD CNT
+0C HEADER END CNT (disk or tape, will contain ending counts for last two blocks transferred)
+0E DATA END CNT
+10 ID DRV 0 Drive 0 parameters (for IDs, see DISK PARAMETERS)
+12 NUM CYLINDERS 0
+14 HEADS 0 SECTORS 0
+16 PRE-COMP CYL 0
+18 ID DRV 1 Drive 1 parameters
+1A NUM CYLINDERS 1
+1C HEADS 1 SECTORS 1
+1E PRE-COMP CYL 1
15 8 7 0
+20 CTLR ID controller ID
+22 DISK IOPB OFFSET
+24 ECC CNT UNR CNT disk retry counters
+26 DISK ECC OFFSET offset to correction
+28 DISK ECC MASK mask

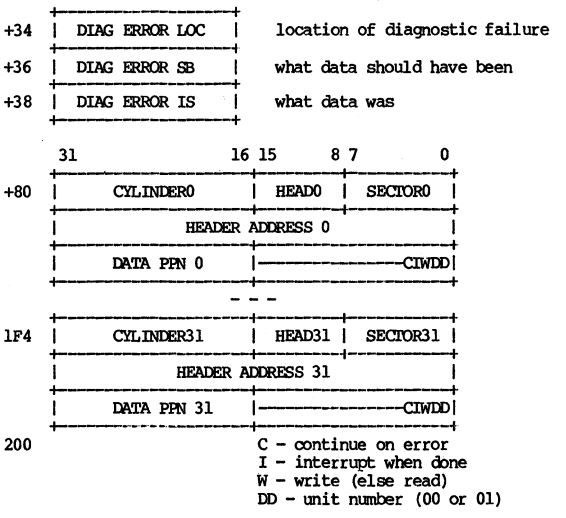
```
      15      8 7      0
     +---------+--------+
+2A  |TAPE ST 0|TAPE ST 1|   status from drive
     +---------+--------+
```

7	80	Status byte 0 bits
6	40	Cartridge not in place
5	20	Unselected drive
4	10	Write-protected cartridge
3	08	End of media (EOM)
2	04	Unrecoverable data error
1	02	Bad block not located
0	01	Filemark detected

7	80	Status byte one bit
6	40	Illegal command
5	20	No data detected
4	10	Marginal block detected
3	08	Beginning of media (BOM)
2	04	Reserved for bus parity error
1	02	Reserved for end of recorded media
0	01	Power on/reset occurred

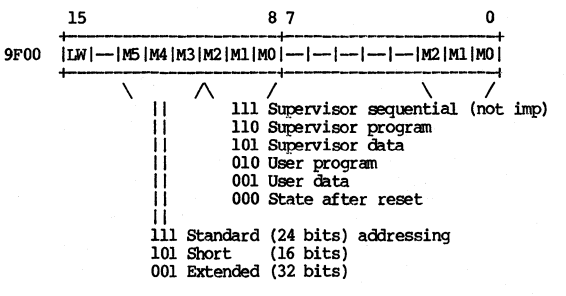
TAPE ST 0	TAPE ST 1	STATUS SUMMARY
11110001	00000000	Drive not ready (controller generated)
110X0000	00000000	No cartridge
11110000	00000000	No drive
10010000	X000X000	Write protected
10001000	00000000	End of media
100X0100	10001000	Read or write abort
100X0100	00000000	Read error, bad block transfer
100X0110	00000000	Read error, filler block transfer
100X0110	10100000	Read error, no data
100X1110	10100000	Read error, no data and EOM
100X0110	101X1XX0	Read error, no data and BOM
100X0001	00000000	Filemark read
XXXX0000	1100X000	Illegal command
XXXX0000	1000X001	Power on/reset
100X0001	00010000	Marginal block detected

```
      15                0
     +------------------+
+2C  |  TAPE ERROR CNTR |    tape error counter
     +------------------+
+2E  | TAPE UNDERRUN CNTR|   tape underrun counter
     +------------------+
+30  |CTLR TIMEOUT STATUS|   timeout status
     +---------+--------+
+32  |LAST CMD |       P|    most recent command executed
     +---------+--------+    P => precomp enabled
```



CALENDAR CONTROLS [9F00 | 0FFAA00]

DISKTAPE VME ADDRESS MODIFIERS [9F00 | 0FFAB00]



FAULT FRAME

Refer to the FAULT FRAME section of CHAPTER 7, DN300,DN320, for this information.

FAULT TYPES

Refer to the FAULT TYPES section of CHAPTER 7, DN300,DN320, for this information.

FAULT VECTORS

Exception vector at [100400 | 0]

Vector	Address	Assignment	
00	000	Reset: Initial SSP	
	004	Reset: Initial PC	
02	008	Bus Error	
03	00C	Address Error	
04	010	Illegal Instruction	
05	014	Zero Divide	
06	018	CHK Instruction	
07	01C	TRAPV Instruction	
08	020	Privilege Violation	
09	024	Trace	
0A	028	Unimplemented instruction	
0B	02C	Unimplemented instruction	
0C-0D	030	(Unassigned, reserved)	
0E	038	Invalid Stack Format	
0F-17	03C	(Unassigned, reserved)	
18	060	Spurious Interrupt	
19-1F	064	Level 1-7 Auto Vector	interrupt level
19	064	SIO (rcv and xmit)	1
1A	068	Display keyboard	2
1B	06C	PEB	3
1C	070	VME	4
1D	074	VME bus error	5
1E	078	Timers 1,2,3	6
1F	07C	Parity error	7
20-2F	080	TRAP Instruction Vectors	
30-3F	0C0	(Unassigned, reserved)	
40	100	Machine ID, auxiliary information	
41	104	getc	
42	108	putc	
43	10C	init_dsk	
44	110	read_dsk	
45	114	reload_font	
46	118	pollc	
47	11C	quiet_ret	

48	120	write_dsk		
49	124	log_error		
4A	128	crash		
4B	12C	led_update		
4C-5F	130	-		
60	180	VME pseudo-vectors	0	(unused)
61	184	ring	1	
62	188	disktape	2	
63	18C	display	3	
64	190	-	4	
65	194	pbu	5	
66	198	-	6	
67	19C	-	7	
68-6F	1A0	-		
70-FF	1C0	-		

FLOATING-POINT FORMAT

Refer to the FLOATING-POINT FORMAT section of CHAPTER 7, DN300,DN320, for this information.

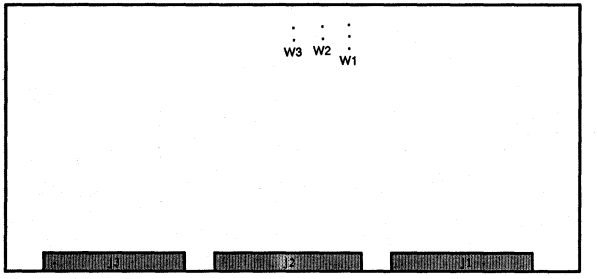
FPU

Refer to the FPU section of CHAPTER 7, DN300,DN320, for this information.

MEMORY CONTROL/STATUS REGISTERS (MCSR)

Refer to the MEMORY CONTROL/STATUS REGISTERS (MCSR) section of CHAPTER 7, DN300,DN320, for this information.

MEMORY BOARD JUMPERS



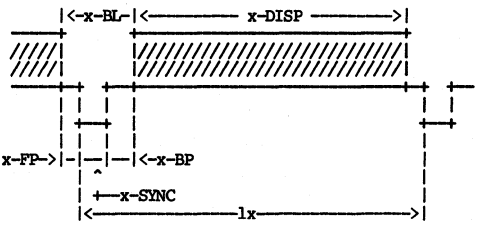
W1	W2	W3
Up = CPU/Mem=2MB, Down = CPU/Mem=0.5MB	Out Out In In	Out = No memory expansion In = Fully populated (APN 3354) Out = 1/2 populated (APN 4151) In = Not allowed

MEMORY MANAGEMENT UNIT (MMU)

Refer to the MEMORY MANAGEMENT UNIT section of CHAPTER 7, DN300,DN320, for this information.

MONITOR TIMING

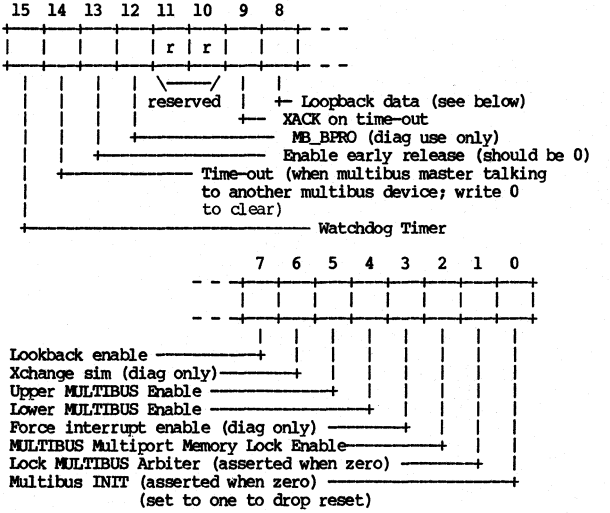
Axis	Item		Duration	Frequency
HORIZONTAL	1H	HORIZONTAL FREQUENCY	29.53 uSec	33.855 Hz
	H-FP	HORIZONTAL FRONT PORCH	0.75 uSec	N/A
	H-SYNC	HORIZONTAL SYNC	2.75 uSec	N/A
	H-BP	HORIZONTAL BACK PORCH	3.5 uSec	N/A
	H-BL	HORIZONTAL BLANKING	7.0 uSec	N/A
	H-DISP	HORIZONTAL DISPLAY AREA	22.53 uSec	N/A
VERTICAL	1V	VERTICAL FREQUENCY	12.75 mSec	78.400 Hz
	V-FP	VERTICAL FRONT PORCH	88.6 uSec	N/A
	V-SYNC	VERTICAL SYNC	88.6 uSec	N/A
	V-BP	VERTICAL BACK PORCH	708.0 uSec	N/A
	V-BL	VERTICAL BLANKING	885.0 uSec	N/A
	V-DISP	VERTICAL DISPLAY AREA	11.865 mSec	N/A



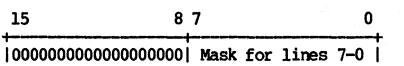
x = V or H

MULTIBUS REGISTERS

MULTIBUS CONTROL STATUS REGISTER [A400 | FF7C00]

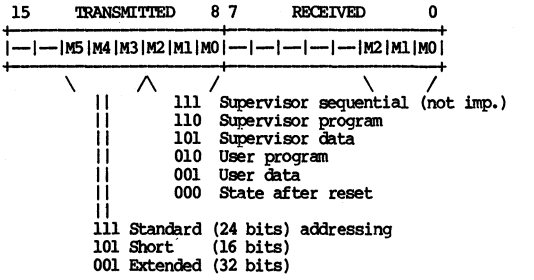


MULTIBUS INTERRUPT CONTROL STATUS REGISTER [A402|FF7C02]

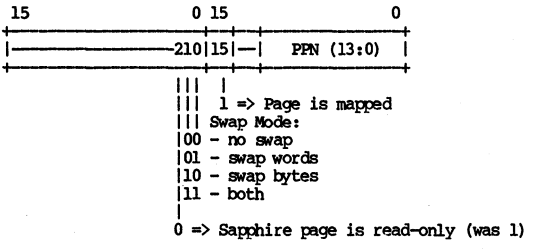


(Interrupt level now in VME interrupt/ID register 5 following interrupt acknowledge cycle.)

MULTIBUS ADDRESS MODIFIER REGISTER [A404 | FF7C04]



MULTIBUS MAP [10000 | FF5000]



LOOPBACK DATA

(No longer exists.)

PAGE FRAME TABLE ENTRY (PFTE)

Refer to the PAGE FRAME TABLE ENTRY (PFTE) section of CHAPTER 7, DN300,DN320, for this information.

PAGE TRANSLATION TABLE ENTRY (PTTE)

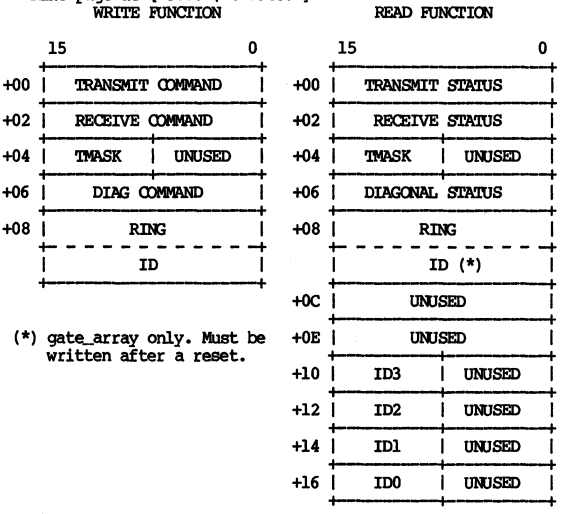
Refer to the PAGE TRANSLATION TABLE ENTRY (PTTE) section of CHAPTER 7, DN300,DN320, for this information.

PROM ENTRY POINTS

```
100:    dc.w  5,0            5 => stingray
104:    ac    getc           returns character in D1
108     ac    putc           prints character in D1
10C     ac    init_dsk       initialize disk
110     ac    read_dsk       read a record from disk
114     ac    reload_font    reload font
118     ac    pollc          returns character in D1,
                             else -1 in d1.w
11C     ac    quiet_ret      quiet return to PROM
120     ac    write_disk
124     ac    log_error      on/off
128     ac    crash
12C     ac    led_update
```

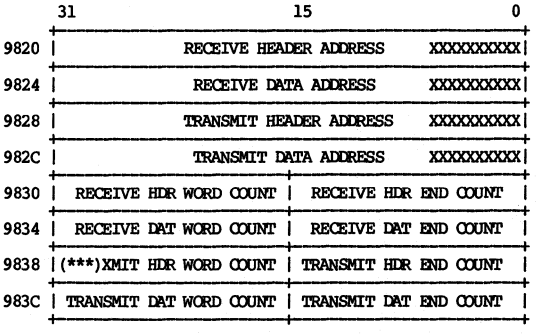
RING REGISTERS

RING page at [9800 | 0FF9C00]



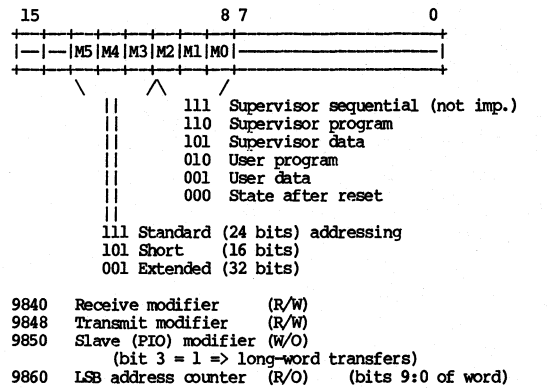
(*) gate_array only. Must be written after a reset.

RING MAP/WORD COUNT REGISTERS [9820 | 0FF9C20]



(*) low-order 10 bits are ignored
(**) bits 15:10 of counts are ignored; read as junk
(***) bit 11 of transmitter header count = 1 => data length = 0

RING ADDRESS MODIFIER REGISTERS [9840 | 0FF9C40]



9840 Receive modifier (R/W)
9848 Transmit modifier (R/W)
9850 Slave (PIO) modifier (W/O)
(bit 3 = 1 => long-word transfers)
9860 LSB address counter (R/O) (bits 9:0 of word)

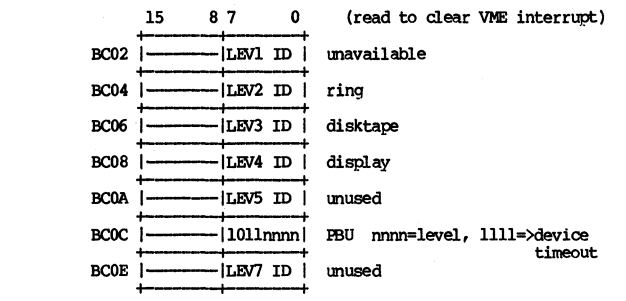
Refer to the RING REGISTERS section of CHAPTER 7, DN300,DN320 for command and status definitions.

SERIAL I/O INTERFACE

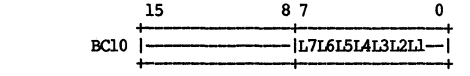
Refer to the SERIAL I/O INTERFACE section of CHAPTER 7, DN300,DN320, for this information.

VME REGISTERS [BC00 | FF9400]

VME INTERRUPT STATUS/ID REGISTERS [BC02 | FF9402]

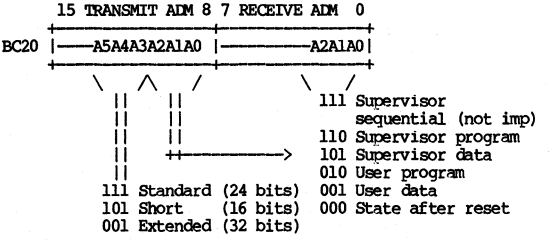


VME INTERRUPT LEVEL REGISTER [BC10 | FF9410]

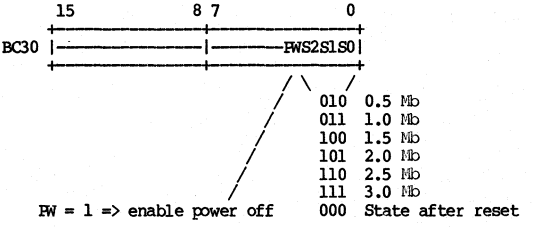


0-bit => that level is interrupting

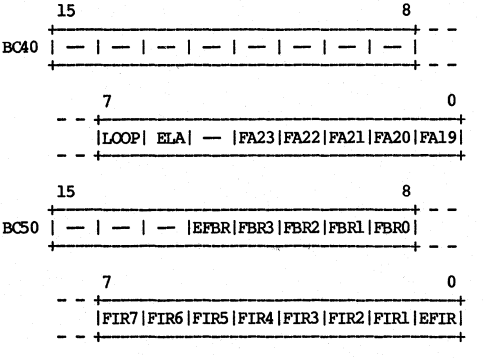
VME ADDRESS MODIFIER REGISTER [BC20 | FF9420]



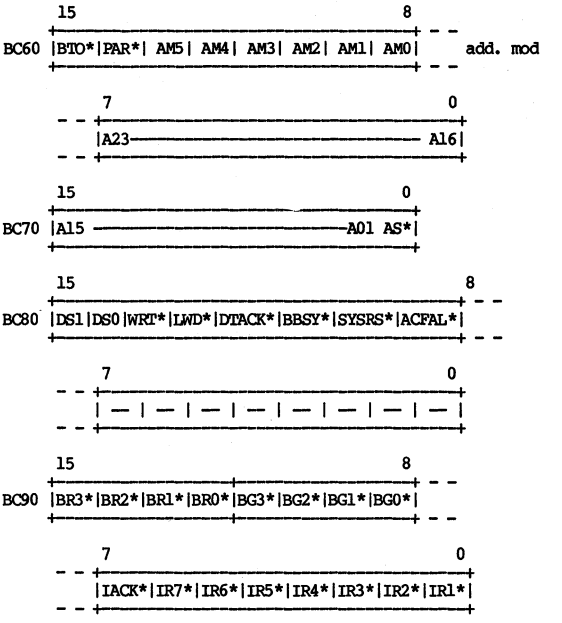
VME MEMORY SIZE REGISTER [BC30 | FF9430]



VME DIAGNOSTIC CONTROL REGISTERS [BC40 | FF9440]



VME ERROR REGISTERS [BC60 | FF9460]



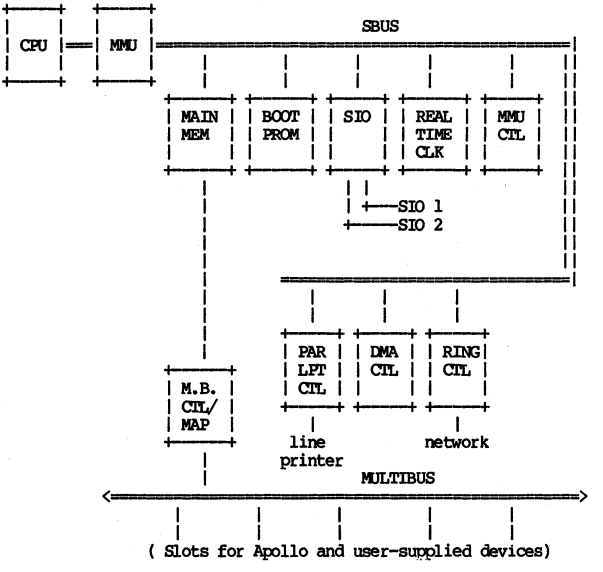
CHAPTER 11

DSP80,DSP80A

ADDRESS SPACE

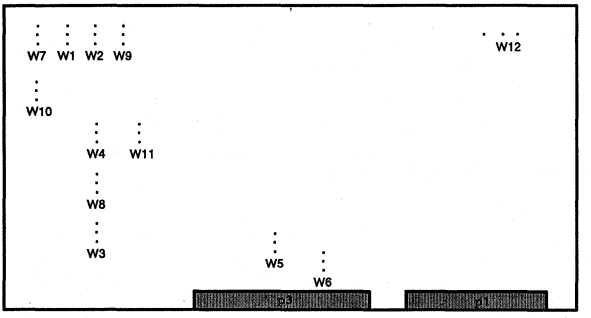
physical		virtual
100400	traps	0
400	prom	400->3FFF (one-to-one)
14000	prom2	-
100800	phys mem	100800->FFFFF
700000	ptt	700000->7FFFFF
380000	pbu 2nd half	-
100000	md stk,data	E00000
80000	pbu 1st half	-
70000	pbu i/o ref	FE0000->FE7FFF
10000	iomap	FF5000->FF5FFF
A400	pbu ctl	FF7C00
A800	lpr	FF8000
9800	ring	FF9C00
9000	dma ctl	FFA000
A000	calendar	FFA400
8800	timers	FFAC00
8400	sios	FFB000
8000	mmu	FFB400
4000	pft	FFB800->FFF7FF

CONFIGURATION



CPU BOARD JUMPERS

DSP80/A CPU Board (APNs 2422, 4834)



W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12
Up	Up	Up	Up	Up	Up	Down	Down	Down	Down	Up	Right

DMA CONTROLLER

DMAC page at [9000 | 0FFA000]

The DMA controller is a Motorola M68450.

9000-907F - ring receive header
9080-90FF - ring receive data
9100-917F - ring transmit
9180-91FF - Unused

Refer to the DMA CONTROLLER section of CHAPTER 7, DN300,DN320, for more information.

FAULT FRAME

Refer to the FAULT FRAME section of CHAPTER 7, DN300,DN320, for this information.

FAULT TYPES

Refer to the FAULT TYPES section of CHAPTER 7, DN300,DN320, for this information.

FAULT VECTORS

Exception vector at [100400 | 0]

Vector	Address	Assignment	int level
00	000	Reset: Initial SSP	
	004	Reset: Initial PC	
02	008	Bus Error	
03	00C	Address Error	
04	010	Illegal Instruction	
05	014	Zero Divide	
06	018	CHK Instruction	
07	01C	TRAPV Instruction	
08	020	Privilege Violation	
09	024	Trace	
0A	028	Unimplemented instruction	
0B	02C	Unimplemented instruction	
0C-0D	030	(Unassigned, reserved)	
0E	038	Invalid Stack Format	
0F-17	03C	(Unassigned, reserved)	
18	060	Spurious Interrupt	
19-1F	064	Level 1-7 Auto-Vector	int level
19	064	SIO (rcv and xmit)	1
1A	068	(Keyboard input)	2
1B	06C	Ring	3
1C	070	PBU	4
1D	074	Line printer	5
1E	078	Timers 1,2,3	6
1F	07C	Parity error	7
20-2F	080	TRAP Instruction Vectors	
30-3F	0C0	(Unassigned, reserved)	
40-8F	100	User Interrupt Vectors - unused	
90-9F	240	Ring/disk board	
A0-AF	280	User Interrupt Vectors - unused	
B0	2C0	INT0/ -	
B1	2C4	INT1/ -	
B2	2C8	INT2/ -	
B3	2CC	INT3/ - Tape Controller	
B4	2D0	INT4/ - Storage Module	
B5	2D4	INT5/ -	
B6	2D8	INT6/ - Line Printer	
B7	2DC	INT7/ - Parallel Input	
B8-C3	2E0	User Interrupt Vectors -- unused	
C4-CF	310	Unused	
D0-DF	340	SIO lines	
E0-EF	380	Unused	
F0	3C0	P - ECCC (Automatic vectors)	
F1	3C4	O -	
F2	3C8	N - Display 2	
F3	3CC	M - Floppy	
F4	3D0	L - Display 1 (BLT)	
F5	3D4	K -	
F6	3D8	J -	

F7	3DC	I -
F8-FA	3E0	Unused
FB	3EC	Timers 1,2,3
FC-FD	3F0	Unused
FE	3F8	CPU B-to-A
FF	3FC	ECCU

MEMORY CONTROL/STATUS REGISTERS (MCSR)

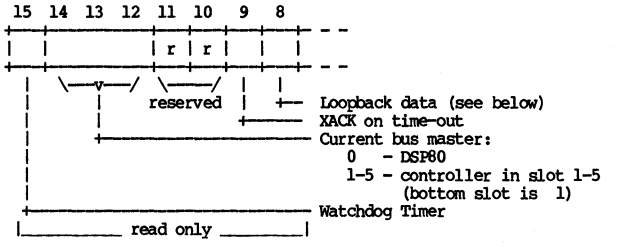
Refer to the MEMORY CONTROL/STATUS REGISTERS (MCSR) section of CHAPTER 7, DN300,DN320, for this information.

MEMORY MANAGEMENT UNIT (MMU)

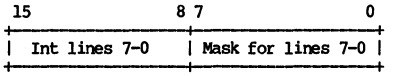
Refer to the MEMORY MANAGEMENT UNIT section of CHAPTER 7, DN300,DN320, for this information.

MULTIBUS REGISTERS

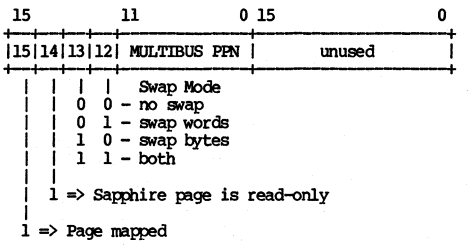
MULTIBUS CONTROL STATUS REGISTER [A400|FF7C00]



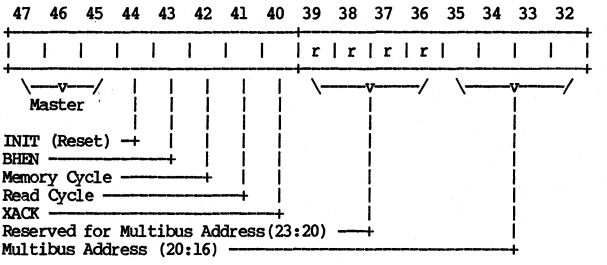
MULTIBUS INTERRUPT CONTROL STATUS REGISTER [A402|FF7C02]

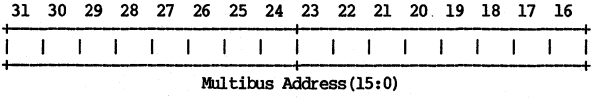


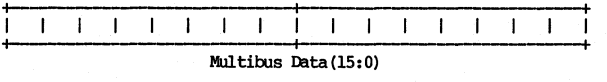
MULTIBUS MAP [10000 | FF5000]



LOOPBACK DATA







PAGE FRAME TABLE ENTRY (PFTE)

Refer to the PAGE FRAME TABLE ENTRY (PFTE) section of CHAPTER 7, DN300,DN320, for this information.

PAGE TRANSLATION TABLE ENTRY (PTTE)

Refer to the PAGE TRANSLATION TABLE ENTRY (PTTE) section of CHAPTER 7, DN300,DN320, for this information.

PROM ENTRY POINTS

```
100:    dc.w  3,0             3 => DSP80, no aux info
104:    ac    getc            returns character in D1
108     ac    putc            prints character in D1
10C     ac    init_dsk        initialize disk
110     ac    read_dsk        reads a record from disk
114     ac    reload_font     reloads font
118     ac    pollc           returns character in D1,
                              else -1 in d1.w
11C     ac    quiet_ret       quiet return to PROM
120     ac    write_disk
124     ac    log_error       on/off
128     ac    crash
12C     ac    led_update
```

RING REGISTERS

Refer to the RING REGISTERS section of CHAPTER 7, DN300,DN320, for this information.

SERIAL I/O INTERFACE

The DSP80,DSP80A uses only the Signetics SC2681 DUART (there is no keyboard interface).

Refer to the SERIAL I/O INTERFACE section of CHAPTER 7, DN300,DN320, for SIO information.

APPENDIX A - ASCII CHARACTER SET

oct	hex		dec	oct	hex		dec	oct	hex		dec
00	00	NUL	00	60	30	0	48	140	60		96
01	01	SOH	01	61	31	1	49	141	61	a	97
02	02	STX	02	62	32	2	50	142	62	b	98
03	03	ETX	03	63	33	3	51	143	63	c	99
04	04	EOT	04	64	34	4	52	144	64	d	100
05	05	ENQ	05	65	35	5	53	145	65	e	101
06	06	ACK	06	66	36	6	54	146	66	f	102
07	07	BEL	07	67	37	7	55	147	67	g	103
10	08	BS	08	70	38	8	56	150	68	h	104
11	09	HT	09	71	39	9	57	151	69	i	105
12	0A	NL(LF)	10	72	3A	:	58	152	6A	j	106
13	0B	VT	11	73	3B	;	59	153	6B	k	107
14	0C	FF	12	74	3C	<	60	154	6C	l	108
15	0D	CR	13	75	3D	=	61	155	6D	m	109
16	0E	RRS	14	76	3E	>	62	156	6E	n	110
17	0F	BRS	15	77	3F	?	63	157	6F	o	111
20	10	RCP	16	100	40		64	160	70	p	112
21	11	XON	17	101	41	A	65	161	71	q	113
22	12	HLF	18	102	42	B	66	162	72	r	114
23	13	XOFF	19	103	43	C	67	163	73	s	115
24	14	HLR	20	104	44	D	68	164	74	t	116
25	15	NAK	21	105	45	E	69	165	75	u	117
26	16	SYN	22	106	46	F	70	166	76	v	118
27	17	ETB	23	107	47	G	71	167	77	w	119
30	18	CAN	24	110	48	H	72	170	78	x	120
31	19	EM	25	111	49	I	73	171	79	y	121
32	1A	SUB	26	112	4A	J	74	172	7A	z	122
33	1B	ESC	27	113	4B	K	75	173	7B	{	123
34	1C	FS	28	114	4C	L	76	174	7C	|	124
35	1D	GS	29	115	4D	M	77	175	7D	}	125
36	1E	RS	30	116	4E	N	78	176	7E	~	126
37	1F	US	31	117	4F	O	79	177	7F	DEL	127
40	20	SP	32	120	50	P	80				
41	21	!	33	121	51	Q	81				
42	22	"	34	122	52	R	82				
43	23	#	45	123	53	S	83				
44	24	$	36	124	54	T	84				
45	25	%	37	125	55	U	85				
46	26	&	38	126	56	V	86				
47	27	'	39	127	57	W	87				
50	28	(	40	130	58	X	88				
51	29	)	41	131	59	Y	89				
52	2A	*	42	132	5A	Z	90				
53	2B	+	43	133	5B	[	91				
54	2C	,	44	134	5C	\	92				
55	2D	-	45	135	5D	]	93				
56	2E	.	46	136	5E	^	94				
57	2F	/	47	137	5F	_	95				

APPENDIX B - POWERS OF TWO

2**n	n	hex
1	0	1
2	1	2
4	2	4
8	3	8
16	4	10
32	5	20
64	6	40
128	7	80
256	8	100
512	9	200
1 024	10	400
2 048	11	800
4 096	12	1000
8 192	13	2000
16 384	14	4000
32 768	15	8000
65 536	16	1 0000
131 072	17	2 0000
262 144	18	4 0000
524 288	19	8 0000
1 048 576	20	10 0000
2 097 152	21	20 0000
4 194 304	22	40 0000
8 388 608	23	80 0000
16 777 216	24	100 0000
33 554 432	25	200 0000
67 108 864	26	400 0000
134 217 728	27	800 0000
268 435 456	28	1000 0000
536 870 912	29	2000 0000
1 073 741 824	30	4000 0000
2 147 483 648	31	8000 0000
4 294 967 296	32	1 0000 0000
8 589 934 592	33	2 0000 0000
17 179 869 184	34	4 0000 0000
34 359 738 368	35	8 0000 0000
68 719 476 736	36	10 0000 0000
137 438 953 472	37	20 0000 0000
274 877 906 944	38	40 0000 0000
549 755 813 888	39	80 0000 0000
1 099 511 627 776	40	100 0000 0000
2 199 023 255 552	41	200 0000 0000
4 398 046 511 104	42	400 0000 0000
8 796 093 022 208	43	800 0000 0000
17 592 186 044 416	44	1000 0000 0000
35 184 372 088 832	45	2000 0000 0000
70 368 744 177 664	46	4000 0000 0000
140 737 488 355 328	47	8000 0000 0000
281 474 976 710 656	48	1 0000 0000 0000

APPENDIX C

F.A.R. CODES

ACTIVITY CODES

CODE	DESCRIPTION	EXPLANATION
A	Remedial Maintenance	Customer requested repair of Apollo supported equipment
B	Preventive Maintenance	Apollo initiated PM effort
C	Modification/Retrofit	Add-on, ECO update, etc.
D	Installation	Self-explanatory
E	Move	Self-explanatory
G	New Equipment Warranty	Equipment is warrantied free of manufacturing and material defects for a period of 90 days after installation, on standard terms of sale.
H	Service Warranty	Serviced equipment is warrantied 30 days from repair date. Warranty only covers the "repaired area of defect".
I	Pre-Contract Inspection	Covers time spent inspecting, diagnosing, etc. system prior to contract acceptance.
J	Pre-Installation Planning	Time spent planning site specifics for customer (at customer request - billable) layout, power requirements, etc.
K	Training (Formal)	Customer training given in a formal environment at customer request (needs approval by Director of Technical Support). Billable to customer.
L	Apollo Responsibility	Activity performed on equipment to correct a problem caused by Apollo personnel, media, etc.
M	Customer Liaison	Used to cover a situation where a shipment, in the opinion of the customer, contains missing, incomplete or incorrect equipment, supplies, software, documentation, etc.
N	Stand-by (on-site)	Covers time spent on-site observing equipment functioning (billable if customer requested)
O	Unit Re-build	Time spent in office repairing machines sub-assemblies for re-use or return to home office.
P	Training (Informal)	Covers time spent answering questions about how the system works.
Q	Presales	Time spent selling equipment (i.e., technical presentations) or researching sales information.
R	Other	If none of the aforementioned activity codes sufficiently explains your repair activity, use this category and explain it in the "comments" section.

CAUSE CODES

CODE	DESCRIPTION
01	Adjustment
02	Alignment
03	Cable connection
04	Component malfunction
05	Connector
06	Customer Supplied Option
07	Dirty
08	Documentation
09	Incomplete
10	Media
12	No Problem Found
13	Operator Problem
15	Defective PCB/Assembly
16	Re-seated PCB/Assembly
19	Unknown
20	Vendor Problem
21	Wiring Error
22	Worn
23	Defective Switch/Pushbutton
24	Handling/Shipping
28	Other - Explain in "Comments" section
29	Customer Initiated
30	Apollo Initiated
98	Corrective Action Rectifies Otherwise DOA
99	DOA System, Assy, or Spare Part

PRODUCT CODES

TYPE	CODE	DESCRIPTION
		DN400/420/600
PCBS	420	8 MHZ CPU Board
	951	10 MHZ CPU Board
	2001	10 MHZ CPU Board Rev 6
	422	Ring Disk Board
	2083	Ring Disk Board
	423	Display Board
	421	1/4 MB Memory Board
	755	1/2 MB Memory Board
	ADM-HMB	1/2 MB Memory Board
	757	1 MB Memory Board
	ADM-1MB	1 MB Memory Board
	979	Performance Enhancement Board
	PEB	Performance Enhancement Board
	624	NPA Board
	749	PBA Board
	1161	Array Board
	1166	Control Board
	1012	Mag-Tape Controller Board
	1380	SMD Controller Board
	2670	PCB Modem - "BNC" Type
	424	PCB Modem - "F" Type
Keyboards (High Prof.)	154	Keyboard
	750	Keyboard
	885	Keyboard w/Touchpad
CRTS	155	15" Green/Black 60HZ
	704	15" Green/Black 50HZ
	879	15" Black/White 60HZ
	1193	15" Black/White 50HZ
	1170	19" Black/White 60HZ
	1449	19" Black/White 50HZ
	2831	19" Black/White 60HZ F.C.C.
	2832	19" Black/White 50HZ F.C.C.
	2131	19" Color 60HZ
	2132	19" Color 50HZ
Power Supplies	93	24 Volt Linar
	88	Joule P/S-Dual
	1767	Todd Power Supply
	691	PNA Power Supply-Single
	1428	Todd Multi-Output - Color
	1593	5+ Color P/S
	4951	5+ Color P/S F.C.C.
	2603	Todd Retrofit P/S Kit
Drives	89	Floppy Drive 60HZ 8" Dual Side
	585	Floppy Drive 50HZ 8" Dual Side
	87	34 MB Winchester 14" Priam
	1616	34 MB Winchester 8"
	783	68 MB Winchester Priam
	780	154 MB Winchester Priam
	633	Cipher Mag Tape Drive

TYPE	CODE	DESCRIPTION
	DN400/420/600 (CONT.)	
Cables	438	Winchester
	439	Floppy
	657	NPA/PBA
	1470	B/W CRT/Node 20' (Old Style)
	2619	B/W CRT/Node 20' (F.C.C.)
	1471	B/W CRT/Node 507
	1584	Color CRT/Node (Old Style)
	2734	Color CRT/Node (F.C.C.)
	945	CPU/SIO Board
	3482-00	SMD Control 20'
	3482-001	SDM Control 50'
	3486	SMD Read/Write 20'
	3486-001	SMD Read/Write 50'
	3488	SMD Adapter
	769	Magtape Cntlr/Drive
	760	NPA/Line Printer 24'
	2011	NPA/Line Printer 40'
	1588	Array/Control Color
	1589	2 Array/Control Color
	1578	Top Hat 2 Board
	1429	Top Hat 3 Board
	1730	Backpanel 10 Slot
	1463	Color Convergence
	2648	Backplane Shield
Other	2529	Grounding Kit
	902	SIO Card
	1730	10 Slot Backplane
	2826	Universal Backplane
	2684	Backplane Shield
	1702	Panel Univ. Chasis Push Buttons
	2677	Cable Adapter for BNC Modem 267
	DN460/660 (TERN)	
	3044	CPU 1 Board
	3049	CPU 2 Board
	3054	I/O Board
	3273	TOPHAT
	2509	AC Dist. 60HZ
	2510	AC Dist. 50HZ
	ADM60-1MB	1 MB Memory Board
	ADM60-2MB	2MB Memory Board
	3408	Standoff 1/4"

LIBERTY DISK DRIVES ASSY

3328	Fan Assembly
3312	Fan
3329	80 MB Comp Drive
3330	167 MB Comp Drive
2670	Modem Board
2748	Adaptor Board
3195	Cable, Control
3196	Cable, Data
3197	Cable, POW
3198	Cable, POW
3199	Cable, POW
3201	Cable

ETHERNET

3613	PCB
3614	Cable
3615	Tranceiver
COM-X.25	

DN550 (STINGRAY)

PCBS/ASSY	3351	TTL Ring PCB
	4141	CPU .5MB
	4145	CPU 2MB
	3354	VME 1MB
	3774	600 VME Color
	3954	Color Control
	3583	Transceiver
	4151	VME .5MB
	3349	CPU Power Supply
	3105	5 1/4 Disk
	3107	Disk/Tape Controller
	3106	1/4" Cart. Tape
	3104	Disk/Tape Module
	3082	Processor, Sub Mod.
	3561	Multibus Module
	4522	PCB, Fan Brd.
	4021	VME Terminator
	4155	VME Memory
	3357	M/B VME Interface
	3525	Filter Assembly
	2846	Floating Pt. PCB
	3101	Triac Assembly
	3088	PCB, SIO
	4339	PCB, Surge Suppressor
	4335	PCB, A/C Dist.
	3083	SIO Mod.

TYPE	CODE	DESCRIPTION
	DN550 (STINGRAY CONT.)	
Power Supplies	3794	Disk/Tape Power Supply
	3251	M/B Power Supply
Backplanes	3112	Backplane
	4011	Backplane
Cables	3572	Tape Cable
	3575	Cable, Disk
	3576	Cable
	3577	Cable, Ribbon
	3922	Cable
	3923	Cable
	3925	Cable, I/O
	3948	Cable, Backpanel
	4655	Cable, Fan
	4278	Cable, Disk Controller
	4279	Cable
	3090	Cable
	4090-002	Cable, Fan
	4091-002	Cable, Fan
	4600	Cable, Fan
	3100	SIO Cable
Other	3550	Dual Vault Fan
	3532	Filter
	3536	VME Flex Cir.
	3539	Control Panel
	3545	O/O Switch
	4583	Fan Assembly
	3544	Switch
	540	Fuse
	DN300/DN320	
PCBS	1496	CPU/Memory Board
	1496-002	CPU/Memory Board (DN320 Only)
	1492	Ring/Display Board
	1494	.5 MB Memory Board
	2297	1 MB Memory Board
	2018	Touchpad-Low Profile
	1852	Disk Controller Board
	2181	TTL/Ring Board
	2146	Network Transceiver Board
	2282	Network Transceiver Assy.
	2846	Floating Pt. Board (DN320 Only)
CRTS	1654	17" Black/White 50/60 HZ
Keyboards	1660-000	Keyboard
	1666-001	Keyboard w/Touchpad
	1666-002	Keyboard for use w/Mouse
Power Modules	1645-000	CPU Power Module 60HZ (4-Slot B/P)
	1645-001	CPU Power Module 50HZ (4-Slot B/P)
	3940-000	CPU Power Module 60HZ (5-Slot B/P)
	3940-001	CPU Power Module 50HZ (5-Slot B/P)

TYPE	CODE	DESCRIPTION
	DN300/DN320 (CONT.)	
Power Supplies	1765	CPU Power Supply
	1766	Disk Power Supply
Disk Drives	2009	34 MB Disk Drive
	2009-001	70 MB Disk Drive
Floppy Drive	2020	Floppy Drive
Disk Drive Units	2290-000	34 MB Disk Unit 60HZ
	2290-001	34 MB Disk Unit 50HZ
	2290-003	70 MB Disk Unit 60HZ
	2290-004	70 MB Disk Unit 50HZ
Disk/Floppy Drive Units	2516-000	34 MB Disk/Floppy Unit 60HZ
	2516-001	34 MB Disk/Floppy Unit 50HZ
	2516-003	70 MB Disk/Floppy Unit 60HZ
	2516-004	70 MB Disk/Floppy Unit 50HZ
Cables	2370	Disk Ribbon Cable
	2371	Floppy Ribbon Cable
	2372	Mem Exp Ribbon Cable
	2368	Ribbon Cable
	1650	DC Power Cable (4 Slot)
	1929	Keyboard Ribbon Cable
	2545	VDU Brightness Cable
	3160	CRT Cable
Backpanels	1638	Upper Backpanel (4 Slot)
	1944	Lower Backpanel (4 Slot)
	3489	Backpanel (DN320 5 Slot)
	DSP80 SPARES	
	2422	CPU Board
	4590	CPU Extension Board
	2811	HMB Mem Expan Board
	2420	Lower Back Panel
	2463	Multibus Back Panel
	2440-000	Assembly Chassis
	2360	Cable 28A
	2445	Top Plate
	2446	Facia Sapphire
	2448	Switch
	2451	Bottom Plate
	2459-000	Chassis Module
	3912	Processor Cover
	1765	Multi-Power Supply
	2425-000	DSP80 Power Supply
	2181	TTL/Ring Board
	3451	Network, PCB
	2146	Network Transceiver Board
	2282	Network Transceiver Assy.

TYPE	CODE	DESCRIPTION

OTHER HARDWARE PRODUCT CODES

150	Comm H/W
151	X.25
152	Ethernet
171	GE Printer
172	Spinwriter Printer
173	Printronix Printer
190	Tape Drive
191	Motor Control PCB for 190
198	Misc H/W
Net-Swt-1	Network Switch
1518	"F" Type Connectors
2521	BNC Type Connectors
2824	Mouse

F.A.R. PRODUCT CODES - SOFTWARE

281	Network S/W
282	Network Administration
351	Aegis
352	Fortran 77
353	Pascal
354	C
355	D3M
356	Sig Core
357	Aux
358	DPSS
359	DSEE
360	VERSATEC
361	TCP/IP
362	GPIO
363	LISP
364	
370	MISC S/W

F.A.R. MODEL TYPES

NODES	SERVERS	PRINTERS	STORAGE DEVICES
DN300	DSP80	HCD-MMP	MSD-300M
DN320	DSP80R	HCD-LQP	MSD-500M
DN400	DSP160	HCD-300	MSD-1000M
DN416	DFS-500M	HCD-600	MSD-1600M
DN420	DFS-1000M		
DN460			
DN550			
DN600			
DN660			

READER'S RESPONSE

Apollo Computer uses readers' comments in revising and improving our documents.

Document Title: Apollo DOMAIN Engineering Handbook
Document Revision: 03
Order Number: 002398

What is the best feature of this manual?
_____

_____

_____

Please list any errors, omissions, or problem areas in the manual. (Identify errors by page, section, figure, or table number wherever possible.)
_____

_____

_____

_____

What type of user are you?

___ Systems programmer; language _____
___ Applications programmer; language _____
___ Apollo SSR

How often do you use the Apollo system? _____

Nature of your work on the Apollo system: _____

_____
Your name Date

_____
Organization

_____
Street Address

_____
City State Zip/Country
No postage necessary if mailed in the U.S.