617-482-1100 x2781

NOVA 4 FAMILIARACATION

STEPPER DISPLAYS & CABLES HOOKS ONBOARD

ALU IN
ALU OUT
DISPLAY OCTAL MICRO WORD ADDR

ACCOWS STOPING AT ADDR OR INSERTING ADDR

MONDAY

TEXT INSTRUCTIONOS
OVERVIEW - DOCUMENTATION
- 16, 5 SLOT
- S/X/C DIFFERANCES
- CPU BLOCK
- ARCHITECTURE CAB

TUES
- REVIEW
- CPU OPS CBLOCK DETAIL
- SELF TEST
- VIRTUAL CONSOLE OPS
- VIRTUAL CAB

WED
- CPU PRINT TOUR
- MCORE BREAKDOWN
- DIAGNOSTICS USAGE

NOVA 4 CONT'D

(2)

THUR

MEMORY BLOCK
" PRINT TOUR
P/S BLOCK
" PRINT TOUR

DOCUMATION

INTF DESIGN 014-629-00
FRU4 NOVA 4 S/X 015-95-01
NOVA 4 PROGRAMMERS 014-617-00

CPU X
PRINTS 001/624 - 16-677

CPU C 001-1600-04

BBUMEM 001-1229-03 32-128 671-03
1-64 674-03

P/S 1523-04 16 670-03
1524-08 16-688-10
~~1585~~ -

CONSOLE 1585-00 661-00
WIRE LISTS 008-2599-01
3310 00

WIRING DIAG 001-1607

ASIDE 001-1639 -00

BSIDE 001-1640 -00

WIRE 1607 -01 /

B/P 001-1563-01

/

(5) 001-1616-03 006-861-03 ←

001-1619-01 06-696-01

1621 -00

NOVA INSTRUCT

FAIR ALC - 8

FAIRFAX MRI - 6 W/O AC VMS /VSR/ISZ/DSZ/ LDA/STA W/AC

FAIR I/O = 6

~~FAULT~~

BYTE -BITS I/O DEVICE CODE 01

LOAD BYTE STORE BYTE

LDB

ACS BYTE POINTER

ACD DEST FOR BYTE (8-15)

(0-7) SET TO ZEROES

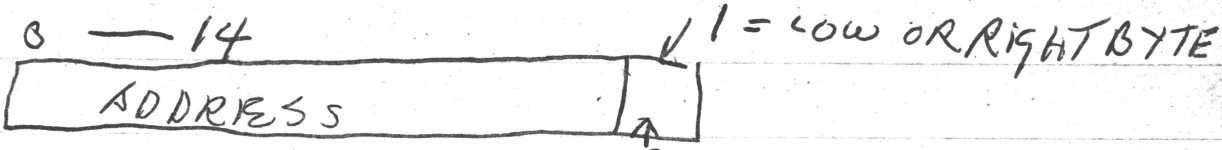
STB ACS BYTE POINTER

ACD ~~DEST~~ BITS (8-15) GO TO MEM UPPER

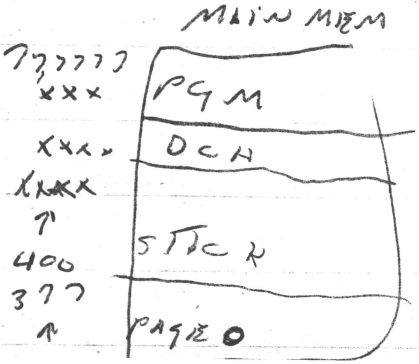
0-7 OR LOWER 8 BITS

~~0-7~~ 8-15 OTHER 8 BITS UNCHANGED

BYTE POINTER



STACK - PORTION OF MAIN MEMORY FOR TEMPORY STORRAGE OF DATA VARIABLES



STACK POINTER - 15 BIT REG
WHICH HOLDS THE ADDRESS
OF THE TOP OF THE STACK

INCREMENTS SP BEFORE PLACEING
WORD INTO MEMORY

FRAME POINTER. 15 BIT REGISTER WHICH
HOLDS AN ADDRESS TO REFERENCE
A FRAME INSIDE THE STACK

STACK OVERFLOW
(1) PUSH DATA ON STACK
(2) SP = 400₈ OR ANY MULTIPLE THEREOF
(3) ION - I

I DISABLES MAP
2 STORE PC+1 ⇒ LOC Ø
3 JUMP @ LOC 3

2044C

STANDARD
- CPU
- ASYNC I/O
- 32KW MEM
- AVC
- PF AUTO RESTART

OPTIONS
MUL/DIV
- 8KW 16KW 32KW
- R.T.C.
- B.B.U.

4S

CPU ASYNC I/O AVC AFF/R
- R.T.C.
- 4WAY INTERLEVED
- PREFETCH PROCESSOR

16KW OR 32KW SC
- B.B.U
- MUL/DIV
- FPU 16 SLOT ONLY

4X

CUP/ASYNC I/O AVC AFF/R
- RTC
- 4 WAY INTERLEVED
- PFP
- MMPU

- MUL DIV
- FPU - 16 SLOT ONLY
- B.B.U

(7)

FRU MANUAL SECTION 1

+

LAB = IDENTIFICATION OF MOUNTED FRU PARTS

BLOCK DIAG OF CPU

HANDOUT

+

CHAPTER 11 P990

2 MODES OF OPS.

~~CP~~ ~~CP~~ CPU IN RUN OR CONSOLE MODE

SAGE

STARTING ADDRESS GENERATOR FOR MICRO CODE

PRIORITY

1. DATA CHANNEL
2. INTERRUPT OR NMI FROM SYSTEM TERMINAL OR R.T.C.
3. PFP
4. INST

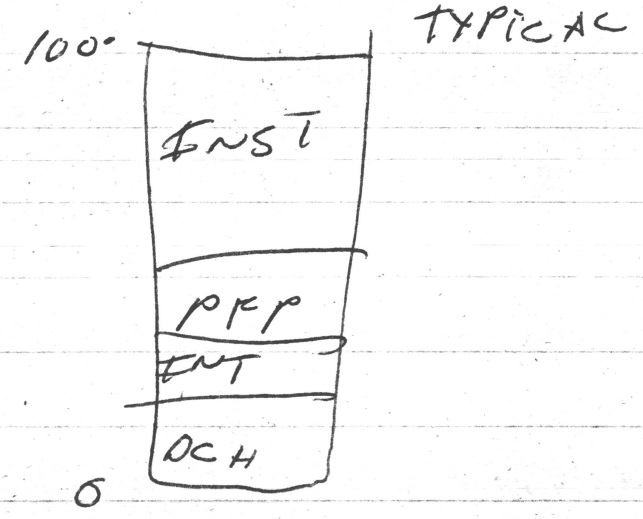
NOVA 4X CONT

~~CONTRO~~ CONTROL STORE

- 1000_{10} LOCATIONS
- 56 BITS LONG
- ROM (14 512 x 8 BIT)

-

μ INST = 200 NS



FOR EACH SAGE CYCLE CHECKS PRIORITIES

MOST OF MICRO CODE INST ARE SEQUENTIAL

MICRO - CODE

2 FUNCTIONS

① CONTROL DATA MANIPULATION UNIT & MEMOR

② DETERMINE NEXT INST TO BE EXECUTED

MICRO CODE NOVA 4 S/S

CONTROL STORIE

FIRMWARE

DATA MANIPULATION	NEXT STATE SELECT

2 MAIN FUNCTIONS

DATA MANIPULTION

0 1	2 3	4-6	7-11	12-16	17-19	20-22	23-25	26
MEM CTL	UNUSED	EXT SOURCE SELECT	A-REG	B-REG	ALU FUNCT	ALU SOURC	ALU DEST	CRY

00 - XMEM

01 REGENB - SYNC TO I/O CTRLS FOR INTERRPTS & DCH

10 - READ

11 WRITE

NEXT STATE SELECT

27-29	30-32	33-35	36-41	42	43-53	54-55
RANDOM	TRUE STATE CHANGE	FALSE STATE CHANGE	TEST CONDITION SELECT	?	LITERAL	UNUSED

000 } GREATER RANDOM FIXED
001
010
011

XRAND (NOOP)

CRF - CLEAR REG TO FETCH/PREP

FPMEM - FLOTPOINT MEM OPS

INC PC

REGISTER

0100 RAND (0)
0101 RAND (1)
0110 RAND (2)
0111 RAND (3)

TEST COND SELECT

TRUE SIGNAL JUMPS TO FALSE FIELD?

T-R STATE CHANGE

000 JUMP – LITERAL GOES TO NEXT MCODE ADDRESS

001 PUSH – LITERAL GOES TO NEXT MCODE ADDRES & CURRENT MCODE ADDRES TO MSTACK

010 DECODE USES IR → DECODE LOGIC FOR NEXT ADDRESS IN MCODE

011 SAGE GENERATE NEW SAGE FOR NO PHANTOM COULD BE DIFFERENT ALOC EACH TIME NEW MICRO CODE ADDR AT END OF INST OR CURRENT MICROINSTOPS

100 POP – MICRO STACK → NEXT MICRO ADDR.

101 CONT CONT AT MPC PLUS 1

LITERAL FIELD

I - MICRO VMP ADDR

~~II~~ - CONSTANT

III RANDOM FUNCTION

CONSTANT → LITERAL GOES TO ALU IN BUS USED IN CONSTANT # BITS

IN DISPOT USE RESET TO T/S... RESET GOES TO ADDR 377 IN MCODE THEN FOLLOWS SET SEQUENCE

VIRTUAL CONSOLE

100000 = PFP is OK

~~Dgct~~

CALL DSC1	1 EXT	SOURCE	2 A REG	2 B REG	4 ALU OPS	5 ALU DEST	6 MEM CONT

(12)

NOVA 4X

DQC2 RAND TEST TSC ESC LITERAL
0 UARTFE
1
D

P/N FOR S/X 077-102-03 NOVA 4 MCODE

STEPER

PQ10 D3 W2G ENAB MCODEADDR
PQ1 B6 W1G (TP3) ENAB ALU OUTREG LOWER
PQ1 D6 TP1 ENAB ALU OUTREG UPPER

200NSCLOCK { PQ3 B4 BUFFQCLKB (FOR MCODEADDR
PAQ1 B7 BUFFQCLKA (FOR ALU IN)

MEMORY

1 BOARD 4 MODULES

A
B
C
D

NOVA 4X MEM

FULLY POPULATED BOARDS
HAVE 2 BANKS

A < 0
    1
B < 1
    0
C < 0
    1
D < 0
    1

	INPUT →	OUTPUT REQ			
MOD	A	w	y	x	z
	B	x	z	y	w
	C	y	w	z	x
	D	z	x	w	y

ASSURES PROPER SEQUENCING OF MEMOUT

OE L

REFRESH
16K RAM – 12.8 μs REFRESH CYCLE STARTED
4K RAMS 25.6 μs
PRIORITY OVER ALL OTHER OPS
CYCLE TAKES 900 ns
RADR (0-6) ONLY ASSERTED DURING REFRESH
EUREKA 1.64 MS @ WOLFBOARD REFRESHED 128×128?

CRADR
RADR

NOVA 4D IAGNOSTICS

→ 06 3501

POWERUP
RESET 3777
RST 2266
3035
3036
3037
3040
3041
3042
43
44
45
46
2435 — 2261 —
2436
2443
44
2267
2220

2313
2314
2333
2334
2335
2314
2315
16
2271
2547
2550
2551
2552
3
4
5
6
3000
1

2561
2
3
4
5
67
74
75
76
77
2600
2272
2273
227[illegible]
CONSOL[illegible]
2277
2301
3001
3002
3006

2303
2313
2333
2335
2314
2315
15
2304
2305
3522
3523
3527
3525
2342
43
SAGE — 44

33
34
35
14
15
16
3540
3477
3500
3501
GETNEXT INST

DISCODE 3261
3536
3537
2306
2307
2311
12
13

RUN

2637

2644

3060

2435

2436

2443

44

3062 - 3070

3522

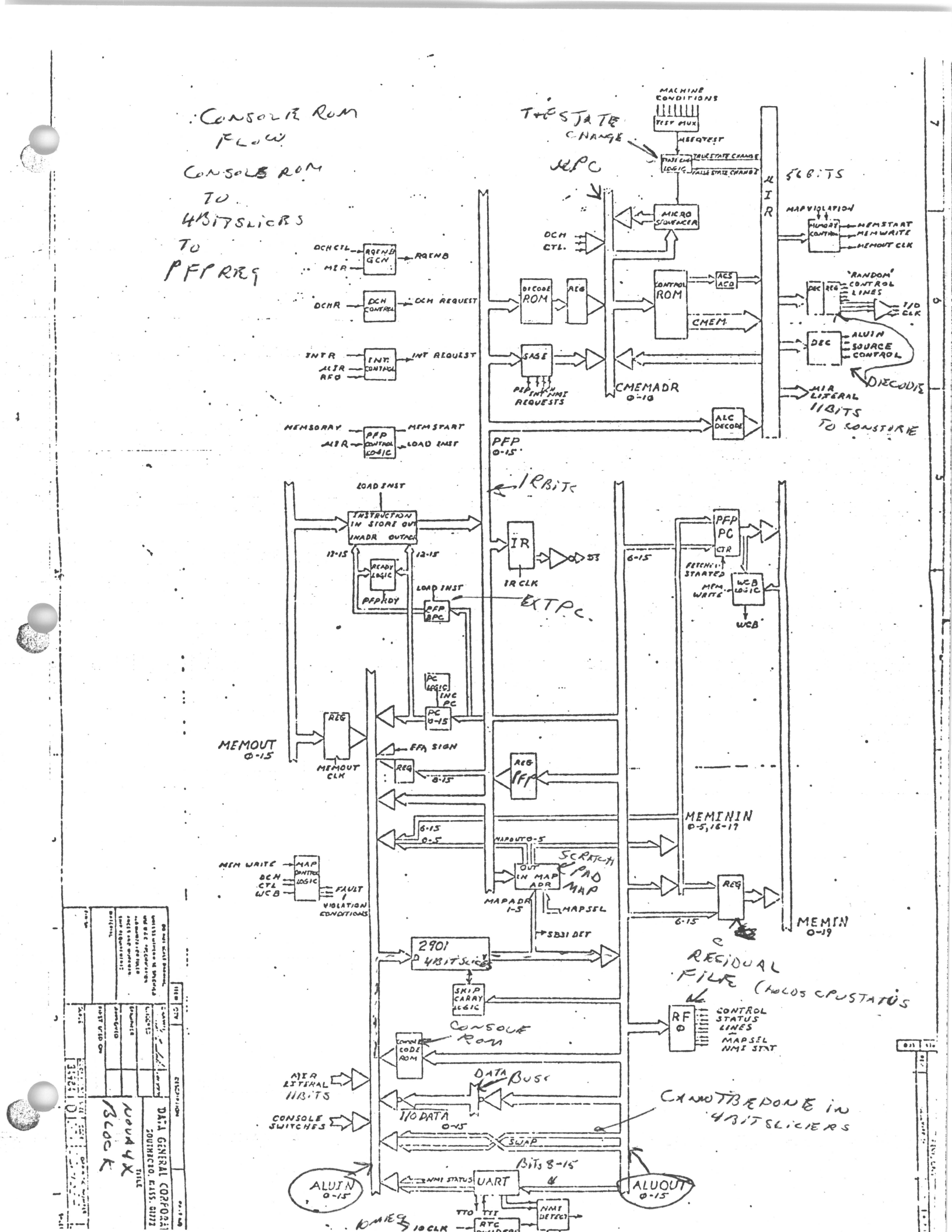
(17)

16 SLOT P/S

5 FUSES IN UNR PS 2 AC
3 DC

PS/ +5 IS MASTER VOLT ON PS OTHERS RELY ON IT

15V DEVELOPED BY +12 AND +4 OR 3V REG


CONSOLE ROM FLOW
CONSOLE ROM TO 4BITSLICES TO PFP REG
TRISTATE CHANGE
MACHINE CONDITIONS
TEST MUX
MREQTEST
TRUE STATE CHANGE
FALSE STATE CHANGE
MPC
MICRO SEQUENCER
MIR
56 BITS
MAP VIOLATION
MEMORY CONTROL
MEMSTART
MEMWRITE
MEMOUT CLK
DCH CTL.
DECODE ROM
REG
CONTROL ROM
ACS
ACD
CMEM
'RANDOM' CONTROL LINES
DEC
REG
I/O CLK
DEC
ALUIN
SOURCE
CONTROL
DECODE
SAGE
PFP INT
NMI REQUESTS
CMEMADR 0-10
MIR LITERAL 11 BITS TO CONSTANT
ALC DECODE
DCH CTL
MIR
REQ DCH
DCHR
DCH CONTROL
DCH REQUEST
INTR
MIR
REQ
INT CONTROL
INT REQUEST
MEMSTART
MEMBUSY
MIR
PFP CONTROL LOGIC
LOAD INST
PFP 0-15
16 BITS
LOAD INST
INSTRUCTION IN STORE OUT
INADR OUTADR
IR
IR CLK
READY LOGIC
PFPRDY
PFP MPC
EXTPC
PFP PC CTR
FETCH STARTED
MEM WRITE
WCB LOGIC
WCB
PC LOGIC
INC PC
PC 0-15
REG
MEMOUT 0-15
MEMOUT CLK
EFA SIGN
REG
0-15
REG FF
MEMININ 0-5, 16-17
6-15
0-5
MAPOUT 0-5
MEM WRITE
MAP CONTROL LOGIC
DCH CTL
WCB
FAULT / VIOLATION CONDITIONS
OUT IN MAP ADR
SCRATCH PAD MAP
MAPADR 1-5
MAPSEL
SBJ31 DET
2901 4 BIT SLICES
SKIP CARRY LOGIC
MEMIN 0-19
RESIDUAL FILE (HOLDS CPUSTATUS)
RF 0
CONTROL STATUS LINES
MAPSEL NMI STAT
CONSOLE ROM
CONSOLE CODE ROM
MIR EXTERNAL 11 BITS
DATA BUS
I/O DATA 0-15
CONSOLE SWITCHES
CANNOT BE DONE IN 4 BIT SLICERS
SWAP
BITS 8-15
ALUIN 0-15
NMI STATUS
UART
ALUOUT 0-15
TTO
TTI
NMI DETECT
RTC
I/O CLK
DATA GENERAL CORPORATION
SOUTHBORO, MASS. 01772
TITLE
NOVA 4X BLOCK

NOVA 4X MICROWORD

7 NOT USED

12 NOT USED

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	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55
MEM CONTROL		unused		EXT. SOURCE SELECT			A REGISTER SELECT					B REGISTER SELECT					ALU FUNCTION			ALU SOURCE			ALU DEST.			CRY	RANDOM			T STATE CHANGE			F STATE CHANGE			TEST COND.SEL.				?	LITERAL																

MEM CONTROL
- NOPE
- ENB ASSERT MOENB
- RD MEM READ REQUEST
- WTE MEM WRITE REQUEST

ALU FUNCTION

(0)	R+S	R plus S
(1)	S-R	S minus R
(2)	R-S	R minus S
(3)	R∨S	R or S
(4)	R∧S	R and S
(5)	R∧S̄	R and S
(6)	R⊻S	R xor S
(7)	R≡S	R xor S, R xor S.

ALU SOURCE (R,S)

R	S
(0) A reg	Q reg
(1) A reg	B reg
(2) 0	Q reg
(3) 0	B reg
(4) 0	A reg
(5) aluin (D)	A reg
(6) aluin (D)	Q reg *
(7) aluin (D)	0

* Q reg used for Mul/Div.

DIRECT?

RANDOM
GREATER)
- 0) XPAND - NOPE
- 1) CRF (clear to request fetch(PFP)
- 2) FPMEM
- 3) INCPC

LESSER)
- 4) RAND (0)
- 5) RAND (1)
- 6) RAND (2)
- 7) RAND (3)

(examine literal field for breakdown of rand 0,1,2,and 3)

* CMEM 46=0 ENABLE LESSER

** CMEM 47=1 DRIVE DATA BUS IN LESSER

CARRY
- (0) Cn=0
- (1) Cn=1

STATE CHANGE
- (0) JUMP LITERAL OF INST. --> NEXT ADDR.
- (1) PUSHJ PUSH μ ADDR. TO SEQ.μ STACK; LITERAL OF INST. --> NEXT ADDR.
- (2) DECODE - Decode which Macro to enter for given Inst.
- (3) SAGE - Generate SAGE A End of Current Macro
- (4) POPJ MICRO STACK --> NEXT ADDR.
- (5) CONT. CONTINUE AT μ PC PLUS 1

CONTROLS ALU IN BUS

EXT SRC SEL
- 0) DATA DATA BUS
- 1) MEMOUT MEMOUT BUS
- 2) EPC EXTERNAL PC
- 3) LIT LITERAL FIELD
- 4) SWAP BITE SWAP
- 5) EFA SHORT EFFECT ADDR.
- 6) PFP PFP BUS
- 7) XESRC NOPE

A&B REG SEL

(03)	ACS	IR(1-2)
(13)	ACD	IR(3-4)
(20)	AC0	
(21)	AC1	
(22)	AC2	
(23)	AC3	
(24)	AC4	SP (STACK PNTR)+FPPC (FPSR 17-31)
(25)	AC5	FP (FRAME PNTR)
(26)	AC6	IOF (I/O FLAGS)
(27)	AC7	MS (MAP STATUS)
(30)	AC8	PC,IPC (INTERNAL PC)
(31)	AC9	RD (RESIDUAL CONTROL)
(32)	AC10	ADR (LAST MEM ADDR.)
(33)	AC11	T0
(34)	AC12	T1
(35)	AC13	T2 micro
(36)	AC14	T3 temp
(37)	AC15	T4 storage

ALU DESTINATION

(0)	F→Q;F→Y	aluout to Q and aluout bus
(1)	F→Y	aluout to aluout bus
(2)	F→B;A→Y	aluout to B,A to aluout bus
(3)	F→B;F→Y	aluout to B,aluout to aluout bus
(4)	F→Y;F/2→B;Q/2→Q	Q shift right to Q; aluout shift right B; aluout to aluout bus
(5)	F→Y;F/2→B	aloout shift right to B; aluout to aluout bus
(6)	F→Y;F*2→B;Q*2→Q	Q shift left to Q; aluout shift left to B; aluout to aluout bus
(7)	F→Y;F*2→B	aluout shift left to B; aluout to aluout bus

Y = ALU BUS

F = FUNCTION BUS ON SCICER

TEST

(00)	DCHV.	DCHVIOL
(01)	PMX.	PMX
(02)	PFPRDY	PFP READY
(03)	MAPWCB.	MAP/WCB
(04)	SIGN	SIGN
(05)	PRSIGN	PRSIGN TEST
(06)		
(07)		
(10)	IODONE.	SYNCSELD
(11)	AUTOX.	AUTO TEST
(12)	ALCSKP.	ALCSKIP
(13)	ALUCOUT	ALU CARRY OUT
(14)	FAST	FAST DCH
(15)	INTR.	SYNCINT
(16)	EXC.	EXCEPTION
(17	IOBUSY.	SYNCSELB
(20)	ALU4	ALUOUT 4
(21)	ALU5	ALUOUT 5
(22)	ALU6	ALUOUT 6
(23)	ALU7	ALUOUT 7
(24)	ALU3	ALUOUT 3
(25)	ALU2	ALUOUT 2
(26)	ALU1	ALUOUT 1
(27)	ALU0	ALUOUT 0
(30)	ALU15	ALUOUT 15
(31)	ALU14	ALUOUT 14
(32)	ALU13	ALUOUT 13
(33)	ALU12	ALUOUT 12
(34)	ALU8	ALUOUT 8
(35)	ALU9	ALUOUT 9
(36)	ALU10	ALUOUT 10
(37)	ALU11	ALUOUT 11
(40)	UARTFE	UARTFE
(41(	FPTB.	FPTB
(42)		
(43)		
(44)	ALUZERO.	ALUZERO
(45)	FPB.	FPB
(46)	MEMOK	MEMOK
(47)	FPSKP.	FPSKP
	(INVERTED) look at inverted signal	
(50)	IR5.	IR5
(51)	CONSOLE.	CONSOLE MODE
(52)	DCHNOW	DCHR
(53)	FZERO	ALUZERO
(54)	DCHR.	SYNCDCH
(55)	FALSE	+5 VOLTS
(56)	CARRY	CARRY
(57)	ALUI0	ALUIN0

Remember when Testing Inverted Signals, True=false False=true

INTERRU
- MICRO JUMP ADD
- CONSTANT
- RANDOM OPERA
  - RAND (0)
    - (0) DCHO
    - (1) DCHON
    - (2) DCHAS
    - (3) CLRDC
    - (4) HOLD
    - (5) MASKO
    - (6) DCHI
    - (7) DCHA
  - RAND (1)
    - (0) TTISR
    - (1) CONRO
    - (2) RFDES
    - (3) RFADC
    - (4) RDMAP
    - (5) PCDES
    - (6) MULEN
    - (7) DIVEN
  - RAND (2)
    - (0) ENBIC
    - (1) TTIRE
    - (2) DATOB
    - (3) CLRPN
    - (4) TTODS
    - (5) CKPWF
    - (6) DATOC
    - (7) DATOA
  - RAND (3)
    - (0) SETCI
    - (1) CLRCI
    - (2) WTMAI
    - (3) FPCPU
    - (4) FPABO
    - (5)
    - (6) INHF
    - (7) PFPD

3677 NORAND

3777 XLIT

* CMEM 46=0 ENA

**CMEM 47=1 DRI

* • Dot

CALL	1	2	3	4	5	6
DGC1	EXT. SOURCE	A REG.	B REG.	ALU OPERATION	ALU DEST.	MEM CONTROL
DGC2	RANDOM	TEST COND.	TSC	FSC	LITERAL	

→ Data Manipulation Logic + Mem.

→ Control Processor Logic

NOVA 4 MICROCODE HAND

ONLY 5 CHAR IDENTIFY A SYMBOL TO THE ASSEMBLER BECAUSE OF THIS SOME OF THE ALU FUNCTION NAMES SEEM RATHER CONTRIVED + THEY ARE. THE OPERANDS ARE IDENTIFIED AS P(PLUS) M(MINUS) OR(OR) ND(AND) XR(XOR). "." AFTER AN OPERAND CAUSES THE OPERAND CONSIDERED AS INVERTED

```
; ALU function field
; Only 5 characters identify a symbol to the assembler
; Because of this, some of the ALU function names seem rather contrived
; and they are. The operands are identified as: A,B,D,Q and Z. The
; operations are identified as: P(plus), M(minus), OR(or), ND(and), and
; XR(xor). A "." after an operand causes the operand to be considered as
; inverted.

; CMEM bits 20,17,18,19,20,21,22
; CMEM bit 26 is the carry in bit.
; CMEM bits 17,18,19 are the function select (15,14,13).
; CMEM bits 20,21,22 are the source select (12,11,10).
```

FUNCTION SELECT
ALU SOURCE
CRY

```
APQ     = 000000 ; 0,0,0   ; A + Q
QPA     = 000000 ; 0,0,0   ; Q + A
APQP1   = 001000 ; 1,0,0   ; A + Q + 1
QPAP1   = 001000 ; 1,0,0   ; Q + A + 1
APB     = 000010 ; 0,0,1   ; A + B
BPA     = 000010 ; 0,0,1   ; B + A
APBP1   = 001010 ; 1,0,1   ; A + B + 1
BPAP1   = 001010 ; 1,0,1   ; B + A + 1
Q       = 000020 ; 0,0,2   ; Q
QP1     = 001020 ; 1,0,2   ; Q + 1
B       = 000030 ; 0,0,3   ; B
BP1     = 001030 ; 1,0,3   ; B + 1
A       = 000040 ; 0,0,4   ; A
AP1     = 001040 ; 1,0,4   ; A + 1
DPA     = 000050 ; 0,0,5   ; D + A
APD     = 000050 ; 0,0,5   ; A + D
DPAP1   = 001050 ; 1,0,5   ; D + A + 1
APDP1   = 001050 ; 1,0,5   ; A + D + 1
DPQ     = 000060 ; 0,0,6   ; D + Q
QPD     = 000060 ; 0,0,6   ; Q + D
DPQP1   = 001060 ; 1,0,6   ; D + Q + 1
QPDP1   = 001060 ; 1,0,6   ; Q + D + 1
DP1     = 001070 ; 1,0,7   ; D + 1
QMAM1   = 000100 ; 0,1,0   ; Q - A - 1
QPA.    = 000100 ; 0,1,0   ; Q + A*
QMA     = 001100 ; 1,1,0   ; Q - A
BMAM1   = 000110 ; 0,1,1   ; B - A - 1
BPA.    = 000110 ; 0,1,1   ; B + A*
BMA     = 001110 ; 1,1,1   ; B - A
QM1     = 000120 ; 0,1,2   ; Q - 1
BM1     = 000130 ; 0,1,3   ; B - 1
BMB     = 001130 ; 1,1,3   ; B - B
AM1     = 000140 ; 0,1,4   ; A - 1
AMDM1   = 000150 ; 0,1,5   ; A - D - 1
APD.    = 000150 ; 0,1,5   ; A + D*
AMD     = 001150 ; 1,1,5   ; A - D
QMDM1   = 000160 ; 0,1,6   ; Q - D - 1
QPD.    = 000160 ; 0,1,6   ; Q + D*
QMD     = 001160 ; 1,1,6   ; Q - D
MDM1    = 000170 ; 0,1,7   ; -D - 1
MD      = 001170 ; 1,1,7   ; -D

AMQM1   = 000200 ; 0,2,0   ; A - Q - 1
APQ.    = 000200 ; 0,2,0   ; A + Q*
AMQ     = 001200 ; 1,2,0   ; A - Q
AMBM1   = 000210 ; 0,2,1   ; A - B - 1
APB.    = 000210 ; 0,2,1   ; A + B*
AMB     = 001210 ; 1,2,1   ; A - B
MQM1    = 000220 ; 0,2,2   ; -Q - 1
MQ      = 001220 ; 1,2,2   ; -Q
MBM1    = 000230 ; 0,2,3   ; -B - 1
MB      = 001230 ; 1,2,3   ; -B
MAM1    = 000240 ; 0,2,4   ; -A - 1
MA      = 001240 ; 1,2,4   ; -A
DMAM1   = 000250 ; 0,2,5   ; D - A - 1
DPA.    = 000250 ; 0,2,5   ; D + A*
DMA     = 001250 ; 1,2,5   ; D - A
DMQM1   = 000260 ; 0,2,6   ; D - Q - 1
DPQ.    = 000260 ; 0,2,6   ; D + Q*
DMQ     = 001260 ; 1,2,6   ; D - Q
DM1     = 000270 ; 0,2,7   ; D - 1
AORQ    = 000300 ; 0,3,0   ; A or Q
QORA    = 000300 ; 0,3,0   ; Q or A
AORB    = 000310 ; 0,3,1   ; A or B
BORA    = 000310 ; 0,3,1   ; B or A
DORA    = 000350 ; 0,3,5   ; D or A
AORD    = 000350 ; 0,3,5   ; A or D
DORQ    = 000360 ; 0,3,6   ; D or Q
QORD    = 000360 ; 0,3,6   ; Q or D
D       = 000370 ; 0,3,7   ; D
ANDQ    = 000400 ; 0,4,0   ; A and Q
QNDA    = 000400 ; 0,4,0   ; Q and A
ANDB    = 000410 ; 0,4,1   ; A and B
BNDA    = 000410 ; 0,4,1   ; B and A
Z       = 000430 ; 0,4,3   ;
DNDA    = 000450 ; 0,4,5   ; D and A
ANDD    = 000450 ; 0,4,5   ; A and D
DNDQ    = 000460 ; 0,4,6   ; D and Q
QNDD    = 000460 ; 0,4,6   ; Q and D
A.NDQ   = 000500 ; 0,5,0   ; A*and Q
QNDA.   = 000500 ; 0,5,0   ; Q and A*
A.NDB   = 000510 ; 0,5,1   ; A*and B
BNDA.   = 000510 ; 0,5,1   ; B and A*
D.NDA   = 000550 ; 0,5,5   ; D*and A
ANDD.   = 000550 ; 0,5,5   ; A and D*
D.NDQ   = 000560 ; 0,5,6   ; D*and Q
QNDD.   = 000560 ; 0,5,6   ; Q and D*

AXRQ    = 000600 ; 0,6,0   ; A xor Q
QXRA    = 000600 ; 0,6,0   ; Q xor A
AXRB    = 000610 ; 0,6,1   ; A xor B
BXRA    = 000610 ; 0,6,1   ; B xor A
DXRA    = 000650 ; 0,6,5   ; D xor A
AXRD    = 000650 ; 0,6,5   ; A xor D
DXRQ    = 000660 ; 0,6,6   ; D xor Q
QXRD    = 000660 ; 0,6,6   ; Q xor D
AXRQ.   = 000700 ; 0,7,0   ; A xor Q*
Q.XRA   = 000700 ; 0,7,0   ; Q*xor A
AXRB.   = 000710 ; 0,7,1   ; A xor B*
B.XRA   = 000710 ; 0,7,1   ; B*xor A
Q.      = 000720 ; 0,7,2   ; Q*
B.      = 000730 ; 0,7,3   ; B*
A.      = 000740 ; 0,7,4   ; A*
DXRA.   = 000750 ; 0,7,5   ; D xor A*
A.XRD   = 000750 ; 0,7,5   ; A*xor D
DXRQ.   = 000760 ; 0,7,6   ; D xor Q*
Q.XRD   = 000760 ; 0,7,6   ; Q*xor D
D.      = 000770 ; 0,7,7   ; D*
XALU    = 000430 ; 0,4,3   ;
NOALU   = 001770 ; 1,7,7   ; Floating outputs (used for ALC).
```

```
; A and B register fields
; A register - CMEM 7,8,9,10,11
; B register - CMEM 12,13,14,15,16

ACS     = 000033 ;ACS ; Source accumulator (IR 1-2)
ACD     = 000013 ;ACD ; Destination accumulator (IR 3-4)
AC0     = 000020 ;  0 ;
AC1     = 000021 ;  1 ;
AC2     = 000022 ;  2 ;
AC3     = 000023 ;  3 ;
SP      = 000024 ;  4 ; Stack pointer
FP      = 000025 ;  5 ; Frame pointer
IOF     = 000026 ;  6 ; I/O flags
MS      = 000027 ;  7 ; Map status
PC      = 000030 ; 10 ; Internal program counter
IPC     = 000030 ; 10 ; Internal program counter
RD      = 000031 ; 11 ; Residual control data register
ADR     = 000032 ; 12 ; Address of the last memory reference
TEMP0   = 000033 ; 13 ; Temporary register for internal
TEMP1   = 000034 ; 14 ; microcode use
TEMP2   = 000035 ; 15 ;
TEMP3   = 000036 ; 16 ;
TEMP4   = 000037 ; 17 ;
AC4     = 000024 ;  4 ;
AC5     = 000025 ;  5 ;
AC6     = 000026 ;  6 ;
AC7     = 000027 ;  7 ;
AC8     = 000030 ; 10 ;
AC9     = 000031 ; 11 ;
AC10    = 000032 ; 12 ;
AC11    = 000033 ; 13 ;
AC12    = 000034 ; 14 ;
AC13    = 000035 ; 15 ;
AC14    = 000036 ; 16 ;
AC15    = 000037 ; 17 ;
XAREG   = 000037 ; 17 ;
XBREG   = 000037 ; 17 ;
```

F = ALU OUTPUT

```
; ALU destination field
; CMEM bits 23,24,25 (18,17,16).

Q.FOUT   = 000000 ;  0 ; Load Q with F, supply ALUOUT with F
.FOUT    = 000001 ;  1 ; No load ; supply ALUOUT with F
B.AOUT   = 000002 ;  2 ; Load B with F, supply ALUOUT with A
B.FOUT   = 000003 ;  3 ; Load B with F, supply ALUOUT with F
BQR.FOUT = 000004 ;  4 ; Load B with F shifted right, shift Q
                       ; supply ALUOUT with F
QBR.FOUT = 000004 ;  4 ; Same as BQR.FOUT
BR.FOUT  = 000005 ;  5 ; Load B with F shifted right, supply A
                       ; with F
BQL.FOUT = 000006 ;  6 ; Load B with F shifted left, shift Q l
                       ; supply ALUOUT with F
QBL.FOUT = 000006 ;  6 ; Same as BQL.FOUT
BL.FOUT  = 000007 ;  7 ; Load B with F shifted left, supply A
                       ; with F
NOOUT    = 000007 ;  7 ; Floating outputs (used for ALC).
```

```
; [illegible]ernal source field - CMEM 4,5,6

[illegible] = 000000 ; 0   ; Data bus
[illegible] = 000040 ; 1   ; Memory output bus
[illegible] = 000100 ; 2   ; External program counter
[illegible] = 000140 ; 3   ; Literal field (16 bit)
[illegible] = 000200 ; 4   ; Byte swap of ALUOUT
[illegible] = 000240 ; 5   ; Short effective address
[illegible] = 000300 ; 6   ; Pre-fetcher bus
[illegible] = 000340 ; 7   ; Null
```

F.

...A
...OUT
...ERAL
...SWAP
...A.
...P
...RC (NO EXTERNAL SOURCE

```
; Memory control field - CMEM 0,1

XMEM    = 000000 ; 0   ; Null
RPEND.  = 000001 ; 1   ; Force PEND low during this microinstruction
READ    = 000002 ; 2   ; Request a memory read
WRITE   = 000003 ; 3   ; Request a memory write
```

CMEM 17-22 + 26

CALL | 1 | 2 | 3 | 4 | 5 | 6
DGCI | EXT. SOURCE | A REG. | B REG. | ALU OPERATION | ALU DEST. | MEM CONTROL

ALU FUNCTION + ALU SOURCE

```
Test condition select field
CME' bits 36,37,38,39,40, and 41.
CHV.       = 000000 ; 00 ; Tests for data channel violation
MX.        = 000200 ; 01 ; Tests for program map exception
PFPRDY     = 000400 ; 02 ; Tests whether the PFP bus currently
                           ; represents the contents of the memory
                           ; location addressed by the PC   with Current block.
MAPXCB.    = 000600 ; 03 ; Tests if either MAP or MCH are set
SIGN       = 001000 ; 04 ; Tests the result of a two's complement
                           ; arithmetic operation in the ALU.  (1 = neg. ;
                           ; 0 = positive or zero)
PRSIGN     = 001200 ; 05 ; Tests the partial remainder sign from the
                           ; last micro-cycle, only used for divide
           = 001400 ; 06 ; Unused
           = 001600 ; 07 ; Unused
IODONE.    = 002000 ; 10 ; Tests the backpanel pin SELD for done
AUTOX.     = 002200 ; 11 ; Tests the ALUOUT bus to see if bits 1-15 are
                           ; >17 and <40.  Note: the 2901 F bus should be
                           ; equal to the ALUOUT bus for this test
ALCSKP.    = 002400 ; 12 ; Tests if an ALC instruction should skip.
                           ; Note: the ALUOUT bus and the 2901 F bus
                           ; should both be equal to the result of the
                           ; ALC instruction during this test
ALUCOUT    = 002600 ; 13 ; Tests the ALU carry out of the 2901
FAST       = 003000 ; 14 ; Tests for a high speed data channel request
INTR.      = 003200 ; 15 ; Tests for a valid interrupt request
EXC.       = 003400 ; 16 ; Tests for an exceptional condition.
                           ; Includes: DCM, INT, and NMI requests
IOBUSY.    = 003600 ; 17 ; Tests the backpanel pin SELB for busy
ALU4       = 004000 ; 20 ; Tests ALUOUT bit 4
ALU5       = 004200 ; 21 ; Tests ALUOUT bit 5
ALU6       = 004400 ; 22 ; Tests ALUOUT bit 6
ALU7       = 004600 ; 23 ; Tests ALUOUT bit 7
ALU3       = 005000 ; 24 ; Tests ALUOUT bit 3
ALU2       = 005200 ; 25 ; Tests ALUOUT bit 2
ALU1       = 005400 ; 26 ; Tests ALUOUT bit 1
ALU0       = 005600 ; 27 ; Tests ALUOUT bit 0
ALU15      = 006000 ; 30 ; Tests ALUOUT bit 15
ALU14      = 006200 ; 31 ; Tests ALUOUT bit 14
ALU13      = 006400 ; 32 ; Tests ALUOUT bit 13
ALU12      = 006600 ; 33 ; Tests ALUOUT bit 12
ALU8       = 007000 ; 34 ; Tests ALUOUT bit 8
ALU9       = 007200 ; 35 ; Tests ALUOUT bit 9
ALU10      = 007400 ; 36 ; Tests ALUOUT bit 10
ALU11      = 007600 ; 37 ; Tests ALUOUT bit 11

UARTFE     = 010000 ; 40 ; Tests if the UART overrun flag is set
FPTB.      = 010200 ; 41 ; Tests if the floating point has a trap or
                           ; is busy.
;          = 010600 ; 43 ; Unused
ZERO.      = 011000 ; 44 ; Tests the 2901 F bus for non-zero
FPB.       = 011200 ; 45 ; Tests if the floating point is busy
MEMCK      = 011400 ; 46 ; Tests if the batteries lasted through the
                           ; last power fail
FPSKI.     = 011600 ; 47 ; Tests if the current instruction should
                           ; skip because of a floating point condition
IR5.       = 012000 ; 50 ; Tests the instruction register bit 5.  Used
                           ; for defer detection on short EFA type
                           ; instructions
CONSOLE.   = 012200 ; 51 ; Tests if the console mode state bit is set
DCMRGA     = 012400 ; 52 ; Tests to see if the backpanel DCMR pin is
                           ; set.  This is different from the DCMR' test
                           ; in that this test is a direct indication
                           ; whereas the DCMR' test is synchronized and
                           ; hence delayed
ZEROD      = 012600 ; 53 ; A one microcycle delayed test of the 2901 F
                           ; bus for all zeros.
DCMR.      = 013000 ; 54 ; Tests if a data channel request is pending
FALSE      = 013200 ; 55 ; Always false
CARRY      = 013400 ; 56 ; Tests the state of the CARRY flip-flop
ALUID.     = 013600 ; 57 ; Tests bit zero of the D input to the ALU
FPEXIST    = 010400 ; 42 ; ****
TRUE       = 017600 ; 77 ; Always true
```

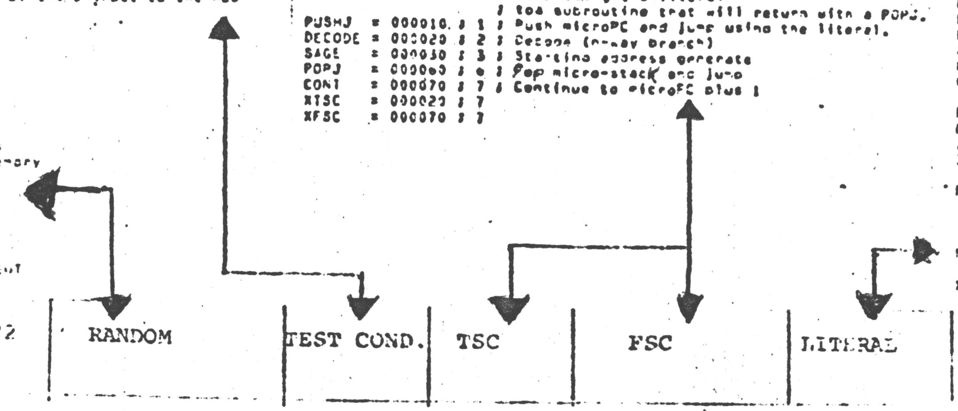
```
; Microcode test conditions for the Nova microcode

; Test conditions for the I/O flags (IOF) register
; These test conditions are prefixed with "I?"
        I?SOV    = ALU0   ; Bit  0 ; Stack overflow
        I?ION    = ALU1   ; Bit  1 ; Interrupt system on
;                 = ALU2   ; Bit  2 ; Unused
;                 = ALU3   ; Bit  3 ; Unused
        I?MPC    = ALU4   ; Bit  4 ; Mapchange
        I?0RTC   = ALU5   ; Bit  5 ; RTC select bit (AC 14)
        I?1RTC   = ALU6   ; Bit  6 ; RTC select bit (AC 15)
        I?BRTC.  = ALU7   ; Bit  7 ; RTC BUSY flag inverted
        I?BTI.   = ALU8   ; Bit  8 ; TTI BUSY flag inverted
        I?BTO.   = ALU9   ; Bit  9 ; TTO BUSY flag inverted
        I?DRTC   = ALU10  ; Bit 10 ; RTC DONE flag
        I?DTI    = ALU11  ; Bit 11 ; TTI DONE flag
        I?DTO    = ALU12  ; Bit 12 ; TTO DONE flag
        I?13MSK  = ALU13  ; Bit 13 ; MSKO bit 13 (RTC)
        I?14MSK  = ALU14  ; Bit 14 ; MSKO bit 14 (TTI)
        I?15MSK  = ALU15  ; Bit 15 ; MSKO bit 15 (TTO)

; Test conditions for the Nova Map status (MS) register
; These test conditions are prefixed with "M?"
        M?PM     = ALU0   ; Bit  0 ; Program mapping enabled
        M?DCM    = ALU1   ; Bit  1 ; Data channel mapping enabled
        M?PMI    = ALU2   ; Bit  2 ; Program map inhibited
        M?ANX    = ALU3   ; Bit  3 ; Autoindex prot. enb. after next defer
        M?DNX    = ALU4   ; Bit  4 ; Defer prot. enb. after next defer
        M?INX    = ALU5   ; Bit  5 ; I/O prot. enabled after next defer
        M?WNX    = ALU6   ; Bit  6 ; Write prot. enb. after next defer
        M?BNX    = ALU7   ; Bit  7 ; Select Map B after next defer
        M?PNX    = ALU8   ; Bit  8 ; Program map enable after next defer
        M?SCW    = ALU9   ; Bit  9 ; Single cycle write protect enable
        M?SCA    = ALU10  ; Bit 10 ; Single cycle select A=0 ; B=1
        M?AIP    = ALU11  ; Bit 11 ; Autoindex protect enable
        M?DP     = ALU12  ; Bit 12 ; Defer protect enable
        M?IOP    = ALU13  ; Bit 13 ; I/O protect enable
        M?WP     = ALU14  ; Bit 14 ; Write protect enable
        M?B      = ALU15  ; Bit 15 ; Program map select A=0 ; B=1
```

```
; Test conditions for the residual control data register(RD) and the
; hardware residual control flags (RF).
; These test conditions are all prefixed with "R?"

        R?PME    = ALU0   ; Bit  0 ; Program mapping enabled
        R?DCE    = ALU1   ; Bit  1 ; Data channel mapping enabled
        R?SCE    = ALU2   ; Bit  2 ; Single cycle mapping enabled
        R?RTCL   = ALU3   ; Bit  3 ; State of RTC at last NMI
        R?DAVL   = ALU4   ; Bit  4 ; State of UART-DAV at last NMI
        R?EOCL   = ALU5   ; Bit  5 ; State of UART-EOC at last NMI
        R?CKFSP  = ALU6   ; Bit  6 ; Masks ISTSP when asserted
        R?ONBI.  = ALU7   ; Bit  7 ; On-board interrupt request'
        R?SRTC   = ALU8   ; Bit  8 ; RTC rate select 0=60Hz ; 1=50Hz
        R?INT    = ALU9   ; Bit  9 ; Interrupts enabled
        R?CON    = ALU10  ; Bit 10 ; Console mode enabled
        R?RDRRN  = ALU11  ; Bit 11 ; Reader run enabled
        R?MS0    = ALU12  ; Bit 12 ; Map ram segment select - 2^0
        R?MS1    = ALU13  ; Bit 13 ; Map ram segment select - 2^1
        R?WPE    = ALU14  ; Bit 14 ; MMPU write protect enabled
        R?MS2    = ALU15  ; Bit 15 ; Map ram segment select - 2^2

;                                 R.MS2   R.MS1   R.MS0
;              User Map A           0       0       0
;              Scratchpad/Console   0       0       1
;              User Map B           0       1       0
;              Unused               0       1       1
;              Data Channel A       1       0       0
;              Unused               1       0       1
;              Data Channel B       1       1       0
;              Unused               1       1       1
```

```
; Lesser random field

DCMO       = 000000 ; IDCMO'  ; Enable DCMO pulse
DCMONXT    = 000400 ; IDCMON' ; Enable DCMO during next memory bus cycle
DCMASAGE   = 001000 ; DCMASAGE' ; Enable the microsequencer logic to do a
                                ; data channel decode to the DCM page
                                ; microcode and assert DCMA
CLRDCMV    = 001400 ; CLRDCMVIOL' ; Clear data channel violation flag
HOLDINT    = 002000 ; SETHOLDOFFINT' ; Set the hold-off-int flag in
                                ; interrupt system
MASKO      = 002400 ; IMSKO'  ; Enable MSKO pulse
DCMI       = 003000 ; IDCMI'  ; Enable DCMI pulse
DCMA       = 003400 ; RANDDCMA' ;
TTISRC     = 000001 ; TTISRCENB' ; Enable UART received data onto ALUIN
CONROMSRC  = 000401 ; CONROMSRCENB' ; Read an entry of the console rom
                                ; onto the ALUIN bus using ALUOUT<7-15> to
                                ; address the console rom contents
RFDEST     = 001001 ; RFDESTENB' ; Load residual control flag register
                                ; with the contents of the ALUOUT bus.
READCON    = 001401 ; READCONENB' ; Read the front console switches
RDMAP      = 002001 ; RDMAP'     ; Read map entry onto the ALUIN bus
PCDEST     = 002401 ; PCDESTENB' ; Load external PC with the ALUOUT bus
MULENB     = 003001 ; MULTIPLY'  ; Enable multiply logic
DIVENB     = 003401 ; DIVIDEENB' ; Enable divide logic
ENBIO      = 000002 ; ENBIO'  ; Enable IO control
TTIRDAV    = 000402 ; UARTDAV' ; Reset the UART data available indicator
DAOB       = 001002 ; IDATOB' ; Enable DATOB pulse
CLRPMX     = 001402 ; CLRPMX' ; Clear PMX flop in protection system
TTODS      = 002002 ; UARTDS' ; Load the UART with output data
PSCLK      = 002402 ; PSCLK'  ; Clock power marginer shift register
DAOC       = 003002 ; IDATOC' ; Enable DATOC pulse
DAOA       = 003402 ; IDATOA' ; Enable DATOA pulse
SETCRY     = 000003 ; SETCARRY' ; Set the carry bit
CLRCRY     = 000403 ; CLEARCARRY' ; Clear the carry bit
WTMAP      = 001003 ; WTMAP'  ; Write map entry using an address from the
                                ; ALUOUT bus and data from the PFP bus.
FPCPU      = 001403 ; FPCPU'  ; Enable CPU to FPP memory transfer
FPABORT    = 002003 ; FPABORT' ;
;          = 002403 ; SPARERAND' ; Unused
INHPFPMTY  = 003003 ; INHPFPEMPTY' ; Inhibit PFP empty. (used during the
                                ; XCT instruction and during console mode)
PFPDEST    = 003403 ; PFPDESTENBENB' ; Drive the PFP bus during the next
                                ; microcycle with the contents of the
                                ; ALUOUT bus during this microcycle

NORAND     = 003677 ; No operation, used with DGAN to drive the data bus
                    ; without enabling any random operations
XLIT       = 003777 ; Default value
```

```
; True and false state change fields
; CMEM bits 30,31,32 for true state change.
; CMEM bits 33,34,35 for false state change.

JUMP     = 000000 ; 0 ; Jump to literal
PJUMP    = 000000 ; 0 ; Jump using the literal
                      ; to a subroutine that will return with a POPJ.
PUSHJ    = 000010 ; 1 ; Push microPC and jump using the literal.
DECODE   = 000020 ; 2 ; Decode (n-way branch)
SAGE     = 000030 ; 3 ; Starting address generate
POPJ     = 000060 ; 6 ; Pop micro-stack and jump
CONT     = 000070 ; 7 ; Continue to microPC plus 1
XTSC     = 000020 ; 7
XFSC     = 000070 ; 7
```

```
No operation
Enable PFP for pre-fetching
Floating point processor control signal
When a FP instruction is in progress, this
signal is used to signal the FPP that a memory
reference is being initiated that will
transfer a FP operand to or from memory.
Increment the PC at the end of this
microcycle
Enable the lesser random field
Enable the lesser random field and drive
the data bus with the contents of the ALUOUT
```



This file contains a compilation of aspects of the DGC-M system that are recognized to be either new or different with respect to the Nova3.

1. Virtual console - Instead of a switches and lights console, minimal hardware console along with a microNova-like ODT debugger console is implemented. The READS function is maintained by reading via microcode, a location from the ODT memory. The ODT console maintains the capability to stop, start, continue, examine, program load, and deposit.

2. No parity memory option exists.

3. Timing relations both absolute and relative have changed between DGC-M and either the Nova3.

4. A UART is used for device TTO and TTI. The double buffering capability of the UART is used meaning that from the time that TTI done is set until the character is missed is equal to a character time rather than a bit time. When a character is sent to device TTO, the character transmit interrupt may occur instantly thereafter.

5. Certain degenerate and purposefully not entirely enumerated cases will yield different results than previous implementations. Some examples are:
   1. An integer divide which causes an overflow will have different results in AC0 and AC1.
   2. Memory read operations that causes a map error will return read data as that of the previous non-fetch memory request.

There are certainly more differences of this type.

6. Program references to non-existent main memory will return all ones rather than all zeros.

7. Up to 256K words of physical memory can be supported. This memory can be accessed by extending the physical page entry of the Nova3 map entry to a 8 bit field. This removes the validity protect bit from the map entry.

8. There are several differences between DGC-M MMPU and Nova 3 MMPU:

a) Write Status always affects status bits 1, 9 and 10 on DGC.
b) The Violation Data and Address registers always contain the values from the most recent violation.
c) NIOC to device 2 does not clear the Violation Address register.
d) Single cycle enable should only be used with non-indirect LDA or STA instructions.
e) I/O and autoindex protection does not work on single cycle.
f) A validity violation aborts an instruction during operation. Thus, an invalid LDA will not modify AC0.
g) An invalid DCH output cycle will put out the data from the most recent valid memory read.
h) Auto increment/decrement isn't supported during the enabling of the Map.

Instructions for general use:

LDB	- Load byte	- 0110DDSS1SSFFF01
STB	- Store byte	- 0110DDSSSSSFFF01
MULS	- Signed multiply	- 0111111010FFF001
DIVS	- Signed divide	- 0111111000FFF001

Diagnostic instructions:

PSMAR	- Voltage margin	- 0110D1110FFF0001
CONTST	- Microdiagnostic	- 0110D1110100F001
MSZ	- Memory size	- 0110D1111000F001
GALOP	- Memory test	- 0110D1111100F001

where DD means ACD, SS means ACS, and FF means function. LDB, STB, MULS, and DIVS are functionally compatible with their Eclipse counterparts.
Note that LDB and STB will require a new macro assembler opcode definition type because the ACS field resides in bits 8 and 9.

10. If a DOA instruction is issued to the DGC-N real time clock which attempts to set the RTC to the frequency that it is presently at then no change will occur in the timing of the RTC intervals. If the RTC frequency is set to a different frequency, then the time of the next RTC interval should be considered unpredictable.

```
                .TITL   NMEMREF

        ; Nova memory reference microcode

                RLOC    1
        ; Instruction LDA - Load accumulator
        ; -- Nova decode entry point --
        ; Moves a word out of memory and into an accumulator.
        ;
        ; The word addressed by the effective address ("E") is placed in
        ; specified accumulator.  The previous contents of the location
        ; by "E" remain unchanged.

        LDA:    DGC1    XESRC,XAREG,XBREG,XALU,.FOUT,READ
                DGC2    CRF,PMX.,CONT,JUMP,MXR1
003510          .PUSH   .

                DGC1    XESRC,XAREG,XBREG,XALU,.FOUT,XMEM
                DGC2    CRF,TRUE,CONT,XFSC,XLIT
003511          .PUSH   .

                DGC1    MEMOUT,XAREG,ACD,D,B.FOUT,XMEM
                DGC2    XRND,TRUE,SAGE,XFSC,XLIT
003512          .PUSH   .

                RLOC    1
        ; Instruction STA - Store accumulator
        ; -- Nova decode entry point --
        ; Stores the contents of an accumulator into a memory location.
        ;
        ; The contents of the specified accumulator are placed in the wo
        ; addressed by the effective address ("E").  The previous conten
        ; the location addressed by "E" are lost.  The contents of the
        ; specified accumulator remain unchanged.

        STA:    DGC1    XESRC,ACD,ACD,B,B.AOUT,WRITE
                DGC2    XRND,MAPWCB.,SAGE,JUMP,WCB1
003513          .PUSH   .
```

```
5 DGCN.                 STA @ 3100
                ; Short effective address calculation - page 0 relative
                ; -- Nova SAGE entry point --
                                                    H = TRUE FIELD
                                                    L = FALSE FIELD
                        LOC     3761        - 11 BITS
                                        DIMPUT OF 3761
        EA0:    DGC1    EFA,XAREG,ADR,D,B.FOUT,XMEM
                DGC2    XRND,IR5.,DECODE,JUMP,DEF
003761                  .PUSH   .

                ; Short effective address calculation - PC relative
                ; -- Nova SAGE entry point --

                        LOC     3741

        EA1:    DGC1    EPC.,IPC,TEMP0,APD.,B.FOUT,XMEM
                DGC2    XRND,TRUE,CONT,XFSC,XLIT
003741                  .PUSH   .

                DGC1    EFA,TEMP0,ADR,DPA,B.FOUT,XMEM
                DGC2    XRND,IR5.,DECODE,JUMP,DEF
003742                  .PUSH   .

                ; Short effective address calculation - AC2 relative
                ; -- Nova SAGE entry point --

                        LOC     3721

        EA2:    DGC1    EFA,AC2,ADR,DPA,B.FOUT,XMEM
                DGC2    XRND,IR5.,DECODE,JUMP,DEF
003721                  .PUSH   .

                ; Short effective address calculation - AC3 relative
                ; -- Nova SAGE entry point --

                        LOC     3701

        EA3:    DGC1    EFA,AC3,ADR,DPA,B.FOUT,XMEM
                DGC2    XRND,IR5.,DECODE,JUMP,DEF
003701                  .PUSH   .
                ; Trap instruction
                ; -- Nova SAGE entry point --

                        LOC     3405

        TRAPA:  DGC1    XESRC,XAREG,XBREG,XALU,.FOUT,XMEM
                DGC2    XRND,TRUE,JUMP,XFSC,TRAP
003405                  .PUSH   .

                        LOC     3425

        TRAPB:  DGC1    XESRC,XAREG,XBREG,XALU,.FOUT,XMEM
                DGC2    XRND,TRUE,JUMP,XFSC,TRAP
003425                  .PUSH   .

                        LOC     3445

        TRAPC:  DGC1    XESRC,XAREG,XBREG,XALU,.FOUT,XMEM
```

VIRTUAL CONSOLE LAB

1 - Power up the display terminal.
  - Power up the machine and observe the display.
  Does it display the following?

  OK
  !000000
  !

2 - If yes, go on, if no, consult your instructor.

3 - Consult the FRU manual pp. 12,13 for the command summary (Table 2.2)

4 - Examine all of the Internal cells, noting their meanings in table 2.1

5 - Examine some of the memory locations.
  What data pattern do they contain. Why?

6 - Deposit the following program into memory beginning at Loc. 5000, then start (RUN) at Loc. 5000. (See table 2.3)

5000-020421	5015-061111
5001-040421	5016-063511
5002-020421	5017-000777
5003-040020	5020-001400
5004-022020	5021-000006
5005-101300	5022-000000
5006-004407	5023-005023
5007-101300	5024-006412
5010-004405	5025-054517
5011-014411	5026-052440
5012-000772	5027-042111
5013-063077	5030-042040
5014-000764	5031-044524

7 - If you made no errors while depositing and running the program, there will be a message displayed. If you do not get an intelligible message displayed, go back and check your data, beginning at Loc. 5000, and try again.

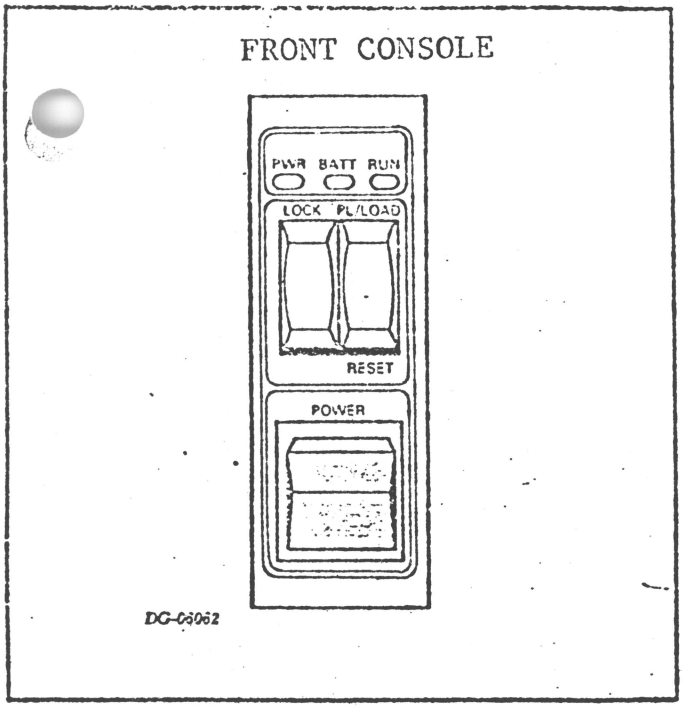


Figure 2.1 FRONT CONSOLE

Table 2.1
INTERNAL CELLS

Internal Cell #	Internal Register
0-3	The contents of the accumulators AC0 through AC3 respectively.
4	Return address (the contents of the program counter when VC was entered).
5	Stack pointer
6	Frame pointer
7	Interrupt enable flag status bit: 0 = interrupts off 1 = interrupts on
10	MMPU status bits before VC was entered (NOVA 4/X only): BIT — MEANING WHEN 1 0 — Program mapping enabled 1 — Data channel mapping enabled 2 — Program map inhibited 3-8 — Reserved for future use 9 — Single cycle write protect enable 10 — Single cycle select (0 = A, 1 = B) 11 — Auto increment/decrement protect enable 12 — Defer protect enable 13 — I/O protect enable 14 — Write protect enable 15 — Program map select (0 = A, 1 = B)
11	Data switch register: Replaces the conventional console data switches. When the system is in run mode (i.e., not in VC mode) and a READS instruction is executed, the 16-bit contents of this register are read by the CPU.
12	Value of the carry bit

Table 2.3
FUNCTION COMMANDS

Command	Function
P Proceed	Starts program execution at the memory location specified by the contents of internal cell number 4 (See Table 2.1).
*n*R RUN	Issues an I/O Reset; clears the MMPU (NOVA 4/X only), and starts program execution at the memory location specified by the octal number *n*.
I IORST	Issues an I/O Reset, and clears the MMPU (NOVA 4/X only).
*n*L LOAD	Performs a program load from the device whose device code is equal to *n*. Bit 0 of *n* is a 0 for a low-speed device, and is a 1 for a high-speed device.
F FLOAD	Performs a DG field service cassette bootstrap load (for DGC use only).
K KILL	Cancels the entire line just typed, and prints a question mark (?).

Table 2.2
CELL COMMANDS

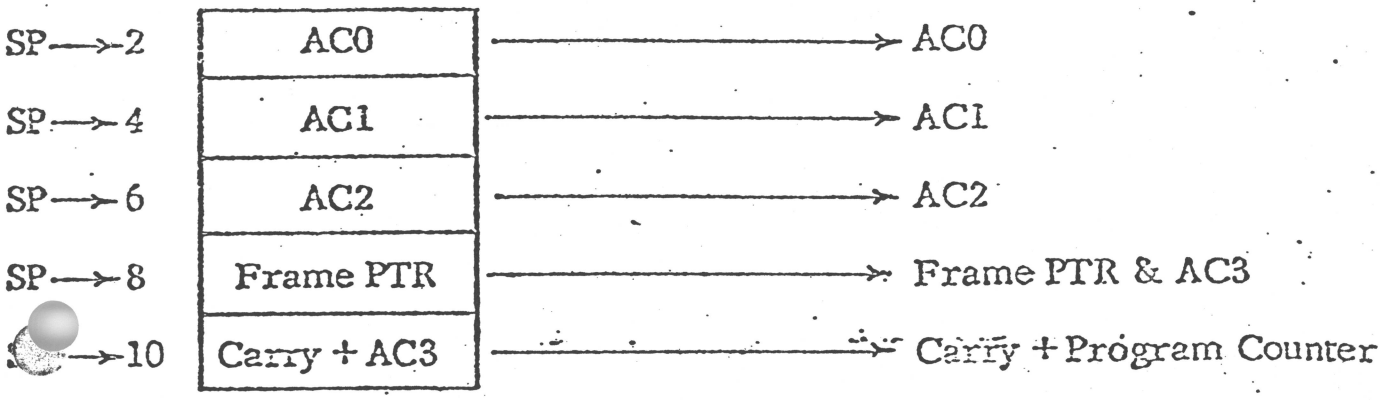
Command	Function
*n*A	Open the internal cell whose internal cell number is equal to *n*. (Refer to Table 2.1 for the internal cell numbers.)
n/	Open the memory location whose address is equal to the octal number *n*: *n* is a 17-bit address.
(Carriage Return)	Close the current cell, and open the next consecutive cell.
(Line Feed or New Line)	Close the current cell, but do not open another.
/	Close the current cell and open the memory cell whose address is equal to the contents of the current memory or internal cell.

SAVE

1. Increment Stack PTR
2. AC0 ⟶ Stack
3. Increment Stack PTR
4. AC1 ⟶ Stack
5. Increment Stack PTR
6. AC2 ⟶ Stack
7. Increment Stack PTR
8. Frame PTR ⟶ Stack
9. Increment Stack PTR
10. Carry + AC3 ⟶ Stack
11. Stack PTR ⟶ Frame PTR
12. Stack PTR ⟶ AC3

RET

1. Frame PTR ⟶ Stack PTR
2. Stack ⟶ Carry + PC
3. Decrement Stack PTR
4. Stack ⟶ Frame PTR
5. Stack ⟶ AC3
6. Decrement Stack PTR
7. Stack ⟶ AC2
8. Decrement Stack PTR
9. Stack ⟶ AC1
10. Decrement Stack PTR
11. Stack ⟶ AC0
12. Decrement Stack PTR



At end of SAVE
Stack PTR, Frame PTR, and AC3 point to last word pushed onto Stack.

At end of RET
AC3 and Frame PTR point to 5th word of last frame and Stack PTR points to last word pushed on stack.

CHAPTER 11
CPU OPERATION

board is a multi-fuctnion board consisting of a central (CPU), main memory, a virtual console, and an munications interface. A real-time clock is optional.

e system's activities by executing assembly language t executes the NOVA instruction set enhanced with load nstructions and, optionally, signed multiply and divide he CPU operates in two modes: run and console. In run s instructions stored in main memory. In console mode, ructions stored in the virtual console.

ovides either 32K bytes (16K words) to 64K bytes (32K c random access memory.

- allows a user to examine and modify the system's state (system terminal) connected to the resident asynchronous

erface - a programmed I/O controller which contains er and a receiver. It provides full-duplex communication asynchronous terminal and the CPU.

- an option which provides the system with four program bases.

of the CPU board are interconnected by three 16-bit LUOUT, MBUS. The CPU board, in turn, is connected to the rs by the 48-line I/O bus. This bus consists of the , which transfers all data, plus 32 lines which carry program interrupt, data channel, and system control 11.1 shows the interconnection of these buses and the he CPU board.

mponents of the CPU are a control processor and a data t.

essor executes the NOVA 4/C instruction set by h assembly language instruction as a macroinstruction. acroinstruction and then executes the sequence of s needed to perform the specified function. When microinstructions control the data paths and the data manipulation unit as well as the operation of main /output.

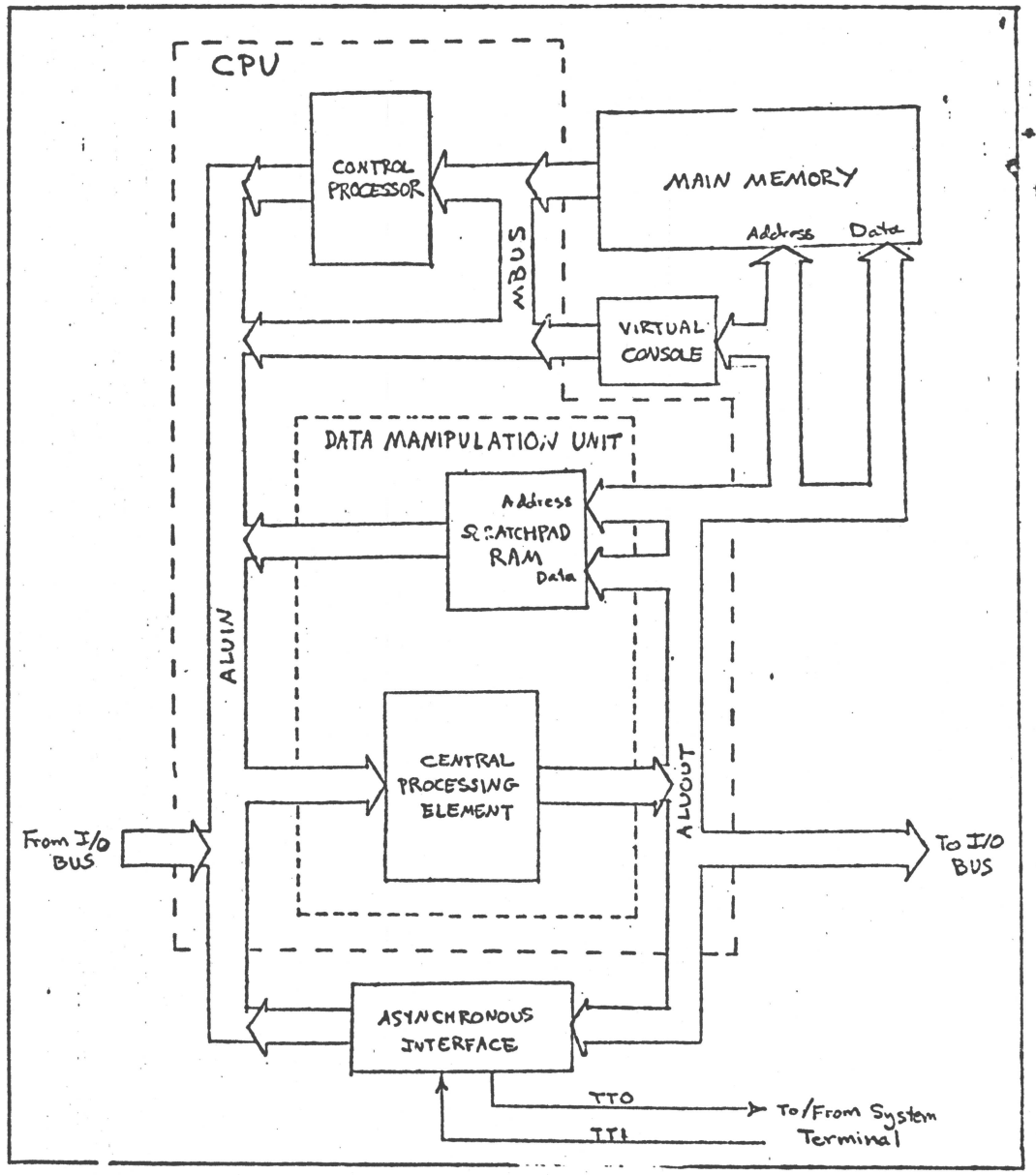
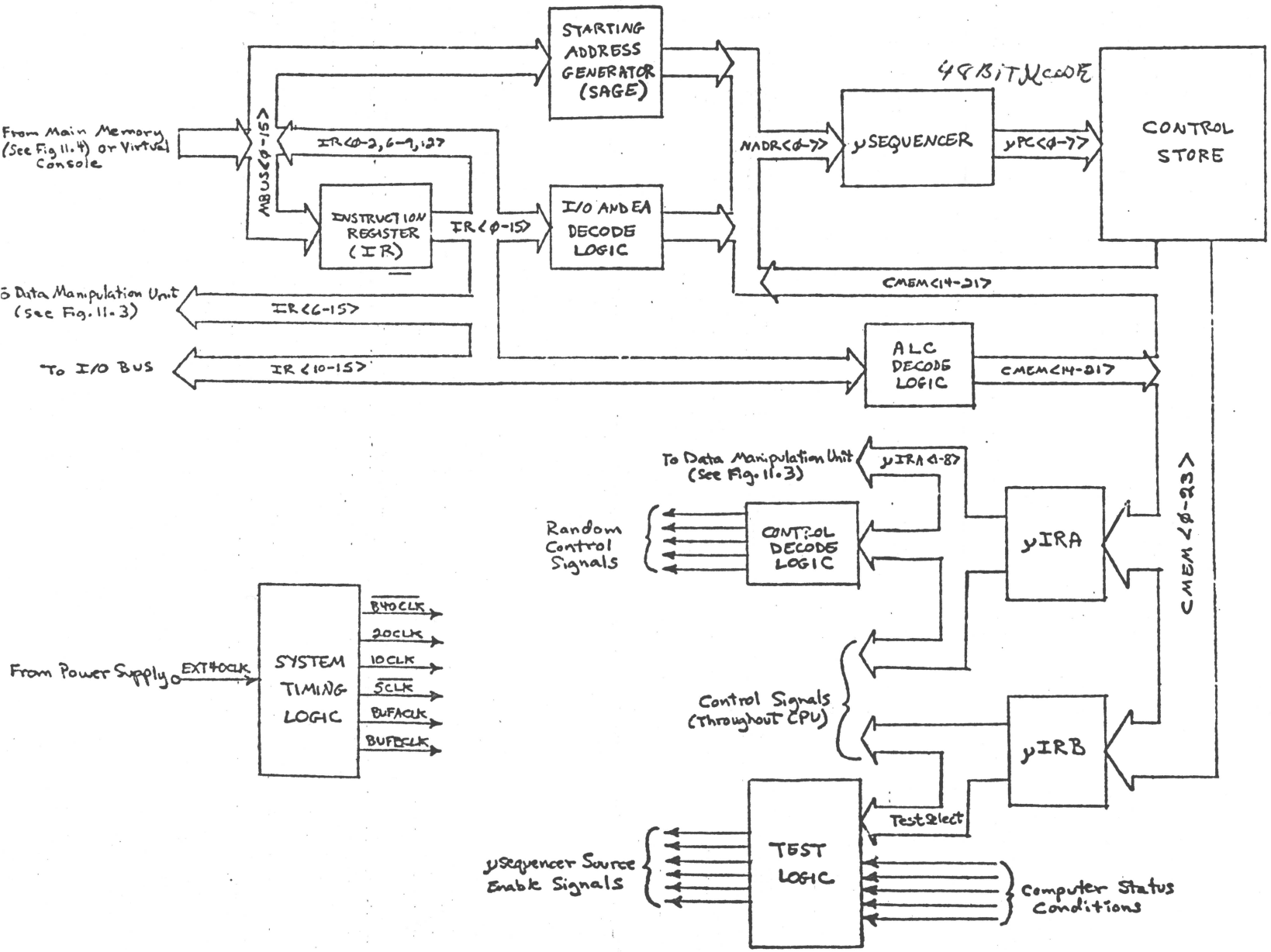


Figure 11.1 CPU Board Block Diagram

Figure 11.2 Control Processor Block Diagram



NOVA 4 Q

Figure 11.3 Data Manipulation Unit Block Diagram

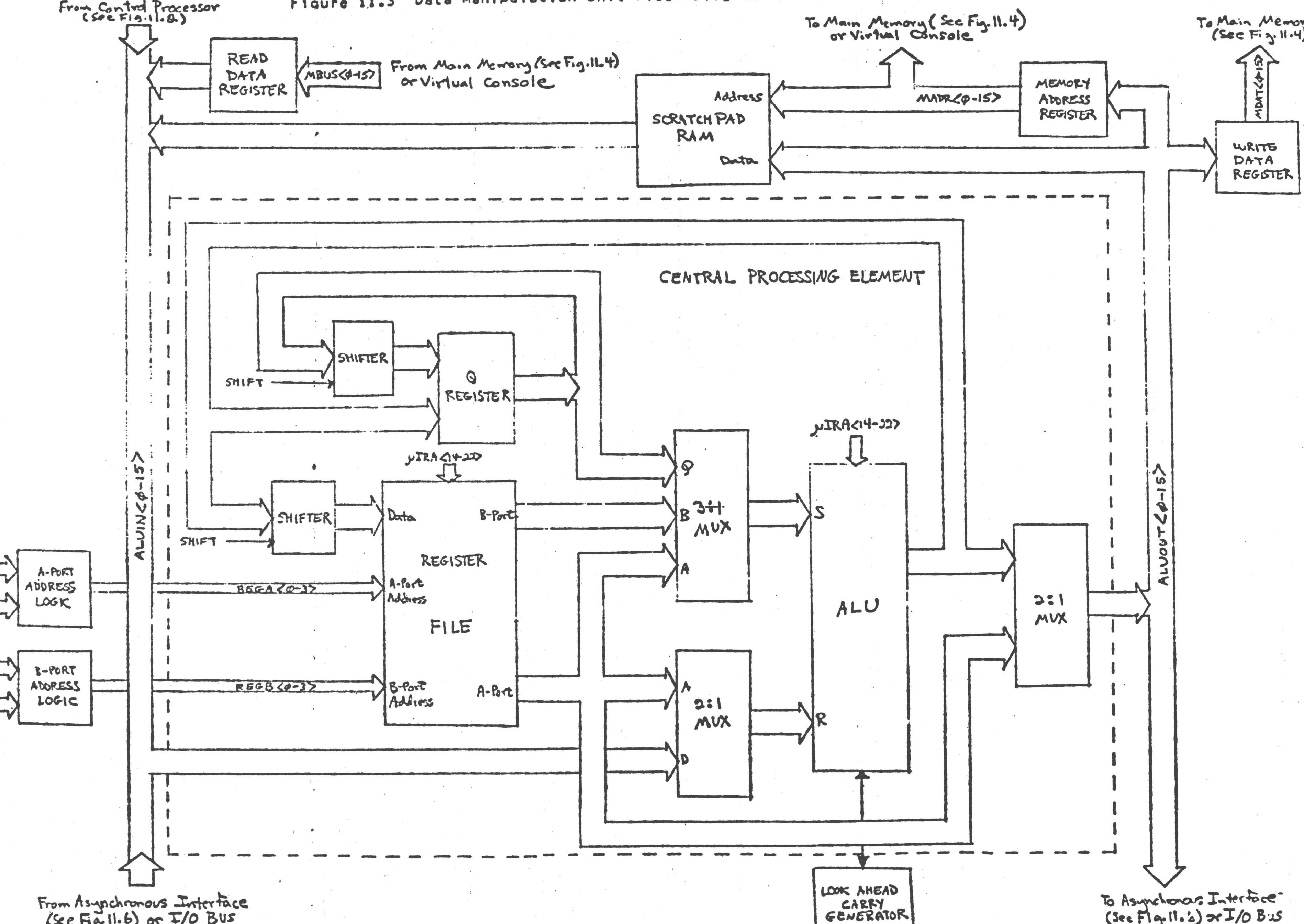
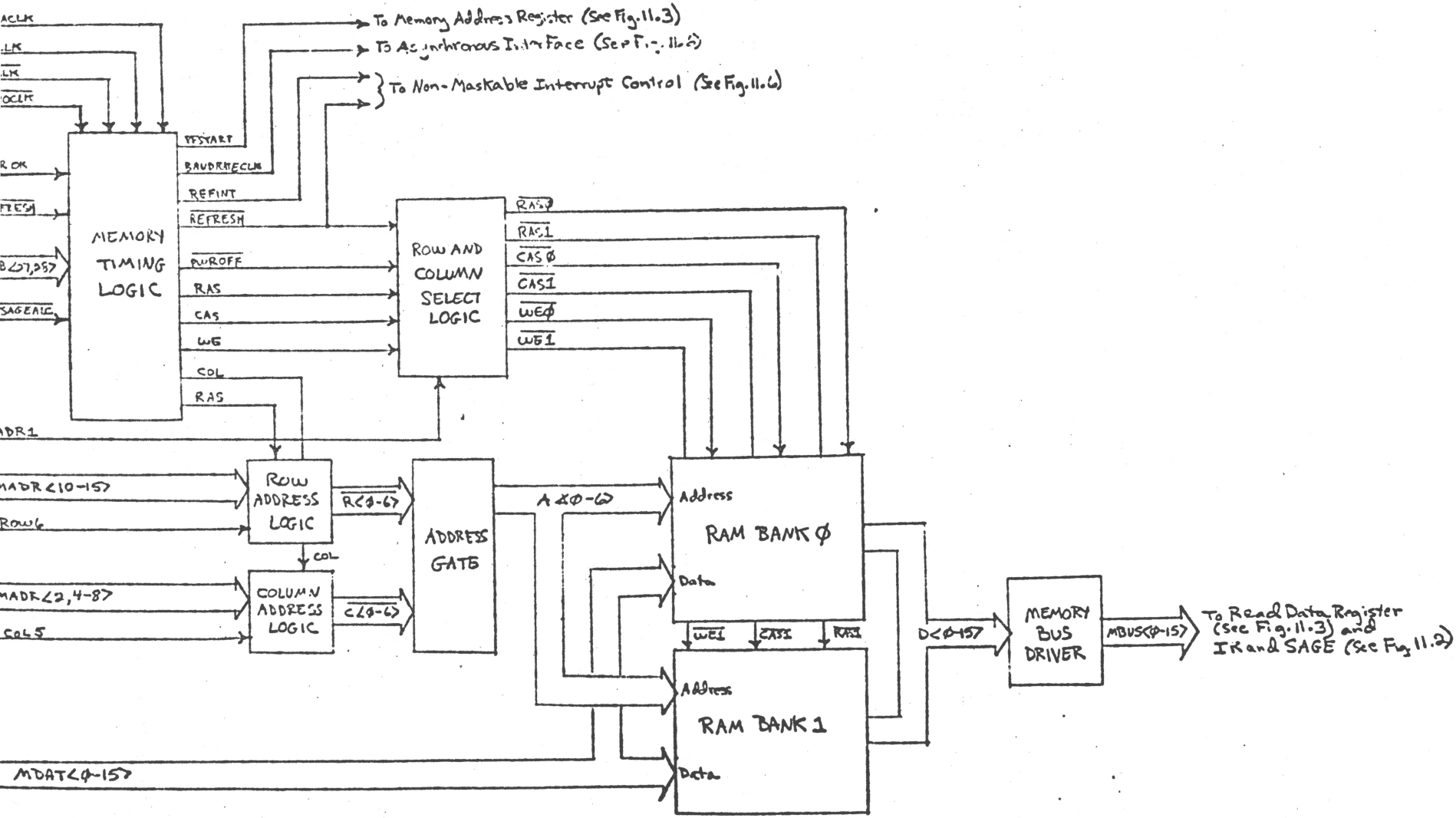


Figure 11.4 Main Memory Block Diagram



b burst refreshes, while the battery backup operates, the gic starts a battery backup refresh operation every 12.8 time, the memory address register provides the row refresh operations. The row address section of this ally a presettable counter; however, it only functions as a power failure.

of a battery refresh, the memory timing logic drives igh state to start the refresh operation. When PFSTART state, the memory address register increments the row The register supplies this address to main memory via ain memory carries out the refresh operation in the same ut a normal refresh operation.

the timing for the refresh operations and a refresh

sole resides in a 512-word by 16-bit ROM. This ROM also embly language instructions needed to implement the st. The memory address register provides addresses to MADR bus.

processor operates in console mode, the virtual console instructions which the control processor executes. When sole ROM is enabled, it sends instructions to the IR and MBUS. Data from the virtual console ROM goes to the er. The virtual console can access main memory by ling the RAM array to use the MBUS.

TERFACE AND REAL-TIME CLOCK

interface is a programmed I/O controller which provides munications between the CPU and a serial, asynchronous ther a 20mA current loop or an EIA RS-232C communications

nent of the interface is a Universal Asynchronous itter (UAR/T). The interconnection between the UAR/T and components of the interface is shown in Figure 11.6.

h contains both a transmitter and receiver, is the link between the interface and the terminal. Jumpers characteristics. The BAUDRATECLK clock from the memory ovides the baud rate. The frequency of this clock is umper in the memory timing logic. BAUDRATECLK supplies nsmitter and receiver clock inputs.

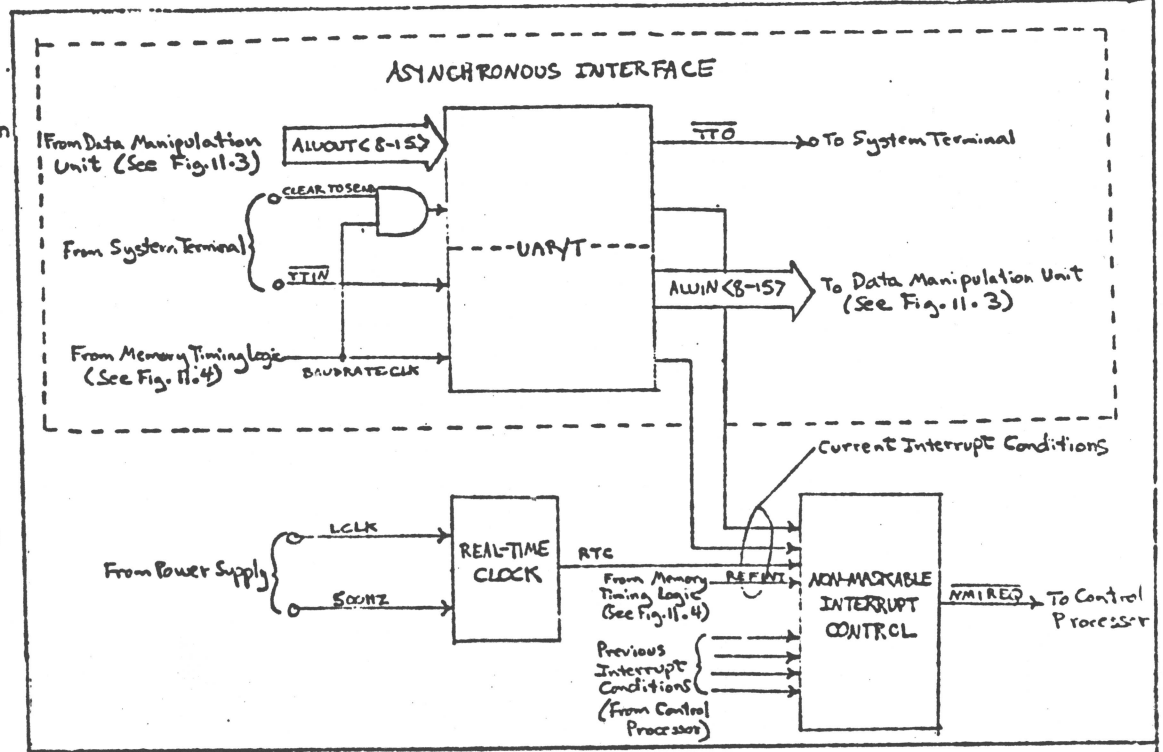
When communicating with the terminal, the UAR/T transmits and receives character codes in serial form. When communicating with the CPU under program control, the UAR/T's transmitter receives data in parallel form from the ALUOUT bus and the receiver places data in parallel form on the ALUIN bus.

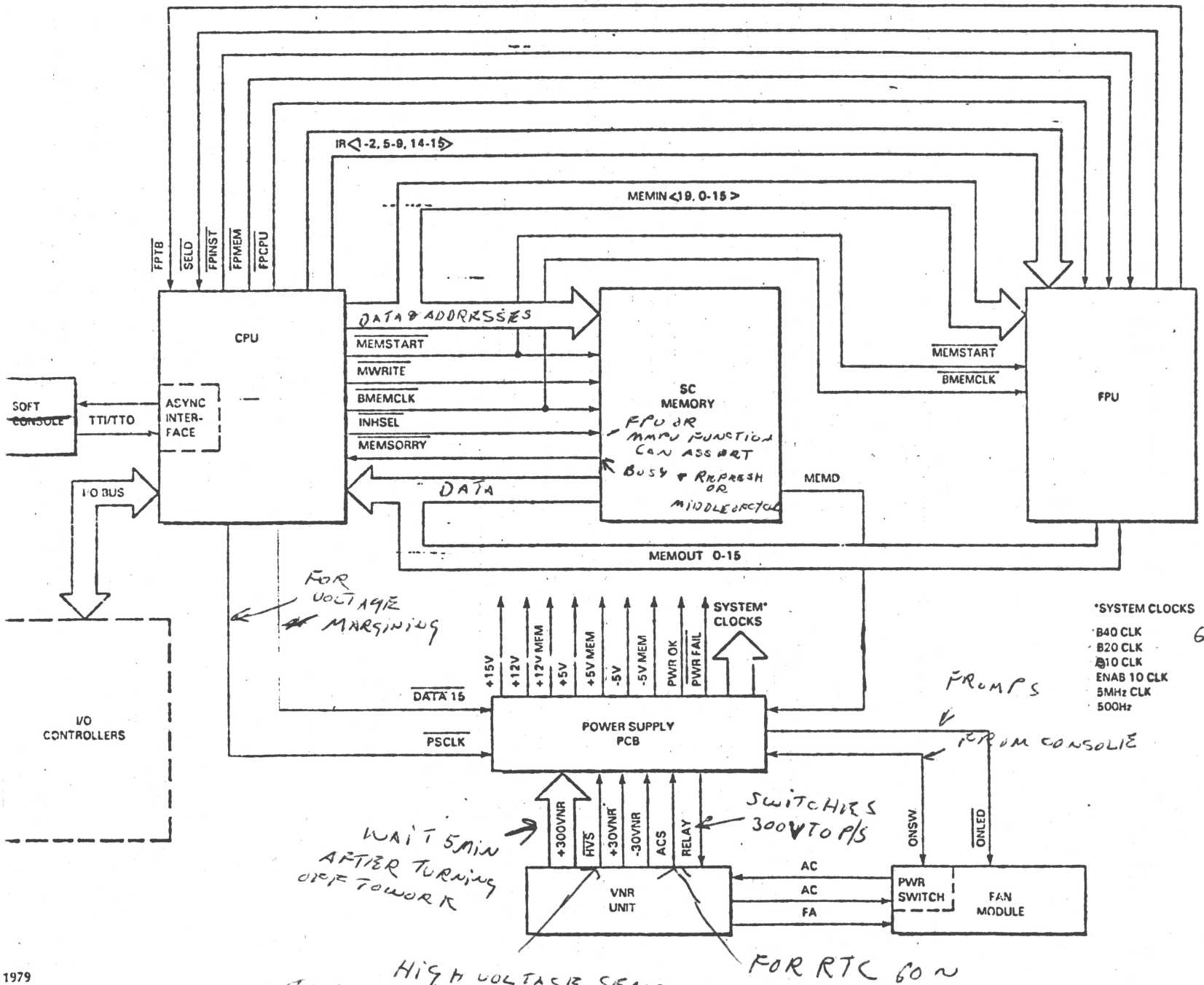
The modem control signal, Clear to Send, is connected to the UAR/T's transmitter clock input line and inhibits data transmission when at the low state.

The real-time clock option provides four program-selectable time bases: power line frequency, 10 Hz, 100 Hz, and 1000 Hz.

Both the interface and the real-time clock option use a non-maskable interrupt control which is independent of the standard I/O interrupt facility. The Busy and Done flags and the priority mask bits for these devices are located in the I/O flag register, which resides in the data manipulation unit.

Figure 11.6 Asynchronous Interface and Real-Time Clock Block Diagram





NOVA 4/S/X SYSTEM

REVIEW EXERCISE

LOC.

```
10 LDA 0@ 20
11 DSZ 50
12 STA 0@ 50
13 ISZ 51
14 JMP .-4
15 JSR 40
16 HALT
   .
   .
   .
40 LDA 2, 52
41 LDA 1@ 30
42 SUB# 1,2 SZR
43 JMP.-2
44 JMP 0,3
```

DATA

```
20-77
30-300 = 77
50-300 -277 -276
51-177775 - 76
52-77
100-7
101-77
102-777
```

STACK INSTRUCTION EXERCISE

Follow the program through and fill in the appropriate accumulators and memory locations when the program halts.

```
Start:  100   LDA Ø, 207
        101   MTSP Ø
        102   LDA 1, 21Ø
        103   MTFP 1
        104   LDA Ø, 211
        105   LDA 1, 212
        106   LDA 2, 213
        107   LDA 3, 214
        110   PSHA Ø
        111   PSHA 1
        112   PSHA 2
        113   POPA Ø
        114   POPA 1
        115   JSR 2ØØ
        116   JMP 2Ø6

        200   SAV
        201   PSHA Ø
        202   PSHA 1
        203   PSHA 2
        204   PSHA 3
        205   RET
        206   TRAP
        207   ØØ1ØØ2
        210   ØØ1ØØ5
        211   111111
        212   111ØØØ
        213   ØØØ111
        214   177ØØØ

        47    ØØ1ØØØ
        1000  MFSP Ø
        1001  MFFP 1
        1002  HALT
```

AC0 = ~~111000~~ 1003

AC1 = ~~111111 111000~~ 1003

AC2 = 000111

AC3 = 1005

SP = 1003

FP = 1005

46 = 206

1003 = ~~111111~~ 111000

1004 = ~~111000~~ 111111

1005 = 000111

1006 = 1005

1007 = x000116

1010 = 111000

1011 = 111111

1012 = 000111

1013 =

1014 =

LDB, STB EXERCISE

LOC.

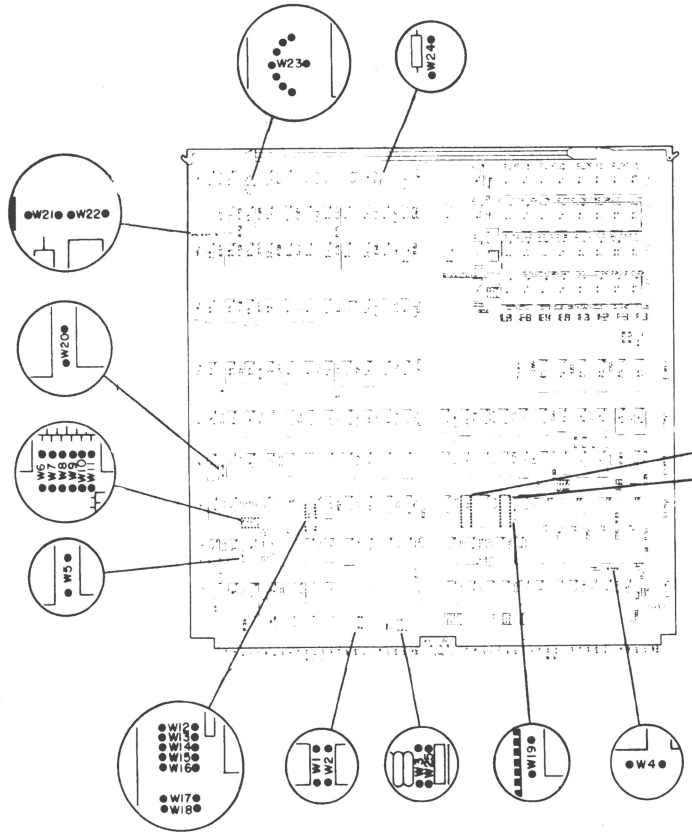
```
30 LDA  3, 45
31 LDA 0, 47
32 MOVOL 0,0
33 LDB 0,1
34 AND 3,0
35 LDB 0,2
36 STB 0,1
37 INC 0,0
40 STB 0,2
41 ISZ 47
42 DSZ 46
43 JMP.-12
44 HALT
```

DATA

45-177776
46-000004
47-000050
50-000377
51-177400
52-000077
53-176000

ACCUMULATORS

AC0
AC1
AC2
AC3-177776



DEVICE CODE JUMPERS FOR FRONT PANEL AUTOMATIC PROGRAM LOAD

SELECT THE PROGRAM LOAD DEVICE CODE BY INSTALLING JUMPERS W11, W8, W6, W7, W9, W10, AS FOLLOWS:

JUMPER OUT = 1 JUMPER IN = 0

EXAMPLE JUMPERING FOR DEVICE CODE 27_8:

W11	W8	W6	W7	W9	W10
IN	OUT	IN	OUT	OUT	OUT

W4 IS NOT INSERTED IF THE PROGRAM LOAD DEVICE IS A HIGH SPEED DEVICE, OTHERWISE IT IS INSERTED.

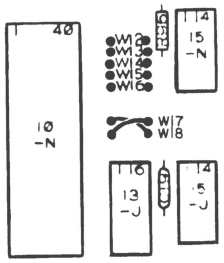
TYPE OF TRANSMISSION JUMPERS

TYPE OF TRANSMISSION	JUMPERS INSERTED*
20MA CURRENT LOOP	W1, W3
EIA RS232-C	W2

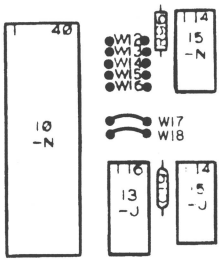
* JUMPER 25 IS INSERTED IF THE SYSTEM TERMINAL IS A TELETYPE, OTHERWISE IT IS NOT INSERTED.

* JUMPERS W17 AND W18 MUST ALSO BE INSERTED AS SHOWN BELOW.

20MA CURRENT LOOP



EIA RS232-C



JUMPERS W17 AND W18 MUST NOT TOUCH!

STOP BIT JUMPERS

NUMBER OF STOP BITS	W15 JUMPER POSITION
1	IN
2	OUT

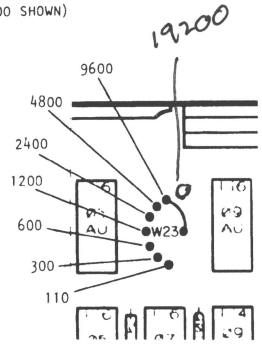
PARITY JUMPERS

TYPE OF PARITY	JUMPER POSITION W12	W16
EVEN	OUT	IN
ODD	IN	IN
NONE	OUT	OUT

CHARACTER LENGTH JUMPERS

CHARACTER LENGTH	JUMPER POSITION W13	W14
5 BITS	IN	IN
6 BITS	OUT	IN
7 BITS	IN	OUT
8 BITS	OUT	OUT

W23 IS INSERTED TO DETERMINE THE BAUD RATE AS SHOWN BELOW: (9600 SHOWN)



W22 IS NEVER INSERTED.

THE FOLLOWING JUMPERS ARE ALWAYS INSERTED:

W5
W19
W20
W21
W24

DG-06191

Figure 12.13 CPU ROM and jumper locations

Assembly No.		ROM Part No.		Description
With ROMs	Without ROMs	Loc 29N	Loc 33N	
005-13554	005-12413	100-1831	100-1832	NOVA 4/C CPU with 32K byte (16K word) memory
005-13553	005-12413	100-1833	100-1834	NOVA 4/C CPU with 32K byte (16K word) memory and real-time clock
005-13552	005-12413	100-1835	100-1836	NOVA 4/C CPU with 32K byte (16K word) memory and multiply/divide option
005-13551	005-12413	100-1837	100-1838	NOVA 4/C CPU with 32K byte (16K word) memory, real-time clock, and multiply/divide option
005-13550	005-12415	100-1831	100-1832	NOVA 4/C CPU with 64K byte (32K word) memory
005-13549	005-12415	100-1833	100-1834	NOVA 4/C CPU with 64K byte (32K word) memory and real-time clock
005-13548	005-12415	100-1835	100-1836	NOVA 4/C CPU with 64K byte (32K word) memory and multiply/divide option
005-13547	005-12415	100-1837	100-1838	NOVA 4/C CPU with 64K byte (32K word) memory, real-time clock, and multiply/divide option

Table 12.1 CPU board assembly numbers and ROM part numbers