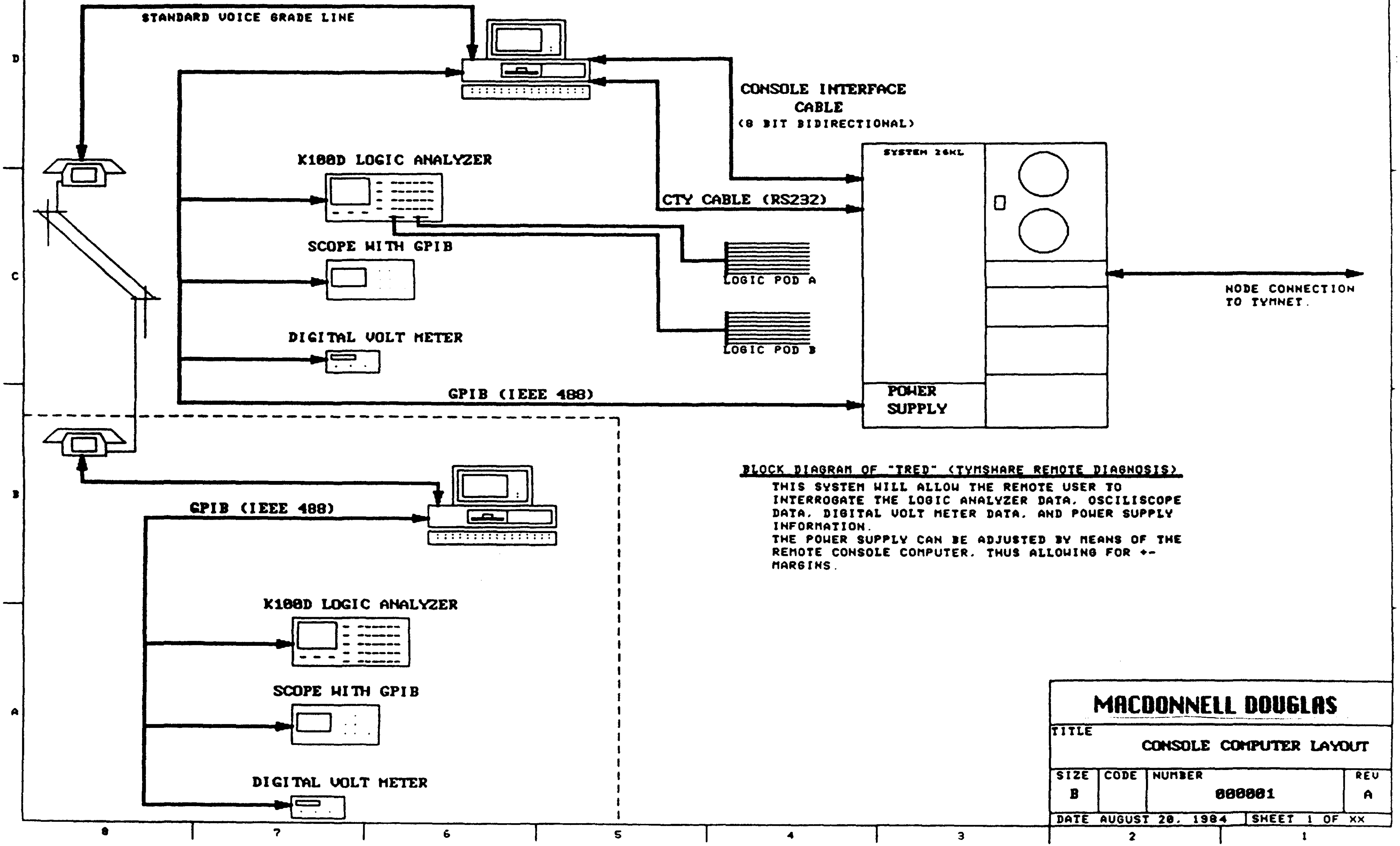
CCBLK.DWG
26KL



BLOCK DIAGRAM OF "TRED" (TYMSHARE REMOTE DIAGNOSIS)

THIS SYSTEM WILL ALLOW THE REMOTE USER TO INTERROGATE THE LOGIC ANALYZER DATA, OSCILISCOPE DATA, DIGITAL VOLT METER DATA, AND POWER SUPPLY INFORMATION.
THE POWER SUPPLY CAN BE ADJUSTED BY MEANS OF THE REMOTE CONSOLE COMPUTER, THUS ALLOWING FOR +- MARGINS.

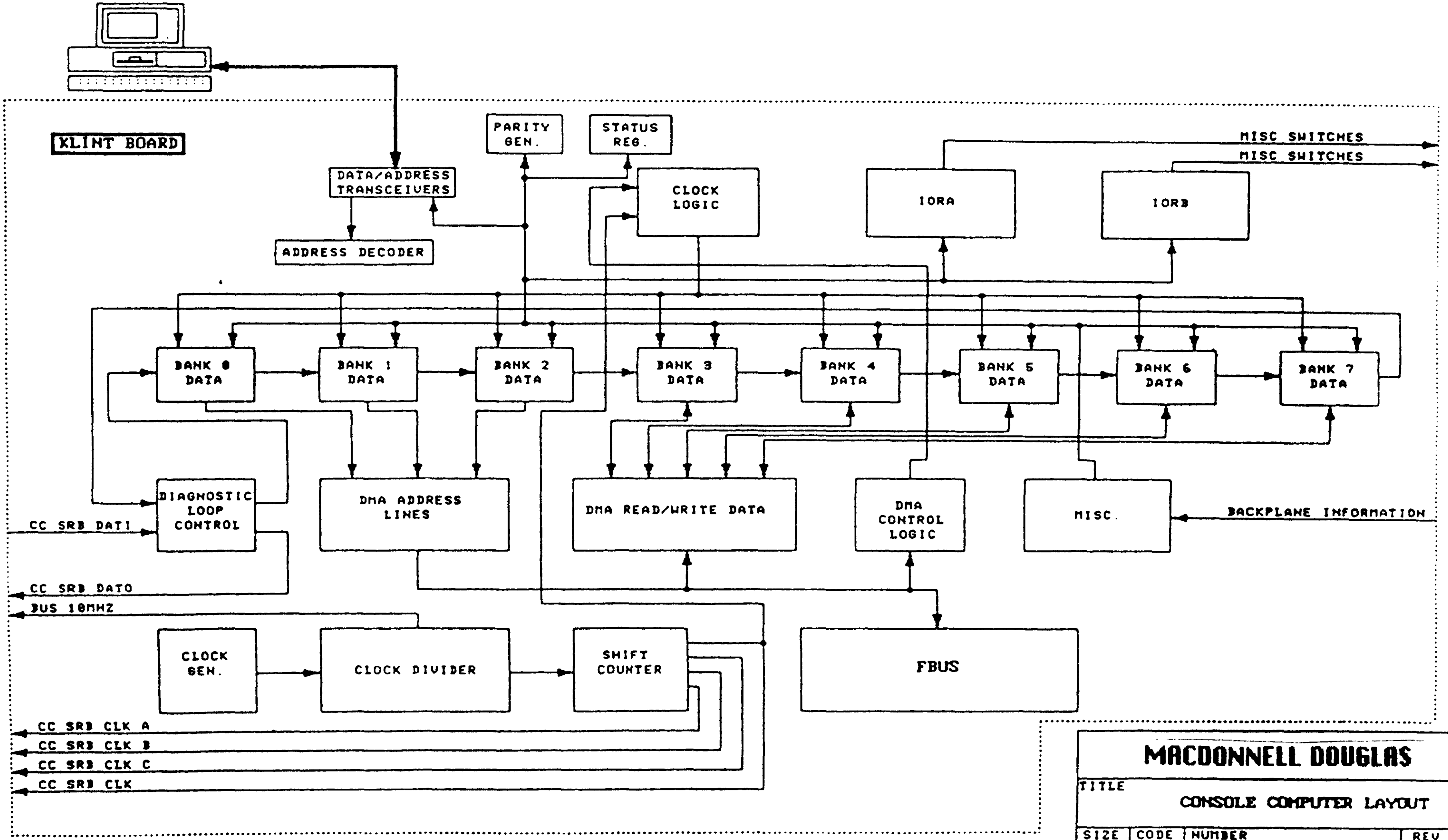
MACDONNELL DOUGLAS

TITLE CONSOLE COMPUTER LAYOUT

SIZE	CODE	NUMBER	REV
B		000001	A

DATE AUGUST 20, 1984 | SHEET 1 OF XX

CCBLK3.DWG
26KL

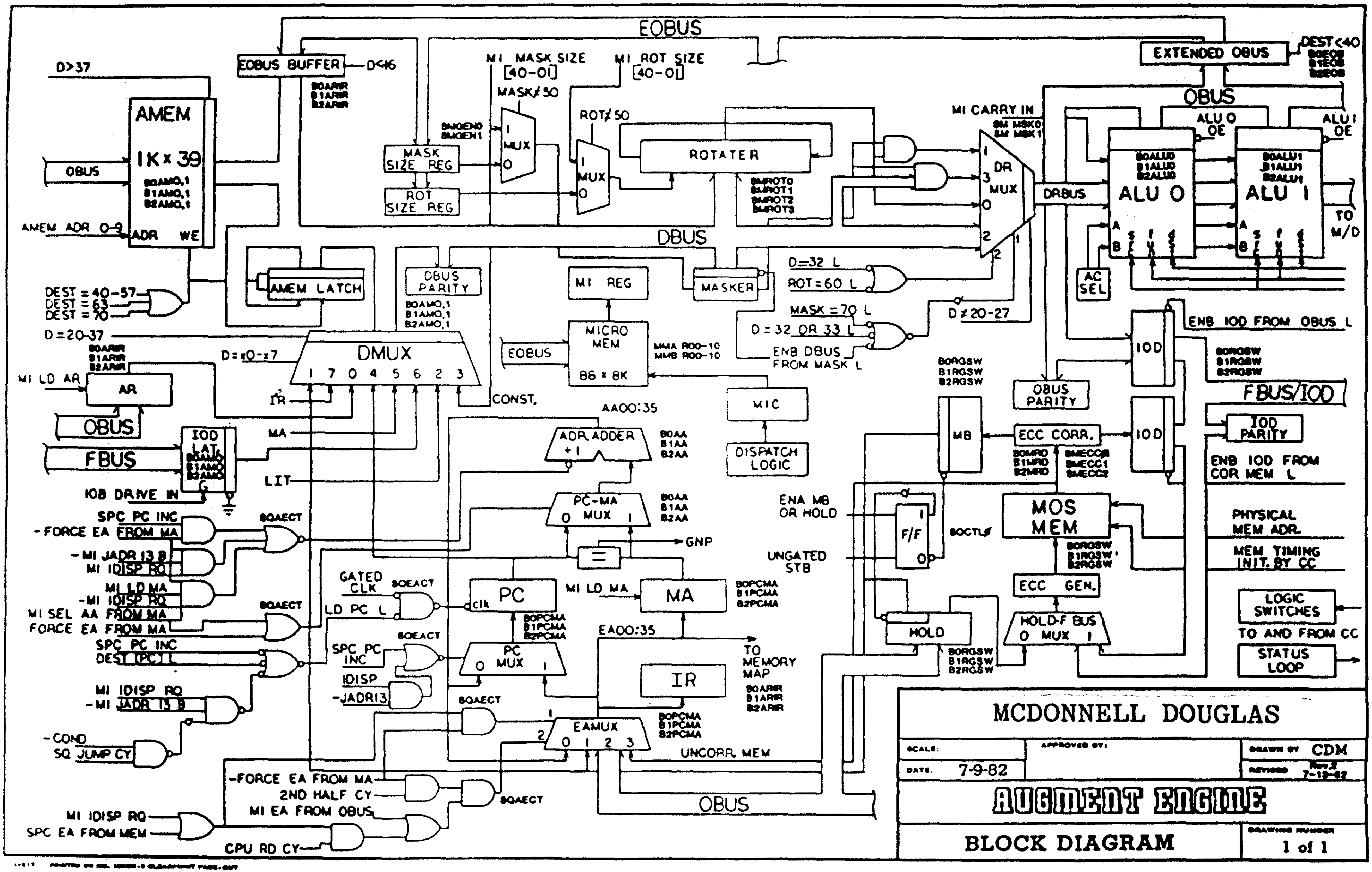


MACDONNELL DOUGLAS

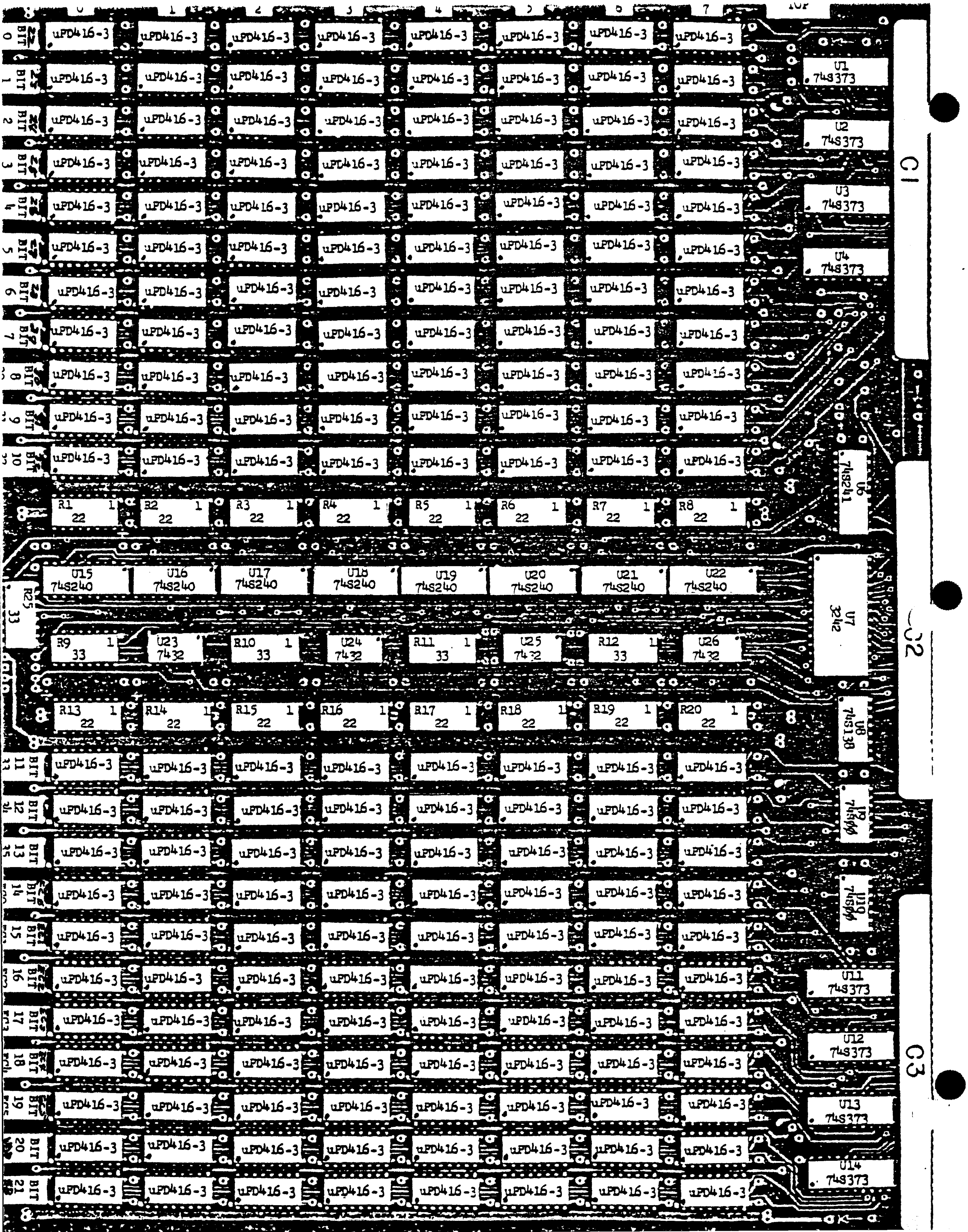
TITLE CONSOLE COMPUTER LAYOUT

SIZE	CODE	NUMBER	REV
B		000081	A

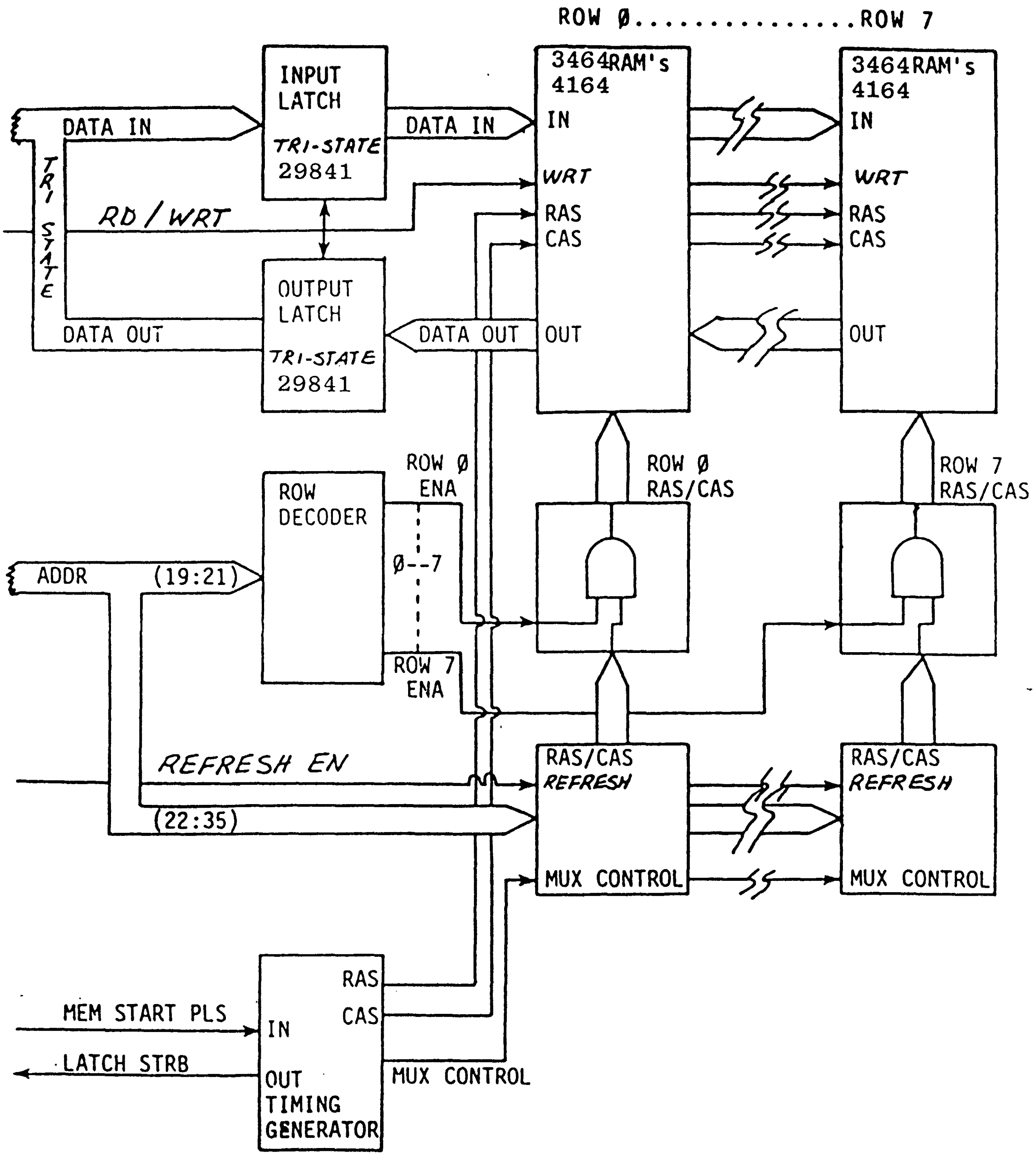
DATE 12-28-84 SHEET 2 OF 3



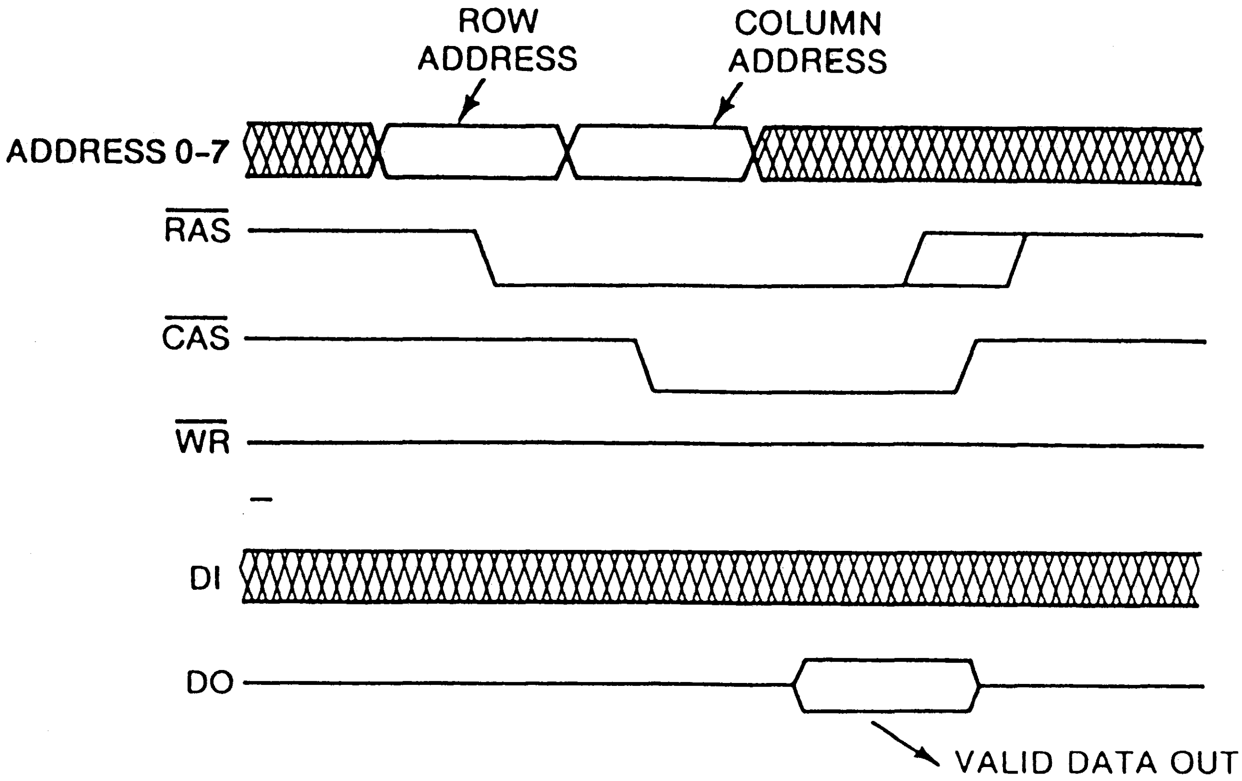
MCDONNELL DOUGLAS		
SCALE:	APPROVED BY:	DRAWN BY CDM
DATE: 7-9-82		REVISED Rev 2 7-12-82
AUGMENT ENGINE		
BLOCK DIAGRAM		DRAWING NUMBER 1 of 1


0
1
2
3
4
5
6
7
10P
0 BIT
1 BIT
2 BIT
3 BIT
4 BIT
5 BIT
6 BIT
7 BIT
8 BIT
9 BIT
10 BIT
11 BIT
12 BIT
13 BIT
14 BIT
15 BIT
16 BIT
17 BIT
18 BIT
19 BIT
20 BIT
21 BIT
uPD416-3
U1 74S373
U2 74S373
U3 74S373
U4 74S373
C1
R1 1 22
R2 1 22
R3 1 22
R4 1 22
R5 1 22
R6 1 22
R7 1 22
R8 1 22
U6 74S241
U15 74S240
U16 74S240
U17 74S240
U18 74S240
U19 74S240
U20 74S240
U21 74S240
U22 74S240
R25 33
U7 3242
C2
R9 1 33
U23 7432
R10 1 33
U24 7432
R11 1 33
U25 7432
R12 1 33
U26 7432
R13 1 22
R14 1 22
R15 1 22
R16 1 22
R17 1 22
R18 1 22
R19 1 22
R20 1 22
U8 74S138
U9 74S00
U10 74S00
U11 74S373
U12 74S373
U13 74S373
U14 74S373
C3

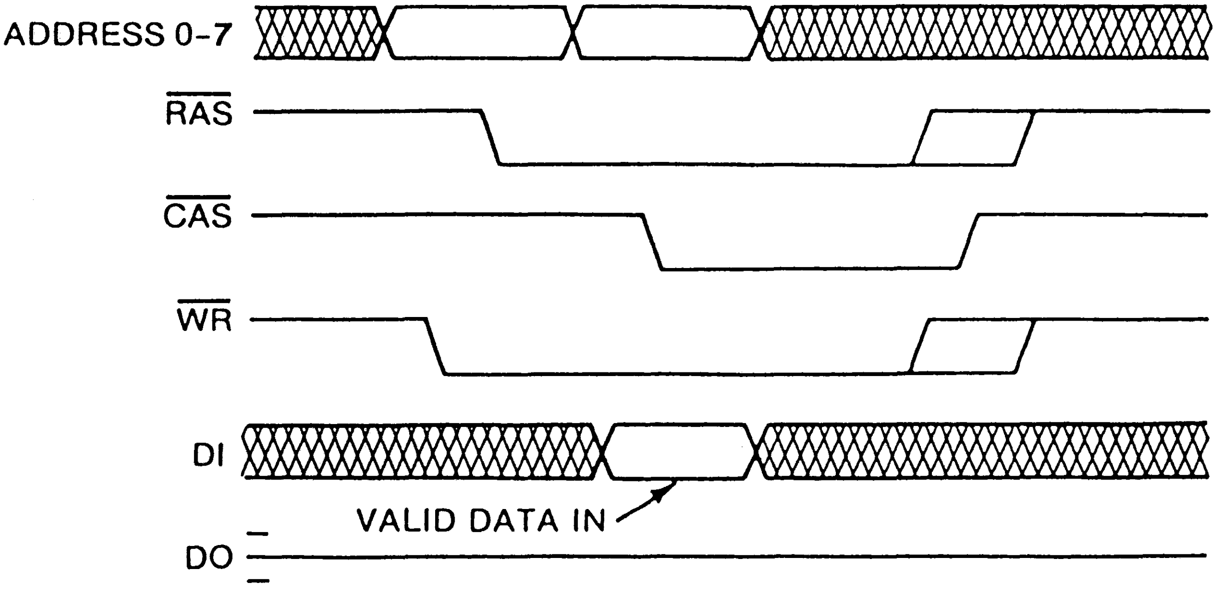
Main Memory



REVISED 21 NOV 85 F.J.R.W.



Read cycle.



write cycle.

4164 RAM cycles.

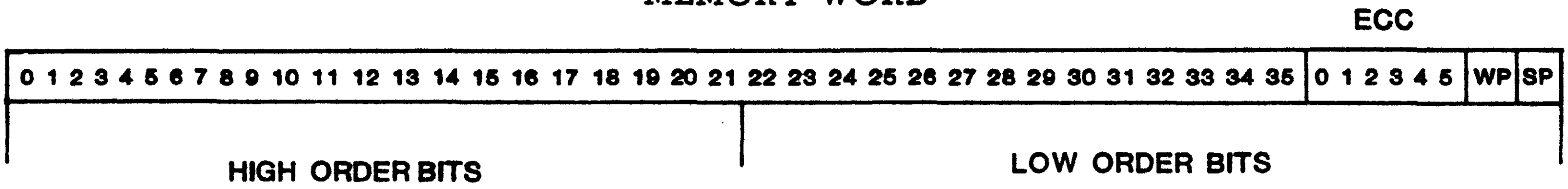
System XXVI

Memory board isolation

SYSTEM XXVI MOS MEMORY BOARDS ARE GROUPED IN PAIRS. ONE BOARD IN A PAIR STORES HIGH ORDER MEMORY WORD BITS 0-21. THE OTHER BOARD IN THE PAIR STORES BITS 22-35, 6 ECC BITS, 1 WORD PARITY BIT AND HAS 1 SPARE BIT. THE BOARDS ARE DESIGNATED 0-15H AND 0-15L FOR HIGH AND LOW ORDER.

THERE ARE 16 PAIRS OF BOARDS POSSIBLE. EACH PAIR STORES 128K of MEMORY WORDS FOR A MAXIMUM OF 2048K OF STORAGE.

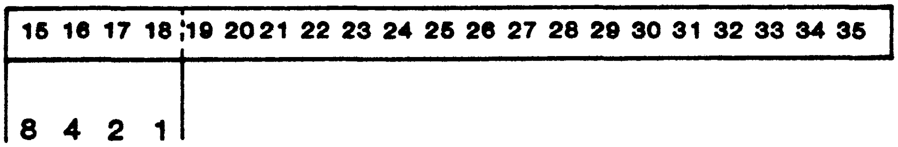
MEMORY WORD



TO ISOLATE A HARD ECC ERROR TO A BOARD PAIR, THE BINARY VALUE OF THE FAILING MEMORY ADDRESS MUST BE EXAMINED. THE SYSTEM XXVI HAS 21 BIT PHYSICAL MEMORY ADDRESSING. BITS 15-18 INDICATE THE FAILING BOARD PAIR.

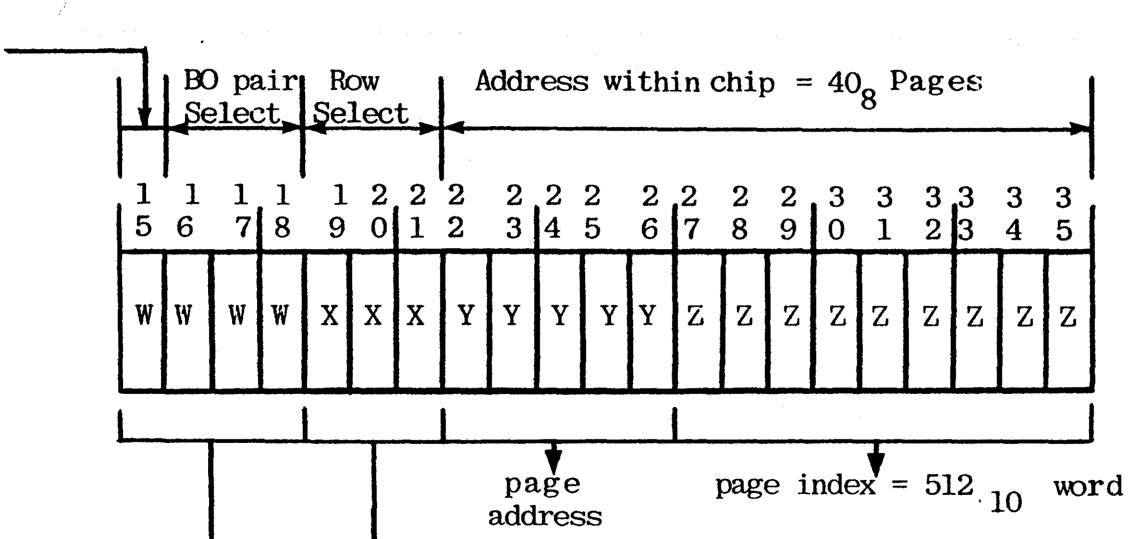
16 K BOARD

MEMORY ADDRESS



EXAMPLE: A BINARY VALUE OF 1010 IN BITS 15-18 WOULD INDICATE BOARD PAIR 10 AS THE FAILING PAIR (BINARY 8+2=10). NOTE THAT COUNT STARTS WITH Ø. THIS WOULD INDICATE THE PAIR OF BOARDS IN SLOTS B06 AND B24.

ONE OF THE BOARDS WOULD THEN BE SWAPPED WITH ANY OTHER BOARD. IF THE FAILING MEMORY ADDRESS REMAINS THE SAME, THEN YOU HAVE MOVED THE GOOD BOARD IN THE FAILING PAIR. IF THE MEMORY ADDRESS CHANGES, THEN YOU HAVE MOVED THE BAD BOARD.



NOTE: USE CHART 2 TO FIND INDIVIDUAL BIT ON BOARD.

BITS 19, 20, 21		ROW
000	=	0
001	=	1
010	=	2
011	=	3
100	=	4
101	=	5
110	=	6
111	=	7

ADDRESS RANGE		BOARD PAIR
0000000-0377777	=	0
0400000-0777777	=	1
1000000-1377777	=	2
1400000-1777777	=	3
2000000-2377777	=	4
2400000-2777777	=	5
3000000-3377777	=	6
3400000-3777777	=	7

16K BOARD

MOS MEMORY CHIP LOCATOR

CHART 1

ADDRESS RANGE	ROW	BOARD PAIR	ADDRESS RANGE	ROW	BOARD PAIR
0000000-0037777 **	0	0	2000000-2037777 **	0	4
0040000-0077777 **	1		2040000-2077777 **	1	
0100000-0137777 **	2		2100000-2137777 **	2	
0140000-0177777 **	3		2140000-2177777 **	3	
0200000-0237777 **	4		2200000-2237777 **	4	
0240000-0277777 **	5		2240000-2277777 **	5	
0300000-0337777 **	6		2300000-2337777 **	6	
0340000-0377777 **	7		2340000-2377777 **	7	
0400000-0437777 **	0	1	2400000-2437777 **	0	5
0440000-0477777 **	1		2440000-2477777 **	1	
0500000-0537777 **	2		2500000-2537777 **	2	
0540000-0577777 **	3		2540000-2577777 **	3	
0600000-0637777 **	4		2600000-2637777 **	4	
0640000-0677777 **	5		2640000-2677777 **	5	
0700000-0737777 **	6		2700000-2737777 **	6	
0740000-0777777 **	7		2740000-2777777 **	7	
1000000-1037777 **	0	2	3000000-3037777 **	0	6
1040000-1077777 **	1		3040000-3077777 **	1	
1100000-1137777 **	2		3100000-3137777 **	2	
1140000-1177777 **	3		3140000-3177777 **	3	
1200000-1237777 **	4		3200000-3237777 **	4	
1240000-1277777 **	5		3240000-3277777 **	5	
1300000-1337777 **	6		3300000-3337777 **	6	
1340000-1377777 **	7		3340000-3377777 **	7	
1400000-1437777 **	0	3	3400000-3437777 **	0	7
1440000-1477777 **	1		3440000-3477777 **	1	
1500000-1537777 **	2		3500000-3537777 **	2	
1540000-1577777 **	3		3540000-3577777 **	3	
1600000-1637777 **	4		3600000-3637777 **	4	
1640000-1677777 **	5		3640000-3677777 **	5	
1700000-1737777 **	6		3700000-3737777 **	6	
1740000-1777777 **	7		3740000-3777777 **	7	

16 K BOARD

CHART 2

AUGMENT ENGINE

Memory board isolation

MOS MEMORY BOARDS ARE GROUPED IN PAIRS. ONE BOARD IN A PAIR STORES HIGH ORDER MEMORY WORD BITS 0-21. THE OTHER BOARD IN THE PAIR STORES BITS 22-35, 6 ECC BITS, 1 WORD PARITY BIT AND HAS 1 SPARE BIT. THE BOARDS ARE DESIGNATED 0-15H AND 0-15L FOR HIGH AND LOW ORDER.

THERE ARE 16 PAIRS OF BOARDS POSSIBLE. EACH PAIR STORES 512K of MEMORY WORDS FOR A MAXIMUM OF 8 MEG OF STORAGE.

MEMORY WORD

		ECC		
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	22 23 24 25 26 27 28 29 30 31 32 33 34 35	0 1 2 3 4 5	WP	SP
HIGH ORDER BITS	LOW ORDER BITS			

TO ISOLATE A HARD ECC ERROR TO A BOARD PAIR, THE BINARY VALUE OF THE FAILING MEMORY ADDRESS MUST BE EXAMINED. THE SYSTEM HAS 23 BIT PHYSICAL MEMORY ADDRESSING. BITS 13-16 INDICATE THE FAILING BOARD PAIR.

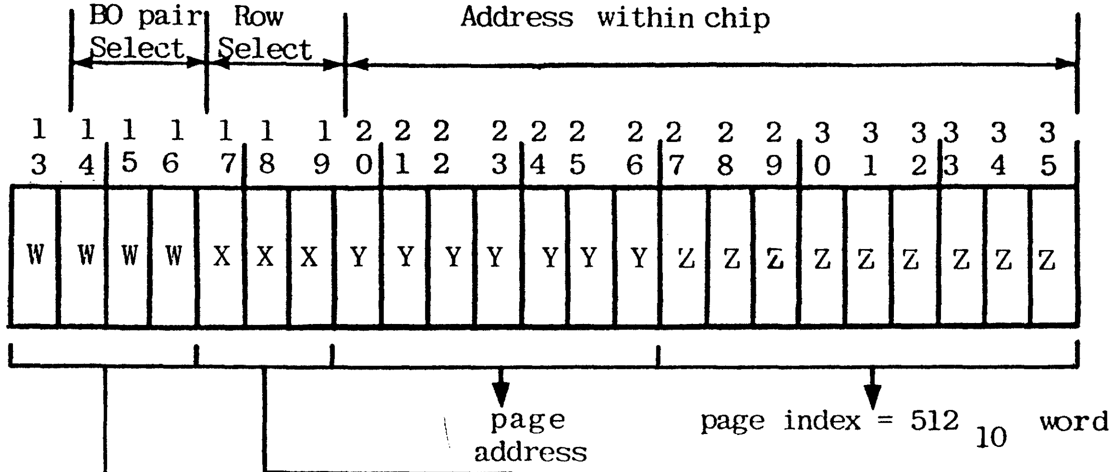
64 K BOARD

MEMORY ADDRESS

13 14 15 16	17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
8 4 2 1	

EXAMPLE: A BINARY VALUE OF 1010 IN BITS 13-16 WOULD INDICATE BOARD PAIR 10 AS THE FAILING PAIR (BINARY 8+2=10). NOTE THAT COUNT STARTS WITH Ø. THIS WOULD INDICATE THE PAIR OF BOARDS IN SLOTS B06 AND B24.

ONE OF THE BOARDS WOULD THEN BE SWAPPED WITH ANY OTHER BOARD. IF THE FAILING MEMORY ADDRESS REMAINS THE SAME, THEN YOU HAVE MOVED THE GOOD BOARD IN THE FAILING PAIR. IF THE MEMORY ADDRESS CHANGES, THEN YOU HAVE MOVED THE BAD BOARD



NOTE: USE CHART 2 TO FIND INDIVIDUAL BIT ON BOARD.

BITS 17, 18, 19		ROW
000	=	0
001	=	1
010	=	2
011	=	3
100	=	4
101	=	5
110	=	6
111	=	7

ADDRESS RANGE		BOARD PAIR
0000000-1777777	=	0
2000000-3777777	=	1
4000000-5777777	=	2
6000000-7777777	=	3

64K BOARD

MOS MEMORY CHIP LOCATOR

CHART 1

ADDRESS RANGE	ROW	BOARD PAIR	ADDRESS RANGE	ROW	BOARD PAIR
00000000-00177777 **	0	0	10000000-10177777 **	0	4
00200000-00377777 **	1		10200000-10377777 **	1	
00400000-00577777 **	2		10400000-10577777 **	2	
00600000-00777777 **	3		10600000-10777777 **	3	
01000000-01177777 **	4		11000000-11177777 **	4	
01200000-01377777 **	5		11200000-11377777 **	5	
01400000-01577777 **	6		11400000-11577777 **	6	
01600000-01777777 **	7		11600000-11777777 **	7	
02000000-02177777 **	0	1	12000000-12177777 **	0	5
02200000-02377777 **	1		12200000-12377777 **	1	
02400000-02577777 **	2		12400000-12577777 **	2	
02600000-02777777 **	3		02600000-02777777 **	3	
03000000-03177777 **	4		13000000-13177777 **	4	
03200000-03377777 **	5		13200000-13377777 **	5	
03400000-03577777 **	6		13400000-13577777 **	6	
03600000-03777777 **	7		13600000-13777777 **	7	
04000000-04177777 **	0	2	14000000-14177777 **	0	6
04200000-04377777 **	1		14200000-14377777 **	1	
04400000-04577777 **	2		14400000-14577777 **	2	
04600000-04777777 **	3		14600000-14777777 **	3	
05000000-05177777 **	4		15000000-15177777 **	4	
05200000-05377777 **	5		15200000-15377777 **	5	
05400000-05577777 **	6		15400000-15577777 **	6	
05600000-05777777 **	7		15600000-15777777 **	7	
06000000-06177777 **	0	3	16000000-16177777 **	0	7
06200000-06377777 **	1		16200000-16377777 **	1	
06400000-06577777 **	2		16400000-16577777 **	2	
06600000-06777777 **	3		16600000-16777777 **	3	
07000000-07177777 **	4		17000000-17177777 **	4	
07200000-07377777 **	5		17200000-17377777 **	5	
07400000-07577777 **	6		17400000-17577777 **	6	
07600000-07777777 **	7		17600000-17777777 **	7	

64 K BOARD

CHART 2

STATUS BIT INTERPRETATION FOR FINDING A BAD BIT VIA THE MONITOR

USE THE LAST 2 OCTAL DIGITS FOR THE CORRECT CORRESPONDENCE

ECC CODE == STATUS BITS -- BAD BIT

(26-MONITOR)

```
00 == 77--NO ERROR
01 == 76--ECC 5
02 == 75--ECC 4
03 == 74--32
04 == 73--ECC 3
05 == 72--27
06 == 71--17
07 == 70--MULTIPLE ERRORS
10 == 67--ECC 2
11 == 66--24
12 == 65--14
13 == 64--35
14 == 63--7
15 == 62--30
16 == 61--20
17 == 60--MULTIPLE ERRORS
20 == 57--ECC 1
21 == 56--22
22 == 55--12
23 == 54--34
24 == 53--5
25 == 52--29
26 == 51--19
27 == 50--MULTIPLE ERRORS
30 == 47--2
31 == 46--26
32 == 45--16
33 == 44--MULTIPLE ERRORS
34 == 43--9
35 == 42--MULTIPLE ERRORS
36 == 41--MULTIPLE ERRORS
37 == 40--MULTIPLE ERRORS
```

```
40 == 37-- ECC 0
41 == 36--21
42 == 35--11
43 == 34--33
44 == 33--4
45 == 32--28
46 == 31--18
47 == 30--MULTIPLE ERRORS
50 == 27--1
51 == 26--25
52 == 25--15
53 == 24--MULTIPLE ERRORS
54 == 23--8
55 == 22--MULTIPLE ERRORS
56 == 21--MULTIPLE ERRORS
57 == 20--MULTIPLE ERRORS
60 == 17--0
61 == 16--23
62 == 15--13
63 == 14--MULTIPLE ERRORS
64 == 13--6
65 == 12--MULTIPLE ERRORS
66 == 11--MULTIPLE ERRORS
67 == 10--MULTIPLE ERRORS
70 == 07--3
71 == 06--NO BOARD RESPONDED
72 == 05--MULTIPLE ERRORS
73 == 04--MULTIPLE ERRORS
74 == 03--10
75 == 02--31
76 == 01--MULTIPLE ERRORS
77 == 00--MULTIPLE ERRORS
```

DESIGN CONSIDERATIONS FOR A DEMAND PAGED ENVIRONMENT:

The Processor of the 26KL has been designed to provide:

1. Efficient program working set management and demand paging.
2. Extensive sharing of data and programs on a page-by-page basis.

These facilities are provided through a combination of micro-code and hardware henceforth referred to as the KL pager. The KL pager performs all virtual to physical address translations. The majority of the pager has been implemented in micro-code to allow future enhancements and improvements in performance to be loaded rather than requiring hardware modification.

Advantages of Paged Memory Management:

- Execution of programs larger than physical core available during execution.
- Improved programmer productivity by removing core size constraints.
- Execution of large jobs during prime time.
- Efficiency reduction only to programs with large core requirements.
- Sharing of information for simultaneous usage on a page-by-page basis by multiple users.
- Development of large core systems for future core size upgrading without physical memory available.
- Efficient use of all system resources.
- Memory protection to small portions which require security.
- Simultaneous operation of batch and interactive processes which may share common data bases.
- Only a portion o the monitor need tie up physical core.
- Three pointer types (Immediate, Shared, and Indirect) enhance the sharing of pages by making two references equivalent.
- Dynamic changes in users address space as size requirements change during execution.

Conquences of KL Style Paging:

- Page tables require 1 full page of 512 page pointers.
- Each page pointer requires 36 bits to represent a physical page reference.
- Reference to any page is inhibited by any of the access bits encountered duting virtual to physical address translation.
- Page pointers are used to address both memory and data which may be stored on some other medium.

SECTIONING OF VIRTUAL MEMORY

Each user's virtual address space consists of 32 equal sections of 256K words per section (512 pages of 512 words per page). A section is represented by one of 32 section pointers found in the User Process Table (UPT). For EXEC sections, the 32 section pointers are in the EXEC Process Table (EPT). The monitor may divide the EXEC address space into per-process and per-job areas through the use of indirect pointers and therefore no such division is built into the pager.

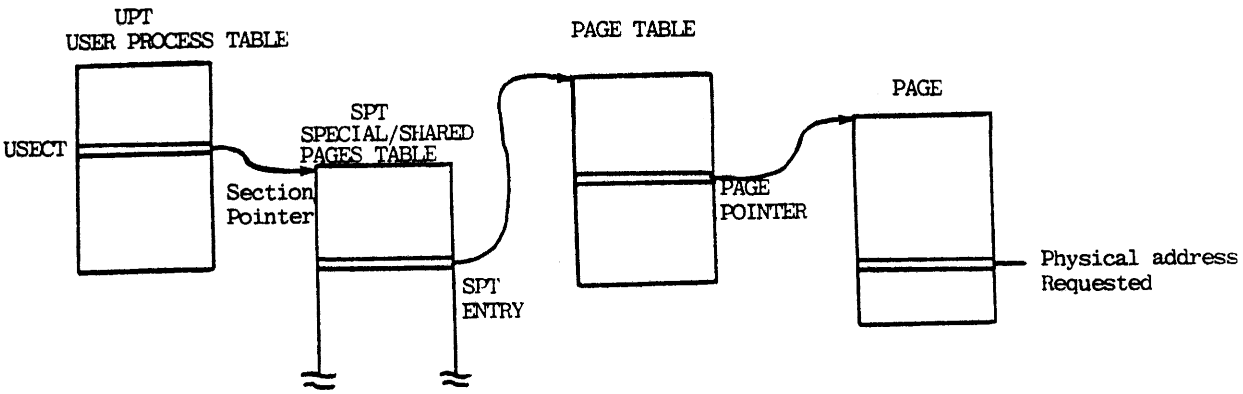
A section pointer will eventually address a page table which represents each page of a 256K virtual address space. The section pointer may be either:

immediate,
shared, or
indirect

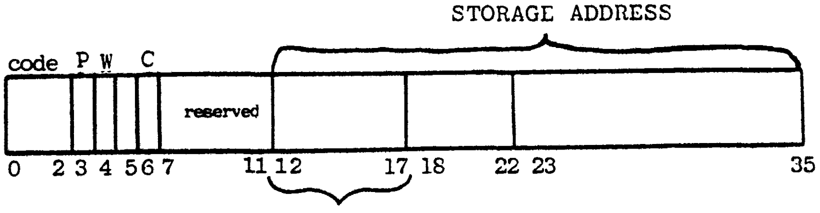
but must yield a physical address of a page table. This page table represents each page of the section.

The format of a page pointer may be found in Figure A and is divided into three parts:

1. Type Code
2. Access Bits
3. Storage Address



PAGE POINTER



B 12-17 = 0 = page in memory
B23-35 - physical page no
B18-22 - must be zero

B 12-17 ≠ 0 = page on disk
address on disk is found in
B18-35

CODE

0	no access
1	immediate
2	shared
3	indirect
4-7	not used, reserved

BIT meaning of 0 in bit position

P = PUBLIC - reference only from concealed or kernel mode
W = WRITE - write references not allowed
C = CACHE - data from page may not be placed in cache

P and C ignored on 26KL

STORAGE ADDRESS

*This example is the most elementary type pf page mapping.
The Section Pointer points through the SPT to a page table.

VIRTUAL TO PHYSICAL PAGE MAPPING

FIGURE A

First follow through the basic pointer interpretation presented in Figure A. It is important at this point to realize only the sequence of steps taken.

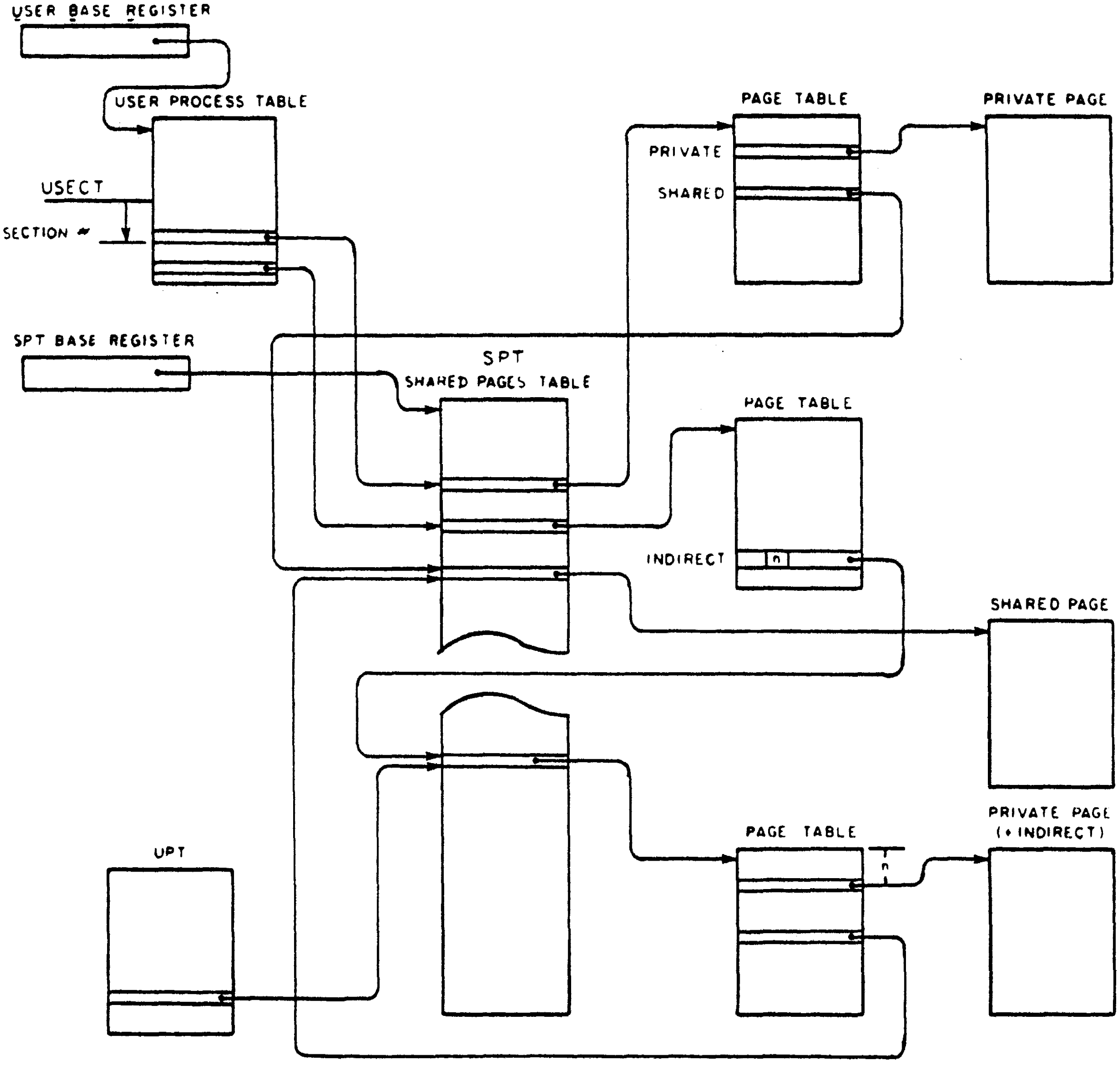
Step 1 Virtual memory reference addresses a section pointer in the User Process Table (UPT) or EPT for EXEC references.

Step 2 The section pointer is used to fetch an entry from the Special/Shared Pages Table. (This is a pointer to a page table).

Step 3 The SPT entry points to a location within a page table representing 512 pages by one page pointer for each page.

Step 4 The page table holds the physical page number required to complete the virtual to physical address mapping.

These steps are a description of the most elementary and immediate type of reference. The complexity of other types requires a discussion of pointer types.



LAYOUT OF KL PAGING

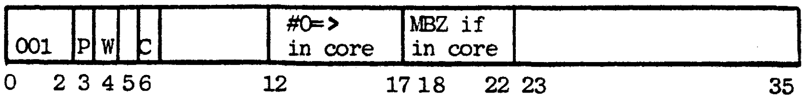
FIGURE B

PAGE POINTER

The pointer type is encoded in bits 0-2 of the page pointer (Refer to Figure A). The types are:

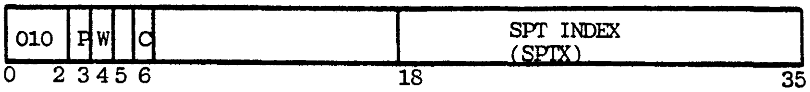
Code	
0	No Access
1	Immediate or Private
2	Shared
3	Indirec
4-7	Not Used, Reserved for future use by DEC

The immediate pointer holds a 13 bit physical page number in bits 23-25. This is also called a private pointer since it is private to the page table containing the pointer. This should not be confused with the Public bit which describes the type of access allowed.



IMMEDIATE POINTER (CODE=1)

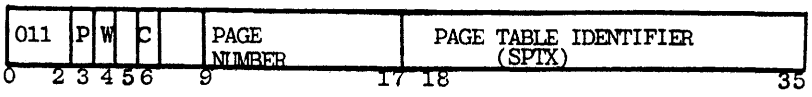
The shared pointer contains an index which addresses into the Special/Shared Pages Table (SPT). The SPT base register, SBR, (reserved AC block) points to the beginning of the SPT. The sum of the SBR and SPT index (SPTX) points to a word containing the storage address of the desired page. The line number from the virtual address is used to complete the reference.



SHARED POINTER (CODE=2)

Regardless of the number of page tables holding a particular shared pointer, the physical address is recorded only once in the SPT. Hence, the monitor may move the page with only one address to update.

The indirect pointer indentifies both another page table and a new pointer within that page table. This allows one page to be exactly equivalent to another page in a separate address space. The object page is located by using the SPT index.

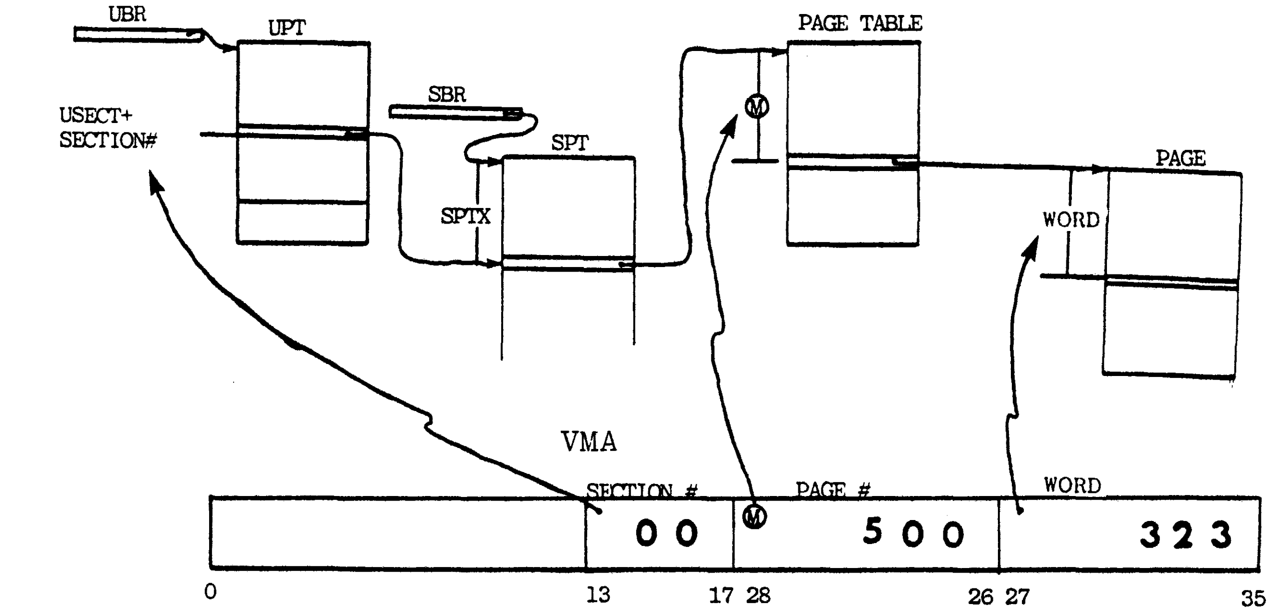


INDIRECT POINTER (CODE=3)

The SPT index, like a shared pointer, allows the physical address of the page to be stored in only one place. If the associated page is in memory, the page number field of the indirect pointer is used to select a new pointer word from the page table. This pointer may be any of the three types or no access and the access bits are ANDed with the access bits of the indirect pointer. The indirect chaining may be arbitrary in depth but is limited by the requirement to service a priority interrupt in the case of long indirect chains or indirect loops.

Study the examples which follow. Remember that TOPS-20 uses shared and indirect section pointers. This complete pointer tracing is performed at each reference. A flow chart is provided in Figure D to aid in working through the examples.

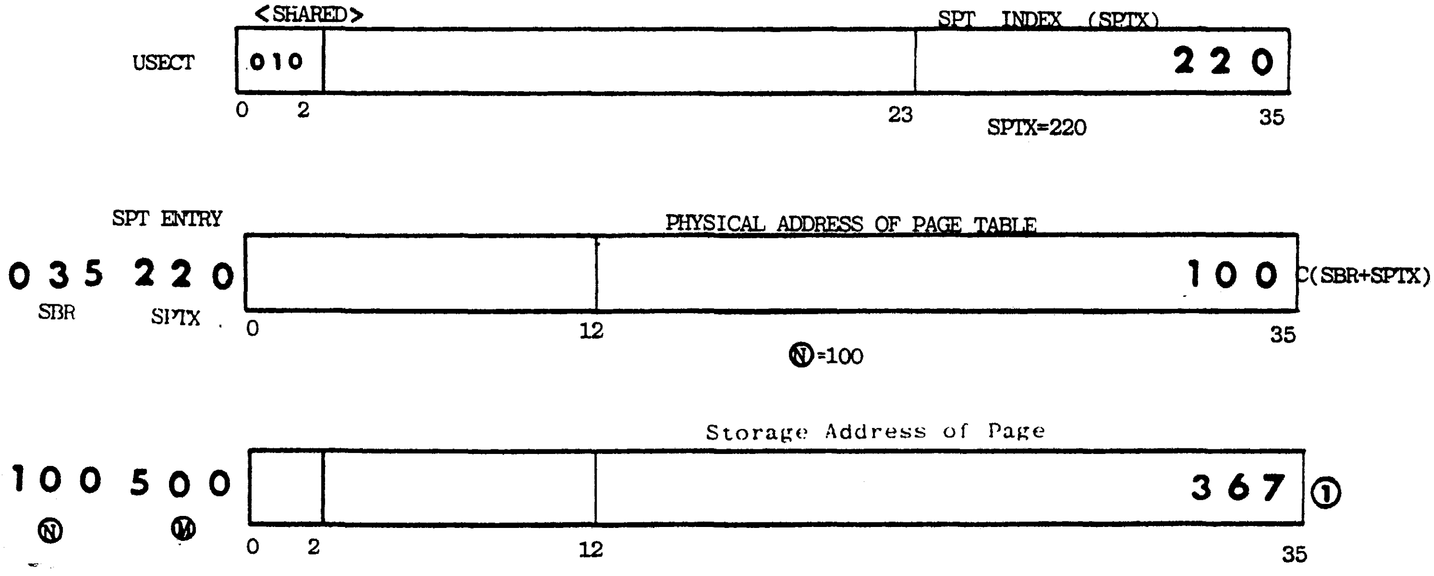
EXAMPLE 1 POINTER INTERPRETATION
(NORMAL SECTION POINTER-SHARED)



VIRTUAL → PHYSICAL

SECTION# = 00 ⟶ SECTION# = 00
PAGE# = 500 -①→ PAGE# = 367
WORD = 323 ⟶ WORD = 323

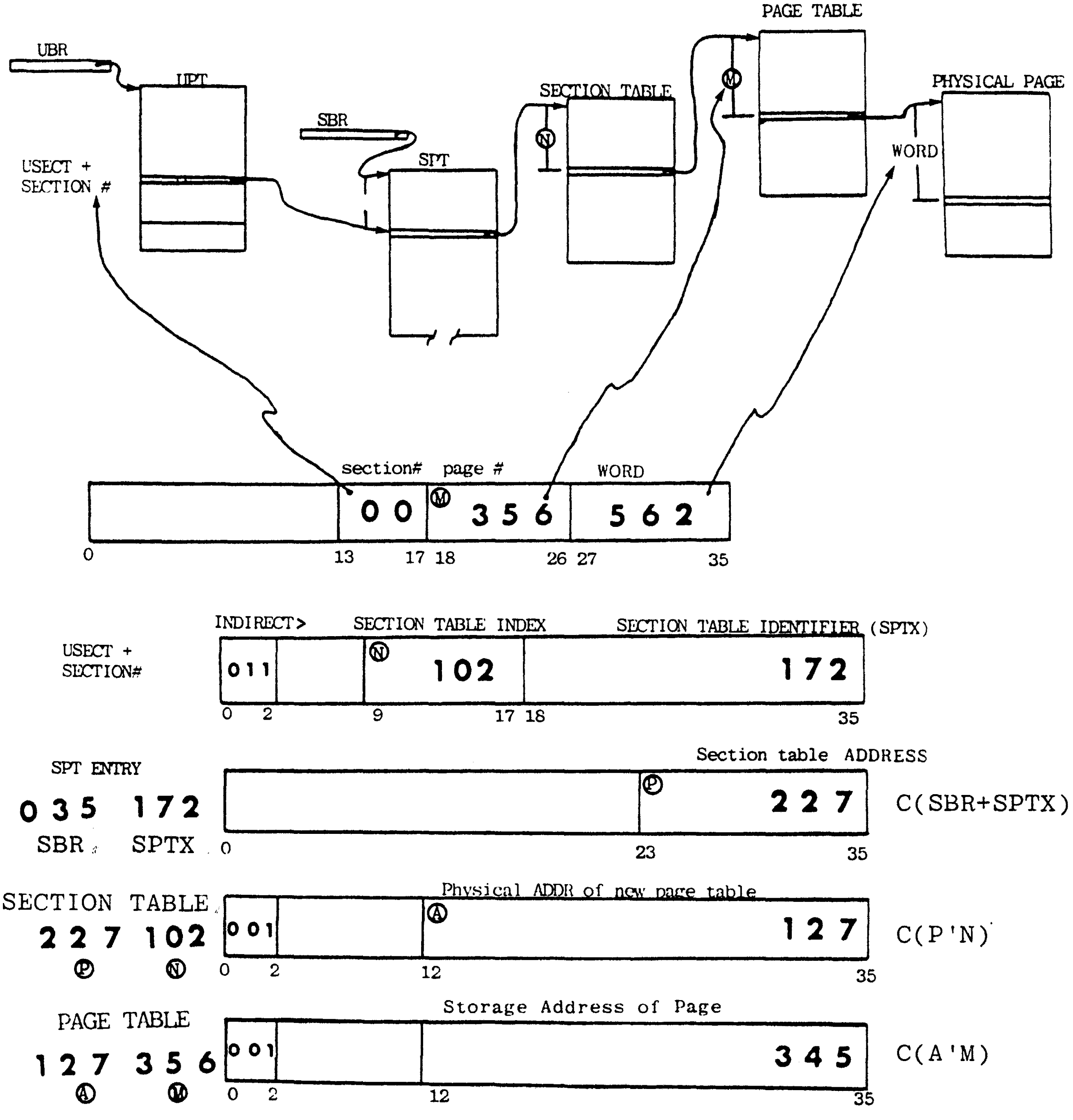
HARDWARE PAGE TABLE
500 =>367



<IMMEDIATE POINTER (CODE = 1)>

C(UBR'USECT + SECTION#) contains Shr PTR with SPTX
C(SBR + SPTX) contains page table page number N
C(N'M) contains Imm PTR with storage address of desired page
(assume page is in core)

EXAMPLE 2 POINTER INTERPRETATION (INDIRECT SECTION POINTER)



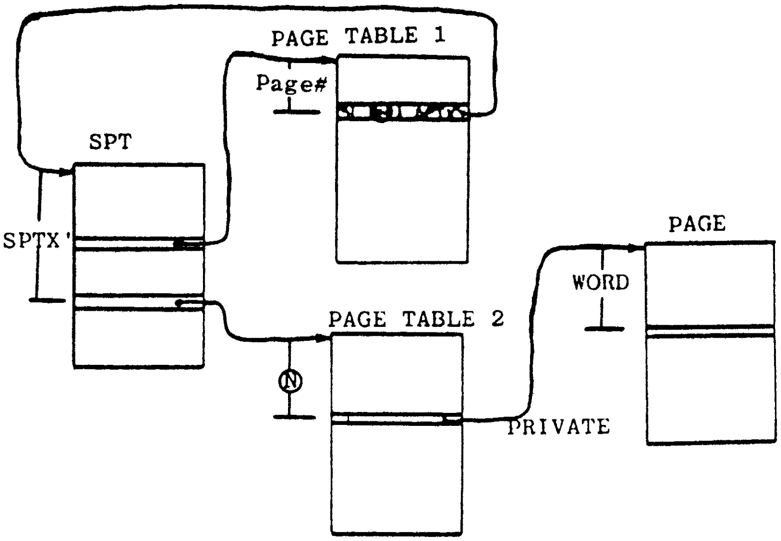
VMA   PMA

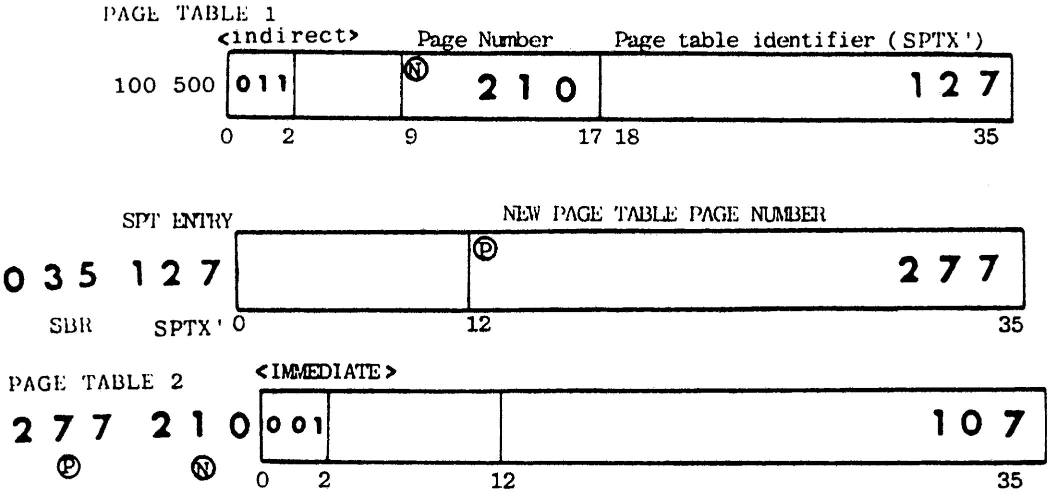
0 0 3 5 6 5 6 2 = 0 0 3 4 5 5 6 2

(Assume Page in Core)

EXAMPLE 3 POINTER INTERPRETATION (INDIRECT PAGE POINTER)

See EXAMPLE 1 Page table pointer now indirect instead of immediate. From example 1 the UPT addressed page table 1. Now since page table pointer is indirect, we go back through the SPT again for a new page table 2.





V M A

00500 323

P M A

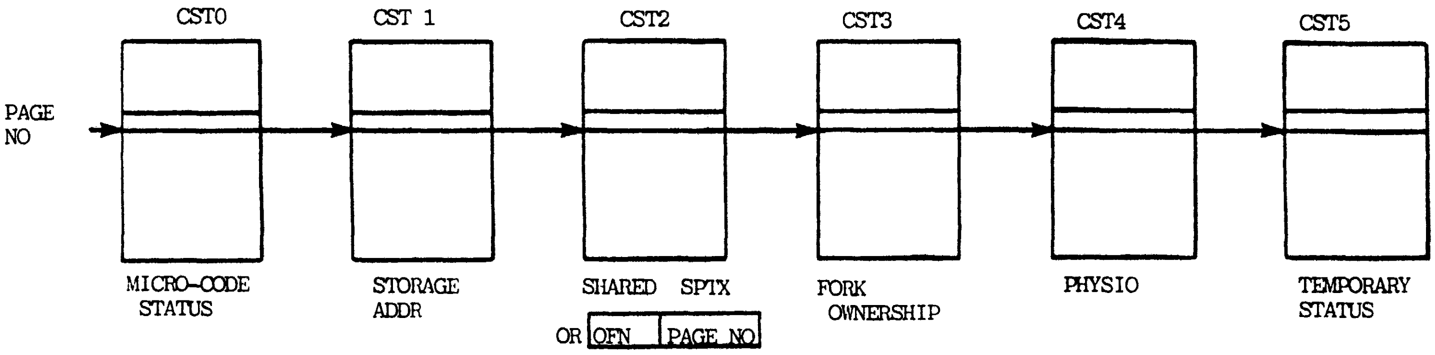
00 107 323

which is now equivalent to a VMA of

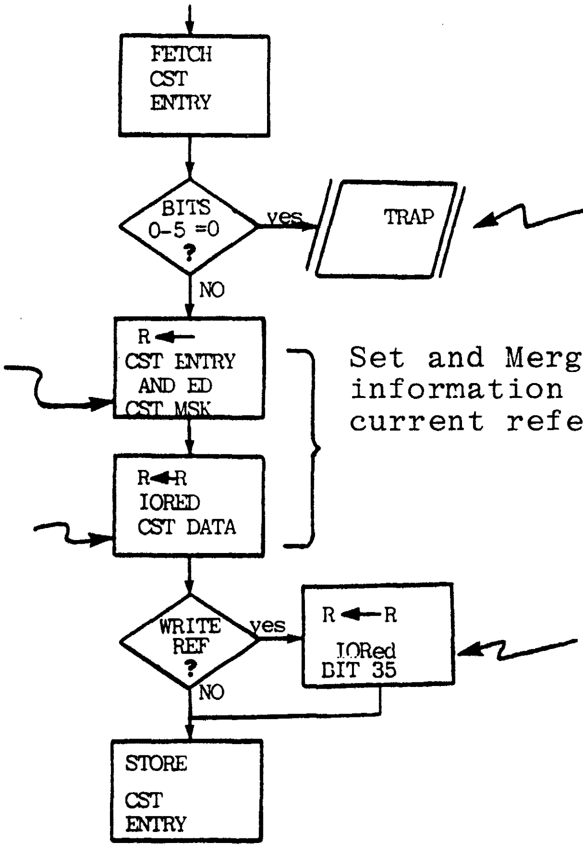
00 210 323
Ⓝ

with SPTX' = 127

(assume page in core)



CORE STATUS TABLES
(Each addressed by page number)



Page is inaccessible

AC Block 6
Word 0

Set and Merge
information fields about
current reference

AC Block 6
Word 1

Set Modified Bit

CST ENTRY UPDATING

Figure C KL Core Status Tables

SPECIAL/SHARED PAGES TABLE (SPT)

The SPT contains the physical addresses of pages which are shared by many page tables or of pages used in a special way, e.g., as page tables. They are stored in one common location to allow modification to these pages by changing a single entry. The SPT index is added to the SPT base address to form a physical address of the associated entry.

(SPT Base Address=reserved AC block 6 word 3)

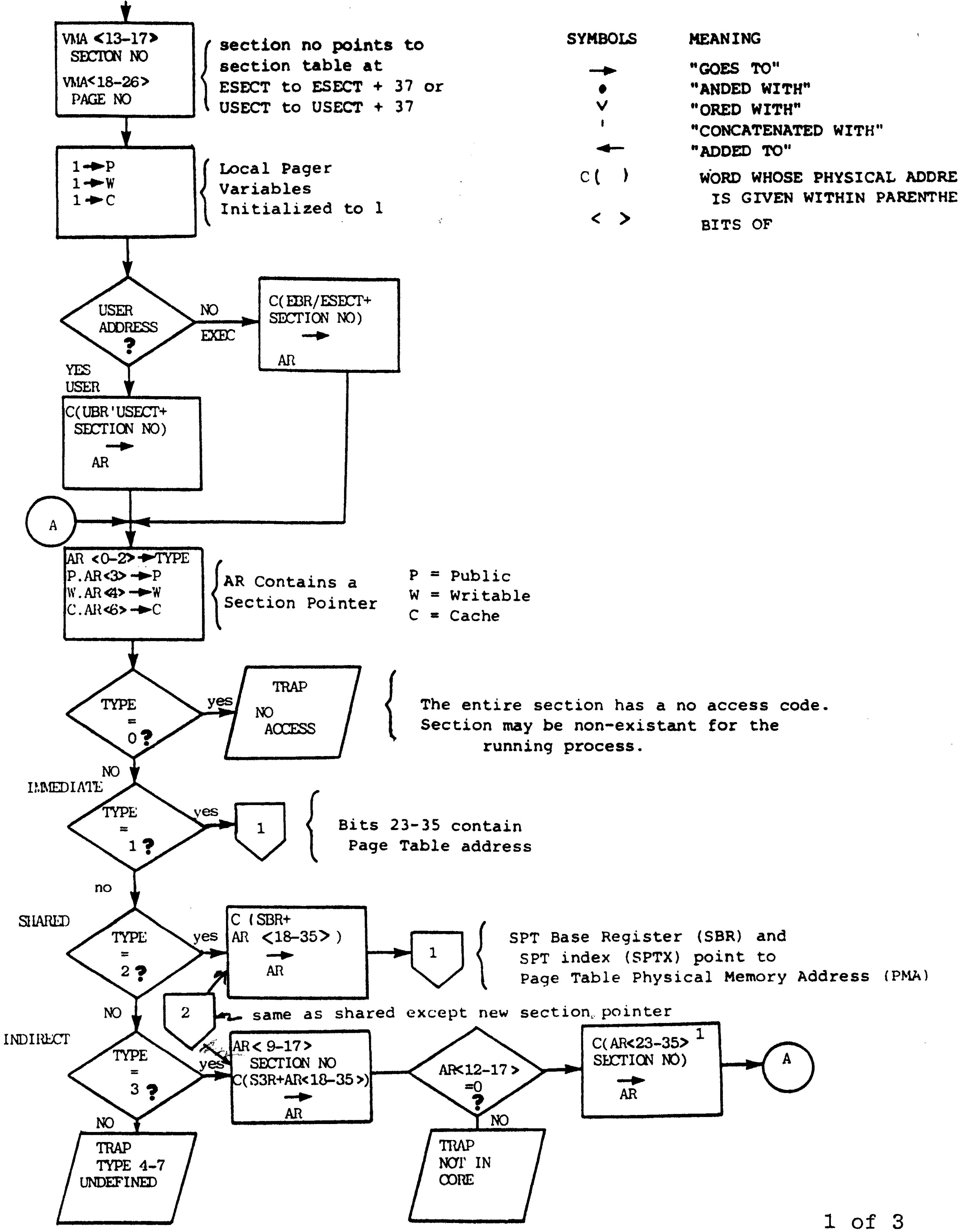
NOTE: Under TOPS-20 each section pointer points to a page table and as such it requires a special/shared pages tables entry. The virtual to physical address translation from the UPT must therefore include the SPT translation process.

CORE STATUS TABLE (CST)

Virtual Memory management requires information about memory references generated by each user's processes. Adding the CST base register (AC block 6,word 2) to the physical page number from a storage address allows the monitor to address and update information regarding the page reference. Figure C shows the flow of updating and using a CST Entry. This allows pages to be ordered by "age" (time of last reference) and classified by the type of process referencing the page. The reference indication is carried by assigning one bit to each active process. By placing a 1 in that bit position in the pager data word, when a reference is made, the one is placed in the CST word in the bit position assigned to the process making the reference. The Modified Bit (B35) will be set if the page is modified. This allows the monitor to eliminate the swapping out of pages to which only read references are made.

Here when Paging Cache Requires translation information

POINTER INTERPRETATION FLOW
FIGURE D



SYMBOLS	MEANING
→	"GOES TO"
•	"ANDED WITH"
V	"ORED WITH"
'	"CONCATENATED WITH"
←	"ADDED TO"
C( )	WORD WHOSE PHYSICAL ADDRESS IS GIVEN WITHIN PARENTHESES
< >	BITS OF

section no points to section table at ESECT to ESECT + 37 or USECT to USECT + 37

Local Pager Variables Initialized to 1

AR Contains a Section Pointer

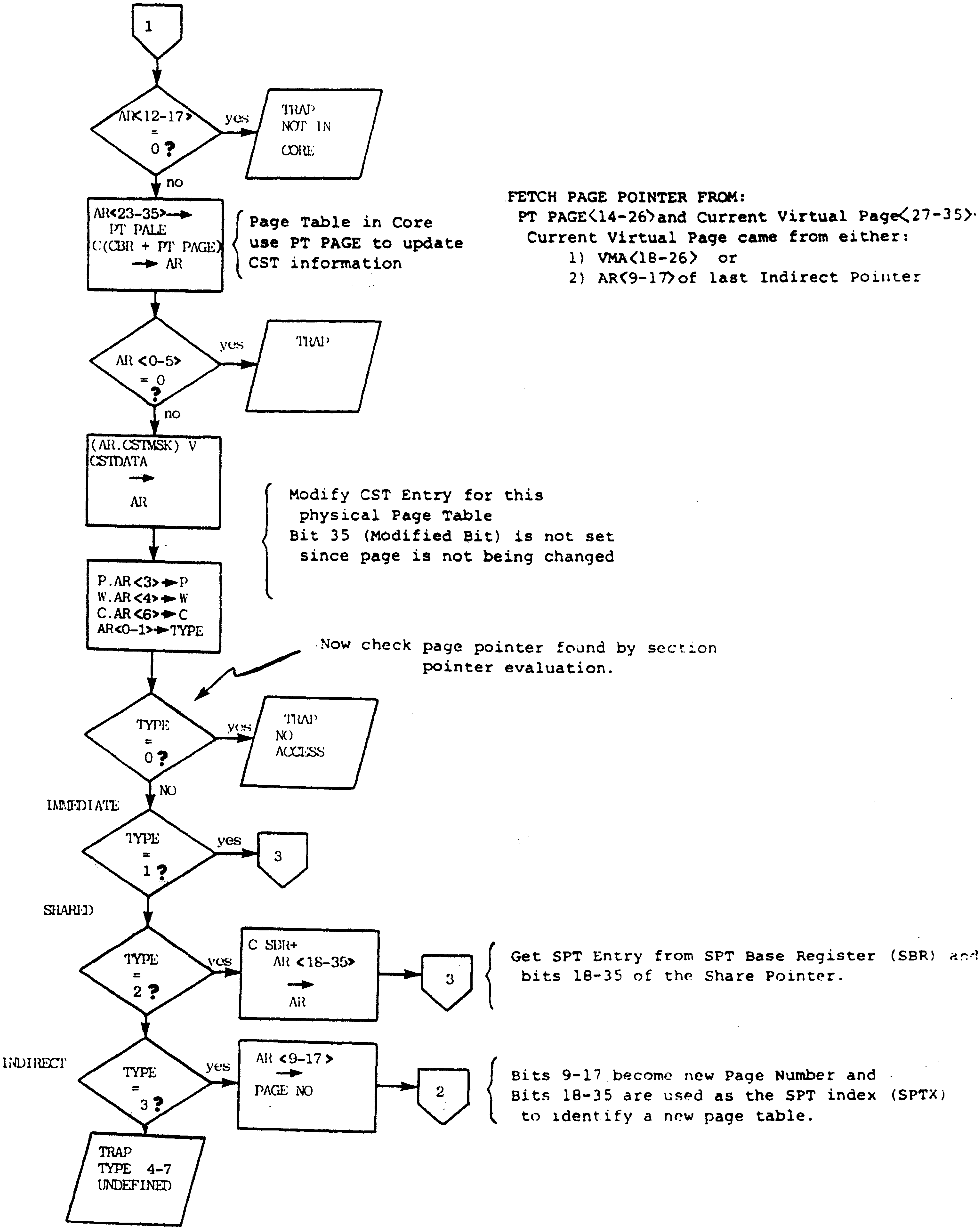
P = Public
W = Writable
C = Cache

The entire section has a no access code. Section may be non-existant for the running process.

Bits 23-35 contain Page Table address

SPT Base Register (SBR) and SPT index (SPTX) point to Page Table Physical Memory Address (PMA)

same as shared except new section pointer



Page Table in Core use PT PAGE to update CST information

FETCH PAGE POINTER FROM:
PT PAGE<14-26>and Current Virtual Page<27-35>.
Current Virtual Page came from either:
1) VMA<18-26> or
2) AR<9-17>of last Indirect Pointer

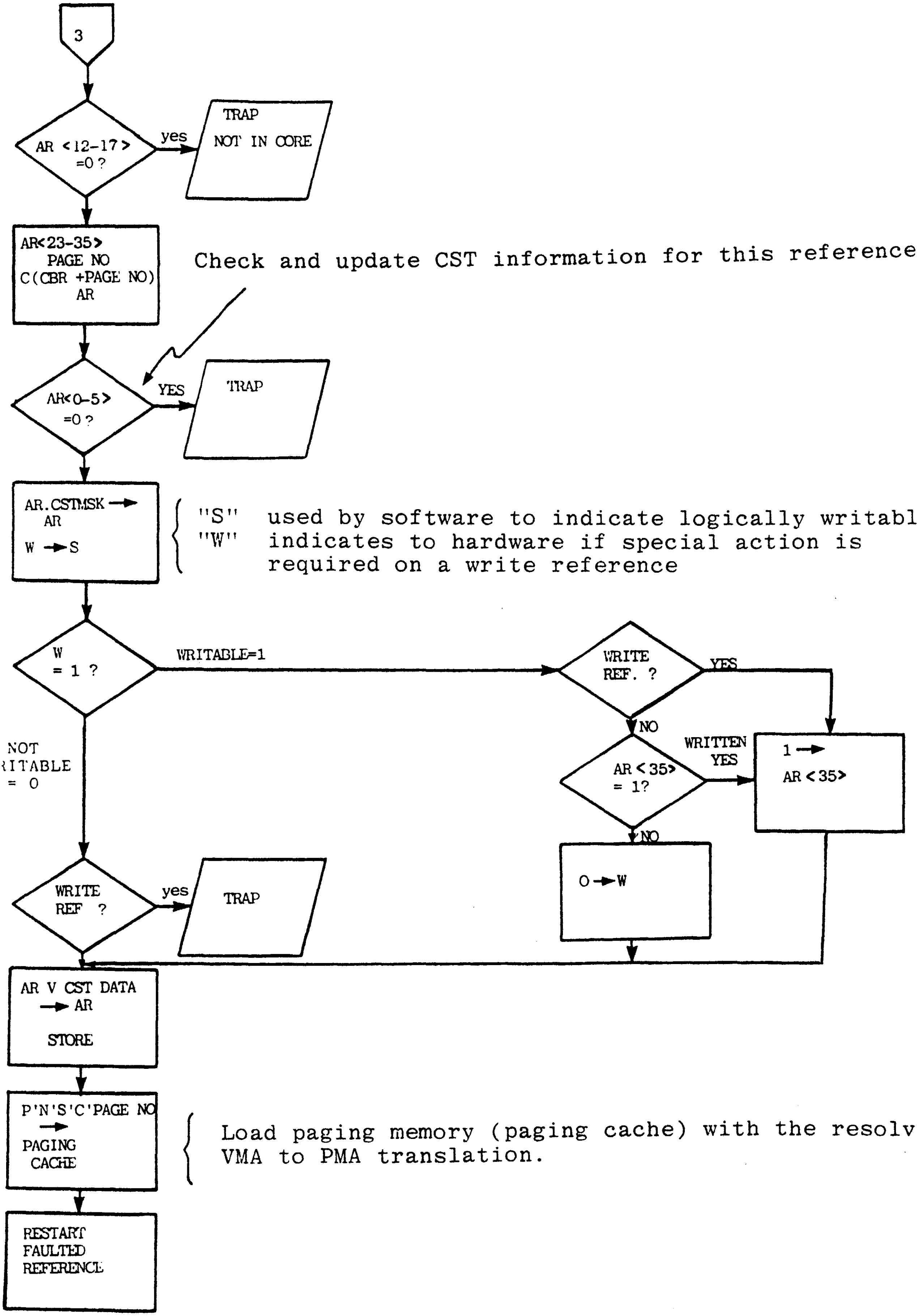
Modify CST Entry for this physical Page Table
Bit 35 (Modified Bit) is not set since page is not being changed

Now check page pointer found by section pointer evaluation.

Get SPT Entry from SPT Base Register (SBR) and bits 18-35 of the Share Pointer.

Bits 9-17 become new Page Number and
Bits 18-35 are used as the SPT index (SPTX) to identify a new page table.

Figure D 2 of 3



Check and update CST information for this reference

"S" used by software to indicate logically writable
"W" indicates to hardware if special action is required on a write reference

Load paging memory (paging cache) with the resolved VMA to PMA translation.

HARDWARE SUPPORT FOR PAGING

The paging hardware is designed to be transparent to the user. All memory both virtual and physical in both user and monitor space is divided into pages. The two main justifications for a paged monitor are:

1. To protect the monitor from unwarranted modifications and

2. To allow efficiency in memory usage by requiring only a portion of the monitor to be core resident.

The virtual address consists of 18 bits; 9 high order bits for virtual page number and 9 low order bits (word number) which addresses the location within the page. The virtual page number is first used as an index into a hardware page table. This hardware page table contains up to 512 direct virtual to physical address translations. (See Cached Paging Data). If the 13 bit physical address is found in the hardware page table, a 22 bit physical address is formed by concatenating this 13 bit physical address with the 9 bit line number. If the entry does not exist in the hardware page table, a sequence of translations are initiated to locate a page table in memory which contains a physical address for the virtual page if one exists.

The number 512 in the above statement is for the KL. For the 26KL that value would change to 4 K.

INSTRUCTIONS RELEVANT TO PAGING (PAG device 10 octal)

1. Load UBR (DATAO PAG,)
   Sets up the paging hardware according
   to the contents of E.

If B2=1 loads the physical page number in which the processor will find the User Process Table.

2. Load EBR (CONO PAG,)
   This is an immediate instruction. It
   always clears the hardware page table.

Bit 21 determines the type of paging to be used, however on the 26 KL it will always be KL. Bits 23-25 High Order 13 bits of the physical address of the Executive Process Table are loaded here.

3. Clear paging Memory (DATAO PAG, CONO PAG,)
   First set the UBR then clear the hardware
   page table.

4. Clear Paging Memory entry for monitor Virtual
   Address E (CLRPT E)

5. Paging ON/OFF (CONO PAG,)
   Set Bit 22 to 1 to turn on traps and paging.
   When 0 traps are not permitted and all addresses
   are physical.

6. Deposit values for paging constants into
   locations in reserved AC block (SPT base
   address; CST base address; CSTMSK: CSTDATA).

7. Hardware Page Table Invalidate (BLKO PAG,)
   The half word E is interpreted as follows:

Bit	Symbol	Description
18	UPI	1 =>invalidate a user page 0 =>invalidate an executive page
19-21	MBZ	Must be zero
23-35	VPN	Virtual Page Number whose entry in the page table will be invalidated.

8. BLKI PAG,E

   E = special 26KL functions see recode listings.

CACHED PAGING DATA

The hardware page table referred to at the beginning of this section is effectively a cache of paging data which has been accumulated by previously fetching this data from memory or by previous pointer interpretation. A virtual address is first checked against the current contents of this hardware pager and if found returns without delay a physical address. (This function was implemented by associative memory on the KI). If the physical address is not found, the pointer interpretation flow chart in Figure D fetches information from memory to resolve the virtual address. When completed, this translation may be placed in the hardware page table forming the cache of recently used page addresses.

This hardware page table is handled by the micro-code in KI style. It should not be confused with the memory cache of a 1080 system, which is normally referred to as the "cache". the paging cache is implemented as 4 K entries, one for each page of a users virtual address space. The EXEC and USER share the same 4 K entries but are offset from each other. Therefore, the paging cache at any given time will hold translation information about most of the active pages. It cannot be guaranteed that the 4 K most recently used pages will be addressed by the paging cache but the least recently used page will always be there.

When the monitor takes any action which would invalidate information about existing virtual-to-physical address translation, the paging cache must be either partially or completely cleared.

Examples of such instances are:

1. Change of User Process - Clear entire paging memory (entire user address space has changed).

2. One Page Removed from core - Clear entire paging mamory cause -(several Shared and Indirect pointers may have used the page).

3. Pointer is Removed from UPT - clear entire paging memory (association for many pages through UPT is changed).

4. Monitor Mapped Page to EXEC space for local use - only one entry cleared (when page is unmapped only that one pointer must be cleared. Since this facility is provided by the pager, it may be used to reduce reload overhead).

If the paging data is not found, the flow in Figure D must be followed. A special trap is initiated and the micro-code saves vulnerable EBox data before starting on the pointer tracing algorithm. If the Algorithm is successful, the resolved pointer and associated information are loaded into the paging memory, the EBOX registers are restored, and the memory request is again issued. The micro-code must also handle the first write request trap, inhibiting the write until the modified bit can be set (see Core Status Table). The pager must maintain this modified bit. The micro-code implements this as follows:

> On a paging memory reload, the write access (W) bit is set in the paging memory only if the current memory reference is a write (and a write is legal for the page). Thus if the first reference to a page is read, the W bit in the corresponding paging memory entry will be set to 0, and a subsequent write reference will cause another trap to the micro-code. On this second trap, the pointer interpretation will be repeated and the paging memory reloaded, this time with the W bit set.

BASE REGISTERS AND PROCESS TABLES (see Figure E)

The User Base Register (UBR) and Executive Base Register (EBR) are used to locate the User Process Table (UPT) and Executive Process Table (EPT). These base registers are privileged registers which contain the physical page number of the UPT or EPT in bits 14-26.

The UPT differs from the KI in the following locations:

Location	Usage KL	Usage KI
420	User & Exec Page Fail Word	User page failure trap instruction
426	PC Word of Page Fail	EXEC Page Fail Word
427	Page Fail New PC Word (eliminates usage of any trap instruction upon page fail)	User Page Fail Word
500	Prev. Context Section and CWSX	

The EPT differs from the KI in locations 0-0137 and 60-177.

Locations 0-037 are used for built-in channels with a quad-word block reserved for each of the eight channels.

Word	Definition
0	Internal Channel Command Word (CCW)
1	Error Status and Command List Pointer on errors
2	Last Word Count and Address
3	Reserved for future use by DEC

Locations 60-77 compries a block of 4 quad-words one for each DTE20. The block contains an "in byte pointer", "out byte pointer", an "interrupt vector" and one reserved for future use. The KI allows software to use locations 60-177. The KL reserves the use of locations 100-177 for future hardware usage.

Further information on the UPT and EPT will be givin in Book 5 Monitor.

Summary of Paging Word Formats (see Figure F)

PAGE FAIL DATA

The following information is made available upon a page fail trap:

1. Virtual Address of Reference
2. User bit, Public bit, Write Reference bit
3. Page Fail Code

A Page Failure code will be stored by any of the following conditions:

1. Proprietary Violation
2. Refill Error
3. Address Compare
4. Page Table parity error
5. Memory Data parity error

All other Page Fail conditions cause only address, User, Public and Write data to be stored for software interpretation.

Figure E TOPS–20 Process Table Configuration

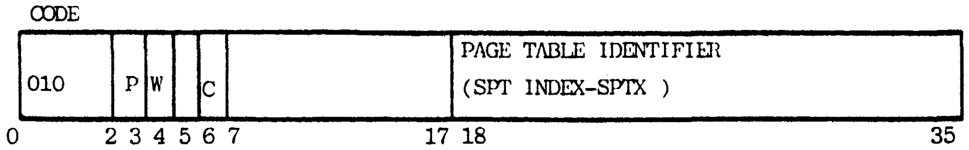
Extended USER PROCESS TABLE

0 417	RESERVED	NOTE: ASTERISKS INDICATE LOCATIONS WHOSE USE DIFFERS FROM THOSE IN THE SINGLE-SECTION PROCESS TABLE LISTED ON THE NEXT PAGE.	
420	ADDRESS OF LUUO BLOCK		•
421	USER ARITHMETIC OVERFLOW TRAP INSTRUCTION		
422	USER STACK OVERFLOW TRAP INSTRUCTION		
423	USER TRAP 3 TRAP INSTRUCTION		
424	MUUO FLAGS	MUUO OP CODE, A	•
425	MUUO OLD PC		•
426	E OF MUUO		•
427	MUUO PROCESS CONTEXT WORD		
430	KERNEL NO TRAP MUUO NEW PC		•
431	KERNEL TRAP MUUO NEW PC		•
432	SUPERVISOR NO TRAP MUUO NEW PC		•
433	SUPERVISOR TRAP MUUO NEW PC		•
434	CONCEALED NO TRAP MUUO NEW PC		•
435	CONCEALED TRAP MUUO NEW PC		•
436	PUBLIC NO TRAP MUUO NEW PC		•
437	PUBLIC TRAP MUUO NEW PC		•
440 477	RESERVED		
500	PAGE FAIL WORD		•
501	PAGE FAIL FLAGS		•
502	PAGE FAIL OLD PC		•
503	PAGE FAIL NEW PC		•
504 505	USER PROCESS EXECUTION TIME		
506 507	USER MEMORY REFERENCE COUNT		
510 537	RESERVED		
540 577	USER SECTION 0 USER SECTION 37		
600 777	RESERVED		

EXECUTIVE PROCESS TABLE

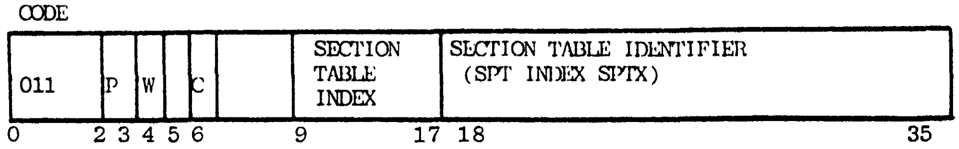
0 37	EIGHT CHANNEL LOGOUT AREAS EACH: 0 INITIAL CHANNEL COMMAND 1 GETS CHANNEL STATUS WORD 2 GETS LAST UPDATED COMMAND 3 RESERVED
40 41	RESERVED
42 57	STANDARD PRIORITY INTERRUPT INSTRUCTIONS
60 63	FOUR CHANNEL BLOCK FILL WORDS
64 137	RESERVED
140 177	FOUR DTE20 CONTROL BLOCKS
200 420	RESERVED
421	EXECUTIVE ARITHMETIC OVERFLOW TRAP INSTRUCTION
422	EXECUTIVE STACK OVERFLOW TRAP INSTRUCTION
423	EXECUTIVE TRAP 3 TRAP INSTRUCTION
424 507	RESERVED
510 511	TIME BASE
512 513	PERFORMANCE ANALYSIS COUNT
514	INTERVAL COUNTER INTERRUPT INSTRUCTION
515 537	RESERVED
540 577	EXECUTIVE SECTION 0 EXECUTIVE SECTION 37
600 777	RESERVED

SECTION POINTER

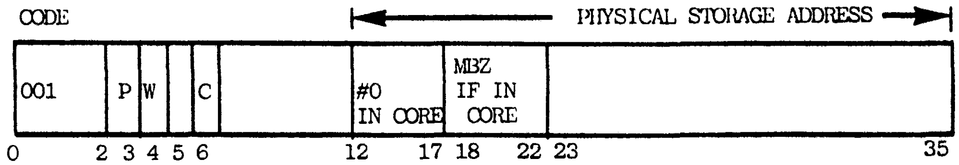


NORMAL SECTION POINTER (CODE=2)

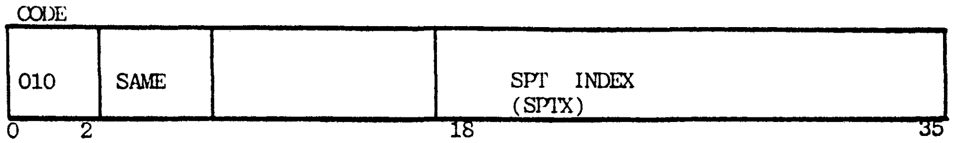
PAGE POINTERS



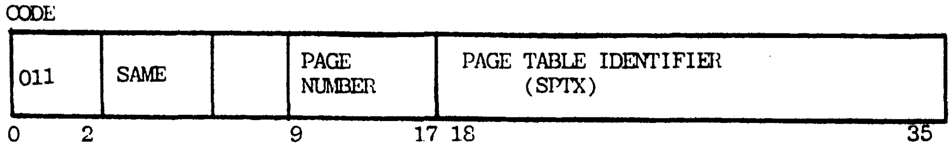
INDIRECT SECTION POINTER (CODE=3)



IMMEDIATE POINTER (CODE =1)

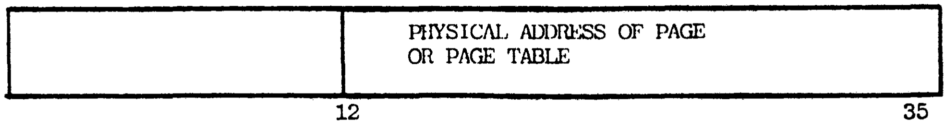


SHARED POINTER (CODE=2) SBR+SPTX=PHYSICAL ADDRESS OF WORD HOLDING ADDRESS OF PAGE

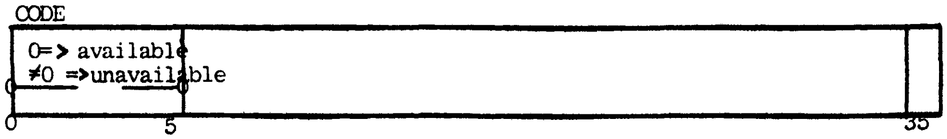


INDIRECT POINTER (CODE=3) PAGE NUMBER IS NEW DISPLACEMENT INTO ANOTHER PAGE TABLE

SPT ENTRY



CORE STATUS TABLE ENTRY

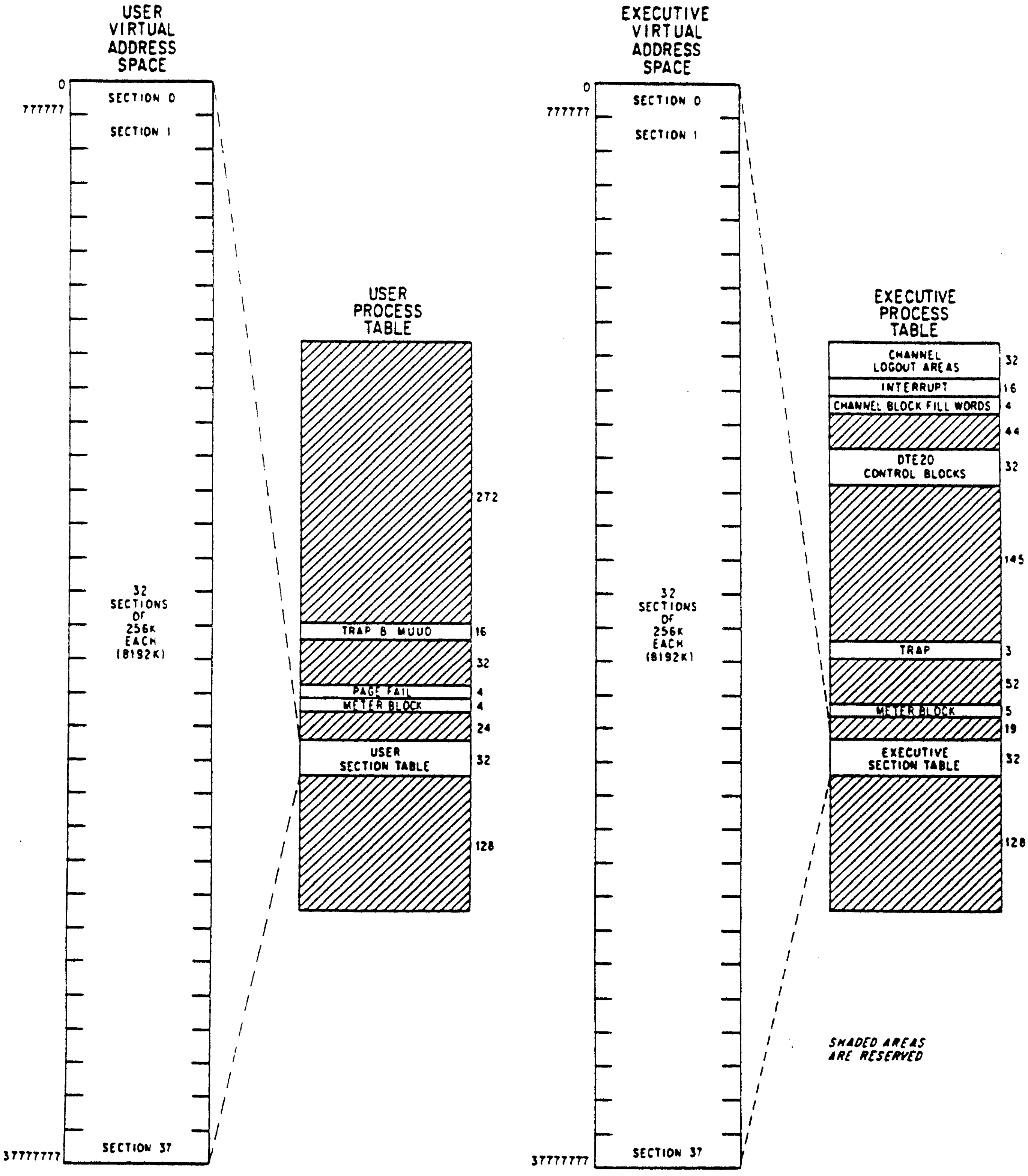


All CSTs are indexed into by the page number.

SUMMARY
PAGING WORD FORMATS
FIGURE F

B 32-34 reserved by DEC·
B 35 modified bit.
All unused bits are reserved for future use by DEC

Figure G TOPS–20 Virtual Address Space and Process Table Layout



KL PAGING

REF.

1080, 2040, 2060 ENGINEERING FUNCTIONAL SPEC.
-CHAP 2.8 BY DEC
TITLE: KL10 PAGING-REV 2

DECSYSTEM 20 UPDATE COURSE
STUDENT GUIDE
BOOK 4: KL HARDWARE

KL10 SYSTEM OPERATIONS
SECTION 3
PAGES 3-20 THROUGH 3-43

HISTORY REF
A DOCUMENT OF THE KI10
THE KI10 AS A PAGING MACHINE

MAP Problems - Card MP5

ADDRESS ENGINE

The address engine is the logic that furnishes memory addresses for MOS MEMORY. These addresses must be put through the "MAP" and converted to physical memory addresses (PMA). Normally the addresses are used for fetching instructions, fetching data for an instruction, or storing data for an instruction.

The Map also has another input from the TYBUS that supplies memory addresses for "DMA" transfers on the TYBU~~S.~~ F-BUS

All logic is based on the maximum use of MOS Memory. The memory has five basic types of cycles.

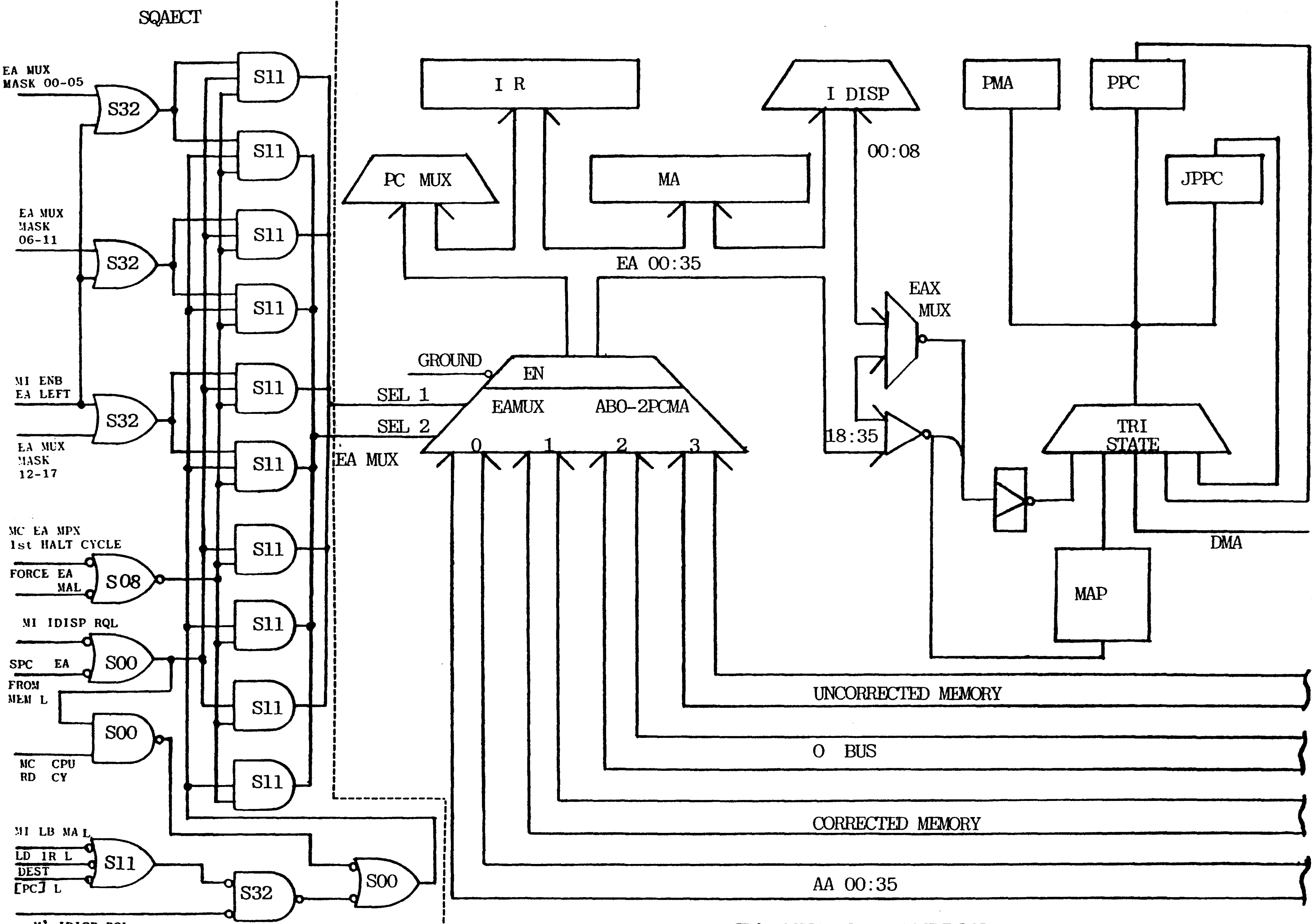
Each cycle type has a priority of it's own. Zero is the highest priority.

Memory Cycle Types ....

```
Priority ----Cycle Type

    0        Refresh             * Special Cycles.  CPU cannot use
                                 *
                                 *

    1        DMA read            * Memory during these cycles

                                 *
             DMA Write

    2        CPU Read-----------Data Fetch and instruction Fetch
             CPU Write----------Data Store
```

Cycle TYPE address source.................

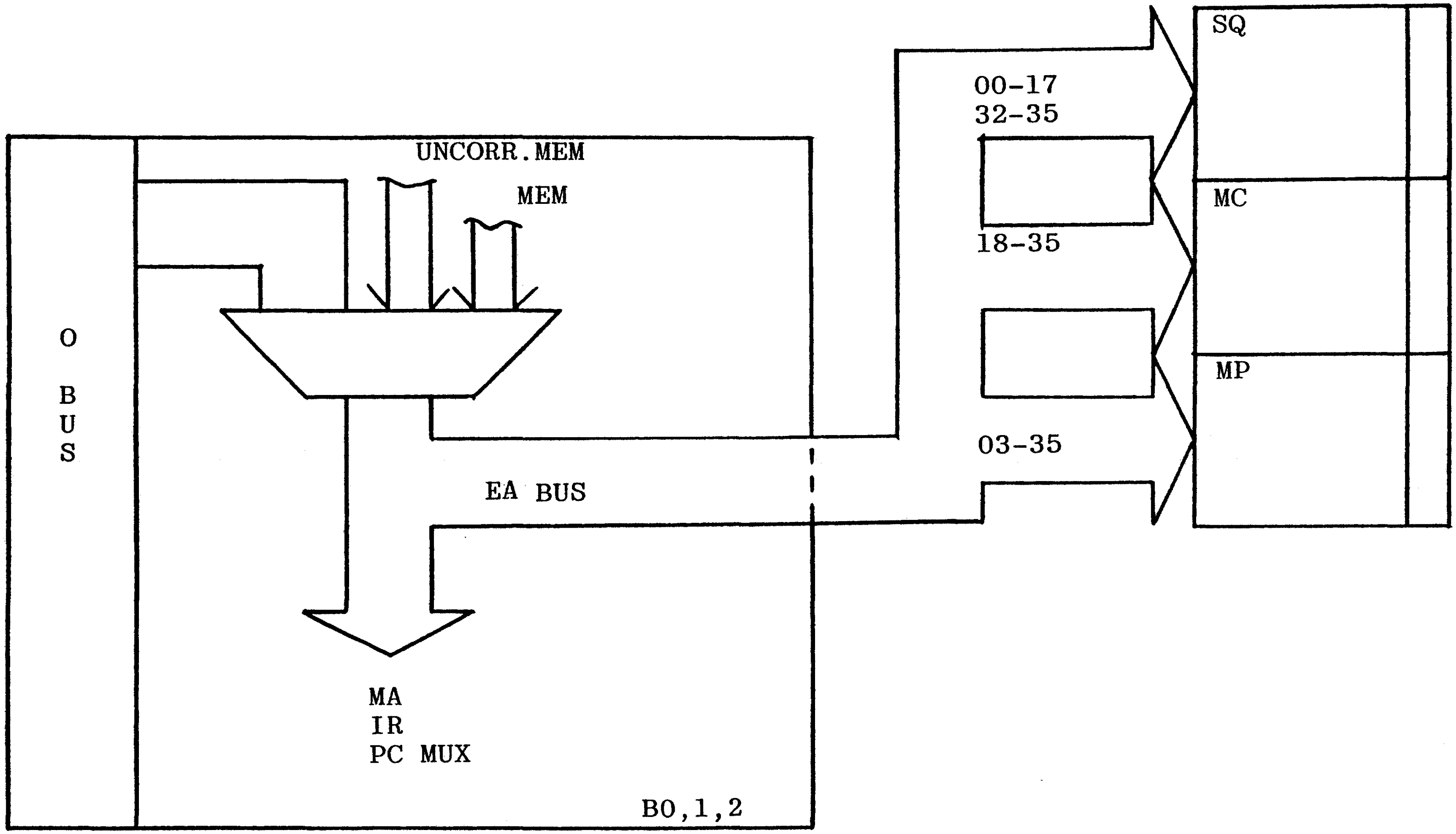
```
TYPE         ADDRESS SOURCE
****         **************

REFRESH------GENERATED BY INTERNAL MEMORY HARDWARE

DMA READ-----TYBUS I/O ADDRESS LINES

DMA WRITE----TYBUS I/O ADDRESS LINES

CPU READ
    (DF)-----EA MUX/MAP
    (IF)-----PPC OR JPPC

CPU WRITE
    (DS)-----EA MUX/MAP (DATA STORE)
```

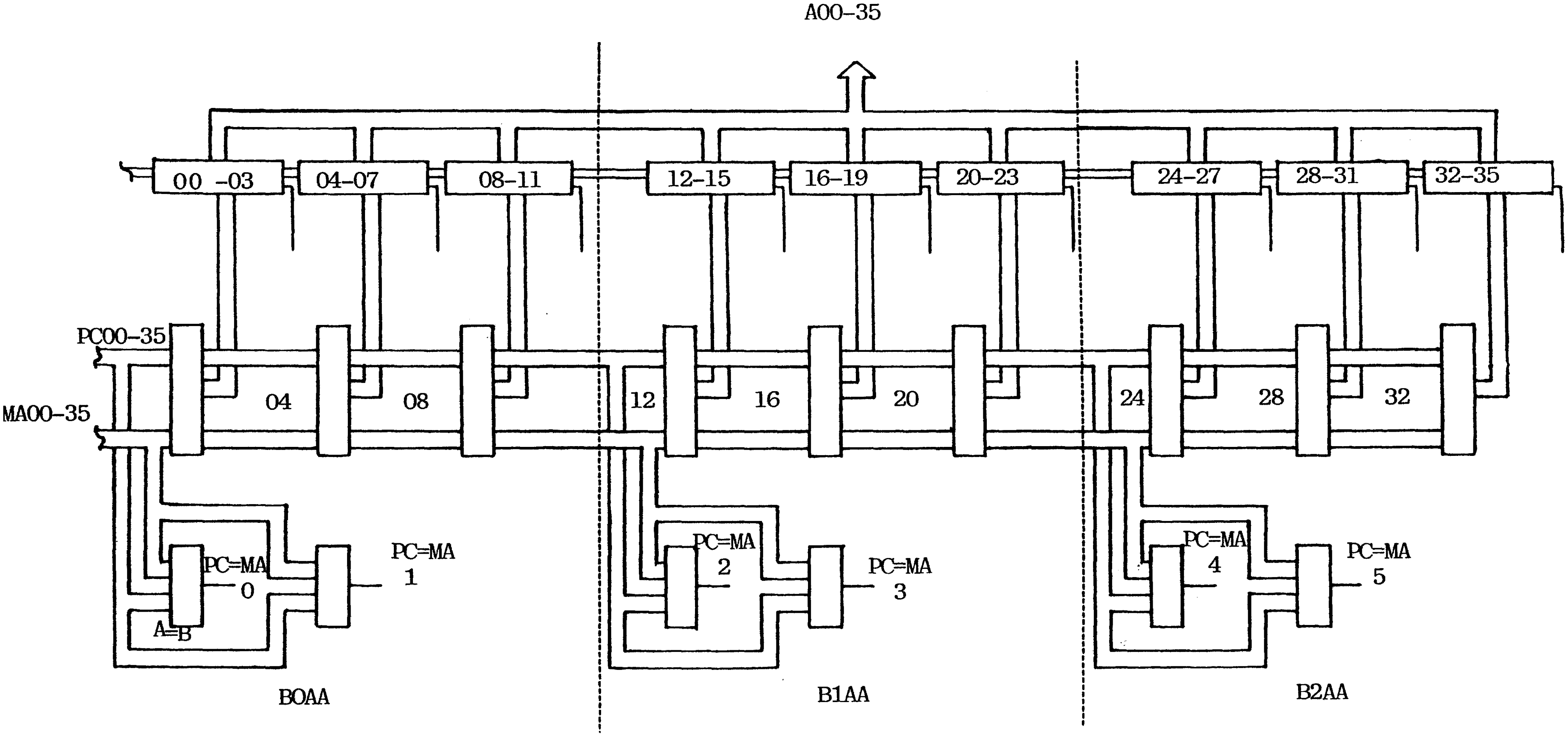


D.MARTIN

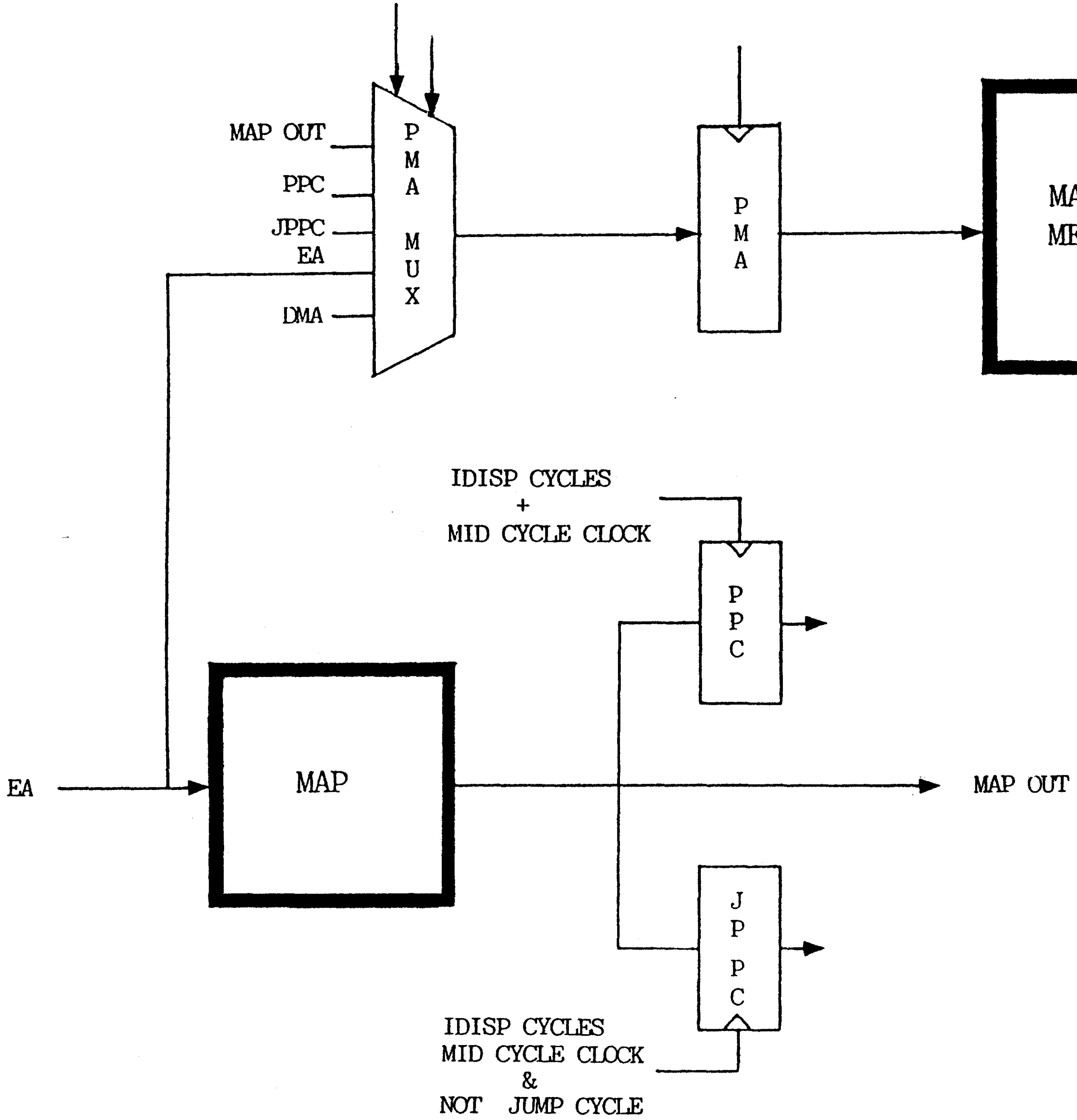
EA MUX SELECTION

EA BUS

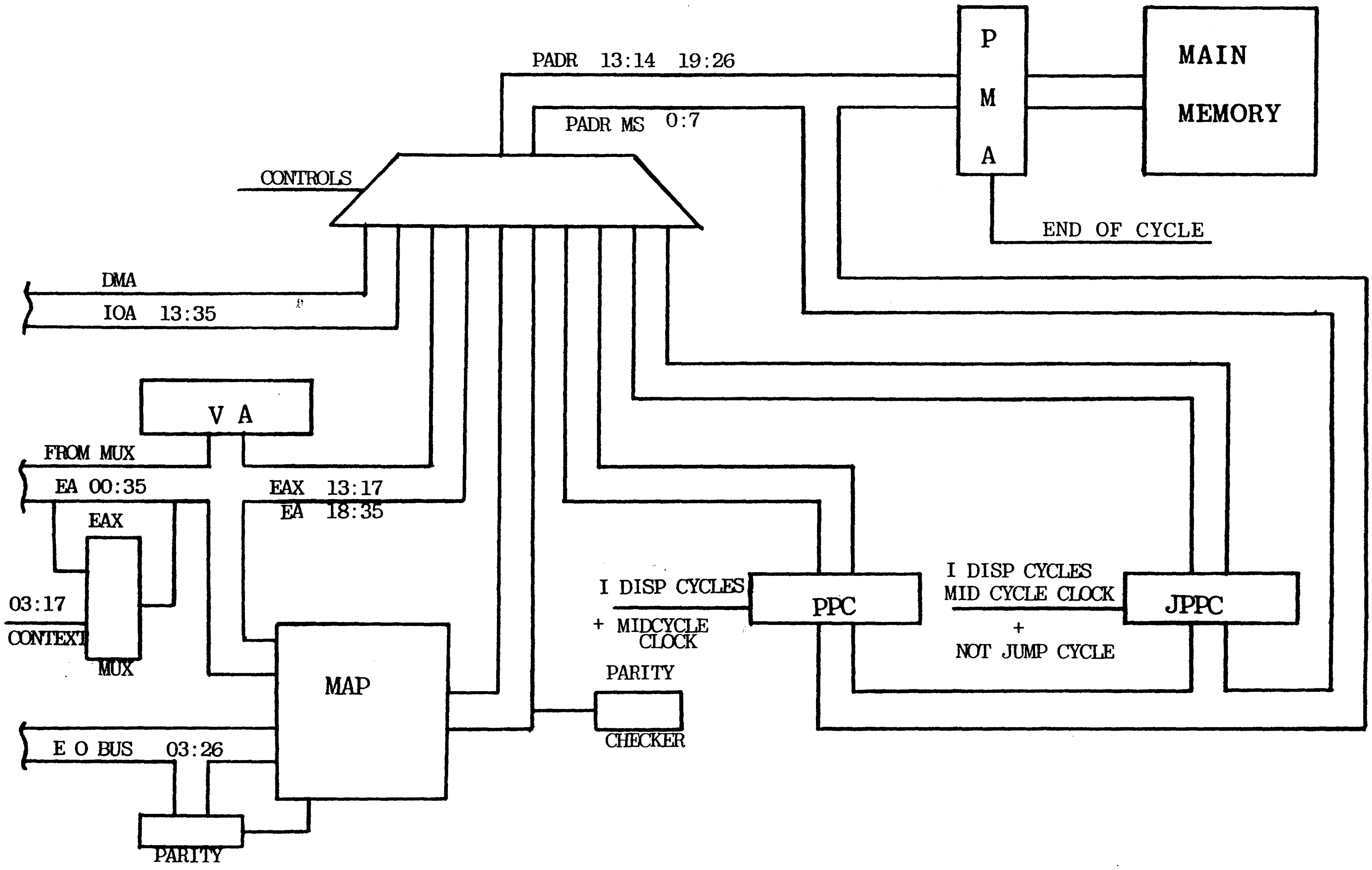

SQ
00-17
32-35
UNCORR.MEM
MEM
MC
18-35
O
BUS
MP
03-35
EA BUS
MA
IR
PC MUX
B0,1,2

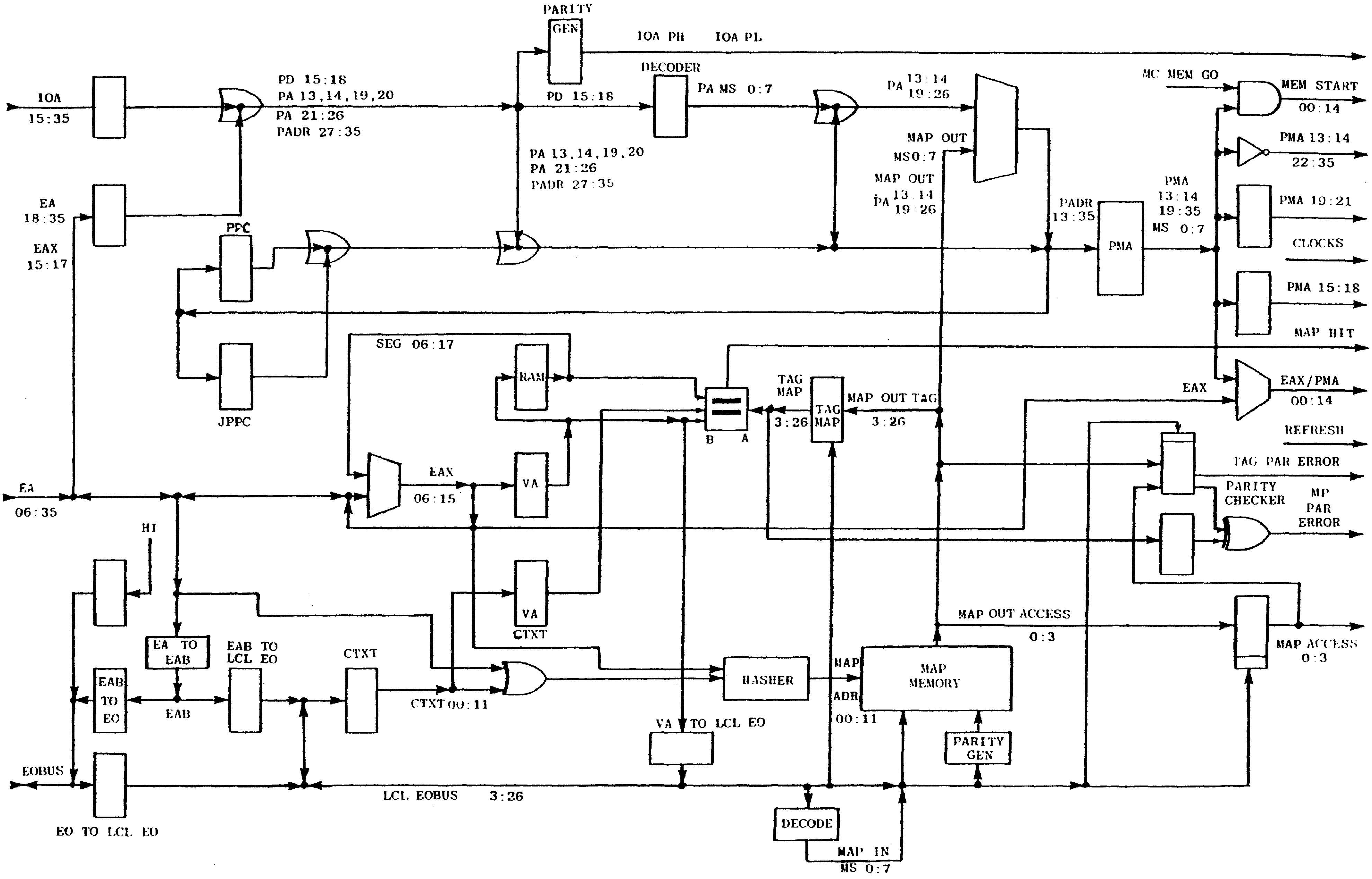

A00-35
00 -03
04-07
08-11
12-15
16-19
20-23
24-27
28-31
32-35
PC00-35
MA00-35
04
08
12
16
20
24
28
32
PC=MA
0
A=B
PC=MA
1
PC=MA
2
PC=MA
3
PC=MA
4
PC=MA
5
BOAA
B1AA
B2AA

BASIC MAP OPERATION

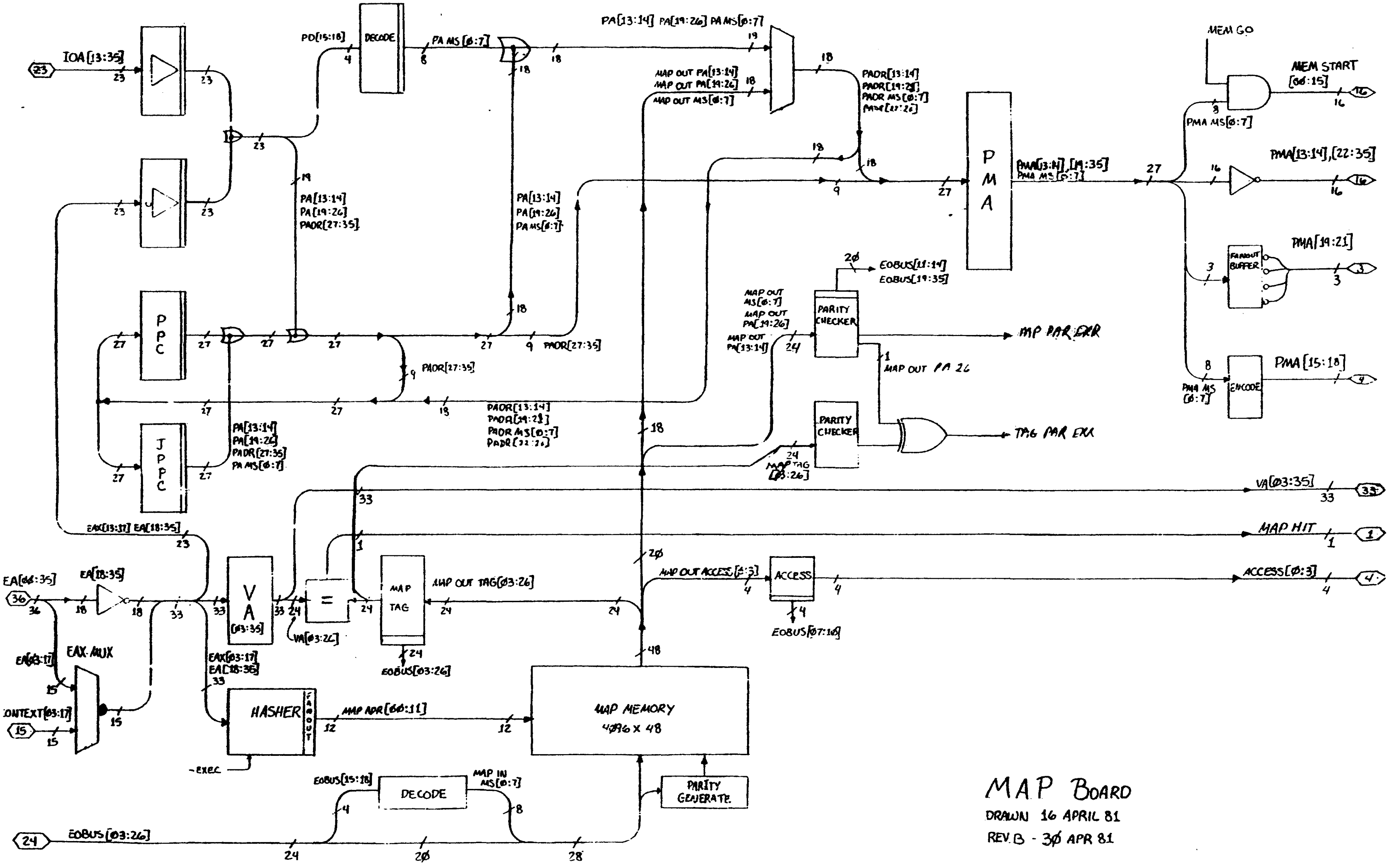


BASIC MAP OPERATION




PARITY
GEN
IOA PH
IOA PL
DECODER
IOA
15:35
PD 15:18
PA 13,14,19,20
PA 21:26
PADR 27:35
PD 15:18
PA MS 0:7
PA 13:14
19:26
MC MEM GO
MEM START
00:14
MAP OUT
MS0:7
PMA 13:14
22:35
EA
18:35
EAX
15:17
PA 13,14,19,20
PA 21:26
PADR 27:35
MAP OUT
PA 13:14
19:26
PPC
PADR
13:35
PMA
13:14
19:35
MS 0:7
PMA
PMA 19:21
CLOCKS
PMA 15:18
JPPC
SEG 06:17
RAM
MAP HIT
TAG
MAP
TAG
MAP
MAP OUT TAG
3:26
3:26
B
A
EAX
EAX/PMA
00:14
REFRESH
EA
06:35
EAX
06:15
VA
TAG PAR ERROR
PARITY
CHECKER
MP
PAR
ERROR
HI
VA
CTXT
MAP OUT ACCESS
0:3
EA TO
EAB
EAB TO
LCL EO
CTXT
EAB
TO
EO
EAB
CTXT 00:11
HASHER
MAP
ADR
00:11
MAP
MEMORY
MAP ACCESS
0:3
VA TO LCL EO
PARITY
GEN
EOBUS
LCL EOBUS 3:26
EO TO LCL EO
DECODE
MAP IN
MS 0:7

9 + 18 address Bits



MAP BOARD
DRAWN 16 APRIL 81
REV. B - 3Ø APR 81

D. MARTIN

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
									(4)	(2)	(1)	(4)	(2)	(1)	(4)	(2)	(1)			(20)	(10)	(4)	(2)	(1)	R	(10)	(4)	(2)	(1)
ENB EA LEFT	ENB EA FROM OBUS	LD MA	IDISP RQ	MEM WAIT	SEL AA FROM MA	PAR 0	SPARE 1	CARRY IN	ALU SRC			ALU FUN			ALU DST			ALU1 SEL	LD AR	JCOND						MAPF			

30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61
(40)	(20)	(10)	(4)	(2)	(1)	00	01	02	03	04	05	06	07	08	09	10	11	12	13	(40)	(20)	(10)	(4)	(2)	(1)	(40)	(20)	(10)	(4)	(2)	(1)
SPC						JADR OR IDISP CONTROLS														ROT SIZE						MASK SIZE					
						X MODE	CND ABRT (02)	IDX CY	IND ENB	IDX ENB	CND ABRT FLAG	JUMP FLAG	NOT CND ABRT (10)	NOT CND ABRT (04)	DISP FROM MEM	NOT USED	EOI IDISP	PRINCIPLE IDISP	HOLD PC												

62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87
(40)	(20)	(10)	(4)	(2)	(1)		(10)	(4)	(2)	(1)	(4)	(2)	(1)	(40)	(20)	(10)	(4)	(2)	(1)	(10)	(4)	(2)	(1)		
DEST						SPARE 2	JCODE				AC SEL			D						CYLEN				IF RQ	DF RQ

JADR
14 bits - any 8K Addr
in Micro Memory

AUGMENT ENGINE

SCALE:	APPROVED BY:	DRAWN BY HRK
DATE: 6-24-82	C.D. MARTIN	REVISED 7-9-82

26 MICROWORD

BIT ASSIGNMENTS

DRAWING NUMBER
1 of 2

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
ENB EA LEFT	ENB EA FROM OBUS	LD MA	DISP RQ	MEM WAIT	SEL AA FROM MA	PAR 0	SPARE 1	CARRY IN	ALU SRC			ALU FUN			ALU DST			ALU1 SEL	LD AR	JCOND					

26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61
LITERAL FIELD																																			

62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87
DEST						SPARE 2	JCODE				AC SEL			D						CYLEN				IF RQ	DF RQ

AUGMENT ENGINE

SCALE:

DATE: 6-28-82

APPROVED BY: C.D. MARTIN

DRAWN BY HRK

REVISED 7-9-82

26 MICROWORD

BIT ASSIGNMENTS

DRAWING NUMBER 2 of 2

MIWORD1.DWG

SECTION A

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
MI END EA LEFT	MI EA FROM OBUS	MI LD MA	MI IDISP RQ	MI MEM WAIT	MI SEL AA FROM MA	MI PAR 0	MI SP 01	MI CARRY IN	MI ALU SRC [4]	MI ALU SRC [2]	MI ALU SRC [1]	MI ALU FUN [4]	MI ALU FUN [2]	MI ALU FUN [1]	MI ALU DST [4]

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MI ALU DST [2]	MI ALU DST [1]	MI ALU1 SEL	MI LD AR	MI JCOND [20]	MI JCOND [10]	MI JCOND [04]	MI JCOND [02]	MI JCOND [01]	MI JCOND R	MI MAPF [10]	MI MAPF [04]	MI MAPF [02]	MI MAPF [01]	MI SPC [40]	MI SPC [20]

SECTION B

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
MI SPC [10]	MI SPC [04]	MI SPC [02]	MI SPC [01]	MI JADR 00	MI JADR 01	MI JADR 02	MI JADR 03	MI JADR 04	MI JADR 05	MI JADR 06	MI JADR 07	MI JADR 08	MI JADR 09	MI JADR 10	MI JADR 11
				JADR OR IDISP CONTROLS											
				X MODE	CND ABORT [02]	IDX CY	IND ENB	IDX ENB	CND ABORT FLAG	JUMP FLAG	~CND ABORT [10]	~CND ABORT [04]	DISP FROM MEM	NU	EOI IDSIP

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
MI JADR 12	MI JADR 13	MI ROT SIZE [40]	MI ROT SIZE [20]	MI ROT SIZE [10]	MI ROT SIZE [04]	MI ROT SIZE [02]	MI ROT SIZE [01]	MI MASK [40]	MI MASK [20]	MI MASK [10]	MI MASK [04]	MI MASK [02]	MI MASK [01]	MI DEST [40]	MI DEST [20]
PRIN-CIPLE IDISP	HOLD PC														

SECTION C

00	01	02	03	04	05	06	07	08	09	10	11
64	65	66	67	68	69	70	71	72	73	74	75
MI DEST [illegible] /0	MI DEST [illegible] 4	MI DEST [illegible] 2	MI DEST [illegible] 1	MI SP 02	MI JCODE [10]	MI JCODE [04]	MI JCODE [02]	MI JCODE [01]	MI ACSEL [04]	MI ACSEL [02]	MI ACSEL [01]

12	13	14	15	16	17	18	19	20	21	22	23
76	77	78	79	80	81	82	83	84	85	86	87
MI D [40]	MI D [20]	MI D [10]	MI D [04]	MI D [02]	MI D [01]	MI CYLEN [10]	MI CYLEN [04]	MI CYLEN [02]	MI CYLEN [01]	MI IF RQ	MI DF RQ

MACDONNELL DOUGLAS			
TITLE 26KL MICRO INSTRUCTION WORD 1			
SIZE B	CODE	NUMBER	REV A
DATE AUGUST 28,1984		SHEET 01 OF 02	

MIWORD2.DWG

SECTION A

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MI END EA LEFT	MI EA FROM OBUS	MI LD MA	MI IDISP RQ	MI MEM WAIT	MI SEL AA FROM MA	MI PAR 0	MI SP 01	MI CARRY IN	MI ALU SRC [4]	MI ALU SRC [2]	MI ALU SRC [1]	MI ALU FUN [4]	MI ALU FUN [2]	MI ALU FUN [1]	MI ALU DST [4]	MI ALU DST [2]	MI ALU DST [1]	MI ALU1 SEL	MI LD AR	MI JCOND [20]	MI JCOND [10]	MI JCOND [04]	MI JCOND [02]	MI JCOND [01]	MI JCOND R						

SECTION B

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
LITERAL FIELD																														MI DEST [40]	MI DEST [20]

SECTION C

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87
MI DEST [40]	MI DEST [40]	MI DEST [40]	MI DEST [40]	MI SP 02	MI JCODE [10]	MI JCODE [04]	MI JCODE [02]	MI JCODE [01]	MI ACSEL [04]	MI ACSEL [02]	MI ACSEL [01]	MI D [40]	MI D [20]	MI D [10]	MI D [04]	MI D [02]	MI D [01]	MI CYLEN [10]	MI CYLEN [04]	MI CYLEN [02]	MI CYLEN [01]	MI IF RQ	MI DF RQ

MACDONNELL DOUGLAS			
TITLE 26KL MICRO INSTRUCTION WORD 2			
SIZE B	CODE	NUMBER	REV A
DATE AUGUST 28,1984		SHEET 02 OF 02	

CHG SA LEFT	ENB SA FROM CBUS	LD MA	DISP RQ	MEM WAIT	SEL AA FROM MA	PAR 0	SPARE 1	CARRY IN	ALU SRC			ALU FUN			ALU DST			ALU1 SEL	LD AR	JCOND						MAPF				SPC						JADR OR DISP CONTROLS														ROT SIZE						MASK SIZE						DEST						SPARE 2	JCODE				AC SEL			D						CYLEN				IF RQ	DF RQ
									(4)	(2)	(1)	(4)	(2)	(1)	(4)	(2)	(1)			(20)	(10)	(4)	(2)	(1)	R	(10)	(4)	(2)	(1)	(40)	(20)	(10)	(4)	(2)	(1)	00	01	02	03	04	05	06	07	08	09	10	11	12	13	(40)	(20)	(10)	(4)	(2)	(1)	(40)	(20)	(10)	(4)	(2)	(1)	(40)	(20)	(10)	(4)	(2)	(1)		(10)	(4)	(2)	(1)	(4)	(2)	(1)	(40)	(20)	(10)	(4)	(2)	(1)	(10)	(4)	(2)	(1)		
																																				S MODE	ENB ABRT ENB	DIS CT	ENB ENB	DIS ENB	ENB ABRT FLAG	ABRT FLAG	RST ENB ABRT (16)	RST ENB ABRT (64)	DISP FROM MEM	ROT MEM	[illegible]	PREVIOUS DISP	HOLD PC																																						
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87

GLOSSARY OF TERMS FOR MICROWORD

ENB EA LEFT - Enables left half of EAMUX

ENB EA From OBUS - Allows OBUS thru EAMUX

LD MA - Allows MA reg.loaded from EA BUS

I DISP-RQ - Instruction Dispatch Request. Takes a (001 CMPSL
Dispatch to (2*OPCODE) + Dispatch Base Reg. 6002)
'5 (AC) Then +1 more.

MEM WAIT - Inhibits Gated Clock

SEL AA FROM MA - Selects MA for the Address Adder

PAR 0 - Parity Bit for Microword

SPARE 1 - Not Used

CARRY IN - Bit Added to ALU

ALU SRC - ALU SOURCE)
)   NOTE:
ALU FUN - ALU FUNCTION)--   See Table For 2901C
)
ALU DST - ALU DESTINATION)

ALU1 SEL - SELECTS THE 2nd ALU GROUP

LD AR - ENABLES LOADING OF AR REG

J COND - CONDITION FIELD FOR MICROWORD JUMP

REV - REVERSE CONDITION OF JCOND.

MAPF - MULTIPLE FUNCTION

SPC - SPECIAL FUNCTIONS

JADR) - ADDRESS FOR MICR MEMORY TO JUMP TO
)-=(ONE IN THE SAME)
I DISP CONTROL) - INSTRUCTION DISPATCH CONTROLS IF NOT A JUMP CYCLE

SUB FIELDS •
X MODE - NOT USED ALWAYS ZERO

CND ABRT (02) - 02 Bit of Condition Abort Dispatch

IDX CY - INDEX CYCLE TRAP ON INDEX BITS

IND ENB - INDIRECT ENABLE

CMD ABRT FLAG - CONDITION ABORT FLAG

JUMP FLAG - JUMP

NOT CND ABRT (10)- 10 BIT OF CONDITION ABORT DISPATCH

GLOSSARY OF TERMS FOR MICROWORD

NOT CND ABORT (04)	BIT OF CONDITION ABORT DISPATCH
DISP FROM MEM	EA MUX SOURCE DURING INSTRUCTION DISPATCH (JADR 09)
FIELD 46 BIT (N/U)	INDICATES AN INDIRECT CYCLE (NOT USED)
EOI I DISP	END OF INSTRUCTION
PRINCIPLE I DISP	MOST DISPATCHES ARE PRINCIPLE
HOLD PC	HOLD PROGRAM COUNT
ROT SIZE	ROTATES D BUS LEFT BY VALUE
MASK SIZE	NUMBER OF BITS ON, RIGHT JUSTIFIED CAN SPECIFY RIGHT OR LEFT MASK
DEST	SPECIFIES WHERE OUTPUT OF ALU IS TO GO.
SPARE 2	NOT USED
J CODE	TYPE OF JUMP
AC SEL	SPECIFIES WHAT ADDRESSES THE AC INSIDE THE ALU.
D	SPECIFIES WHAT IS TO BE PUT ON THE D BUS
CYLEN	SPECIFIED CYCLE LENGTH
IF RQ *	SPECIFIES INSTRUCTION FETCH REQUEST
DF RQ *	SPECIFIES DATA FETCH FROM MAIN MEMORY

* NOTE BITS ARE INVERTED

JCOND Field

JCOND+REV	=	0	OBUS NOT EQUAL 0
JCOND+REV	=	1	OBUS EQUAL 0
JCOND+REV	=	2	OBUS LESS THAN 0
JCOND+REV	=	3	OBUS NOT LESS THAN 0
JCOND+REV	=	4	OBUS NOT GREATER THAN 0
JCOND+REV	=	5	OBUS GREATER THAN 0
JCOND+REV	=	6	TO BE ADDRESSED
JCOND+REV	=	7	TO BE ADDRESSED
JCOND+REV	=	10	OBUS18
JCOND+REV	=	11	NOT OBUS18
JCOND+REV	=	12	Q0-35, BIT SHIFTED INTO Q36
JCOND+REV	=	13	NOT Q0-35, NO BIT SHIFTED INTO Q36
JCOND+REV	=	14	CRYO
JCOND+REV	=	15	NOT CRYO
JCOND+REV	=	16	HALF
JCOND+REV	=	17	NOT HALF
JCOND+REV	=	20	MM-ACC-COND (FOR DEPOSITING MICRO MEM)
JCOND+REV	=	21	-MM-ACC-COND(FOR DEPOSITING MICRO MEM)
JCOND+REV	=	22	TO BE ADDRESSED
JCOND+REV	=	23	TO BE ADDRESSED
JCOND+REV	=	24	TO BE ADDRESSED
JCOND+REV	=	25	TO BE ADDRESSED
JCOND+REV	=	26	TO BE ADDRESSED
JCOND+REV	=	27	TO BE ADDRESSED

JCOND+REV	=	30	TO BE ADDRESSED
JCOND+REV	=	31	TO BE ADDRESSED
JCOND+REV	=	32	TO BE ADDRESSED
JCOND+REV	=	33	TO BE ADDRESSED
JCOND+REV	=	34	MAPF = 0 NO-MAPF
JCOND+REV	=	35	TO BE ADDRESSED
JCOND+REV	=	36	TO BE ADDRESSED
JCOND+REV	=	37	TO BE ADDRESSED
JCOND+REV	=	40	BYTE-OVF (LOOKS AT PTR ON EA)
JCOND+REV	=	41	NOT BYTE-OVF
JCOND+REV	=	42	INTRPT
JCOND+REV	=	43	NOT INTRPT
JCOND+REV	=	44	MA < 20
JCOND+REV	=	45	MA > =20
JCOND+REV	=	46	AC EQUAL 0
JCOND+REV	=	47	AC NOT EQUAL 0
JCOND+REV	=	50	EA<20
JCOND+REV	=	51	EA > =20
JCOND+REV	=	52	USER
JCOND+REV	=	53	EXEC
JCOND+REV	=	54	TO BE ADDRESSED
JCOND+REV	=	55	TO BE ADDRESSED
JCOND+REV	=	56	MAPF = 10 R-M-W, USED FOR READ MODIF WRITE DURING D (MEM) CYCLE
JCOND+REV	=	57	MAPF = 0 NO-MAPF

JCOND+REV	=	60	TO BE ADDRESSED
JCOND+REV	=	61	TO BE ADDRESSED
JCOND+REV	=	62	TO BE ADDRESSED
JCOND+REV	=	63	TO BE ADDRESSED
JCOND+REV	=	64	TO BE ADDRESSED
JCOND+REV	=	65	TO BE ADDRESSED
JCOND+REV	=	66	LAST-COND
JCOND+REV	=	67	NOT LAST-COND
JCOND+REV	=	70	CONTINUE (FORCES .+1 REGARDLESS OF JCODE
JCOND+REV	=	71	TO BE ADDRESSED
JCOND+REV	=	72	TO BE ADDRESSED
JCOND+REV	=	73	TO BE ADDRESSED
JCOND+REV	=	74	TRUE (DEFAULT)
JCOND+REV	=	75	FALSE
JCOND+REV	=	76	TO BE ADDRESSED
JCOND+REV	=	77	TO BE ADDRESSED

MAPF Field

MAPF	=	0	NO-MAPF
MAPF	=	1	TO BE ADDRESSED
MAPF	=	2	TO BE ADDRESSED
MAPF	=	3	TO BE ADDRESSED
MAPF	=	4	TO BE ADDRESSED
MAPF	=	5	TO BE ADDRESSED
MAPF	=	6	TO BE ADDRESSED
MAPF	=	7	TO BE ADDRESSED
MAPF	=	10	R-M-W, USED FOR READ MODIFY WRITE DURING D(MEM)CYCLE
MAPF	=	11	TO BE ADDRESSED
MAPF	=	12	TO BE ADDRESSED
MAPF	=	13	TO BE ADDRESSED
MAPF	=	14	TO BE ADDRESSED
MAPF	=	15	TO BE ADDRESSED
MAPF	=	16	TO BE ADDRESSED
MAPF	=	17	TO BE ADDRESSED

JCODE Field

JCODE	=0	JUMP, SEE JADR
JCODE	=1	JUMPLC, JUMP TO JADR IF LAST CONDITION
JCODE	=2	LBJUMP, JUMP TO JADR=1 IF TRUE CONDITION ELSE JADR
JCODE	=3	LBJUMPLC, JUMP TO JADR+1 IF LAST CONDITION ELSE JADR
JCODE	=4	PUSHJ, JUMP TO JADR AND SAVE MIC ON STACK
JCODE	=5	PUSHJLC, JUMP TO JADR IF LAST CONDITION AND SAVE MIC ON STACK
JCODE	=6	IDISP (SEE IDISP CONTROLS)
JCODE	=7	CONT, CONTINUE AND SELECT MI-BA FROM JADR (DEFAULT)
JCODE	=10	POPJ, PULL NEXT MICRO-INSTRUCTION ADDRESS OFF STACK
JCODE	=11	POPJLC ()
JCODE	=12	LBPOPJ ()
JCODE	=13	LBPOPJLC ()
JCODE	=14	@JADR ()
JCODE	=15	@JADRLC ()
JCODE	=16	ODISP, BRANCH TO LOCATION ON OBUS 00:13 (SKIP DISP)
JCODE	=17	CONT1,CONTINUE BUT SELECT MI-BA FROM JMEM

SPC Field

SPC	=	0	NO-SPEC (DEFAULT)
SPC	=	1	AMEM-P-DECR (DECREMENT AMEM PDL ADDR. CNTR.)
SPC	=	2	MU-PUSH (INCREMENT JMEM PDL ADDR. CNTR.)
SPC	=	3	MU-POP (DECREMENT JMEM PDL ADDR. CNTR.)
SPC	=	4	MM-ACCESS (RD/WRT MM SECTION MAPF AT LOCATION JADR)
SPC	=	5	PC=PC+1
SPC	=	6	MA-FROM-MEM (FORCES EAMUX TO SELECT MEM)
SPC	=	7	AMEM-P-INCR (INCREMENT AMEM PDL ADDRESS COUNTER)
SPC	=	10	LD-LOOP (LOAD LOOP-CTR FROM EOBUS, DEST MUST BE 0)
SPC	=	11	MU-INDEX (MICRO INDEX)
SPC	=	12	MAP-ENABLE
SPC	=	13	MAP-DISABLE
SPC	=	14	REENABLE-TRAP1
SPC	=	15	NOP
SPC	=	16	UHS-ENABLE (TURN ON MICRO HISTORY SYSTEM)
SPC	=	17	USH-DISABLE (TURN ON MICRO HISTORY SYSTEM
SPC	=	20	SUPPRESS-FETCH-TRAPS
SPC	=	21	MHS-DISABLE (TURN OFF MACRO HISTORY SYSTEM)
SPC	=	22	MHS-ENABLE (TURN ON MACRO HISTORY SYSTEM
SPC	=	23	CLR-DMA-ERR
SPC	=	24	FAKE-DFWAIT (D 0 AND D 1 READ PORTIONS OF TRAPPED MA)
SPC	=	25	NOP
SPC	=	26	NOP
SPC	=	27	NOP

SPC	=	30	IOB-IN
SPC	=	31	IOB-OUT
SPC	=	32	PMASTAT-SEL0
SPC	=	33	PMASTAT-SEL1
SPC	=	34	PMASTAT-SEL2
SPC	=	35	SEL-PMA-STAT
SPC	=	36	TO BE ADDRESSED
SPC	=	37	SEE D
SPC	=	40	CLR-DEV-FROM-INTR
SPC	=	41	CLR-PC-TRAP-FLAGS
SPC	=	42	CLR-AROV-TRAP-FLAGS
SPC	=	43	CLR-HALF
SPC	=	44	SET-PC-FLAGS (SET PC FLAGS FROM ALU RESULT)
SPC	=	45	TO BE ADDRESSED
SPC	=	46	CLR-MAP-SR (DOES SET PC FLAGS, MUST DO D+0 TO PREV. CRYS)
SPC	=	47	BUS RESET (DOES SET PC FLAGS, MUST DO D+0 TO PREV CRYS)
SPC	=	50	CLR-DEV-FROM-INTER
SPC	=	51	CLR-PC-TRAP-FLAGS
SPC	=	52	CLR-AROV-TRAP-FLAG
SPC	=	53	CLR-HALF
SPC	=	54	SET-PC-FLAGS (SET PC FLAGS FROM ALU RESULT)
SPC	=	55	TO BE ADDRESSED
SPC	=	56	CLR-MAP-SR (DOES SET PC FLAGS, MUST DO D+0 to PREV. CRYS)
SPC	=	57	BUS-RESET (DOES SET PC FLAGS, MUST DO D+0 TO PREV. CRYS)

SPC	=	60	TO BE ADDRESSED
SPC	=	61	TO BE ADDRESSED
SPC	=	62	TO BE ADDRFSSED
SPC	=	63	TO BE ADDRESSED
SPC	=	64	TO BE ADDRESSED
SPC	=	65	TO BE ADDRESSED
SPC	=	66	TO BE ADDRESSED
SPC	=	67	TO BE ADDRESSED

SPC	=	70	TO BE ADDRESSED
SPC	=	71	TO BE ADDRESSED
SPC	=	72	TO BE ADDRESSED
SPC	=	73	TO BE ADDRESSED
SPC	=	74	TO BE ADDRESSED
SPC	=	75	TO BE ADDRESSED
SPC	=	76	TO BE ADDRESSED
SPC	=	77	TO BE ADDRESSED

ROT SIZE Field

ROTATER SIZE	=	0	ROTATED BY THIS VALUE
ROTATER SIZE	=	1	ROTATED BY THIS VALUE
ROTATER SIZE	=	2	TNOBIDN
ROTATER SIZE	=	3	TNIRIBN
ROTATER SIZE	=	4	ROTATED BY THIS VALUE
ROTATER SIZE	=	5	ROTATED BY THIS VALUE
ROTATER SIZE	=	6	TNODOBN
ROTATER SIZE	=	7	TNIROBN
ROTATER SIZE	=	10	TNIROBNPO
ROTATER SIZE	=	11	ROTATED BY THIS VALUE
ROTATER SIZE	=	12	ROTATED BY THIS VALUE
ROTATER SIZE	=	13	ROTATED BY THIS VALUE
ROTATER SIZE	=	14	ROTATED BY THIS VALUE
ROTATER SIZE	=	15	ROTATED BY THIS VALUE
ROTATER SIZE	=	16	ROTATED BY THIS VALUE
ROTATER SIZE	=	17	ROTATED BY THIS VALUE
ROTATER SIZE	=	20	ROTATED BY THIS VALUE
ROTATER SIZE	=	21	ROTATED BY THIS VALUE
ROTATER SIZE	=	22	ROTATED BY THIS VALUE
ROTATER SIZE	=	23	ROTATED BY THIS VALUE
ROTATER SIZE	=	24	ROTATED BY THIS VALUE
ROTATER SIZE	=	25	ROTATED BY THIS VALUE
ROTATER SIZE	=	26	ROTATED BY THIS VALUE
ROTATER SIZE	=	27	ROTATED BY THIS VALUE

ROTATER SIZE	=	30	ROTATED BY THIS VALUE
ROTATER SIZE	=	31	ROTATED BY THIS VALUE
ROTATER SIZE	=	32	ROTATED BY THIS VALUE
ROTATER SIZE	=	33	ROTATED BY THIS VALUE
ROTATER SIZE	=	34	ROTATED BY THIS VALUE
ROTATER SIZE	=	35	ROTATED BY THIS VALUE
ROTATER SIZE	=	36	ROTATED BY THIS VALUE
ROTATER SIZE	=	37	ROTATED BY THIS VALUE
ROTATER SIZE	=	40	ROTATED BY THIS VALUE
ROTATER SIZE	=	41	ROTATED BY THIS VALUE
ROTATER SIZE	=	42	ROTATED BY THIS VALUE
ROTATER SIZE	=	43	ROTATED BY THIS VALUE
ROTATER SIZE	=	44	ROTATED BY THIS VALUE
ROTATER SIZE	=	45	ROTATED BY THIS VALUE
ROTATER SIZE	=	46	ROTATED BY THIS VALUE
ROTATER SIZE	=	47	ROTATED BY THIS VALUE
ROTATER SIZE	=	50	USE ROT SIZE REGISTER
ROTATER SIZE	=	51	ROTATED BY THIS VALUE
ROTATER SIZE	=	52	ROTATED BY THIS VALUE
ROTATER SIZE	=	53	ROTATED BY THIS VALUE
ROTATER SIZE	=	54	ROTATED BY THIS VALUE
ROTATER SIZE	=	55	ROTATED BY THIS VALUE
ROTATER SIZE	=	56	ROTATED BY THIS VALUE
ROTATER SIZE	=	57	ROTATED BY THIS VALUE

ROTATER SIZE	=	60	ROTATER DISABLED (DEFAULT)
ROTATER SIZE	=	61	ROTATED BY THIS VALUE
ROTATER SIZE	=	62	ROTATED BY THIS VALUE
ROTATER SIZE	=	63	ROTATED BY THIS VALUE
ROTATER SIZE	=	64	ROTATED BY THIS VALUE
ROTATER SIZE	=	65	ROTATED BY THIS VALUE
ROTATER SIZE	=	66	ROTATED BY THIS VALUE
ROTATER SIZE	=	67	ROTATED BY THIS VALUE

ROTATER SIZE	=	70	ROTATED BY THIS VALUE
ROTATER SIZE	=	71	ROTATED BY THIS VALUE
ROTATER SIZE	=	72	ROTATED BY THIS VALUE
ROTATER SIZE	=	73	ROTATED BY THIS VALUE
ROTATER SIZE	=	74	ROTATED BY THIS VALUE
ROTATER SIZE	=	75	ROTATED BY THIS VALUE
ROTATER SIZE	=	76	ROTATED BY THIS VALUE
ROTATER SIZE	=	77	ROTATED BY THIS VALUE

MASK Field

MASK	=	01	SEE MASKER VALUE
MASK	=	02	SEE MASKER VALUE
MASK	=	03	SEE MASKER VALUE
MASK	=	04	SEE MASKER VALUE
MASK	=	05	SEE MASKER VALUE
MASK	=	06	SEE MASKER VALUE
MASK	=	07	SEE MASKER VALUE
MASK	=	10	SEE MASKER VALUE
MASK	=	11	SEE MASKER VALUE
MASK	=	12	SEE MASKER VALUE
MASK	=	13	SEE MASKER VALUE
MASK	=	14	SEE MASKER VALUE
MASK	=	15	SEE MASKER VALUE
MASK	=	16	SEE MASKER VALUE
MASK	=	17	SEE MASKER VALUE
MASK	=	20	SEE MASKER VALUE
MASK	=	21	SEE MASKER VALUE
MASK	=	22	SEE MASKER VALUE
MASK	=	23	SEE MASKER VALUE
MASK	=	24	SEE MASKER VALUE
MASK	=	25	SEE MASKER VALUE
MASK	=	26	SEE MASKER VALUE
MASK	=	27	SEE MASKER VALUE

MASK	=	30	SEE MASKER VALUE
MASK	=	31	SEE MASKER VALUE
MASK	=	32	SEE MASKER VALUE
MASK	=	33	SEE MASKER VALUE
MASK	=	34	SEE MASKER VALUE
MASK	=	35	SEE MASKER VALUE
MASK	=	36	SEE MASKER VALUE
MASK	=	37	SEE MASKER VALUE
MASK	=	40	SEE MASKER VALUE
MASK	=	41	SEE MASKER VALUE
MASK	=	41	SEE MASKER VALUE
MASK	=	43	SEE MASKER VALUE
MASK	=	44	SEE MASKER VALUE
MASK	=	45	SEE MASKER VALUE
MASK	=	46	SEE MASKER VALUE
MASK	=	47	SEE MASKER VALUE
MASK	=	50	USE MASK SIZE REGISTER
MASK	=	51	SEE MASKER VALUE
MASK	=	52	SEE MASKER VALUE
MASK	=	53	SEE MASKER VALUE
MASK	=	54	SEE MASKER VALUE
MASK	=	55	SEE MASKER VALUE
MASK	=	56	SEE MASKER VALUE
MASK	=	57	SEE MASKER VALUE

MASK		=	60	SEE MASKER VALUE
MASK		=	61	SEE MASKER VALUE
MASK		=	62	SEE MASKER VALUE
MASK		=	63	SEE MASKER VALUE
MASK		=	64	SEE MASKER VALUE
MASK		=	65	SEE MASKER VALUE
MASK		=	66	SEE MASKER VALUE
MASK		=	67	SEE MASKER VALUE
MASK		=	70	NO-MASK (DEFAULT)
MASK	(ENDCONN)	=	71	ACO_SIGN, Q35_0
MASK	(ENDCONN)	=	72	ACO_0,Q35_0
MASK	(ENDCONN)	=	73	ACO_OVF#SIGN, Q35_ -SIGN
MASK		=	74	SEE MASKER VALUE
MASK		=	75	SEE MASKER VALUE
MASK		=	76	SEE MASKER VALUE
MASK		=	77	SEE MASKER VALUE

DEST Field

```
DEST            =  0           TO BE ADDRESSED

DEST            =  1           JMEM-MIC, DATA IS MICROINSTRUCTION COUNTER

DEST            =  2           JMEM, DATA IS EOBUS

DEST            =  3           JMEM-P, LOADS JMEM STACK PTR FROM OBUSO:9

DEST            =  4 SPARE 2              = 0      NOT USED

DEST            =  5           AMEM-P, LOADS AMEM STACK PTR FROM OBUSO:9

DEST            =  6 SPARE 2              = 0      NOT USED

DEST            =  7           COND-AC-STO, FORCE 2 BIT OF ALU DST IF AC ADD
                               <> 0


DEST            =  10          VA-MODE, ADDRESS ENGINE MA SOURCE CONTROL

DEST            =  11          CPU-MODE, XMODE AND ECC INHIBIT AND MAP ON

DEST            =  12          IDISP-REG, INTERPRETER BASE ADDRESS

DEST            =  13          SELECT-HOLD, CAUSES HOLD TO APPEAR ON MEM BUS

DEST            =  14          SELECT-MB, CAUSES MB TO APPEAR ON MEM BUS

DEST            =  15 SPARE 2             = 0      NOT USED

DEST            =  16 SPARE 2             = 0      NOT USED

DEST            =  17 SPARE 2             = 0      NOT USED


DEST            =  20          MAP-EXEC-SR

DEST            =  21          PC-FLAGS, JUST THE FLAGS

DEST            =  22          ROTR, ROT SIZE REGISTER

DEST            =  23          DEV-ADR, DEVICE ADDRESS REGISTER

DEST            =  24          MASKR, MASK SIZE REGISTER

DEST            =  25          TO BE ADDRESSED

DEST            =  26          MAP-MA, HALF OF MAP MEMORY

DEST            =  27          MAP-TAG, HALF OF MAP MEMORY
```

DEST	=	30	EXEC-CTXT, ONE OF TWO
DEST	=	31	USER-CTXT, ONE OF TWO
DEST	=	32	HS-ADR, MACRO BREAK COMPARISON ADDRESS
DEST	=	33	HS-CTRL, MACRO BREAK CONTROL FLAGS AND RECORD ADDRESS
DEST	=	34	HS-COUNT, MACRO BREAK HIT COUNTER AND BREAK DELAY
DEST	=	35	MERGE, CAUSES EA03:17 TO BE MERGED WITH EA18:35
DEST	=	36	CLR-ECC-ERR
DEST	=	37	TO BE ADDRESSED

DEST	=	40	AMEM 0
DEST	=	41	AMEM 1
DEST	=	42	AMEM 2
DEST	=	43	AMEM 3
DEST	=	44	AMEM 4
DEST	=	45	AMEM 5
DEST	=	46	AMEM 6
DEST	=	47	AMEM 7

DEST	=	50	AMEM 10
DEST	=	51	AMEM 11
DEST	=	52	AMEM 12
DEST	=	53	AMEM 13
DEST	=	54	AMEM 14
DEST	=	55	AMEM 15
DEST	=	56	AMEM 16
DEST	=	57	AMEM 17

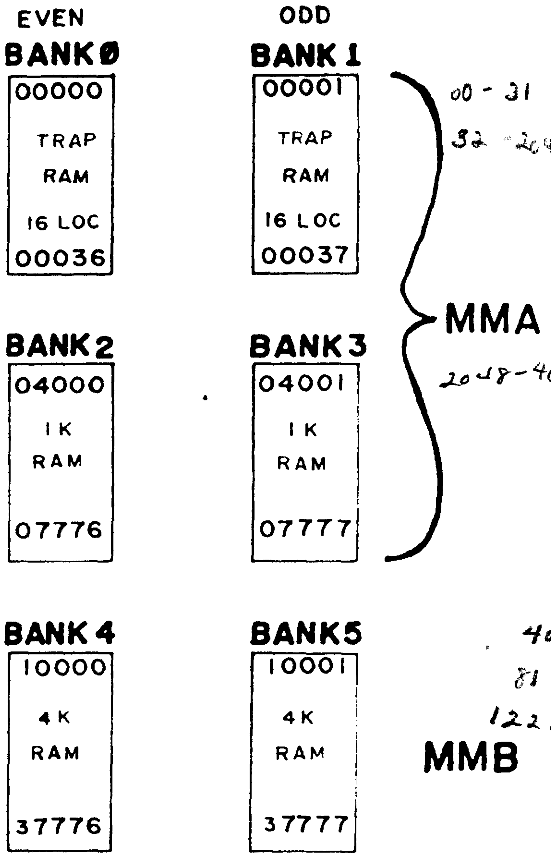
DEST	=	60	IR-ALL, LOADS OF THE IR
DEST	=	61	LD-IR-23, LOADS IR13:35
DEST	=	62	IR-ADR, LOADS IR18:35
DEST	=	63	LD-PC
DEST	=	64	HOLD, LOAD HOLD REG WITHOUT STARTING MEMORY CYCLE
DEST	=	65	IOD
DEST	=	66 SPARE 2	= 0 NOT USED
DEST	=	67	MEMSTO, STORE INTO HOLD REG AND START WRITE CYCLE
DEST	=	70	TO BE ADDRESSED
DEST	=	71	NO-DEST (DEFAULT)
DEST	=	72	STR-WRT, START WRITE WITHOUT LOADING HOLD
DEST	=	73	FORCE-MMAD SQ, FORCE IDISP TO USE SQ MM AD FOR MM AD
DEST	=	74	AMEM@P, ADDRESS AMEM VIA AMEM-P
DEST	=	75	TO BE ADDRESSED
DEST	=	76	TO BE ADDRESSED
DEST	=	77	MM-PRE-WRT, WRITE INTO MICRO MEMORY ON NEXT CYCLE

D Field

D	=	0	EOBUS ONTO D MUX (HS-ADR)
D	=	1	EOBUS ONTO D MUX (HS-CTRL)
D	=	2	EOBUS ONTO D MUX (HS-COUNT)
D	=	3	EOBUS ONTO D MUX (MAP-MA)
D	=	4	EOBUS ONTO D MUX (MAP-TAG)
D	=	5	EOBUS ONTO D MUX (DMA-STATUS, ERROR DATA FOR TYBUS)
D	=	6	EOBUS ONTO D MUX (MB-STATUS)
D	=	7	EOBUS ONTO D MUS (MUHS-CNT)
D	=	10	EOBUS ONTO D MUX (AMEM-P 00:09 LC 10 MBFF 11)
D	=	11	EOBUS ONTO D MUX (LOOP-CTR)
D	=	12	EOBUS ONTO D MUX (MI-BA, WHERE WE WOULD GO IF BRANCH)
D	=	13	EOBUS ONTO D MUX (JMEM-P)
D	=	14	EOBUS ONTO D MUX (PC-FLAGS)
D	=	15	EOBUS ONTO D MUX (DEVADR, INTADR, USER SR)
D	=	16	ENABLE DBUS FROM MASK
D	=	17	TO BE ADDRESSED
D	=	20	TO BE ADDRESSED
D	=	21	TO BE ADDRESSED
D	=	22	TO BE ADDRESSED
D	=	23	TO BE ADDRESSED
D	=	24	TO BE ADDRESSED
D	=	25	TO BE ADDRESSED
D	=	26	TO BE ADDRESSED
D	=	27	TO BE ADDRESSED

D	=	30	AR (DEFAULT)
D	=	31	MEM, POSSIBLY HOLD REGISTER
D	=	32	LITERAL VALUE (000030266022) INTO D MUX
D	=	33	CONSTANT VALUE INTO D MUX FROM MASK (0000000000 22
D	=	34	PC
D	=	35	MA
D	=	36	IOD
D	=	37	IR
D	=	40	AMEM 0
D	=	41	AMEM 1
D	=	42	AMEM 2
D	=	43	AMEM 3
D	=	44	AMEM 4
D	=	45	AMEM 5
D	=	46	AMEM 6
D	=	47	AMEM 7
D	=	50	AMEM 10
D	=	51	AMEM 11
D	=	52	AMEM 12
D	=	53	AMEM 13
D	=	54	AMEM 14
D	=	55	AMEM 15
D	=	56	AMEM 16
D	=	57	AMEM 17

D	=	60	AMEM-ABS (ABSOLUTE AMEM ADR, SEE JADR)
D	=	61	TO BE ADDRESSED
D	=	62	TO BE ADDRESSED
D	=	63	TO BE ADDRESSED
D	=	64	TO BE ADDRESSED
D	=	65	TO BE ADDRESSED
D	=	66	TO BE ADDRESSED
D	=	67	TO BE ADDRESSED
D	=	70	AMEM@P (ADR AMEM VIA AMEM-P)
D	=	71	TO BE ADDRESSED
D	=	72	TO BE ADDRESSED
D	=	73	TO BE ADDRESSED
D	=	74	TO BE ADDRESSED
D	=	75	TO BE ADDRESSED
D	=	76	TO BE ADDRESSED
D	=	77	TO BE ADDRESSED



00 - 31

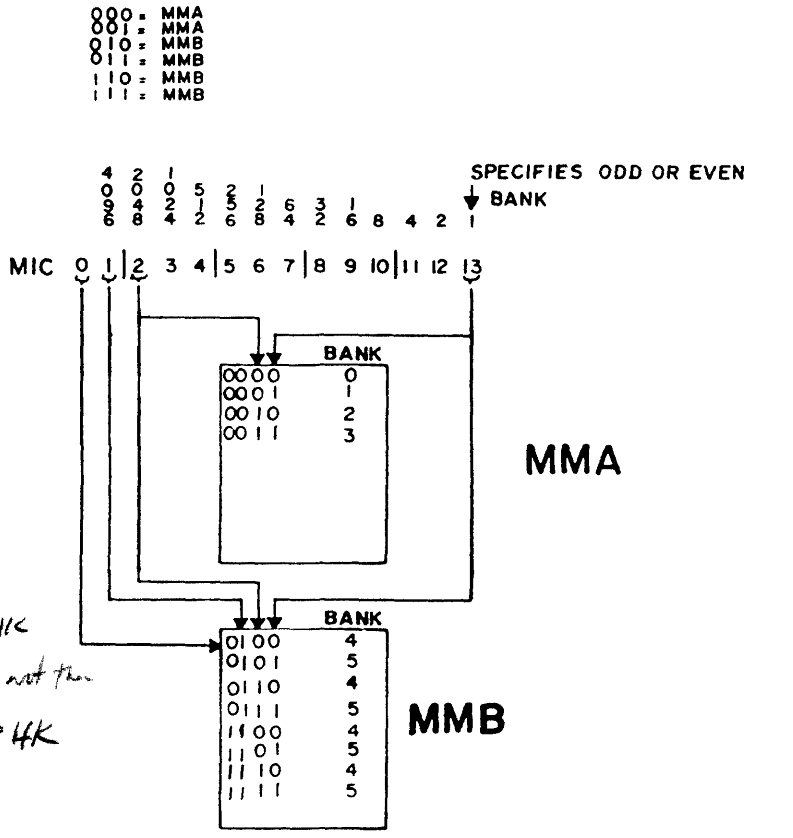
32 - 2047 = not there

2048 - 4095 = 2K

4096 - 8191 = 4K

8192 - 12287 = not there

12288 - 16383 = 4K



00-37	TRAP RAM BANK Ø OR BANK 1
40-3777	REALLY DOES NOT EXIST; WRAPS AROUND
20000-27777	REALLY DOES NOT EXIST; WRAPS AROUND

16K - 4K - 2016 = 10K + 32 words

00 - 37 does not have all 88 bits
+ kee up as * on the IBM PC

— 26 KL —

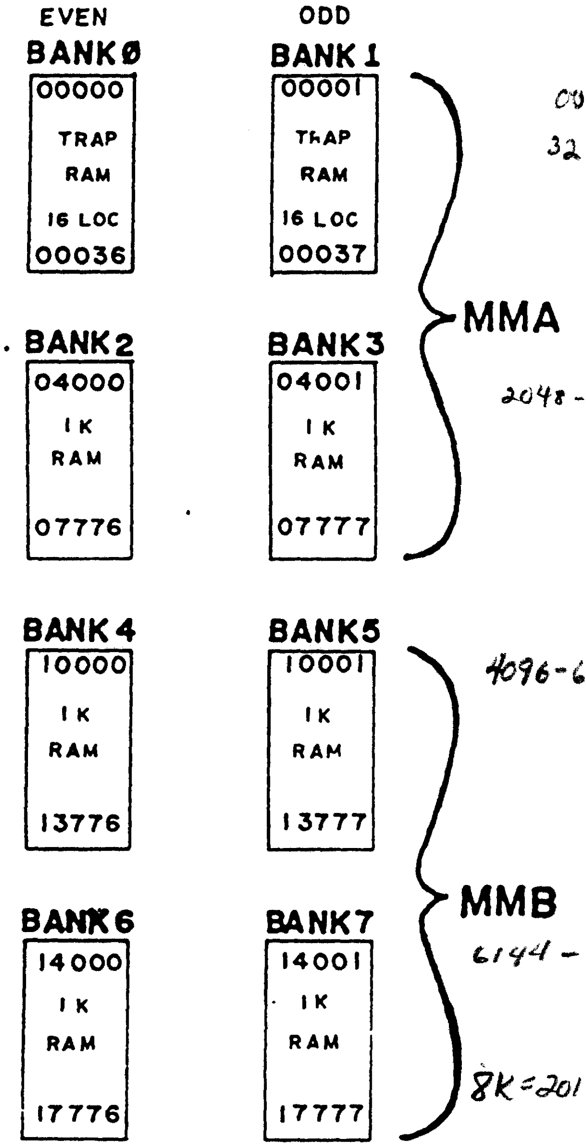
MACDONNELL DOUGLAS

SCALE:	APPROVED BY: Dan MARTIN	DRAWN BY HRK
DATE: 8-22-83		REVISED F.J.A.W.

MICRO-MEMORY ADDRESSING

AUGMENT ENGINE

DRAWING NUMBER



00-31

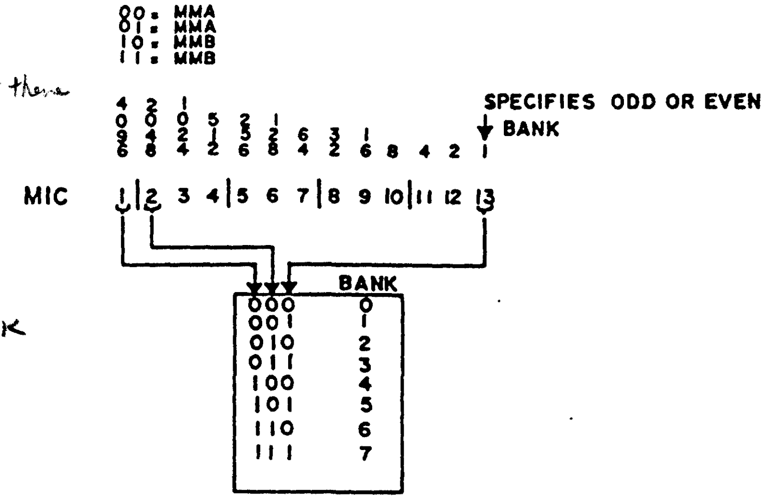
32-2047 = not there

2048-4095 = 2K

4096-6143 = 2K

6144-8191 = 2K

8K = 2016 = 6K + 32



	BANK
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

00-37 TRAP RAM BANK Ø OR BANK 1

40-3777 REALLY DOES NOT EXIST; WRAPS AROUND

MACDONNELL DOUGLAS

SCALE:

DATE: 8-22-83

APPROVED BY: Dan MARTIN

DRAWN BY HRK

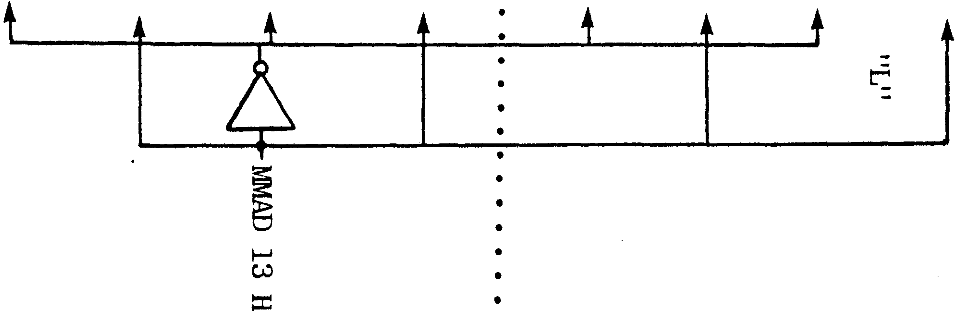
REVISED

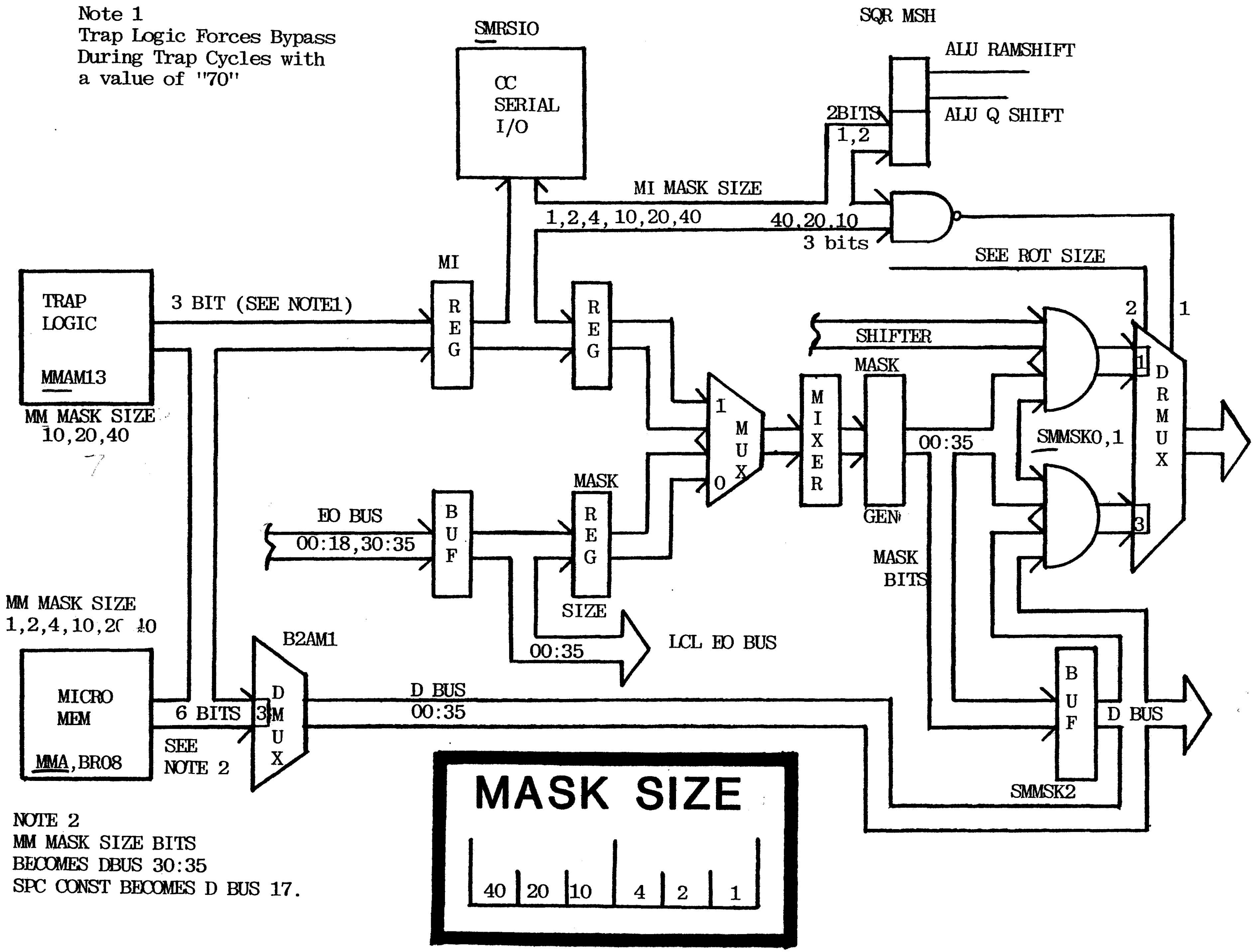
MICRO-MEMORY ADDRESSING

~~[illegible]~~

DRAWING NUMBER

	MMB				MMA			"R"
JADR 12,13 DEST[20,40]	13-46	13-35	13-24	13-13	9-46	9-36	9-18	9-9
JADR 8,9,10,11	15	15	15	15	19	11	11-18	11-9
JADR 4,5,6,7	17	17	17	17	13	13	13-18	13-9
JADR 0,1,2,3	19	19	19	19	15	15	15-18	15-9
MASK SIZE [10,4,2,1]	21	21	21	21	17	17		
MASK SIZE[40,20] ROT SIZ[2,1]	23	23	23	23	19	19		
ROT SIZ[40,20,10,4]	25	25	25	25	21	21		
SPC [10,4,2,1]	27	27	27	27	23	23		
IF, DF CYLEN [2,1]	31	31	31	31	25	25	25-18	25-09
D [2,1] CYLEN[10,4]	33	33	33	33	27	27	27-18	27-09
D [40,20,10,4]	36	36	36	36	31	31	31-18	31-09
AGSEL [4,2,1] DEST [10]	40-46	40-35	40-24	40-13	33-46	33-36	33-18	33-09
BANK #	7	6	5	4	3	2	1	Ø
JCOND [4,2,1,R]	50-46	50-35	50-24	50-13	50-46	50-36	54-18	54-09
JCOND [20,10] SPC[40,20]	52	52	52	52	52	52	56-18	56-09
JCODE [10,4,2,1]	54	54	54	54	54	54		
DEST [4,2,1] SPARE	56	56	56	56	56	56		
MAPF [10,4,2,1]	58	58	58	58	58	58		
ALUDST [2,1] ALU1 SEL LD AR	60	60	60	60	60	60		
ALUDST [4] ALU FUN[4,2,1]	62	62	62	62	62	62	62-18	62-09
ALU SRC[4,2,1] CARRY IN	64	64	64	64	64	64	64-18	64-09
AA from MA-MEM WAIT-PAR-SP1	66	66	66	66	66	66	66-18	66-09
IDISP RQ,LD MA, END EA LEFT EA FROM OBUS	68-48	68-35	68-24	68-13	68-46	68-36	68-18	68-09

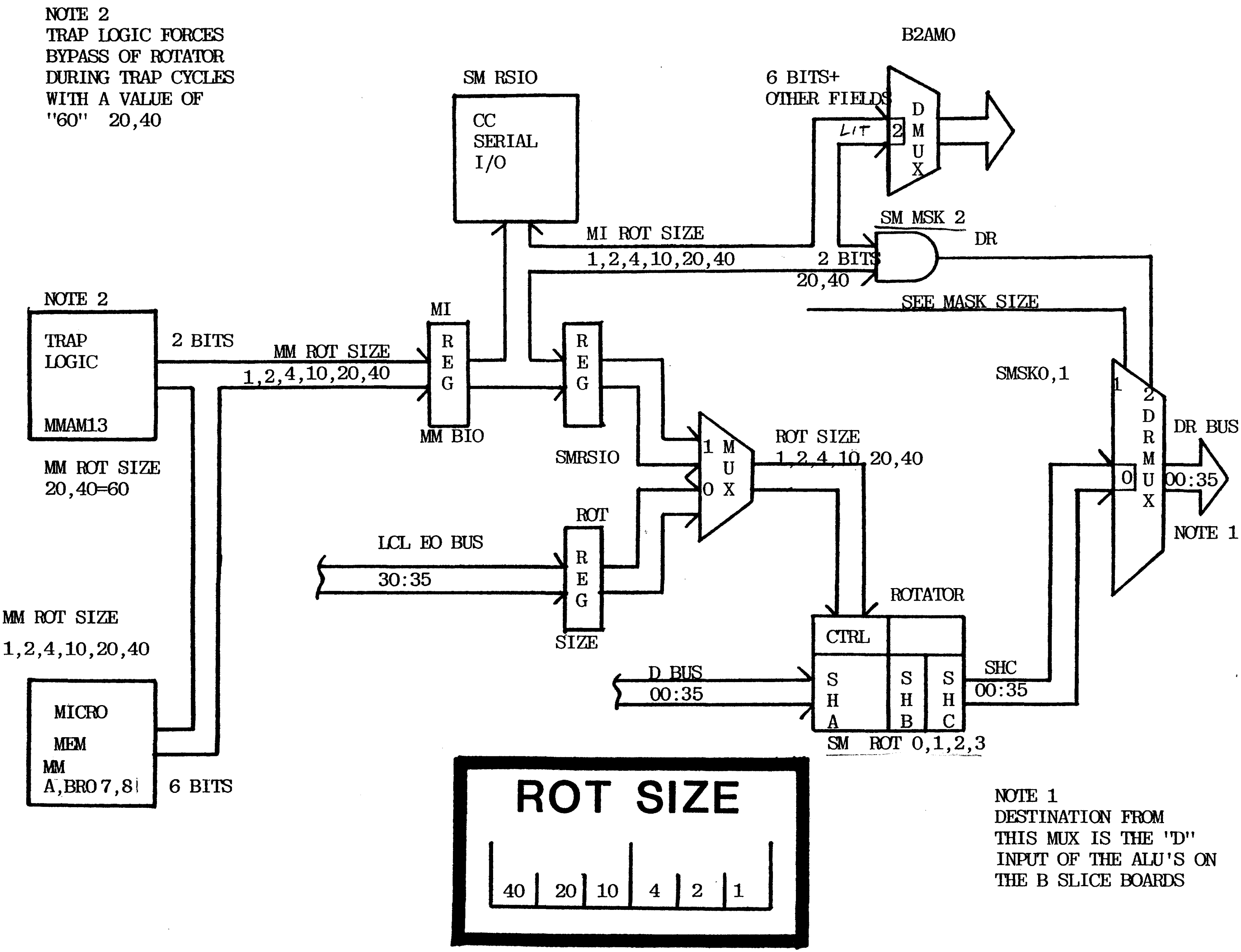




Note 1
Trap Logic Forces Bypass
During Trap Cycles with
a value of "70"

NOTE 2
MM MASK SIZE BITS
BECOMES DBUS 30:35
SPC CONST BECOMES D BUS 17.

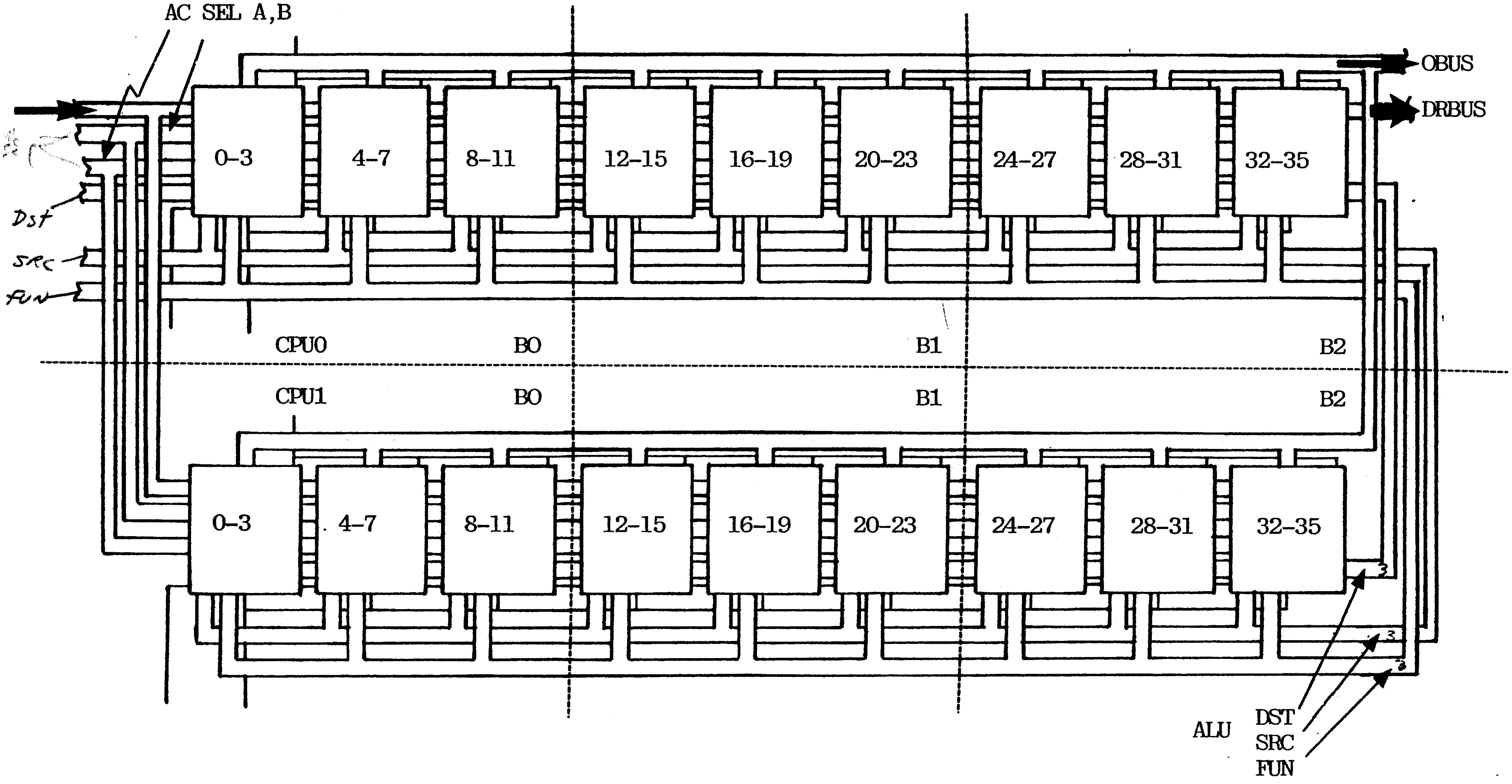
MASK SIZE



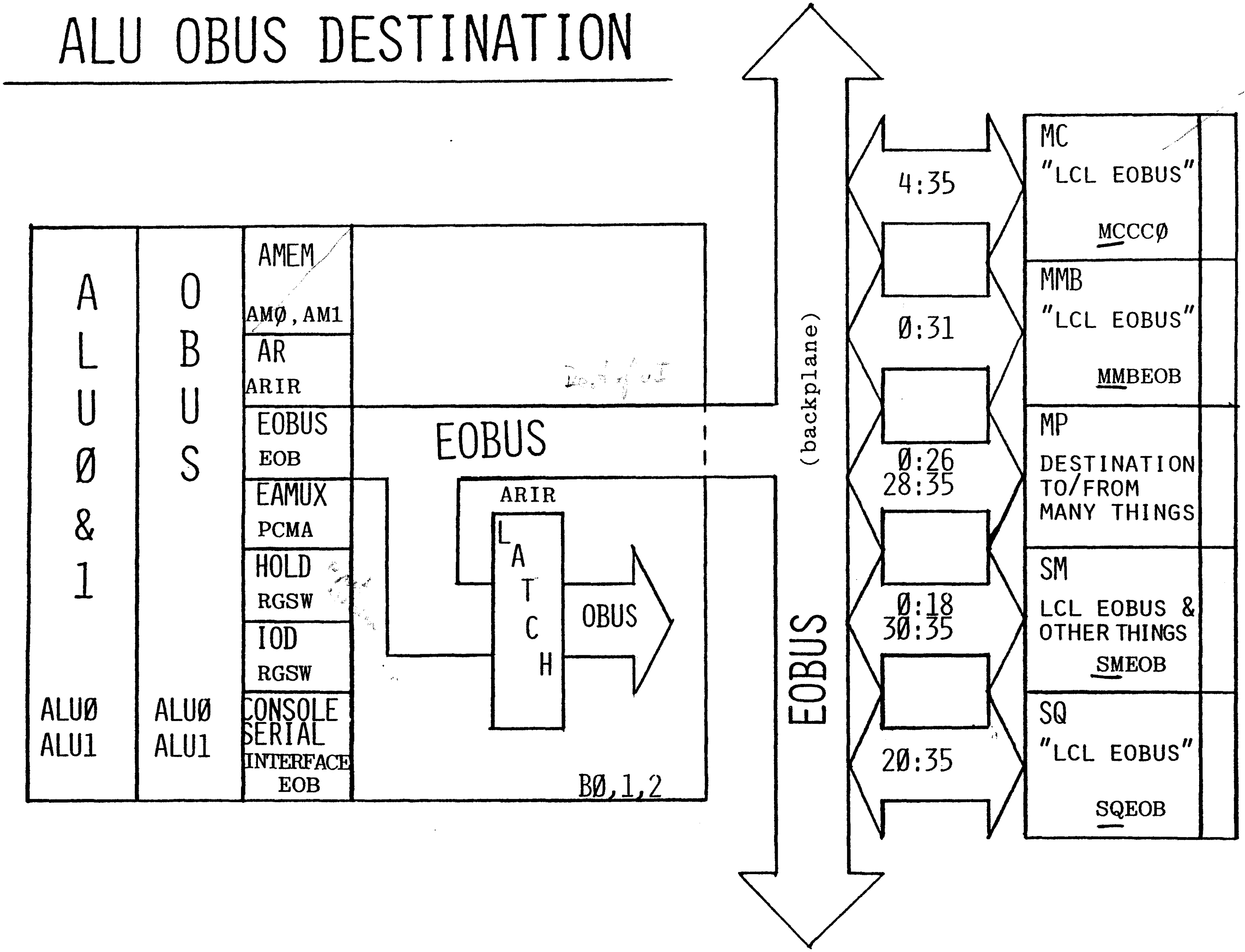
NOTE 2
TRAP LOGIC FORCES
BYPASS OF ROTATOR
DURING TRAP CYCLES
WITH A VALUE OF
"60" 20,40

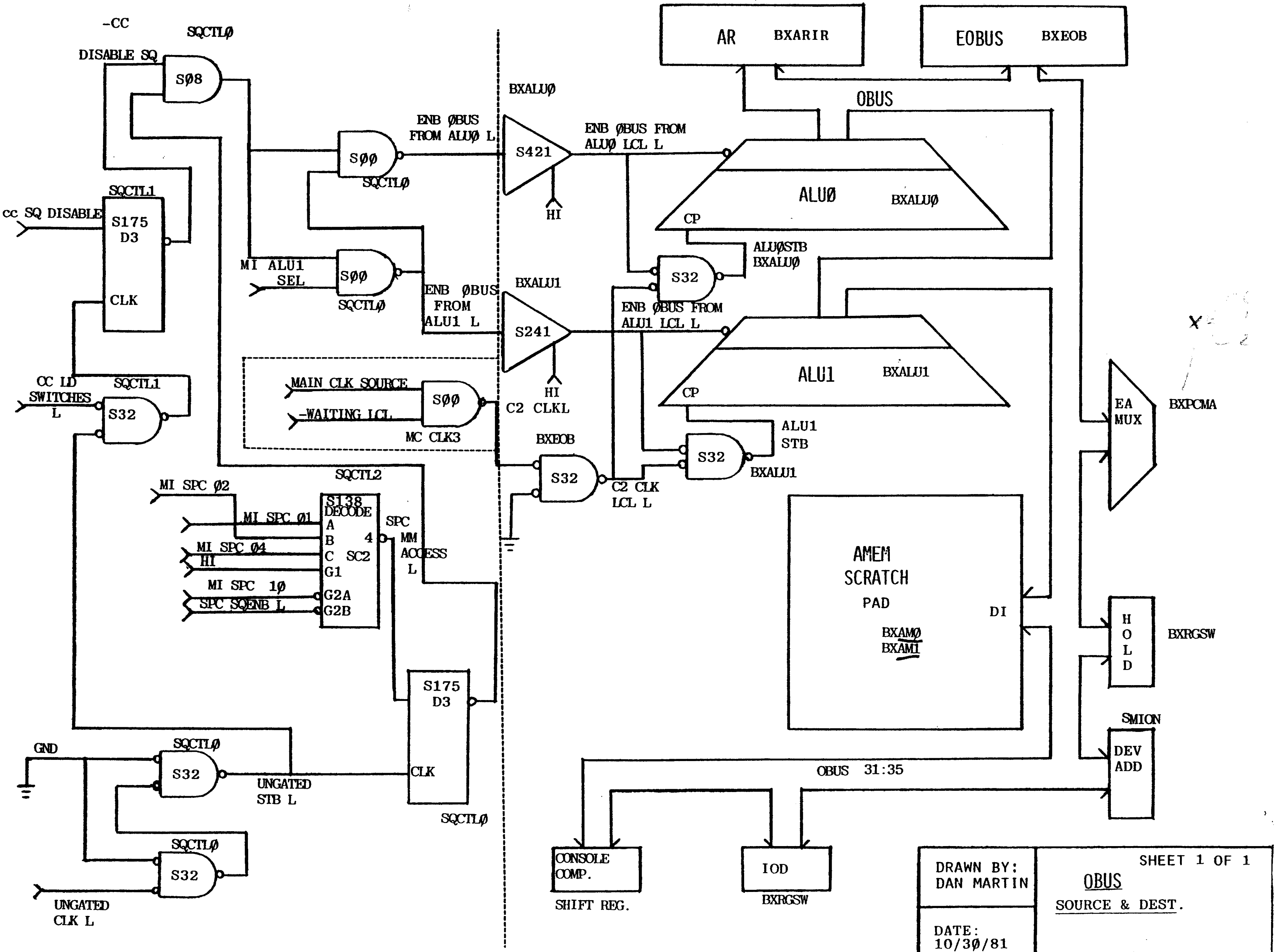
NOTE 1
DESTINATION FROM
THIS MUX IS THE 'D"
INPUT OF THE ALU'S ON
THE B SLICE BOARDS

OVERALL ALU BLOCK



ALU OBUS DESTINATION





DRAWN BY:
DAN MARTIN

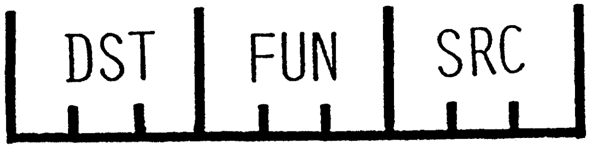
SHEET 1 OF 1

OBUS
SOURCE & DEST.

DATE:
10/3Ø/81

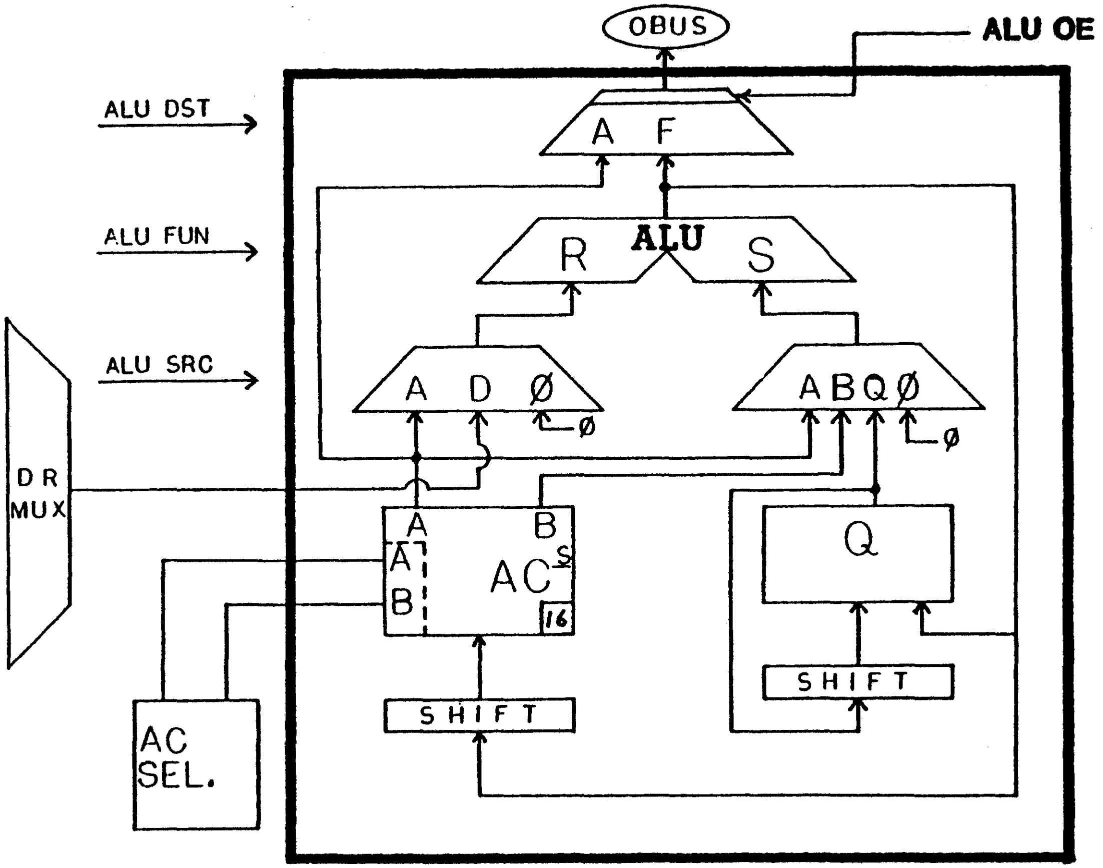
MICROCODE

ALU FIELD



ALU CONTROL

	DST	FUN		SRC
			Ch	R S
Ø	F→OBUS →Q	R+S	(+1)H	A , Q
1	F→OBUS ©	S–R	(–1)L	A , B
2	A→OBUS F→AC(B)	R–S	(–1)L	Ø , Q
3	F→OBUS →AC(B)	R∨S		Ø , B
4	F→OBUS F/2→AC(B) Q/2→Q	R∧S Ⓑ		Ø , A Ⓐ
5	F→OBUS F/2→AC(B)	$\bar{R}$∧S		D , A
6	F→OBUS 2F→AC(B) 2Q→Q	R⊻S		D , Q
7	F→OBUS 2F→AC(B)	$\overline{R \veebar S}$		D , Ø



NOTES:

1. The ALU DST,FUN,SRC Controls are 3 lines each for a total o 9 lines. They relate directly to the 9 bits in the microcode "ALU" field.
2. An "ALU" field value of 144 would do the following:
   Ⓐ. Select"Ø"as R source (Ø means R source is unused) Select"A" output of AC as S source
   Ⓑ. Perform R∧S function (passes "A" through S selector since R is unused)
   ©. Passes "A" output of AC through F and onto OBUS.

∧ =AND ∨ =OR ⊻= EXCLUSIVE OR

MICROCODE ACSEL Field

4	2	1

The AM2901B accumulators (AC^{s}) are numbered 0-17 for a total of 16 registers.

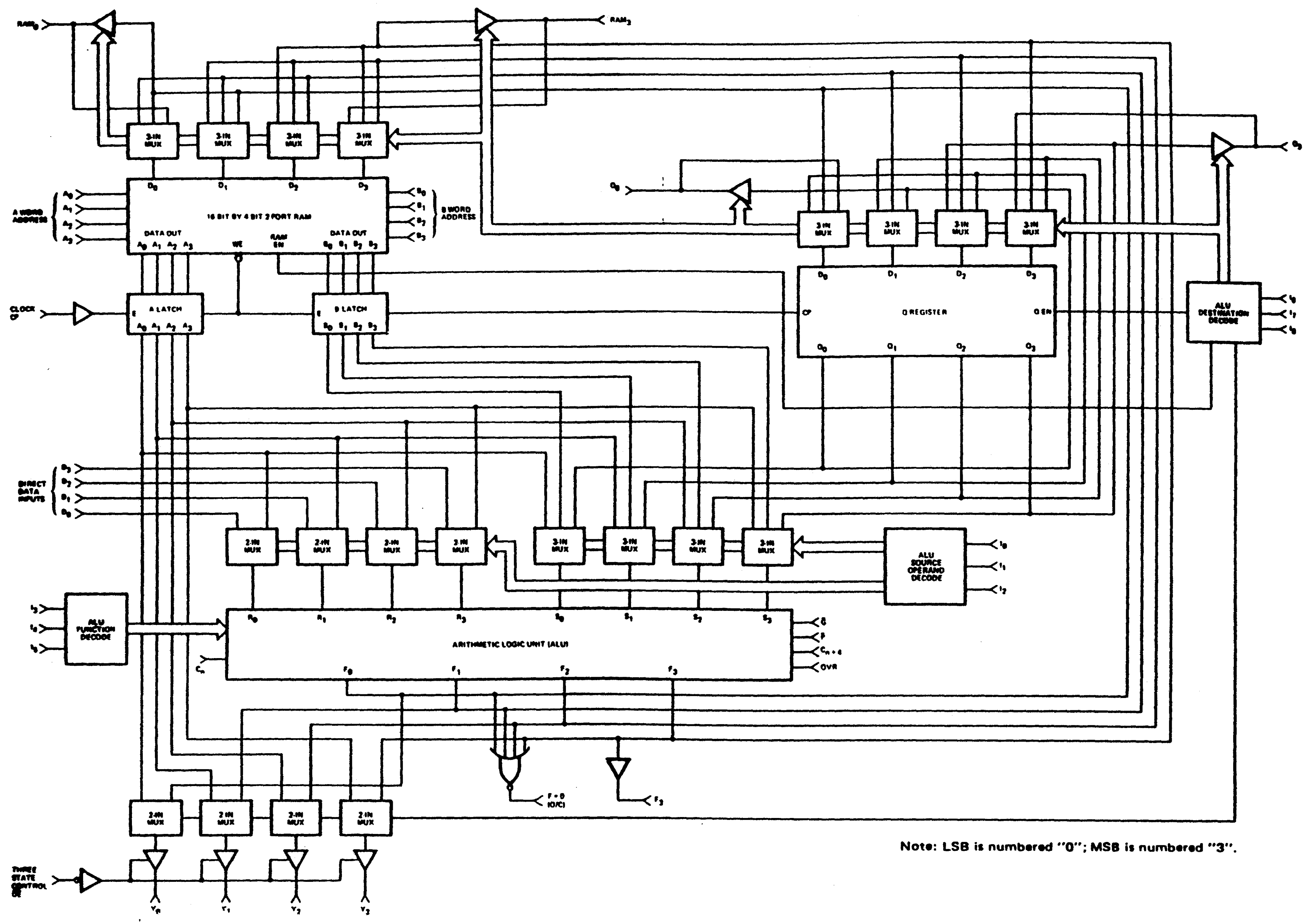
The AC^{s} have two sets of address lines (A&B) and two 36 bit data outputs (A&B). There is also one data input (Din).

"A" addresses are read only. "B" addresses may be read or write depending on the ALU "DST" microcode field.

Any two accumulators addressed by A&B address lines may be displayed simultaneously on the A&B data outputs <u>or</u> if the ALU "DST" field is octal 2-7. then the AC addresses by "A" address lines will be displayed on the "A" data output and the AC addressed by the "B" address will be written into via "Din".

"A"&"B" addresses may be the same. The octal code in the microcode ACSEL field specifies the source for "A"&"B" accumulator addresses.

	ADDRESS SOURCES	
Value (octal)	A	B
0	MA 32:35	IR 09:12
1	IR 09:12	MA 09:12
2	IR 14:17	IR 14:17
3	IR 09:12+1	
4	IR 02:05	
5	IR 09:12	
6	D 10:01	AMEM-P
7	D 10:01	DEST 10:01



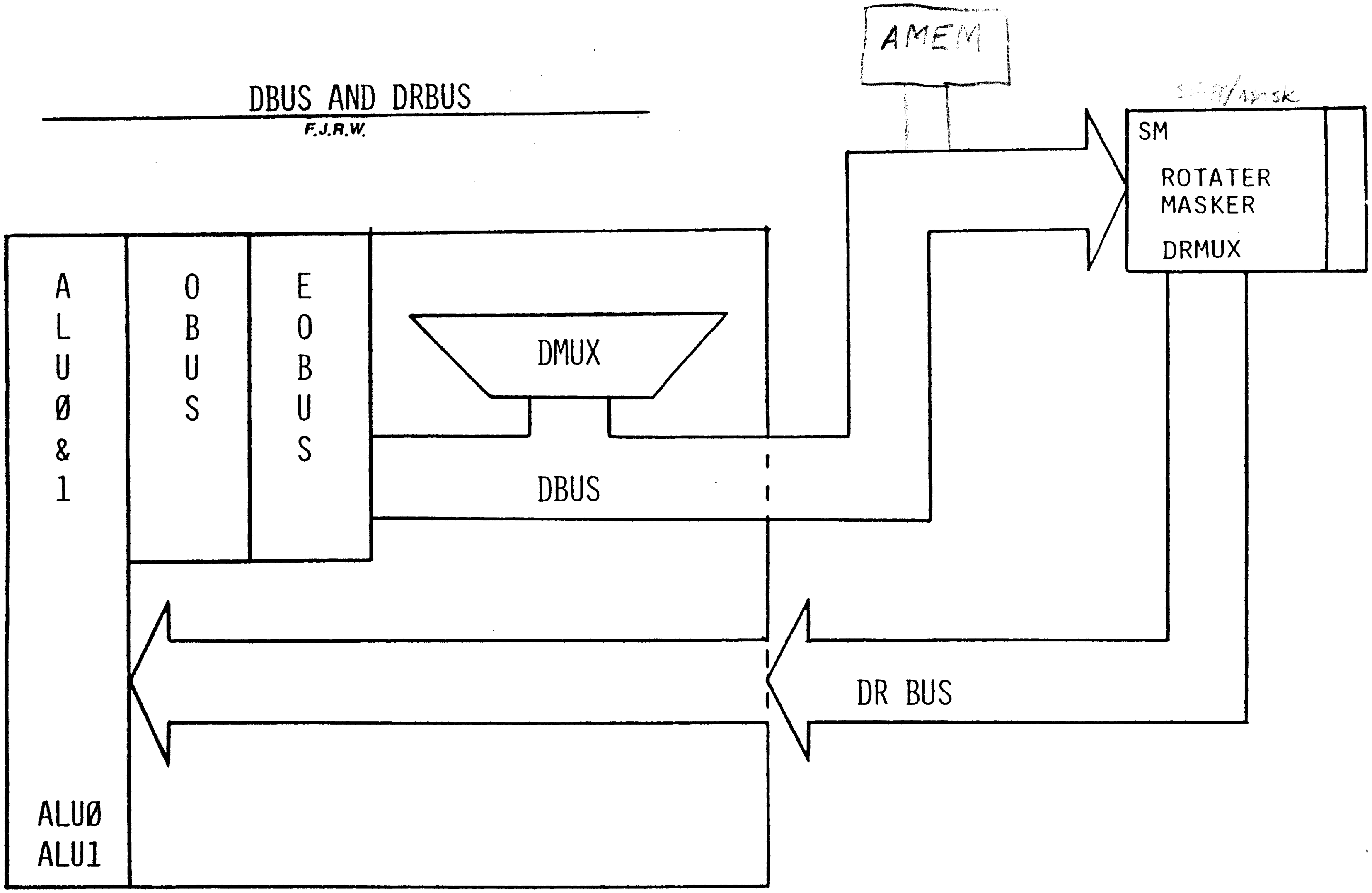
Note: LSB is numbered "0"; MSB is numbered "3".

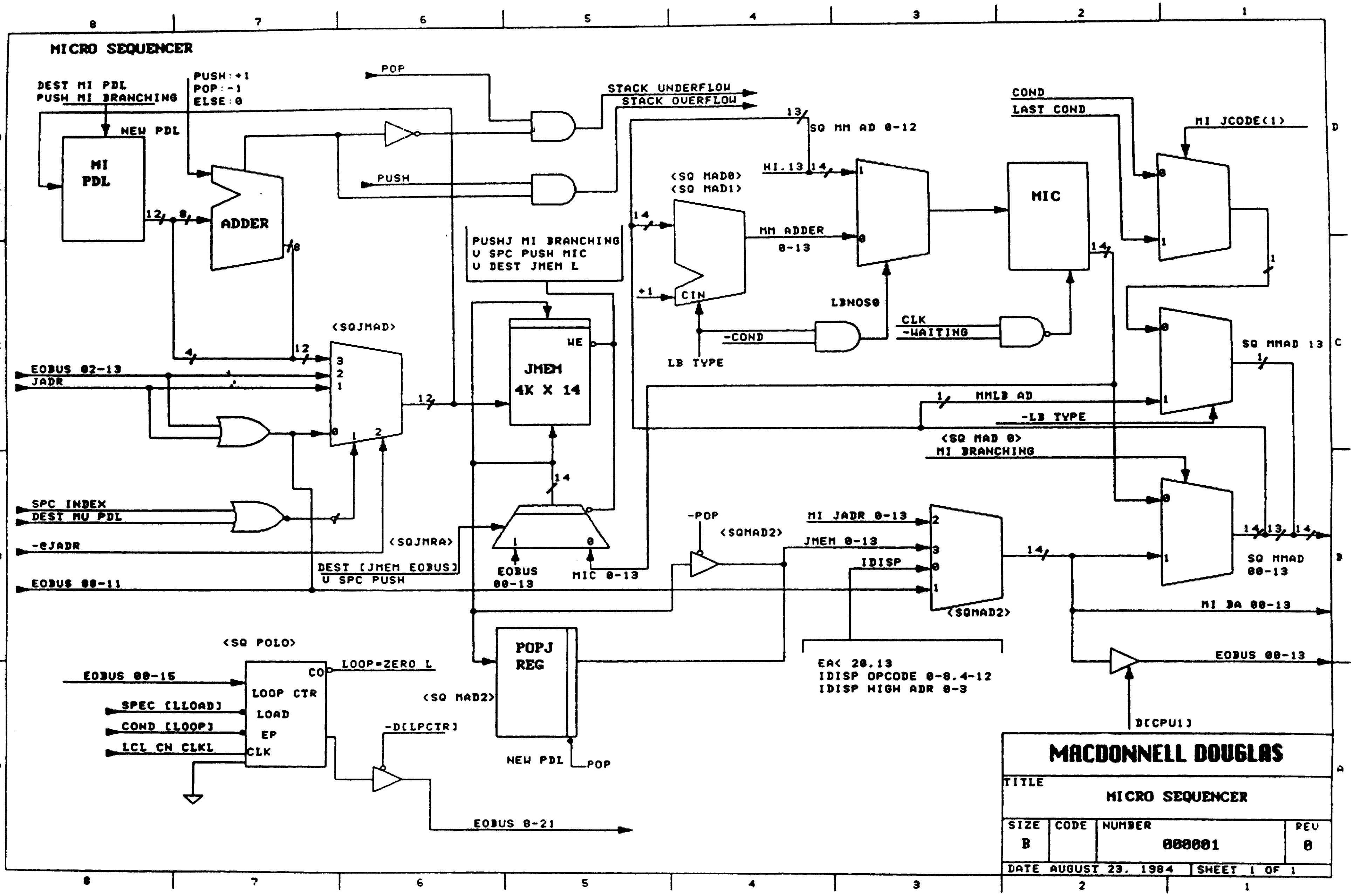
Figure 1. Detailed Am2901B Microprocessor Block Diagram.

MPR-005

DBUS AND DRBUS

F.J.R.W.



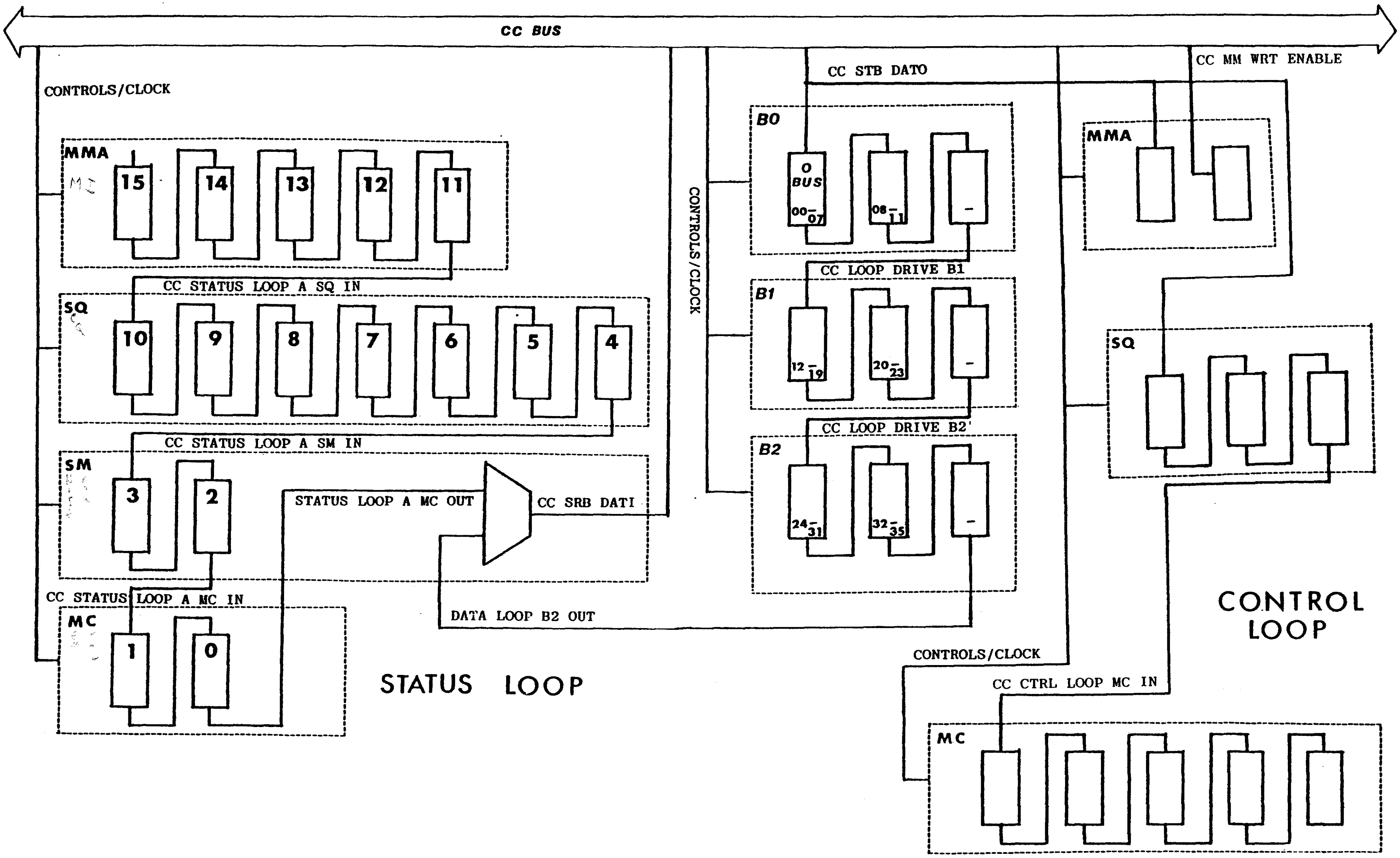


MACDONNELL DOUGLAS

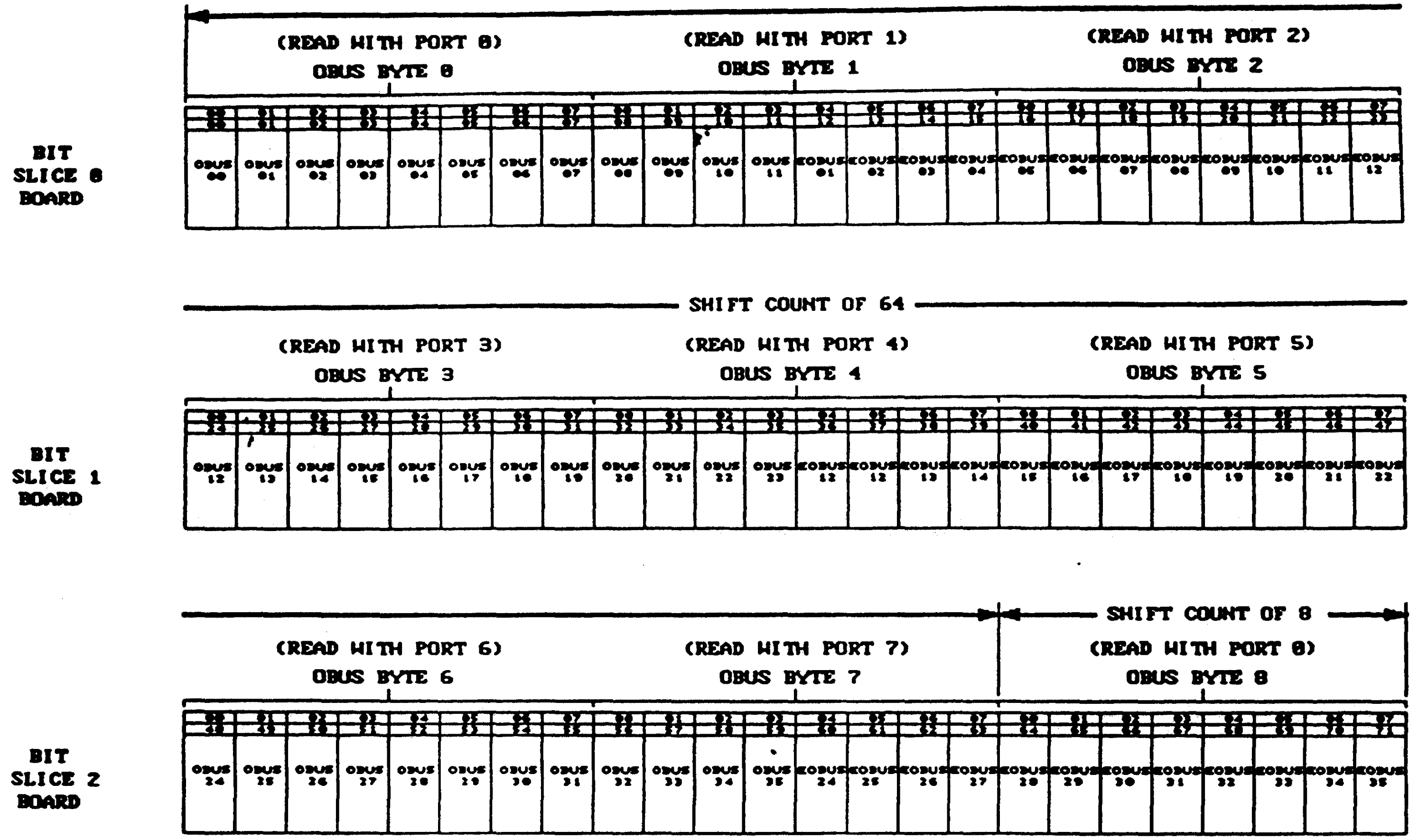
TITLE MICRO SEQUENCER

SIZE	CODE	NUMBER	REV
B		000001	0

DATE AUGUST 23, 1984 | SHEET 1 OF 1


CC BUS
CONTROLS/CLOCK
MMA
15
14
13
12
11
CC STATUS LOOP A SQ IN
SQ
10
9
8
7
6
5
4
CC STATUS LOOP A SM IN
SM
3
2
STATUS LOOP A MC OUT
CC SRB DATI
CC STATUS LOOP A MC IN
MC
1
0
DATA LOOP B2 OUT
STATUS LOOP
CONTROLS/CLOCK
CC STB DATO
B0
O BUS
00-07
08-11
-
CC LOOP DRIVE B1
B1
12-19
20-23
-
CC LOOP DRIVE B2'
B2
24-31
32-35
-
CC MM WRT ENABLE
MMA
SQ
CONTROL LOOP
CONTROLS/CLOCK
CC CTRL LOOP MC IN
MC

CCOBUSLP.DWG



BIT SLICE 0 BOARD

00	01	02	03	04	05	06	07	00	01	02	03	04	05	06	07	00	01	02	03	04	05	06	07
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
OBUS 00	OBUS 01	OBUS 02	OBUS 03	OBUS 04	OBUS 05	OBUS 06	OBUS 07	OBUS 08	OBUS 09	OBUS 10	OBUS 11	EOBUS 01	EOBUS 02	EOBUS 03	EOBUS 04	EOBUS 05	EOBUS 06	EOBUS 07	EOBUS 08	EOBUS 09	EOBUS 10	EOBUS 11	EOBUS 12

BIT SLICE 1 BOARD

00	01	02	03	04	05	06	07	00	01	02	03	04	05	06	07	00	01	02	03	04	05	06	07
24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
OBUS 12	OBUS 13	OBUS 14	OBUS 15	OBUS 16	OBUS 17	OBUS 18	OBUS 19	OBUS 20	OBUS 21	OBUS 22	OBUS 23	EOBUS 12	EOBUS 12	EOBUS 13	EOBUS 14	EOBUS 15	EOBUS 16	EOBUS 17	EOBUS 18	EOBUS 19	EOBUS 20	EOBUS 21	EOBUS 22

BIT SLICE 2 BOARD

00	01	02	03	04	05	06	07	00	01	02	03	04	05	06	07	00	01	02	03	04	05	06	07
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71
OBUS 24	OBUS 25	OBUS 26	OBUS 27	OBUS 28	OBUS 29	OBUS 30	OBUS 31	OBUS 32	OBUS 33	OBUS 34	OBUS 35	EOBUS 24	EOBUS 25	EOBUS 26	EOBUS 27	EOBUS 28	EOBUS 29	EOBUS 30	EOBUS 31	EOBUS 32	EOBUS 33	EOBUS 34	EOBUS 35

MACDONNELL DOUGLAS

TITLE
CC OBUS LOOP

SIZE	CODE	NUMBER	REV
B			A

DATE	AUGUST 28,1984	SHEET 01 OF 01

CCSTAT.DWG

STATUS BYTE 0

00	01	02	03	04	05	06	07
00	01	02	03	04	05	06	07
MI SP 01	MI PAR	MI SEL AA FROM MA	MI MEM WAIT	MI IDISP RQ	MI LD MA	MI EA FROM DBUS	MI END EA LEFT

STATUS BYTE 1

00	01	02	03	04	05	06	07
08	09	10	11	12	13	14	15
MI ALU DST [4]	MI ALU FUN [1]	MI ALU FUN [2]	MI ALU FUN [4]	MI ALU SRC [1]	MI ALU SRC [2]	MI ALU SRC [4]	MI CARRY IN

STATUS BYTE 2

00	01	02	03	04	05	06	07
16	17	18	19	20	21	22	23
MI JCOND [02]	MI JCOND [04]	MI JCOND [10]	MI JCOND [20]	MI LD AR	MI ALU1 SEL	MI ALU DST [1]	MI ALU DST [2]

STATUS BYTE 3

00	01	02	03	04	05	06	07
24	25	26	27	28	29	30	31
MI SPC [01]	MI SPC [02]	MI SPC [04]	MI SPC [10]	MI SPC [20]	MI SPC [40]	MI JCOND R	MI JCOND [01]

STATUS BYTE 4

00	01	02	03	04	05	06	07
32	33	34	35	36	37	38	39
NC	NC	NC	MI JCODE [01]	MI JCODE [02]	MI JCODE [04]	MI JCODE [10]	MI SP 02

STATUS BYTE 5

00	01	02	03	04	05	06	07
40	41	42	43	44	45	46	47
MI IF RQ	MI DF RQ	MI JADR 00	MI JADR 01	MI JADR 02	MI JADR 03	MI JADR 04	MI JADR 05

STATUS BYTE 6

00	01	02	03	04	05	06	07
48	49	50	51	52	53	54	55
MI JADR 06	MI JADR 07	MI JADR 08	MI JADR 09	MI JADR 10	MI JADR 11	MI JADR 12	MI JADR 13

STATUS BYTE 7

00	01	02	03	04	05	06	07
56	57	58	59	60	61	62	63
MI DEST [40]	MI DEST [20]	MI DEST [10]	MI DEST [04]	MI DEST [02]	MI DEST [01]	MI ACSEL [4]	MI ACSEL [2]

STATUS BYTE 8

00	01	02	03	04	05	06	07
64	65	66	67	68	69	70	71
MI ACSEL [1]	MI D [40]	MI D [20]	MI D [10]	MI D [04]	MI D [02]	MI D [01]	NC

STATUS BYTE 9

00	01	02	03	04	05	06	07
72	73	74	75	76	77	78	79
MIC 00	MIC 01	MIC 02	MIC 03	MIC 04	MIC 05	MIC 06	MIC 07

STATUS BYTE 10

00	01	02	03	04	05	06	07
80	81	82	83	84	85	86	87
MIC 08	MIC 09	MIC 10	MIC 11	MIC 12	MIC 13	-MI BRANCHING D	-CPU CONT

STATUS BYTE 11

00	01	02	03	04	05	06	07
88	89	90	91	92	93	94	95
NC	NC	NC	NC	NC	NC	NC	NC

STATUS BYTE 12

00	01	02	03	04	05	06	07
96	97	98	99	100	101	102	103
MI MASK SIZE [40]	MI MASK SIZE [20]	MI MASK SIZE [10]	MI MASK SIZE [04]	MI MASK SIZE [02]	MI MASK SIZE [01]	NC	NC

STATUS BYTE 13

00	01	02	03	04	05	06	07
104	105	106	107	108	109	110	111
MI ROT SIZE [40]	MI ROT SIZE [20]	MI ROT SIZE [10]	MI ROT SIZE [04]	MI ROT SIZE [02]	MI ROT SIZE [01]	NC	NC

STATUS BYTE 14

00	01	02	03	04	05	06	07
112	113	114	115	116	117	118	119
NC	NC	NC	NC	NC	NC	NC	NC

STATUS BYTE 15

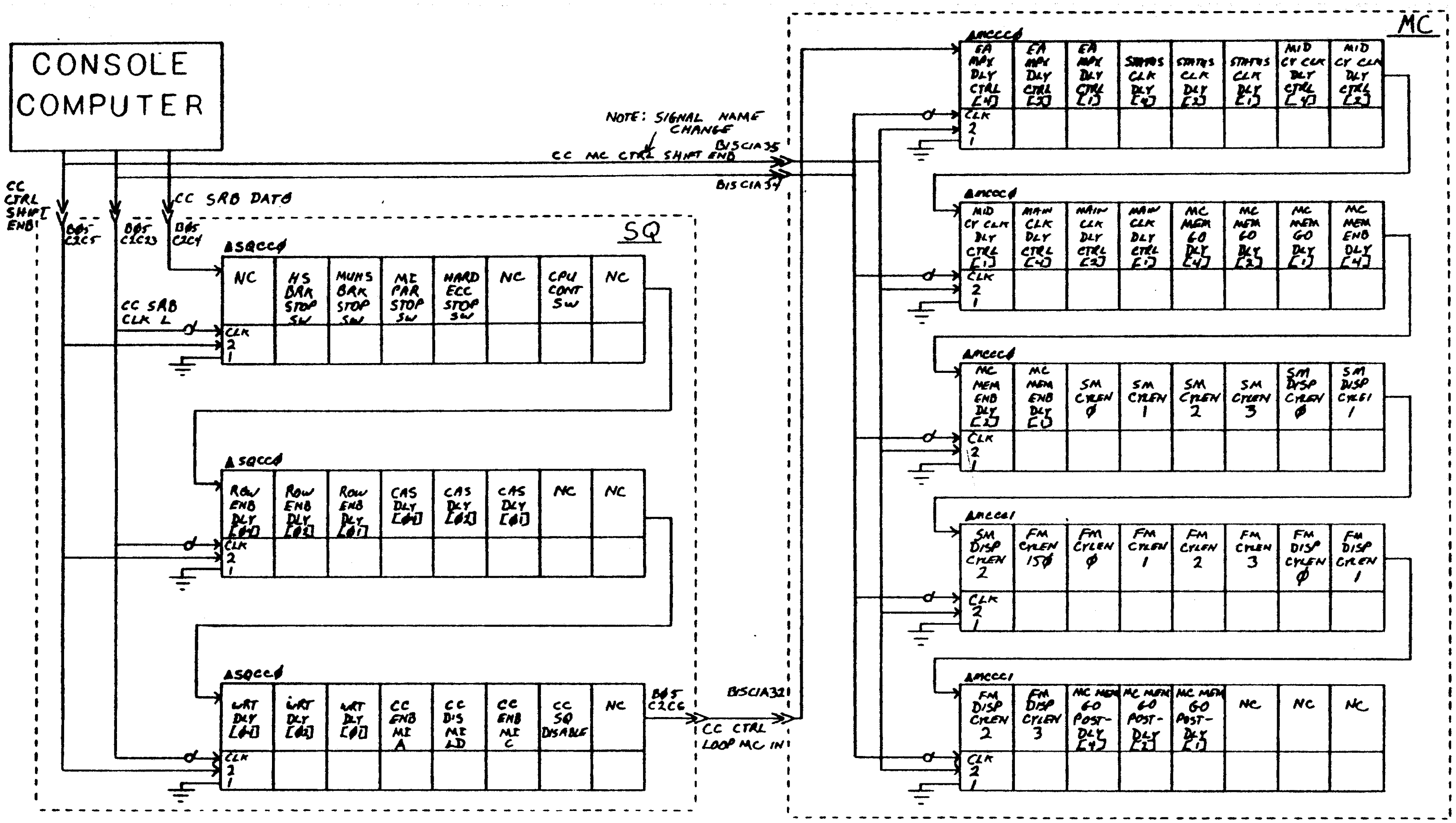
00	01	02	03	04	05	06	07
120	121	122	123	124	125	126	127
MI CYLEN [10]	MI CYLEN [04]	MI CYLEN [02]	MI CYLEN [01]	MI MAPF [10]	MI MAPF [04]	MI MAPF [02]	MI MAPF [01]

MACDONNELL DOUGLAS

TITLE CC STATUS LOOP

SIZE	CODE	NUMBER	REV
B			A

DATE AUGUST 20,1984 SHEET 01 OF 01



SYSTEM XXVI

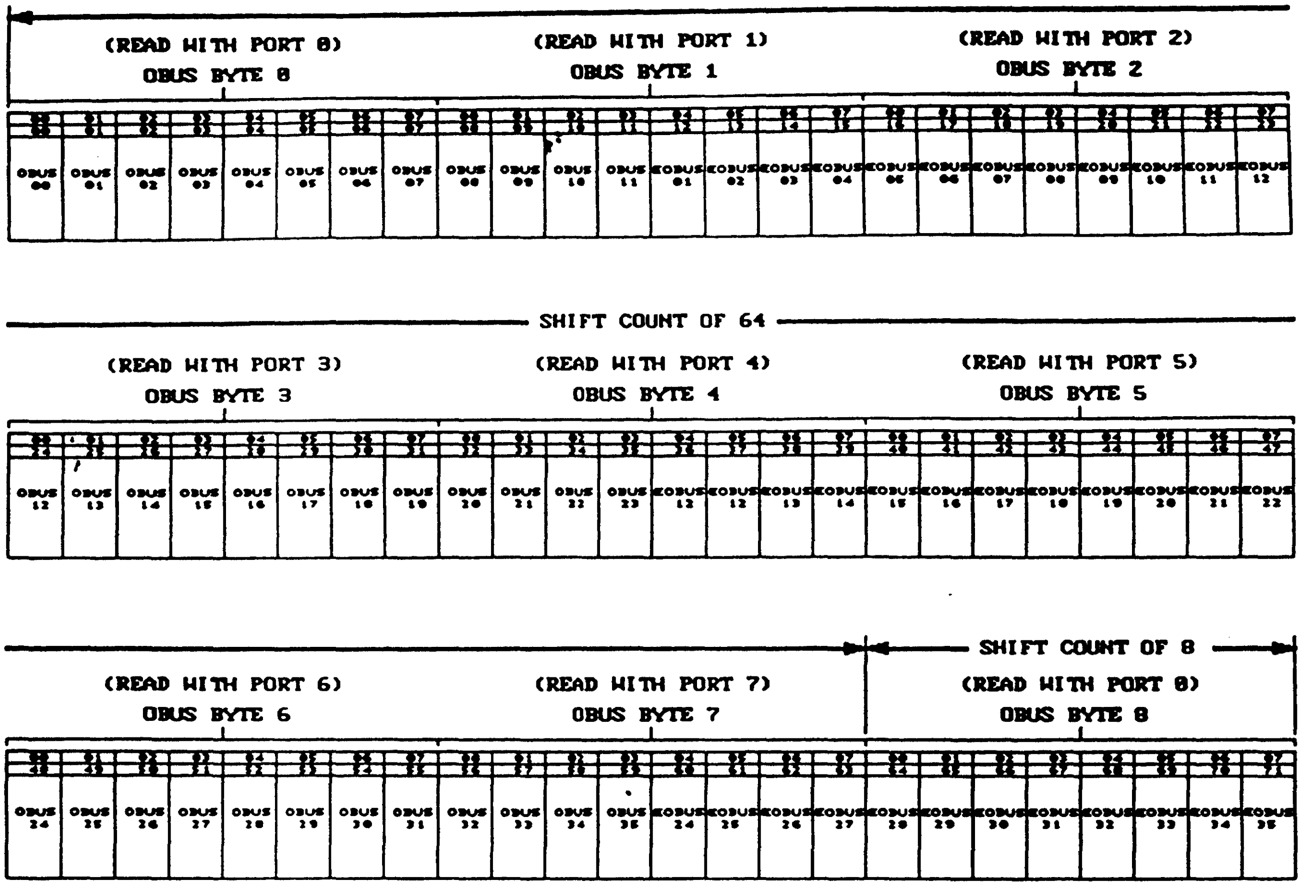
SCALE:	APPROVED BY:	DRAWN BY CDM
DATE: 3-23-82		REVISED

CONSOLE COMPUTER
CTRL SWITCHES

DRAWING NUMBER
1 OF 1

11X17 PRINTED ON NO. 1000H-8 CLEARPRINT FADE-OUT

CCOBUSLP.DWG



BIT SLICE 0 BOARD

	(READ WITH PORT 0) OBUS BYTE 0								(READ WITH PORT 1) OBUS BYTE 1								(READ WITH PORT 2) OBUS BYTE 2							
	00	01	02	03	04	05	06	07	00	01	02	03	04	05	06	07	00	01	02	03	04	05	06	07
	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
	OBUS 00	OBUS 01	OBUS 02	OBUS 03	OBUS 04	OBUS 05	OBUS 06	OBUS 07	OBUS 08	OBUS 09	OBUS 10	OBUS 11	EOBUS 01	EOBUS 02	EOBUS 03	EOBUS 04	EOBUS 05	EOBUS 06	EOBUS 07	EOBUS 08	EOBUS 09	EOBUS 10	EOBUS 11	EOBUS 12

BIT SLICE 1 BOARD

	(READ WITH PORT 3) OBUS BYTE 3								(READ WITH PORT 4) OBUS BYTE 4								(READ WITH PORT 5) OBUS BYTE 5							
	00	01	02	03	04	05	06	07	00	01	02	03	04	05	06	07	00	01	02	03	04	05	06	07
	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
	OBUS 12	OBUS 13	OBUS 14	OBUS 15	OBUS 16	OBUS 17	OBUS 18	OBUS 19	OBUS 20	OBUS 21	OBUS 22	OBUS 23	EOBUS 12	EOBUS 12	EOBUS 13	EOBUS 14	EOBUS 15	EOBUS 16	EOBUS 17	EOBUS 18	EOBUS 19	EOBUS 20	EOBUS 21	EOBUS 22

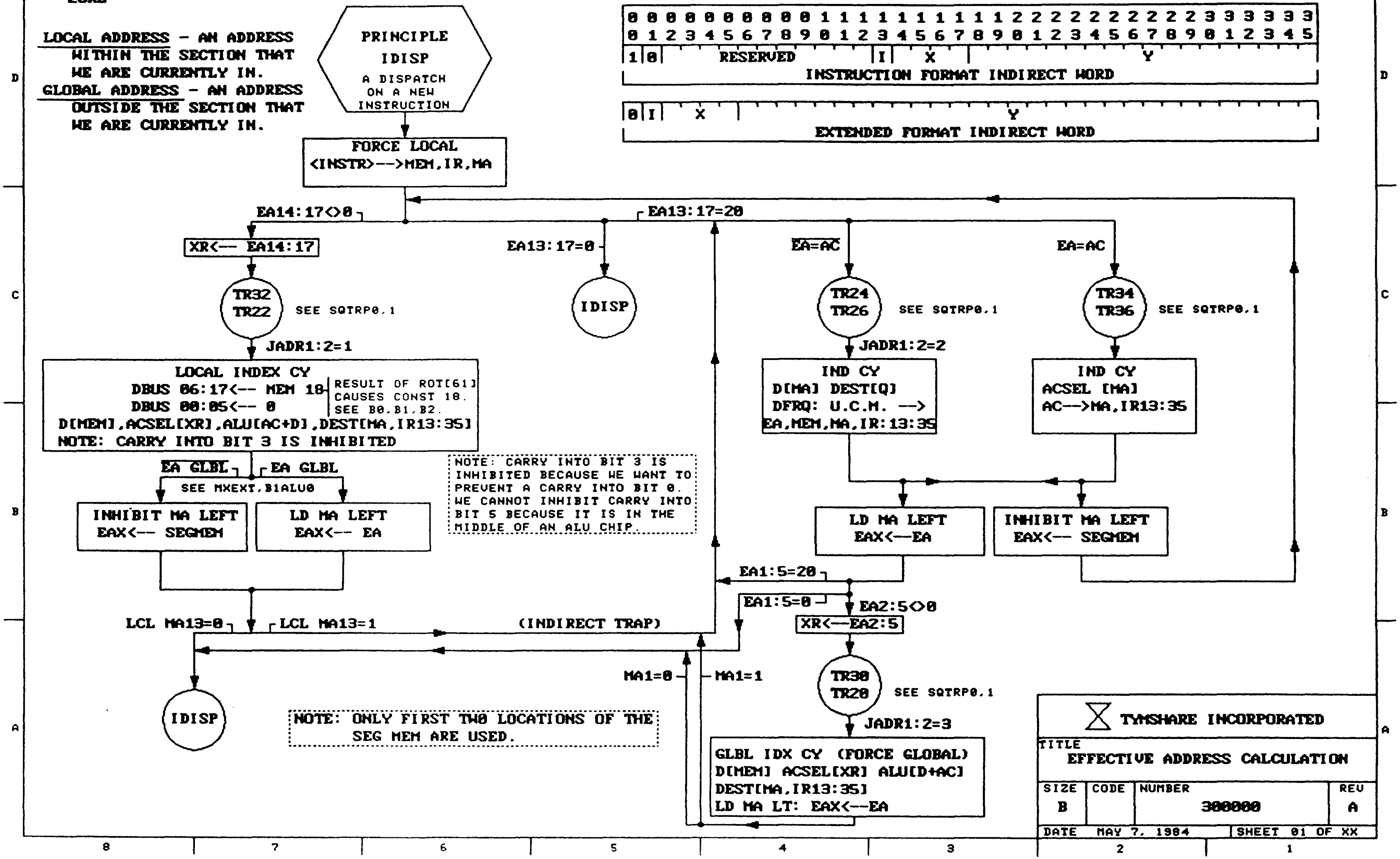
BIT SLICE 2 BOARD

	(READ WITH PORT 6) OBUS BYTE 6								(READ WITH PORT 7) OBUS BYTE 7								(READ WITH PORT 8) OBUS BYTE 8							
	00	01	02	03	04	05	06	07	00	01	02	03	04	05	06	07	00	01	02	03	04	05	06	07
	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71
	OBUS 24	OBUS 25	OBUS 26	OBUS 27	OBUS 28	OBUS 29	OBUS 30	OBUS 31	OBUS 32	OBUS 33	OBUS 34	OBUS 35	EOBUS 24	EOBUS 25	EOBUS 26	EOBUS 27	EOBUS 28	EOBUS 29	EOBUS 30	EOBUS 31	EOBUS 32	EOBUS 33	EOBUS 34	EOBUS 35

MACDONNELL DOUGLAS			
TITLE CC OBUS LOOP			
SIZE B	CODE	NUMBER	REV A
DATE AUGUST 28.1984		SHEET 01 OF 01	

EFFADD.DWG
26KL

LOCAL ADDRESS - AN ADDRESS WITHIN THE SECTION THAT WE ARE CURRENTLY IN.
GLOBAL ADDRESS - AN ADDRESS OUTSIDE THE SECTION THAT WE ARE CURRENTLY IN.

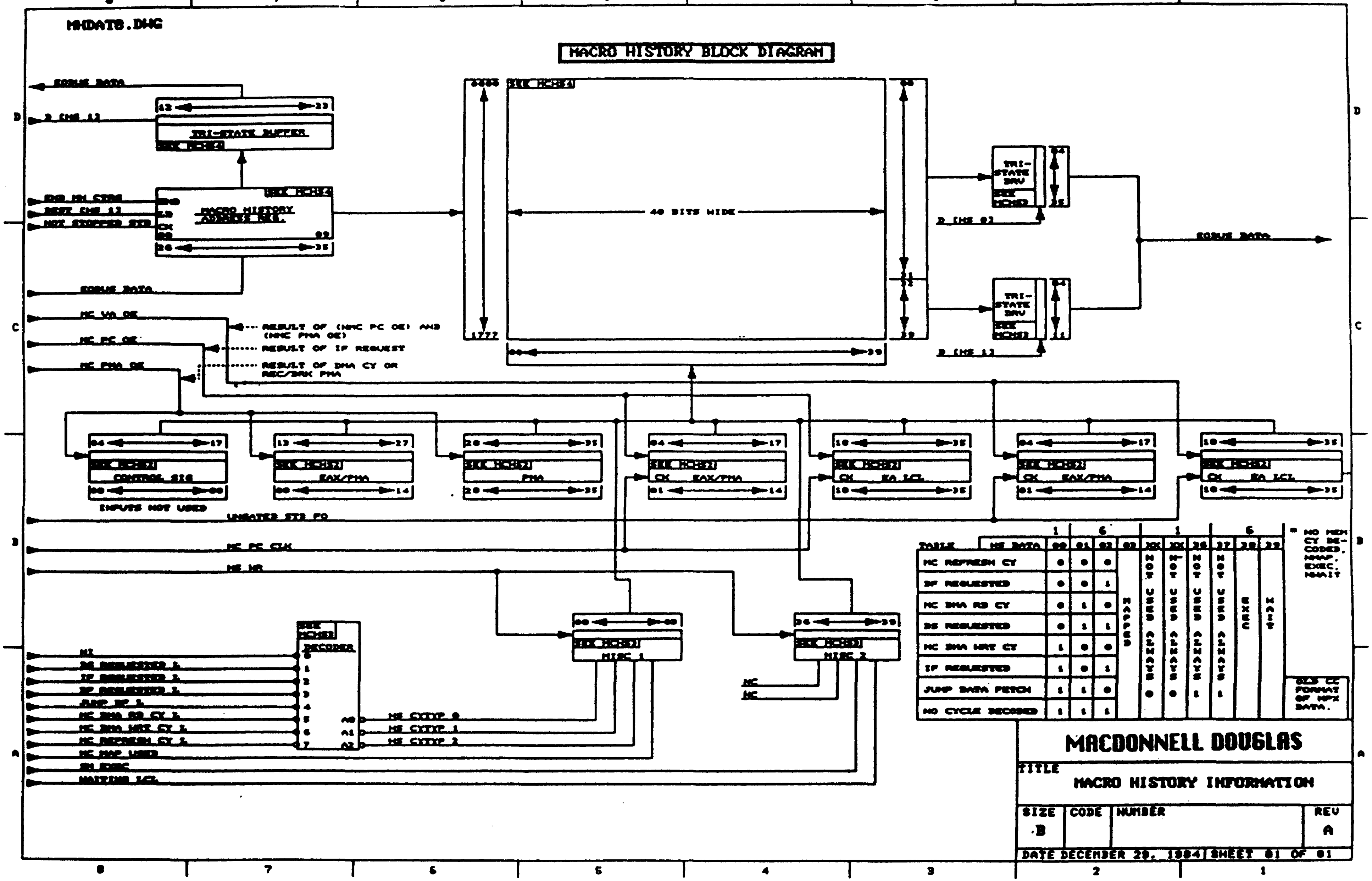


TYMSHARE INCORPORATED

TITLE			
EFFECTIVE ADDRESS CALCULATION			
SIZE	CODE	NUMBER	REV
B		300000	A
DATE MAY 7, 1984		SHEET 01 OF XX	

MHDAT8.DWG

MACRO HISTORY BLOCK DIAGRAM



		1	6			1			6		
TABLE	MX DATA	00	01	02	03	XX	XX	26	37	38	39
	MC REFRESH CY	0	0	0	MAPPED	NOT USED ALWAYS 0	NOT USED ALWAYS 0	NOT USED ALWAYS 1	NOT USED ALWAYS 1	EXEC	WAIT
	DF REQUESTED	0	0	1							
	MC DMA RD CY	0	1	0							
	DS REQUESTED	0	1	1							
	MC DMA WRT CY	1	0	0							
	IF REQUESTED	1	0	1							
	JUMP DATA FETCH	1	1	0							
	NO CYCLE DECODED	1	1	1							

* NO MEM CY DECODED, MMAP, EXEC, WAIT

OLD CC FORMAT OF MPX DATA.

MACDONNELL DOUGLAS

TITLE MACRO HISTORY INFORMATION

SIZE	CODE	NUMBER	REV
B			A

DATE DECEMBER 29, 1984 SHEET 01 OF 01

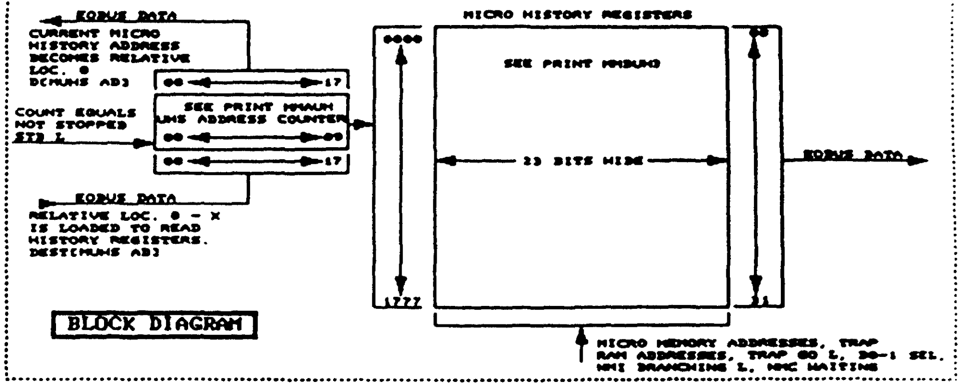
UHDAT8.DWG

LOADING AND READING A MICRO HISTORY ADDRESS

WEIGHT	512	256	128	64	32	16	8	4	2	1
EOBUS BIT	08	09	10	11	12	13	14	15	16	17
	MUHS AD 00	MUHS AD 01	MUHS AD 02	MUHS AD 03	MUHS AD 04	MUHS AD 05	MUHS AD 06	MUHS AD 07	MUHS AD 08	MUHS AD 09

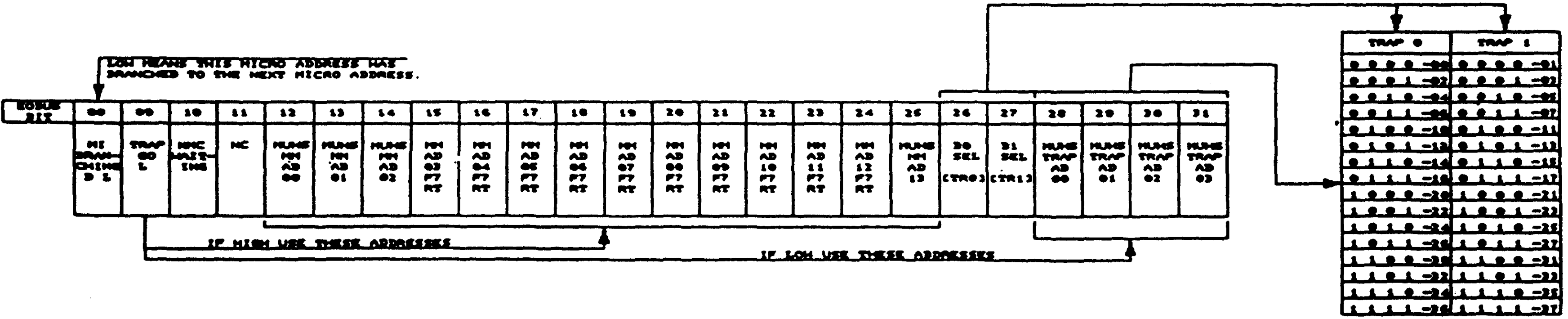
READ WITH A MICRO INST. WITH D[MUHS AD]

LOADED WITH A MICRO INST. WITH DEST[MUHS AD]


EOBUS DATA
CURRENT MICRO HISTORY ADDRESS BECOMES RELATIVE LOC. 0 D[MUHS AD]
00 ◄—► 17
COUNT EQUALS NOT STOPPED STB L
SEE PRINT MMAUH
UHS ADDRESS COUNTER
00 ◄—► 09
00 ◄—► 17
EOBUS DATA
RELATIVE LOC. 0 - X IS LOADED TO READ HISTORY REGISTERS. DEST[MUHS AD]
MICRO HISTORY REGISTERS
0000
1777
SEE PRINT MMBUH
23 BITS WIDE
08
31
EOBUS DATA
MICRO MEMORY ADDRESSES, TRAP RAM ADDRESSES, TRAP GO L, 30-1 SEL, MMI BRANCHING L, MMC WAITING

BLOCK DIAGRAM

READING BACK MICRO HISTORY DATA


LOW MEANS THIS MICRO ADDRESS HAS BRANCHED TO THE NEXT MICRO ADDRESS.
IF HIGH USE THESE ADDRESSES
IF LOW USE THESE ADDRESSES

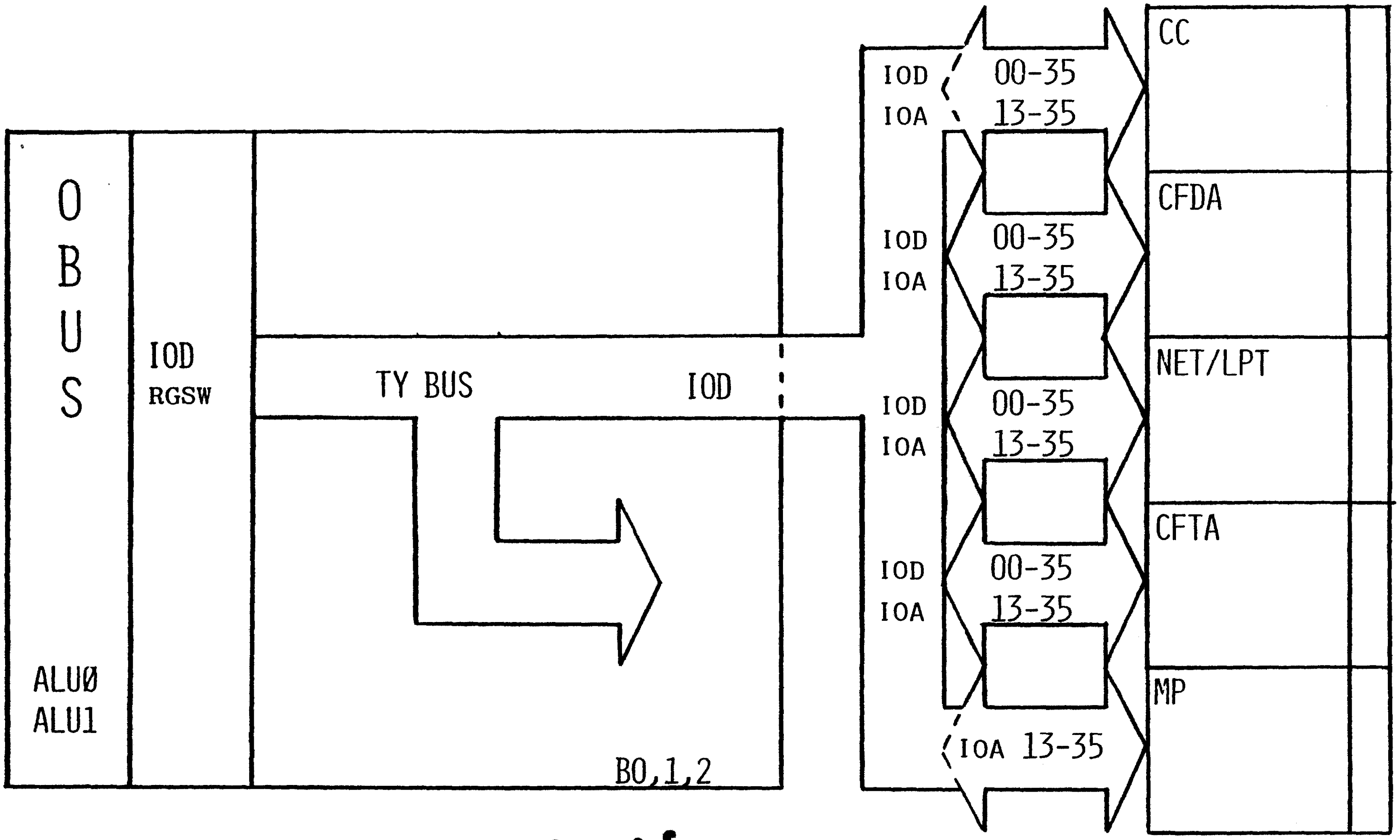
EOBUS BIT	08	09	10	11	12	13	14	15	16	17	18	19
	MI BRAN-CHING D L	TRAP GO L	MMC WAIT-ING	NC	MUHS MM AD 00	MUHS MM AD 01	MUHS MM AD 02	MM AD 03 F7 RT	MM AD 04 F7 RT	MM AD 05 F7 RT	MM AD 06 F7 RT	MM AD 07 F7 RT

EOBUS BIT	20	21	22	23	24	25	26	27	28	29	30	31
	MM AD 08 F7 RT	MM AD 09 F7 RT	MM AD 10 F7 RT	MM AD 11 F7 RT	MM AD 12 F7 RT	MUHS MM AD 13	30 SEL (TR0)	31 SEL (TR1)	MUHS TRAP AD 00	MUHS TRAP AD 01	MUHS TRAP AD 02	MUHS TRAP AD 03

TRAP 0	TRAP 1
0 0 0 0 -00	0 0 0 0 -01
0 0 0 1 -02	0 0 0 1 -03
0 0 1 0 -04	0 0 1 0 -05
0 0 1 1 -06	0 0 1 1 -07
0 1 0 0 -10	0 1 0 0 -11
0 1 0 1 -12	0 1 0 1 -13
0 1 1 0 -14	0 1 1 0 -15
0 1 1 1 -16	0 1 1 1 -17
1 0 0 0 -20	1 0 0 0 -21
1 0 0 1 -22	1 0 0 1 -23
1 0 1 0 -24	1 0 1 0 -25
1 0 1 1 -26	1 0 1 1 -27
1 1 0 0 -30	1 1 0 0 -31
1 1 0 1 -32	1 1 0 1 -33
1 1 1 0 -34	1 1 1 0 -35
1 1 1 1 -36	1 1 1 1 -37

MACDONNELL DOUGLAS			
TITLE MICRO HISTORY INFORMATION			
SIZE B	CODE	NUMBER	REV A
DATE DECEMBER 20, 1984		SHEET 01 OF 01	

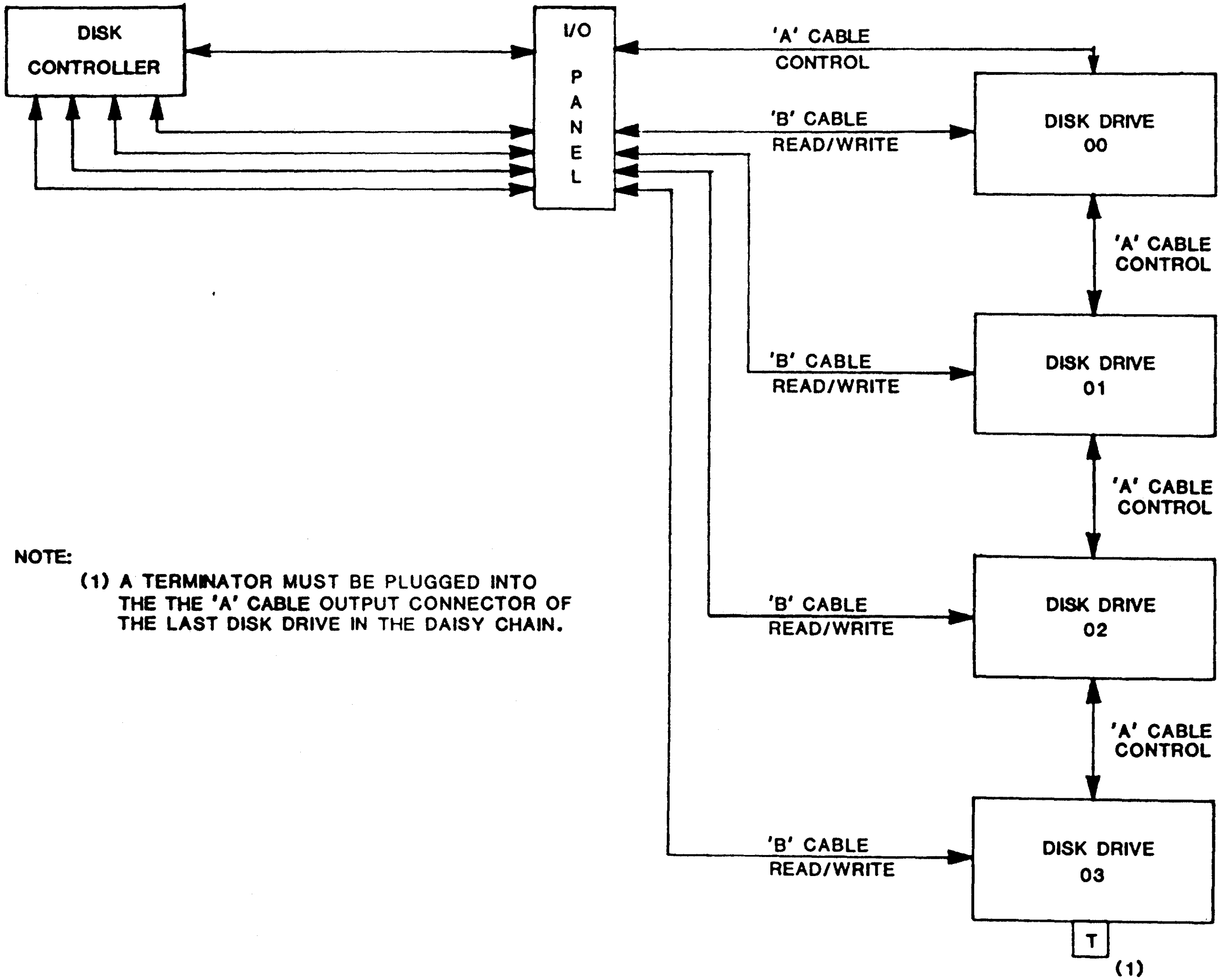
TY BUS



Also Ref
AMEM
from TYBUS

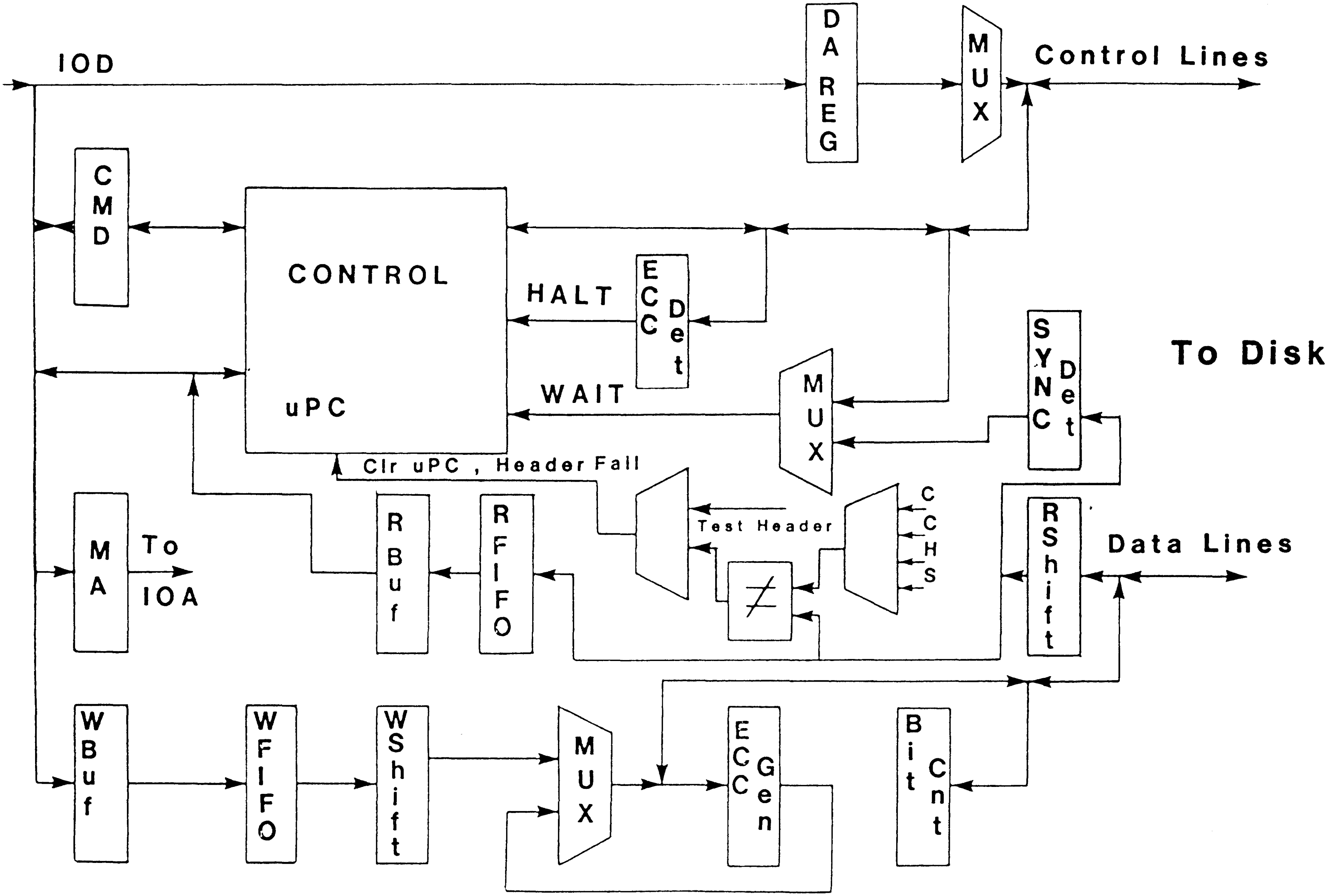
AUGMENT ENGINE

DISK DRIVE DAISY CHAIN CONFIGURATION



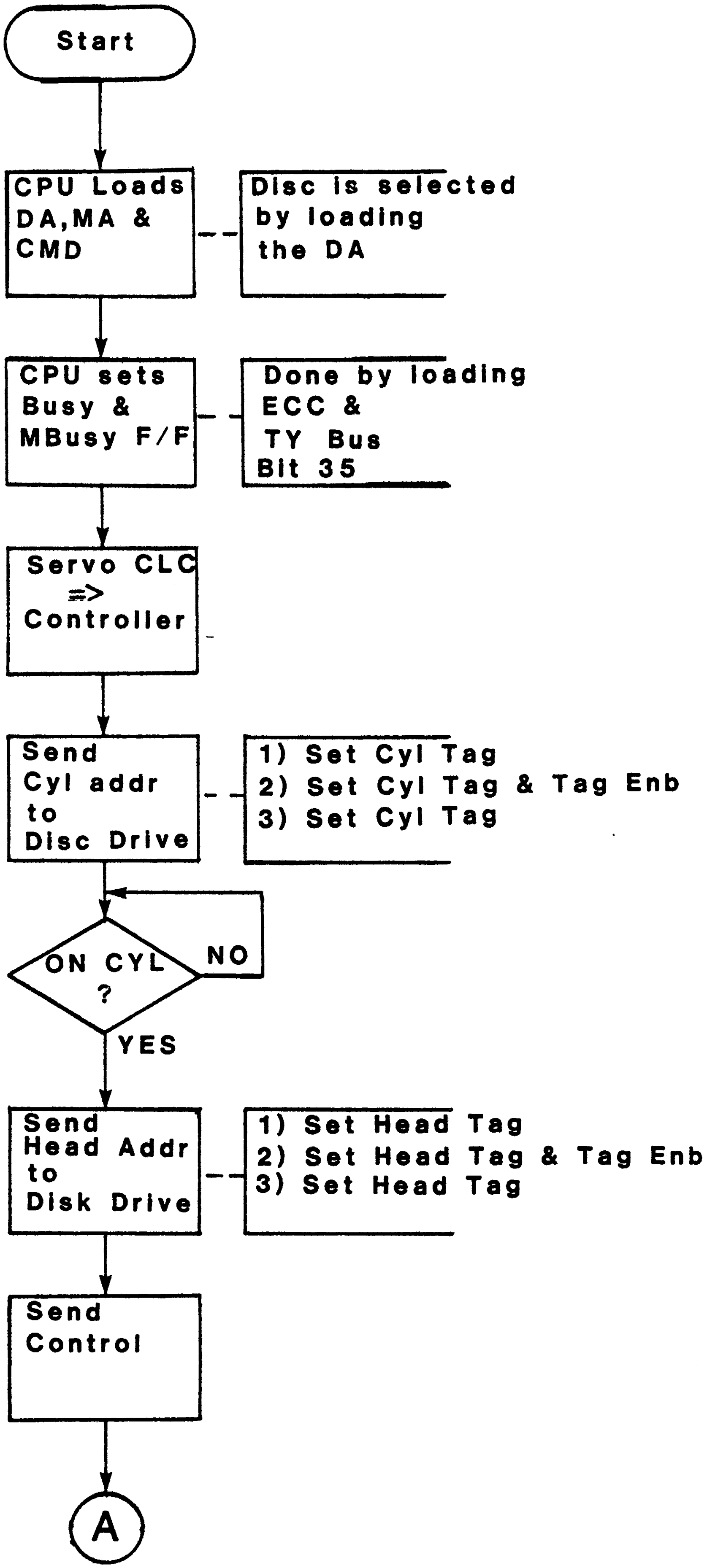
NOTE:

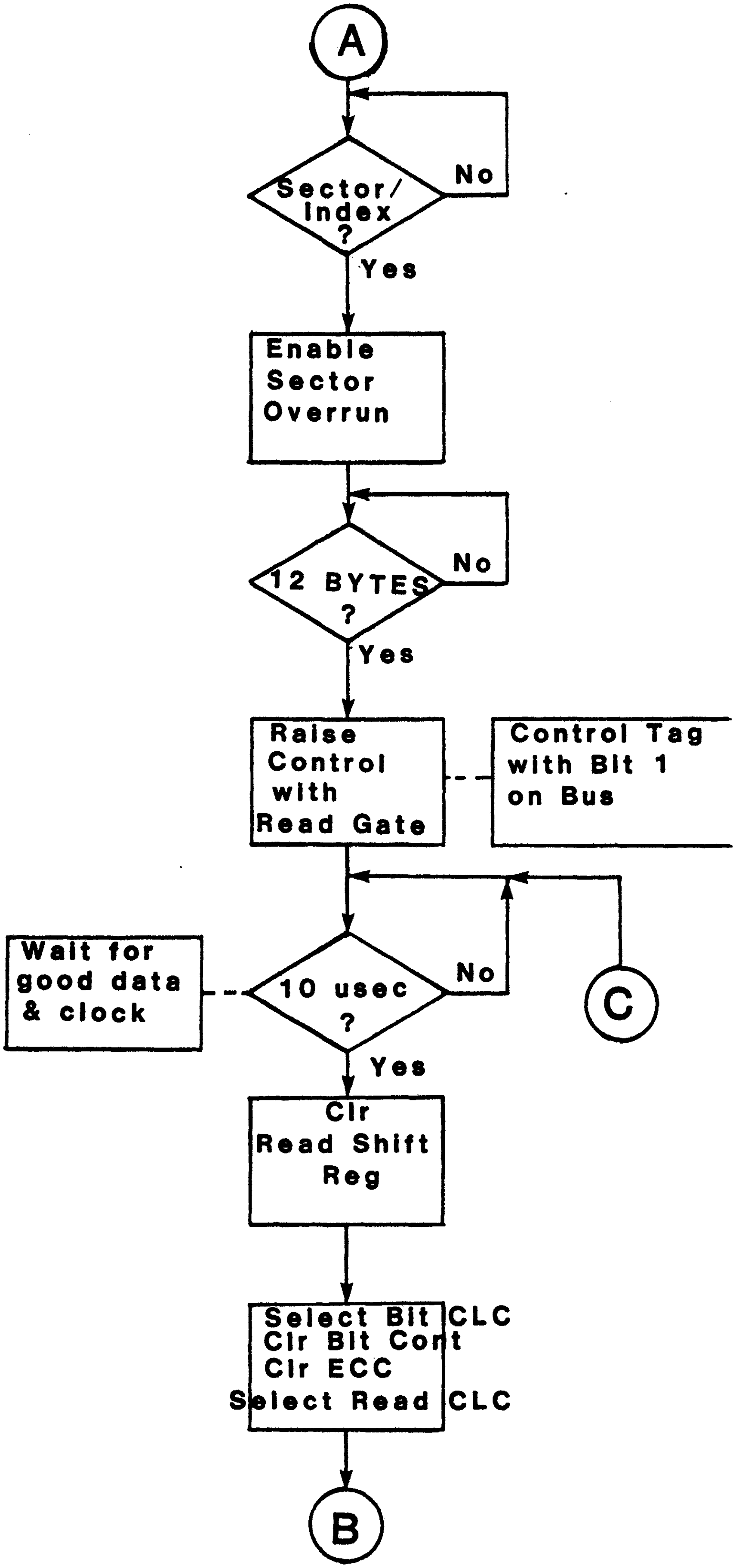
(1) A TERMINATOR MUST BE PLUGGED INTO THE THE 'A' CABLE OUTPUT CONNECTOR OF THE LAST DISK DRIVE IN THE DAISY CHAIN.



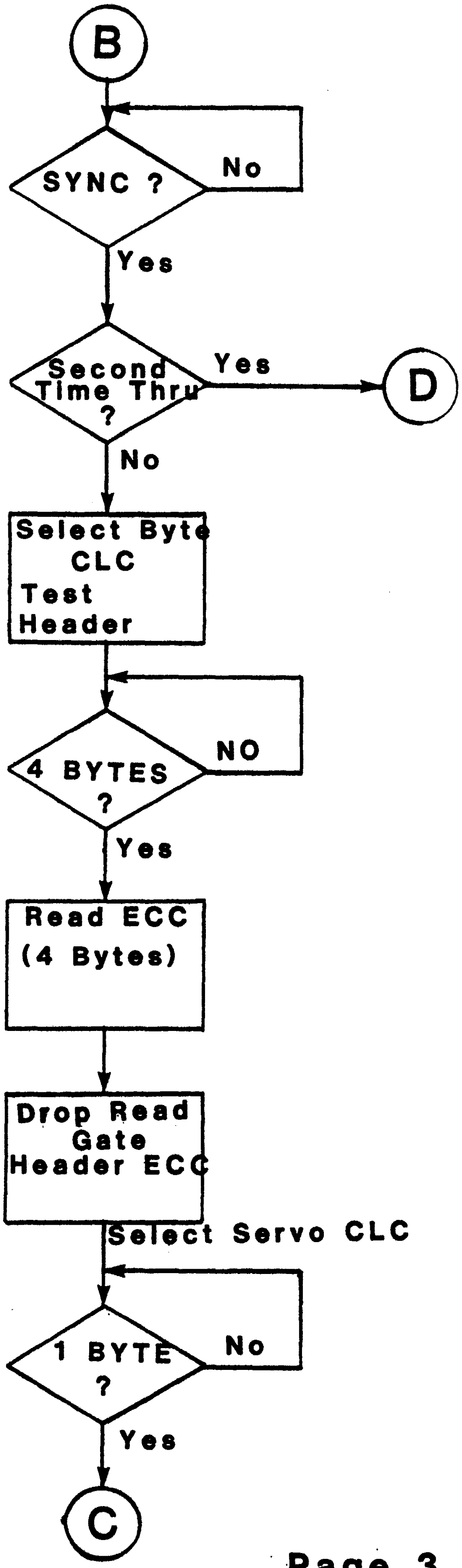
Disk Control Card Block Diagram

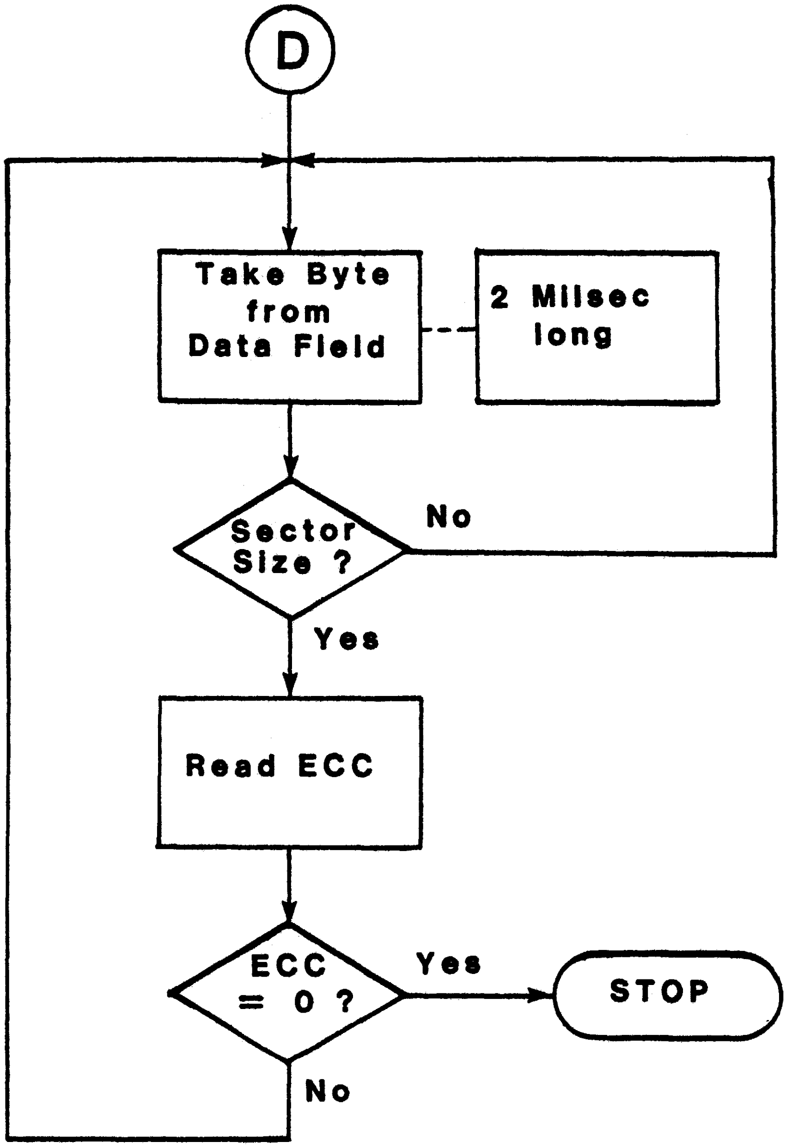
Disk Controller Micro Instruction Flow Chart


Start
CPU Loads DA,MA & CMD
Disc is selected by loading the DA
CPU sets Busy & MBusy F/F
Done by loading ECC & TY Bus Bit 35
Servo CLC => Controller
Send Cyl addr to Disc Drive
1) Set Cyl Tag
2) Set Cyl Tag & Tag Enb
3) Set Cyl Tag
ON CYL ?
NO
YES
Send Head Addr to Disk Drive
1) Set Head Tag
2) Set Head Tag & Tag Enb
3) Set Head Tag
Send Control
A


A
Sector/ Index ?
No
Yes
Enable Sector Overrun
12 BYTES ?
No
Yes
Raise Control with Read Gate
Control Tag with Bit 1 on Bus
Wait for good data & clock
10 usec ?
No
C
Yes
Clr Read Shift Reg
Select Bit CLC
Clr Bit Cont
Clr ECC
Select Read CLC
B

Page 2


B
SYNC ?
No
Yes
Second Time Thru ?
Yes
D
No
Select Byte CLC
Test Header
NO
4 BYTES ?
Yes
Read ECC (4 Bytes)
Drop Read Gate Header ECC
Select Servo CLC
1 BYTE ?
No
Yes
C


D
Take Byte from Data Field
2 Milsec long
Sector Size ?
No
Yes
Read ECC
ECC = 0 ?
Yes
STOP
No

THIS FILE IS DCTRL.DOC --- DESCRIPTION OF DISK CONTROLLER

DISK CONTROLLER OPCODES   I.E. PDP-10 I/O INSTRUCTION CODES

715	READ CMD
716	READ MA
717	READ DA
720	READ ECC
721	LOAD CMD
722	LOAD MA
723	LOAD DA
724	LOAD ECC

LOAD ECC

PRIMARILY CLEARS ERROR CORRECTING CODE LOGIC.
ALSO, DOES SPECIAL FUNCTION DEPENDING ON CONTENTS OF E.

CONTENTS OF E	FUNCTION
1	START A COMMAND
2	INITIALIZE CONTROLLER (SEND BEFORE STARTING CMD)

LOAD DA, READ DA

LOAD AND READ DISK ADDRESS. SELECT UNIT.

BITS	CONTENTS
04-06	UNIT NUMBER
07	SELECT UNIT THIS BIT MUST BE CLEARED, THEN SET, TO SELECT UNIT
08-19	CYLINDER NUMBER
20-27	HEAD NUMBER
28-35	SECTOR NUMBER

```
DETAIL FOR LOAD CMD:
BIT #   FUNCTION
=====   ========
15      USE SECTOR COUNTER
                FOR ROTATIONAL POSITION SENSING - TELLS CONTROLLER
                TO LISTEN TO SIGNALS FROM DRIVE TO TELL WHERE IT IS.
16      RELEASE
                RELEASE DRIVE FOR USE BY ANOTHER COMPUTER
17      RECALIBRATE
                SIDE EFFECT: RESETS SEEK ERROR
---------
18      FAULT CLEAR
                SIDE EFFECT: RESETS SEL UNIT FAULT
                USE BITS 17,18 WITH COMMAND=4
19      DATA STROBE LATE
20      DATA STROBE EARLY
---
21      SERVO OFFSET MINUS
22      SERVO OFFSET PLUS
                BITS 15-22 ARE 0 BY DEFAULT
                BITS 19-22 ARE FOR READING MARGINAL DATA
23      CMD FROM MEMORY
                COMMAND SENDS DATA FROM MEMORY TO DISK
                (USE WITH WRITE COMMANDS)
---
24      CMD 0
25      CMD 1
26      CMD 2
                BITS 24-26 ARE DISK CONTROLLER SEQUENCER START ADDRESS
------
27      32 BIT MODE
28      ***
29      ***
---
30      ANY ATTENTION INTERRUPT ENABLE
                ALLOWS INTERRUPT WHEN ANY UNIT IS AT ATTENTION
31      ***
32      DONE INTERRUPT ENABLE
---
33      ***
34      ***
35      ***

DETAIL FOR CMD (BITS 24-26)
        CMD     FN
        ---     --
        0       READ
        1       WRITE
        3       WRITE ALL               (SECTOR + FORMAT DATA)
        4       CONTROL FUNCTIONS       (RECAL, FAULT CLEAR)
        5       (WILL BE 'SEEK')
```

```
DETAIL FOR READ CMD:
BIT #   FUNCTION
=====   ========
00      SELECT ERROR
                IF 0, SUCCESSFULLY TALKING TO A DISK UNIT
                IF 1, BITS 01-06 NOT VALID.
01      SEL UNIT WRITE PROTECTED
02      -SEL UNIT READY
                -READY LIGHT ON DRIVE
---
03      SEL UNIT ON CYLINDER
                SEEK COMPLETE
04      SEL UNIT SEEK ERROR
                DETECTED BY DRIVE
05      SEL UNIT FAULT
                DETECTED BY DRIVE
---
06      SEL UNIT ATTENTION
                SET BY LEADING EDGE OF BIT 03  (ON CYL)
07      HEADER COMPARE ERROR
                SECTOR COUNTER /= SECTOR HEADER
                (FOR USE WITH ROTATIONAL POSITION SENSING)
08      NOT BUSY
                DISK CONTROLLER SEQUENCER IS STOPPED
09      FIFO EMPTY -- SHOULD BE ON AFTER NORMAL READ OR WRITE
------
10      ***
11      READ OVERRUN ERROR
---
12      WRITE OVERRUN ERROR
13      SECTOR OVERRUN ERROR
14      INTERNAL PARITY ERROR
                BITS 11-14 ARE DETECTED BY CONTROLLER
---
15-17   (SAME AS LOAD CMD)
---------
18-20   (SAME AS LOAD CMD)
---
21-23   (SAME AS LOAD CMD)
---
24-26   (SAME AS LOAD CMD)
------
27      (SAME AS LOAD CMD)
28      ANY ERROR
                OR OF ALL ERROR BITS EXCEPT 14 (INTERNAL PARITY)
29      ANY ATTENTION
                AY UNIT IS AT ATTENTION
---
30      (SAME AS LOAD CMD)
31      -ACTIVE
                SEQUENCER STOPPED AND FIFO EMPTY (ON INPUT TO MEMORY)
32      (SAME AS LOAD CMD)
---
33      READ COMPARE ERROR
                CONTROLLER COMPARES MEMORY TO DISK (NOT IMPLEMENTED)
34      TIMOUT ERROR
                *** NOT IMPLEMENTED ***
35      MEM PAR ERR
```

BYTE PACKING BY HARDWARE:

1ST 4 BYTES GO TO (OR COME FROM) WORD N, BITS 0-31. NEXT BYTE GETS SPLIT 4 BITS TO WORD N, BITS 32-35, AND 4 BITS TO WORD N+1, BITS 32-35. LAST 4 BYTES TO WORD N+1, BITS 0-31.

32 BIT MODE: DATA TAKEN FROM HIGH ORDER 4 BYTES ONLY.

!

FORMAT DISC FROM EDDT

Load U

Disc Test (DT)

MT File 6

MG 140

Start Mos.Mem.

CTY

EDDT

RD Esc G

Prints Information

Help ESC G

```
**Unit/     -1   0 LF
 *CYL# /    -1   1777 LF
 *HEAD # /  -1   17 LF
 *SECTOR #/-1     7 LF
  MAXCYL #/ 0    1777 LF
  MAXHED #/ 0     17 LF
  MAXSEC #/ 0      7 LF

  MINCYL #/ 0     LF
  MINHED #/ 0     LF
  MINSEC #/ 0     LF
  MINSAF #/ 0     LF
  MAXSAF #/ 0    1777 LF

  BAT1CY #/    0 LF
  BAT1HD #/    0 LF
  BAT2SC #/    0 4 LF
  BAT2CY #/    0   LF
  BAT2HD #/    0   LF
  BAT2SC #/    0 4 LF
```

** = Changeable for another unit

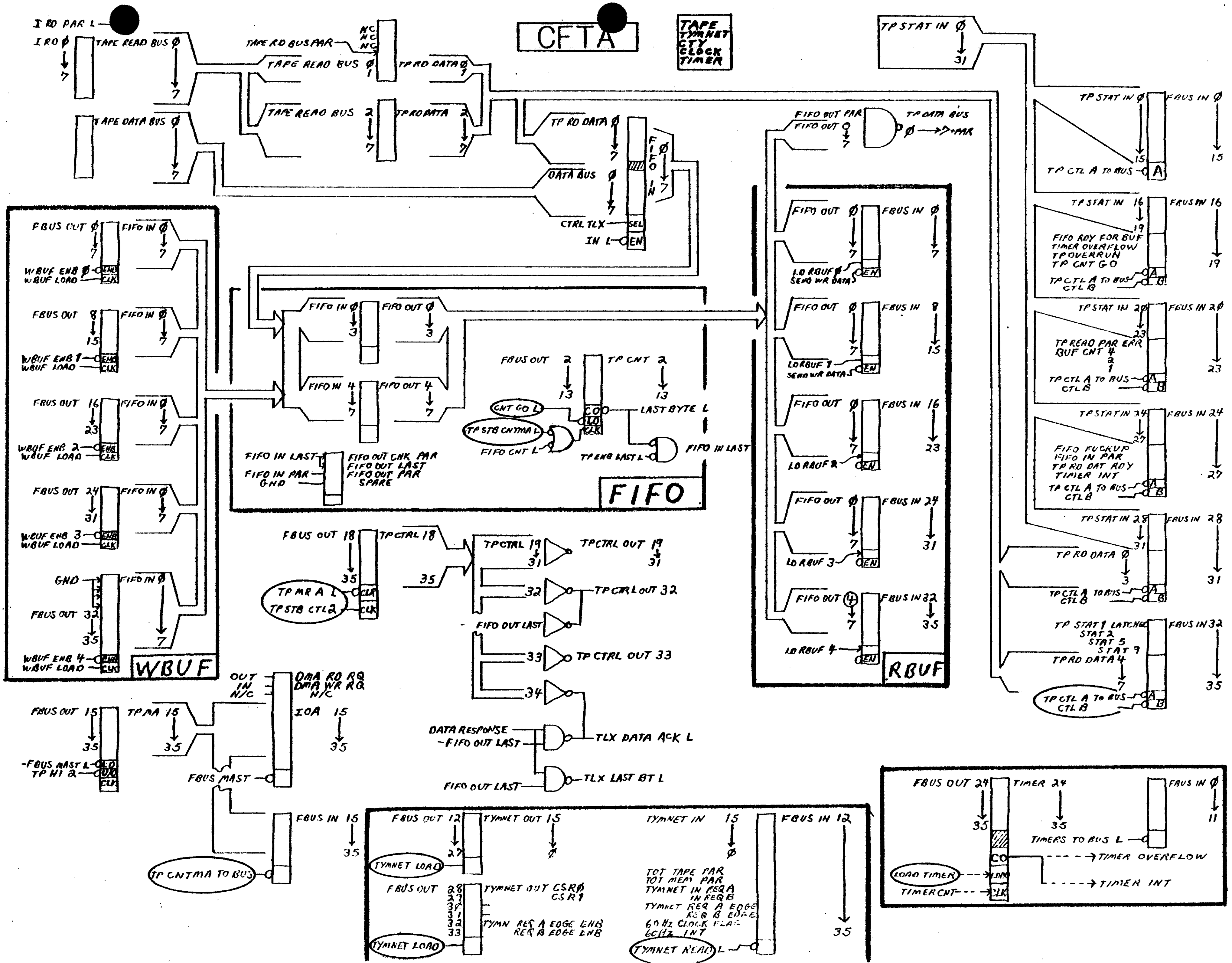
* = must be reset each run.

FMT ESC G   Not Read Only

CTR T   Indicate where format is

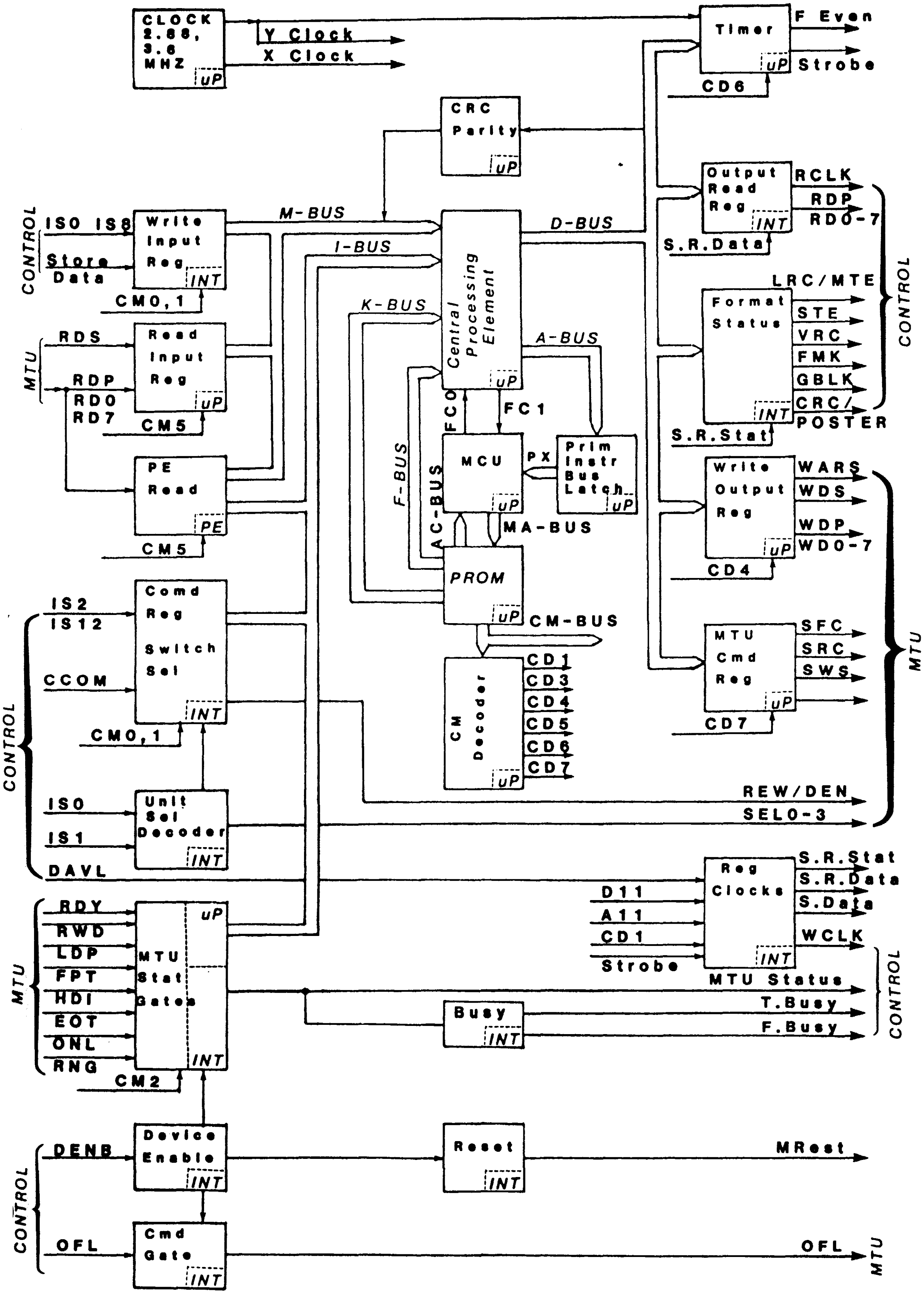
UNIT,CYL HEAD, SECTOR, & COMMAND

PS=SYSTEM STRUCTURE


CFTA
TAPE TYMNET CTY CLOCK TIMER
I RD PAR L
I RD Ø 7
TAPE READ BUS Ø 7
TAPE DATA BUS Ø 7
TAPE RD BUS PAR
NC NC NC
TAPE READ BUS Ø 1
TP RD DATA Ø 1
TAPE READ BUS 2 7
TP RD DATA 2 7
TP RD DATA Ø 7
DATA BUS Ø 7
FIFO Ø IN 7
CTRL TLX
SEL
IN L
EN
TP STAT IN Ø 31
FIFO OUT PAR
FIFO OUT Ø 7
TP DATA BUS Ø → 7 + PAR
TP STAT IN Ø 15
FBUS IN Ø 15
TP CTL A TO BUS
A
TP STAT IN 16 19
FBUS IN 16 19
FIFO RDY FOR BUF
TIMER OVERFLOW
TP OVERRUN
TP CNT GO
TP CTL A TO BUS
CTL B
B
TP STAT IN 20 23
FBUS IN 20 23
TP READ PAR ERR
BUF CNT 4 2 1
TP CTL A TO BUS
CTL B
TP STAT IN 24 27
FBUS IN 24 27
FIFO FUCKUP
FIFO IN PAR
TP RD DAT RDY
TIMER INT
TP CTL A TO BUS
CTL B
TP STAT IN 28 31
FBUS IN 28 31
TP RD DATA Ø 3
TP CTL A TO BUS
CTL B
TP STAT 1 LATCHED
STAT 2
STAT 5
STAT 9
TP RD DATA 4 7
FBUS IN 32 35
TP CTL A TO BUS
CTL B
FBUS OUT Ø 7
FIFO IN Ø 7
WBUF ENB Ø
WBUF LOAD
ENB
CLK
FBUS OUT 8 15
FIFO IN Ø 7
WBUF ENB 1
WBUF LOAD
FBUS OUT 16 23
FIFO IN Ø 7
WBUF ENB 2
WBUF LOAD
FBUS OUT 24 31
FIFO IN Ø 7
WBUF ENB 3
WBUF LOAD
GND
FBUS OUT 32 35
FIFO IN Ø 7
WBUF ENB 4
WBUF LOAD
WBUF
FIFO IN Ø 3
FIFO OUT Ø 3
FIFO IN 4 7
FIFO OUT 4 7
FIFO IN LAST
FIFO IN PAR
GND
FIFO OUT CHK PAR
FIFO OUT LAST
FIFO OUT PAR
SPARE
FBUS OUT 2 13
TP CNT 2 13
CNT GO L
TP STB CNTMA L
FIFO CNT L
CO
LD
CLK
LAST BYTE L
TP ENB LAST L
FIFO IN LAST
FIFO
FIFO OUT Ø 7
FBUS IN Ø 7
LD RBUF Ø
SEND WR DATA
EN
FIFO OUT Ø 7
FBUS IN 8 15
LD RBUF 1
SEND WR DATA
FIFO OUT Ø 7
FBUS IN 16 23
LD RBUF 2
FIFO OUT Ø 7
FBUS IN 24 31
LD RBUF 3
FIFO OUT 4 7
FBUS IN 32 35
LD RBUF 4
RBUF
FBUS OUT 18 35
TP CTRL 18
TP MR A L
TP STB CTL 2
CLR
CLK
TP CTRL 19 31
TP CTRL OUT 19 31
32
TP CTRL OUT 32
FIFO OUT LAST
33
TP CTRL OUT 33
34
DATA RESPONSE
-FIFO OUT LAST
TLX DATA ACK L
FIFO OUT LAST
TLX LAST BT L
FBUS OUT 15 35
-FBUS MAST L
TP HI 2
LD
U/D
CLK
TP MA 15 35
OUT
IN
N/C
DMA RD RQ
DMA WR RQ
N/C
IOA 15 35
FBUS MAST
FBUS IN 15 35
TP CNTMA TO BUS
FBUS OUT 12 27
TYMNET LOAD
TYMNET OUT 15 Ø
TYMNET IN 15 Ø
FBUS IN 12 35
FBUS OUT 28 29 30 31 32 33
TYMNET OUT CSR Ø
CSR 1
TYMN REQ A EDGE ENB
REQ B EDGE ENB
TYMNET LOAD
TOT TAPE PAR
TOT MEM PAR
TYMNET IN REQ A
IN REQ B
TYMNET REQ A EDGE
REQ B EDGE
60 Hz CLOCK FLAG
60 Hz INT
TYMNET READ L
FBUS OUT 24 35
TIMER 24 35
FBUS IN Ø 11
TIMERS TO BUS L
CO
LOAD TIMER
LDA
TIMER CNT
CLK
TIMER OVERFLOW
TIMER INT

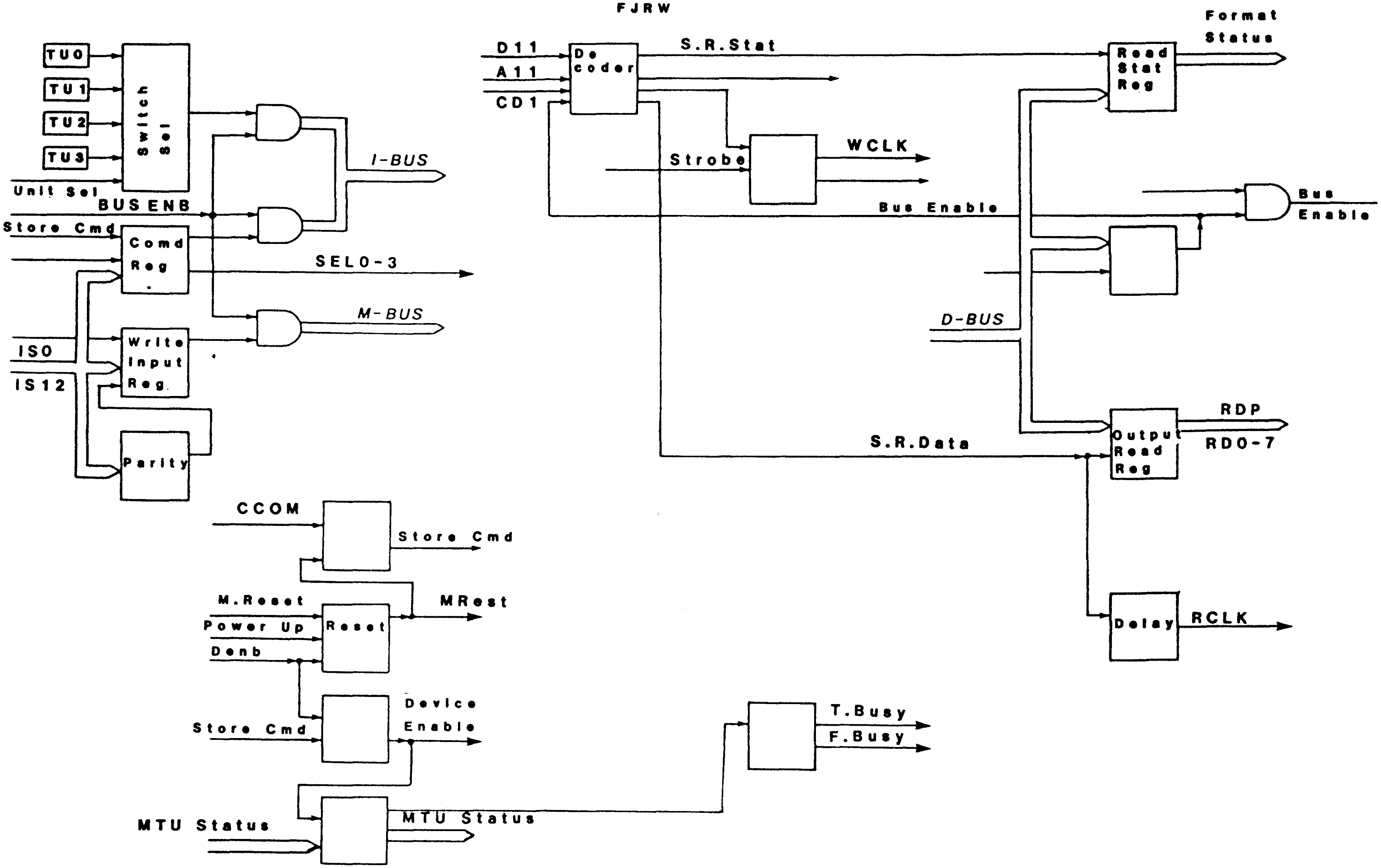
FORMATTER BLOCK

FJRW



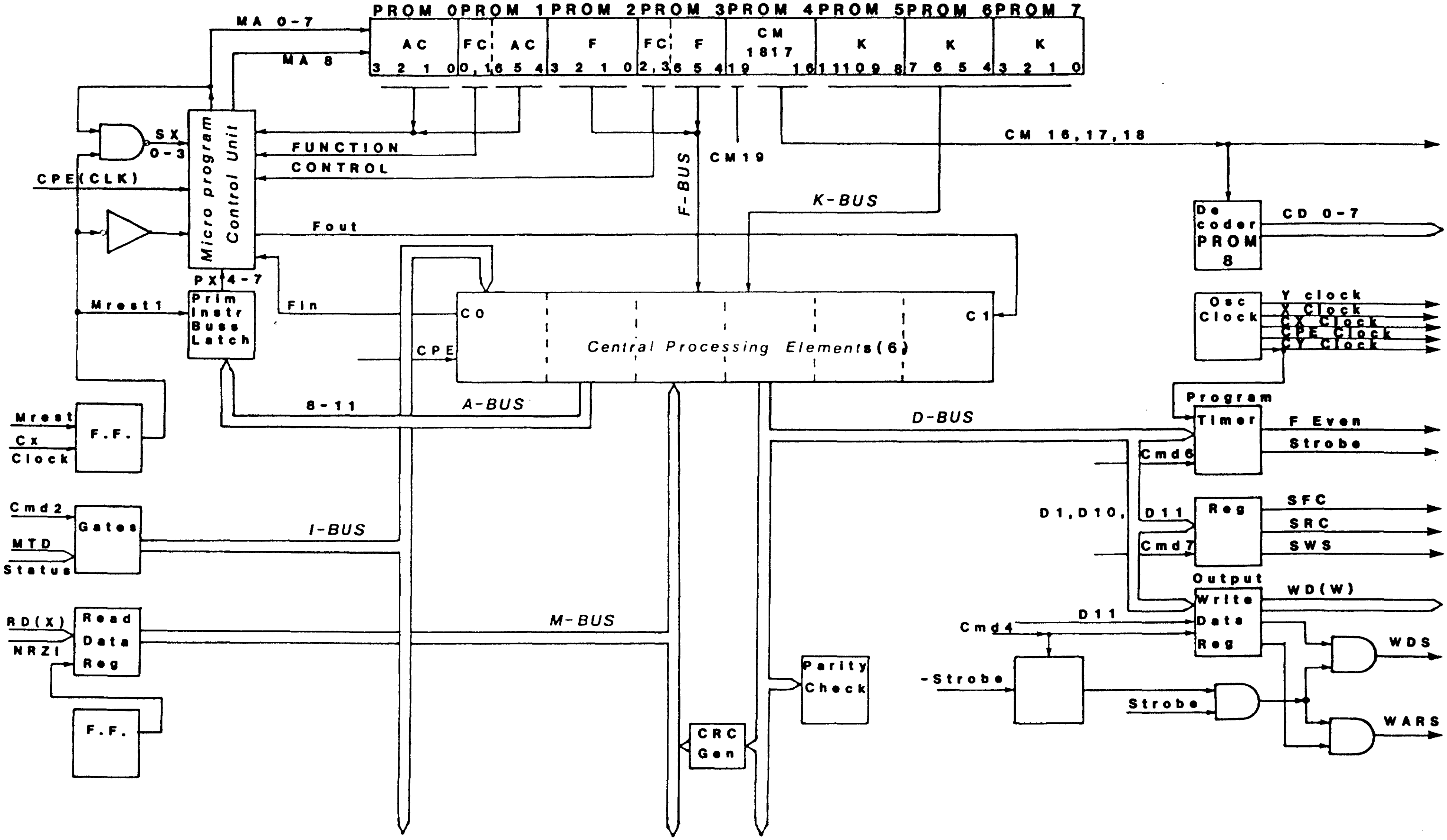
INTERFACE BLOCK

FJRW



uP BLOCK

FJRW



KENNEDY WRITE AMP DESKEW

After a write channel card is changed the electronic write deskew must be adjusted for. The 4 channel write board contains the write deskew circuitry for tracks P,0,1,2 (P has no switch settings).
S1 is for track 0.
S2 is for track 1.
S3 is for track 2.

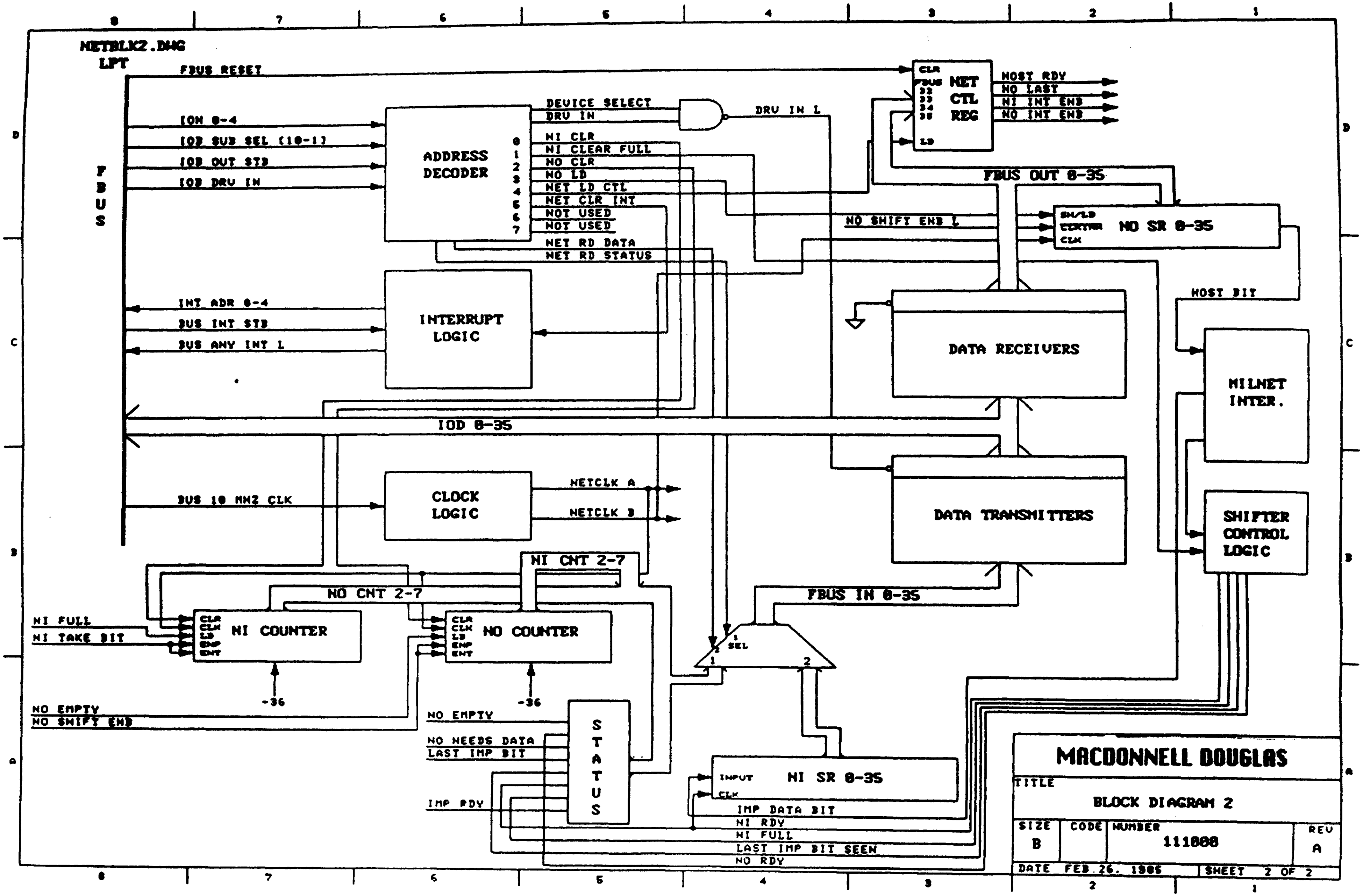
The 5 channel write board contains the write deskew circuitry for tracks 3,4,5,6,7.
S1 is for track 3.
S2 is for track 4.
S3 is for track 5.
S4 is for track 6.
S5 is for track 7.

The correct switch settings should have been placed on a sticker. To set a track to proper deskew remove the write channel amp. Find the switch setting on top of the card cage. If you wanted to set a switch to 7 the switch would look like the example below.

	1	2	3	4
on	x	x	x	
				x

The value of this switch is 7.
The switches are in binary.
1 is the 1's column.
2 is the 2's column.
3 is the 4's column.
4 is the 8's column.

As the head wears these switches may have to be changed. This can be done with an extender card.

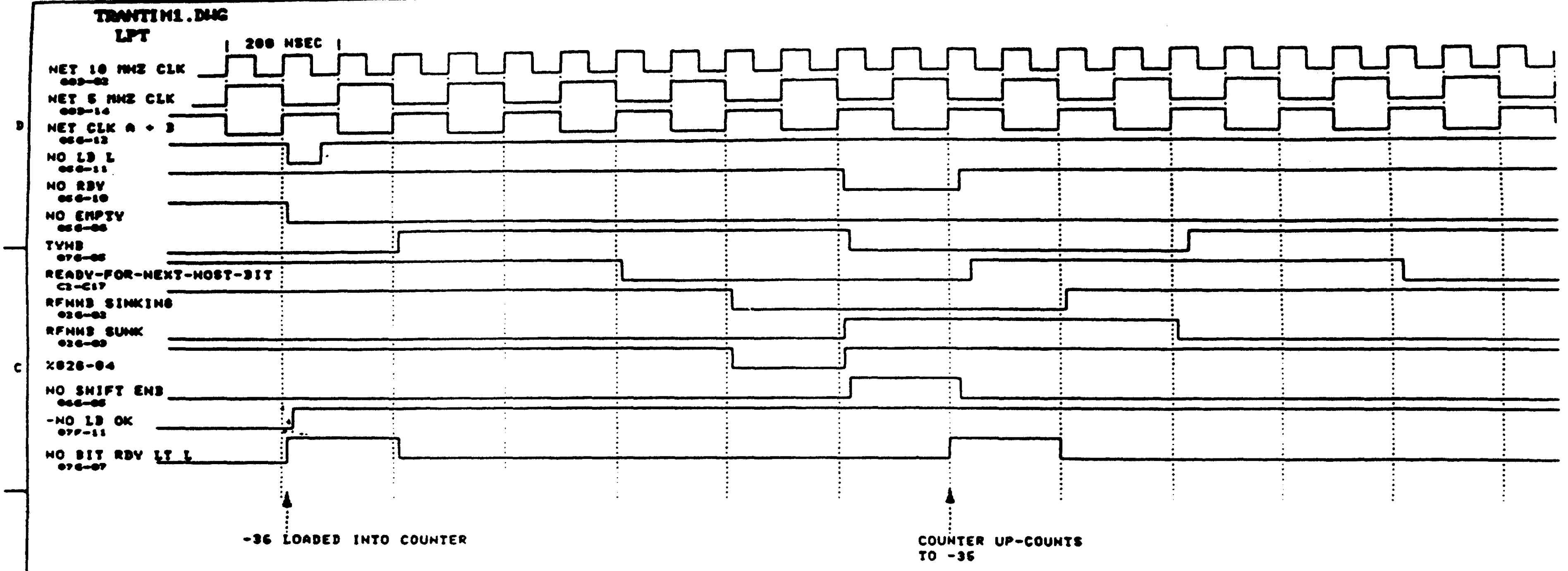


MACDONNELL DOUGLAS

TITLE BLOCK DIAGRAM 2

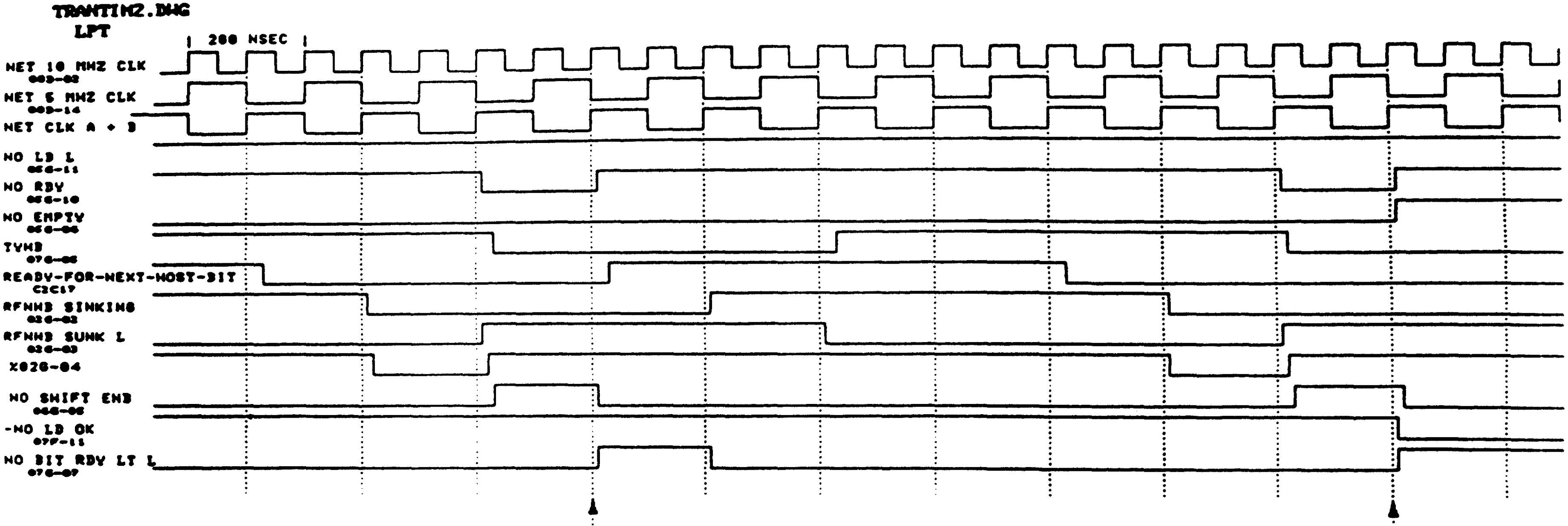
SIZE	CODE	NUMBER	REV
B		111000	A

DATE FEB.26. 1985 SHEET 2 OF 2



MACDONNELL DOUGLAS

TITLE	TIMING CHART-TRANSMISSION		
SIZE B	CODE	NUMBER 000001	REV A
DATE FEB. 25, 1986		SHEET 1 OF 2	



COUNTER UP-COUNTS
TO -1

COUNTER UP-COUNTS
TO ZERO

MACDONNELL DOUGLAS

TITLE
TIMING CHART-TRANSMISSION

SIZE	CODE	NUMBER	REV
B		0000002	A

DATE FEB 26, 1985 SHEET 2 OF 2

```
+------------------------------+
|          TYMSHARE            |
| Network Technology Division |
+------------------------------+
```

Hardware Configuration Specification LSI-11 Based TYMBASE

Author...J.M.Stammers.
Date.......Sep 18 1980
Update....9/15/82;DRE

2 Components

The following is a list of components which can be integrated into TYMBASE configurations.

2.1 Processors

Three processor boards can be used in a TYMBASE configuration, two are revisions of the KD11 module and the third is the KDF11 module. These are the M7264 quad height board with on board memory, the M7270 dual height board, and the M8186 dual height board. All three are described in the 1979-80 edition of "Microcomputer Processor Handbook" by Digital Equipment Corporation.

2.2 Bootstraps

The bootstrap board used in a TYMBASe configuration is the BDV11-AA (M8012).

The BDV11 board is described in the 1978-1979 edition of "Memories and Peripherals".

The bootstrap has a prom developed by Tymshare which resides on the board. This bootstrap allows the down line loading of the TYMBASE from TYMNET using LOADII or equivalent program, or from a connected host. Hosts currently supported are:-

1. DEC-2020 Tymcom-X
2. PDP-11 RSX-11M
3. PDP-11 RSTS
4. F3 Tenex
5. F4 Tenex

The BDV11 is a quad height board.

2.3 Memory

The TYMBASe is provided with a 64K byte dual height MSV11-D (M8044) or MSV11-E (M8045) 18 bit MOS memory board.

The MSV11 memory is described in the 1978-1979 edition of "Memories and Peripherals".

3 Configurations

There are two possible configurations that can be supported by the TYMBASE software. These are detailed below.

3.1 Large Box

The modules in the large box are inserted as follows:-

1. KD11 or KDF11 processor

2. MSV11 memory board

3. 0-2 DPV (or DUV11) synchronous serial interfaces

4. 1-5 DLV11-J asynchronous serial interfaces

5. 0-4 DZV11 asynchronous multiplexors

6. 1-2 DRV11 parallel interfaces

7. BDV11 bootstrap board (with prom)

```
+---------------+---------------+
|    KD11       |               | 1.
+---------------+---------------+
|    MSV11      |               | 2.
+---------------+---------------+
|    DPV11      |               | 3.
+---------------+---------------+
|       DPV11  / DLV11-J        | 4.
+-------------------------------+
|   DLV11-J / DZV11 / DRV11     | 5.
+-------------------------------+
|   DLV11-J / DZV11 / DRV11     | 6.
+-------------------------------+
|   DLV11-J / DZV11 / DRV11     | 7.
+-------------------------------+
|   DLV11-J / DZV11 / DRV11     | 8.
+-------------------------------+
|             BDV11             | 9.
+-------------------------------+
```

The DPV11(s) (or DUV11(s)) must be installed in the bus closest to the processor because it is the most time critical device. The DRV11(s) must be installed last in the bus (but before the BDV11) in order for the node to perform well. The BDV11 acts as bus terminator, so it must always be inserted in the last slot after the other boards. (i.e. there must not exist any 'holes' between interrupting devices in the bus; the BDV11 is not an interrupting

2 Small box

The insertion of boards in the small box (with expander) is as follows:-

1. KD11 or KDF11 processor

2. MSV11 memory module

3. 1-2 DPV11 (or DUV11) synchronous serial interfaces

4. 1-5 DLV11-J 4 SLU mooule

5. 0-4 DZV11 asynchronous multiplexors

6. 1-2 DRV11 parallel interface

7. 1 Q bus expander

```
+------------------------------------+
|       KD11       |      MSV11      | 1.
+------------------+-----------------+
|      DLV11-J     |      DPV11      | 2.
+------------------------------------+
|      (2 x DLV11-J) / DZV11         | 3.
+------------------------------------+
| ::::::::::::     |    DLV11-J      | 4.
+-::::::::::::-----------------------+
  ::Expander::
+-::::::::::::-----------------------+
| ::::::::::::     |    DLV11-J      | 5.
+------------------------------------+
|          DZV11 / DRV11             | 6.
+------------------------------------+
|          DZV11 / DRV11             | 7.
+------------------------------------+
|               BDV11                | 8.
+------------------------------------+
```

All boards must be contiguous from slot 1. The DPV11(s) must be installed in the slot(s) closest to the processor because it is the most time critical device. The DRV11(s) must be installed last in the bus (but before the BDV11) in order for the node to perform well. If no bus expander is provided, then the BDV11 must be inserted into slot 4, and the DPV11 in slot 2 (right side).

	176650	210
	176660	220
	176670	230
DLV11 5	176700	240
	176710	250
	176720	260
	176730	270

These addresses and vectors MUST be observed for consistency.

E21 switches are set as follows:-

```
                 E21
     +-------------------+
     | 0 | 0 | 0 | 1 | 1 |
     +-------------------+
       1   2   3   4   5
```

On (Closed) = 1, Off (Open) = 0

E15 switch settings are as follows:-

```
                     E15
     +-------------------------------+
     | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
     +-------------------------------+
       1   2   3   4   5   6   7   8
```

2. Address and vector switches are set to the standard values.

```
                 E38
+-------------------------------+
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | x |
+-------------------------------+
  1   2   3   4   5   6   7   8

                 E39
+-------------------------------+
| x | x | 1 | 0 | 0 | 0 | x | x |
+-------------------------------+
  1   2   3   4   5   6   7   8
E38-8  => 1 for DUV11 number 4
else   => 0
E39-1  => 1 for DUV11 number 2 and 3
else   => 0
E39-2  => 1 for DUV11 number 1 and 3
else   => 0
E39-3  => 1 for all DUV11s
else   => 0
E39-7  => 1 for DUV11 number 3 and 4
else   => 0
E39-8  => 1 for DUV11 number 2 and 4
else   => 0
```

0 is the off or open position; 1 is the on or closed position of the switch.

Switch E55 is on the "handles" end of the board, E38 at the "fingers"
end, and E39 is in between.

6 DLV11-J (M8043) and DZV11 (M7957) Asynchronous Interfaces

4.6.1 DLV11-J

To set up the standarc addresses and vectors for the 4 ports on thc DLV11-J, it is necessary to jumper the board accordingly.

1. Adoress selection

   o A12 => X to 1

   o A11 => X to 1

   o A10 => X to 1

   o A9 => X to 0

   o A8 => X to 1

   o A7 => remove jumper for DLV11 1 and 2 otherwise install jumper.

   o A6 => remove jumper for DLV11 3 and 4 otherwise install jumper.

   o A5 => X to 0 for DLV11 1, 3 and 5 otherwise X to 1.

2. Vector selection

   o V7 => Installed for DLV11 1, 2 , 4 and 5 otherwise removed

   o V6 => Installed for DLV11 1 , 2 and 3 otherwise removed

   o V5 => X to 0 for DLV11 1

   V5 => X to 1 for DLV11 2 , 3 and 5 otherwise removed

3. Console selection

   o C1 => X to 1 for DLV11 1 otherwise X to 0

   o C2 => X to 1 for DLV11 1 otherwise X to 0

   o Break selection (B X H) => Remove jumper

4. Channel parameters

   For channels 0 through 3;

4.6.2 DZV11

Addresses and vectors for DZV11s are set up as follows:-

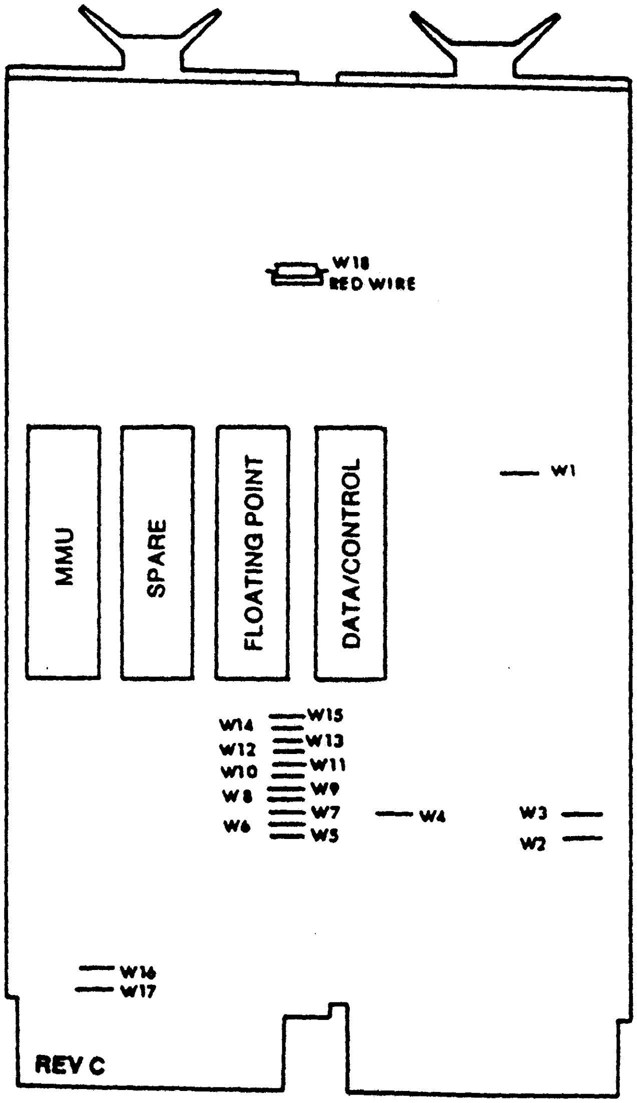
1. Address selection

```
                E30
+---------------------------------------+
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | x | x |
+---------------------------------------+
  1   2   3   4   5   6   7   8   9   10
E30-9  => 1 for DZV11 number 3 and 4
else   => 0
E30-10 => 1 for DZV11 number 2 and 4
else   => 0
```

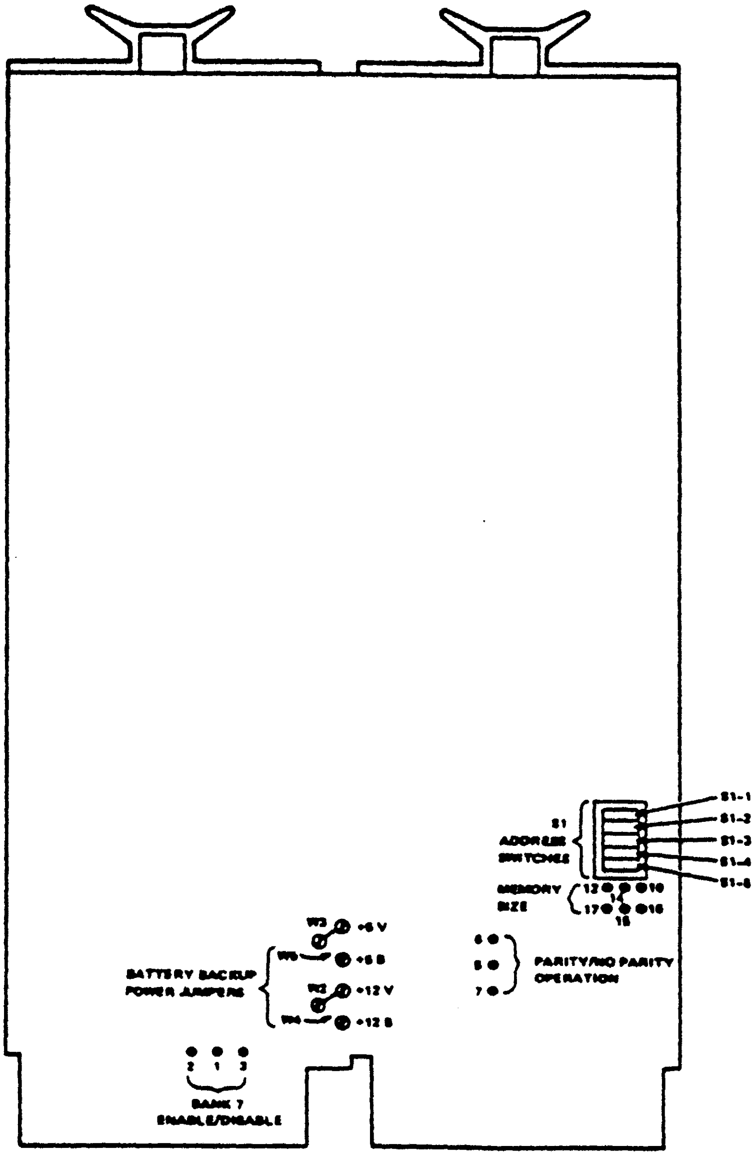
2. Vector selection

```
               E2
+-------------------------------+
| 1 | 0 | 0 | 1 | x | x | 0 | 0 |
+-------------------------------+
  1   2   3   4   5   6   7   8

E2-5  => 1 for DZV11 number 3 and 4
else  => 0
E2-6  => 1 for DZV11 number 2 and 4
else  => 0
```

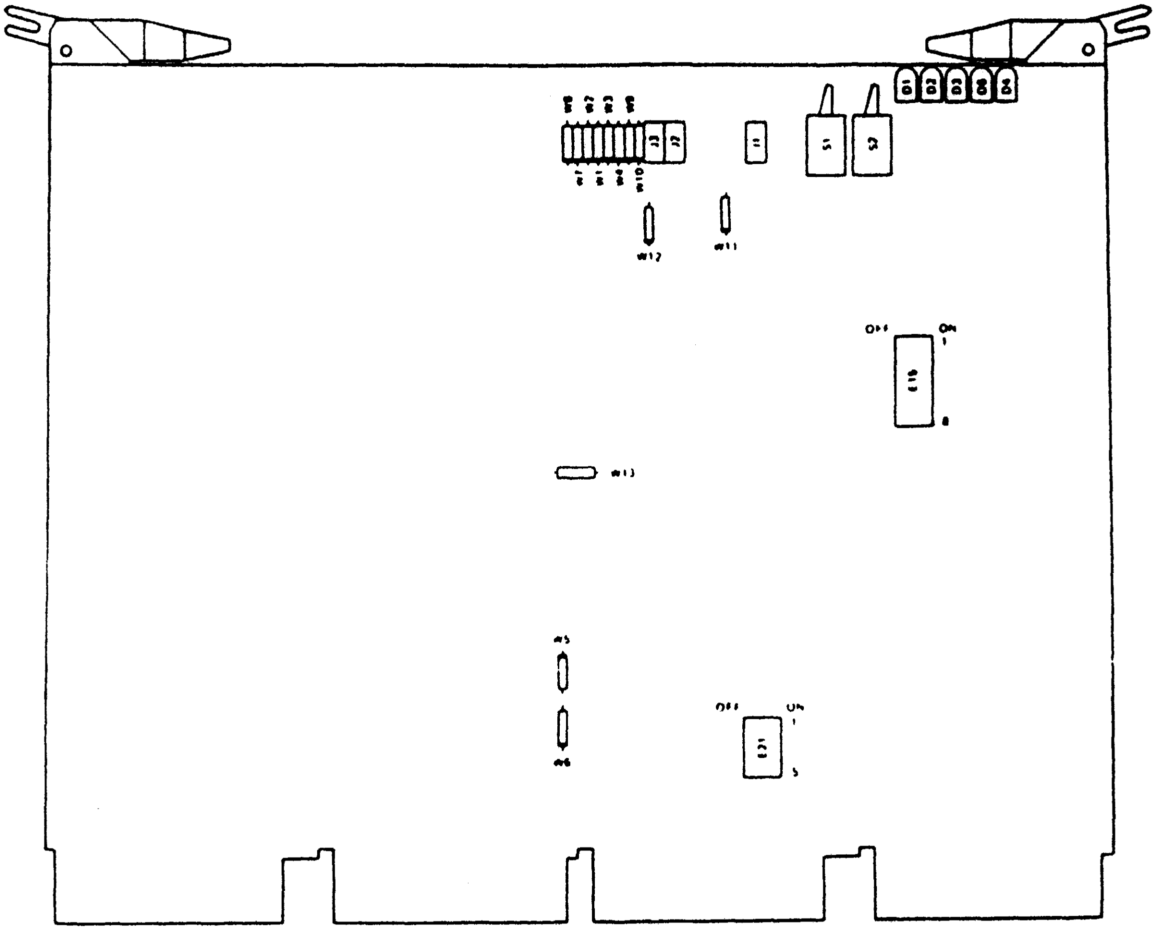
0 is the off or open position; 1 is the on or closed position of the switch.



KDF11-AA Jumper Locations (Rev. C)

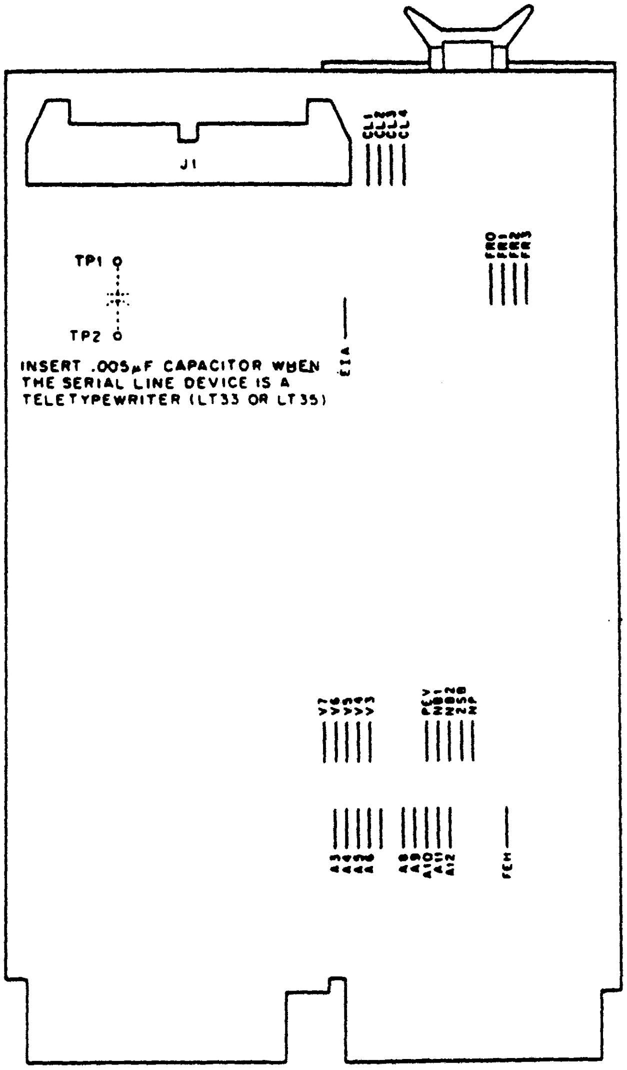


MSV11-D, MSV11-E Switch and Jumper Locations



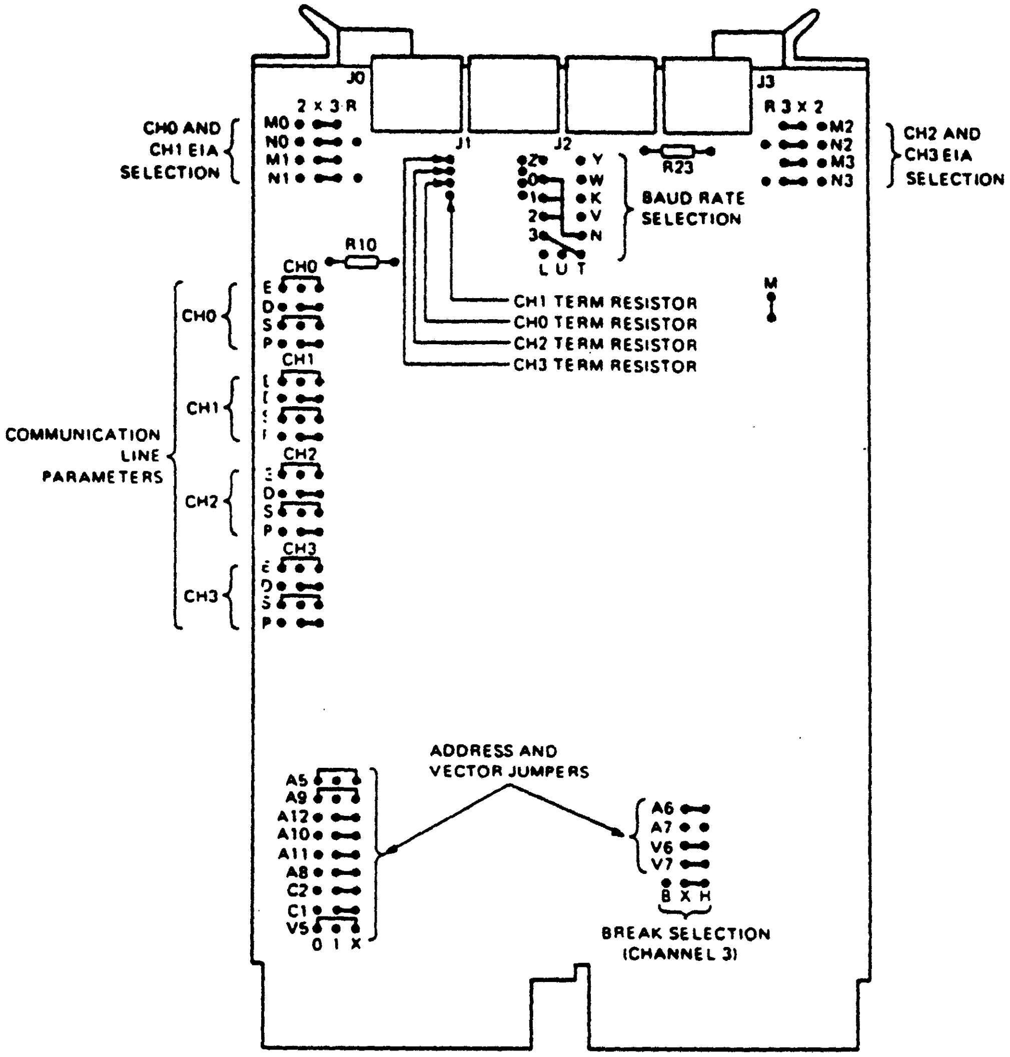
BDV11 Switch and Jumper Locations

DLV11



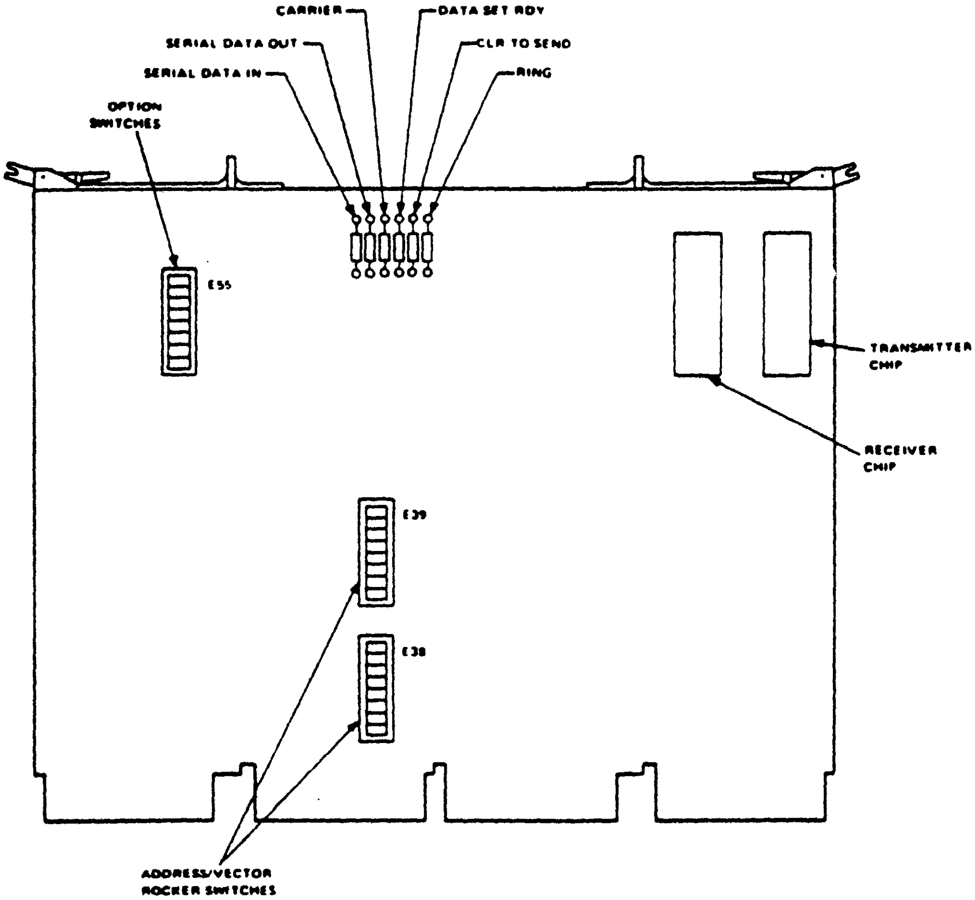
INSERT .005μF CAPACITOR WHEN THE SERIAL LINE DEVICE IS A TELETYPEWRITER (LT33 OR LT35)

DLV11 Jumper Locations

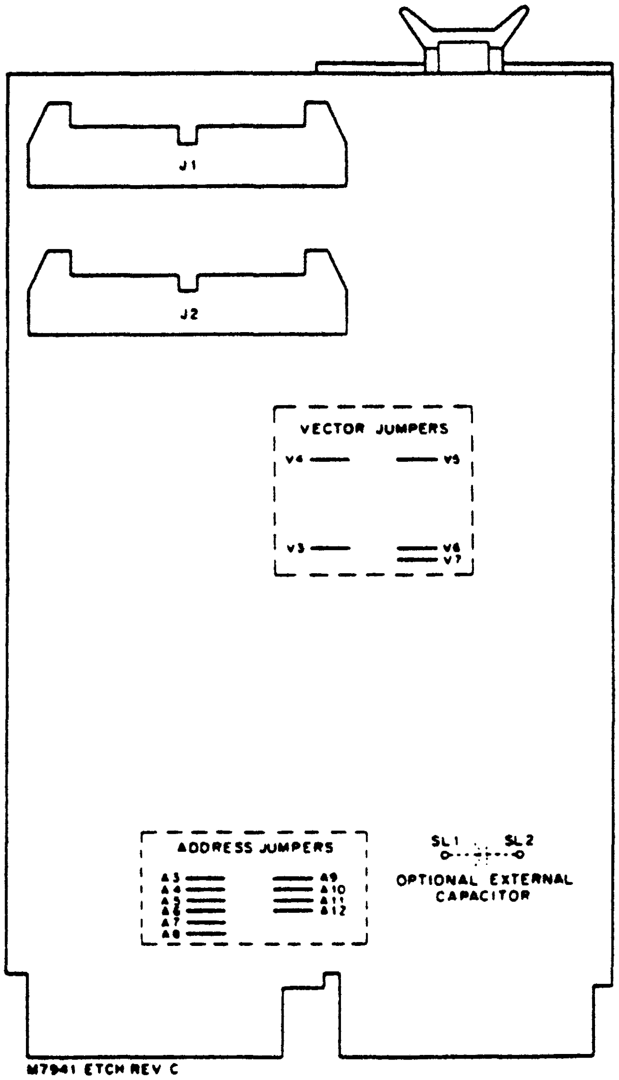


DLV11-J Jumper Locations

DUV11



DUV11 (M7951) Major Components



DRV11 Jumper Locations

- o E => X to 0
- o D => X to 1
- o S => X to 0
- o P => X to 1

5. Speed selection

- o U = 150 Baud
- o T = 300 Baud
- o V = 600 Baud
- o W = 1200 Baud
- o Y = 2400 Baud
- o L = 4800 Baud
- o N = 9600 Baud
- o K = 19200 Baud
- o Z = 38400 Baud

Jumper from 0, 1, 2 or 3 to one of the above to select appropriate clock rate.

4.5 DRV-11 (M7941) Parallel Interface

To set up the standard addresses and vectors on the DRV11, it is necessary to install and/or remove several wire straps.

4.5.1 Address Selection

The DRV11 address is encoded in 10 address straps A12 through A3, A12 being the most significant.

```
     A3 o    o      o    o A9
     A4 o    o      o    o A10
     A5 o    o      o    o A11
     A6 o    o      o---o A12
     A7 o    o
     A8 o    o
o A3  => install jumper for DRV11  2 and  4
o A4  => install jumper for DRV11  3 and  4
o A12 => install jumper for all DRV11s
```

4.5.2 Vector selection

Vector selection is accomplished by installing and/or removing straps between v7 through v3.

```
     V4 o    o     o---o V5


     V3 o    o     o    o V6
                   o    o V7
o V3  => install jumper for DRV11  2
o V4  => install jumper for DRV11  2 and  3
o V5  => install jumper for DRV11  1
o V6  => install jumper for DRV11  2,  3 and  4
o V7  => install jumper for DRV11  2,  3 and  4
```

4.4 DPV-11 (M8020) and DUV-11 (M7951) Synchronous Interfaces

The DPV11 is now the DEC standard synchronous interface board for the LSI-11. It is configured using wire-wrap straps. The following are standard straps that are factory set and need only be checked.

W1-W2
W3-W4
W5-W6
W8-W9
W18-W20
W19-W21
W22-W23
W24-W26
W25-W26
W27-W26
W28-W26

The following are addressing straps and are unique for each DPV11.

W29-W35 for DPV 1 and 3
W29-W34 for DPV 2 and 3
W29-W37 for DPV 4

W43-W46 for all DPVs
W45-W46 for DPV 2 and 4
W44-W46 for DPV 3 and 4

W31,W32,W33,W36,W38,W39,W40,W41,W42 should not be connected to anything.

The DUV11 is an older synchronous serial interface supported for compatibility. The following defines the switch settings that must be used.

1. Option switches are set to allow single character synchronizing unless the connection is Micro-node to Micro-node (Remote nodes).

```
                 E55
+--------------------------------+
| 1 | 0 | 0 | x | 0 | 0 | 0 | 0 |
+--------------------------------+
  1   2   3   4   5   6   7   8
 x <= 1(on) if connection is to TYMNET otherwise 0 (off).
```

4.3 BDV-11 (M8012) Bootstrap

1. Remove jumper W5

2. Install jumper W13

   (If W13 does not exist on your board, carefully install the jumper as indicated in the DEC installation guide).

3. Install the EPROM in the socket indicated (49).

4. Configure both switches as indicated.

```
                                      Halt Run
 \:/                                     | /   /          \:/
-----------------------------------------|/---/---------------
| --- --- --- --- --- ---                [1]  [1]  0 0 0 0 0   |
| | | | | | | | | | | | |                S1  S2              |
| | | | | | | | | | | | |                        ---         |
| | | | | | | | | | | | |                        |E|         |
| --- --- --- --- --- ---                        |1|         |
|                                                |5|         |
| --- --- --- --- --- ---                        ---         |
| | | | | | | | | | | | |                                    |
| | | | | | | | | | | | |                                    |
| | | | | | | | | | | | |                                    |
| --- --- --- --- --- ---                                    |
|                    o....o<--- Install a Jumper here        |
| --- --- --- --- --- ---       (W13)                        |
| | | |D| |*| | | | | | |                                    |
| | | |E| |*|<----------------- Install EPROM here           |
| | | |C| |*| | | | | | |                                    |
| --- --- --- --- --- ---       ---                          |
|                               |E|                          |
| --- --- --- --- --- ---       |3|                          |
| | | | | |D| | | | | | |     o-|2|                          |
| | | | | |E| | | | | | | .-->: ---                          |
| | | | | |C| | | | | | | |   :         ---                  |
| --- --- --- --- --- --- | o-o         |E|                  |
|         Remove this     | | |         |2|                  |
|           Jumper (W5) --.     ---     |1|                  |
|                               |E|     ---                  |
|                               |3|                          |
|                               |1|                          |
|                               ---                          |
--           ----           --------         ----           --
  |         |    |         |        |       |    |         |
  |         |    |         |        |       |    |         |
   ---------      ---------          -------      ---------
```

4.0 Configuring the Boards

This section describes the way in which the component boards must be configured to make them work in a TYMBASE.

It is assumed that the reader has nearby a copy of the relevant DEC reference handbook.

4.1 Processor board

The processor must be configured to jump to location 173000 upon power-up. This is done by the insertion of jumper W5 and the removal of jumper W6. On the LSI-11/23 the jumpers W4 and W7 must be removed and the jumper W8 must be installed.

4.2 External Interfaces

The following is a list of 'standard' addresses and vectors that can be assigned to various interface boards in a TYMBASE configuration.

Device	Address	Vector
DRV11 1	167770	330
DRV11 2	167760	40
DRV11 3	167750	50
DRV11 4	167740	70
DPV11 1	160010	400
DPV11 2	160020	410
DPV11 3	160030	420
DPV11 4	160040	430
DUV11 1	160010	400
DUV11 2	160020	410
DUV11 3	160030	420
DUV11 4	160040	430
DZV11 1	160100	440
DZV11 2	160110	450
DZV11 3	160120	460
DZV11 4	160130	470
DLV11 1	176500	300
	176510	310
	176520	320
(cty)	177560	60
DLV11 2	176540	340
	176550	350
	176560	360
	176570	370
DLV11 3	176600	140
	176610	150
	176620	160
	176630	170
DLV11 4	176640	200

device). The line time clock must be enabled (i.e., jumper W1 on the backplane must be installed).

2.4 Asynchronous Interfaces

Two serial asynchronous port options can be used in a TYMBASE. These are the DLV11-J (M8043) and the DZV11 (M7957).

The DLV11-J is a dual height module with 4 separate serial line units (SLU). A DLV11 SLU interfaces one asynchronous serial line I/O device (either 20ma or EIA) to the LSI-11 bus (without modem control). One DLV11 port must be configured as a console terminal (cty).

A DZV11 multiplexes asynchronous serial line I/O for 4 devices. It is a quad height module and has full modem control.

2.5 Synchronous Interfaces

Two synchronous serial line interfaces are available for the TYMBASE configuration, the DPV-11 (M8020) and the DUV-11 (M7951).

The DPV-11 is a dual height board and the DUV-11 is a quad height board described in the 1978-1979 edition of "Memories and Peripherals".

2.6 Backplanes

Two boxes can be used for the TYMBASE configuration, the BA11-N or large 9 slot, and the PDP-11/03 or small 4 slot box (BA11-M) with an optional expansion box (BA11-ME). These boxes are described in the 1979-1980 edition of "Microcomputer Interfaces Handbook".

For the 9 slot box, the slots serviced by the Q-bus, are the leftmost two (as viewed from the rear of the box). The rightmost two slots do not interface to the bus.

All slots in the 4 slot box (and the expansion box), are serviced by the Q-bus.

1.0 Introduction

This document describes the hardware configuration specification for the TYMBASEs based on LSI-11 computers.

It is assumed that the reader is familiar with the architecture of the LSI-11 and that the appropriate DEC hardware manuals are at hand and available for reference.

For Tymshare employees only

MICRONODE

FAILURE RECOVERY PROCEDURES

October 1980
Journal Number 75427

Office Automation Division
Tymshare, Inc.
20705 Valley Green Drive
Cupertino, California 95014

INTRODUCTION

The document tells you how to bring up a micronode, also called an LSI11, that is down. The procedure documented here is very simple, but very important, for the micronode is what connects the System XXV (a "Foonly" machine) to TYMNET, the network that attaches users to hosts and hosts to each other. All information passed to and from the System XXV and the other AUGMENT hosts, as well as all interaction with individual users at their terminals, must go through TYMNET and thus must be fed through the micronode.

There is one exception to this rule. While virtually everything communicates with the System XXV by going through TYMNET to the micronode and then through the micronode to the System XXV, the console terminal is wired directly to the System XXV. This means the console terminal and the System XXV communicate with each other without going through the micronode and TYMNET. Thus they can continue to interact even when TYMNET and the micronode are not functioning correctly.

System XXV calls the micronode TYMBASE and uses a software module called TYMSRV to connect to it. Thus, when you see the term "TYMBASE" in a console message, it means the micronode; the term TYMSRV means the System XXV program that communicates with the micronode.

HOW TO TELL IF THE MICRONODE IS DOWN

There is no one sure way to tell if the micronode is down, but the following symptoms probably mean it is not functioning properly.

1) The console is printing messages indicating problems with TYMBASE or TYMSRV. For example, "TYMBASE APPARENTLY DISABLED" or "TYMSRV: FAILED TO RESYNC WITH NODE".

2) You are getting irate calls from users who cannot reach the System XXV or have been detached.

3) The RUN light is off.

BRINGING UP THE MICRONODE

Introduction

The micronode is what connects users to the System XXV and allows them to use it. For this reason, you should never bring up the micronode and open the system to users unless you are sure the system is in good shape. If you are recycling the micronode after a system crash, make sure the system has come up correctly as described in the document "System XXV AUGMENT Host Failure Recovery Procedures", in the section called "Normal Bringup".

The following procedure for bringing up the micronode (also referred to as "rebooting") has two parts. First, you actually bring up the micronode. This is described in steps 1 through 18 of the "Procedure Summary". The process is not complicated; however, the procedure may not always work the first time. If you try to bring up the micronode and do not succeed, simply retry the same procedure several times. If you still do not succeed, you may want to have the micronode examine its memory for hardware problems. To find out how to do this, see the section titled "Repeated Failure" under the heading "Errors and Recoveries".

Once you are successful in bringing up the micronode, you then need to check the micronode and TYMNET by attempting to log in to a different AUGMENT host. This procedure is documented in steps 19 through 21 in the "Procedure Summary".

Procedure Summary

1) Make sure the switch on the side of the console terminal is at the position marked "Foonly" and type a few carriage returns. You should see the prompt "@" or "!". If you get "@", you are not enabled. Type "ena<CR>" to enable yourself; you should then get the "!" prompt.

2) Type "<CTRL e>tymnet<SP>off<CR>".

3) Check the power light, the right light on the front of the micronode, to ensure that the micronode has power. When the power is on, the power light is lit. If the light is off, turn on the power switch on the back of the micronode.

4) Put off (down) the three switches on the front of the micronode.

5) Put the switch on the side of the console terminal to the position marked "micronode". (This is usually down.)

6) Put on (up) LTC, the switch on the right.

7) Put on (up) HALT, the switch on the left.

8) Put on INIT, the middle switch. The RUN light (the left light on the front of the micronode) should light.

9) Type a few carriage returns on the console terminal. You should get "$", the prompt for the micronode's Command Processor.

10) Type "DO<CR>"; "0" here is a zero. Note that all commands to the micronode must be capitalized.

11) Turn the switch on the side of the console to the position labeled "Foonly". (This is usually up.)

12) Type a few carriage returns; you should then get the prompt "!".

13) Type "run<SP><system>nodebo<ESC><CR>".

14) When you see "MICRONODE IMAGE FILENAME >", type "<ESC><CR>".

15) You will get messages that the System XXV (Foonly) is trying to reset, load, and start the micronode. If this process is successful, you will get the message "NODE BOOTSTRAP COMPLETED SUCCESSFULLY" followed by some system messages.

16) When the system messages are over and you get the "!" prompt, type "<CTRL e>tymnet<SP>on<CR>".

17) You may get some more system messages, and a few minutes later, you should see your last message, "TYMBASE UP".

18) If you do not get "TYMBASE UP", or you get the message "MUST REPEAT ENTIRE REBOOT PROCEDURE", something has gone wrong. Turn the power off and back on, and begin again with step 1.

19) Once the micronode comes up, push the switch on the side of the console to the position marked "micronode" (usually down) and type a few carriage returns. If you then see "Please log in:", the micronode is OK. If you are NOT asked to log in, the micronode has come up wrong. Push the switch on the side of the console terminal to the position marked "Foonly", and start the bringup procedure again from step 1.

20) If the micronode is good and you are asked to log in, try to log in to another AUGMENT host. If you succeed, TYMNET is OK. Log out from the other AUGMENT host, push the switch on the side of the console terminal to the position marked "Foonly" (usually up), and again type a few carriage returns. Once you get the "!" prompt, the console terminal is again communicating with the System XXV.

21) If you do not succeed in logging in to another AUGMENT host, try again. If you cannot log in within five minutes, something may be wrong with TYMNET. Call the TYMNET Network Control Center and report the problem.

Procedure Description

The NEXILIS micronode connects the System XXV (or Foonly) to the TYMNET network, which in turn connects it to other AUGMENT hosts and to AUGMENT users. The micronode then funnels information to and from the System XXV, allowing users and other hosts to interact with the system. To do this, the micronode runs a program stored in its memory. When the micronode goes down, this program is lost and must be reloaded from the System XXV where it is permanently stored. The micronode bringup procedure first prepares the micronode to receive this program and then instructs the System XXV to send it over. Once the program has been successfully transmitted and the micronode is running it, the System XXV must resynchronize with the micronode so they can pass information back and forth. If the System XXV succeeds in synchronizing with the micronode, then the micronode has probably come up correctly. Once the micronode is up, to make sure that the micronode and TYMNET really are all right, try to use them to log in to another AUGMENT host. If you can do this, then you know the bringup procedure has worked.

You begin the micronode bringup by making sure that you are at the AUGUST Operating System's EXEC, and that you are enabled. Make sure the switch on the side of the console terminal is at the position marked "Foonly", and type a few carriage returns. If you get the prompt "!", you are already enabled. If your prompt is "@", you are not enabled. Since you must be enabled to bring up the micronode, if you get the "@" prompt, enable yourself by typing "ena<CR>" at the EXEC. When you enable, you tell the system that you are a person with special powers who should be allowed to do things normal users cannot do. After you enable, you will be prompted with "!".

You now prepare the System XXV and the micronode for rebooting. Since you do not want the System XXV to use the micronode as you reboot it, you begin by typing "<CTRL e>tymnet<SP>off<CR>". This tells the System XXV that TYMNET is no longer available. Next, check the micronode itself to make sure it has power. The right light on the front of the micronode is lit when the power is on. If this light is off, turn on the power switch at the back of the micronode. Now you need to reset the micronode before bringing it up; put off (down) the three switches on the front of the micronode. Finally, so that you can communicate with the micronode, put the switch on the console teletype to the position marked "micronode" (usually down). Doing this connects the console terminal to the micronode instead of the System XXV.

You now are ready to begin bringing up the micronode. You do this by putting on the three switches on the front of the micronode. Make sure that you put on these switches in the proper order; putting them on in the wrong order, can cause serious damage. First, put on (up) the right switch on the micronode labeled "LTC". This switch turns on the line clock timer, an internal timer that makes sure computer procedures happen at the correct intervals and are synchronized. Next, put on (up) the left switch on the micronode labeled "HALT". Last, put on the middle switch on the micronode labeled "INIT".

When you put on the "INIT" switch, the RUN light (the right light on the micronode) should come on. The micronode is now running a small Command Processor. Type a few carriage returns on the console; you should get a "$", the Command Processor's prompt. If you get an "@" prompt instead of the "$", check to make sure the switch on the console terminal is pushed to the position marked "micronode". If the switch accidentally has been left at the position marked "Foonly", then the "@" you are getting is the prompt for the EXEC in AUGUST. Push the switch to the position marked "micronode", and again try typing a few carriage returns; you should now get the "$" prompt. If the switch was already set correctly when you got the "@" prompt, you have accidentally entered the micronode's debugger. This is a serious problem; immediately start the bringup procedure again with step 4.

Once you get the "$" prompt, the micronode is ready and waiting to be rebooted. Type "DO<CR>". Note that "0" is zero, and that all commands to the micronode must be capitalized. The DO command tells the micronode that it should prepare to take its reboot program from the System XXV. To tell the System XXV to deliver the reboot program, put the switch on the side of the console to the position marked "Foonly". This reconnects the console terminal with the System XXV. When you get the AUGUST Exec prompt "!", type "run<SP><system>nodebo<CR>". The System XXV will then run the micronode booting program. This program will feed the micronode the program it needs to run.

Once the System XXV has loaded the micronode booting program, it will print, "MICRONODE IMAGE FILENAME >". The System XXV is asking you for the name of the image file it should give the micronode. An image file is a file that can be read directly into memory. In this case, it is like a snapshot of the memory as it should be for the micronode to operate correctly. When you see "MICRONODE IMAGE FILENAME >", type "<ESC>". The program will then fill out the name of the correct image file followed by the comment "[Old Version]". Type a carriage return to confirm this.

The System XXV will now attempt to deliver the micronode image file to the micronode and to bring it up. As it attempts this, the System XXV will print various messages about its progress, such as, "resetting the micronode", "loading the micronode", or "starting the micronode". If and when this process is finished, the System XXV tell you "NODE BOOTSTRAP COMPLETED SUCCESSFULLY".

Once the micronode has the program it needs to run, the System XXV can use TYMNET. To tell this to the System XXV, type "<CTRL e>tymnet<SP>on<CR>". The System XXV will then try to synchronize itself with the micronode and attempt to send and receive TYMNET information. If this is successful, you will see some more system messages, and after a while, "TYMBASE UP".

After waiting several minutes, if you do not get the message "TYMBASE UP", or if you get the message "MUST REPEAT ENTIRE REBOOT PROCEDURE", something is wrong with the way the micronode came up. Either the System XXV did not deliver the image file successfully, or it could not resynchronize with the micronode. At this point, simply restart the bringup procedure with step 1.

Once the micronode does come up and you see the message "TYMBASE UP", you still need to make sure the micronode and TYMNET really are functioning correctly. To do this, you use the console terminal and micronode as if you were a regular user trying to log in to an AUGMENT host. Turn the switch on the side of the console to the position marked "micronode". This connects the console terminal to the micronode instead of connecting it directly to the System XXV. Next, type a few carriage returns. The micronode should notice the existence of the console terminal, treat it just like any other terminal, and assume it must be waiting to log in. Thus, you should see "please log in:". If this happens, the micronode has come up correctly. If you are not asked to log in, there is still something wrong. Push the switch on the side of the console terminal to the position marked "micronode", and again begin the bringup procedure at step 1.

If the micronode does correctly ask you to log in, you now want to find out if this micronode can communicate with the rest of TYMNET. Test this by trying to log in to another AUGMENT host. If you can log in to another AUGMENT host, the micronode bringup has been completely successful. If after five minutes all attempts to reach another AUGMENT host have failed, call the TYMNET Network Control Center, and report the problem to them.

If you can log in to another AUGMENT host without any trouble, you know that the micronode is fine, that the connection with TYMNET is good, and that information is successfully passing between the micronode and the network. Log out from the AUGMENT host where you just logged in, and push the switch on the console terminal back to the position marked "Foonly". This once again connects the console terminal directly to the System XXV, and you should see the AUGUST EXEC prompt, either "!" or "@". Once you have done this, you have completed the procedure of bringing up the micronode and testing it.

Errors and Recoveries

"Must Repeat Entire Reboot Procedure"

If you get this error message, repeat the entire procedure.

Repeated Failure to Reboot the Micronode

Introduction

If the System XXV repeatedly fails to reboot the micronode, it may mean that the micronode itself has a memory problem. To determine if this is the case, you run a short program that checks the memory of the micronode.

Procedure Summary

1) Repeat steps 1 though 9 of the regular recovery procedure.

2) At step 10, where you would normally type "DO<CR>", instead type "XM<CR>". Note that all commands to the micronode must be capitalized.

3) The micronode will now examine its memory. If it types out a series of numbers, there is something wrong with the memory. Call Tymshare Maintenance.

4) If the micronode does not type out anything, the memory is good, but you may still have a hardware problem. Push the switch on the side of the console terminal to the position marked "Foonly" and retry the entire procedure from step 1. Do this a couple of times; if you still are not successful, notify Tymshare Maintenance.

ARPANET BACKPLANE WIRING FOR THE F4 ON THE I/O PANEL

```
                                      ---------
                                     | 14     1 |
                                     |          |
                                     |          |
                                     |          |
ANSLEY 26 PIN CONNECTOR              |          |
ON THE I/O PANEL                     |          |
                                     |          |
                                     |          |
                                     | 26    13 |
                                     |----------|
```

COLOR -----	I/O PANEL 26 PIN CONNECTOR ---------	SLOT A19 (IMP CONTROLLOR SLOT) ---------------------
WHITE	1	C2A20
BLUE	14	C2B20
WHITE	2	C2A19
BLUE	15	C2B19
WHITE	3	C2A18
BLUE	16	C2B18
WHITE	4	C2A17
BLUE	17	C2B17
	6+	
	19+	
WHITE	7	C2A35
BLUE	20	C2B36
WHITE	9	C2A22
BLUE	22	C2B22
WHITE	10	C2A23
BLUE	23	C2B23
WHITE	11	C2A25
BLUE	24	C2B25
WHITE	12	C2A24
BLUE	25	C2B24
WHITE	13	GND
BLUE	26	GND

```
*******************************************************************
*     ALL PINS WITH A + FOLLOWING THE PIN NUMBER SHOULD BE JUMPERED    *
*                TOGETHER                                              *
*******************************************************************
```

ARPANET CABLE WIRE LIST

FOONLEY END		IMP END		
COLOR	PIN	COLOR	PIN	SIGNAL NAME
BROWN	1	BROWN	17	LAST IMP BIT (+)
TAN	2	BLACK	35	LAST IMP BIT (-)
RED	3	RED	19	DATA: IMP TO HOST (+)
TAN	4	BLACK	37	DATA: IMP TO HOST (-)
ORANGE	5	ORANGE	15	THERE IS YOUR IMP BIT (+)
TAN	6	BLACK	33	THERE IS YOUR IMP BIT (-)
YELLOW	7	YELLOW	13	READY FOR NEXT HOST BIT (+)
TAN	8	BLACK	32	READY FOR NEXT HOST BIT (-)
BLUE	11	GREEN	11	HOST MASTER READY
TAN	12	BLACK	12	HOST READY TEST
VIOLET	13	BLUE	9	IMP READY TEST
TAN	14	BLACK	10	IMP MASTER READY
WHITE	17	WHITE	1	READY FOR NEXT IMP BIT (+)
TAN	18	BLACK	2	READY FOR NEXT IMP BIT (-)
BLACK	19	YELLOW	6	THERE IS YOUR HOST BIT (+)
TAN	20	RED	25	THERE IS YOUR HOST BIT (-)
BROWN	21	GREEN	7	LAST HOST BIT (+)
TAN	22	RED	26	LAST HOST BIT (-)
RED	23	BLUE	8	DATA: HOST TO IMP (+)
TAN	24	RED	27	DATA:HOST TO IMP (-)
TAN	26	DRAIN	22	SHIELD--PIG TAIL FROM SHIELD

```
**************************************************************************************
*  THE NON-TAN WIRES ABOVE ARE TWISTED PAIR WITH THE TAN WIRE CLOSEST TO IT          *
*        ON THE FOONLEY END                                                          *
**************************************************************************************
```

FOONLEY END OF ARPANET CABLE CONNECTOR

```
 ---------
| 1     2 |
|         |
|         |
|         |
|         |
|         |
|         |
|         |
| 25   26 |
|_________|
```

DOCUMENT FOR THE WIRING OF THE "CTY" CONN ON THE F4 I/O PANEL

[illegible] BUSS ON LEFT COLUMN OF I/O HEADERS

```
                        ANSLEY 26 PIN CONNECTOR

                           _____________
                           ¦14        1 ¦
                           ¦            ¦
                           ¦            ¦
                           ¦            ¦
                           ¦            ¦
                           ¦            ¦
                           ¦            ¦
                           ¦            ¦
                           ¦            ¦
                           ¦            ¦
                           ¦26        13¦
                           ¦____________¦

FROM ANSLEY, PIN NO.:              TO "A" SECTION PIN NO.:

        1                                  GND       ¦ TWISTED PAIR
        2                                  A23C3A10 ¦
        3                                  A23C3A11  ¦ TWISTED PAIR
        7                                  GND       ¦
        8 JUMPERED TO 20
        ON ANSLEY
```

```
                            VERSATEC.I/O-CONN
                        (ALL SIGNALS TWISTED PAIR)

26 PIN CONN. 1                 CONN. 2                    CONN. 3
                             _____
  1. )  IN01    C3B17       )READY    C3A15               )         C3B25
  2. )  RET     C3B18       )         C3A16               )         C3B26
  3. )  IN02    C3B15       )PRINT    C3C14{*---          )         C3B27
  4. )          C3B16       )_____    C3C15{*---          )         C3B28
  5. )  IN03    C3B13       )PAREN                        )         C3B29
  6. )          C3B14       )___                          )_____    C3B30
  7. )  IN04    C3B11       )SPP      C3C16{*---          )RTTON    C3C12
  8. )          C3B12       )_____    C3C17{*---          )         C3C13
  9. )  IN05    C3B09       )RESET    C3B23{*---          )DXTER    C3A13
 10. )          C3B10       )_____.   C3B24{*---          )______   C3A14
 11. )  IN06    C3B07       )RFFED    C3B33{*---          )TNERON   C3A11
 12. )          C3B08       )_____    C3B34{*---          )_____    C3A12
 13. )  IN07    C3B05       )REOTR    C3C10{*---          )BUFUL    C3A07
 14. )          C3B06       )_____    C3C11{*---          )         C3A08
 15. )  IN08    C3B03       )RLTER    C3B31{*---          )LOSUP    C3A05
 16. )  _____   C3B04       )         C3B32{*---          )         C3A06
 17. )  CLEAR   C3B21{*SWAP )NOPAP    C3A09               )
 18. )          C3B22{*SWAP )_______  C3A10               )_____
 19. )  PICLK   C3B19       )ON LINE  C3A03               )PWRON
 20. )          C3B20       )         C3A04               )
 21. )
 22. )
 23. )
 24. )
 25. )
 26. )             |_____________________|_____________________________|
                       SLOT B05
```

TAPE.I/O-CONN

FOR KENNEDY TAPE DRIVES

ON EACH I/O CONN, SOLDER BUSS STRIP ON I/O CONN PINS 14-26, AND RUN A WIRE TO BACKPLANE GROUND.

I/O 28

FROM BACKPLANE PIN:	TO I/O CONN PIN:
A23C3C21	1
C2C28	2
C3C23	3
(C3)C24	4
C25	6
C26	7
C27	8
A35	9
A28	10
A30	11
A32	12
A34	13

I/O 29

FROM BACKPLANE PIN:	TO I/O CONN PIN:
A23C3B9	1
B10	2
B11	3
B12	4
B13	5
B14	6
B15	7
B2	8
B35	9
B28	10
B30	11
B32	12
B34	13

I/O 30

FROM BACKPLANE PIN:	TO I/O CONN PIN:
A23C3C20	1
C22	2
C28	3
C29	4
C30	5
C2C25	7
C2C26	8
C2C27	9
C3A27	10
A29	11
A31	12
A33	13

I/O 3I

FROM BACKPLANE PIN:	TO I/O CONN PIN:
A23C3B16	1
B17	2
B18	3
B19	4
B20	5
B21	6
B22	7
C10	8
B27	9
B29	10
B31	11
B33	12

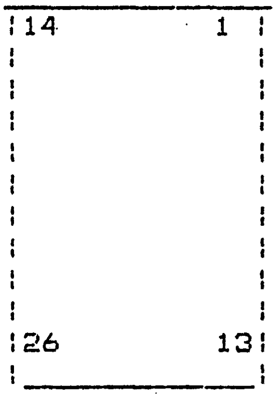
```
TYMNET.I/O-CONN LIST

[illegible] BUSS ON LEFT COLUMN OF I/O HEADERS

BACKPLANE CONN A23                              I/O CONN I12

A23C2  B17                                      2
  "    B18                                      5
  "    B21                                      18
  "    B22                                      19
  "    B23                                      20
  "    B24                                      21
  "    B20                                      9
  "    B25                                      10
  "    B26                                      23

                                  GND PINS:17,6,8,22
                                          (TO BACKPLANE GROUND)



                                                I/O CONN I13

A23C2  B27                                      14
  "    B28                                      2
  "    B29                                      15
  "    C24    TWISTED PAIR                      16
  "    B30  (USE GND FROM GND PINS BELOW)       17
  "    B31                                      5
  "    B32                                      18
  "    C17    TWISTED PAIR                      19
  "    B19  (USE GND FROM GND PINS BELOW)       20
  "    B19                                      21
  "    B33    TWISTED PAIR                      23
         (USE GND FROM GND PINS BELOW)

                                        GND PINS:1,3,4,6,7,8,9,10
                                        (TO BACKPLANE GROUND)


                                                I/O CONN I14
```

```
A23C2  B34    TWISTED PAIR                       2
  "    A19  (USE GND FROM GND PINS BELOW)        3
  "    A19                                       4
  "    C22    TWISTED PAIR                       5
  "    A32  (USE GND FROM GND PINS BELOW)        6
  "    A31                                       19
  "    A30                                       7
  "    C19    TWISTED PAIR                       8
  "    A29  (USE GND FROM GND PINS BELOW)        9
  "    A28                                       22
  "    A27                                       10

                                         GND PINS: 17, 18, 20, 21, 23
                                         (TO BACKPLANE GROUND)


                                                 I/O CONN I15

A23C2  A26                                       1
  "    A25                                       14
  "    A20                                       15
  "    A24                                       3
  "    A23                                       4
  "    A22                                       5
  "    A21                                       6
  "    A18                                       19
  "    A17                                       22

                                         GND PINS: 2, 7, 8, 9, 10, 16, 18
                                         (TO BACKPLANE GROUND)
```

DOCUMENT FOR THE WIRING OF THE "DISK CONTROL" CONNS ON THE F4 I/O PANEL

NO BUSS ON LEFT COLUMN OF I/O HEADERS

ANSLEY 26 PIN CONNECTOR



THESE ARE ALL DIFFERENTIAL PAIRS AND AS SUCH, DON'T GET THEIR LEFT COLUMN PINS GROUNDED.

26 PIN ANSLEY CONNECTOR: SLOT A7:

DC 1:

PIN:	PIN:	SIGNAL NAME:
1 {WHITE}	A7C1B13 {WHITE}	SET CYCLE -
14 {BLUE}	A7C1B12 {BLUE}	SET CYCLE +
2 {WHITE}	A7C1B15 {WHITE}	SET HEAD -
15 {BLUE}	A7C1B14 {BLUE}	SET HEAD +
3 {WHITE}	A7C1B17 {WHITE}	CONTROL -
16 {BLUE}	A7C1B16 {BLUE}	CONTROL +
4 {WHITE}	A7C1B19 {WHITE}	BUS 0 -
17 {BLUE}	A7C1B18 {BLUE}	BUS 0 +
5 {WHITE}	A7C1B21 {WHITE}	BUS 1 -
18 {BLUE}	A7C1B20 {BLUE}	BUS 1 +
6 {WHITE}	A7C1B23 {WHITE}	BUS 2 -
19 {BLUE}	A7C1B22 {BLUE}	BUS 2 +
7 {WHITE}	A7C1B25 {WHITE}	BUS 3 -
20 {BLUE}	A7C1B24 {BLUE}	BUS 3 +
8 {WHITE}	A7C1B27 {WHITE}	BUS 4 -
21 {BLUE}	A7C1B26 {BLUE}	BUS 4 +

```
9 {WHITE}                      A7C1B29 {WHITE}      BUS 5 -
22 {BLUE}                      A7C1B28 {BLUE}       BUS 5 +

10 {WHITE}                     A7C1B31 {WHITE}      BUS 6 -
23 {BLUE}                      A7C1B30 {BLUE}       BUS 6 +

DC 2:

PIN:                           PIN:
1 {WHITE}                      A7C1B33 {WHITE}      BUS 7 -
14 {BLUE}                      A7C1B32 {BLUE}       BUS 7 +

2 {WHITE}                      A7C1B35 {WHITE}      BUS 8 -
15 {BLUE}                      A7C1B34 {BLUE}       BUS 8 +

3 {WHITE}                      A7C1C3 {WHITE}       BUS 9 -
16 {BLUE}                      A7C1C2 {BLUE}        BUS 9 +

4 {WHITE}                      A7C1C10 {WHITE}      DEV ENABLE -
17 {BLUE}                      A7C1C9 {BLUE}        DEV ENABLE +

5 {WHITE}                      A7C1A7 {WHITE}       UNSAFE -
18 {BLUE}                      A7C1A6 {BLUE}        UNSAFE +

6 {WHITE}                      A7C1A9 {WHITE}       SEEK ERROR -
19 {BLUE}                      A7C1A8 {BLUE}        SEEK ERROR +

7 {WHITE}                      A7C1A11 {WHITE}      ON CYCLE -
20 {BLUE}                      A7C1A10 {BLUE}       ON CYCLE +

8 {WHITE}                      A7C1A3 {WHITE}       INDEX -
21 {BLUE}                      A7C1A2 {BLUE}        INDEX +

9 {WHITE}                      A7C1A13 {WHITE}      UNIT READY -
22 {BLUE}                      A7C1A12 {BLUE}       UNIT READY +

10 {WHITE}                     A7C1A17 {WHITE}      ADDRESS MARK DLT -
23 {BLUE}                      A7C1A16 {BLUE}       ADDRESS MARK DLT +

DC 3:

PIN:                           PIN:
1 SKIP                         NONE                 BUSY -
14 SKIP                        NONE                 BUSY +

2 {WHITE}                      A7C1B11 {WHITE}      SEL ENABLE -
15 {BLUE}                      A7C1B10 {BLUE}       SEL ENABLE +

3 {WHITE}                      A7C1B9 {WHITE}       SELECT 0 -
16 {BLUE}                      A7C1B8 {BLUE}        SELECT 0 +
4 {WHITE}                       A7C1B7 {WHITE}        SELECT 1 -
17 {BLUE}                       A7C1B6 {BLUE}         SELECT 1 +

5 {WHITE}                       A7C1A5 {WHITE}        SECTOR MK -
18 {BLUE}                       A7C1A4 {BLUE}         SECTOR MK +

6 {WHITE}                       A7C1B5 {WHITE}        SELECT 2 -
19 {BLUE}                       A7C1B4 {BLUE}         SELECT 2 +

7 {WHITE}                       A7C1B3 {WHITE}        SELECT 3 -
20 {BLUE                        A7C1B2 {BLUE}         SELECT 3 +

8 {WHITE}                       A7C1A15 {WHITE}       WRITE PROTECT -
21 {BLUE}                       A7C1A14 {BLUE}        WRITE PROTECT +

9 {WHITE}                       NO CONNECTION         SEQ IN
22 {BLUE}                       NO CONNECTION         HOLD

10 {WHITE}                      A7C1A35 {WHITE}       BUS 10 -
23 {BLUE}                       A7C1A34 {BLUE}        BUS 10 +

^L
```

PART OF ANSLEY 26 PIN CONNECTOR:

PIN #:
1 TO-4 AND TO GROUND
4 TO-
15 TO- DAISY CHAIN ALL TOGETHER,
18 TO-
7 TO-
21 TO-
11 TO-
25 TO-GROUND

		CONNECTOR:
	26 PIN ANSLEY CONNECTOR:	I/O 9:
SIGNAL NAME:	PIN #:	DD 1:
SERVO CLOCK	2 {WHITE}	A7C2C30 {WHITE}
SERVO CLOCK	14 {BLUE}	A7C2C29 {BLUE}
READ DATA -	3 {WHITE}	A7C3A24 {WHITE}
READ DATA +	16 {BLUE}	A7C3A23 {BLUE}
READ CLOCK	5 {WHITE}	A7C3A22 {WHITE}
READ CLOCK	17 {BLUE}	A7C3A21 {BLUE}
WRITE CLOCK	6 {WHITE}	A7C3A34 {WHITE}
WRITE CLOCK	19 {BLUE}	A7C3A33 {BLUE}
WRITE DATA	8 {WHITE}	A7C3B35 {WHITE}
WRITE DATA	20 {BLUE}	A7C3A35 {BLUE}
SELECTED -	22 {WHITE}	A7C3A26 {WHITE}
SELECTED +	9 {BLUE}	A7C3A25 {BLUE}
SEEK END	10 {WHITE}	A7C3A28 {WHITE}
SEEK END	23 {BLUE}	A7C3A27 {BLUE}

PART OF ANSLEY 26 PIN CONNECTOR:

PIN #:
1 TO 4 AND TO GROUND.
4 TO-
15 TO- DAISY CHAIN ALL TOGETHER,
18 TO-
7 TO-
21 TO-
11 TO-
25 TO GROUND.

		CONNECTOR:
	ANSLEY 26 PIN CONNECTOR:	I/O 10:
SIGNAL NAMES:	PIN #:	DD 2:
SERVO CLOCK	2 {WHITE}	A7C3B22 {WHITE}
SERVO CLOCK	14 {BLUE}	A7C3B21 {BLUE}

READ DATA -	3 {WHITE}	A7C3B26 {WHITE}
READ DATA +	16 {BLUE}	A7C3B25 {BLUE}
READ CLOCK	5 {WHITE}	A7C3B24 {WHITE}
READ CLOCK	17 {BLUE}	A7C3B23 {BLUE}
WRITE CLOCK	6 {WHITE}	A7C3C3 {WHITE}
WRITE CLOCK	19 {BLUE}	A7C3C2 {BLUE}
WRITE DATA	8 {WHITE}	A7C3C10 {WHITE}
WRITE DATA	20 {BLUE}	A7C3C9 {BLUE}
SELECTED -	22 {WHITE}	A7C3B28 {WHITE}
SELECTED +	9 {BLUE}	A7C3B27 {BLUE}
SEEK END	10 {WHITE}	A7C3B30 {WHITE}
SEEK END	23 {BLUE}	A7C3B29 {BLUE}

```
PART OF ANSLEY 26 PIN CONNECTOR:
        PIN #:
        1 TO 4 AND TO GROUND.
        4 TO-
        15 TO-  DAISY CHAIN ALL TOGETHER,
        18 TO-
        7 TO-
        21 TO-
        11 TO-
        25 TO GROUND.
```

		CONNECTOR:
	ANSLEY 26 PIN CONNECTOR:	I/O 11:
SIGNAL NAME:	PIN #:	DD 3:
SERVO CLOCK	2 {WHITE}	A7C3C12 {WHITE}
SERVO CLOCK	14 {BLUE}	A7C3C11 {BLUE}
READ DATA -	3 {WHITE}	A7C3C16 {WHITE}
READ DATA +	16 {BLUE}	A7C3C15 {BLUE}
READ CLOCK	5 {WHITE}	A7C3C14 {WHITE}
READ CLOCK	17 {BLUE}	A7C3C13 {BLUE}
WRITE CLOCK	6 {WHITE}	A7C3C26 {WHITE}
WRITE CLOCK	19 {BLUE}	A7C3C25 {BLUE}
WRITE DATA	8 {WHITE}	A7C3C28 {WHITE}
WRITE DATA	20 {BLUE}	A7C3C27 {BLUE}
SELECTED -	22 {WHITE}	A7C3C18 {WHITE}
SELECTED +	9 {BLUE}	A7C3C17 {BLUE}

```
SEEK END        10 {WHITE}                        A7C3C20 {WHITE}
SEEK END        23 {BLUE}                         A7C3C19 {BLUE}

PART OF THE ANSLEY 26 PIN CONNECTOR:
        PIN #:
        1 TO 4 AND GROUND.
        4 TO-
        15 TO-  DAISY CHAIN ALL TOGETHER,
        18 TO-
        7 TO-
        21 TO-
        11 TO-
        25 TO GROUND.
```

DOCUMENT FOR THE WIRING OF THE "IMP" CONN ON THE F4 I/O PANEL

NO BUSS ON LEFT COLUMN OF I/O HEADERS

```
                        ANSLEY 26 PIN CONNECTOR

                           ______________
                           ¦14         1 ¦
                           ¦             ¦
                           ¦             ¦
                           ¦             ¦
                           ¦             ¦
                           ¦             ¦
                           ¦             ¦
                           ¦             ¦
                           ¦             ¦
                           ¦             ¦
                           ¦26        13¦
                           ¦_____________¦
```

WIRING FOR IMP CONNECTOR 3-25-81 (ALIAS ARPANET)

26 PIN ANSLEY CONNECTOR:	SLOT A19
PIN:	PIN:

1 {WHITE}
14 {BLUE}
	C2A20 {WHITE}
	C2B20 {BLUE}

2 {WHITE}
15 {BLUE}
	C2A19 {WHITE}
	C2B19 {BLUE}

3 {WHITE}
16 {BLUE}
	C2A18 {WHITE}
	C2B18 {BLUE}

4 {WHITE}
17 {BLUE}
	C2A17 {WHITE}
	C2B17 {BLUE}

6----¦ JUMPER TOGETHER 19---¦	

7 {WHITE}
20 {BLUE}
	C2A35 {WHITE}
	C2A36 {BLUE}

9 {WHITE}
22 {BLUE}
	C2A22 {WHITE}
	C2B22 {BLUE}

10 {WHITE}
23 {BLUE}
	C2A23 {WHITE}
	C2B23 {BLUE}

11 {WHITE}
24 {BLUE}
	C2A25 {WHITE}
	C2B25 {BLUE}

```
12 {WHITE}                     C2A24 {WHITE}
[illegible] {BLUE}                      C2B24 {BLUE}

13 {WHITE}                     TO NEAREST GROUND {WHITE}
26 {BLUE}                      TO NEAREST GROUND {BLUE}
```

```
DOCUMENT FOR THE WIRING OF THE "CC" CONNS ON THE F4 I/O PANEL

NO BUSS ON LEFT COLUMN OF I/O HEADERS

                      ANSLEY 26 PIN CONNECTOR

                         ______________
                         ¦14         1 ¦
                         ¦             ¦
                         ¦             ¦
                         ¦             ¦
                         ¦             ¦
                         ¦             ¦
                         ¦             ¦
                         ¦             ¦
                         ¦             ¦
                         ¦             ¦
                         ¦26         13¦
                         ¦_____________¦

FROM ANSLEY, PIN NO.:            TO "A" SECTION PIN NO.:


  I/O HEADER PIN
        1               TO              7 OF I/O HEADER AND THEN TO B.P. GND
CCO     2                               A1C3A30 ¦
        3                               A1C3A29  ¦
A1C3A31 JUMPER TO A1C3A32
A1C3A33 TO A1C3A34 TO A1C3A35

  I/O HEADER PIN
        1               TO              7 OF I/O HEADER AND THEN TO B.P. GND
        2                               A01C3B29
CC1     3                               A01C3B30
        8                               A01C3B34
        20                              A01C3B33
        22                              A01C3B35
A01C3B31 JUMPER TO A01C3B32

  I/O HEADER PIN
CC2
        1               TO              7 OF I/O HEADER AND THEN TO B.P. GND
        2               TO              A1C3A22
        3               TO              A1C3A23
A1C3A24 JUMPER TO A1C3A25
A1C3A26 TO A1C3A27 TO A1C3A28

  I/O HEADER PIN
CC3
        1               TO              7 OF I/O HEADER AND THEN TO B.P. GND
        2               TO              A1C3B22
        3               TO              A1C3B23
A1C3B24 JUMPER TO A1C3B25
A1C3B26 TO A1C3B27 TO A1C3B28
```

DOCUMENT FOR THE WIRING OF THE "DISK DATA" CONNS ON THE F4 I/O PANEL

ANSLEY 26 PIN CONNECTOR

```
 ____________
:14        1 :
:            :
:            :
:            :
:            :
:            :
:            :
:            :
:            :
:26       13 :
:____________:
```

DISK DRIVE CONNECTORS

READ/WRITE BACKPANEL CABLE WIRING:

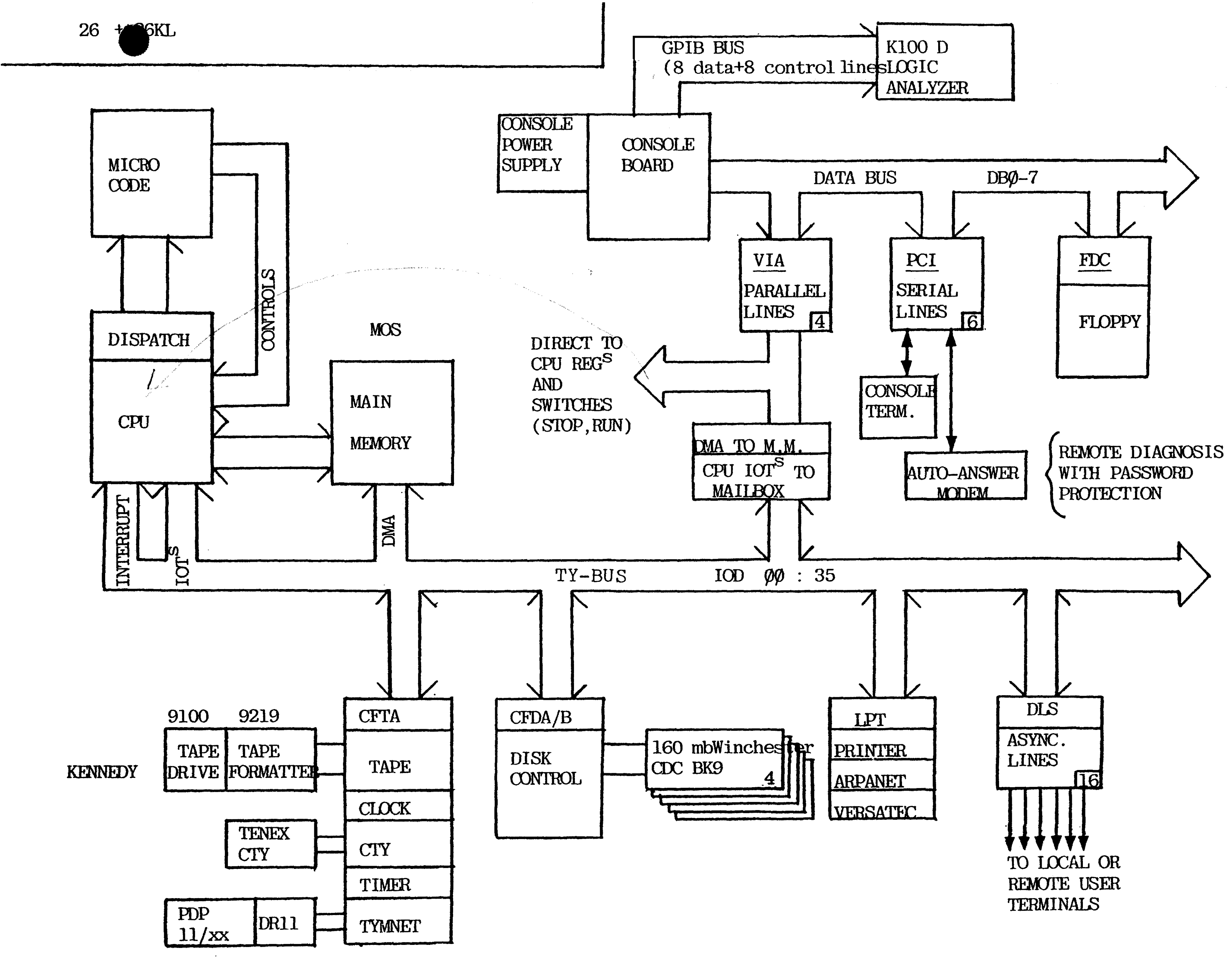
ALL DISK SIGNALS ARE TWISTED PAIRS.

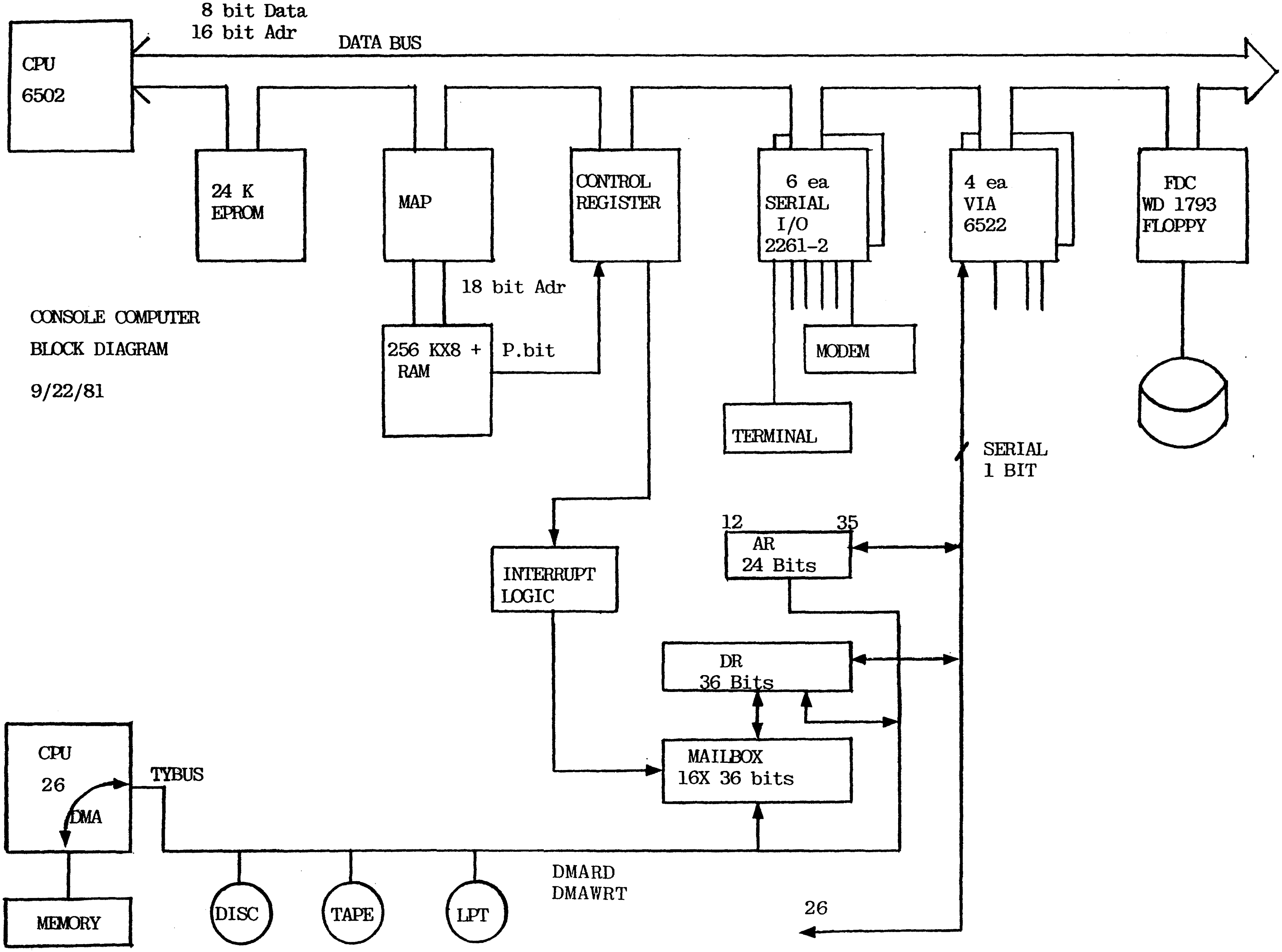
USE 26 PIN CONNECTOR IN I/O SECTION OF BACKPLANE.

		CONNECTORS:
	26 PIN ANSLEY CONNECTOR:	I/O 8:
SIGNAL NAME:	PIN #:	DD 0:
SERVO CLOCK	2 {WHITE	A7C2A30 {WHITE
SERVO CLOCK	14 {BLUE}	A7C2A29 {BLUE}
READ DATA -	3 {WHITE}	A7C2A34 {WHITE}
READ DATA +	16 {BLUE}	A7C2A33 {BLUE}
READ CLOCK	5 {WHITE}	A7C2A32 {WHITE}
READ CLOCK	17 {BLUE}	A7C2A31 {BLUE}
WRITE CLOCK	6 {WHITE}	A7C2B34 {WHITE}
WRITE CLOCK	19 {BLUE}	A7C2B33 {BLUE}
WRITE DATA	8 {WHITE}	A7C2C28 {WHITE}
WRITE DATA	20 {BLUE}	A7C2C27 {BLUE}
SELECTED -	22 {WHITE}	A7C2B35 {WHITE}
SELECTED +	9 {BLUE}	A7C2A35 {BLUE}
SEEK END	10 {WHITE}	A7C2B28 {WHITE}
SEEK END	23 {BLUE}	A7C2B27 {BLUE}

```
  I/O HEADER PIN
CC4
        1               TO              7 OF I/O HEADER AND THEN TO B.P. GND
        2               TO              A1C3A15
        3               TO              A1C3A16
A1C3A17 JUMPER TO A1C3A18
A1C3A19 TO A1C3A20 TO A1C3A21

  I/O HEADER PIN
CC5
        1               TO              7 OF I/O HEADER AND THEN TO B.P. GND
        2               TO              A1C3B15
        3               TO              A1C3B16
A1C3B17 JUMPER TO A1C3B18
A1C3B19 TO A1C3B20 TO A1C3B21
```

26 +[illegible]6KL

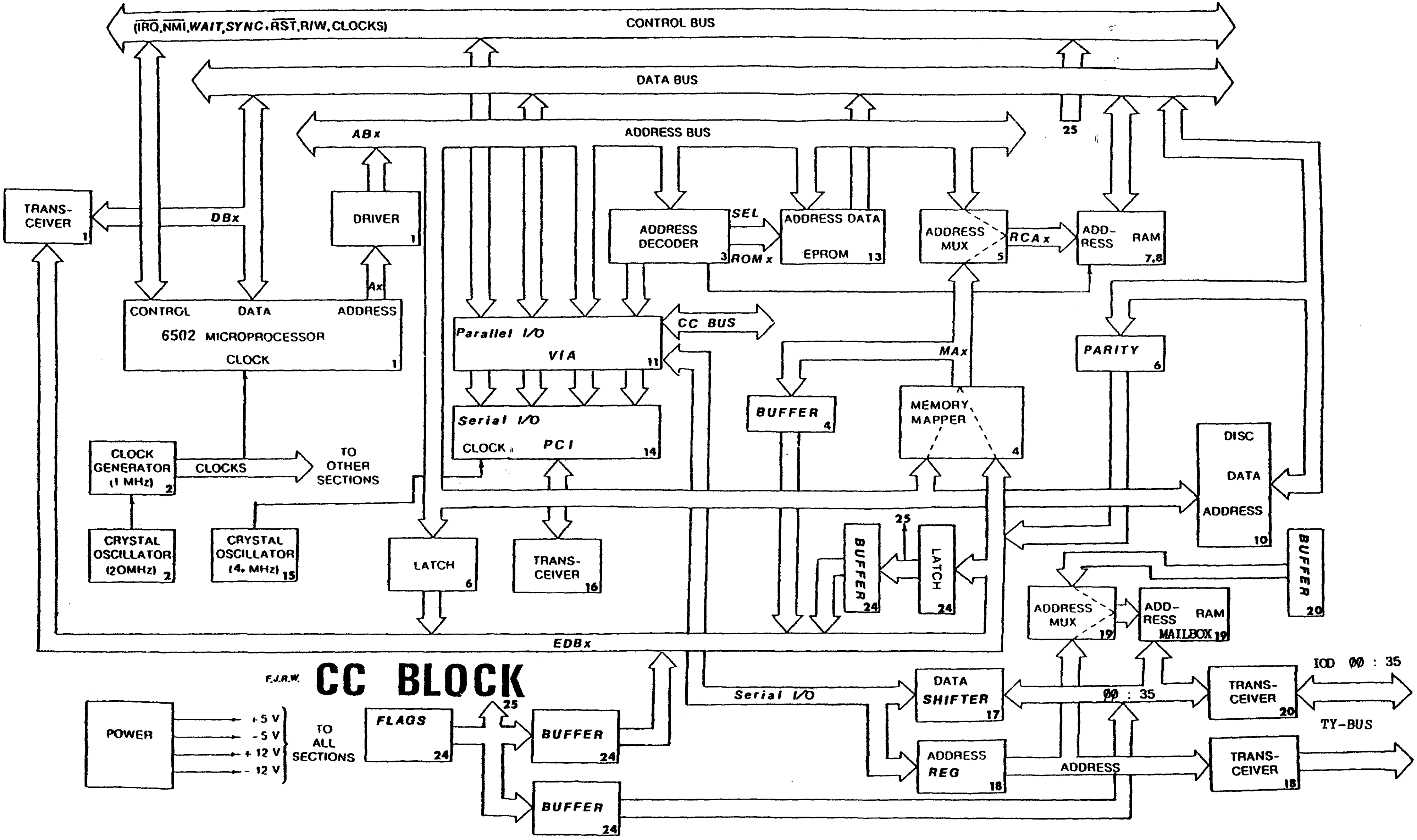

GPIB BUS
(8 data+8 control lines
K100 D
LOGIC
ANALYZER
CONSOLE
POWER
SUPPLY
CONSOLE
BOARD
DATA BUS
DBØ-7
MICRO
CODE
DISPATCH
CPU
CONTROLS
MOS
MAIN
MEMORY
DIRECT TO
CPU REG^S
AND
SWITCHES
(STOP,RUN)
VIA
PARALLEL
LINES
4
PCI
SERIAL
LINES
6
FDC
FLOPPY
CONSOLE
TERM.
DMA TO M.M.
CPU IOT^S TO
MAILBOX
AUTO-ANSWER
MODEM
REMOTE DIAGNOSIS
WITH PASSWORD
PROTECTION
INTERRUPT
IOT^S
DMA
TY-BUS
IOD ØØ : 35
9100
9219
KENNEDY
TAPE
DRIVE
TAPE
FORMATTER
CFTA
TAPE
CLOCK
CTY
TIMER
TYMNET
TENEX
CTY
PDP
11/xx
DR11
CFDA/B
DISK
CONTROL
160 mbWinchester
CDC BK9
4
LPT
PRINTER
ARPANET
VERSATEC
DLS
ASYNC.
LINES
16
TO LOCAL OR
REMOTE USER
TERMINALS



CONSOLE COMPUTER

BLOCK DIAGRAM

9/22/81


(IRQ,NMI,WAIT,SYNC,RST,R/W,CLOCKS)
CONTROL BUS
DATA BUS
ABx
ADDRESS BUS
25
TRANS-CEIVER 1
DBx
DRIVER 1
Ax
CONTROL
DATA
ADDRESS
6502 MICROPROCESSOR
CLOCK 1
ADDRESS DECODER 3
SEL
ROM x
ADDRESS DATA
EPROM 13
ADDRESS MUX 5
RCA x
ADD-RESS RAM 7,8
CC BUS
Parallel I/O
VIA 11
Serial I/O
CLOCK
PCI 14
BUFFER 4
MAx
MEMORY MAPPER 4
PARITY 6
DISC
DATA
ADDRESS 10
CLOCK GENERATOR (1 MHz) 2
CLOCKS
TO OTHER SECTIONS
CRYSTAL OSCILLATOR (20MHz) 2
CRYSTAL OSCILLATOR (4. MHz) 15
LATCH 6
TRANS-CEIVER 16
25
BUFFER 24
LATCH 24
BUFFER 20
ADDRESS MUX 19
ADD-RESS RAM MAILBOX 19
EDBx
F.J.R.W.
POWER
+5 V
-5 V
+12 V
-12 V
TO ALL SECTIONS
FLAGS 24
25
BUFFER 24
BUFFER 24
Serial I/O
DATA SHIFTER 17
ØØ : 35
ADDRESS REG 18
ADDRESS
TRANS-CEIVER 20
IOD ØØ : 35
TY-BUS
TRANS-CEIVER 18

CC BLOCK

Console Computer - Theory of Operation

References:

> 6500 Hardware Manual by Rockwell, Synertek or MOS Technology for the 6502 and 6522 chips.
>
> Signetics 2661 Enhanced Programmable Communications Interface Data Sheet.
>
> Western Digital FD179x and WD1691 Data Sheets and application notes.
>
> Other IC data books for LSTTL, RAM's, EPROM'S, etc.

In the following descriptions, cryptic symbols enclosed in brackets such as (CC2ADR) are cross-references to other drawings.

CC Address map

The 64-Kbyte address space of the 6502 is allocated as follows ($ means hex)

$0000-$DFFF	(56K)RAM, mapped in 4-Kbyte blocks
$E000-$E003	2661 PCI #0 - console terminal interface
$E010-$E013	PCI #1 - modem interface
$E020-$E023	PCI #2
$E030-$E033	PCI #3
$E040-$E043	PCI #4
$E050-$E053	PCI #5
$E060-$E06F	(16 bytes) unused I/O select
$E070-$E07F	(16 bytes) Map - loc $E07x maps virtual addresses $x000-$xFFF
$E080-$E08F	6522 VIA#0 - system control interface
$E090-$E09F	VIA #1 - 26parallel controls, 26 serial input
$EOAO-$EOAF	VIA #2 - ROM bank select, 26 serial output
$EOBO-$EOBF	VIA #3 - spare
$EOCO-$EOCF	(16 bytes) unused I/O select
$EODO-$EODF	(16 bytes) unused I/O select
$EOEO-$EOE3	FD1797 Floppy Disc Controller (not used currently)
$EOEB-$EOEF	Shadow of FD1797 Using these addresses makes the CPU hang until the FDC's Data Request line comes on
$EOFO-	Parity Error PC Latch, low byte
$EOF1	high byte
$EOF2	Parity Error Address Latch, low byte
$EOF3	high byte
$EOF4	(1 byte) unused I/O select
$EOF5	(1 byte) unused I/O select
$EOF6	TYBUS Interface Interrupt Flags
$EOF7	TYBUS Interface Interrupt Enables
$E800-$EFFF	(2K) ROM 1
$F000-$F7FF	(2K) ROM 2
$F800-$FFFF	(2K) ROM 3
$FFFA-$FFFB	NMI Vector (in ROM 3)
$FFFC-$FFFD	RESET Vector (in ROM 3)
$FFFE-$FFFF	IRQ Vector (in ROM 3)

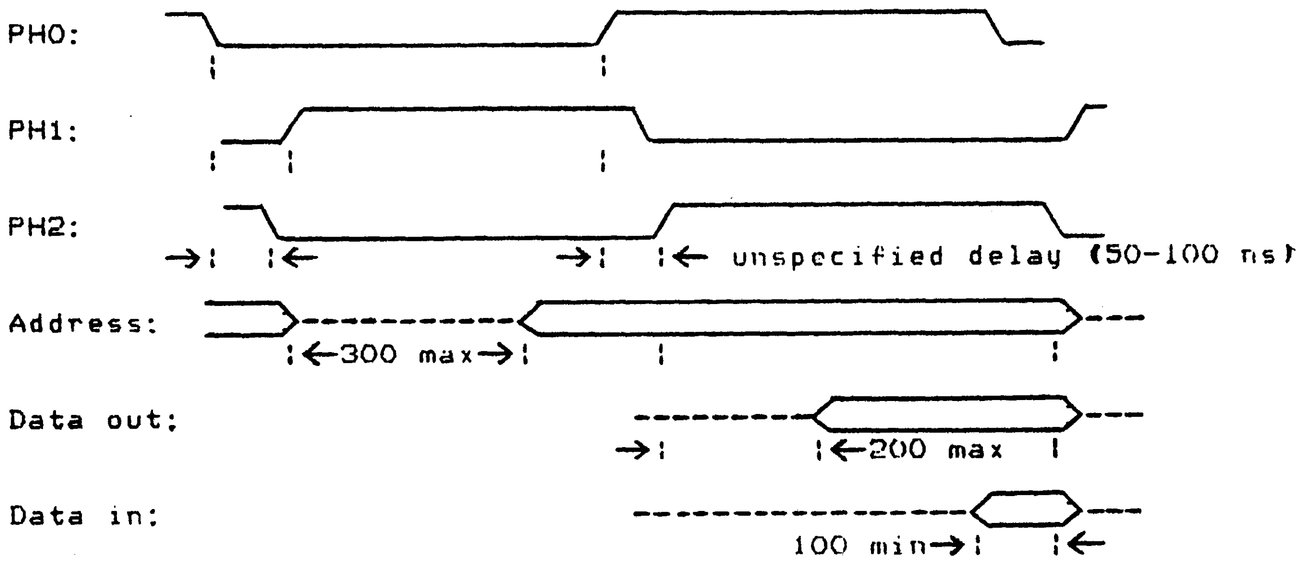
The I/O address space $E000-$E7FF is not fully decoded. The devices listed above appear in various "shadow" locations in that address range.

Clock Generator (drawings CCOCPU and CC1CLK)

The system timing comes from a 20.000 MHz crystal oscillator at 68-46. The 74S163 binary counter at 66-58 counts up on each positive edge of the 20MHz clock. When the count reaches 12 the output of the LS00 goes low, enabling the counter's LOAD input. On the next clock edge, instead of counting up, the counter loads with the value 3 wired into its D inputs. Thus the count goes through ten states altogether, dividing the input clock frequency by 10. The states the counter goes through (3,4,...,11,12) produce a square wave with a 50% duty cycle at the high order output. The resulting 2MHz clock is used by the Floppy Disc controller and further divided by the LS74 to make the 1 MHz clock (PHO H) that drives the 6502 CPU chip. The counter's low bit produces the 10 MHz TYBUS clock.

The CPU takes in the PHO clock and puts out two non-over lapping clocks, PH1 and PH2, at the same frequency but with some unspecified delay from PHO. PH2 is the master clock for all devices on the data bus. The CPU puts out a new address during each PH2 low cycle in time for it to be strobed on the rising PH2 edge. The address stays valid throughout PH2 high. During a write cycle, the CPU's output data are valid no later than 200 ns after the falling edge. During a read cycle, the CPU expects data to be valid at least 100 ns before the PH2 falling edge.

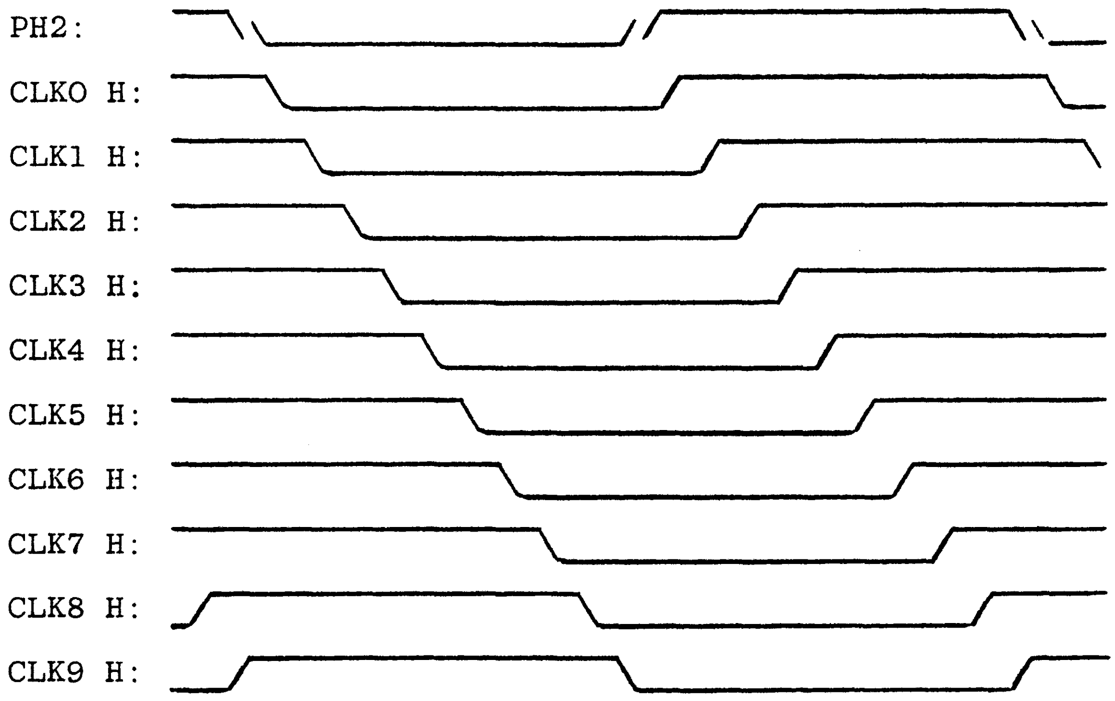
CPU CLOCK TIMING

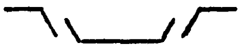


RAM timing (drawings CC1CLK, CC4RMA)

The PH2 clock from the CPU feeds the input of a 10-bit shift register formed by the LS174 at 64-47 and the LS175 at 66-47. This register clocks at 20 MHz, so its outputs are PH2 delayed by any multiple of 50 ns. The input can be selected by the 62-46 dipswitch to be PH2 delayed by two inverters in case the PH2 transitions fall too close to the 20MHz clock edge.

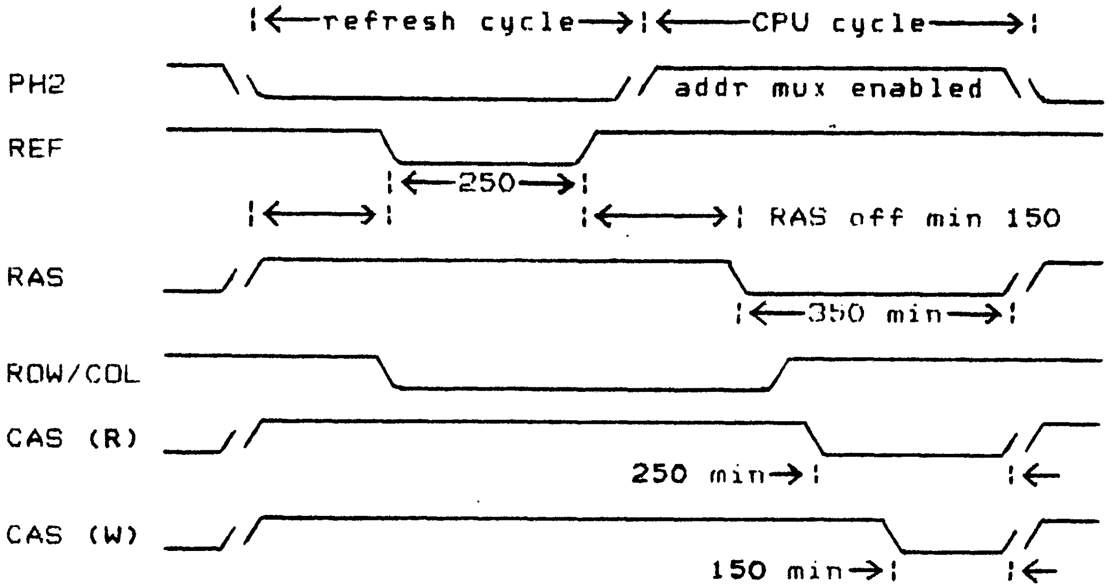
Delayed Clocks



Note:  symbolizes the uncertainty in the phase relationship between the 20 MHz clock and PH2.

The gates at the bottom of the drawing combine various of these delayed clocks to generate the RAM strobes. The two LS10's are enabled by the LS08 when RAM is selected and produce RAS and CAS during PH2 high. CAS happens early on a read cycle to allow time for slow RAM access, and it comes late on a write to give the cpu time to set up the data. During PH2 low the row/column mux is disabled and the refresh address buffer is enabled, and REF ENB does a RAS-only refresh cycle. The refresh counter increments at the end of the refresh cycle when PH2 goes high.

RAM timing



Data and Address Busses (drawings CC0CPU, CC2ADR)

The CPU data lines, DB0-DB7, run directly to all the MOS loads. The LS245 at 42-45 buffers the data bus (EDB0-EDB7 for Extended Data Bus) to drive the TTL loads.

The LS244's at 46-45 and 44-45 buffer the CPU address bus to all of its loads. The address lines are AB00-AB15, a 64-KByte address space.

The address space is divided in two parts: locations $0000-$DFFF are in RAM under control of the map, and $E000-$FFFF (the last 8 Kbytes) are wired to specific devices. The LS10 at the upper left detects this address boundary: when the high three address bits are all ones, it enables the upper left LS139 to decode the next two address bits; otherwise SEL RAM (H/L) becomes true. The outputs of the LS139 are

Address	$E000-$E7FF:	SEL I/O L
	$E800-$EFFF:	SEL ROM1 L
	$F000-$F7FF:	SEL ROM2 L
	$F800-$FFFF:	SEL ROM3 L.

The SEL ROM lines go directly to the ROM chip selects (drawing CC9ROM). SEL I/O L enables one of the LS138's at the left depending on the state of AB07. The LS138's further decode AB04-AB06, giving 16 different selects of 16 byte blocks for the I/O devices. The upper LS138 is gated by PH2 so its outputs are in effect data stobes; it selects the six 2661 PCI's and the memory map. The lower LS138 is not gated because the 6522 VIA's have PH2 as an input and contain their own strobe logic.

SEL MISC L enables the lower right LS138, which decodes AB00-AB02 to enable miscellaneous control registers that are implemented in MSI parts. This decoder is gated with PH2 to generate data strobes to the registers. The two gates feeding its high enable input allow it to generate only read strobes to addresses 0-6 which are read-only registers, and a select on read or write to address 7.

The LS139 at top right decodes two address lines from the memory map during CPU write cycles to drive write-enable for one of the four RAM banks. The LS155 in the center decodes the same address lines to drive RAS and CAS. During refresh cycles, the LS08's drive RAS to all four banks together.

Memory Map and RAM Addressing (drawings CC3MAP and CC4RMA)

The map consists of two 27S07 16x4 register files arranged as 16 words of 8 bits. When SEL I/O is true (low) the LS257 multiplexor gates AB00-AB03 (low four bits of the address) to the address inputs of the map. The LS10 at the bottom of the drawing generates a write-enable strobe to write data from EDB0-EDB7 into the map, and the LS00 above it enables the LS244 to drive the map data onto EDB for reading.

When SEL I/O is false (high) the multiplexor uses AB12-AB15 to address the map. This breaks the 64-Kbyte address space into 16 4-Kbyte pages. The highest two pages are wired to I/O and ROM; the other 14 correspond to RAM. The map translates the page number at its adress inputs into eight bits at its data outputs. The low six bits (MA12-MA17 for Mapped Address) are the number of a 4-Kbyte physical page in the .56-KByte RAM. MA18 is a spare, and the high-inhibits the RAS and CAS logic (CC1CLK).

If we consider the physical memory address to be MA17-MA12 from the map concatenated with AB11-AB00, the bits split up like this:

```
|<------ MA ------>|<--------------- AB -------------->|
17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
| C | R | bank |<---- column ---->|<------- row ---->|
```

Bits 15-14 select one bank of RAM chips. Bits 06-00 are the row address and bits 13-07 are the column address for 16 K chips. Bits 16 and 17 are the eigth address bit in case the board contains 64 K chips (this is a NC pin on 5v-only as long as the chips in each bank are all the same size. If a bank consists of 16K chips, its address bits 17-16 are "don't cares" and the same memory will show up in four places in the physical address space.

Memory Parity Logic (drawing CC5PAR)

The memory chips that hold data (DB0-DB7) have their inputs and outputs tied together to the data bus. The chips that hold the parity check bits have their data lines split, with data in connected to PAR WRT and data out to PAR RD.

During a CPU write cycle, the RAM array's PAR RD lines are open. The LS125 is enabled and normally drives PAR RD to zero. If the number of ones that the CPU is putting onto the data bus is even, the S280's EVEN output goes high and puts a one on PAR WRT, making the parity odd in the nine bits written into RAM. For diagnostic purpose, the program can set the LS125's input high (CCBVIA). Doing so will force bad parity to be written into memory.

During a read cycle, the LS125 is turned off and the addressed RAM chip drives the PAR RD line. The S280 checks parity on the nine bits read out and sets EVEN high if the parity was wrong. Even parity during a read enables the K input to the LS109 at upper right; if this was really a RAM cycle, CAS ENB will be low to the end of the cycle and its rising edge will clock the LS109 to the zero state, indicating that an error happened.

The LS374's near the center latch the contents of the CPU address bus during every instruction fetch cycle (SYNC high) as long as the error latch is off. Once an error has happened the clocks to this register are inhibited; the register retains the address of the first byte of the instruction that hit the bad data.

The LS374's at the right latch the contents of the address bus when the error latch comes on. This register then retains the address of the bad data. The error address register outputs are enabled onto the bus by the MISC decoder CC2ADR for reading by the CPU. Reading the high byte of the error address latch as the side effect of clearing the error latch, allowing subsequent errors to be recognized.

Reset, Parrallel I/O, Shift Registers (drawing CC8VIA and CC9CTL)

Reset Logic

During power-on, the R/C circuit at the lower left holds PO RST L low for a while. Grounding CC INTL L (C3B03) simulates a power-on reset. The LS08 at bottom generates CPU RST L, which resets the 6502 for either a power-on or a panic. PO RST L resets 6522 VIA #0 (top left), putting all of its outputs in high-Z state. That makes SYS RST H high and SYS RST L low, producing a RESET to the rest of the board. Since SYS RST H is a VIA output it is under control of the CPU. PANIC does not reset VIA0 so the CPU can tell the difference between PANIC and POWER-ON, and decide whether to reset the rest of the system.

Shift Registers (data path to/from 26 and TYBUS)

The shifter in VIA #2 is used to shift out, and VIA #1 to shift in. VIA #2 generates the shift clock. The clock and data are buffered out by the LS367 at 09-14, and four sections of LS367 at 05-14 produce delayed clocks for the 26. The LS125 at 42-01 buffers shift data in from the backplane; it is disabled when either of the on-board shifters (DMA address and data) is selected. Serial data in is latched in the LS74 at 09-26 a half cycle early by inverted SR CLK. This holds the serial input bit long enough for the VIA1 shifter to catch it.

VIA #0 - system control

This chip's interrupt request output is connected to NMI to provide a high-priority interrupt for error conditions. All other interrupts in the system are connected to IRQ. The I/O pins are used as follows:

Pin	Signal	Description
PA0	SR CTL0	Unused.
PA1	SR CTL1	Controls direction of VIA1 buffers (CC9CTL), enables input form.FUTURE 26.
PA2	SR CTL2	FUTURE 26.control output strobe
PA3	SR CTL3	FUTURE 26.data output strobe PA2 and PA3 are used ot control the programming voltage in the EPROM programmer.
PA4	CLR FLGS L	Low pulse clears the TYBUS interrupt flags.
PA5	REAL DMA H	Controls DMA/mailbox operation in TYBUS interface. Low for mailbox, high for DMA.
PA6	WRT RG H	DMA/mailbox control - low is read, high write
PA7 (in)	DM ACK	Cleared by DM GO L, set when DMA/mailbox cycle is finished.
CA1 (in)	PAR ERR L	Causes NMI when a parity error happens.

VIA #0 - system control continued

Pin	SIGNAL	Description
CA2	FDC IRQ H	Makes NMI to terminate Floppy data transfer loop when FDC requests an interupt.
PB0	SEL SR00 L	Low selects the TYBUS data register.
PB1	SEL SE01 L	Low selects the TYBUS address register.
PB2	PZ SENSE 2	Used only in EPROM programmer.
PB3	PZ SENSE 3	Ditto.
PB4	SR CLK	Used for single-clocking of 26 shifters.
PB5	WRT WRONG PAR H	For memory diagnostics.
PB6 (in)	SYNC H	Planned to use for CDDT one-stepping, turned out not to work. Timer 2 used instead
PB7	SYS RST H	Set high-Z by power-on, holds all other I/O reset.
CB1	BRK INT H	Causes NMI when a BREAK is detected in PCI0 or PCI1 - for calling CDDT.
CB2	NC	Spare.

VIA #1 - 26 Control

The PA and PB lines form a 16-bit I/O port for data transfers to and form the FUTURE 26. These are all outputs when controlling a 26, with alternate names shown at the right side of the LS245's in (CC9CTL). These 16 bits are also used as address and control by the EPROM programmer.

CA2	DM GO L	Pulsed low to initiate a DMA/mailbox cycle in the TYBUS interface.

VIA #2

The EPROMS each occupy a 2-Kbyte address space. PA0 and PA1 supply high-order address bits when larger EPROMs are places in the board, allowing bank switching in the ROMs.

PB0-PB7 are used as the data port for the EPROM programmer.

VIA #3 is entirely a spare. It can be used to control two Centronics - type line printers, or whatever.

EPROMs (drawing CC9ROM)

There are three 28 pin sockets which can contain 2716 (2-Kbyte 24-pin), 2732 (4-Kbyte 24-pin) or 2764 (8-Kbyte 28-pin) EPROMs. Pin 23 of each socket is wired to a pair of switches at 62-46. When a 2716 is inserted, the corresponding switch must be set to the Vcc position. For larger EPROMs, the switch should be set to the All position to allow the high address bit to select the different EPROM banks. Address bit 12 is not connected when a 24-pin package is in the socket, so it needs no switching.

RAM Array (drawings CC6RMO and CC6RM2)

The array is organized as four rows of nine chips. When viewing the board from the component side with the lettering right-side up, the near row is memory bank 0, the next is bank 1, etc. Bit 0 (least significant) is at the left, bit 1 is next, and so on with the parity bit at the extreme right. Note that the data bits are wired with input and output pins in common and the parity bit has separate in and out busses. Each row has its own RAS,CAS and WE line in order to keep the loading on the drivers down.

Floppy Disk Controller (drawings CC7FDA and CC7FDC) (FUTURE USE)

The two gates at the upper right of (CC7FDA) generate the read and write strobes to the 1797 during PH2 high. The LS74 to the left synchronizes the data request (FDC DRQ H) with the end of each CPU cycle. With DRQ off and address bit 3 on and no interrupt request (FDC IRQ H), the 7420 in the middle pulls WAIT L down when the CPU addressed the FDC. If the cycle is a read, the CPU will hang until WAIT L goes high. That happens when either DRQ goes high meaning that the FDC is ready to transfer a byte, or FDC IRQ H goes high meaning that the data transfer is finished.

The 7406's buffer the FDC outputs onto the disc drive cable. The 7414's and their input pullups receive the inputs from the disc, as recommended by the Shugart 851 interfacing manual.

The 74123 triggers ehen the FDC sets the head load command FD LOAD. About 30 ms later it times out and FD LOADED becomes true. This gives the right delay for loading the heads on a Shugart 851 drive.

(CC7FDC) shows the floppy controller chip as a 1793. We are really using a 1797. The two chips are identical except for the function of pin 25, which is Side Select on a 1797 and something else on a 1793. The entire circuit consisting of the 1797, 1691, 74S124 and all the resistors and capacitors is copied directly from the Western Digital WD1691 data sheet, so look there for the explanation of how it works.

Serial Line Interfaces (or PCI's) (drawings CCAPCI, CCBBRC, CCCEIA)

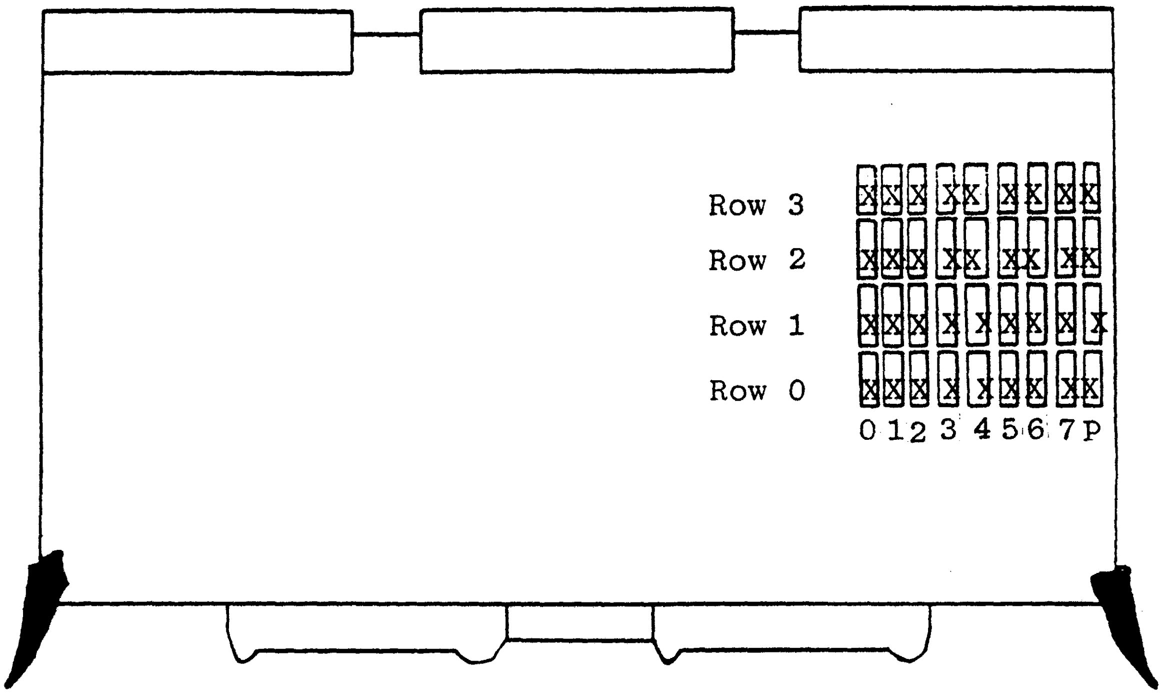
(CCBBRC) shows a standard type of oscillator using a 4.9152 MHz crystal. The output, BRCLK, is bussed to the clock inputs of the six PCI's. Each PCI contains dividers and control logic to generate any of 16 standard baud rates under program control. The lone pullup resistor at the upper left of the drawing us wired to the second input of each two-input 1488EIA driver to enable its output.

(CCAPCI) shows the six 2661's. All of their data and modem control signals are wired directly to 1488 drivers or 1489 receivers on (CCCEIA), except for the DSR (Data Set Ready) line of PCI #0. The LS08 at upper left is for a special hack on line 0 to allow the program to read the terminal's bit rate. With DTR high (false) the DSR input sees the RXD (Received Data) signal allowing the CPU to watch the data line and time its pulse widths. When DTR is set low (true), DSR is held true and the CPU no longer gets interrupts on data transitions.

Lines 0 and 1 have their RXC/BKDET pins programmed (by the setup routine in CDDT) to put out the Break Detect condition. The LS32 at middle left OR's the two break detects for input to VIA#0 (CCBVIA), where the rising edge of that input causes a high-priority interrupt request to the CPU. This mechanism lets the BREAK key on either controlling terminal call CDDT.

The transmit and receive clock inputs of PCI #2-#5 are open at present. These may be bussed to a common clock derived from BRCLK if we want to implement split speed on these lines.

Each PCI has three interrupt-request output pins. These are all wire-or connected to IRQ for a normal-priority interrupt to the CPU.



X Indicated a memory IC TYPE 4164

WHEN THE CONSOLE COMPUTER CRASHES WITH A

MEM PAR ERR Y1 Y2 Y3 Y4 Y5 Y6

(Y1-Y6 are bytes in hex)

this DATA can be INTERPRETED AS FOLLOWS:

PC = Y1 Y2 (2 bytes Long)

ADDR = Y3 Y4 (2 bytes Long)

DATA = Y5 (1 byte Long)

PAGE = Y6 (1 byte Long)

PAGE REFERS TO WHICH ROW HAD THE FAILURE.

AT THIS POINT YOU ARE IN A DDT AND CAN CHECK THE ADDRESS AND THE DATA.

```
    Y3 Y4/ (data)  new input data)    (there is no space
                                       between the numbers Y3
                                       and Y4)
                                      (new input data is the data
                                       you are putting in to test)
```

IF NO BIT IS A SOLID ERROR REPLACE OUT THE ENTIRE ROW.

CONSOLE OPERATIONS

*** POWER UP ***

At power up the console will do a self test and then go into "CDDT". Then you may issue the following console commands. If the console terminal is hung, you may hit local reset on the console keyboard.

```
              ;;CCL = Console Computer Language

    $G        ;;LOADS BASIC CCL CODE FROM CONSOLE EPROMS.
                        (INTERPRETER)
    LOADU     ;;LOADS DISK HANDLING PORTION OF THE MICRO-
              ;;CODE FROM CONSOLE EPROMS

    BOOTC     ;;LOADS CCL CODE FROM DISK TO MOS MEMORY

    TBOOTC    ;;LOADS CCL CODE FROM TAPE TO MOS MEMORY

    LOADC     ;;LOADS CCL CODE FROM MOS MEMORY TO CONSOLE

    ***** SOMETIMES BOOTU DOES NOT WORK PROPERLY ***
    ****** AND TYPING "BLAST" HELPS (CPU RESET) ****

    BOOTU     ;;LOADS REAL MICROCODE FROM DISK

    ** THE SYSTEM SHOULD NOW BE READY TO**
    ** LOAD THE MONITOR OR DIAGNOSTICS**

    *** LOADING THE MONITOR OR DIAGNOSTICS ***

    MBOOT 0   ;;LOADS DIAGNOSTICS FROM FILE #0 ON THE
              ;;TAPE DRIVE. OTHER FILE NUMBERS POSSIBLE

    DBEDDT 0  ;;LOADS MONITOR WITH EDDT FROM DISK DRIVE 0.
              ;;OTHER DISK NUMBERS MAY BE USED IF THERE IS
              ;;RESIDENT MONITOR ON THAT DISK.

          *** AT THIS POINT THE "EDDT" PROMPT SHOULD ***
          * APPEAR ON THE OPERATOR TERMINAL. THE COMMAND*
          * TO START THE MONITOR WOULD BE "START$G" AND *
          * WOULD BE TYPED ON THE OPERATOR TERMINAL. ***
```

NOTES: 1. THE "$" STANDS FOR ESCAPE OR ALTMODE.

***** 2. THE OCTAL CODES THAT APPEAR AFTER THE EXECUTION OF BRING-UP COMMANDS ARE CALLED "AR FLAGS". THESE FLAGS ARE STORED IN THE AR REGISTER AFTER ANY MACHINE HALTS. THEY ARE DECODED ELSEWHERE IN THIS DOCUMENT.

CONSOLE MAINTENANCE COMMANDS

PREFACE: DO NOT TOUCH THE CONSOLE WHILE THE MONITOR IS RUNNING!!!!! THE SYSTEM WILL PROBABLY CRASH!!!!!

THE MICROCODE MUST BE STOPPED BEFORE ANY CONSOLE COMMANDS ARE USED.

```
*  AR FLAGS  *
*
*01=RESET DONE
*11=JRST 4
*22=?
*33=?
*44=INTERRUPT
*   FROM ILLEGAL
*   DEVICE
*55=?
*66=ECC ERROR
*70=MONITOR
*   BOOTSTRAP
*   READ OK
*71=MONITOR
*   BOOTSTRAP
*   READ FAILED
*72=MICROCODE
*   READ OK
*73=MICROCODE
*   READ FAILED
```

Note:
New codes are being added frequently.

REGISTER DEPOSIT

```
LDHOLD (DATA)      ;;LOAD HOLD REG WITH 36 BIT WORD
LDPC   (DATA)      ;;LOAD PC
LDMA   (DATA)      ;;LOAD MA
LDIR   (DATA)      ;;LOAD IR
LDAR   (DATA)      ;;LOAD AR
LDQ    (DATA)      ;;LOAD ALU Q REG
LDDEV  (DATA)      ;;LOAD DEVICE REG
DAC 0 (DATA)       ;;LOAD AC 0 WITH 36 BIT WORD
DAM 0 (DATA)       ;;LOAD AMEM 0 WITH 39 BIT WORD
```

*** REGISTER EXAMINE ***

```
XAC 0              ;;EXAMINE ACCUMULATOR 0
XAM 0              ;;EXAMINE AMEM LOCATION 0
XPCF               ;;READ PC FLAGS
DD                 ;;DISPLAYS REGISTER CONTENTS ON UPPER
                   ;;PORTION OF CONSOLE SCREEN
```

CONSOLE MAINTENANCE COMMANDS

```
*** MOS MEMORY ***

DMRD (ADDR)          ;;EXAMINE MOS MEM LOCATION VIA CONSOLE
                     ;;DMA TYBUS PATH.
                      ;ALL MEM CAN BE ACCESSED.

DMWRT(ADDR) (DATA);;DEPOSIT 36 BIT DATA INTO MOS MEMORY WITH
                     ;;NO RESTRICTIONS

EX (ADDR)            ;;EXAMINE MOS MEMORY VIA CONSOLE DIRECT PATH.
                     ;;ONLY 256 K  GREATER THAN 256K RAPS AROUND

DE (ADDR)  (DATA) ;;DEPOSIT 36 BIT WORD TO MOS MEMORY.


MG (ADDR)            ;;START EXECUTING MOS MEMORY INSTRUCTIONS AT (ADDR)

*** MICROCODE ***

UST (ADDR)           ;;SET THE MICRO PC COUNTER ADDRESS AND SETUP UI.xx
                      in the CCL CODE
SETQ  UI.DEST 0
MML (ADDR) UI
UST (ADDR) TO VERIFY THAT IT DID LOAD
```

-----------------EXAMINE AND DEPOSIT BY FIELDS --------------

EXAMINE	*DEPOSIT*	MAX OCTAL VALUE*
UI.DEST	SETQ UI.DEST	77
UI.SP2	SETQ UI.SP2	1
UI.JCODE	SETQ UI.JCODE (DATA)	17
UI.ACSEL	SETQ UI.ACSEL (DATA)	7
UI.D	SETQ UI.D (DATA)	77
UI.CYLEN	SETQ UI.CYLEN (DATA)	17
UI.IF	SETQ UI.IF (DATA)	1
UI.DF	SETQ UI.DF (DATA)	1
UI.SPC	SETQ UI.SPC (DATA)	77
UI.JADR	SETQ UI.JADR.(DATA)	37777
UI.ROT	SETQ UI.ROT (DATA)	77
UI.MASK	SETQ UI.MASK (DATA)	77
UI.EEAL	SETQ UI.EEAL (DATA)	1
UI.EAFO	SETQ UI.EAFO (DATA)	1
UI.LDMA	SETQ UI.LDMA (DATA)	1
UI.IDSPR	SETQ UI.IDSPR (DATA)	1
UI.MWAIT	SETQ UI.MWAIT (DATA)	1
UI.SAFMA	SETQ UI.SAFMA (DATA)	1
UI.PO	SETQ UI.PO (DATA)	1
UI.SP1	SETQ UI.SP1 (DATA)	1
UI.ACRY	SETQ UI.ACRY (DATA)	1
UI.ASRC	SETQ UI.ASRC (DATA)	7
UI.AFUN	SETQ UI.AFUN (DATA)	7
UI.ADST	SETQ UI.ADST (DATA)	7
UI.ALU1	SETQ UL.ALU1 (DATA)	1
UI.LDAR	SETQ UI.LDAR (DATA)	1

CONSOLE MAINTENANCE COMMANDS (CONTINUED)

--------------------EXAMINE AND DEPOSIT BY FIELDS--------------

EXAMINE	*DEPOSIT*	* MAX OCTAL VALUE
UI.JCOND	SETQ UI.JCOND (DATA)	37
UI.REV	SETQ UI.REV (DATA)	1
UI.MAPF	SETQ UI.MAPF (DATA	17
UI.LIT	SETQ UI.LIT (DATA)	777777777777 (36 BITS)

ADM 36 SET UP

LINE MODE	LOCAL/REMOTE
TERMINAL MODE	ANSI
CHARACTER SET	US ASCII
KEYBOARD TYPE	Q WERTY
CHARACTERS/ROW	80
80/132 CLEARS SCREEN	ON
LINE FREQUENCY	60
FLAG	0
TRANSMIT BAUD RATE	19200
RECEIVE BAUD RATE	19200
BITS/CHARACTER	8
STOP BITS	1
PARITY	OFF
PARITY SENSE	ODD
PRINTER BUAD RATE	1200
PRINTER HANDSHAKE	XON/XOFF
PRINTER BITS/CHAR	8
PRINTER STOP BITS	1
PRINTER PARITY	OFF
PRINTER PARITY SENSE	ODD
PRINT EXTENT	PARTIAL
PRINT TERMINATOR	NONE
LOCAL ECHO	OFF
MODEM CONTROL MODE	OFF
SCROLL MODE	SMOOTH
BACKGROUND	DARK
MAGIN BELL	OFF
KEY CLICK	OFF
CURSOR MODE	BLOCK
AUTO REPEAT	ON
WRAP AROUND	ON
NEW LINE	OFF
AUTO XON/XOFF	OFF

Load Diagnostics from an initial POWER ON or cold starting the 26 system

1	$G	will load basic CCL Code from EPROMS on Console Interface Board
2	LOADU	Loads Microcode for Disk handling
3A	BOOTC	Loads CCL from Disk - Takes 15 min
3B	TBOOTC	Loads CCL from Tape
4	LOADC	Loads Console with CCL
5	BOOTU	Loads Code from Disk
6	MBOOT X	LOADS MACROS

NOTE: $G = ESC G

The system is ready now to load diagnostics. There are 2 types of diagnostics available at this point. To load monitor see <u>USER GUIDE</u>

PUT APPROPRIATE TAPE IN DRIVE

1. UTOBJ A CPU Diagnostic which checks the function operation of the bit slice boards, EOBUS,OBUS, DBUS, Memory, Traps, MUS's, etc.

2. TYFOON A system diagnostic which tests disk drives, tape drives & the system in general.

Procedure for warm start or reloading diagnostic on the 26 system

1. PRESS Function 1 Key- places terminal in CCL (PF1/F1 KEY)
   TYPE
2. LOADU Loads disk handling code from CC
3. BOOTU Loads disk handling code from CC
4. MBOOT Loads Macros

The system is ready now to load diagnostics. There are 2 types of diagnostics available at this point. To load monitor see USER GUIDE

PUT APPROPRIATE TAPE IN DRIVE

1. UTOBJ A CPU Diagnostic which checks the function operation of the bit slice boards, EOBUS,OBUS, DBUS, Memory, Traps, MUS's, etc.
2. TYFOON A system diagnostic which tests disk drives, tape drives & the system in general.

Load Diagnostics from an initial power on or cold starting the system.

1	$ G	will load basic CCL Code from EPROMS on Console Interface Board
2	LOADU	Loads Microcode for Disk handling
3A	BOOTC	Loads CCL from Disk - Takes 15 min-
3B	TBOOTC	Loads CCL from Tape
4	LOAD C	Loads Console with CCL
5	BOOTU	Loads Code from Disk
6	MBOOT X	

The system is ready now to load diagnostics. There are 2 types of diagnostics available at this point. To load monitor see USER GUIDE

1 UTOBJ — A CPU Diagnostic which checks the functional operation of the bit slice boards, EOBUS,OBUS, DBUS, Memory, Traps MUX's.

2 TYFOON — A system diagnostic which tests disk drives, tape drives & the system in general.

Procedure for warm start of 26KL

1.	PF1	Function -Function 1 Key- Put terminal in CCL
2.	LOADU	Loads disk handling code from CC
3.	BOOTU	Loads Code from Disk
4.	MBOOT 0	

System is now ready to load diagnostics or System Monitor

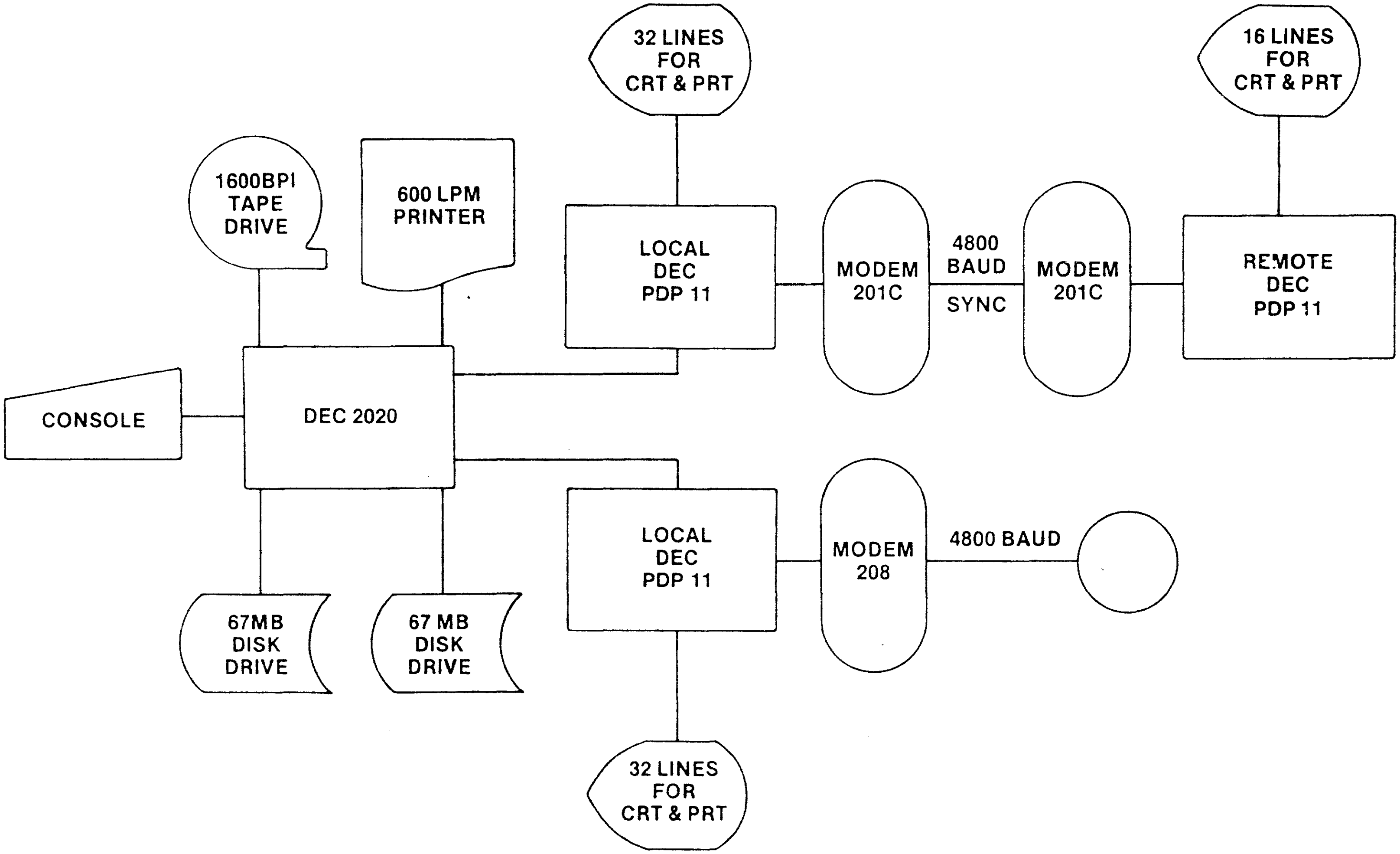
TROUBLE SHOOTING PROCEDURE FOR DOWN SYSTEM

The system must be down software wise before starting this procedure.

CPU Problem Use the Following

```
1.    Hit the (PF1) key          The screen should print
                                 CCL
                                 you are now connected to the console
                                 computer.

2.    Type:  TMM 15(CR)          This u-location has holes in the
                                 u-word, you will see garbage in
                                 the holes.
             TMM 4000(CR)
             TMM 10000(CR)
             TMM 17000(CR)
             TMM 30000(CR)       TMM tests micromemory.
                                 The output of the word should look like
                                 000...00   17...777
                                 177...77   00....000     These words
                                 0525...25   1252...52    are 88 bits
                                 1252...52   0525...25    wide.
                                 This will show you if all 6 banks
                                 of u-memory can be written into
                                 and read from.

3.    Type:  DMRD 100(CR)        This does a direct memory read at
                                 Loc. 100.
             DMWRT 100 VM1(CR)   This does a direct memory write of
                                 all 1's.
             DMRD 100(CR)        Data should equal 777777,,777777.
             DMWRT 100 0(CR)     Writes all 0's.
             DMRD 100(CR)        Data should equal 000000,,000000.

4.    Type:  EX 100(CR)          Data should equal all 0's.
                                 These commands will write a u-INST
                                 then execute the u-INST.
             DE 100 VM1(CR)      EX = EXAMINE   DE = DEPOSIT.
             EX 100(CR)          Data should equal all 1's.

5.    Load the u-diag for the System 26.
```

Typical Tymshare Hardware Configuration



Tymshare

FONNLY F3 MICRO COMPUTER

THE EQUIPMENT THAT MAKES UP A FOONLY COMPUTER SYSTEM IS INSTALLED IN 2 CABINETS AND CONSIST OF THE FOLLOWING.

ONE CABINET WITH

1- A NET COM COMMUNICATIONS INTERFACE

2- A FOONLY MICRO COMPUTER WITH 512 K OF 36 BIT WORDS OF MEMORY

ONE I/O CABINET WITH

1- WHICH HAS 3 CDC MODEL NUM. BZ9AX 160 MEGA BYTE DISK DRIVES
   (WITH NON REMOVABLE DISK PACKS)

2-. MODEL NUM. 9100 KENNDY TAPE DRIVE

   (75 IPS 9 TRACK 800/1600 BPI)

3- A MODEL NUM. 9219 FORMATTER USED IN CONJUNCTION WITH TAPE DRIVE.

INDEX

POWER SWITCHES FOR EMERGENCY AND NORMAL POWER DOWNS

THE FOONLY MICRO COMPUTER CABINET

1- HAS A 2 POLE CIRCUIT BREAKER THAT SHUTS OFF ALL AC POWER TO FONNLY CABINET, LOCATED ON THE BACK OF THE CABINET AT THE BOTTOM CENTER.

2- HAS A SINGLE SWITCH FOR DC POWER ONLY, LOCATED JUST ABOVE THE AC BREAKER.

3- THE NET COM INTERFACE HAS NO SWITCHES AND MUST BE UNPLUGGED

THE I/O CABINET

1- THE CDC DISK DRIVES HAVE A CIRCUIT BREAKER ON EACH DRIVE WHICH SERVES AS AN ON OFF SWITCH AND A CIRCUIT BREAKER. LOCATED AT LEFT REAR CORNER OF EACH DISK DRIVE.

2- THE KENNDY TAPE DRIVE HAS POWER OFF SWITCH ON TOP FRONT OF TAPE DRIVE

3- THE FORMATTER MUST BE UNPLUGGED FROM REAR OF CABINET.

FRONT CONSOLE

THE FONNLY MICRO COMPUTER HAS MANY SWITCHES ON THE FRONT CONSOLE, BUT FOR MOST OPERATION AND MAINTENANCE USES WE ARE ONLY CONCERNED WITH JUST A FEW OF THE SWITCHES SHOWN ON FIG 1 AND MENTIONED IN LOADING MICRO CODE ETC ON PAGE 4

THE SWITCHES WE ARE MOST CONCERNED WITH ARE

COSOLE
- START
- EXM AND DEP

MICRO PROCESSOR
- MI STOP
- MI CONT
- MI CLR
- MI PC

ADDRESS SWITCHES
- DEPENDS ON ADDRESS

THE SWITCHES WE USE THE MOST ARE DARKENED ON FIG 1

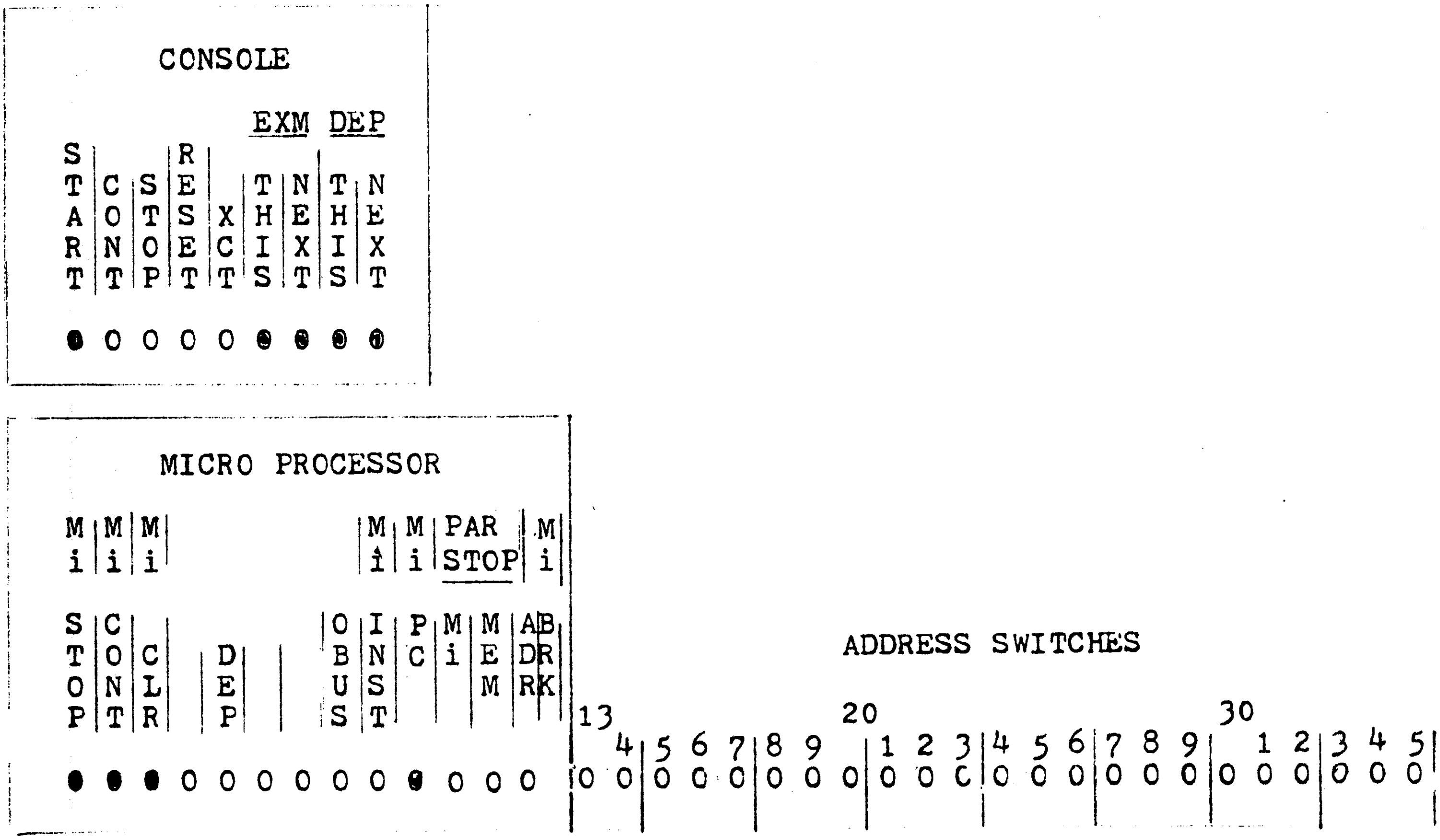


FIG 1

FRONT CONSOLE

THE FONNLY MICRO COMPUTER HAS MANY SWITCHES ON THE FRONT CONSOLE, BUT FOR MOST OPERATION AND MAINTENANCE USES WE ARE ONLY CONCERNED WITH JUST A FEW OF THE SWITCHES SHOWN ON FIG 1 AND MENTIONED IN LOADING MICRO CODE ETC ON PAGE 4
THE SWITCHES WE ARE MOST CONCERNED WITH ARE

COSOLE
- START
- EXM AND DEP

MICRO PROCESSOR
- MI STOP
- MI CONT
- MI CLR
- MI PC

ADDRESS SWITCHES
- DEPENDS ON ADDRESS

THE SWITCHES WE USE THE MOST ARE DARKENED ON FIG 1

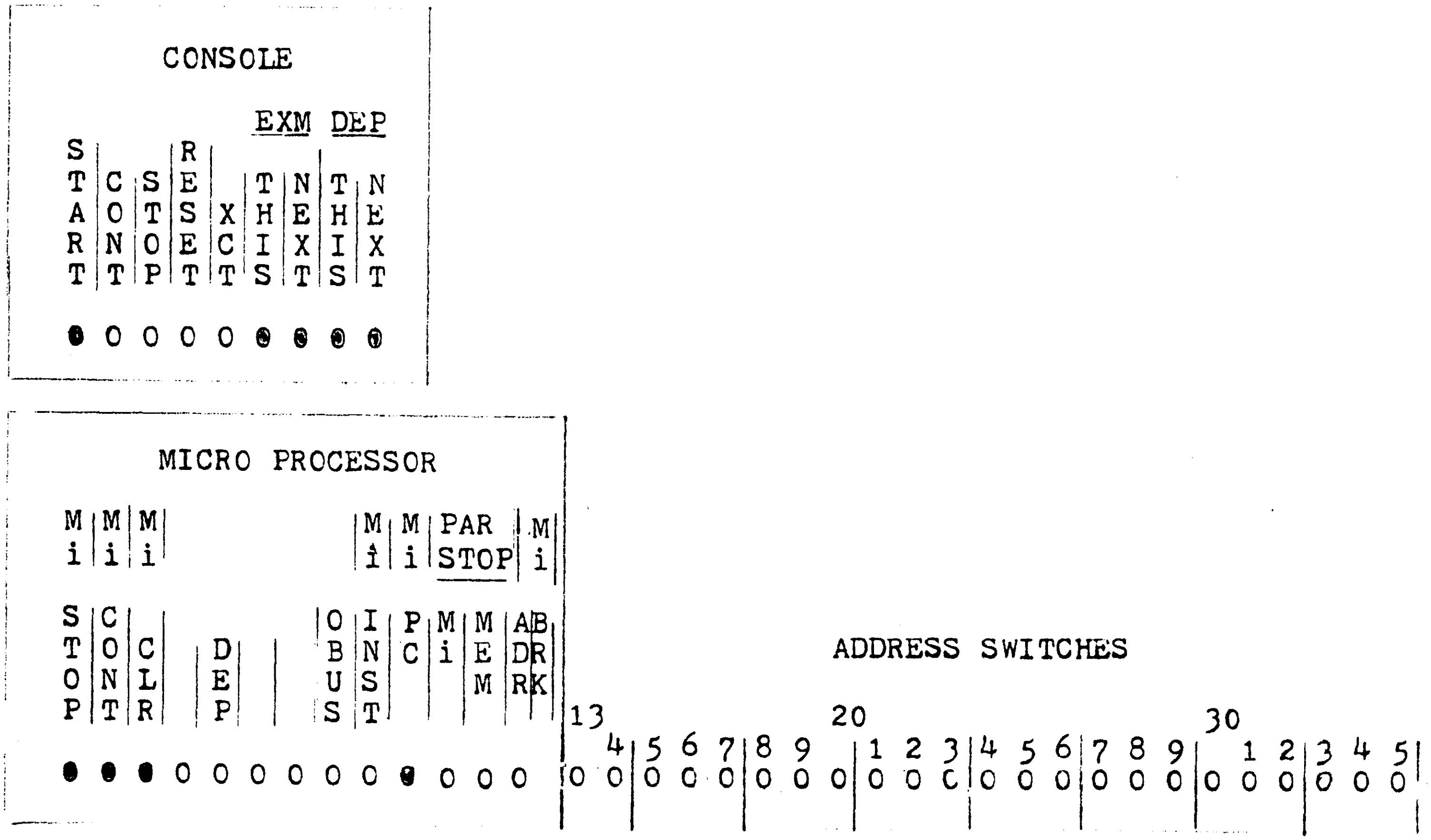


FIG 1

THE 9100 KENNDY TAPE DRIVE HAS ALL OF ITS SWITCHES
ON THE FRONT PANEL

THE LOAD POINT LIGHT IS UNDER THE SWING UP TOP PANEL.
THE PANEL MUST BE RAISED TO SEE LOAD POINT.

MAINTENANCE SWITCHES AND LAMPS ARE ALSO UNDER THE SWING UP TOP PANEL.

THERE IS A TAPE CLEANING PROCEDURE ON PAGE 9 OF THIS SECTION.

THE KENNDY FORMATTER IS JUST AND INTERFACE BETWEEN THE FOONLY CONTROLLER AND THE KENNDY TAPE DRIVE. THERE ARE NO LIGHTS OR SWITCHES TO BE CONCERNED WITH.

IN ORDER TO LOAD DISK DIAGNOSTICS OR A SYSTEM MONITOR YOU MUST LOAD A MICRO LOADER INTO THE SYSTEM.

(THIS WOULD ONLY BE NECESSARY IF THE SYSTEM HAD LOST POWER OR HAD A POWER GLITCH)

TO VERIFY MICRO CODE
SET MI STOP AND MI PC SWITCHES
SET ADDRESS SWITCHES TO 4000
PRESS MI CLR AND MI CONT
RESET MI STOP AND MI PC SWITCHES
PRESS MI CONT
YOU SHOULD NOW BE ABLE TO USE CONSOLE EXM AND DEP SWITCHES WHICH INDICATE THAT THE MICRO CODE IS OK.

LOADING MICRO CODE
INSTALL MICRO CODE TAPE
CHECK FOR CORRECT BPI 800/1600
SET ADDRESS SWITCHES TO 10
SET MI STOP AND MI PC SWITCHES
PRESS MI CLR AND MI CONT
RESET MI STOP AND MI PC SWITCHES
PRESS MI CONT
THE TAPE WILL ONLY MOVE A SHORT DISTANCE. YOU CAN THEN VERIFY IF THE MICRO LOADED USEING THE ABOVE PROCEDURE ARE GO AHEAD WITH LOADING OF NEXT TAPE MONITOR,DIGS ETC.

LOADING DISK DIAGNOSTICS OR A MONITOR TAPE
INSTALL CORRECT TAPE DIAG OR MONITOR
CHECK FOR CORRECT BPI 800/1600
SET ADDRESS SWITCHES TO 5000
SET MI STOP AND MI PC SWITCHES
PRESS MI CLR AND MI CONT
RESET MI STOP AND MI PC SWITCHES
PRESS MI CONT
TAPE SHOULD NOW BE LOADING
WHEN TAPE STOPS
SET ADDRESS SWITCHES TO 140 WHEN LOADING DIAGS
SET ADDRESS SWITCHES TO 100 WHEN LOADING MONITOR
YOU MUST NOW USE THE CONSOLE SWITCHES
PRESS CONSOLE START SWITCH 2 TIMES
THE TTY SHOULD NOW RESPOND AND BE WAITING FOR YOUR COMMANDS

SEE PAGE 2 FIG 1 FOR SWITCHS

SEE TENEX OPERATING PROCEDURES FOR MONITOR
SEE NEXT PAGE FOR BRIEF DISCRIPTION OF DISK DIAGNOSTICS.

- WARNING -

BECAUSE THE CDC DISK DRIVE DOESNT HAVE A REMOVABLE PACK
CARE MUST BE TAKEN WHEN RUNNIG DIAGNOSTICS ON A SYSTEM PACK.
AS OF NOW THERE IS ONLY 1 MAINTENANCE CYL WHICH WE CAN WRITE
ON. THAT IS CYL 1466 OCTAL 822 DECIMAL. IT WOULD BE A GOOD
IDEA NOT TO RUN ANY WRITE DIAGS UNLESS IT WAS REALLY NECESSARY
OR A BACK UP HAD JUST BEEN TAKEN.
TO BE SURE THAT WE DONT DO A WRITE ACCIDENTLY WE SHOULD KEEP
THE DISK IN WRITE PROTECT WHILE RUNNING DIAGNOSTICS.

WITH DIAGS LOADED AND TTY WAITING AT EDDT WE CAN NOW RUN DIAGS.

```
TYPE IN   CDC160$G
   UNIT/    X              A LINE FEED WILL GET YOU TO CYL,ETC
    CYL/    X              822. IS THE DECIMAL EQIV. OF 1466 OCTAL
   HEAD/    XX
 SECTOR/    X

   TYPE   RD$G  THE TTY WILL PRINT OUT SOME INFORMATION

   TYPE   RD$G  THIS WILL INITALIZE THE  DISK AND YOU COULD GET
                AN ERROR

   TYPE   RECAL$G   YOU CAN NOW PROCEDE WITH THE DIAGNOSTICS.

                    ( ALL 3 DISK DRIVES SHOULD BE INITALIZED
                      STARTING WITH 2 RD$G BEFORE RUNNIG DIAGS)
```

SOME OF THE DISK DIAGNOSTICS TO RUN ARE BELOW BUT DOES NOT INCLUDE
ALL OF THEM ARE GIVE YOU PARAMETERS FOR SOME OF THEM
TYPE HELP$G FOR MORE DETAILED INFORMATION.

```
RD$G
WA$G
TEST1$G
TEST2$G
TEST3$G
TEST4$G
TEST5$G
TEST6$G
```

SEE DIAGNOSTIC LISTING FOR DISCRIPTION OF ABOVE DIAGNOSTICS

THE DISK DRIVES ARE POWERED UP AND DOWN BY THE CIRCUIT BREAKERS ON THE BACK OF EACH DISK DRIVE.

WHEN POWERING THE DISKS UP CARE SHOULD BE TAKEN TO ALLOW SOME TIME FOR THE FIRST DISK TO POWER UP BEFORE POWERING UP THE NEXT DISK DRIVE. THE REASON BEING THAT THE CIRCUIT BREAKER ON THE MAIN PANEL MAY TRIP IF MORE THAN 1 DIAK DRIVE IS POWERED UP AT THE SAME TIME.

THERE ARE 3 LIGHTS ON FRONT OF DISK DRIVE

READY	FAULT CLEAR	WRITE PROTECT

READY IS JUST A YELLOW LIGHT
FAULT CLEAR IS BOTH A RED LAMP AND CLEAR SWITCH, WHEN A FAULT OCCURS YOU SHOULD BE ABLE TO CLEAR IT BY PRESSING SWITCH, IF YOU CANT CLEAR THE SWITCH THIS WAY, SOME TIMES YOU CAN POWER THE DISK DRIVE OFF AND THEN BACK ON.

WRITE POTECT IS A RED LAMP AND SWITCH TO PROTECT THE DRIVE FROM BEING WRITTEN ON. IF LAMP IS ON SWITCH IS SET ON.

UNDER THE FRONT PANEL IS A FAULT CLEAR DISPLAY LAMP USED TO RECORD DISK ERRORS THERE ARE 9 LOACTIONS THAT ARE RECORDED. THE 1ST 2 LAMPS RECORDE THE ERROR AND THE OTER 1 RECORDES THE AMOUNT OF ERRORS. THIS CAN BE CLEARED BY THE SWITCH. THIS WILL BE DESCUSSED IN FURTER DETAIL IN THE MAINTENANCE SECTION.