DOMAIN
ENGINEERING HANDBOOK

Order No. 002398
Revision 01
Software Release 6.0

APOLLO COMPUTER INC. 15 Elizabeth Drive
Chelmsford, Massachusetts 01824

First printing: September, 1981
Last printing: April, 1983

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

TABLE OF CONTENTS

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 8 chapters and two appendices.

Chapter 1 describes principal control blocks associated with memory and process management.

Chapter 2 describes hardware architecture of Apollo nodes; basic processor and Memory Management Unit (MMU).

Chapter 3 describes display hardware including display boards, control and status registers, and BLT registers.

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

Chapter 5 describes formats of file objects.

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

Chapter 7 describes miscellaneous information useful to those programming in the DOMAIN environment.

Chapter 8 describes system debugging tools, including the mnemonic debugger, extensions to the HLL debugger, and system dumps.

Appendix A describes the ASCII character set.

Appendix B is a powers of two table.

Related Documents

For information about system calls, see Apollo System Programmer's Reference Manual

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

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

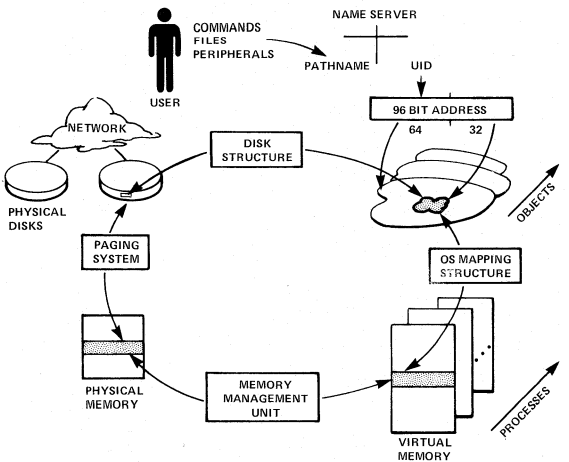
Conventions

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
=>	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

CHAPTER 1

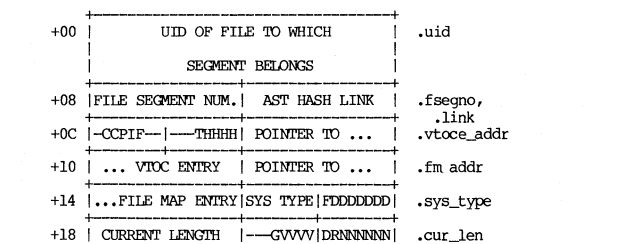
AEGIS

AEGIS SYSTEM RELATIONSHIPS



ACTIVE SEGMENT TABLE (AST)

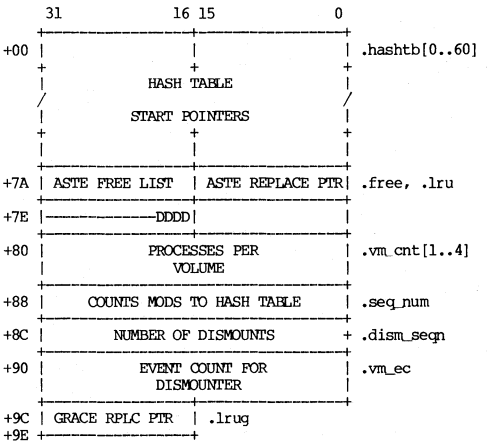
AST: ARRAY[1..64] OF ASTE_T
ACTIVE SEGMENT TABLE ENTRY (ASTE):
(type "aste_t" in vm.ins.pas)



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 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.

ACTIVE SEGMENT TABLE HEADER



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

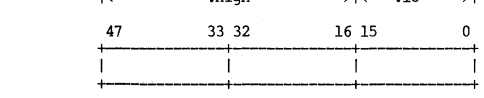
BOOT SHELL COMMANDS

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
DMTVOL	{W | S | F} <lvno> [<pathname>]	dismount a logical volume
EX	<pathname>	execute (rfc) file
GLOB		list installed globals
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 lites address
LO	<pathname> [-D] [-S | -NS]	invoke the loader w/ the given OPC file
MA	<pathname> [<l> <sz>] [-E]	map file
MTVOL	{W | S | F} <lvno> [<pathname>]	mount a logical volume
ND	<pathname>	set naming directory
RE	<pathname>	restore file into static mem
REL	[-A]	release proc-mgr assigned storage
SA	<pathname> [<l> <h> <s>]	save file given lo, hi, start addrs
SB	<pathname> [<l> <h>]	save raw data file given lo, hi addrs
SHUT		shutdown system
STCODE	<status-code>	print textual 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>	un-catalog pathname from namespace
UMA	{<pathname> | <l> <sz>}	unmap file by name or addr/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, mt, boot }

CLOCK

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



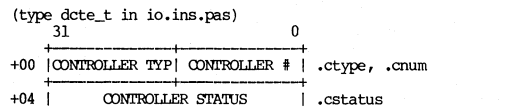
Low-order bit represents 4 microseconds.

Hardware clock is only 16 bits. Upper 32 bits are maintained by clock interrupt routine. There are 3 hardware clocks (not counting calendar clock):

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

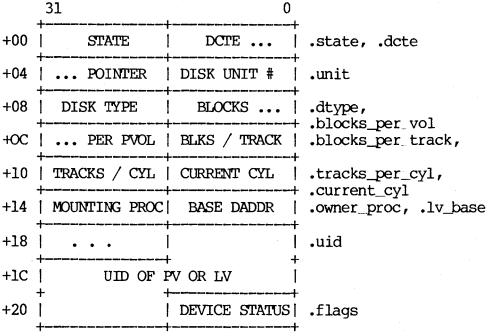
DISK CONTROLLER TABLE ENTRY

(type dcte_t in io.ins.pas)



DISK VOLUME TABLE ENTRY

(type dvte_t in disk.pvt.pas)

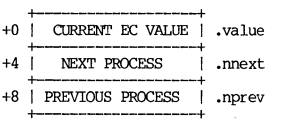


state: free
being_mounted
assigned
mounted

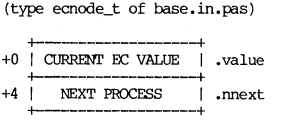
flags: write_protect

EVENT COUNT

(type eventcount_t of base.ins.pas)

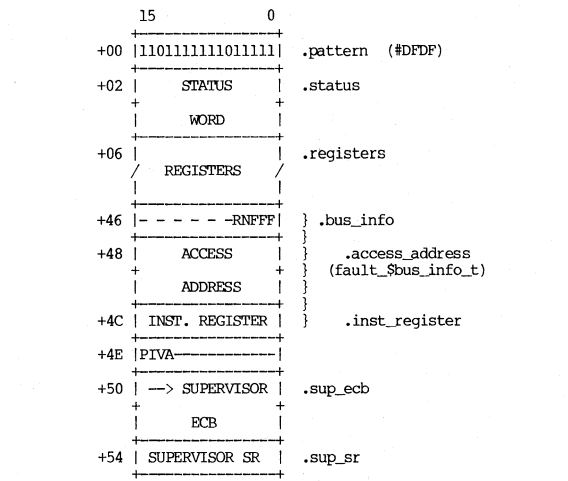


(type ecnode_t of base.in.pas)



FAULT DIAGNOSTIC RECORD

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



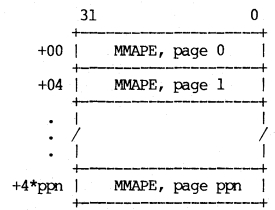
R - Read operation (.write_op)
N - Not instruction reference (.not_inst)
FFF - Instruction function code (.function_code):
  001 User data
  010 User program
  101 Supervisor data
  110 Supervisor program
  111 Interrupt acknowledge
P - Permit return from fault (.return_permitted)
I - In supervisor (.in_supervisor)
V - Registers valid (.registers_valid)
A - AB valid (fault_$bus_info_t is valid) (.ab_valid)

MAPPED SEGMENT TABLE (MST)

```
MST: ARRAY[0..15,0..319] OF MSTE

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

     31              16 15              0
    +------------------+-----------------+
+00 |            UID OF FILE             |  .uid
    +------------------+-----------------+
+08 | FILE SEGMENT # |EAAAAAGPPPPPPPPP|  .fsegno
    +------------------+-----------------+
+0C |   INDEX OF VTOC ENTRY (VTOCX)      |  .locx
+10 +------------------------------------+

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)
```

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

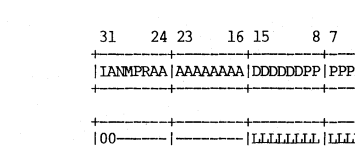
MEMORY MAP (MMAP)



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

MEMORY MAP ENTRY (MMAPE)

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

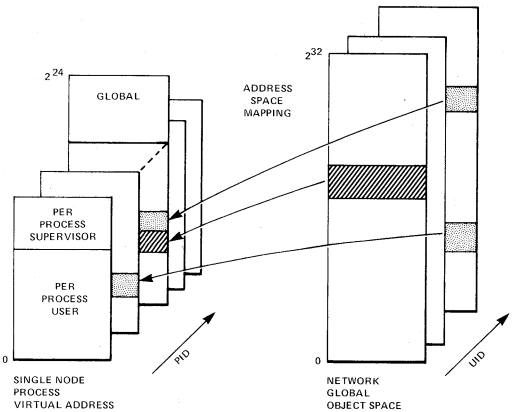


I = 1 => Entry in use (.inuse)
A - False if page wired or in transit (.avail)
N = 1 => No copy on disk (.null)
M - Page mod'ed but dtm shouldn't be updated (.rmod)
P = 1 => Purifier used (.usedp)
R = 1 => Replacer used (.usedr)
A..A - ASTE index (.astex)
D..D - Disk addr high (.daddr_h)
P..P - PTT index (.pttx)
S..S - Segment page number (.mspgno)
L..L - Link to next free MMAP entry (.link)
L..L*4 to get displacement in table of next entry

IA Meaning

00 Free page or permanent non-file system (e.g., PROM).
01 Not used.
10 Page not available for paging out.
11 Page is available -- purifier & replacer will look at it.

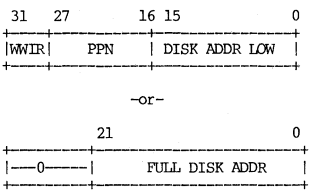
OS MAPPING


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

PAGE MAP

PMAP: ARRAY[0..31] OF PMAPE (IN ASTE_T)

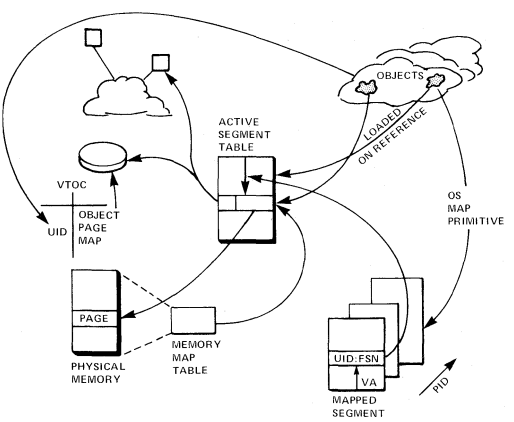
PAGE MAP ENTRY:
(type "pmape" in vm.ins.pas)



WW - Wired count (.wired)
I - Page in transition (.in_trans)
R - Page resident (.resident)

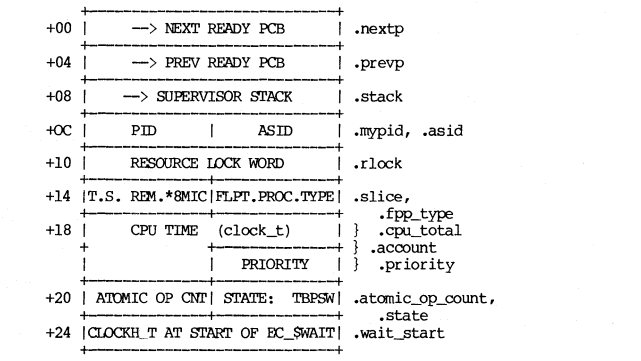
A page map lives in an AST entry. If resident, the high-order bits of the disk address are in the MMAP entry for the page.

PAGING SYSTEM



PROCESS CONTROL BLOCK (PCB)

(type "procl_t" in PROCl.PAS)



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

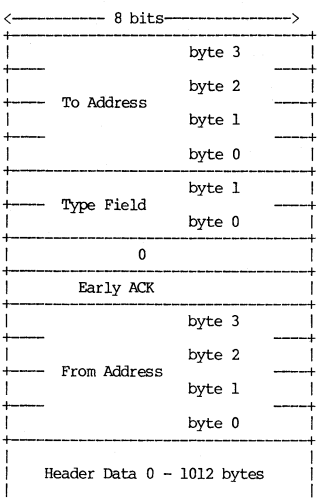
PID	ASID	Description
1	1	Initial user process and DM
2	0	Null process
3	0	Clock process
4	0	Page purifier
5	0	Network-receive server
6	0	Network-paging server
7	0	Network-general server
8	0	Terminal helper
9	2	First user process
10	3	Second user process
11	4	.
12	5	.
13	6	.
14	7	.
22	15	Last user process

Note: Processes whose ASID is 0 run entirely in global space.

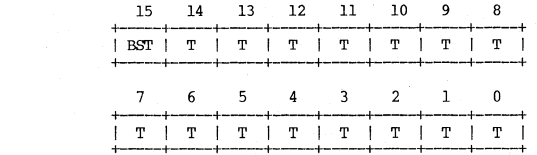
Note: If the debugger's lights command has been given, there will be a Lights process with id 9 or greater with a 0 ASID.

RING PACKET FORMAT

Message Header



Type Field

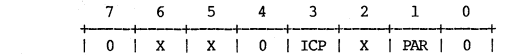


Bit 15 - BST - Broadcast - This bit is set in a packet intended to be broadcast to all receivers. If it is set, the To Address field is ignored.

Bit 14 thru 0 - T - Type - This field is used to determine 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 only implements Bits 14 through 8, Bit 7 through 0 never match.

Early ACK 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 as if it is a byte of 0's. This is to allow 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.

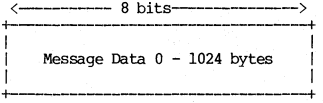
Bit 3 - ICP - Intend to Copy - This bit is set by an addressed receiver if it was set up to copy a message and the type field matched. A NAK (negative acknowledge) condition is indicated when no receiver sets this bit.

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

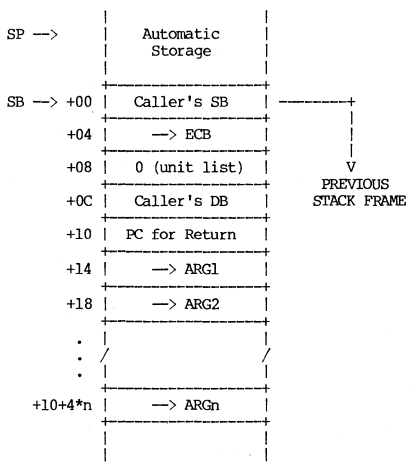


The Message Data field may vary in length from 0 to 1024 bytes but must always be an even number of bytes (on the Apollo I the controller always uses word DMA). The contents of Message Data field is determined by the software, it is transmitted verbatim.

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 runnable 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 data base }
```

STACK FRAME



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

STATUS WORD

("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 also CHAPTER 4 ERROR CODES.

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
/ftn       Fortran (optional)
/gpio      General Purpose I/O (optional)
/install   Installation scripts directory
/lib       System libraries directory
/pas       Pascal (optional)
/sau       Stand-alone utilities directory
/saul      Stand-alone utilities directory
/sau2      Stand-alone utilities directory
/sys
   /sys/boot              Boot shell file
   /sys/cdict             Dictionary for FSERR command
   /sys/dict              Dictionary for FSERR command
   /sys/dictdx            Index for dictionary
   /sys/dm                Display manager programs
   /sys/dmenv             Command to invoke display manager
   /sys/dm/fonts          Display manager font definitions
   /sys/env               Software tools environment (no dm)
   /sys/help              Help text files
   /sys/ins               User insert files
   /sys/node_data[.nn]    Per node read/write data files
   /sys/stream_$sfcbs     Stream mgr control blocks
/sysboot   System boot file
/systest   On-line system tests directory
```

TRAP CODES

```
0    SVC — 0 arguments
1    SVC — 1 arguments
2    SVC — 2 arguments
3    SVC — 3 arguments
4    SVC — 4 arguments
5    SVC — any number of arguments
6    Undefined (reflected to user space)
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

CPU AND MEMORY

Three of the tables in this chapter -- Addressing Modes, Condition Codes, and Conditional Tests -- are extracted from the MC68000 16-Bit Microprocessor User's Manual, Second Edition. (Austin, Texas: Motorola Semiconductor Products, January 1980).

ADDRESS SPACE

DN300

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
9400	disp l	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->FFC7FF
5000	big pft	FFC800->FFF7FF

DN4xx and DN600

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
	iomap	FFF800->FFF9FF

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.

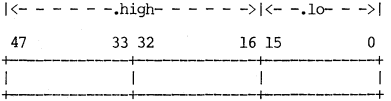
CACHE

[C400|FF7C00] [C800|FF8000]

To turn cache off, write a 5 to the PEB control register. See Chapter 6, PEB.

CLOCK

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



Low-order bit represents 4 microseconds.

Hardware clock is only 16 bits. Upper 32 bits are maintained by clock interrupt routine. There are 3 hardware clocks (not counting calendar clock):

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

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	*	*	?	?	?	$V = Sm \cdot Dm \cdot \overline{Rm} + \overline{Sm} \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot \overline{Dm} + \overline{Rm} \cdot Dm + Sm \cdot \overline{Rm}$ $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
AND, ANDI, EOR, EORI, MOVEQ, MOVE, OR, ORI, CLR, EXT, NOT, TAS, TST	–	*	*	0	0	
CHK	–	*	U	U	U	
SUB, SUBI SUBQ	*	*	*	?	?	$V = \overline{Sm} \cdot \underline{Dm} \cdot \overline{Rm} + \underline{Sm} \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot \overline{Dm} + Rm \cdot \overline{Dm} + Sm \cdot Rm$
SUBX	*	*	?	?	?	$V = \overline{Sm} \cdot \underline{Dm} \cdot \overline{Rm} + \underline{Sm} \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot \overline{Dm} + Rm \cdot \overline{Dm} + Sm \cdot Rm$ $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
CMP, CMPI, CMPM	–	*	*	?	?	$V = \overline{Sm} \cdot \underline{Dm} \cdot \overline{Rm} + \underline{Sm} \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot \overline{Dm} + Rm \cdot \overline{Dm} + Sm \cdot Rm$
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	*	*	*	?	?	$V = Dm \cdot Rm$, $C = Dm + Rm$
NEGX	*	*	?	?	?	$V = Dm \cdot Rm$, $C = Dm + Rm$ $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
BTST, BCHG, BSET, BCLR	–	–	?	–	–	$Z = \overline{Dn}$
ASL	*	*	*	?	?	$V = \underline{Dm} \cdot (\overline{D_{m-1} + \ldots + D_{m-r}}) + \overline{Dm} \cdot (D_{m-1} + \ldots + D_{m-r})$ $C = D_{m-r+1}$
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 affected
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.

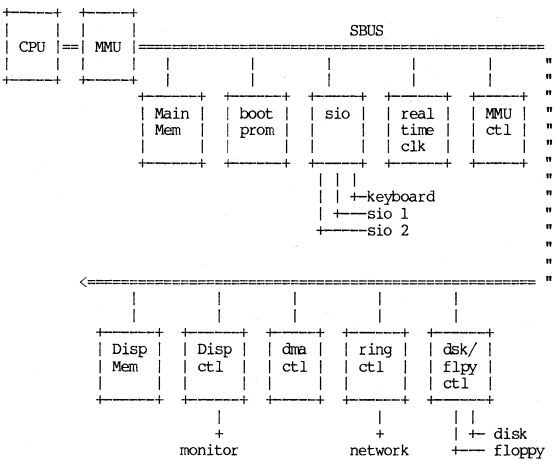
CONDITIONAL TESTS*

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	$Z + N \cdot \overline{V} + \overline{N} \cdot V$

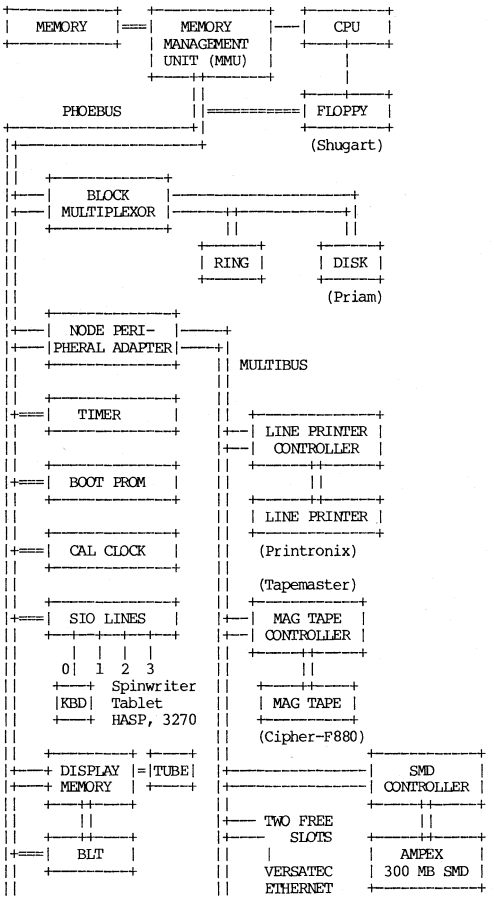
*Reprinted from MC68000, page A-4.

CONFIGURATION

DN300



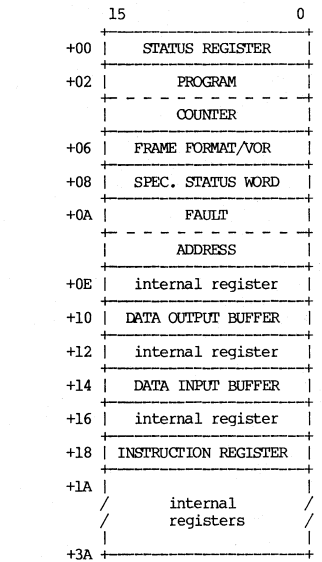
DN4xx and DN600



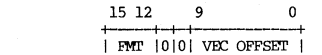
EXCEPTION ERROR STACK FRAME

DN300

BUS/ADDRESS ERROR STACK FRAME FORMAT

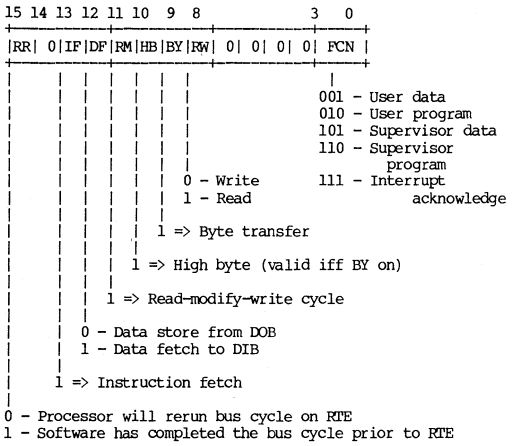


FRAME FORMAT/VECTOR OFFSET WORD



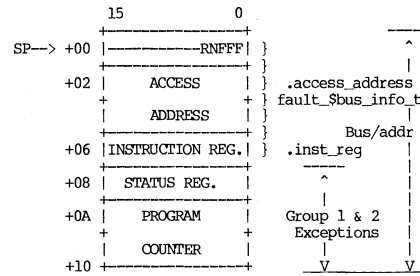
0000 - 4-word format: SR, PC, VOR
1000 - 29-word 68010 format.

SPECIAL STATUS WORD



DN4xx and DN600

(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

EXCEPTION 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.

EXCEPTION VECTORS

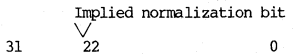
Exception vector at [100400 | 0].
TRAP PAGE (+ => DN300 only; - => unused on DN300)

	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	int level
+	19	064	SIO (rcv and xmit)	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 Int Vectors - unused	
-	90-9F	240	Ring/disk board	
	A0-AF	280	User Int 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 Int 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

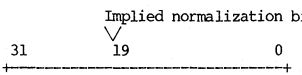
FLOATING POINT FORMAT

Single Precision Floating Point Format



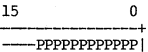
S - Sign; S = 1 => Negative
E..E - Exponent plus 127
M..M - Mantissa

Double Precision Floating Point Format



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

IO MAP



P..P - Physical page number (PPN)
(type "ppn_t" in base.spo.bbas)

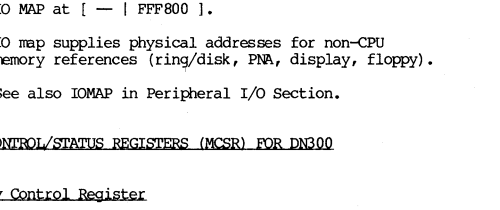
IO MAP at [— | FFF800].

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

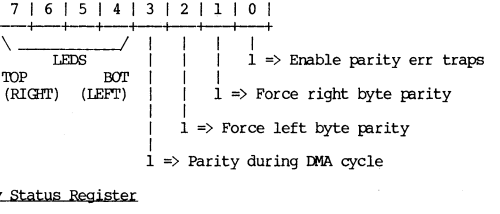
See also IOMAP in Peripheral I/O Section.

MEMORY CONTROL/STATUS REGISTERS (MCSR) FOR DN300

Memory Control Register



Memory Status Register



Writing MSR clears parity error condition.

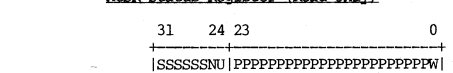
```
Board 1:  [ FC02 | FF9002 ]
Board 2:  [ FD02 | FF9102 ]
Board 3:  [ FE02 | FF9202 ]
Board 4:  [ FF02 | FF9302 ]
```

MCSR Control (Write-Only)

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 FOR DN4XX AND DN600

Board #	JP3	JP4	Abs 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 2 512kb boards or 1 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

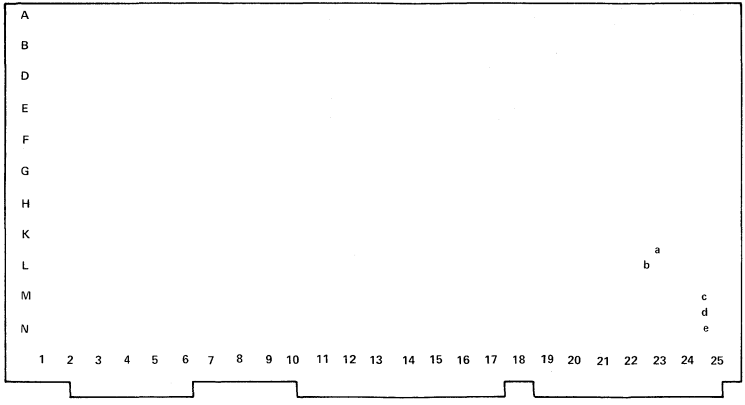
1MB Board #	JP13	JP14	JP15	Address Range
1	in	out	out	100000-1fffff
2	out	in	out	200000-2fffff
3	out	out	out	300000-3fffff
4	*** not allowed ***			

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

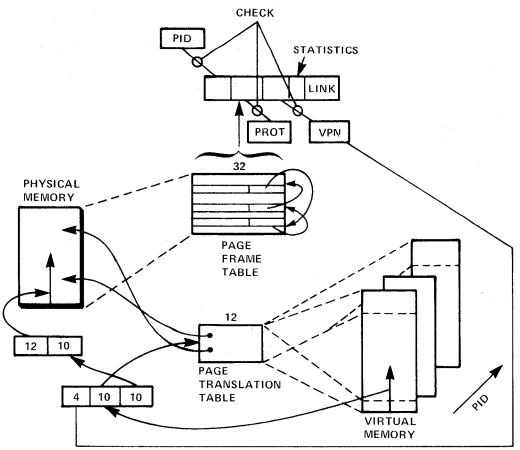
Size	256K	512K	1M	256K adr	512K adr	1M adr
1.0 M	0	0	1	–	–	1:100
2.0 M*	0	0	2	–	–	1:100 2:200
3.0 M*	0	0	3	–	–	1:100 2:200 3:300
0.5 M	0	1	0	–	1:100	–
1.5 M	0	1	1	–	1:100	2:200
2.5 M*	0	1	2	–	1:100	2:200 3:300
3.5 M*	0	1	3	–	4:080	1:100 2:200 3:300
1.0 M	0	2	0	–	1:100 2:180	–
2.0 M*	0	2	1	–	1:100 2:180	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.

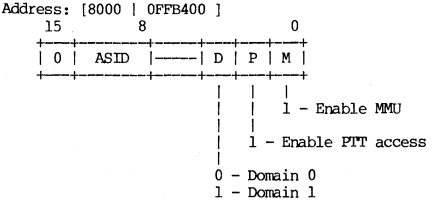


LABEL			COORDINATE LOCATION (APPROXIMATE)
a	=	JP3	L-23
b	=	JP4	L-23
c	=	JP13	M-25
d	=	JP14	M-25
e	=	JP15	N-25

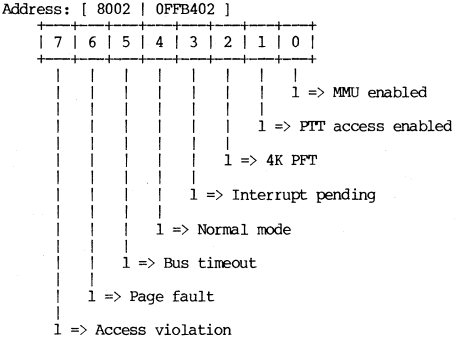


	DN4xx and DN600		DN300
PID/PRIV/POWER	[8000	FFB400]	[8000	FFB400]
CPU A Control	[8002	FFB402]	
MMU Status, High	[8004	FFB404]	[8002	FFB402]
MMU Status, Low	[8006	FFB406]	
Clear MMU Status	[8008	FFB408]	
Bus Status	[800A	FFB40A]	
Enable CPU B	[800E	FFB40E]	

DN300 PID/PRIV Register



DN300 MMU Status Register

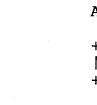


DN4xx and DN600 PID/PRIV Register

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 - PIT access enable
M - Enable MMU (virtual memory operations)

DN4xx and DN600 CPU A Control Register

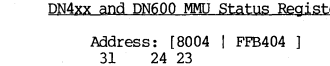
Address: [8002 | FFB402]



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

DN4xx and DN600 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

DN4xx and DN600 Clear MMU Status

Address: [8008 | FFB408]

When written, clears MMU status conditions.

DN4xx and DN600 Bus Status Register

Address: [800A | FFB40A]

I = 0 => unacknowledged interrupt pending
M - MMU enabled
P - PFT 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)

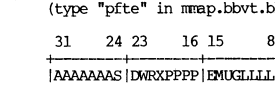
DN4xx and DN600 Enable CPU B Register

Address: [800E | FFB40E]

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

PAGE FRAME TABLE ENTRY (PFTE)

(type "pfte" in mmap.bbvt.bbas)



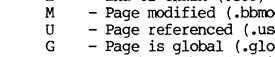
A..A -- Address space ID (0-127) (.elsid)
S - Supervisor domain (.elccess)
D - Domain (0 or 1)
W - Write access
R - Read access
X - Execute access
P..P - Excess virtual page number (.xsvpn)
E - End of chain (.eoc)
M - Page modified (.bbmod)
U - Page referenced (.used)
G - Page is global (.global)
L..L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

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

One entry per physical page of memory.

PAGE TRANSLATION TABLE ENTRY (PTTE)

(type "ppn_t" in base.spo.bbas)

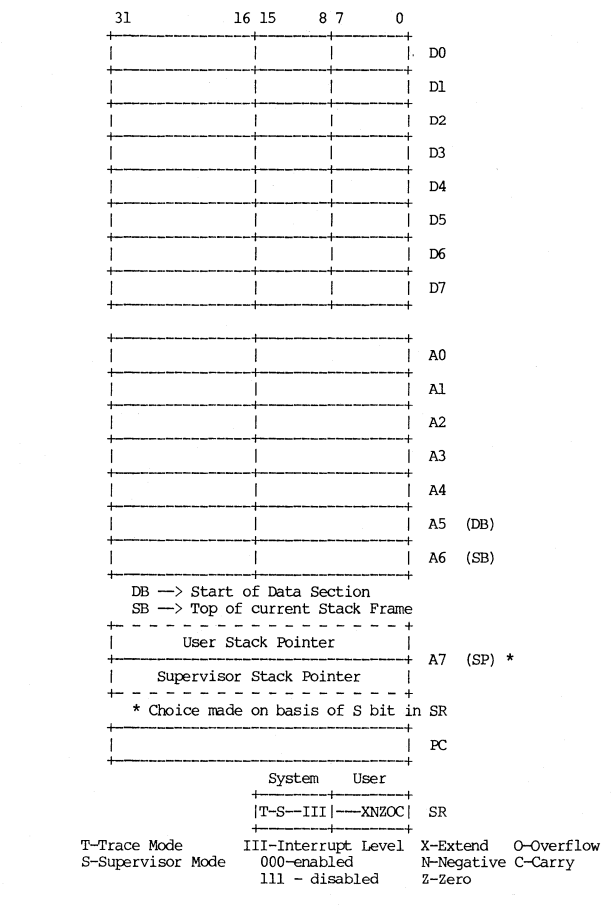


P..P - Physical page number (PPN)

Page Translation Table at [n/a | 700000].

One PPTE every 1024 bytes in table.

REGISTER SET



CHAPTER 3

DISPLAY HARDWARE

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

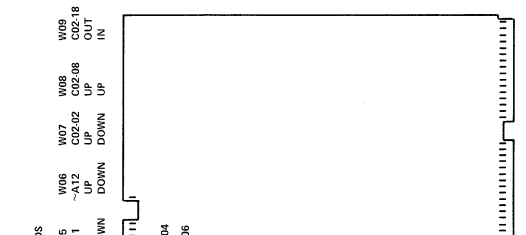
19 BLACK & WHITE CRT

W01-DOWN	W02-UP	W03-UP	W04-DOWN	W05-DOWN
W06-DOWN	W07- *	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

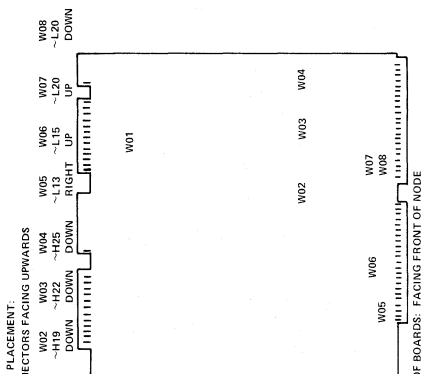
* W07 AND W16 SHOULD BE SET BY SPECIFICATIONS ON CLOTH TAG, MOUNTED ON PCB.

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

ARRAY BOARD JUMPER PLACEMENT:



CONTROL BOARD JUMPER PLACEMENT:



DISPLAY CONTROL AND STATUS REGISTER (DCSR)

DN300

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]

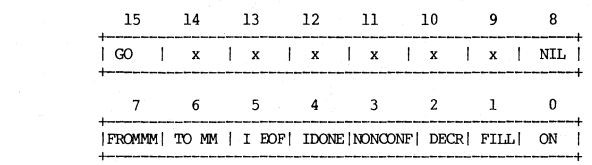
8000	- GO (Start blt operation)	15
0020	- Interrupt at end of frame	5
0010	- Interrupt at end of blt operation	4
0008	- Increment Y coordinate	3
0004	- Increment X coordinate	2
0002	- Fill mode blt operation	1
0001	- Enable display (blank if reset)	0

DISPLAY STATUS REGISTER [9400 | 0FF9800]

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

DN4xx

[F000 | FF9800]



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 at 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 BLT's 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 BLT's 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.

DN600

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

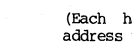
BIT	NAME	FUNCTION
15	DONE-	This bit is asserted by micro code to indicate that the micro machine is ready to handle reads or writes to its command pages
14	VBLANK-	When this bit is asserted the display is currently in the vertical blanking period.
13	INST_DONE-	This bit is asserted by micro code to indicate that 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.

BLT REGISTERS

DN300

(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 of # lines in height of destination block

DESTINATION COUNT X REGISTER [9416 | 0FF9816]

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

= two's complement 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

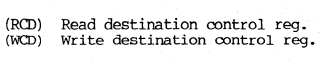
DN4xx

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

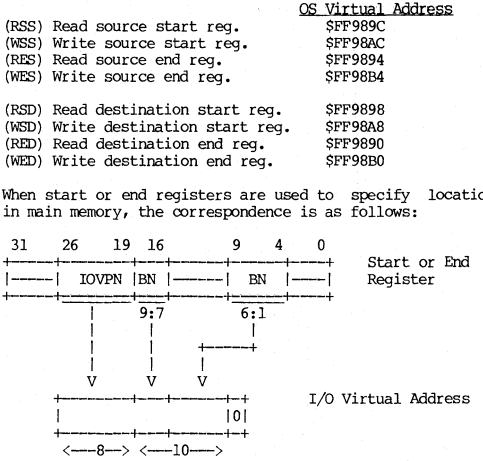
Source/Destination Start/End Registers:



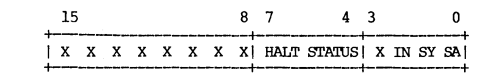
V - Used in BLT's to or from main memory

		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:



DN600



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.

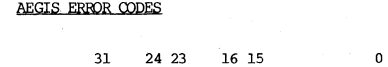
HALT STATUS

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

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

("status_$t" in base.spoas)

00000000   status_$ok

00000001 - 0000000F Program return codes

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
```

```
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

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

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

OS / MMU manager:
(00070001)  mmu miss

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) storage module manager logic error
(00080023) unknown error status from storage module controller
(00080024) unrecognized drive id

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

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)

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

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

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

OS / import/export manager:
(001A0001) entry directory is not cataloged 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

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

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

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

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

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

Streams Modules:
(0101xxxx) stream manager / open
(0102xxxx) stream manager / create
(0103xxxx) stream manager / get_rec
(0104xxxx) stream manager / get_prior_rec
(0105xxxx) stream manager / seek
(0106xxxx) stream manager / create_here
(0107xxxx) stream manager / truncate
(0108xxxx) stream manager / put_rec
(0109xxxx) stream manager / replace
(010Axxxx) stream manager / put_chr
(010Bxxxx) stream manager / close
(010Cxxxx) stream manager / delete
(010Dxxxx) stream manager / inquire
(010Exxxx) stream manager / redefine
(010Fxxxx) stream manager / open_rec
(0110xxxx) stream manager / pad_create
(0111xxxx) stream manager / memory manager
(0112xxxx) stream manager / get_conditional
(0113xxxx) stream manager / internal module
(0114xxxx) stream manager / get_ec
(0115xxxx) stream manager / internal module get_fcb.sfcb

(0116xxxx) stream manager / internal module get_fcb.pfcb
(0117xxxx) stream manager / fork
(0118xxxx) stream manager / dup
(0119xxxx) stream manager / get_buf
(011Axxxx) stream manager / alloc_sh_sk
(011Bxxxx) stream manager / free_sh_sk
(011Cxxxx) stream manager / init_sh_sk_pool
(011Dxxxx) stream manager / bypass

Streams codes:
(01xxFFFB) warning: new name added but can't delete old name
(01xxFFFC) warning: fix lgth recs & partial rec at EOF
(01xxFFFD) warning: close anomaly
(01xxFFFE) inquire warning - object open on multiple streams
(01xxFFFF) warning: file trouble flag is set
(01xx0000) unused
(01xx0001) operation attempted on unopened stream
(01xx0002) attempted operation illegal on this stream
(01xx0003) "from" stream not open on switch request
(01xx0004) no available target stream for switch
(01xx0005) can't switch - no available stream ids
(01xx0006) can't open stream to this object type
(01xx0007) no more available streams
(01xx0008) requested object is in use
(01xx0009) end of file
(01xx000A) ran out of address space
(01xx000B) permanent files must have names
(01xx000C) "put" of wrong length record
(01xx000D) internal fatal error; table re-verify failed
(01xx000E) parameter combination illegal for this stream operation
(01xx000F) operation illegal with variable length records
(01xx0010) bad position in relative record seek - cur. pos. not aligned on record boundary
(01xx0011) attempt to seek beyond beginning of file (key=0?)
(01xx0012) bad character seek (to before start of current record)
(01xx0013) bad open_XP - must be already open on stream elsewhere
(01xx0014) count field for current record is in error
(01xx0015) name not found
(01xx0016) bad file header - CRC doesn't check
(01xx0017) bad location parameter in create call
(01xx0018) inquire error - can't open stream to this type of object
(01xx0019) object already open on another stream by a different name
(01xx001A) stream id out of range
(01xx001B) unused
(01xx001C) object not found even though name exists
(01xx001D) unused

(01xx001E) illegal pad close - must close input pad first
(01xx001F) attempted name change would require file move
(01xx0020) object deleted while open on this stream
(01xx0021) name required - open with no name is illegal
(01xx0022) EOF pad put error - "put" legal only at EOF on pads
(01xx0023) bad related pad - invalid or not open
(01xx0024) unable to lock needed resources
(01xx0025) this object type is illegal for pad create
(01xx0026) internal memory management error - fatal
(01xx0027) attempted write to read only stream
(01xx0028) replace length error; record size cannot change
(01xx0029) cannot redefine this attribute for a pad
(01xx002A) something failed (inq/redef err_mask not empty)
(01xx002B) illegal attempt to open stream to remote device
(01xx002C) no access rights to object
(01xx002D) insufficient rights to object
(01xx002E) invalid data in write to pad
(01xx002F) internal table space error
(01xx0030) move mode is needed - forced locate is on
(01xx0031) illegal forced locate request - legal only for disk files
(01xx0032) no more shared file cursors available on this node
(01xx0033) internal error: shared file cursor has bad reference count (below zero)
(01xx0034) — (01xx0064) unused
(01xx0065) impl. restr.: char seek outside of current record illegal
(01xx0066) impl. restr.: either record length or offset must be <= 65535
(01xx0067) creation of edit pads unimplemented
(01xx0068) client side pad create unimplemented
(01xx0069) inquire for this object type not yet implemented

display manager / DM process:
(02010001) Command syntax error
(02010002) Unknown command name
(02010003) Too many command arguments
(02010004) Position specification syntax error
(02010005) Key definition incomplete
(02010006) File not found
(02010007) Unable to access file
(02010008) Text is read-only
(02010009) No text under cursor
(0201000A) Too many windows
(0201000B) Too many pads
(0201000C) Display manager internal error #1
(0201000D) Unable to write new file
(0201000E) You must finish current command input request before making another

(0201000F) Key definition too long
(02010010) Cursor is not in any window
(02010011) Line too long - truncated
(02010012) CC command requires position mark
(02010013) Unable to create new process
(02010014) Unable to create new pad
(02010015) Input pads cannot be replicated
(02010016) Bad font file
(02010017) Font table full
(02010018) Unknown function key name
(02010019) Window too small for any text
(0201001A) Display manager internal error #2
(0201001B) Pad deleted by client process
(0201001C) Pad still active
(0201001D) Illegal grow coordinates
(0201001E) Display manager pads cannot be named
(0201001F) File is not an ascii file
(02010020) Processes still active
(02010021) Bad text range
(02010022) Paste buffer empty
(02010023) Invalid regular expression pattern
(02010024) Pattern too long
(02010025) Illegal substitution expression
(02010026) No match
(02010027) DM input window cannot be deleted
(02010028) No room for more text at that position
(02010029) Invalid hexadecimal digit
(0201002A) Invalid tabstop value
(0201002B) Edit pad already open on that file
(0201002C) File already modified
(0201002D) No window selected for grow or move
(0201002E) Line contains non-editable controls
(0201002F) Cannot delete .bak file
(02010030) Heap file inconsistent during recovery
(02010031) Process name already exists
(02010032) Search aborted
(02010033) Operation illegal during search or substitute
(02010034) Argument value out of range
(02010035) User logging out
(02010036) Error processing user_data/key_defs
(02010037) Previous fatal error prevents typing (try RW -R)
(02010038) Nothing left to UNDO

display manager / Heap manager:
(02020001) heap table full
(02020002) invalid length for heap get
(02020003) invalid heap return
(02020004) heap space already free on return

display manager / Pad manager:
(02030001) stream is not a pad
(02030002) input pad required for this operation
(02030003) stream id out of range
(02030004) operation attempted on unopened stream

(02030005) transcript pad required for this operation
(02030006) pad not in raw mode
(02030007) value out of range
(02030008) too many fonts loaded in this pad
(02030009) error loading font file
(0203000A) unknown function key name
(0203000B) only one input pad allowed per transcript
(0203000C) illegal parameter combination
(0203000D) existing pad must be an ascii file
(0203000E) operation illegal with > 1 client process

process manager / process manager:
(03010001) invalid parameters to pm_$invoke
(03010002) process had a fatal error
(03010003) no space for Known Global Table
(03010004) process already named
(03010005) process has no name
(03010006) error initializing global read/write storage
(03010007) error allocating PFM error mask
(03010008) too many libraries — table space exceeded
(03010009) mark/release list full

process manager / loader:
(03030001) too many undefined globals
(03030002) object module has too many sections
(03030003) not enough address space to copy procedure
(03030004) file is not an object module
(03030005) object module format version is obsolete
(03030006) not enough address space to map procedure
(03030007) not enough address space for static storage
(03030008) Known-Global-Table is full
(03030009) reference to undefined global
(0303000A) library contains unresolved global(s)
(0303000B) peb required to execute this object module

process manager / process fault manager:
(03040001) cleanup handler released out of order
(03040002) invalid cleanup record
(03040003) cleanup handler established successfully
(03040004) pfm_$cleanup_set was signalled
(03040005) no rws space to create static cleanup handler
(03040006) static cleanup handler not found

process manager / program manager:
(03050001) argument does not exist
(03050002) argument is too big for supplied buffer
(03050003) stream vector is too large (>8) or permutes streams
(03050004) not a program

process manager / mapped segment manager:
(03060001) unsupported access rights requested
(03060002) attempt to release segment mapped by previous level
(03060003) no object mapped at virtual address supplied

```
(03060004)  object already locked in a way that
              precludes your requested use
(03060005)  no space
(03060006)  bad length

process manager / r/w storage manager:
none

US / file utility:
(04010001)  object must be a leaf
(04010002)  unrecognized sys_type returned from
              file_$attributes
(04010003)  unrecognized entry type returned from
              naming server
(04010004)  unable to replace or delete a system
              directory
(04010005)  compare failed
(04010006)  destination directory is contained within
              source directory
(04010007)  can't copy a file or tree to itself
(04010008)  source and destination are on different disk
              volumes

US / print utility:
(04020001)  /sys/print is not a link!
(04020002)  print queue directory specified by link was
              not found
(04020003)  file to be printed is not an ascii text file

US / DB:
(04030001)  normal exit from debugger

US / object module IO:
(04040001)  too many IO channels in use
(04040002)  channel is not open
(04040003)  get or put outside file
(04040004)  ran out of free read/write storage

US / TU58 tape:
(04050001)  bad block number
(04050002)  invalid op code
(04050003)  motor stopped
(04050004)  seek error
(04050005)  data check
(04050006)  write protected
(04050007)  no cartridge
(04050008)  bad unit number
(04050009)  end of tape
(0405000A)  failed self-test

US / sio:
(04060001)  object on this stream is not sio line
(04060002)  bad option parameter
(04060003)  illegal stream id
(04060004)  stream not open
```

US / pattern matcher:
(04070001) too many tag specifiers ("{ }")
(04070002) too many end-tag characters ("}")
(04070003) missing end-class character ("]")
(04070004) invalid use of closure ("*")
(04070005) missing end-tag character ("}")
(04070006) pattern too big to fit in supplied buffer
(04070007) invalid escape sequence
(04070008) bad pattern
(04070009) given string and pattern do not match
(0407000A) output string too small

US / wildcard processor:
(04080001) can't get free space for pathname
(04080002) invalid wildcard length
(04080003) invalid wildcard name
(04080004) invalid use of directory closure ("*" or "+")
(04080005) too many components
(04080006) internal error #1
(04080007) circular link detected
(04080008) first name found
(04080009) excess end tags ("}") in wildcard
(0408000A) missing end tags ("}") in wildcard

US / font manager:
(04090001) font table is full
(04090002) bad font file
(04090003) unrecognized font version #
(04090004) bad font id argument
(04090005) bad value id
(04090006) character is not in font
(04090007) free storage space full
(04090008) character is too large
(04090009) descriptor or header value is out of range
(0409000A) invalid character code
(0409000B) no character image
(0409000C) font is read-only
(0409000D) one or more images are required
(0409000E) font table is full

US / shell:
(040A0001) shell program too large: syntax tree space full
(040A0002) shell program too large: token buffer space full
(040A0003) shell programs nest too deeply (may invoke itself)

US / aclm:
(040B0001) no right to perform operation
(040B0002) insufficient rights to perform operation
(040B0003) exit_super called more often than enter_super
(040B0004) wrong type - operation illegal on system objects

(040B0005) entry already exists
(040B0006) ACL is remote
(040B0007) ACL is on different volume than object
(040B0008) ACL protects wrong type of object
(040B0009) insufficient address space to open ACL
(040B000A) entry was matched by a wild card
(040B000B) no entry - entry number too large
(040B000C) image buffer too small or incorrect size
(040B000D) ACL object not found
(040B000E) ACL would be unchangeable
(040B000F) object may not be readable by backup procedure
(040B0010) reserved code 10
(040B0011) reserved code 11
(040B0012) reserved code 12
(040B0013) reserved code 13
(040B0014) invalid right name
(040B0015) invalid person identifier
(040B0016) invalid project identifier
(040B0017) invalid organization ID
(040B0018) invalid node ID
(040B0019) non-existent SID

US / uvtoc:
(040C0001) object is not a local entry directory
(040C0002) no available address space
(040C0003) end of vtoc

library / open:
(05010001) do not provide name for SCRATCH file
(05010002) need name of file to open
(05010003) invalid STATUS= option
(05010004) direct access not allowed with variable length records
(05010005) given record length is not compatible with this file
(05010006) need record length > zero
(05010007) invalid ACCESS= option
(05010008) invalid FORM= option
(05010009) invalid UNIT= specifier
(0501000A) no space to create requested IO unit
(0501000B) FORM= option value not compatible with this file
(0501000C) BLANK= option not allowed for unformatted file
(0501000D) invalid BLANK= option
(0501000E) unrecognized special name beginning with "-"
(0501000F) command line arguments are named ^1 thru ^9
(05010010) file not found

library / close:
(05020001) KEEP status not allowed for SCRATCH file
(05020002) invalid STATUS= option
(05020003) can't delete file

```
library / mark/release:
(05030001)  cannot find old Fortran IO state record

library / IO:
(05040001)  no space to create IO unit
(05040002)  unit is not connected
(05040003)  recursive use of FORTRAN IO
(05040004)  direct access not allowed to sequential file
(05040005)  need record number for direct access
(05040006)  invalid option(s) used to reference internal file
(05040007)  feature not implemented yet
(05040008)  end of file
(05040009)  unformatted operation not allowed on file
             opened for formatted IO
(0504000A)  formatted operation not allowed on file
             opened for unformatted IO

library / IO transfer:
(05050001)  Internal format error
(05050002)  No edit specifier in format
(05050003)  Improper character in input data
(05050004)  Real exponent too large
(05050005)  Improper format descriptor for variable type
(05050006)  Data overflows buffer size

library / read:
(05060001)  end of file
(05060002)  record is larger than available buffer

library / write:
(05070001)  nyi

library / position:
(05080001)  no space to create unit
(05080002)  unit is not connected
(05080003)  recursive IO

library / floating point:
(05090001)  overflow in add/sub
(05090002)  underflow in add/sub
(05090003)  loss of significance in add/sub
(05090004)  overflow in multiply
(05090005)  underflow in multiply
(05090006)  overflow in divide
(05090007)  underflow in divide
(05090008)  division by zero
(05090009)  overflow in convert to word integer
(0509000A)  overflow in convert to long integer
(0509000B)  overflow in dp add/sub
(0509000C)  underflow in dp add/sub
(0509000D)  loss of significance in dp add/sub
(0509000E)  overflow in dp multiply
(0509000F)  underflow in dp multiply
(05090010)  overflow in dp divide
(05090011)  underflow in dp divide
(05090012)  dp division by zero
```

(05090013) overflow in convert double to single
(05090014) underflow in convert double to single
(05090015) square root of a negative number
(05090016) overflow in exp function
(05090017) overflow in dp exp function
(05090018) arg of log function less than or equal to zero
(05090019) dp arg of log function less than or equal to zero

library / inquire:
(050A0001) cmd line args are named ^1 thru ^9
(050A0002) recursive io
(050A0003) target string too small

library / format:
(050B0001) left parenthesis expected
(050B0002) right parenthesis expected
(050B0003) period expected
(050B0004) comma, colon, slash, or right parenthesis expected
(050B0005) unsigned integer constant expected
(050B0006) p format not followed by f, e, d, or g
(050B0007) bad hollerith or character constant
(050B0008) unrecognized format specifier

library / tfp:
(050CFFFF) computed block count doesn't match that in EOF label
(050C0001) volume not found
(050C0002) volume not open
(050C0003) volume already open
(050C0004) file not open
(050C0005) file already open
(050C0006) file not found
(050C0007) tape i/o error
(050C0008) end of volume encountered
(050C0009) invalid file section value
(050C000A) invalid file sequence value
(050C000B) invalid generation number value
(050C000C) invalid generation version number value
(050C000D) invalid block count value
(050C000E) invalid block length value
(050C000F) invalid record length value
(050C0010) invalid buffer offset value
(050C0011) no tcb is available (internal table full)
(050C0012) invalid unit number
(050C0013) first label on volume is not VOL1 label
(050C0014) label version number in VOL1 label is not "3"
(050C0015) tape drive is already in use by this process
(050C0016) end of file
(050C0017) tape limit: eot/bot marker encountered
(050C0018) label size is in error; not 80 characters
(050C0019) block size is too large
(050C001A) invalid record format specifier

(050C001B) unlabeled operation attempted on a labeled volume
(050C001C) labeled operation attempted on an unlabeled volume
(050C001D) invalid volume id
(050C001E) conflicting blocking information
(050C001F) file does not exist on this volume
(050C0020) internal or exception procedure returned bad new-volume info
(050C0021) a HDR1 label is missing where one is required
(050C0022) an EOF1 (or EOV1) label is missing where one is required
(050C0023) a double filemark was encountered unexpectedly
(050C0024) inconsistent file sequence numbers
(050C0025) file sequence number tracking error
(050C0026) read attempted on file open for writing
(050C0027) write attempted on file open for reading
(050C0028) block list size is invalid
(050C0029) an EOV1 (or EOF1) label is missing where one is required
(050C002A) wrong volume, file header is inconsistent with previous trailer
(050C002B) operation not started due to error in previous block list
(050C002C) record i/o attempted on volume open for block i/o only
(050C002D) variable record with invalid record control word encountered
(050C002E) spanned record with invalid segment control word encountered
(050C002F) erroneous spanning indicator or segment out of sequence encountered
(050C0030) invalid mode set passed to tfp_$set_mode
(050C0031) invalid environment for mt_status required
(050C0032) more info returned from mt_$wait that expected
(050C0033) attempt to enqueue too many operations
(050C0034) magtape is offline or floppy is not ready
(050C0035) wait not performed before go
(050C0036) go called with no work to do
(050C0037) invalid operation code
(050C0038) OS magtape manager out of pb's
(050C0039) wait called when not required
(050C003A) OS magtape manager didn't purge ops on error
(050C003B) # ops executed does not match # ops completed
(050C003C) i/o error recovery failed
(050C003D) invalid recovery value returned by execption handler
(050C003E) zero length block list unexpected
(050C003F) tape rewind error
(050C0040) tape space-record error
(050C0041) tape space-filemark error
(050C0042) tape write-filemark error

(050C0043) tape or floppy is write-protected
(050C0044) redundant call to wire i/o buffers
(050C0045) attempt to already-unwired i/o buffers
(050C0046) unable to re-open volume; current position is unknown
(050C0047) unable to re-open volume; info file is bad
(050C0048) tape read-foreign error

library / Pascal IO:
(050D0001) too many open files
(050D0002) use of unopened file
(050D0003) REWRITE required before writing to file
(050D0004) RESET required before reading from file
(050D0005) read past end of file
(050D0006) real exponent too large
(050D0007) invalid operation on file
(050D0008) invalid OPEN parameter
(050D0009) inconsistent file usage
(050D000A) invalid read data

library / MBX manager:
(050E0001) size parameter too large
(050E0002) size parameter too small
(050E0003) too many channels
(050E0004) illegal mbx handle
(050E0005) file already in use
(050E0006) no room in channel
(050E0007) msg too big for channel
(050E0008) no active servers
(050E0009) open rejected
(050E000A) end of file
(050E000B) no available channels
(050E000C) channel not open
(050E000D) unexpected control message received
(050E000E) remote transmit failed
(050E000F) remote reply wait timed out
(050E0010) remote service currently unavailable
(050E0011) invalid mbx handle
(050E0012) wrong mbx version number
(050E0013) unknown remote request
(050E0014) remote service denied
(050E0015) channel empty
(050E0016) supplied buffer too small
(050E0017) record was truncated
(050E0018) no more resources available
(050E0019) bad key
(050E001A) returned data does not complete a record

library / ptx plot package:
(050F0001) image too small in the x-dimension
(050F0002) image too small in the y-dimension
(050F0003) invalid value for horizontal start

library / floppy seq i/o manager:
(0510FFFE) warning: record truncated
(0510FFFF) warning: disk almost full

(05100001) invalid floppy unit #
(05100002) floppy is full
(05100003) control block is too large
(05100004) end of file
(05100005) end of tape
(05100006) unit is already open
(05100007) unit is not open
(05100008) attempted write on read-only floppy
(05100009) invalid marker specification
(0510000A) data error: filler expected where none was found
(0510000B) data error: filler found where not expected
(0510000C) data error: invalid control code
(0510000D) data error: record trailer missing
(0510000E) data error: record trailer doesn't match header
(0510000F) beginning of tape
(05100010) data error: beginning of tape found where unexpected
(05100011) floppy is physically write-protected
(05100012) floppy is not ready
(05100013) invalid label record -- probably not written with WBAK

library / Ethernet driver:
(05110001) controller does not exist
(05110002) invalid ethernet controller number
(05110003) controller not online
(05110004) invalid buffer size parameter
(05110005) invalid statistics type
(05110006) invalid mode parameter
(05110007) invalid options parameter
(05110008) illegal number of addresses
(05110009) address is not a valid group address
(0511000A) no receive available
(0511000B) quit while waiting
(0511000C) receive packet truncated
(0511000D) too many collisions
(0511000E) device timed out
(0511000F) operation failed
(05110010) internal frame header invalid
(05110011) internal dma length error
(05110012) unexpected status from device
(05110013) NM10 scratchpad test failed
(05110014) NM10 dma test failed
(05110015) NM10 transmitter test failed
(05110016) NM10 receiver test failed
(05110017) NM10 loopback test failed

library / versatec driver:
(05120001) invalid action code
(05120002) invalid buffer size
(05120003) no PBU or Versatec controller is installed
(05120004) device is offline
(05120005) device is out of paper

library / pipe manager:
(05130001) supplied buffer too large
(05130002) pipe is empty
(05130003) pipe is full
(05130004) illegal pipe operation
(05130005) pipe has no readers
(05130006) pipe has no writers

library / amlc manager:
(05140001) controller does not exist
(05140002) bad unit number

library / streams-magtape manager:
(05150001) object is not a magtape file descriptor
(05150002) magtape file descriptor contains invalid data
(05150003) invalid attribute specifier
(05150004) invalid creation/expiration date
(05150005) invalid record format
(05150006) descriptor file is read-only
(05150007) invalid unit number
(05150008) invalid file sequence number
(05150009) invalid file section number
(0515000A) invalid block length
(0515000B) invalid record length
(0515000C) invalid generation number
(0515000D) invalid generation version number
(0515000E) invalid buffer offset

graphics / primitives:
(06010001) Primitives not initialized
(06010002) Primitives already initialized
(06010003) Wrong display hardware
(06010004) Operation illegal for DM frame
(06010005) Must borrow display for this operation
(06010006) No attributes defined for bitmap
(06010007) No more bitmap space available
(06010008) Dimension too big
(06010009) Dimension too small
(0601000A) Bad bitmap descriptor
(0601000B) Bad attribute block descriptor
(0601000C) Window origin out of bitmap bounds
(0601000D) Source window origin out of bitmap bounds
(0601000E) Destination window origin out of bitmap bounds
(0601000F) Invalid plane number
(06010010) Cannot deallocate this bitmap
(06010011) Coordinate value out of bounds
(06010012) Invalid color map
(06010013) Invalid raster operation value
(06010014) Bitmap is read-only
(06010015) Internal error
(06010016) Font table is full
(06010017) Bad font file
(06010018) Invalid font id

graphics / DM support:
(06020001) Invalid bitmap address
(06020002) Invalid bitmap length

graphics / core graphics:
(06030001) 3D inquiry function needed
(06030002) Count parameter or array size is less than 1
(06030003) Array size is less than 3
(06030004) There is no selected view surface
(06030005) Inconsistant viewing specification
(06030006) A segment is currently open
(06030007) There is no open segment
(06030008) Color attribute not supported by view surface
(06030009) Intensity attribute not supported by view surface
(0603000A) Requested linestyle attribute not supported
(0603000B) Requested linewidth attribute not supported
(0603000C) Requested pen attribute not supported
(0603000D) Requested font not supported
(0603000E) String contains undefined characters
(0603000F) Charplane and charup vectors are parallel
(06030010) Requested marker symbol not supported
(06030011) There already is an open segment
(06030012) The requested retained segment name already exists
(06030013) Current image transformation is of wrong type
(06030014) There is no open retained segment
(06030015) There is no retained segment matching given name
(06030016) There already exists a segment with given name
(06030017) There is no open temporary segment
(06030018) One or more of the attribute values is invalid
(06030019) All charplane normal components are zero
(0603001A) All charup components are zero
(0603001B) Image transformation type is invalid
(0603001C) Image transformation higher type than requested
(0603001D) Segment image xform type not compatible with inq
(0603001E) Segment image xform type higher than inquiry
(0603001F) Maximum less than minimun value (2D)
(06030020) View up vector components both equal zero
(06030021) Program has already set NDC space
(06030022) NDC default space has already been established
(06030023) A parameter is not in NDC system : 0->1
(06030024) Neither width nor height is equal to 1
(06030025) Width or height is equal to zero
(06030026) Viewport corner outside of NDC space
(06030027) A 3D viewing function was needed, not 2D
(06030028) Specified NDC position is outside viewport
(06030029) The world coordinate xform is not invertible

(0603002A) World coordinate position outside clipping window
(0603002B) All 3 components of view plane normal are zero
(0603002C) Front distance is greater than back distance
(0603002D) Parallel projection and direction components = 0
(0603002E) Window maximum less than window minimum
(0603002F) All 3 viewup normal vector components are = 0
(06030030) Maximum less than minimun value (3D)
(06030031) Projection of world coord. position outside window
(06030032) World coordinate position in front of clip plane
(06030033) World coordinate position in back of clip plane
(06030034) World coord. position behind center of projection
(06030035) Coordinate system type has already been set
(06030036) Too late - viewing established or segment exists
(06030037) View plane normal and viewup vector parallel
(06030038) Front plane not between cop and back plane
(06030039) Direction of projection aparallel to view plane
(0603003A) Center of projection on or behind view plane
(0603003B) Front and back clipping planes coincident
(0603003C) Device already initialized
(0603003D) Device class or device number is invalid
(0603003E) Device class is invalid
(0603003F) Array contains invalid device number
(06030040) Device is not initialized
(06030041) Device is already enabled
(06030042) Member of group not initialized
(06030043) Device is not enabled
(06030044) Locator device not enabled
(06030045) Valuator device not enabled
(06030046) Time parameter is less than zero
(06030047) Event device not initialized
(06030048) Invalid event device class
(06030049) Association already exists
(0603004A) Event class is invalid
(0603004B) One or both devices not initialized
(0603004C) Association does not exist
(0603004D) Event report not from pick device
(0603004E) Event report not from keyboard device
(0603004F) Event report not from stroke device
(06030050) Needed 3D input function
(06030051) Locator not enabled or associated
(06030052) Valuator not enabled or associated
(06030053) No button devices are initialized
(06030054) Specified pick device not initialized
(06030055) Specified keyboard device not initialized
(06030056) Specified stroke device not initialized
(06030057) Specified locator device not initialized
(06030058) Specified valuator device not initialized

(06030059) Echo type is invalid for specified class
(0603005A) Specified device does not accept echo segment
(0603005B) Image xform of segment incompatible with echo type
(0603005C) Echo position outside of NDC space
(0603005D) Pick aperature is less than or equal to zero
(0603005E) Buffer size out of bounds
(0603005F) Cursor start position out of bounds
(06030060) Initial string contains undefined characters
(06030061) Specified button device is not initialized
(06030062) Distance or time less than or equal to zero
(06030063) Specified locator position outside of NDC space
(06030064) Maximum less than minimun (3D)
(06030065) Locport corner outside of NDC space
(06030066) Low value is greater than high value
(06030067) Initial value not within defined range
(06030068) Associated size less than or equal to zero
(06030069) Duplication size less than or equal to zero
(0603006A) Core has already been initialized
(0603006B) Specified output level cannot be supported
(0603006C) Specified input level cannot be supported
(0603006D) Specified dimension cannot be supported
(0603006E) View surface is already initialized
(0603006F) No output device associated with view surface
(06030070) No other view surface can be initialized
(06030071) View surface is not initialized
(06030072) View surface is already selected
(06030073) View surface cannot be selected
(06030074) View surface has not been selected
(06030075) Immediate visibility state is invalid
(06030076) Currently in batch of updates
(06030077) Not in batch of updates
(06030078) Element of segment array does not exist
(06030079) Invalid visibility array element
(0603007A) Core has not been initialized
(0603007B) Function at a higher level than current suppory
(0603007C) Invalid error log invocation
(0603007D) Function is not supported
(0603007E) Parameter count is incorrect
(0603007F) One or more parameters invalid
(06030080) Vertex array is too small
(06030081) Intensity value is invalid
(06030082) Color value is invalid
(06030083) Specified table range is invalid
(06030084) Low or high table index is invalid
(06030085) Low table index is greater than high
(06030086) Too few values to fill specified range
(06030087) Invalid intensity array element
(06030088) Invalid color array element
(06030089) Origin outside NDC space
(0603008A) Array size invalid
(0603008B) Array row invalid

(0603008C) Array column invalid
(0603008D) Invalid parameter value
(0603008E) View surface is initialized to intensity
(0603008F) Invalid index value
(06030090) Shading vertex mismatch
(06030091) Vertices not coplanar
(06030092) Less than three vertices received
(06030093) Index range is too large
(06030094) Table is too small
(06030095) View surface is of type color
(06030096) View surface is of type intensity
(06030097) There is no hidden surface support
(06030098) The specified mode is invalid
(06030099) View surface is initialized
(0603009A) Invalid view surface type parameter
(0603009B) Input currently not supported in frame mode

graphics / metafile manager:
(06040001) Bits/inch parameter is negative
(06040002) X dimension is not positive
(06040003) Y dimension is not positive
(06040004) Words/line parameter is too small
(06040005) Position parameter is illegal

registry manager / PPO supervisor:
(07010001) Name not defined
(07010002) Name already defined
(07010003) Name is too long
(07010004) Incompatible program and file versions
(07010005) No more memory space available
(07010006) File open for read, not write
(07010007) Invalid file, internal error
(07010008) PPO salvage only partially complete
(07010009) No valid PPO file found at any registry site
(0701000A) Name has invalid characters
(0701000B) Cannot delete reserved name
(0701000C) Unable to access ppo file at any registry site

registry manager / Accounts supervisor:
(07020001) Account not defined
(07020002) Last entry cannot be deleted
(07020003) Invalid operation on local registry
(07020004) Incompatible program and file versions
(07020005) No more memory space available
(07020006) File open for read, not write
(07020007) Invalid file, internal error
(07020008) Account salvage only partially complete
(07020009) No valid account file found at any registry site
(0702000A) Unable to access account file at any registry site

registry manager / Registry supervisor:
(07030001) Invalid registry file, internal error
(07030002) Registry already open for that file type

(07030003) Registry is open for read. Cannot open for write
(07030004) Index exceeds number of sites in registry
(07030005) Attempt to open two registries simultaneously
(07030006) Name is not a site in this registry
(07030007) Site pathname too long, cannot append filename
(07030008) Attempt to delete last site of registry
(07030009) Invalid pathname
(0703000A) Full site pathname required for existing sites.

registry manager / Login supervisor:
(07040001) Shutting down
(07040002) Exiting

D3M data management / d3mlib:
(08010001) database not open
(08010002) dbk for wrong area
(08010003) database already open
(08010004) unused
(08010005) duplicates error
(08010006) current unknown
(08010007) end of set or area
(08010008) unknown name; not in schema
(08010009) database is open read only
(0801000A) wrong password
(0801000B) no disk space
(0801000C) dbk not available/no good
(0801000D) unused
(0801000E) no current of this record type
(0801000F) set has no optional members
(08010010) record is already member
(08010011) violation of concatenated set integrity
(08010012) unused
(08010013) unused
(08010014) current of rec wrong type
(08010015) current of rununit of wrong type
(08010016) record not member in set
(08010017) record not in area
(08010018) unused
(08010019) unused
(0801001A) no record satisfies record selection
(0801001B) unused
(0801001C) area already open
(0801001D) area not open
(0801001E) delete set not empty
(0801001F) internal stack overflow
(08010020) invalid literal argument
(08010021) invalid virtual item
(08010022) unused
(08010023) mod var lgth needs 2 bufs
(08010024) unused
(08010025) unused
(08010026) unused

(08010027) unused
(08010028) findv requires sorted set
(08010029) findc requires calced record
(0801002A) record not indexed
(0801002B) unused
(0801002C) improper set mode
(0801002D) set, record, etc. not in subschema
(0801002E) unused
(0801002F) unused
(08010030) unused
(08010031) unused
(08010032) unused
(08010033) deleted rec referenced
(08010034) key relation mismatch
(08010035) HLDML call error
(08010036) HLDML syntax error
(08010037) HLDML semantic error
(08010038) HLDML execution error from D3MLIB
(08010039) unexpected HLDML execution error
(0801003A) HLDML internal error
(0801003B) unused
(0801003C) unused
(0801003D) close error
(0801003E) name not found
(0801003F) file not found
(08010040) required resources not available
(08010041) end of file
(08010042) wrong object type
(08010043) unable to open
(08010044) bucket initialization error
(08010045) unused
(08010046) loc mode select required in all sets
(08010047) too many sets in path
(08010048) multischema inconsistency
(08010049) distributed DML operation not supported
(0801004A) dynamic store error
(0801004B) violation of distributed set integrity
(0801004C) unused
(0801004D) unused
(0801004E) unused
(0801004F) unused
(08010050) internal bad bucket number
(08010051) transaction call err
(08010052) active transaction required
(08010053) unused
(08010054) unused
(08010055) unused
(08010056) unused
(08010057) unused
(08010058) UWA is too large; insufficient memory
(08010059) schema/subschema confusion
(0801005A) recompile schema
(0801005B) read error on (sub)schema file
(0801005C) (sub)schema file open err
(0801005D) database write error
(0801005E) database read error

(0801005F) concurrency violation
(08010060) device unavailable
(08010061) insufficient virtual memory
(08010062) must preallocate file (use INDB)
(08010063) bad bucket adress

AUX / signal:
(09010001) hangup fault
(09010002) interrupt fault
(09010003) quit fault
(09010004) illegal instruction fault
(09010005) trace trap fault
(09010006) IOT instruction fault
(09010007) EMT instruction fault
(09010008) floating point exception fault
(09010009) process kill fault
(0901000A) bus error fault
(0901000B) segmentation violation fault
(0901000C) bad argument to system call fault
(0901000D) broken pipe fault
(0901000E) alarm clock fault
(0901000F) software termination fault
(09010010) user-defined fault (1)
(09010011) user-defined fault (2)
(09010012) child death fault
(09010013) power failure fault

BOOT ERRORS (PROM)

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

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

SC = Status Code
RCD = Record Address
Unit = Disk Unit No.
W/F = Winchester/Floppy

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

DIAGNOSTIC ERROR CODES

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

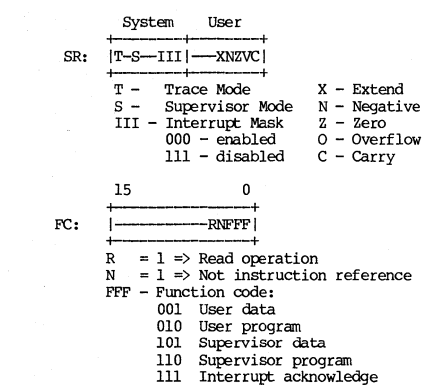
test 1 = checksum
" 2 = PFT
" 3 = PTT
" 4 = IOMAP
" 5 = PFT/PTT/IOMAP interactions
" 6 = First 256KB (physical)
" 7 = " " (mapped)
" 8 = " " verify mapping

MNEMONIC DEBUGGER ERROR CODES (PROM)

Printed on system crash or other entry to MD. See also System Crash Analysis under Operational Procedures.

```
A  <PC> <SR> <IR> <FA> <FC>   -  Address Error
   <PC> <Contents>
B  <PC> <SR> <IR> <FA> <FC>   -  Bus Error
   <PC> <Contents>
E                             -  Operational Error
I  <PC> <SR>                  -  Unexpected Interrupt
   <PC> <Contents>
J                             -  Spurious Interrupt
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>
Z  <PC> <SR>                  -  Divide by zero trap
   <PC> <Contents>
```

PC = Program Counter
SR = Status Register
IR = Instruction Register
FA = Fault Address
FC = Fault Code



```
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
```

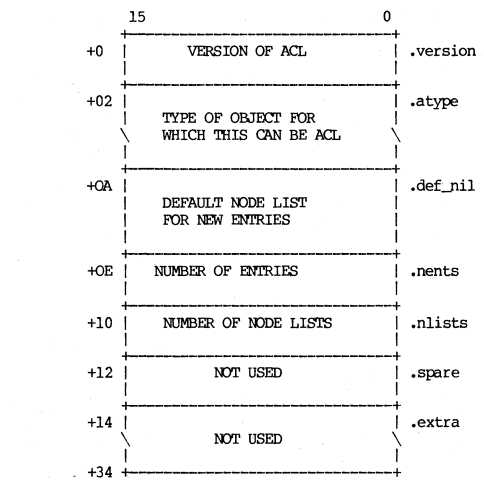
CHAPTER 5

FILE SYSTEM

ACLS STRUCTURE

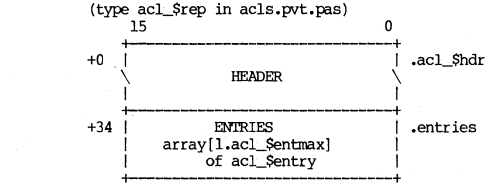
ACL Header Record

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



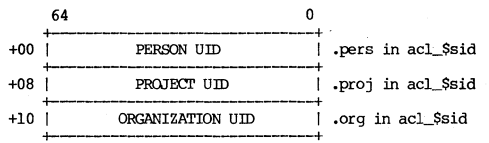
ACL Record

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



ACL Entry

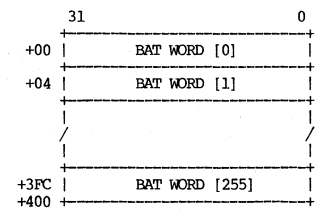
(type acl_$entry in acl.ins.pas)



BLOCK AVAILABILITY TABLE (BAT)

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

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

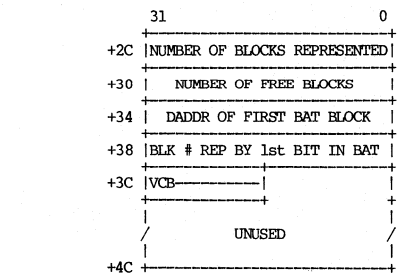


First BAT block pointed to by BAT header in logical volume label (pv_label). BAT resides in contiguous records.

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

BLOCK AVAILABILITY TABLE HEADER

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



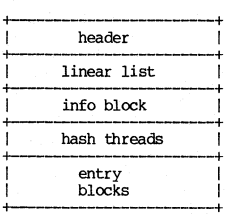
V - Volume trouble, set by OS if volume needs salvaging, cleared by SALVOL.
C - Volume chuvoled
B - Volume being chuvoled

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

DIRECTORY STRUCTURE

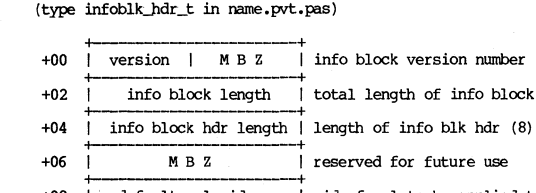
Directory Overview

(type dir_t in name.pvt.pas)



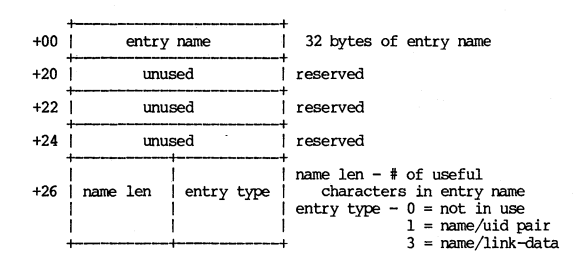
Directory Info Block

(type infoblk_hdr_t in name.pvt.pas)



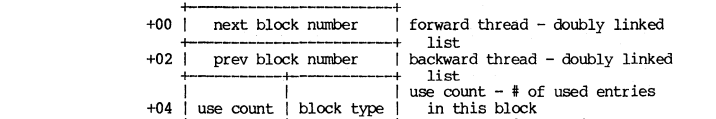
Directory Entry

(type dir_entry_t in name.pvt.pas)



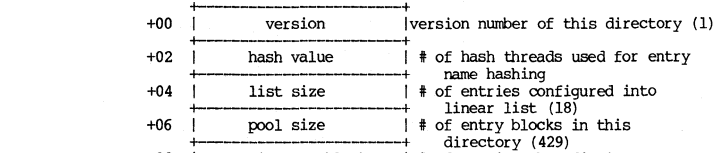
Directory Entry Block

(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
- lnth is useful lnth of name
   sum: =0;
   For i : = 1 to lnth 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 apprpriate 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 NETX BLOCK field and goto 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.

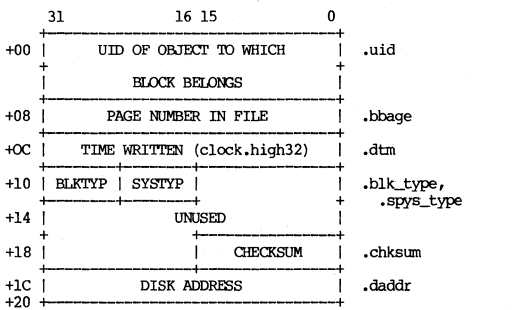
   D. Search all (3) of the entries in the found entry block

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. Goto step (d) with the newly found block.

DISK BLOCK HEADER

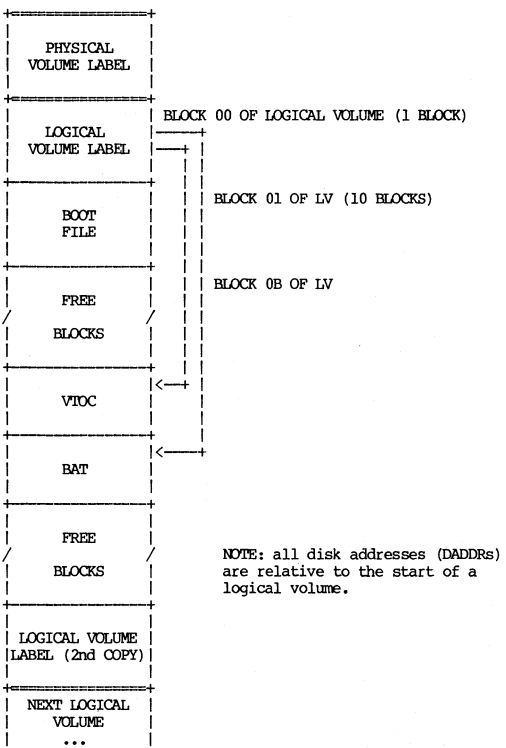
(type "blk_hdr_t" in base.spo.bbas)



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

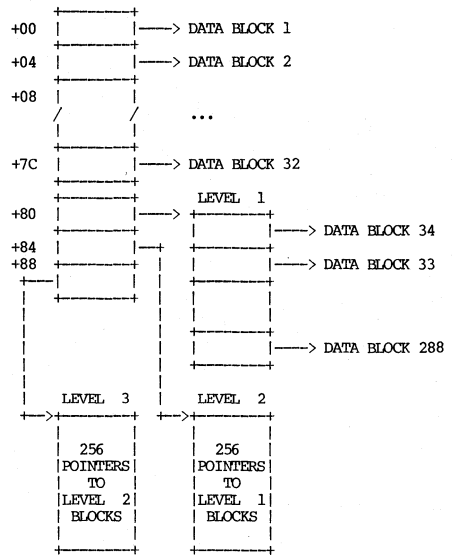
SYSTYP: 0 - File
1 - Directory
2 - System directory

DISK/VOLUME FORMAT



Badspots may cause the VTOC to be incontiguous and not adjacent to BAT. There may exist dead space between logical volumes.

FILE MAP



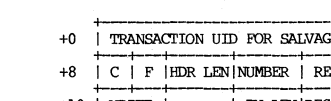
Maximum file size = (32+256+256**2+256**3)*1024 bytes
= 17,247,300,000 bytes.

File map resides in VTOC and AST entry.

REGISTRY FORMAT

Header Record

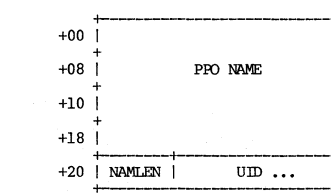
(type ppo_$header_t in ppo.ins.pas)



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)

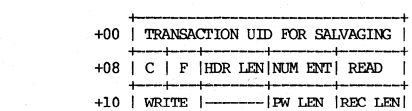
PPO Record

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



Account Header

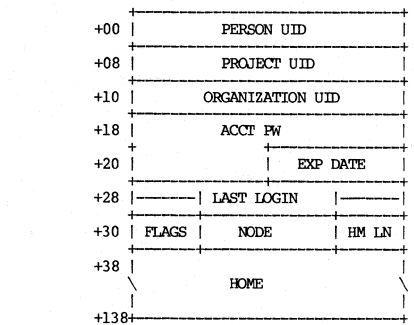
(type acct_$header in acct.ins.pas)



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

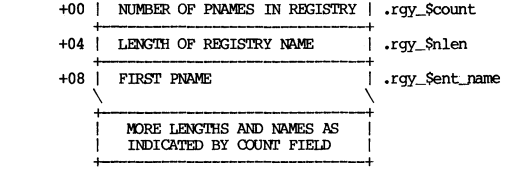
(type acct_$record_t in acct.ins.pas)



* 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

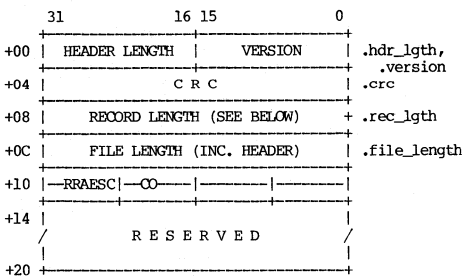
(type rgy_$registry_t in rgy.ins.pas)



First pname is path of original registry, used as lock

STREAM FILE HEADER

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



CRC = -(integer sum of (1 + 3 thru 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)

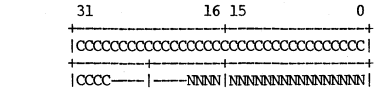
CO - Concurrency (.conc, stream_$fconc_t):
00 - N readers or 1 writer (stream_$n_or_1)
01 - N readers and 1 writer (stream_$n_and_1)
10 - N readers and N writers (steam_$n_and_n)

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)

("uid_t" in base.ins.pas)



C..C - Top 36-bits of clock (16 msec units)

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

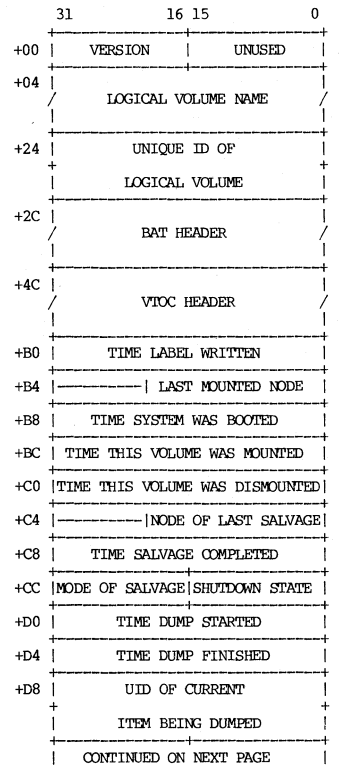
(from /os/nuc/uid_list.asm)

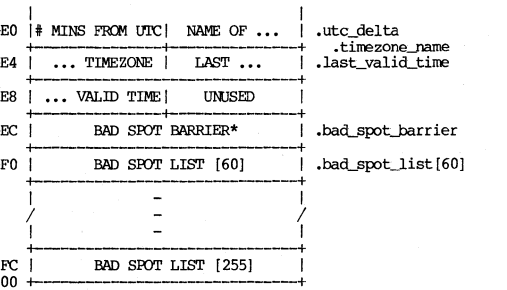
```
uid_$nil                00000000,0
acl_$nil                00000100,0

canned ACL uids (file acls) (0001.xxxx series)

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

canned ACL uids (directory ACLs) (0002.xxxx series)

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

disk structure canned UIDs (0000.02xx series)

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

canned object type UIDs (0000.03xx series)

records_$uid            00000300,0
hdr_undef_$uid          00000301,0
object_file_$uid        00000302,0
UNDEF_$uid              00000304,0
pad_$uid                00000305,0
input_pad_$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 (0000.04xx series)

display1_$uid           00000400,0
display2_$uid           00000401,0
name_$canned_root_uid   00000308,0

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

```
canned persons (0000.050x series)

acl_$sys_user_uid       00000500,0

canned projects (0000.054x series)

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

canned organizations (0000.058x series)

acl_$sys_org_uid        00000580,0

canned subsystems (0000.05Cx series)

acl_$nil_subs_uid       000005C0,0

canned acl type UIDs (0000.06xx series)

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

VOLUME LABEL — LOGICAL

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

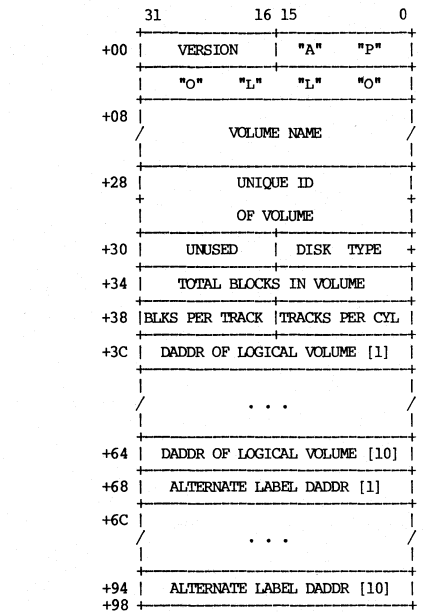




bad_spot_list allocated from end of list.
* FFFFFFFF -> no continue
  FFFFFFFE -> +F0 is DADDR of continue block

VOLUME LABEL — PHYSICAL

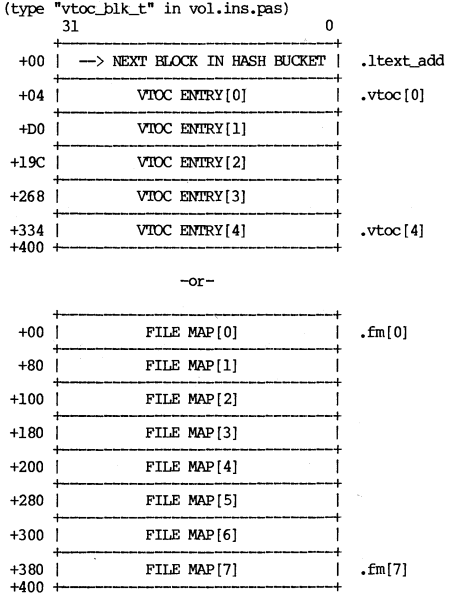
(type "pv_label_t" in vol.ins.pas)



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

VTOC BLOCK

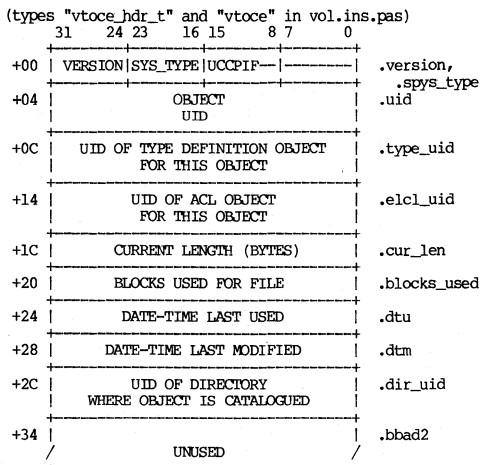
(type "vtoc_blk_t" in vol.ins.pas)



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

VTOC ENTRY

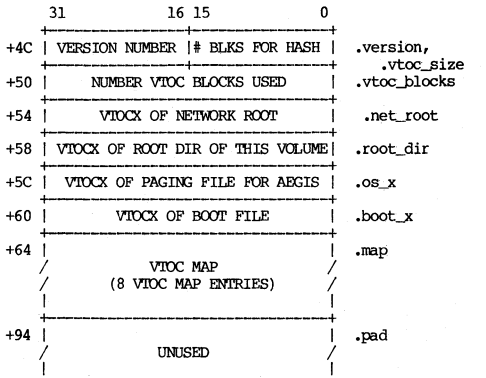
(types "vtoce_hdr_t" and "vtoce" in vol.ins.pas)



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

VTOC HEADER

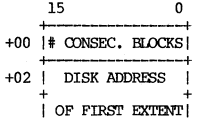
(type "vtoc_hdr_t" in vol.ins.pas)



VTOC header lines in logical volume label. Offsets given are from start of label.

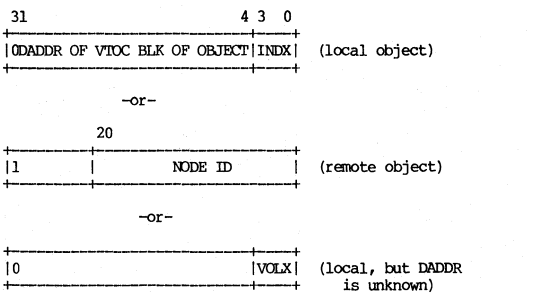
VTOC MAP ENTRY

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



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 6

PERIPHERAL I/O

DEVICE ADDRESSES (PIO)

DEVICE	VIRTUAL DN4xx and	PHYSICAL DN600	VIRTUAL DN300	PHYSICAL
Multibus (64 pages)	FE0000	10000		
Line printer ctl	FEFC00	1FC00		
Multibus int ctl	FEFC80	1FC80		
MCSRs	FF9000	FC00	FFB404	8004
Display 2 control	FF9400	F000		
Display 1 control	FF9800	F400	FF9800	9400
Color control store	FF	E800		
Color user page		E400		
Color super page		E000		
PEB	FF7000	B000		
Ring 2	FF9C00	BC00	FF9C00	9800
Ring 1	FFA000	B800		
Floppy controller	FFA800	8C00	FFA800	9C00
Timers	FFAC00	8800		
SIO lines	FFB000	8400	FFB000	8400
MMU	FFB400	8000	FFB400	8000
Page frame table	FFB800	4000		

DISK PARAMETERS

	CYLS	HEADS	BLK/TRK	TOT BLKS	
33MB	561	3	18	30294	(7656)
66MB	1121	3	18	60534	(EC76)
154MB	1121	7	18	141246	(227BE)
300MB	823	19	18	281466	(44B7A)
8" PRIAM 3450	525	5	12	31500	(7B0C)
8" PRIAM 7050	1049	5	12	62940	(F5DC)
8" MICROPOLIS	580	5	13	37700	(9344)
FLOPPY	77	2	8	1232	(4D0)

	SEC START	SEC SIZE	SEC DELTA	ID
33MB	34	1118	3	1
66MB	34	1118	3	6
154MB	34	1118	3	7
300MB	34(?)	1118(?)	9	—
8" PRIAM 3450	0	1120	3(?)	104
8" PRIAM 7050	0	1120	3(?)	5
8" MICROPOLIS	0	1181	3(?)	1203
FLOPPY	146	1202	2	—

	T-to-T SEEK	AVER. SEEK	MAX SEEK
33MB	8 msec	45 msec	85 msec
66MB	8 msec	45 msec	85 msec
154MB	8 msec	45 msec	75 msec
300MB			
8" PRIAM 3450	8 msec	42 msec	75 msec
8" PRIAM 7050	8 msec	42 msec	75 msec
8" MICROPOLIS	12 msec	42 msec	85 msec
FLOPPY			

	RPM	AVER. LAT	TRANSFER RATE
33MB	3100	9.7 msec	1.04 MBS
66MB	3100	9.7 msec	1.04 MBS
154MB	3100	9.7 msec	1.04 MBS
300MB			
8" PRIAM 3450	3600	8.3 msec	0.8 MBS
8" MICROPOLIS	3600	8.3 msec	0.92 MBS
FLOPPY			

	AVERAGE READ
33MB	45+9.7+1.0 = 55.7
66MB	45+9.7+1.0 = 55.7
154MB	45+9.7+1.0 = 55.7
300MB	
8" PRIAM 3450	42+8.3+1.3 = 51.6
8" MICROPOLIS	42+8,3+1.1 = 51.4
FLOPPY	

	BADSPOT DADDR	DIAG DADDR
33MB	7620 (cyl 560)	75AE (cyl 559)
66MB	EC40 (cyl 1120)	EC0A (cyl 1119)
154MB	22740 (cyl 1120)	226C2 (cyl 1119)
300MB	448CE (cyl 821)	44A24 (cyl 822)
8" PRIAM 3450	7AD0 (cyl 524)	7A94 (cyl 523)
8" PRIAM 7050	F5A0 (cyl 1048)	F564 (cyl 1047)
8" MICROPOLIS	9303 (cyl 579)	92C2 (cyl 578)
FLOPPY	4C0 (cyl 76)	-----

DMA CONTROLLER (DN300 ONLY)

DMAC page at [9000 | 0FFA000]

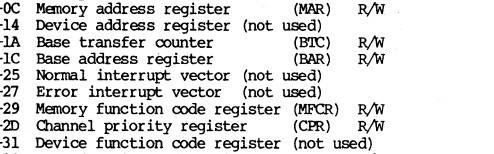
DMA controller is a Motorola M68450.

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

Register summary (for each channel):

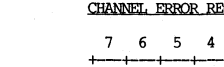
+00	Channel status register	(CSR)	R/W
+01	Channel error register	(CER)	R
+04	Device control register	(DCR)	R/W
+05	Operation control register	(OCR)	R/W
+06	Sequence control register	(SCR)	R/W
+07	Channel control register	(CCR)	R/W
+0A	Memory transfer counter	(MTC)	R/W
+0C	Memory address register	(MAR)	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]



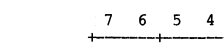
(*) Bit cleared by writing a 1 bit to CSR.
(**) Ditto, and clearing also clears CER.

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



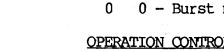
00 - No error
01 - Configuration error
02 - Operation timing error
03 - (undefined, reserved)
05 - Address error: memory address or memory counter
06 - Address error: device address
07 - Address error: base address or base counter
09 - Bus error: memory address or memory counter
0A - Bus error: device 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]



0 0 - PCL = Status input
1 - 16-bit port
1 0 - Device with ACK, implicitly addressed
0 0 - Burst mode transfers

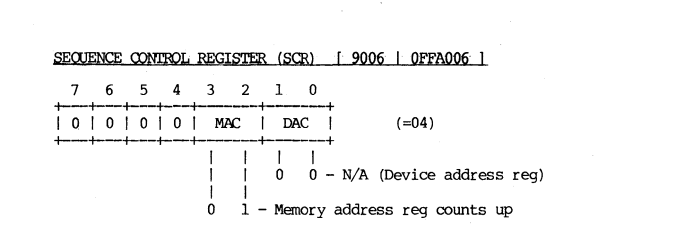
OPERATION CONTROL REGISTER (OCR) [9005 | 0FFA005]



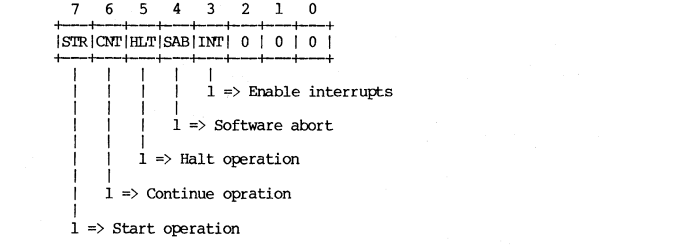
1 0 - REQ line initiates xfer
0 0 - Chain operation disabled
0 1 - Word transfers

0 - Transfer from memory to device
1 - Transfer from device to memory

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



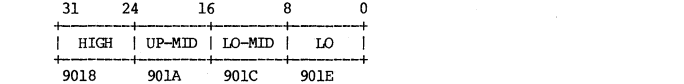
CHANNEL CONTROL REGISTER (CCR) [9007 | 0FFA007]



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.

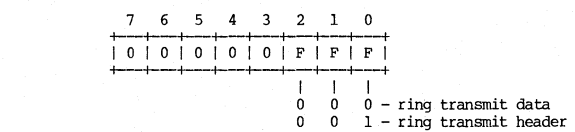
NORMAL INTERRUPT VECTOR REGISTER [9025 | 0FFA025]

Not used.

ERROR INTERRUPT VECTOR REGISTER [9027 | 0FFA027]

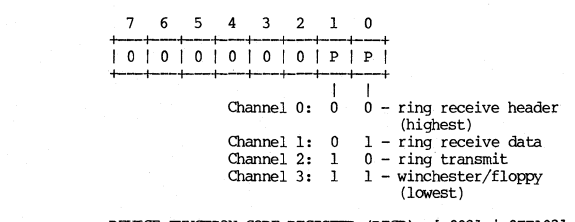
Not used.

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



Function code 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.

FLOPPY CONTROLLER

DN300

Address: [9C00 | 0FFA800]

	WRITE		READ
+00	ANSI COMMAND		ATTENTION STATUS
+02	ANSI PARM OUT		ANSI PARM IN
+04			DRIVE # OF STATUS
+06	SECTOR		CONTROLLER STAT-HI
+07			CONTROLLER STAT-LO
+08	CYLINDER-HI		
+09	CYLINDER-LO		
+0A	HEAD		
+0C	INTERRUPT CONTROL		
+0E	CONTROLLER CMND		
+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)	Cntr, if status avail enab
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	Timeout	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 timeout	Read attention status reg
0002	01	Parity error during DMA	Write to controller command reg

ANSI COMMANDS [9000 | 0FFA800]

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

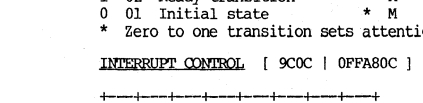
				mand/opt
7	80	Vendor unique errors	*	O
6	40	Other errors	*	O
5	20	Command reject	*	M
4	10	Speed error	*	O
3	08	R/W permit violation	*	O
2	04	Power fault	*	O
1	02	Read/write fault	*	M
0	01	Seek error	*	M

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]

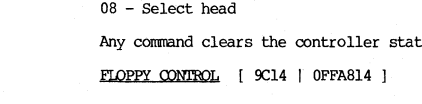


CONTROLLER COMMANDS [9C0E | 0FFA80E }

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]



DN4xx and DN600

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

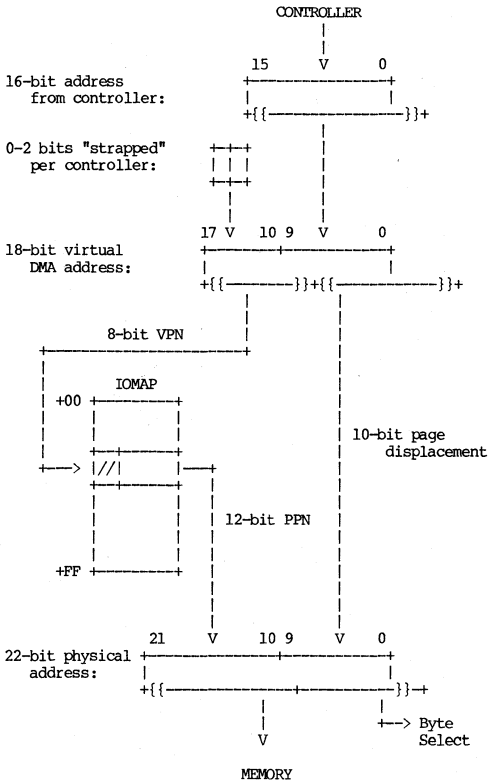
D7	D6	D5	D4	D3	D2	D1	D0

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

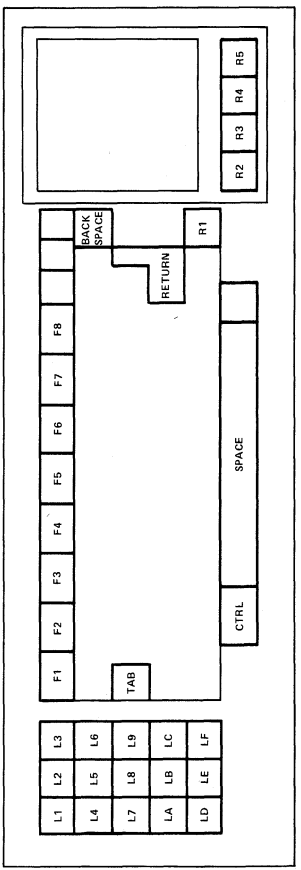
D7 - Fault from FDD
D6 - Write Protected
D5 - Ready from FDD
D4 - Track 0
D3 - Two Side
D2 - Head Address
D1 - Unit Select 1 Status
D0 - Unit Select 0 Status



IOMAP has 256 one-word entries from FFF800 - FFF9FE. FFFA00-FFFC00 reserved for future expansion (IOMAP 2).

I/O MAP ALLOCATION

DEVICE	IO PAGES	IOMAP 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)



Apollo I Keyboard - Map

Apollo I Keyboard Chart - Physical

Apollo I Keyboard Chart - Physical

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	^`	^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	

Apollo I Keyboard Chart - Translated (user mode)

Apollo I 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	^`	^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	N9U	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	

Apollo II Keyboard - Map

Apollo II Keyboard Chart - Physical

Apollo II Keyboard Chart - Physical

Low Order Nibble / ASCII Codes	Hign 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	

Apollo II Keyboard Chart - Translated (user mode)

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

MAGTAPE CONTROLLER

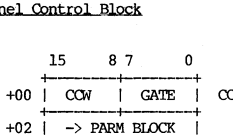
Controller control page: FE8000
Interrupt vector number: B3 ($2CC in page 0)
Multibus interrupt level: 3 (See MIC)
IOMAP entries: FFF980-FFF9FE (64 pages)

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

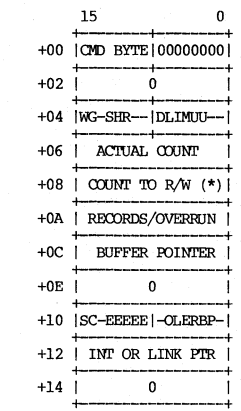
(Set up parameter block)

```
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	Online	(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 DEVICE ADDRESSES

	VIRTUAL	PHYSICAL
Multibus (64 pages)	FE0000	10000
line printer ctl	FEFC00	1FC00
Multibus int ctl	FEFC80	1FC80

PEB

[B000 | FF7000]

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

Definitions

peb_ctrl	FF7000	peb control register
peb_status	FF7000	peb status register (sys 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 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_$TSTX	Test S.P. or D.P. FAC for -,0,+; set cc's & D0.L then exit
FPP_$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 Addr)
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 & D0.L then
               exit
FPP_$DCCX     D.P. Compare FAC : Next 8 bytes; set cc's & D0.L
               then exit
FPP_$DLWV     D.P. FAC := Float[(SP)+]     (Float 16 bit integer)
```

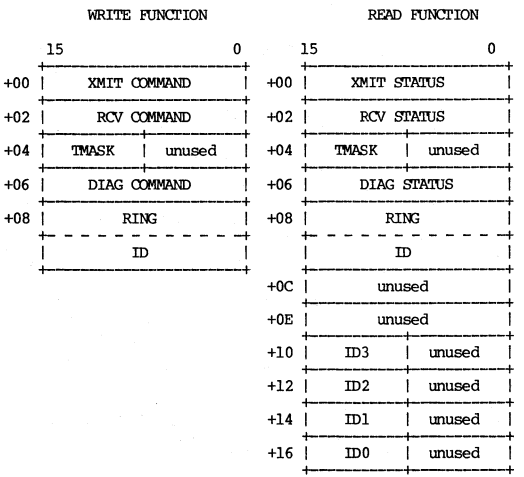
```
FPP_$DLWA     D.P. FAC := Float[((SP)+)]   (Float 16 bit integer)
FPP_$DLLV     D.P. FAC := Float[(SP)+]     (Float 32 bit integer)
FPP_$DLLA     D.P. FAC := Float[((SP)+)]   (Float 32 bit integer)
FPP_$DSTWX    D.P. D0.W := Fix[FAC]; then Exit
FPP_$DSTLX    D.P. D0.L := Fix[FAC]; then Exit
FPP_$SSCNV    Convert D.P. FAC to Single Precision
FPP_$DCNV     Convert Single Precision FAC to D.P.
FPP_$SSSQR    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_$CSTA     Complex Store FAC thru (SP)+
FPP_$CSTX     Complex Store FAC thru (SP)+ and exit
FPP_$CSWAP    Complex Exchange Real and Imaginary parts of FAC
FPP_$CCNV     Convert Single Prec to Complex (set imagFAC := 0)
FPP_$CCONJ    Complex Conjugate (imagFAC := -imagFAC)
```

RING/DISK

DN300 Ring/Disk

RING page at [9800 | 0FF9C00]



TRANSMIT COMMAND [9800 | 0FF9C00]

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

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

TRANSMIT STATUS [9800 | 0FF9C00]

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

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

RECEIVE STATUS [9802 | 0FF9C02]

8000 interrupt pending
4000 interrupt enabled
2000 busy
1000 disconnected
0800 biphase error
0400 elastic store bfr err
0200 timeout (The hdr of a msg was seen, but it never ended)
0100 crc error
0080 ack parity error (0=no error, 1=error detected)
0040 external error (err 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 pkt)
0002 wack (somebody before me WACKed the ptk)
0001 overrun (DMA didn't keep up with rcv 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 biphase error (error seen since bit was cleared)
0800 delay on (the delay is enabled)
0400 sticky good_seen (good pkt seen since bit was cleared)
0200 sticky elastic store bfr err (error seen since bit was cleared)
01FF bad packet count (9-bit counter for 1st detecting errs)

NOTES:

1. Counter is number of times this node found an error in a pkt going by (regardless of pkt target node ID), found the error bit in the ackbyte 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. first uses its highest order bit).

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

4. Writing anything to low byte of diagnostic command register (see below) clears interrupt and all sticky bits.

DIAGNOSTIC COMMAND [9806 | 0FF9C06]

8000 dma test (loop xmit DMA to rcv DMA)
4000 enable interrupt (bad_pkt_cnt overflow intterupt)
2000 connect (to the network)
1000 disconnect (from the network)
0800 delay off (disable the delay)
0400 delay on (enable the delay)
0200 snoop (accept all pkts but only set ackbyte for packets actually addressed to me)

NOTE:
Writing anything to low byte of clears interrupt and all sticky bits in 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.

DN4xx and DN600 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

	10	9	8
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]

15	14	13	12	11	10	9	8
BUSY	READY\	CRCERR	TIMOUT	0	0	BUSERR	OVRUN
7	6	5	4	3	2	1	0
0	0	x	x	x	x	x	x

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 3 revolutions of the disk. During a READ or WRITE operation it is an indication that the drive has has been positioned to the wrong cylinder, sector number is too large, surface is unformatted, or there was a CRC error on the disk header.

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

OVRUN - DMA overrun occurred during last disk operation.

Status	1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 5 4 3 2 1 0 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 over run	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]

15	14	13	12	11	10	9	8
BRDCST	T6	T5	T4	T3	T2	T1	T0

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.

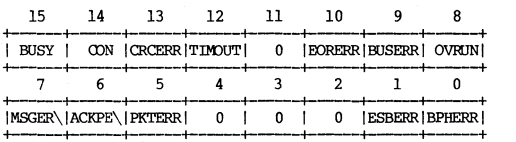
CON - Connect to the ring.

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 1 location past the point of the error.

OVRUN - A DMA overrun occurred during the last receive.

MSGER\ - No message error occured 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 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 msg
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 modems elastic store buffer.

BPHERR - An error occurred in the modems 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 over run	0	x	x	x	0	x	0	1	x	x	x	0	0	0	x	x
Biphase 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 0 length data field in the message.

DELAY - Enable an additional 7 bit delay into the length of the network. This may be required to support the recirculation of the token, which is 9 bits.

NDELAY - Disable the 7 bit delay.

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

Transmit Status Register

address: [BC14 | FF9C14]

15	14	13	12	11	10	9	8
BUSY	0	0	TIMOUT	0	0	BUSERR	OVRUN
7	6	5	4	3	2	1	0
MSGER\	ACKPE\	PKTERR	NCOPY	COPY	WACK	ESBERR	BPHERR

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 1 location past the point of the error.

OVRUN - A DMA overrun occurred during the last transmit. transmit. 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 occured 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 over run	0	0	0	x	0	0	0	1	x	x	x	x	x	x	x	x
Biphase 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 1.

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 1.

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]

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

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 - undef
1,0 - channel select

DMA Request Register

address: [base + 12]

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

BIT 2 - Request bit
1,0 - channel select

DMA Mask Register

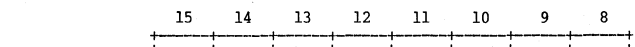
address: [base + 14]

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

BIT 2 - Set mask bit
1,0 - channel select

DMA ALL Mask Register

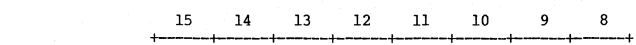
address: [base + 1E]



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]



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

SERIAL I/O INTERFACE FOR DN300

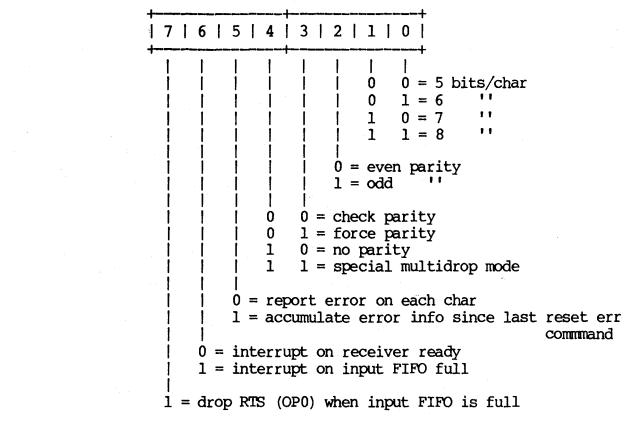
SIO page at [8400 | 0FFB000]

Sio lines implemented with a Signetics SC2681 DUART. Display keyboard interface implemented with a Motorola MC6850.

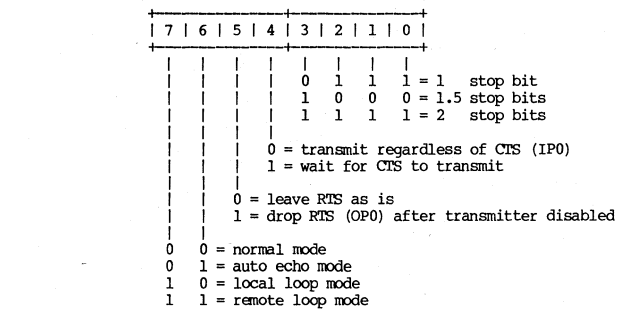
Register summary:

		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 Hld Reg. A (RHRA)	Transmit Hld 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 Hld Reg. B (RHRB)	Transmit Hld 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	

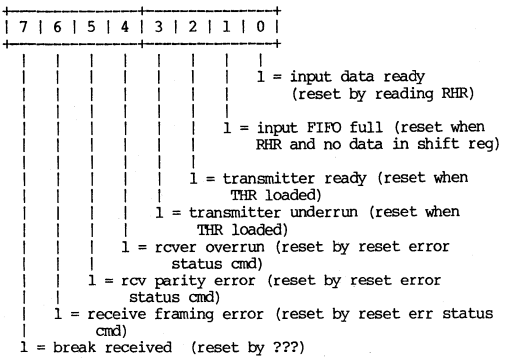
MODE REGISTER A, first access [8400 | 0FFB000]



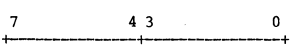
MODE REGISTER A, second & subsequent accesses
(until mode register pointer reset)



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

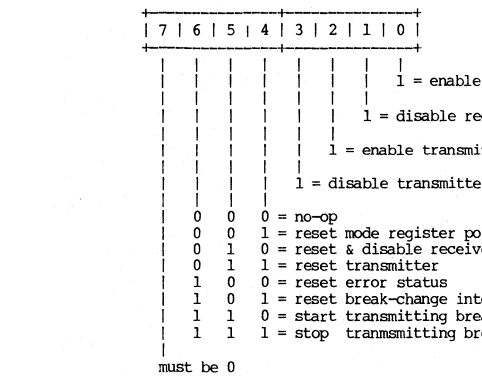


CLOCK SELECT REGISTER A [8402 + 0FFB002] write-only

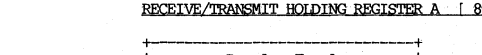


	ACR[7]=0	ACR[7]=1		ACR[7]=0	ACR[7]=1
0 -	50	75	8 -	2400	2400
1 -	110	110	9 -	4800	4800
2 -	134.5	134.5	A -	7200	1800(??)
3 -	200	150	B -	9600	9600
4 -	300	300	C -	38.4K	19.2K
5 -	600	600	D -	Timer	Timer
6 -	1200	1200	E -	IP4-16X	IP4-16X
7 -	1050	2000	F -	IP4-1X	IP4-1X

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

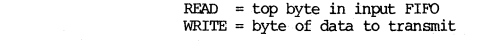


RECEIVE/TRANSMIT HOLDING REGISTER A [8406 | 0FFB006]



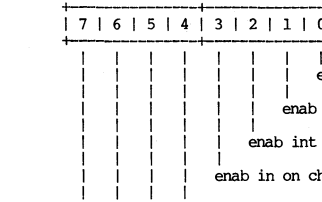
READ = top byte in input FIFO
WRITE = byte of data to transmit

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

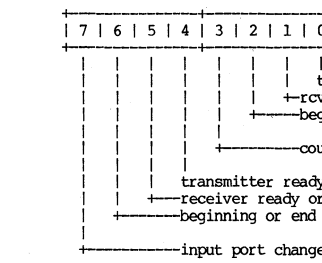


IP0 - Clear to Send (CTS) - A
IP1 - Clear to Send (CTS) - B
IP2 - Data carrier Detect (DCD) - A
IP3 - Data carrier Detect (DCD) - B

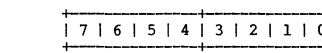
AUXILIARY CONTROL REGISTER [8408 | 0FFB008] write-only



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 0. 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]

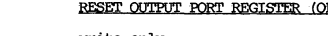
write-only

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]

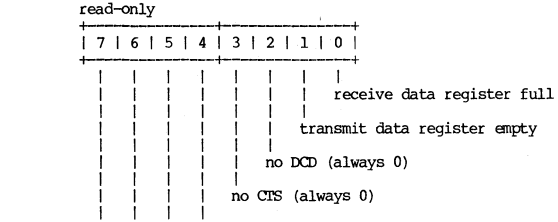
write-only



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

```
+---+---+---+---+---+---+---+---+
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
+---+---+---+---+---+---+---+---+
  |   |   |   |   |   |   |   |
  |   |   |   |   |   |   0   0 = clock/1
  |   |   |   |   |   |   0   1 = clock/16
  |   |   |   |   |   |   1   0 = clock/64
  |   |   |   |   |   |   1   1 = master reset
  |   |   |   |   |   |
  |   |   |   0   0   0 =7 bits, even parity, 2 stop bits
  |   |   |   0   0   1 =7 bits,  odd parity, 2 stop bits
  |   |   |   0   1   0 =7 bits, even parity, 1 stop bit
  |   |   |   0   1   1 =7 bits,  odd parity, 1 stop bit
  |   |   |   1   0   0 =8 bits, 2 stop bits
  |   |   |   1   0   1 =8 bits, 1 stop bits
  |   |   |   1   1   0 =8 bits, even parity, 1 stop bit
  |   |   |   1   1   1 =8 bits,  odd parity, 1 stop bit
  |   |   |
  |   0   0 = Set RTS, disable transmitter interrupt
  |   0   1 = Set RTS, enable       ''          ''
  |   1   0 = Reset RTS, disable    ''          ''
  |   1   1 = Set RTS, transmit break, disable transmitter
  |           interrupt
  1 = Enable receiver interrupts (receive data register
      full, overrun, loss of DCD).
```

DISPLAY KEYBOARD DATA REGISTER [8422 | 0FFB022]

```
+-------------------------------+
|       D   A   T   A           |
+-------------------------------+
```

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

```
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 Int. on next Rx Char		
			1	0	1	Reset Tx Int. pending		
			1	1	0	Error Reset		
			1	1	1	Return from Int (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 Int. Disable
 0 1 Rx Int. 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 Int. only
D0 - EXT Int. 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

TIMERS

Write CR1 or CR3	[8800	FFAC00]
Rd Status, Wr CR2	[8802	FFAC02]
Rd/Wr Timer 1, Hi	[8804	FFAC04]
Rd/Wr Timer 1, Lo	[8806	FFAC06]
Rd/Wr Timer 2, Hi	[8808	FFAC08]
Rd/Wr Timer 2, Lo	[880A	FFAC0A]
Rd/Wr Timer 3, Hi	[880C	FFAC0C]
Rd/Wr Timer 3, Lo	[880E	FFAC0E]
Calendar Control	[8880	FFAC80]
Calendar Data Wr	[8882	FFAC82]
Calendar Data Rd	[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 consists of four bytes, as follows:

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

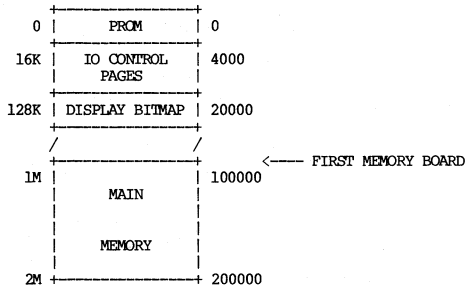
The range of X and Y coordinates is approximately 30 to 1100.

CHAPTER 7

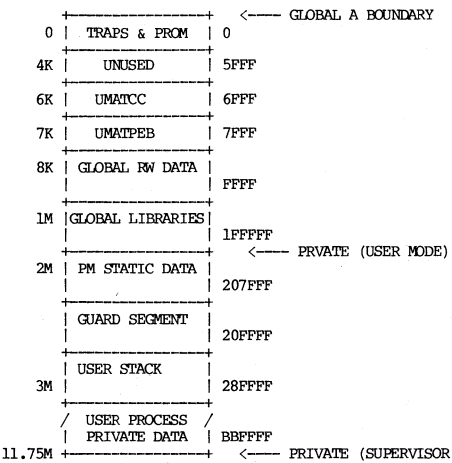
PROGRAMMING INFORMATION

ADDRESS SPACE

Physical Address Space



Virtual memory



ASSEMBLER LANGUAGE SUMMARY

Program format

```
PROGRAM name,start-addr [,proc-sect [,data-sect]]
MODULE name [,proc-sect [,data-sect]]

PROC
pure code and data      (no variables, no relocation)

DATA
read/write  code  and  data  (including  all  address
constants)
    .
    .
    .
    more PROC and DATA sections, as desired
    .
    .
    .
END
```

Notes

1. These pseudo-ops must not start in column 1.
2. Use PROGRAM when writing a main program, to specify the start address. Use MODULE otherwise.

Statement Syntax

Comment: * any characters up to end of line.

Instruction or Pseduo-op:
[label]^opcode^[operand1[,operand2]]^[comment]

Notes:

- * of comment field must be in first character position of line
- label field if present must begin in first character position of the line
- ^ means one or more spaces
- [ ] encloses optional portions
- Assembler is not case-sesitive

Expression Syntax

* means current location

\# means immediate operands

$ means hex constants

DB, SB, and SP are synonyms for registers A5, A6, and A7 respectively

Operators in decreasing priority:

*, /	
+, -	
<, >, <=, >=, =, <>	relationals
<<, >>	shifts
&	bitwise and
!	bitwise or

Op Codes

ABCD	BSET	DBT	MOVEA	SEQ	UNLK
ADD	BSR	DBVC	MOVEM	SF	
ADDA	BTST	DBVS	MOVEP	SGE	
ADDI	BVC	DCNT	MOVEQ	SGT	
ADDQ	BVS	DEFDS	MULS	SHI	
ADDX	CHK	DEFS	MULU	SLE	
AND	CLR	DIVS	NBCD	SLS	
ANDI	CMP	DIVU	NEG	SLT	
ASL	CMPA	ELSE	NEGX	SMI	
ASR	CMPI	ENDC	NOP	SNE	
BCC	CMPM	EOR	NOT	SPL	
BCHG	DBCC	EORI	OR	ST	
BCLR	DBCS	EXG	ORI	STOP	
BCS	DBEQ	EXT	PEA	SUB	
BEQ	DBF	IFEQ	RESET	SUBA	
BGE	DBGE	IFF	ROL	SUBI	
BGT	DBGT	IFNE	ROR	SUBQ	
BHI	DBHI	IFT	ROXL	SUBX	
BLE	DBLE	JMP	ROXR	SVC	
BLS	DBLS	JSR	RTE	SVS	
BLT	DBLT	LEA	RTR	SWAP	
BMI	DBMI	LINK	RTS	TAS	
BNE	DBNE	LSL	SBCD	TRAP	
BPL	DBPL	LSR	SCC	TRAPV	
BRA	DBRA	MOVE	SCS	TST	

Pseudo-ops

```
AC          expr                    Define address constant
DA[.x]      'string'[,...]          Define ASCII string
DATA                                Set location sector to end of
                                     data section
DC[.x]      [repeat-cnt@]           Define constant
             const-expr[,....]
DEFDS       expr                    Define symbols in descending
                                     order
DEFS        expr                    Define symbols in ascending
                                     order
name  DFSECT     [attributes]       Define section (attributes
                                     are: READONLY, OVERLAY,
                                          INSTRUCTION, ZERO)
DROP        An                      End static range of using
DS [.x ]    expr                    Define unintialized storage
EJECT                               Do top of form
END                                 End of source
ENDS                                End of DEFS or DEFDS
ENTRY[.y]   name                    Declare entry point
label EQU          expr             Define label as expr
EXTERN[.y]  name                    Declare external
LIST                                Enable listing
MODULE      name                    Module header
          [,proc-sect-name[,data-set]]

NOLIST                              Inhibit listing
ORG         expr                    Set location counter within
                                     section
PROC                                Set location counter to end
                                     of named section
PROGRAM     name[,start-addr
              [,proc-sect-name[,
              data-set-name]]]      Program header
SECT        name                    Set location counter to end
                                     of named section
USING       An,expr                 Instruction's addressing
                                     modes; fields can assume
                                     expression in register An
```

Notes:

```
.x ->  .B, .W, or .L   (Byte, Word, Long)
.y ->  .D, .P, or .F   (Data, Procedure, Function)
```

Notes

1. Only comments, EJECTS, and EQUs can precede the PROGRAM or MODULE pseudo-op.

2. The USING pseudo-op generates no instructions. It simply tells the assembler that the address of the expression is in address register Ax. The assembler uses this information in forming the effective address

fields of instructions. Program instructions must load the address register with the address of the expression and and preserve it over the static range that begins with the USING pseudo-op. The DROP pseudo-op ends the static range.

3. There is an implicit USING at the beginning of each assembly language module. The assembler assumes that A5 contains the address of the start of the DATA$ section. DROP can be used to disable this.

Register usage

A5 - DB
A6 - SB
A7 - SP

Structures

DEFS defines a structure, addressed in ascending order. DEFDS defines a descending structure. ENDS marks the end of a definition.

For example:

```
DEFS      4(sp)
a     ds.w  1
b     ds.l  1
c     ds.l  1
ENDS
```

Sections

An assembly language module may have any number of sections. The assembler creates two sections automatically — a read-only, concatenate, instruction section named PROCEDURE$, and a read/write, concatenate, data section named DATA$. The names PROCEDURE$ and DATA$ can be changed in the PROGRAM or MODULE statement.

Sections and labels may have the same name. The name is recognized as a section name only in the context of DFSECT and SECT statements.

DFSECT

DFSECT defines a section. Its syntax is:

section-name DFSECT section-attributes-list

e.g. DEBUG$ DFSECT READONLY

Valid section attributes are READONLY, OVERLAY, ZERO, and INSTRUCTION. If no attributes are specified, the section will be a data, concatenate, read/write section. OVERLAY

sections overlay FORTRAN COMMON blocks and Pascal sections of the same name. ZERO instructs the loader to initialize the section to zero before loading.

SECT

SECT directs the assembler to use a named section. Its syntax is:

SECT section-name

e.g. SECT DEBUG$

This pseudo-op directs the assembler to put data or instructions in the specified section. It is similar to the PROC and DATA pseudo-ops, except that any section may be used.

Usage Information

Command line:

ASM source-file [options]

Options: (* indicates default options)

*	-L [<listing file>]	Generate assembly listing
	-NL	Suppress assembly listing
*	-B [<binary file>]	Generate binary file
	-NB	Suppress binary file
	-XREF	Generate cross reference listing
*	-NXREF	Suppress cross reference listing

Source file names must end in .asm. Output listing files and binary files end in .lst and .bin, respectively. For example, asm foo looks for foo.asm as input, and creates foo.bin and foo.lst.

Conditional Assembly Pseudo-ops

Note: ASM has conditional processor directives similar to PASCAL (see CONDITIONAL PROCESSING below).

```
IFEQ expr
IFNE expr
IFT  expr
IFF  expr
ELSE
ENDC
```

Notes

1. IFT and IFNE assemble if expr <> 0.

2. IFF and IFEQ assemble if expr = 0.

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 DATA BASE REG
CLR.L     -(SP)         PUSH A RESERVED WORD
MOVE.L    A0,-(SP)      PUSH ADDRESS OF MY ECB
MOVE.L    6(A0),DB      LOAD MY DATA BASE FROM ECB
LINK      #autosize,SB 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 not preserved.

CONDITIONAL PROCESSING

Conditional processing applies to both PAS and ASM.

Command Line Syntax

```
pas [filename] -CONFIG name1 name2 ...
```

Conditional Compilation Control Structure Syntax

```
%if <predicate> %then                %ifdef <predicate> %then
 text                                 text
%elseif <predicate> %then            %elseifdef <predicate> %then
 text                                 text
%else                                %else
  text                                 text
%endif                               %endif

<predicate> is of the form:
  name                                — attribute name
  not <predicate>                     — highest precedence
  <predicate> and <predicate>         — "and" higher than "or"
  <predicate> or <predicate>          — lowest precedence
  (<predicate>)                       — grouping

Additional Syntax:
 %var name1 name2 ...;               — enumerate (declare) set of
                                       valid attributes
 %enable name1 name2 ...;            — enable attributes
 %config                             — predeclared attribute
 %error 'string'                     — generate compiler diagnostic
 %warning 'string'                   — generate compiler diagnostic
 %exit                               — Stop processing this file
  ((%elsif, %elsifdef, and %end may also be used))
```

Syntax Restrictions

- Lines must exclusively contain either conditional compilation directives or language source text.
- Only attribute names given on the command line and names created with %var may be used in a <predicate>.
- %error and %warning must be on lines by themselves.

Semantics

- Names must be declared in a %VAR statement before being used. Names given on the command line with -CONFIG must also appear in a %VAR statement before being used in a predicate.

- A name evaluates to TRUE in a %IF/%ELSEIF statement if it is enabled on the command line with the -CONFIG option, or enabled with a %ENABLE statement.

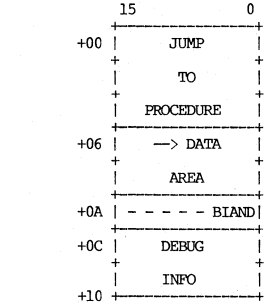
- A name evaluates to TRUE in a %IFDEF/%ELSEIFDEF statement if-and-only-if it is declared with a %VAR statement.

- The pre-declared attribute name %CONFIG is TRUE iff a -CONFIG option was given on the command line with non-null arguments.

- Declaring an already declared name will result in a warning. Using an undeclared name will result in a warning. Enabling a name that is already enabled will result in a warning.

ENTRY CONTROL BLOCK (ECB)

(type db_$ecb_rec in /us/debug/db.debug$.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	Lineprinter
.AUX	Auxillary
.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.

PATHNAME SYNTAX

Symbol	Starting Point
//	Network root directory
/	Node entry directory
~	Naming directory
\	Parent directory
. or none	Working directory
`	/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
```

PASCAL EXTENSIONS

PAS compiler: Features for internal Apollo use only are marked with (*).

A Pascal "procedure" or "function" may be given a variety of attributes or options which may be used to change the behavior of the routine or the system usage of the routine.

VAL_PARAM: (*)

"VAL_PARAM" is an attribute which can be specified for a procedure or function. When it is given, arguments which are not to be changed by the called routine (i.e. 'value' arguments and 'IN' arguments) and which are 32 bits or less, are passed by "value" instead of by "reference". This makes the code a little more efficient. If you use this attribute, you may NOT invoke the routine from Fortran or C. Note that VAL_PARAM is superfluous if the INTERNAL attribute is used, or the routine is nested.

Example:

```
FUNCTION fnd_name (IN pname : IDNLN_REC;
                   IN pdef_flag : BOOLEAN) : OBJECT_PTR;
                   VAL_PARAM; EXTERN;
```

OPTIONS() clause

The "OPTIONS" clause is another way of giving procedure or function attributes. It allows you to group your attributes in one place. If you use the "OPTIONS" clause, you may NOT have attributes outside it. Furthermore, the "OPTIONS" clause permits additional attributes.

In the example above, the function could have been specified as follows:

```
FUNCTION fnd_name (IN pname : IDNLN_REC;
                   IN pdef_flag : BOOLEAN) : OBJECT_PTR;
                   OPTIONS(VAL_PARAM, EXTERN);
```

Note that the attributes of VAL_PARAM and EXTERN are separate by commas in the OPTIONS clause.

Furthermore, the following attributes, if desired, may ONLY be given in an OPTIONS clause. These are:

INHIBIT:(*)

The inhibit attribute sets a bit in the ECB flags which inhibits asynchronous faults for the routine.

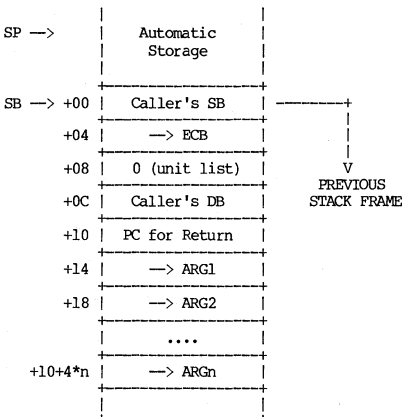
BACKSTOP:(*)

The backstop attribute sets a bit in the ECB flags which stop looking back in stack for "inhibit"ed routines.

VARIABLE:

The variable attribute allows a variable number of arguments to be given to the routine invocation. The number of arguments may be less than or equal to the number of parameters in the declaration. It may NOT be more. It is up to the writer of the routine to know how many arguments were actually passed. Generally, some kind of argument count should be given as one of the arguments (such as the first one).

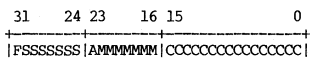
STACK FRAME



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

STATUS WORD

("status_t" in base.spo.bbas)



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 also CHAPTER 4 ERROR CODES.

CHAPTER 8

SYSTEM DEBUGGING

DEBUG COMMAND EXTENSIONS

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

New Commands

- REGS -- Display the registers of the target program.
- DB — Invoke the machine-level debugger DB (see below).
- VA -- Show the Virtual Address of current program location, one named routine, or a list of variables:

  VA [{ var[,...,var] | -Routine [<routine-name>]}]

  VA by itself shows the current symbolic location and the virtual PC.

  VA -Routine [<routine-name>] shows the starting address of the named routine, or the current routine if the name is omitted.

  VA var[,...,var] shows the address of each variable.

Options to Current Commands

Breakpoint

Breakpoint has an option to set a breakpoint on a virtual address. The formats are:

```
Breakpoint -VA va_number
     or
Breakpoint -VA routine-name\rva_number
```

The -VA option must be given. Then, if "va_number" is given, this is take to be some virtual address where a breakpoint will go.

If "routine_name\rva_number" is given, then the routine is found, and the "rva_number" is assumed to be a ZERO-RELATIVE number which is added to the start of the named routine to make a virtual address breakpoint.

Remember that DEBUG assumes base 10; hexadecimal numbers must be given with a leading "16#". Also, NO CHECK is done to see if the number supplied is legal.

Delete

Delete has an option to remove breakpoints made with the -VA option. The formats are:

```
Delete -VA va_number
       or
Delete -VA routine-name\rva_number
```

Miscellaneous

The List command is unchanged; you cannot list these -VA breakpoints exclusively. Listing all breakpoints will show them, however.

DEBUG has another invocation option which will show a brief section map of the loading of the target program. The option is "-smap", and it must come BEFORE the target name. Example:

```
$ debug -smap zd/z.63.bin
Section Map:
#  Location   Size    Name
1 002EA768 000002F4 PROCEDURE$
2 00300000 0000FED0 DATA$
3 002EAA5C 00000A76 DEBUG$
```

The section map is always displayed on ERROUT.

Also, here is some useful information which IS PUBLIC, but which people often forget how to do.

There are 3 debugger names which 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
help
in <path>               install library
lo <path>               load program
ma <path> [-ex]         map a file
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:

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
dp [<pid>]          display pcb (first ten if no pid
                      entered)
dv <addr>           convert db address to virtual
                      address
ff [<addr>]         try to find stack frame in
                     addr - addr+1024
gd <unit>           get (pbu) dcte
m                   enter mapped mode
ms <args>           mapped search (just like md's 's')
p                   enter physical (normal) mode
pv <addr>           convert ppn to virtual address
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
vp <addr>           convert virtual address to ppn
wh[p|d|e] <sym or addr> look up [proc|data|ecb] or
                          address in aegis map
```

You execute from 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 {virtual address of receive register}
```

Do not execute the lights program from a color node. It will crash the network.

The transmit and receive status registers are displayed at the bottom of the screen. These register are described in Chapter 6 Peripheral I/O. 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 DB environment}
! LI 0      {Provide address of zero to lights program}
! q         {exit from DB environment}
<ctrl> F    {remove lights from bottom of 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
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 address and contents according to <size_spec> and <base_spec>.

B [<location>]

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

C <start> <end> <target>

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

CA <start>

Calls the subroutine which 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 bounds <start> to <end> onto terminal printing address followed by specified <items_per_line>. 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 complete, the LOW, HIGH, and START addresses are displayed.

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

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

G [<location>]

Jumps to <location> after inserting 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 rearranged as shown under Address Space in Chapter 2.

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 bounds <start> to <end> for <value> through 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 Winchester unit and acknowledges outstanding interrupts. This command also enables the POWER-OFF key.

P

Turns off 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.

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 4]

<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 4]

<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 dec.

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

:A ::= numbers and immediate constants printed in ASCII. All numeric input defaults to hexadecimal. $num implies hexadecimal. <base>$num implies 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 will be reported by the crash_system routine, which will point address register 0 (A0) at a standard error code (see Chapter 4 ERROR CODES) and execute a TRAP instruction, causing entry to the PROM mnemonic debugger with an "S" code (see MNEMONIC DEBUGGER ERROR CODES in Chapter 4). Type

```
A (A0):L
```

and look up the resulting long-word displayed in the list of Aegis error codes. If A0 is pointing to a standard code, you have the immediate reason for the crash; if it doesn't, something strange has happened. In either case, A6 (SB) will probably point to the stack frame of (1) the routine that called crash_system (which doesn't push a frame) or (2) the routine that blew up. Refer to Stack Frame format to trace back the BCB addresses of the callers leading to the crash.

Other useful things to look at:

Bit 2 of the Bus Status Register (word at FFB40A):
if set then CPU B was running.

MMU Status Register (long word at FFB404):
Contains virtual address of last MMU miss (see MMU Status Register).

In AEGIS' DATA$ section (see map of AEGIS):
PROC_$CURRENT (word) = PID of current process.
PCBR + (current PID*4) = PCB of current process (see PCB).
CPUB_STATUS (long word) = last error code from CPUB (only if now on CPUA).

SYSTEM DUMPS

This routine saves the state of the system at the time of the crash and allows you to write the saved information to one of the following:

- A floppy diskette
- A file on another node

The dump routine copies the MMU, RING, and DISK registers to main memory, and then writes their contents to disk or file. After the registers have been copied, physical memory is dumped, starting at $100000 and continuing through $200000 (1 Mb), skipping missing memory boards where necessary. On completion, the dump routine executes a TRAP instruction and returns to the Mnemonic Debugger (MD).

If a disk error occurs while the dump routine is executing, the operation that encountered the error is retried 25 times. If, after 25 attempts, the operation has not successfully completed, the dump routine skips the current record and page and resumes processing with the next sequential page.

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.)

2. Issue the sequence of MD commands listed below. The commands listed below use the following conventions.

Conventions

<RETURN>	Press the Return key
[junk]	Irrelevant value returned by mnemonic debugger (MD)
[SP]	Number representing the Stack Pointer value
$	Prefix used to identify numbers as hexadecimal to MD

MD Commands	Comments
>D A7:L<RETURN>	Get Stack Pointer
nnnnnnnn: xxxxxxxx	Note value displayed (xxxxxxxx); it is the Stack Pointer in hexadecimal.
>RESET<RETURN>	Reset the system
><RETURN>	
MD mm/dd/yy	
>A 100000:L<RETURN>	Store Stack Pointer
100000: [junk] $xxxxxxxx[SP]/	Replace junk with Stack Pointer value (xxxxxxxx) from above; enter /
Then execute either	
>G 100C00<RETURN>	Start DUMP to floppy
or	
>DI N nn	nn is the node id of the node on which the dump file is created. The memory dump routine returns the name of the dump file.
>G 100C04	Start the DUMP to a file

3. When the dump routine is complete, it executes a TRAP $F instruction and returns to MD.

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.

/SYSTEST/SSR_UTIL

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

- 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 referencable from outside the module are cross referenced.

  Usage: BCR

- DISK_ERR -- Displays information about 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 0. By default Dump_file dumps the entire input file.

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

- FMPD -- FORMAT_PROCESS DUMP formats a process dump and writes the formatted dump to standard output. Process dumps are made by the process manager when a process terminates abnormally and appended to the file "`node_data/proc_dump". Information included is similar to that provided by the FAULT_STATUS (FST) and TRACE_BACK (TB) commands.

  The pathname(s) are the name(s) of files containing unformatted process dumps. If none is given, "`node_data/proc_dump" is assumed. The explicit dump file name facility allows dumps to easily be saved by copying the "`node_data/proc_dump" file.

  Usage: FMPD [pathname...]

- LTBL -- Prints the debug line number table

  Usage: LTBL (enter object module, the help)

- NETEX — Provides network statistics for all nodes in the network.

  Usage: NETEX
  <ctrl> Z to stop and display information

- 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 activate ring network message logging
  -Stop deactivate ring network message logging (if active) and display current contents of the log. (This is the default!)
  -Read display 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, you can issue a <CR> to use a default location for the buffer, which RWVOL then displays. If you specified a start address, you must specify an end address when prompted for it; otherwise just 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 offline) to look at the contents of the record that you have read, in the location the program displayed for the buffer.

  Usage: RWVOL

- SALT -- SALT finds all uncatalogued permanent objects in a volume. It uses or creates a directory ORPHANS in the root of the volume and enters the names of all objects not catalogued elsewhere. Uncatalogued directories are found first, so no redundancy occurs. SALT will catalog the OS paging file which should not be deleted, but simply uncatalogued. Only use SALT if you have full ACL rights.

  Usage: SALT [-V | -VERIFY] [volume_pathname]

  The options are:

  -V, -VERIFY verify only; don't catalog any orphans

- SIOLOGIN -- SIOLOGIN listens to an sio line, invokes the login sequence and if the login is successful, runs the program, prog. If prog is not specified, /com/sh is invoked. Dev_name is the sio device descriptor pathname and must be specified. Options, if specified, must precede the prog and its arguments.

  SIOLOGIN is most useful in conjunction with SIOMONIT.

  Usage: SIOLOGIN dev_name [[-DIALIN] [-N name]
  prog [args...]]

  Options:

  -DIALIN The sio line connection is remote. If the line is a dialup, SIOLOGIN asks for an access password before invoking the login sequence. The access password is a single string read from `node_data/siologin_access. Also for remote lines, SIOLOGIN waits for carrier detect and hangs up after the invoked program returns. If the connection is local, siologin waits for a cr before beginning the login sequence. SIOLOGIN logs invalid logins in `node_data/siologin_log.

  -N name specifies the name to give the process. (Takes precedence over the cpo option -n when siologin is cpo'd directly instead of through siomonit.)

- SIOMONIT -- SIOMONIT allows repeated logins over sio lines, independent of login/logout activity at the node keyboard. The file, filename, contains argument lists to be passed to invocations of siologin (see the description of

siologin for more information). A maximum of three argument lists (one per sio line) are processed. SIOMONIT must be cpo'd from the DM's startup file (`node_data/startup or /sys/dm/startup) to enable the logins to work.

Usage: SIOMONIT filename

Each argument list in the file must have the form:

[-Repeat] siologin_arg_list

where:

-R[epeat]	means re-invoke this process when it returns.
siologin_arg_list	is an siologin argument list of the form:

dev_pathname [[-dialin] [-n procname] prog [args ...]]

For each argument list, siomonit invokes:

/com/siologin siologin_arg_list

Arguments are passed to siologin uninspected. If a pgm_$invoke fails, it is removed from the list. If a process returns from its first invocation within 15 seconds or if it returns from subsequent invocations within 15 seconds, it is removed from the list. Comments may be included in the file and must begin with Siomonit writes a log of its problems in `node_data/siomonit_log.

This program idles, waiting for a fault when it has no processes left to wait for or repeat. Receiving a 'stop' fault causes it to exit; a 'quit' fault causes it to resume from the beginning. (If not idling, quit fault acts like stop fault). If siologin processes fail to stay up, check both siologin_log and siomonit_log, correct the problem(s) and issue 'sigp siomonit' to begin again. Use 'sigp siomonit -stop' to kill siomonit.

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

  Usage: XMT

A-1 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				

B-1 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

-S-

READER'S RESPONSE

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

Document Title: Apollo DOMAIN Engineering Handbook
Document Revision: 01
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