NORD

COMPUTER SYSTEMS

NORD-20

HARDWAFE MANUAL II

A/S NORSK DATA-ELEKTRONIKK

Erich Mogensøns vei 38, Oslo 5

NORD-20

HARDWARE MANUAL II

TABLE OF CONTENTS

+++
+

INTRODUCTION

This is the first edition of the NORD-20 HARDWARE MANUAL II. In order that later editions shall be as complete as possible we want you to state your wishes for further information, suggestions for changes, etc. on the preaddressed sheet for comment and evaluation in the back of this book.

Please note, when reading this book, that some of the information, that belongs together, has been separated into a chapter and one or more appendices.

Wiring lists and circuit diagrams for NORD-20 are contained in the book called NORD-20 HARDWARE MANUAL I.

--ooOoo--

1 GENERAL TIMING CPU

Generally an instruction is composed of several cycles.

Each cycle will consist of the time intervals T1, T2 and T3. A T4 is added after T3 during IC2 to give time to decode the new instruction.
Specific cycle actions:

IC1 : Use P as address for new instruction fetch.

IC2 : Increment P and wait for new instruction from memory.

AC1 : Address computation cycle during memory reference instructions..

AC2 : Additional address computation cycle when required.

EC1 : Used for P→L during JPL or A→IOTD during IOT.

EC2 : Used for argument transfer or arithmetic during memory reference instructions, IOT, shift or ARG.

EC3 : Fetch "source" cycle during SKP, ROP, REG.

EC4: Test or arithmetic cycle during SKP, ROP, REG and BOP.

Actions during a cycle can be devided into two main phases.

During T1 and T2 the ALU is doing some kind of arithmetic and at T3 the result is normally written back into a register.

The table "Main Instruction and Cycle Actions" illustrates this by using the notation: MDB+A→A

This is an example from the ADD instruction during EC2 where T1 and T2 are found to the left of the arrow. To the right of the arrow is indicated what is done with the result during T3.

Normally T1, T2 and T3 are of equal length, but there are a few exeptions for T3.

a) During memory reference, CPU will be in wait state until it receives "data ready" from memory exept in IC1.

b) During the "conditional instructions" IOT, SKP, MIN, CJP, BOP and also MIS in cycle EC2 or EC4, T3 will be "stretched" some 200 ns. This is necessary to detect the conditional result and take appropriate action before leaving the cycle.

c) During shift T3 will be extended in EC2 until the necessary number of shifts are executed.

d) The table "Main Instructions and Cycle Actions" uses indexing to indicate which of several possible actions is appropriate.

Example:

During IC1

$$\begin{Bmatrix} 1^1 \\ 0 \end{Bmatrix} + \begin{Bmatrix} 0^3 \\ P \end{Bmatrix} \rightarrow \begin{Bmatrix} R \\ BA \\ 0^2 \end{Bmatrix}$$

2 NORD-20 ADDRESS COMPUTATION

The address computation within an instruction is executed in the cycles AC1 and AC2.

AC1 will always be entered during a memory reference instruction. However, if there is indirect addressing or double indexing (,B ,X), AC2 will be entered.

During AC1, the displacement Δ will be added to either the R, B or X registers depending on the addressing mode. (R equals old P).

If AC2 is required, there are several possibilities:

1. Indirect addressing differentiates between using the result from AC1 or a possible new indirect address arriving from memory.

2. ,X in AC2 will differentiate between using X or Zero as the other argument.

The results from 1. and 2. are added, forming the effective address.

3 NORD-20 INTERRUPT

General:

There exists two kinds of interrupt in the NORD-20 computer.

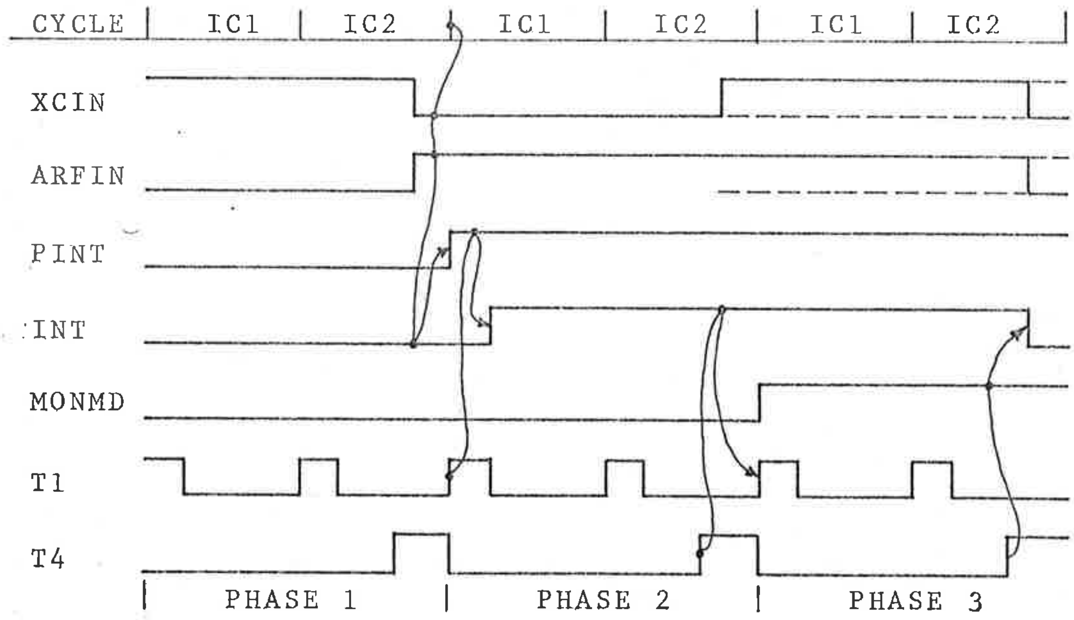
a) External Interrupt (I/O)
b) Programmed Interrupt

These are destinguished by XINT or PINT in the status register being set.

Both may occur at the same time and it must be the responsibiltiy of software to define priority. Any instruction causing PINT to be set will not be executed, but effective address will be computed where appropriate.

Also every instruction where INT is sampled will not be executed.

Programmed Interrupt



The example above shows interrupt on a non-implemented instruction which in Nord-1 code does not include address computation. (SWAP).

Phase 1:

The instruction is fetched and decoded. XCIN drops because of non-implements instruction and ARFIN is raised because of no need to compute address.

Phase 2:

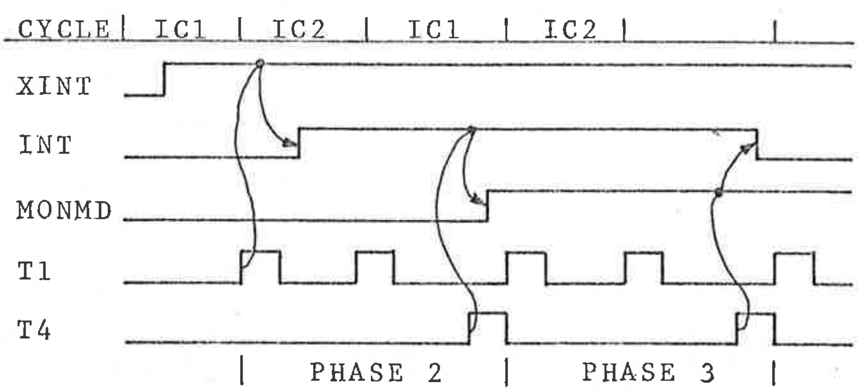
On exit from Phase 1, the CPU is forced back into IC1 on condition $XCIN_0 \cdot INT_0$. Here INT will be set on $IC1_1 \cdot T1_1 \cdot CL_1 \cdot PINT_1$, and on exit from Phase 2 INT will force IC1 to be executed once more. It should be noted that the Phase 2 acts as a normal instruction fetch, but the instruction fetched will have no effect. The P register is not incremented either so, Phase 2 is dummy, and has the only purpose to sample possible interrupt and switch to Monitor mode.

Phase 3:

Before exit from Phase 2, MONMD is set and the next instruction fetch will be requested from the location pointed at by the P register in the Monitor block.

On exit from Phase 3. INT is reset, and processing continues in Monitor mode.

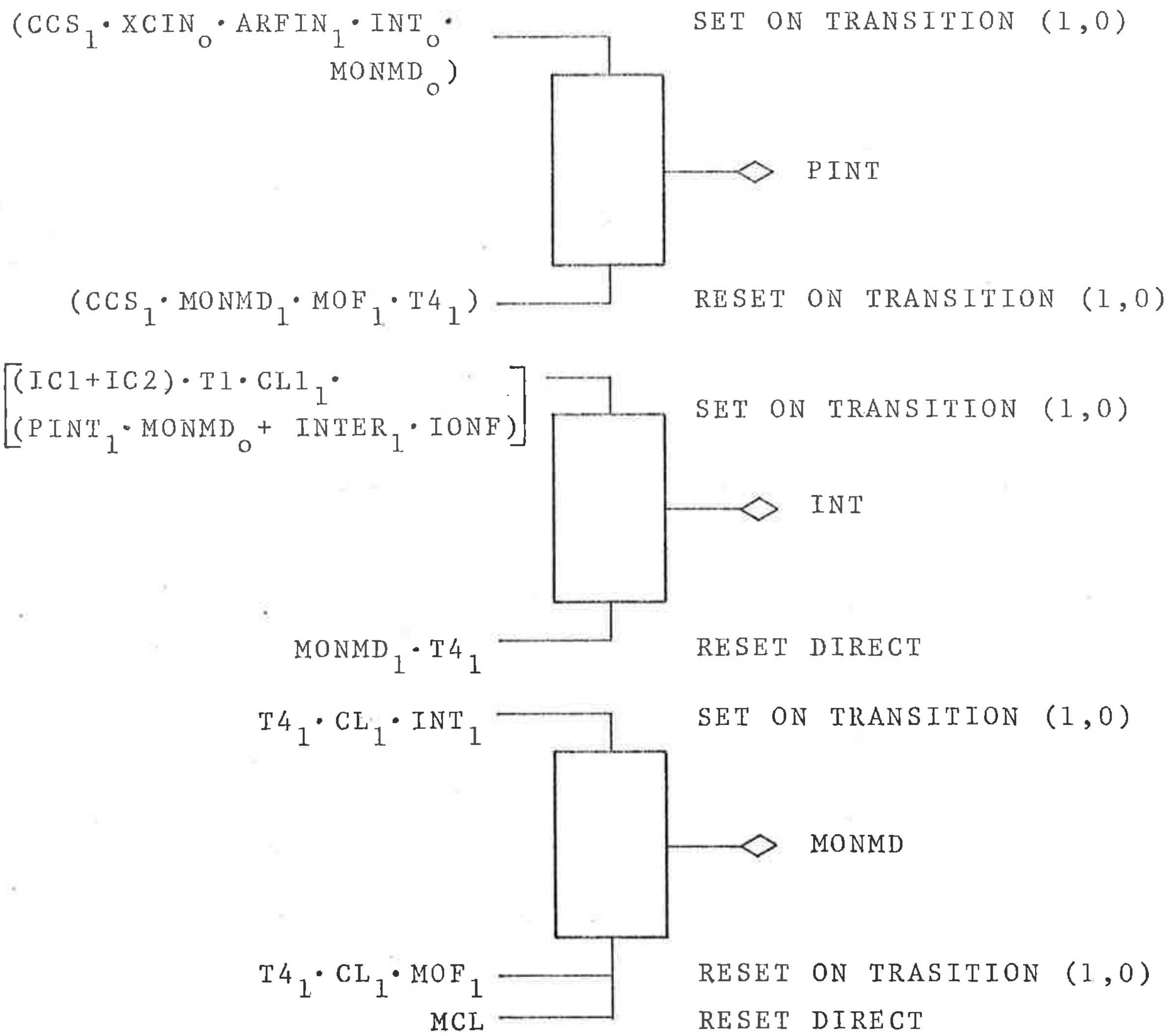
If the instruction interrupting had been a memory reference instruction, address computation would have been finished before exit from Phase 1. This is controlled by ARFIN, which would have been raisedon exit from AC1 or AC2.



This example shows how I/O interrupt is handled. Phase 1, which in programmed interrupt will generate an interrupt condition by setting PINT, is omitted here and replaced by some external mechanism.

However, from entring Phase 2 XINT and PINT are treated exactly in the same way.

The main difference is that XINT will occur asynchronously to IC1 and IC2. Because CPU omits IC1 after an effective jump and goes directly to IC2, interrupt will be sampled on $T1_1 \cdot CL_1$ both in IC1 and IC2.



4 NORD-20 CPU DATA BUSES

1. CPB is a 16 bits wide data line which, depending on time, reflects the information on MDB, IOTD, STS and the BD register.
   CPB fans out to:

   a) Instruction register, IR
   b) Status register, STS
   c) Bit skip selector, CONDB
   d) CPB zero detector, CPBZ
   e) Conditional jump, CONDJ
   f) 16 general registers
   g) Via conditional inverter to SUM.

2. RB contains information from either one of the 16 general registers, or if no register is selected, all zero.

   RB is 16 bits wide, and only fans out to SUM.

3. NOTE.
   Both CPB and RB have logic "1" low.

MULTI-DEVICE CONTROLLER (MDVC)

FUNCTIONAL DESCRIPTION

General Information:

1. Master Clear puts the MDVC in Reset mode. Now all 32 INTE (i) flip-flops are sequectially scanned from 31 to 0 and set to "0". Reset mode is cleared when passing INTE (0).

2. During interrupt scan, which is performed continuously when IOTE for this particular MDVC is zero, scanning starts form device number 31_{10} and down. ($i=31_{10}$). When both DVC(i) are "1", (i) is copied into Interrupt Priority Register and INTER is set to one. Then the scan counter is reset to 31_{10} re-starting the scan.

3. If (i) becomes equal to the content of the Interrupt Priority Encoder, scanning restarts from $i=31_{10}$ and neither Inter nor Interrupt Priority Register are set.

4. MDVC handles 32 consequtive device numbers. 28 are used for external devices and 29-30-31-32 are used for internal housekeeping on MDVC.

5. MDVC is program compatible to Simplex Control 160, but it also provides several other features:

   Dev.no. 31: Set Interrupt Priority Register
   " " 30: Read Interrupt Priority Encoder
   " " 29: Start real time clock
   " " 28: Not used

6. Interrupt Priority Encoder indicates the highest device no. having interrupt pending at the moment.

7. IOT SKA 30_{10} skips if a device in this 32 dev. group has given interrupt.

8. Interrupt will be disabled from all devices corresponding to numbers less than the content of the Interrupt Priority Register.

9. The Real-Time Clock period will be fixed wired with selected components for times between 100 µs and 100 ms, according to wish. Probable accuracy will be less than 10% due to variations in the ambient temperature.

Sub-Functions:

1. Interrupt Enable Memory: INTE(i) Contains 32 separate interrupt enable flip-flops, one for each device number. Each can be set and reset by IOT.

2. Interrupt Scanner:
Oscillator with 5 bits counter scanning each device number in sequence for coincident Interrupt Enable and Completion.

3. Device no./SCAN no. Multiplexer:
Selects IOT device no. during IOT, otherwise SCAN Counter.

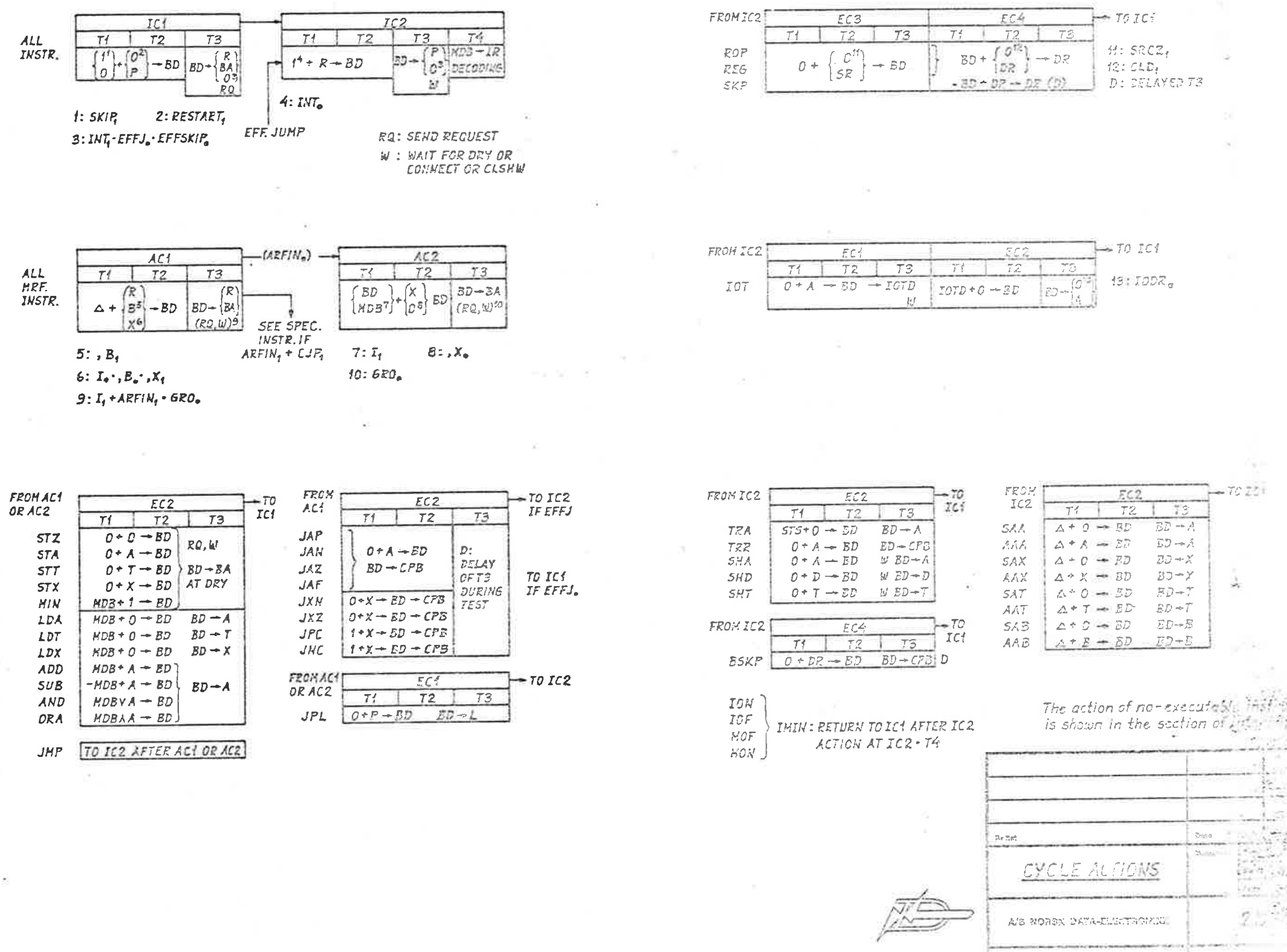
4. Device Completion Multiplexer:
Selects the Device Completion signal specified by the Device no./SCAN no. Multiplexer.

5. Interrupt Priority Tester:
A 5 bits Interrupt Priority Register to be set by program, and a comparator. Disables interrupt when SCAN Counter value has become equal to the Priority Register. ("Equal" state does not disable).

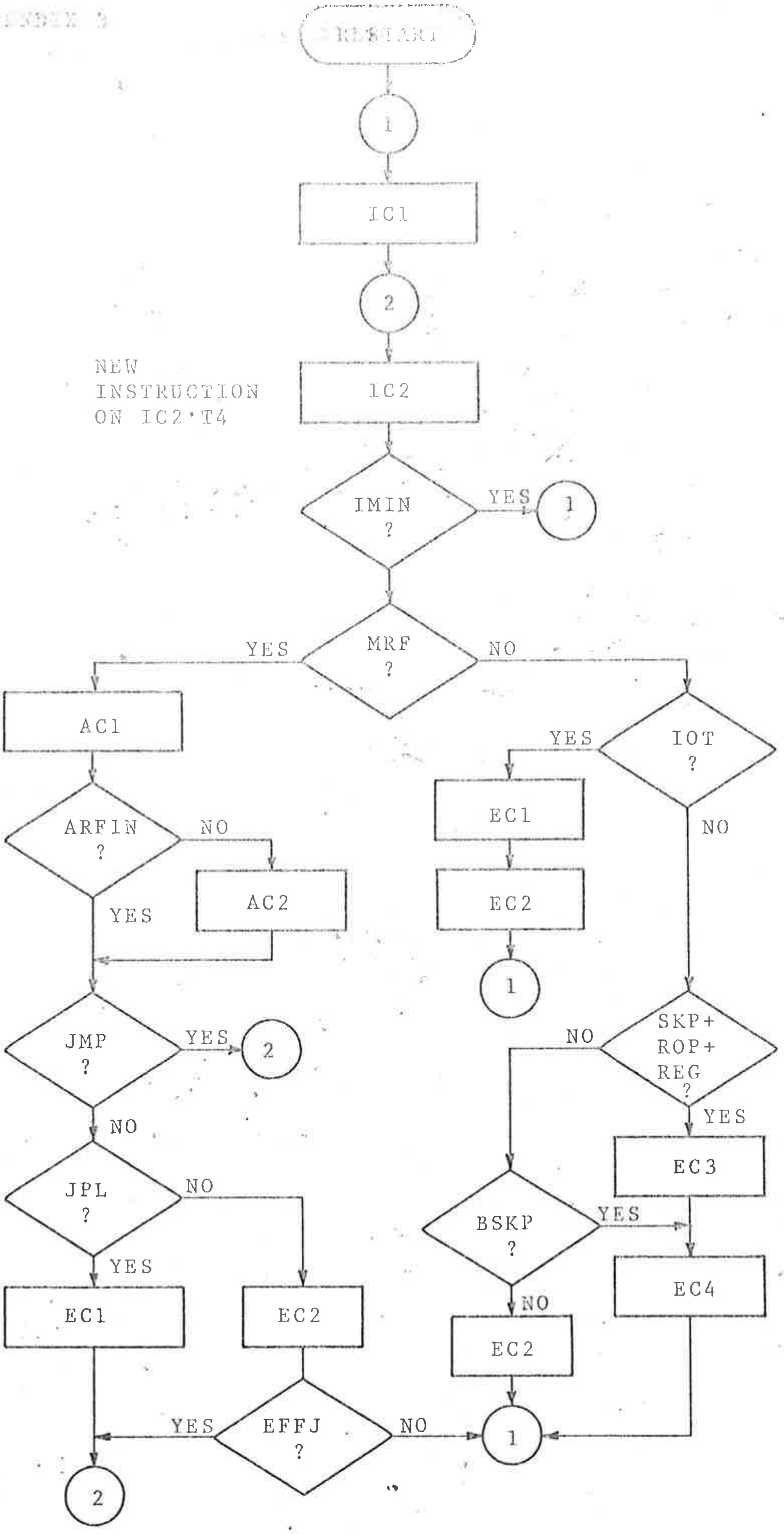
6. Interrupt Priority Encoder:
5 bits Interrupt Code Register which is set equal to the SCAN Counter on interrupt detection.

7. IOT Instruction Handler:
Synchronized IOT activity with the asynchronous SCAN and Completion activities. Replies CONN, IODRY, IOSKIP and resets INT appropriately.

8. Real Time Clock:
Is started by IOT ACT 29_{10}. Gives interrupt if IOT ACT PIN 29_{10}.


ALL INSTR.
IC1 T1 T2 T3
{1, 0} + {0, P} → BD BD → {R, BA, 0, RQ}
IC2 T1 T2 T3 T4
1 + R → BD BD → {P, 0, W} MDB → IR DECODING
4: INT₀
EFF. JUMP
1: SKIP₁ 2: RESTART₁
3: INT₁ · EFFJ₀ · EFFSKIP₀
RQ: SEND REQUEST
W: WAIT FOR DRY OR CONNECT OR CLSHW
ALL MRF. INSTR.
AC1 T1 T2 T3
Δ + {R, B, X} → BD BD → {R, BA, (RQ,W)}
(ARFIN₀)
AC2 T1 T2 T3
{BD, MDB} + {X, 0} BD BD → BA (RQ,W)
SEE SPEC. INSTR. IF ARFIN₁ + CJP₁
5: , B₁
6: I₀ · , B₀ · , X₁
9: I₁ + ARFIN₁ · GRO₀
7: I₁ 8: , X₀
10: GRO₀
FROM AC1 OR AC2
EC2 T1 T2 T3
TO IC1
STZ 0 + 0 → BD RQ, W
STA 0 + A → BD
STT 0 + T → BD BD → BA AT DRY
STX 0 + X → BD
MIN MDB + 1 → BD
LDA MDB + 0 → BD BD → A
LDT MDB + 0 → BD BD → T
LDX MDB + 0 → BD BD → X
ADD MDB + A → BD
SUB -MDB + A → BD BD → A
AND MDB ∨ A → BD
ORA MDB ∧ A → BD
JMP TO IC2 AFTER AC1 OR AC2
FROM AC1
EC2 T1 T2 T3
TO IC2 IF EFFJ
JAP JAN JAZ JAF 0 + A → BD BD → CPB
D: DELAY OF T3 DURING TEST
JXN 0 + X → BD → CPB
JXZ 0 + X → BD → CPB
JPC 1 + X → BD → CPB
JNC 1 + X → BD → CPB
TO IC1 IF EFFJ.
FROM AC1 OR AC2
EC1 T1 T2 T3
TO IC2
JPL 0 + P → BD BD → L
FROM IC2
EC3 T1 T2 T3
EC4 T1 T2 T3
TO IC1
ROP REG SKP 0 + {C, SR} → BD BD + {0, DR} → DR
- BD - DR → DR (D)
11: SRCZ₁
12: CLD₁
D: DELAYED T3
FROM IC2
EC1 T1 T2 T3
EC2 T1 T2 T3
TO IC1
IOT 0 + A → BD → IOTD W IOTD + 0 → BD BD → {0, A}
13: IODR₀
FROM IC2
EC2 T1 T2 T3
TO IC1
TRA STS + 0 → BD BD → A
TRR 0 + A → BD BD → CPB
SHA 0 + A → BD W BD → A
SHD 0 + D → BD W BD → D
SHT 0 + T → BD W BD → T
FROM IC2
EC4 T1 T2 T3
TO IC1
BSKP 0 + DR → BD BD → CPB D
FROM IC2
EC2 T1 T2 T3
TO IC1
SAA Δ + 0 → BD BD → A
AAA Δ + A → BD BD → A
SAX Δ + 0 → BD BD → X
AAX Δ + X → BD BD → X
SAT Δ + 0 → BD BD → T
AAT Δ + T → BD BD → T
SAB Δ + 0 → BD BD → B
AAB Δ + B → BD BD → B
ION IOF MOF MON
IMIN: RETURN TO IC1 AFTER IC2 ACTION AT IC2 · T4
The action of no-executable instructions is shown in the section of
CYCLE ACTIONS
A/S NORSK DATA-ELEKTRONIKK

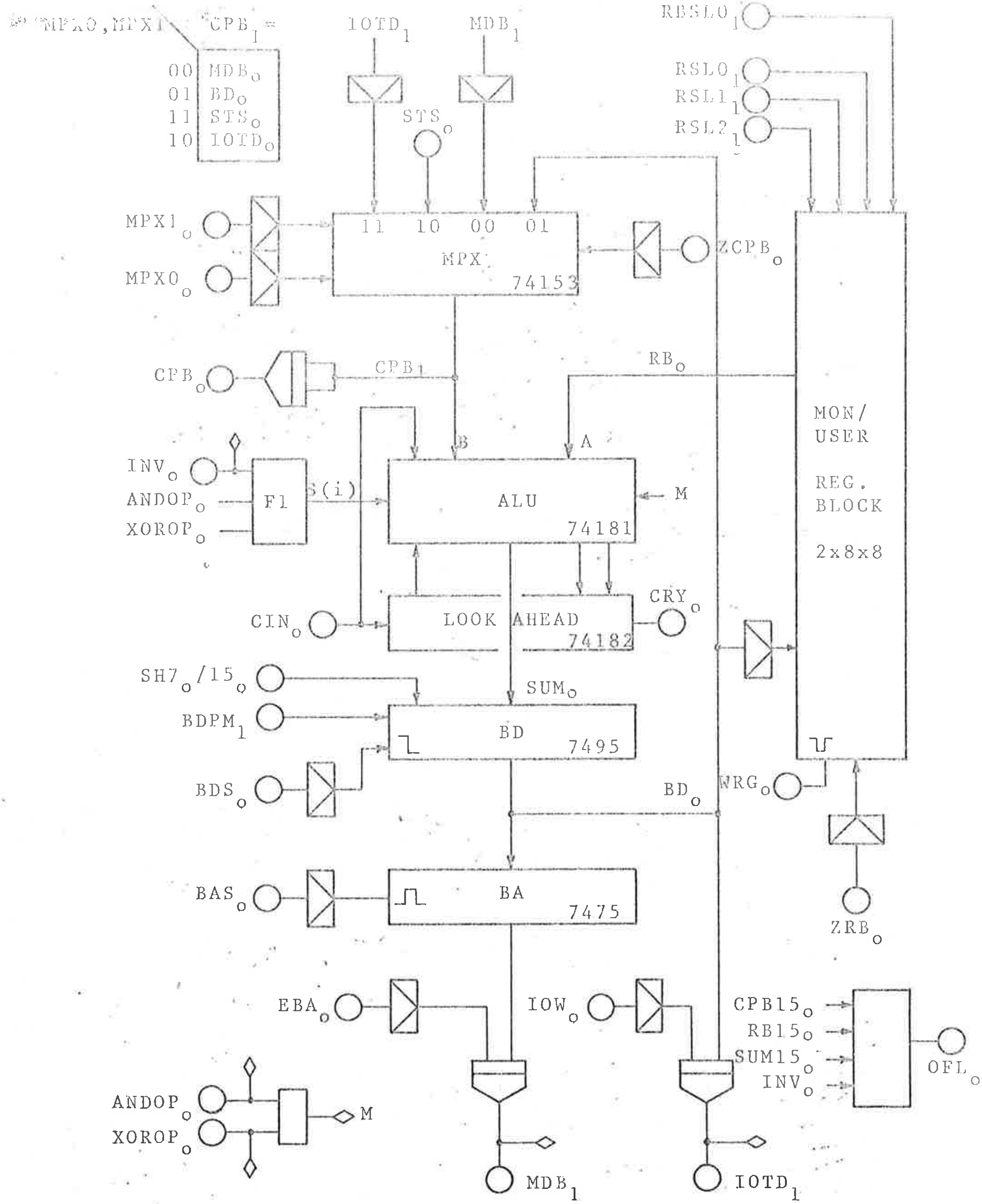
APPENDIX 3



CYCLE FLOW DIAGRAM

APPENDIX

REGISTER MANIPULATION



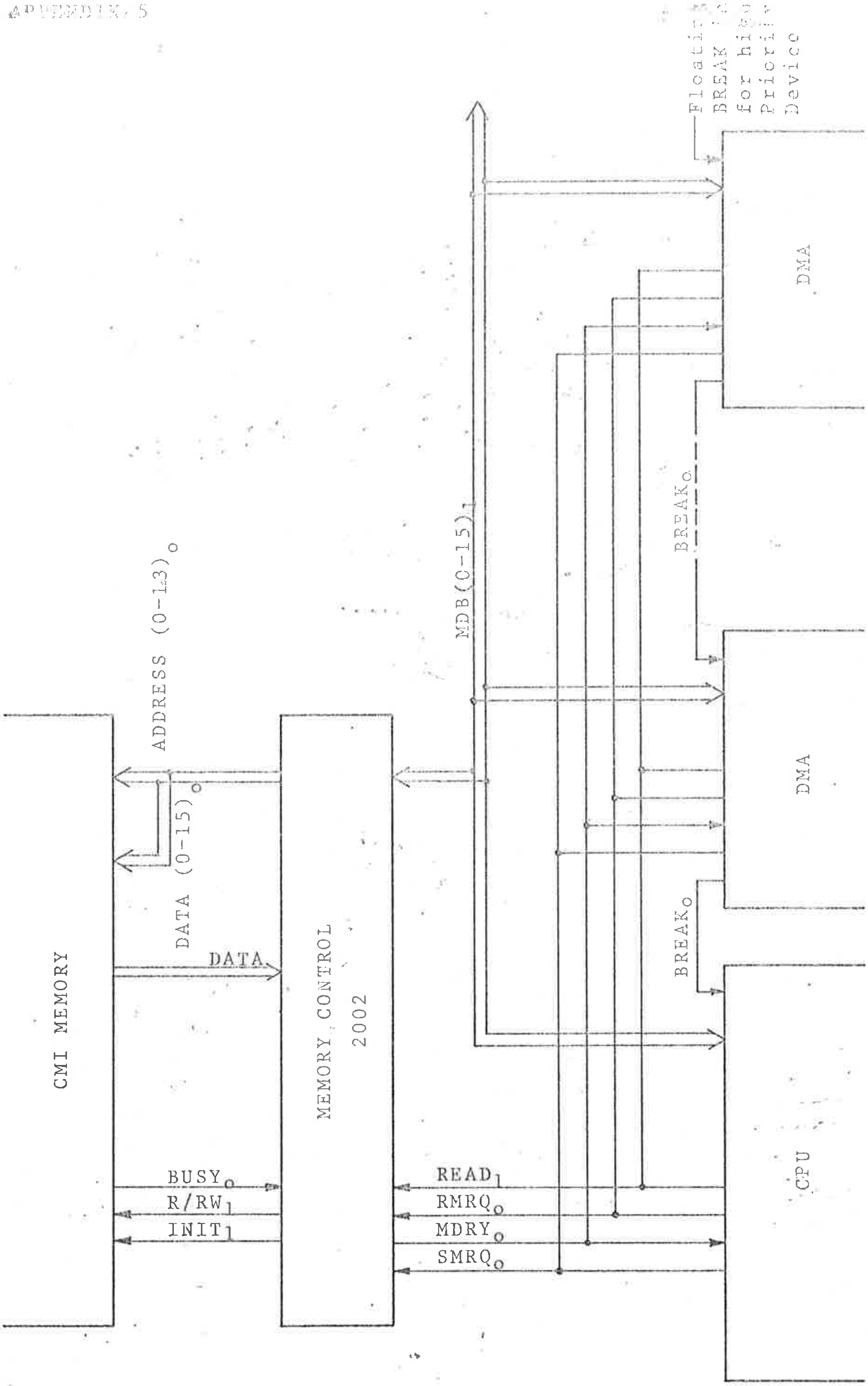
ANDOP, XOROP	F1 INV=0				F1 INV=1				
0 0	1	0	0	1	0	1	1	0	A PLUS B
0 1	1	0	0	1	0	1	1	0	A XOR B
1 0	1	1	1	0	1	1	0	1	A AND B
1 1	1	0	1	1	0	1	1	1	A OR B
S(i)	3	2	1	0	3	2	1	0	

OFL

RB15, CPB15	SUM15 INV=0: 0	1	INV=1: 0	1
0 0	0	1	0	0
0 1	0	0	0	1
1 0	0	0	1	0
1 1	1	0	0	0

NB: $CP3_1$ = "$OPER$"$_0$

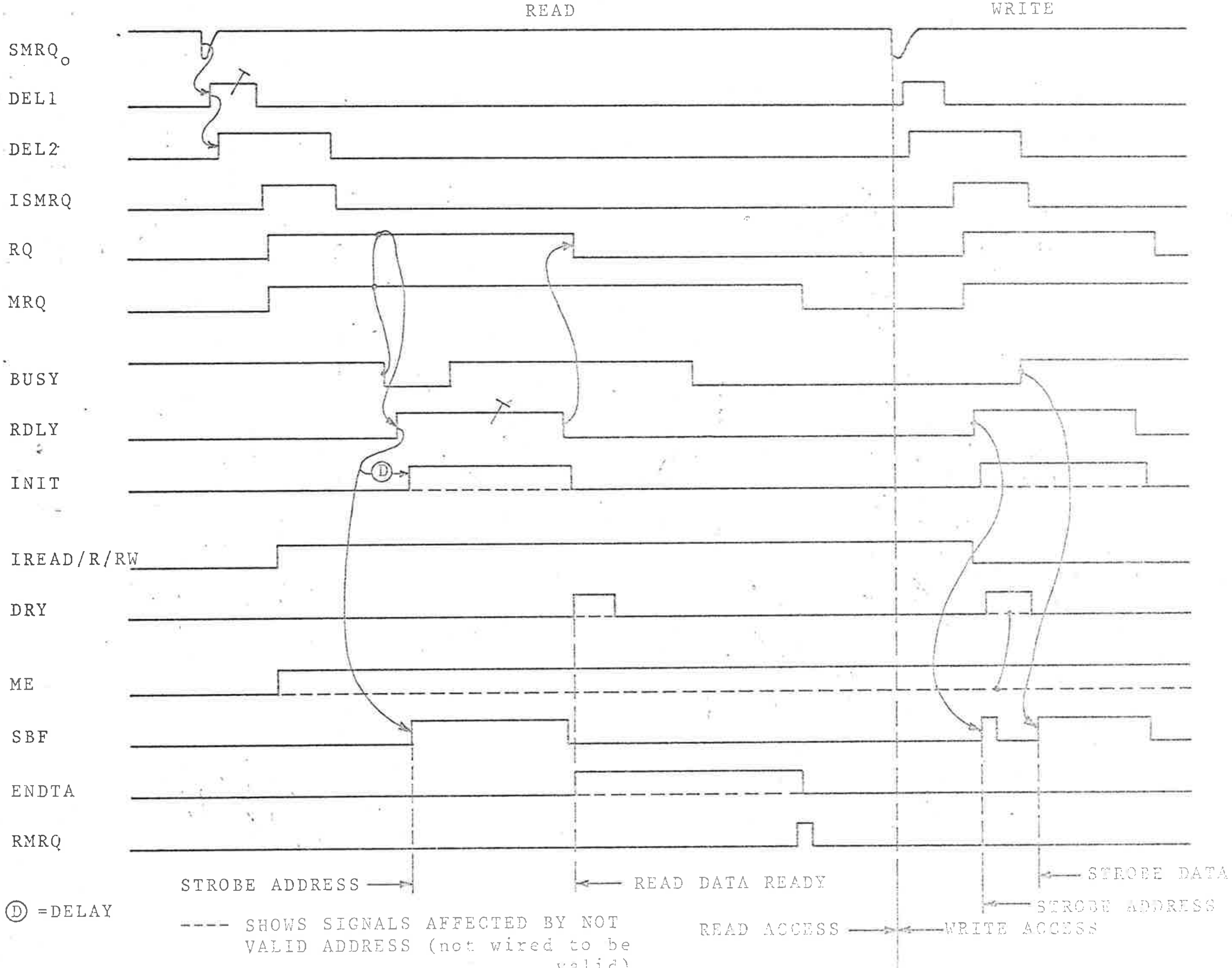
APPENDIX 5



MEMORY INTERFACING

APPENDIX 6

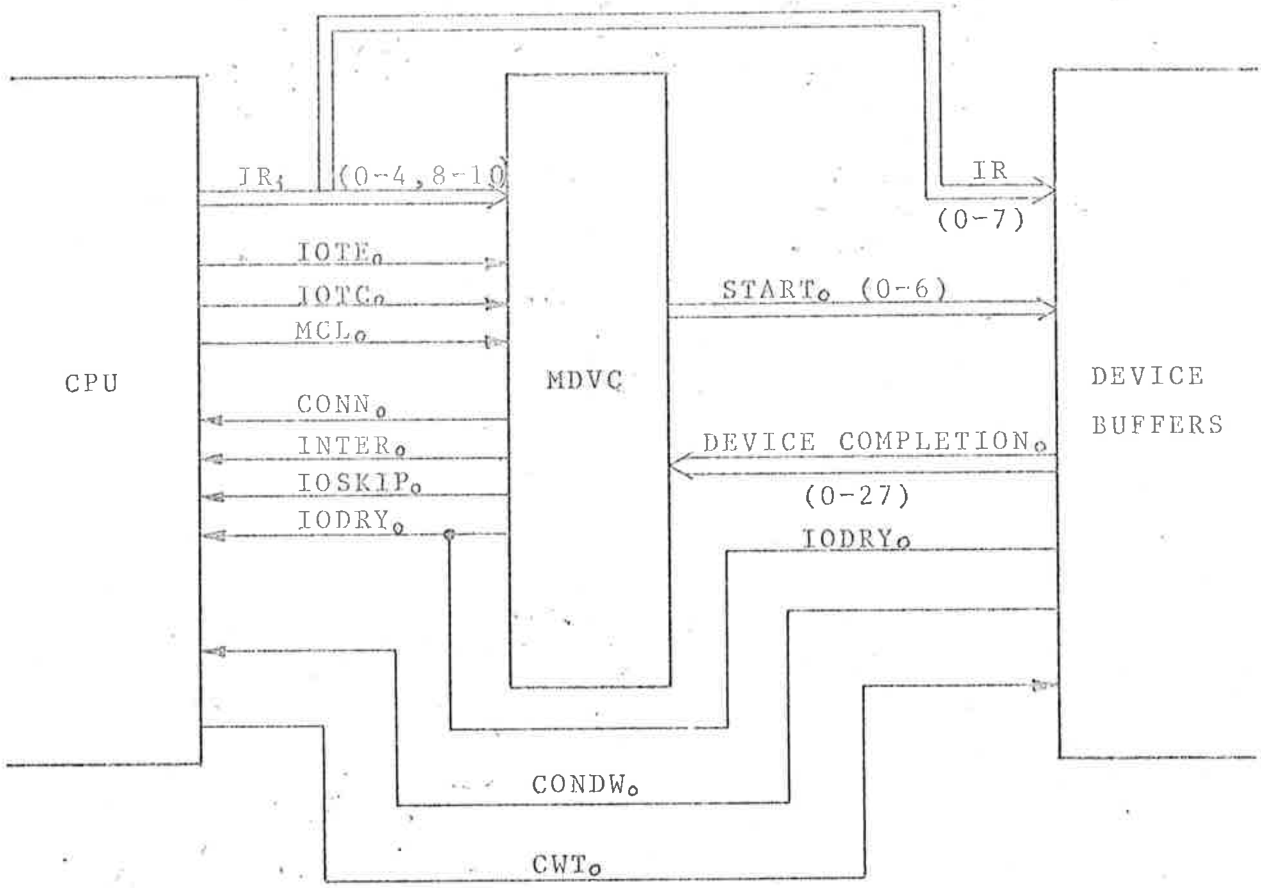
MEMORY CONTROL 2002 TIMING



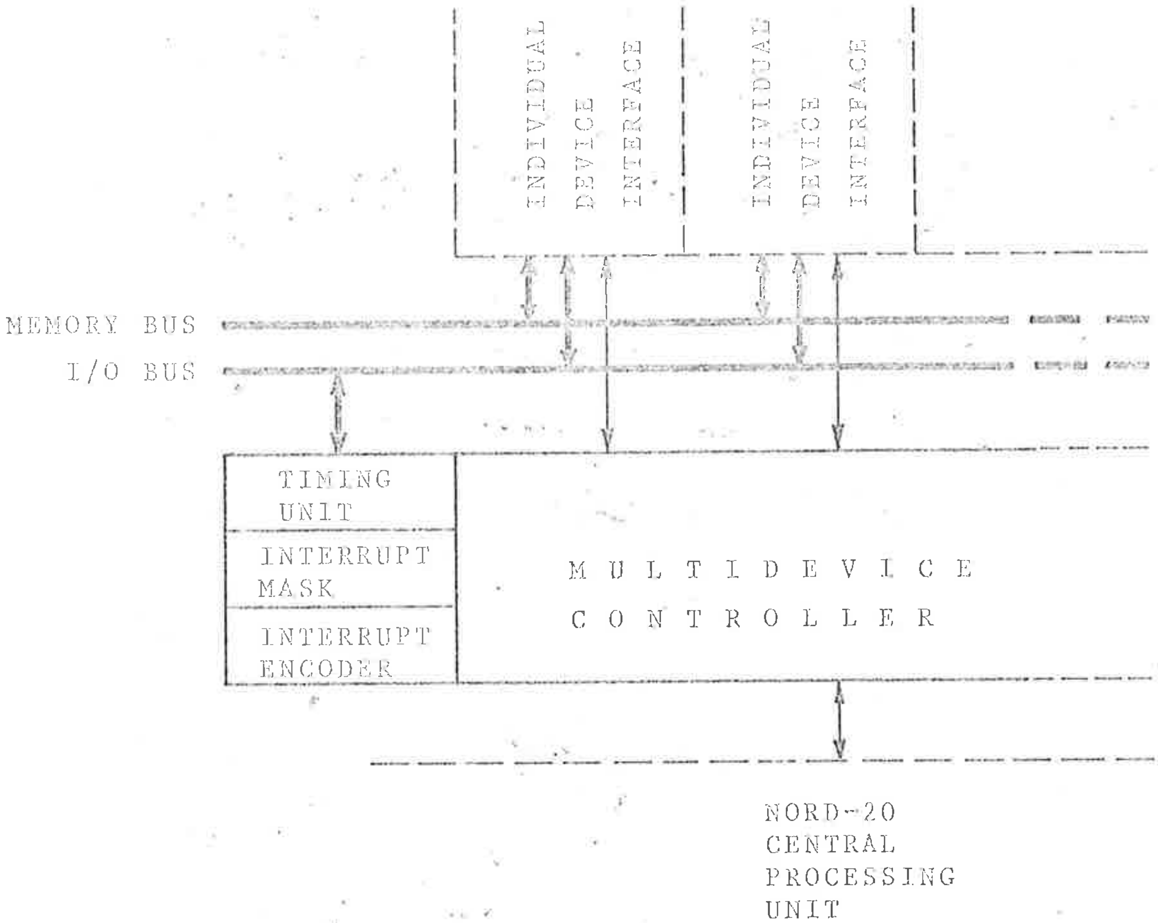
APPENDIX 7

M D V C

FUNCTIONAL DESCRIPTION



APPENDIX 7.3

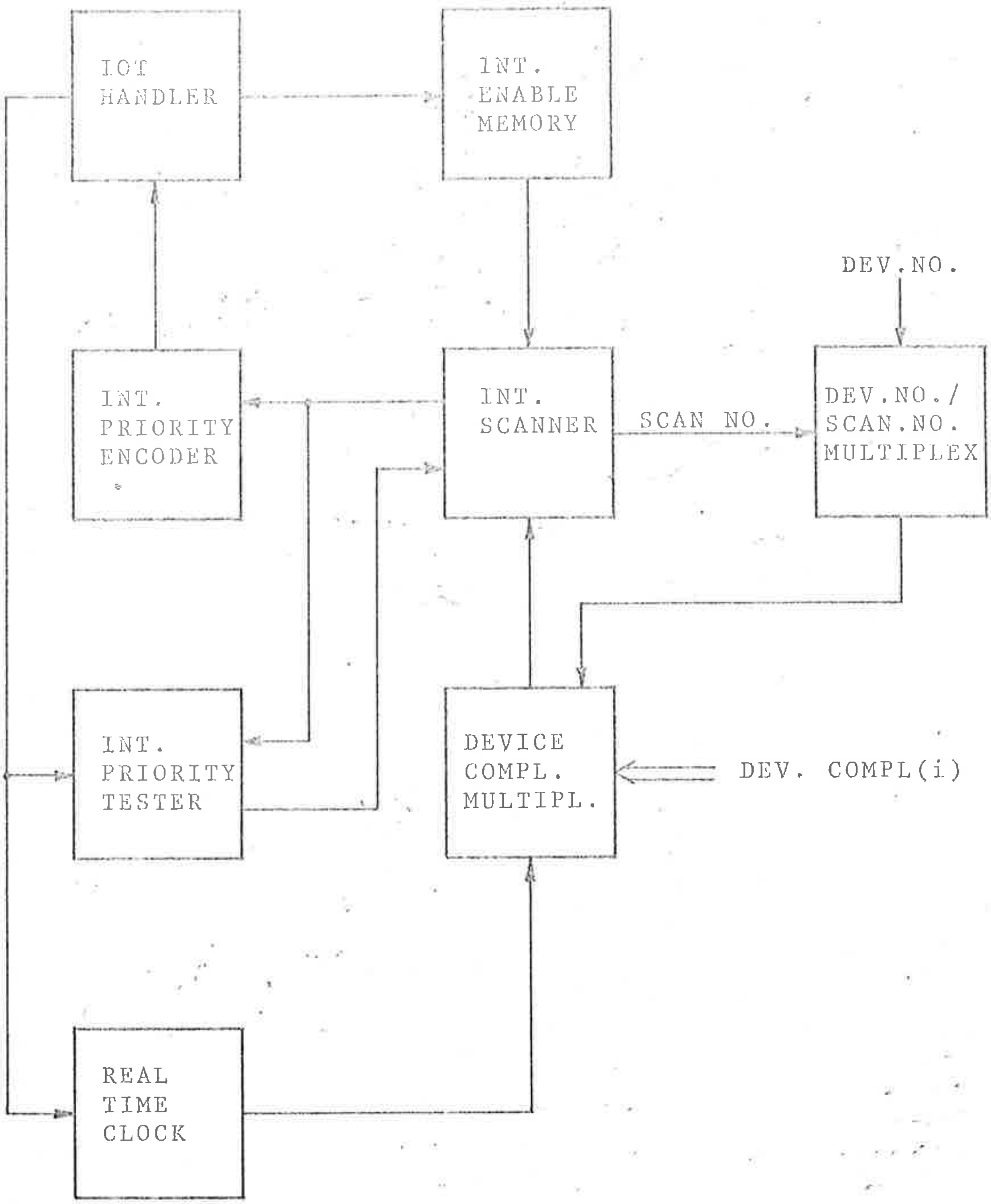


(NORD-20 INPUT/OUTPUT)

MULTI-DEVICE CONTROLLER (MDVC)

FUNCTIONAL DESCRIPTION

BLOCK DIAGRAM



MULTI-DEVICE CONTROLLER (MDVC)
SYMBOL DEFINITIONS

NAME	FUNCTION
BUSY	Flip-Flop sampling selected Device Completion status.
START(i)	Signal to Device Buffer initiating operation
RESET	Mode after Master Clear to reset all INTE(i)
CONN	Response to CPU on IOT for synchronizition
IODRY	Signal to CPU telling data are ready on IOT data bus for input.
IOSKIP	Signal during IOT telling CPU to skip next instruction.
INTER	Signal interrupting current isntruction execution in CPU and start new program.
COMPL(i)	Operation on Device (i) completed.
INTE(i)	Possible interrupt enabled for device (i).
DVB(i)	Device number bit (i). Equals IR(0,1,···4) during IOT and CNT (0,1,···4) during SCAN.
SPIN(i)	Set INTE(i).
RSPIN(i)	Reset INTE(i).
SCAN	Interrupt scan oscillator
PRIDS(i)	Interrupt priority disable register bit (i).

NAME	FUNCTION
IOSCAN	Sync. signal for interference IOT-SCAN.
RSCNT	Reset SCAN counter. Sets CNT=31.
WIEN	Write interrupt enable pusle.
PRIEX	Priority exceeded. Restart scan sequence.
SINTC	Set interrupt code register pulse.
INTC	Interrupt code register.
RTC	Real time clock oscillator.
SCANZ	CNT (0,1,2,3) = zero.
IOSCAND	IOSCAN delayed.
IOTE	IOT enable.
IOTC	IOT complete.
IOTD(i)	IOT data bus bit (i).
MCL	Master Clear.

MULTI-DEVICE CONTROLLER (MDVC)
EQUATIONS

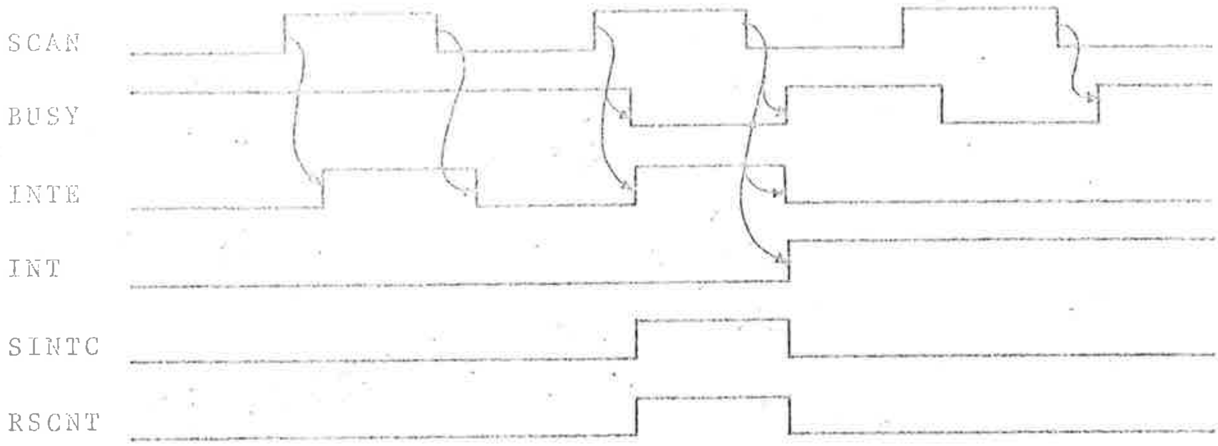
NAME	FUNCTION
DVB(j)	$IR(i)\cdot IOSCAN_1 + CNT(i)\cdot IOSCAN_0$
INTE	$INTE_1(i)\cdot (CONN_1 + SCAN_1)(0,1) + INTE_1\cdot (CONN_0\cdot SCAN_0)_0$
PIN(i)	$[INTE_1(i)\cdot WIEN(0,1) + SPIN(i) + PIN_1(i)\cdot RSPIN_0]\cdot RESET_0$
SPIN(i)	$(WIEN_1\cdot IR10_1)_1(i)$
RSPIN(i)	$(WIEN_1\cdot (SNI_1\cdot BUSY_0 + ACT_1\cdot PIN_0)_1(i)$
BUSY	$(CONN_1 + SCAN_1)(D0,1)\cdot COMPL_1(i) + BUSY_1\cdot (CONN_0\cdot SCAN_0\cdot BUSY_0)_0$
SCAN	$(MCL_1 + IOSCAN_1 + IOSCAND_1)_0\cdot SCAN_0$ (Delayed)
PRIDS(j)	$IOTD(j)\cdot SPRIDS(0,1)$
START(i)	$(CONN_1\cdot BUSY_0\cdot ACT_1\cdot IOTC_0)_1\cdot (i)$
INT	$SINTC_1\cdot SCAN(1,0) + INT_1\cdot (SNI_1\cdot WIEN_1 + RESET_1)_0$
CONN	$IOSCAND_1\cdot IOTE_1$
IOSCAN	$IOTE_1\cdot SCAN_0$
IOSCAND	$IOSCAN_1$ (Delayed $\sim$ 100 ns)
IODRY	START (30)
IOSKIP	$CONN_1\cdot (SNI_1\cdot (INTE_1\cdot BUSY_0)_0 + SKA_1\cdot BUSY_0)_1$

NAME	FUNCTION
RSCNT	$PRIEX_1 + MCL_1 + SINTC_1$
WIEN	$IOTE_1 \cdot IOTC_1 + RESET_1$
RESET	$MCL_1 + RESET_1 \cdot (SCANZ_1 \cdot SCAN_1 \cdot CNT4_o)_o$
PRIEX	$(CNT \hat{=} PRIDS)_1 \cdot SCAN(D\ 0,1) + PRIEX_1 \cdot (PRIEX_1 \cdot SCAN_o)_o$
SINTC	$INTE_1 \cdot BUSY_o \cdot RESET_o \cdot PRIEX_o$
INTC	$CNT(j) \cdot SINTC(D\ 0,1)$
RTC	$TIMEOSC(0,1) + RTC_1 \cdot START(29)_o$

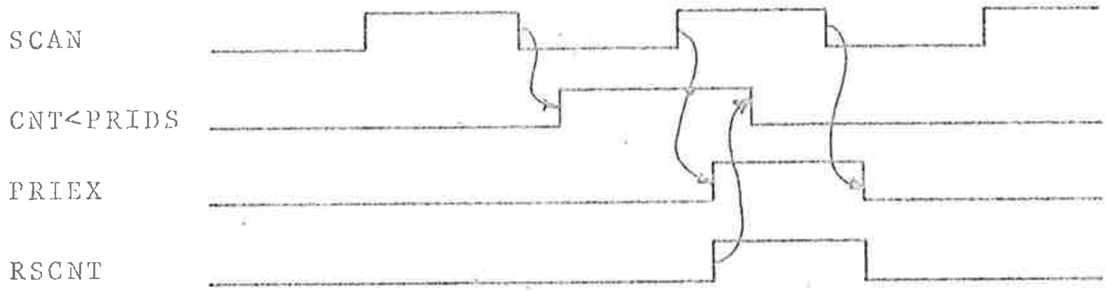
MULTI-DEVICE CONTROL

TIMING DIAGRAM

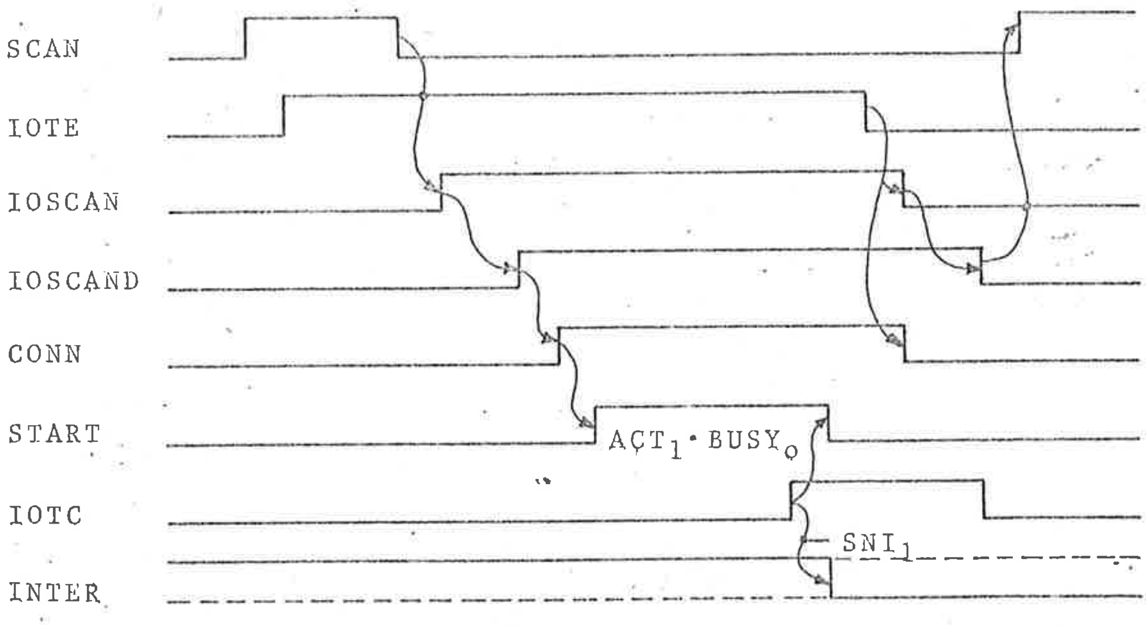
INTERRUPT SCAN



PRIORITY TEST



IOT SYNCHRONIZATION



APPENDIX C

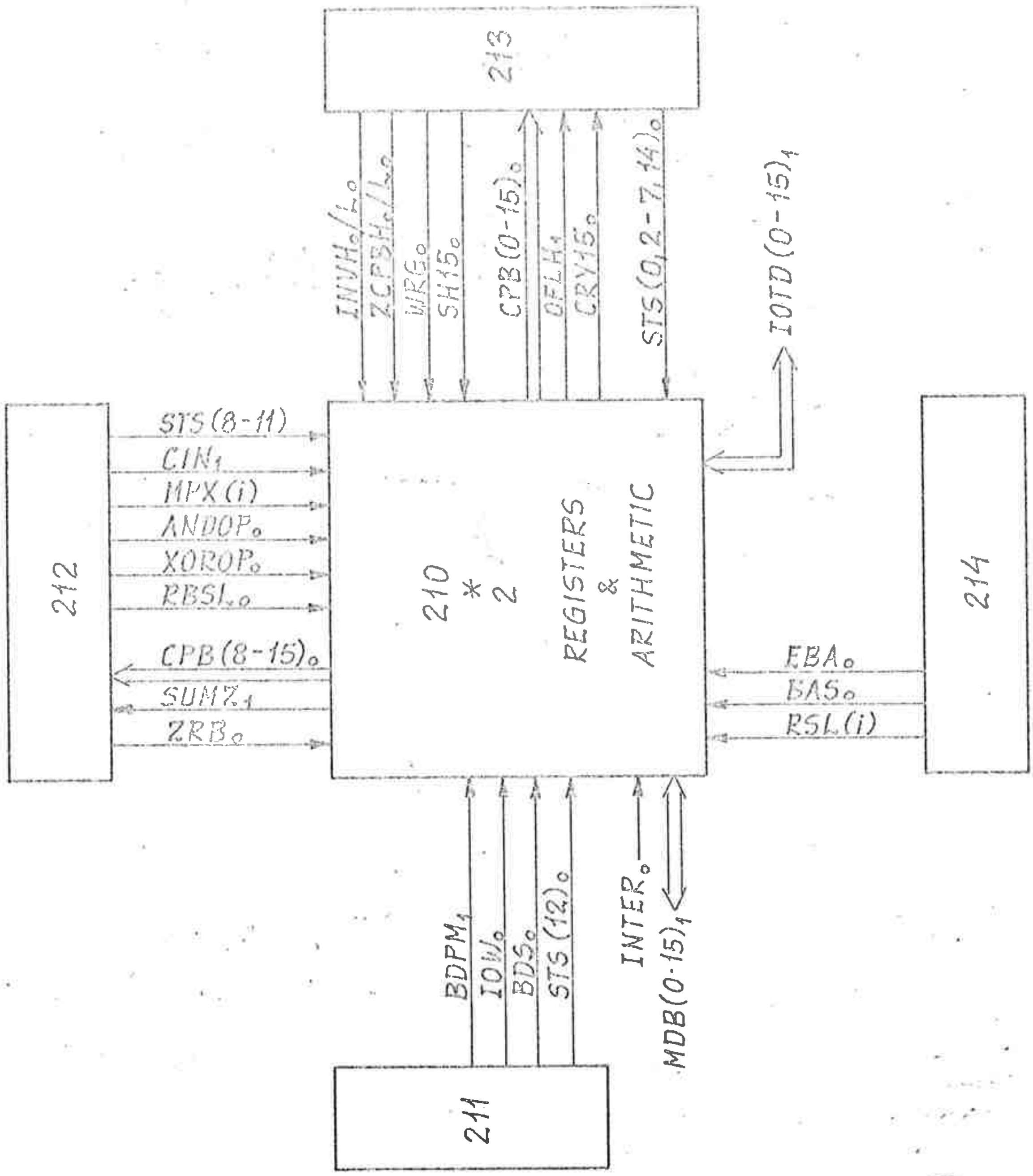
SIGNAL DEFINITIONS

ANDOP	Tells ALU to execute logical "AND".
ARFIN	Address computation Finished.
AU181	Inhibits reset of PMRQ.
BA(i)	Buffer Address register.
BAS	BA Set pulse.
BD(i)	Buffer Data register.
BDS	BD Set pulse.
BDPM	BD Parallel load Mode.
C	Carry flip-flop.
CC(i)	Cycle Counter.
CCS	CC Set pulse.
CIN	Carry In to arithmetic.
CL	Oscillator Clock pulse.
CLSHW	Clear "Shift Wait" pulse.
CLSTS	Clear Status register.
CONDB	Conditional Bit output true during BSKP.
CONDINST	Conditional Instruction, (MIN, SKP, IOT, BOP, CJP, MIS).
CONDJ	Conditional Jump effective.
CONDLY	Conditional Delay imposed by CONDINST.
CONDW	Conditional Wait response on CWT.
CONN	Connect signal from IOT interface.
CONND	CONN : Delayed.
CPB(i)	CPU internal data Bus.
CPBZ	CPB equals all Zero.
CRY7/15	Carry out from arithmetic bit 7 or 15.
CRYS	C Set pulse. Also used for Q and O.
DPMRQ	Decoded Processor Memory Request condition.
EADR	Enable CPU Address to MDB.
EBA	Enable BA to MDB.
EFFJ	Effective Jump condition true.
EFFJD	EFFJ Delayed.
ENRSTR	Enable Restricted mode. CPU will be interrupted on attempt to execute a priviledged instruction. (IOT, REG, ION, IOF, TRR).
IMIN	Immidiate Instruction. (Finished in IC2).

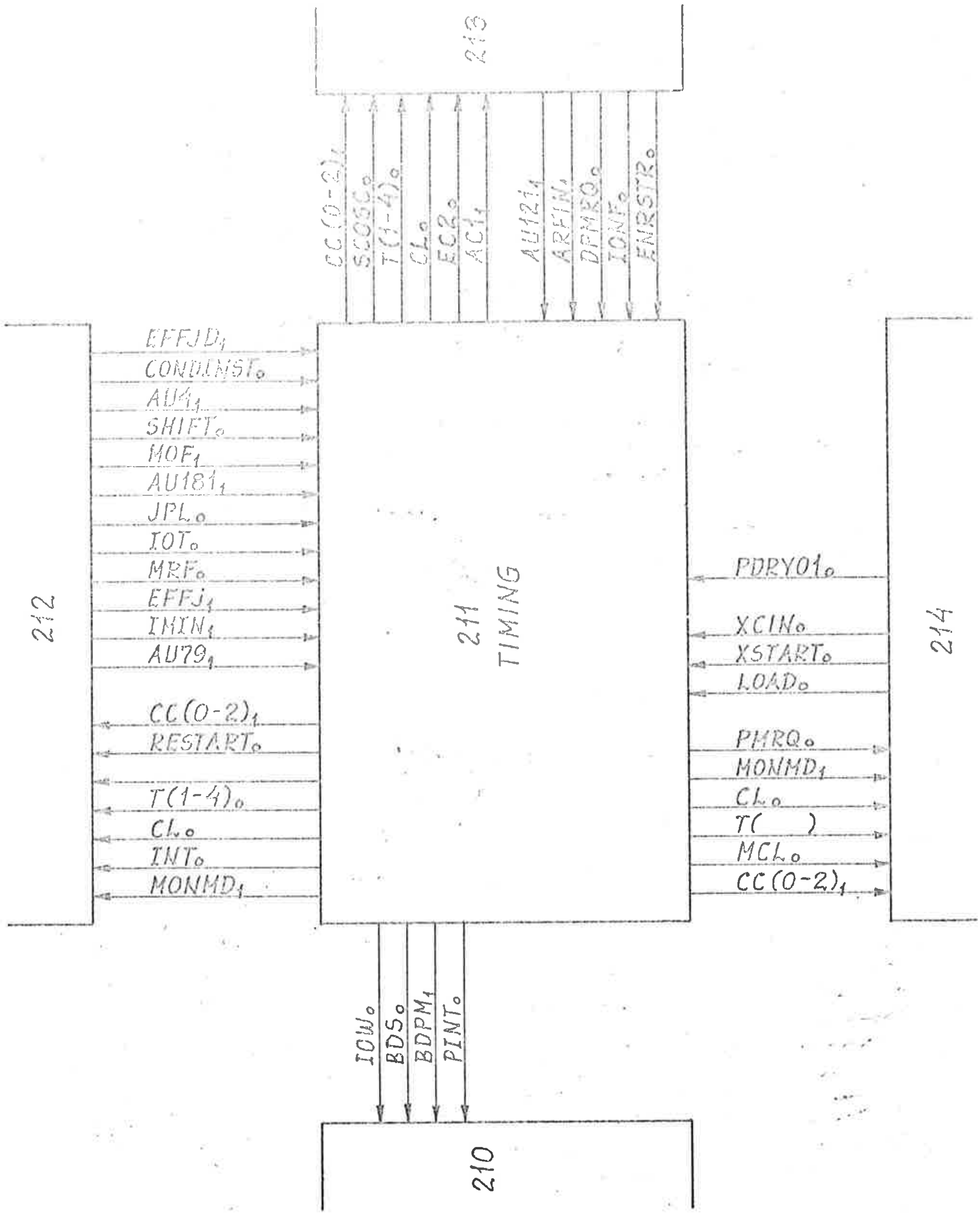
INT	CPU has discovered an interrupt condition. Will be reset on entring monitor mode.
INTER	IOT interface Interrupt signal.
INVH	Invert input to arithmetic bit (8-15).
INVL	" " " " " (0-7). Invert means use 1's complement of ALU input from CPB.
IODRY	IOT Data Ready signal from device to CPU.
IOSKIP	IOT Skip signal from MDVC to CPU.
IOTC	IOT Completion pulse from CPU to MDVC.
IOTD(i)	IOT Data bus.
IOTE	IOT Enable signal from CPU to MDVC defining when CPU control of IOT is valid.
IONF	I/O interrupt system is on flip-flop.
IOW	IOT Write enables BD to IOTD.
IR(i)	Instruction Register.
K	
LOAD	Load freezes CPU and initiates Restart and Master Clear.
M	Multi link flip-flop for shift operations.
MCL	Master Clear.
MDB(i)	Memory Data Bus.
MONMD	Monitor Mode flip-flop.
MPX	Multiplexerselect code for input to CPB.
MRF	Memory reference instruction.
MRQ	Common Memory Request signal.
O	Overflow (static) flip-flop.
OFLH	Arithmetic Overflow indication.
OPWT	Special Wait during the operations Shift or IOT.
PDRY	Processor Data Ready from memory.
PDRY01	PDRY has changed from "0" to "1" after PMRQ was sent.
PINT	Program Interrupt caused by instruction.
PMRQ	Processor Memory Request.
PSMRQ	Processor Set MRQ pulse.
Q	Dynamic overflow condition flip-flop.

RAD1	Add 1 during ROP or REG in addition to contents of Source and Destination.
RB(i)	Register block Bus.
READ	Read command to memory.
RESTART	Generates a MCL pulse and resets Program counter to 0.
RMRQ	Reset MRQ pulse.
RBSL(i)	Register Block Select code.
RSL(i)	Register Select code.
RSTR	Restricted mode flip-flop.
SCZ	Shift Counter equals Zero.
SCOSC	Shift Oscillator pulse.
SMRQ	Set MRQ pulse.
SH15	Serial input to BD(15) during Shift.
SKIPF	Skip flip-flop holding skip information until it is used by the next I-fetch.
SSTS	Set Status register pulse.
SUM(i)	ALU output.
SUMZ	ALU output equals Zero.
T(i)	Time counter output.
T4DLY	Additional decoding delay imposed during T4.
WAIT	Signal causing clock oscillator to Wait.
WRG	Write pulse to Register block.
XCIN	Executable Instruction as opposed to one causing program interrupt.
XOROP	Tells ALU to execute "exclusive or".
Z	Floating point overflow indicator.
ZCPBL	Force low half of CPB to Zero.
ZCPBH	" high " " " " "
ZRB	Force RB bus to Zero.

APPENDIX 9

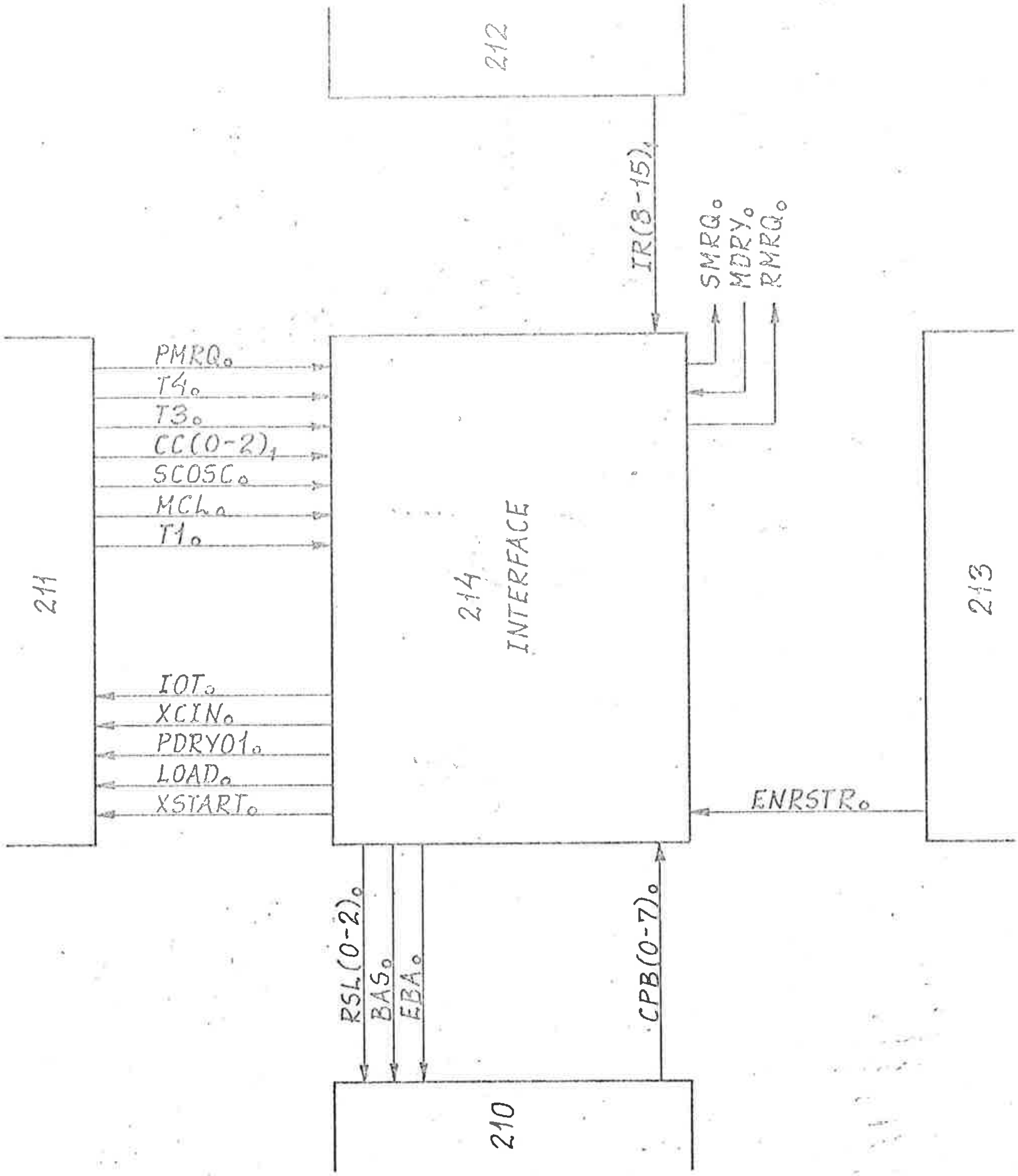


DRAWN BY Emil	Remarks	Replacement for	Date
APPROVED BY E.Ø.	BLOCK INTERCONNECTIONS	Replaced by	Date
DATE 16.2.72			

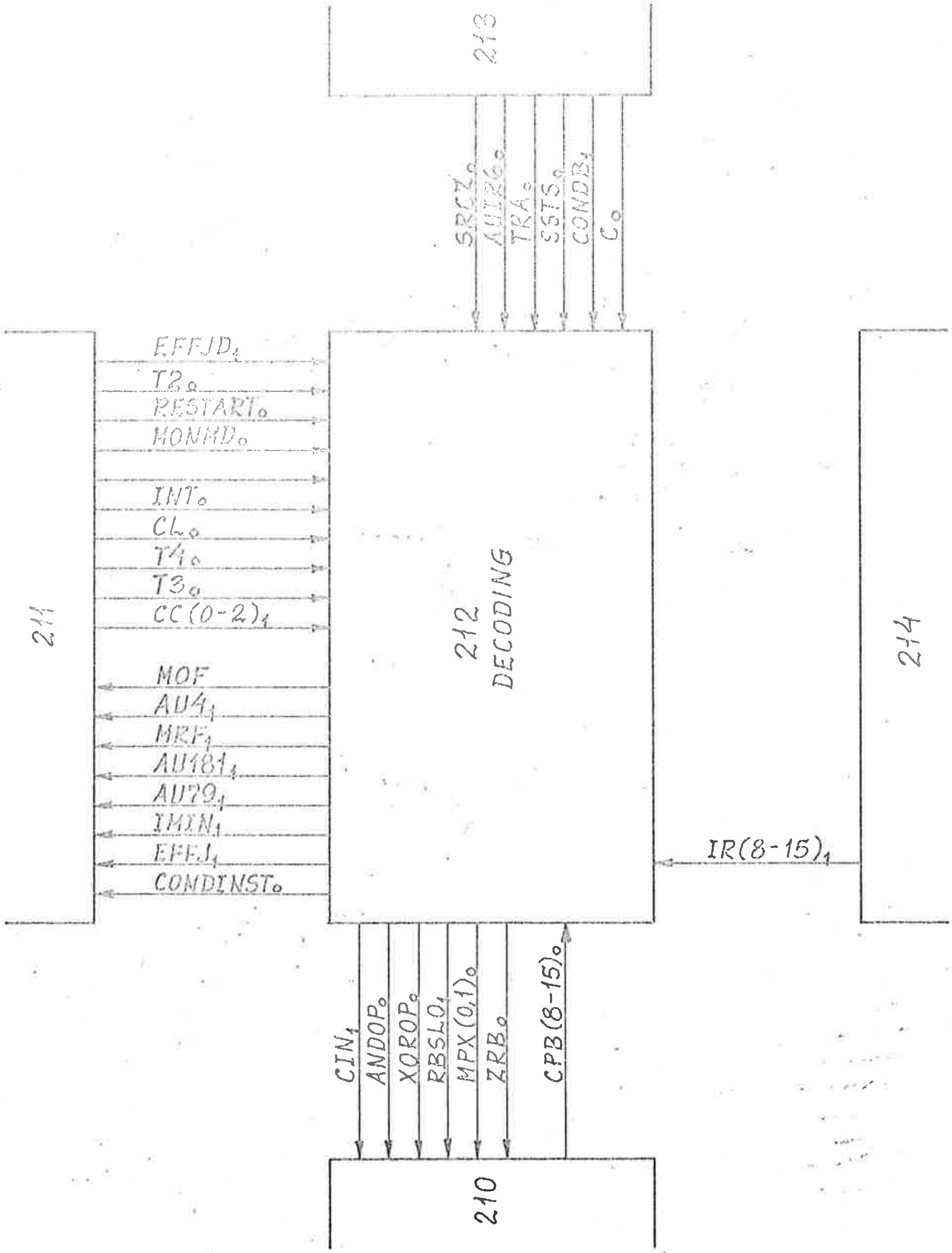


DRAWN BY Emil	Remarks	Replacement for	Date
APPROVED BY E.Ø.		Replaced by	Date
DATE 16.2.72			

Drawing no

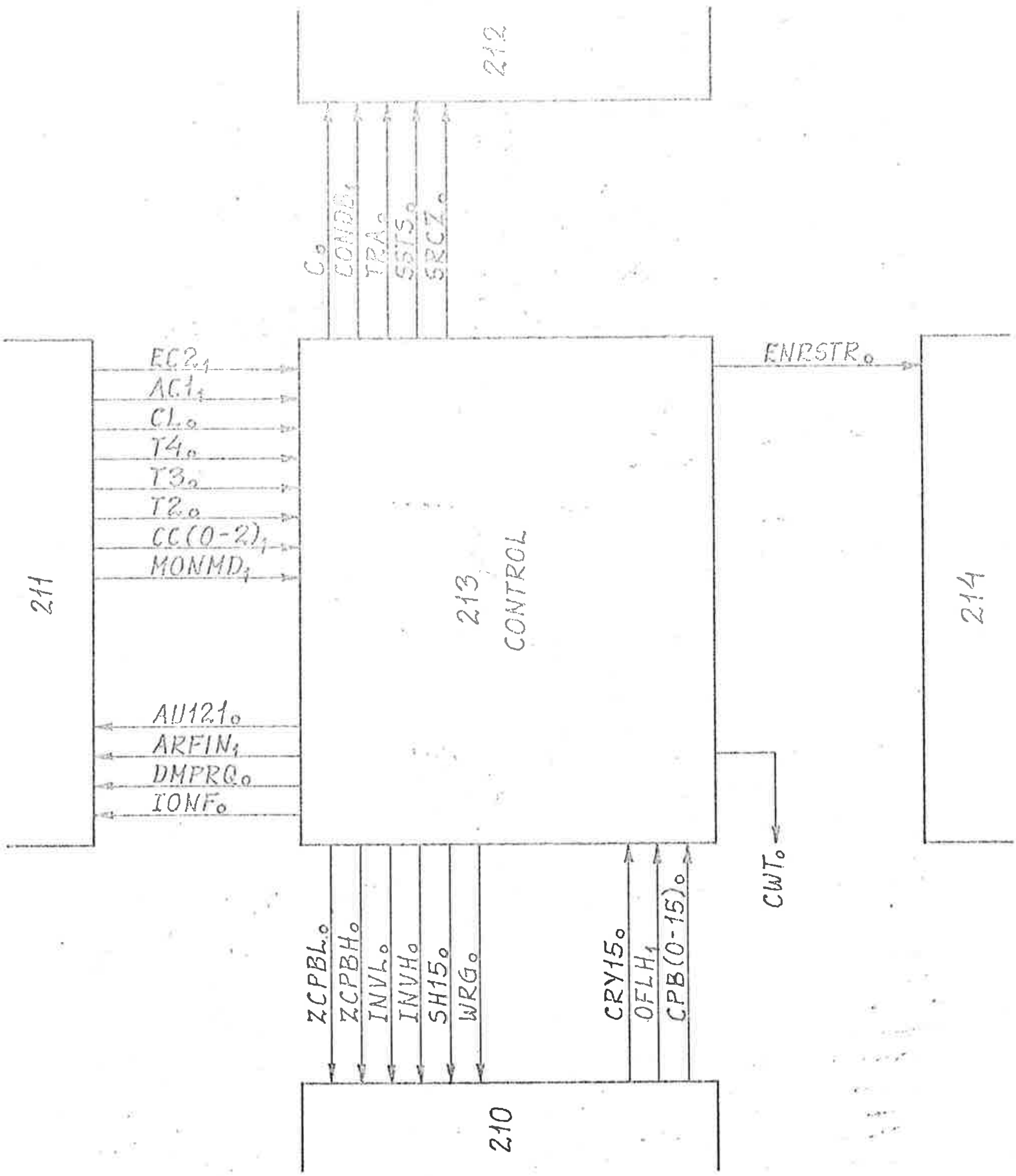


DRAWN BY	Emil	Remarks	Replacement for	Date
APPROVED BY	E.Ø.		Replaced by	Date
DATE	16.2.72			



DRAWN BY Emil	Remarks	Replacement for	Date
APPROVED BY . E.Ø.			
DATE 16.2.72		Replaced by	Date

A/S NORSK DATA-ELEKTRONIKK	CONTROL 213	



DRAWN BY	Eeul	Remarks	Replacement for	Date
APPROVED BY	E.Ø.		Replaced by	Date
DATE	16.2.72			

			15	14	13	12	11	10 9 8 7 6 5 4 3 2 1 0
0	000 000	STZ	0	0	0	0	0	
	004 000	STA	0	0	0	0	1	
	010 000	STT	0	0	0	1	0	
	014 000	STX	0	0	0	1	1	
1	020 000	FREE	0	0	1	0	0	
	024 000	FREE	0	0	1	0	1	
	030 000	FREE	0	0	1	1	0	
	034 000	FREE	0	0	1	1	1	
2	040 000	MIN	0	1	0	0	0	
	044 000	LDA	0	1	0	0	1	XIB | Displacement △
	050 000	LDT	0	1	0	1	0	
	054 000	LDX	0	1	0	1	1	
3	060 000	ADD	0	1	1	0	0	
	064 000	SUB	0	1	1	0	1	
	070 000	AND	0	1	1	1	0	
	074 000	ORA	0	1	1	1	1	
4	100 000	FREE	1	0	0	0	0	
	104 000	FREE	1	0	0	0	1	
	110 000	FREE	1	0	0	1	0	
	114 000	FREE	1	0	0	1	1	
5	120 000	FREE	1	0	1	0	0	
	124 000	JMP	1	0	1	0	1	
	130 000	CJP	1	0	1	1	0	Sub in.
	134 000	JPL	1	0	1	1	1	
6	140 000	SKP	1	1	0	0	0	NOT | GRE | | S | D
	144 000	ROP x)	1	1	0	0	1	RAD | ADC | AD1 | CM1 | CLD | S | D
	150 000	MIS x)	1	1	0	1	0	Sub instructions
	154 000	SHT x)	1	1	0	1	1	ZIN | ROT | SHA | SHD | Right count
7	160 000	IOT	1	1	1	0	0	PIN | SKA | ACT | Device number
	164 000	REG	1	1	1	0	1	RAD | SM | AD1 | CM1 | CLD | S | D
	170 000	ARG	1	1	1	1	0	Sub in. | Argument
	174 000	BOP x)	1	1	1	1	1	Sub in. | Bit no. | D
			100 000	40 000	20 000	10 000	4 000	2 000, 1 000, 400, 200, 100, 40, 20, 10, 4, 2, 1

x) Partially implemented NORD-1 instructions

NORD-20 INSTRUCTION CODE

APPENDIX 4

NORD-20 SUB-INSTRUCTIONS

MIS:	MON	- 153000
	MOF	- 153000
	TRA STS	- 150001
	TRR STS	- 150101
	TRR MPR	- 150103
	ION	- 150500
	IOF	- 150400
	CWT	- 153400 + No. (0-377)
REG:	REG	- 164000
	SM	- 001000
	SU	- 000000
	RAD, AD1, CM1, CLD, S, D: SEE ROP	

STS:	Bit	0 - IONF	- Interrupt enable	
	"	1 - INTER	- I/O Interrupt line	
	"	2 - K	- ONE BIT ACCUMULATOR	
	"	3 - Z	- Floating point overflow)[1]	
	"	4 - Q	- Dynamic overflow	
	"	5 - O	- Static overflow	
	"	6 - C	- Carry indicator	
	"	7 - M	- Multishift link	
	"	8-11 - PIL	- Inter. level indicator)[1]	
	"	12 - PINT	- Program interrupt flag	
	"	13 - MPRU	- Mem. Prot. violation flag	
	"	14 - RSTR	- Restricted mode	
	"	15 - PROTON	- Mem. Prot. system on	

)[1] Dummy flip-flops, may be set or read by TRR OR TRA STS

[illegible]

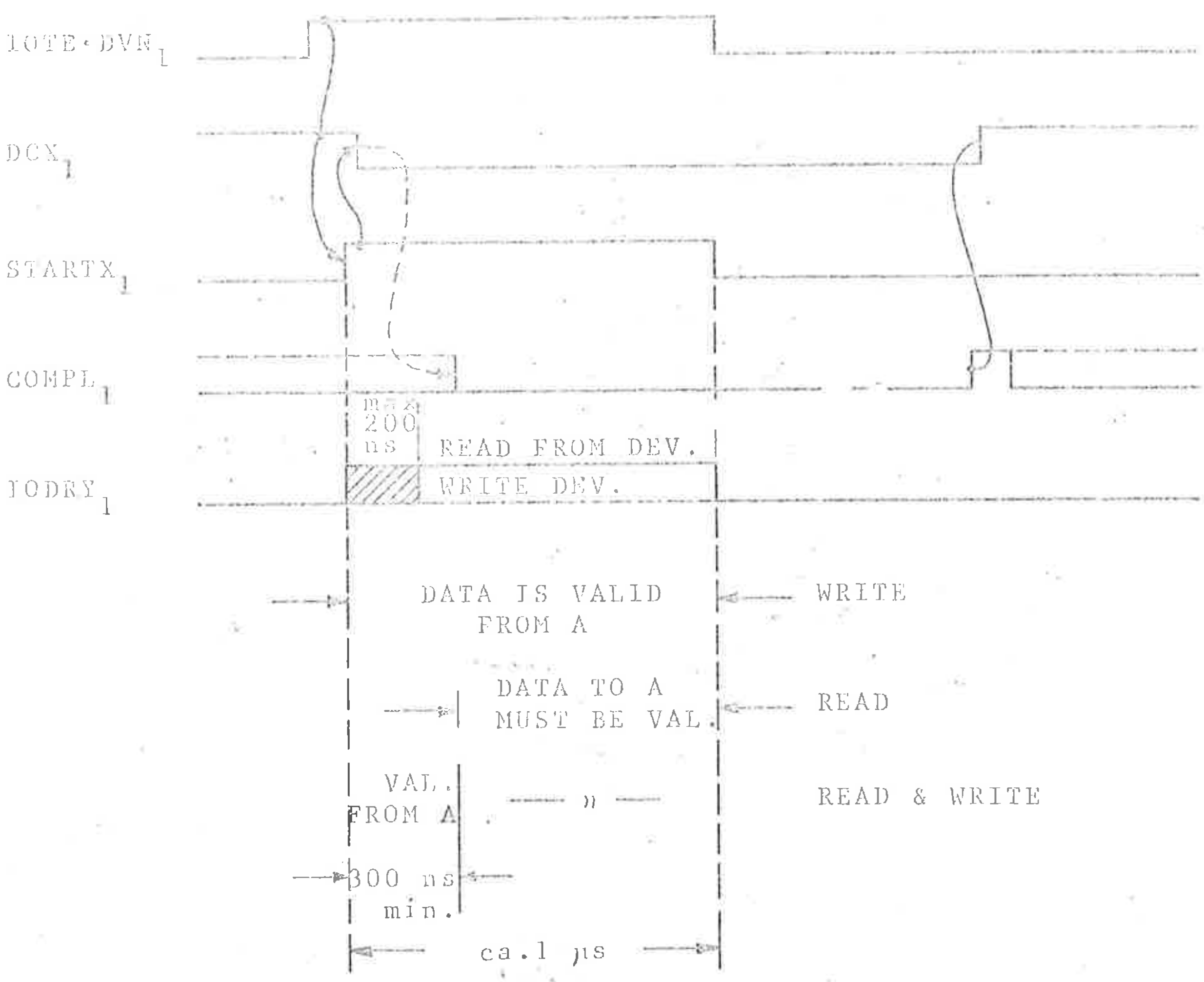
[illegible]

T	G	NAME	P	T	G	NAME	P
1		+5V		31	W	IOTD14	B
2		GROUND		32	W	MDB14	1
3	W	IOTD0	B	33	W	IOTD15	B
4	W	MDB0	1	34	W	MDB15	1
5	W	IOTD1	B	35		JR0	1
6	W	MDB1	1	36		JR1	1
7	W	IOTD2	B	37		JR2	1
8	W	MDB2	1	38		JR3	1
9	W	IOTD3	B	39		JR4	1
10	W	MDB3	1	40		JR5	1
11	W	IOTD4	B	41		JR6	1
12	W	MDB4	1	42		JR7	1
13	W	IOTD5	B	43	W	IODRY	0
14	W	MDB5	1	44	W	RMRQ	0
15	W	IOTD6	B	45	G	DCX3	0
16	W	MDB6	1	46		IBREAK	0
17	W	IOTD7	B	47	G	DCX2	0
18	W	MDB7	1	48	G	OBREAK	0
19	W	IOTD8	B	49	G	DCX1	0
20	W	MDB8	1	50	W	READ	1
21	W	IOTD9	B	51	G	DCX0	0
22	W	MDB9	1	52	W	SMRQ	0
23	W	IOTD10	B	53		STARTX	0
24	W	MDB10	1	54		DRY	0
25	W	IOTD11	B	55	W	CONDW	0
26	W	MDB11	1	56	W	LOAD	0
27	W	IOTD12	B	57		CWT	0
28	W	MDB12	1	58		MCL	0
29	W	IOTD13	B	59		+5V	
30	W	MDB13	1	60		GROUND	

T: Terminal no. on BURNDY PLUG

G: W= Wired-or line, G= Output

P: Polarity B= 1 for output 0 for input



DCX: Device-completion flag, ready signal to "MDVC". No STARTX will be sent from MDVC if DCX is off.

STARTX: General start signal to the four device numbers in one position. The special start must be decoded from IR0 and IR1 and strobed by STARTX. STARTX·DVN(0,1) usually sets DCX off.

COMPL: Operation completed (i.e. Dev.Ready) from device. Usually sets DCX on.

IODRY: I/O Data Ready from dev. must be present within 200 ns after STARTX and must be off when STARTX is off.

DRAWN BY IB/eml	Remarks	Replacement for	Date
APPROVED BY		Replaced by	Date
DATE 3/3/72			

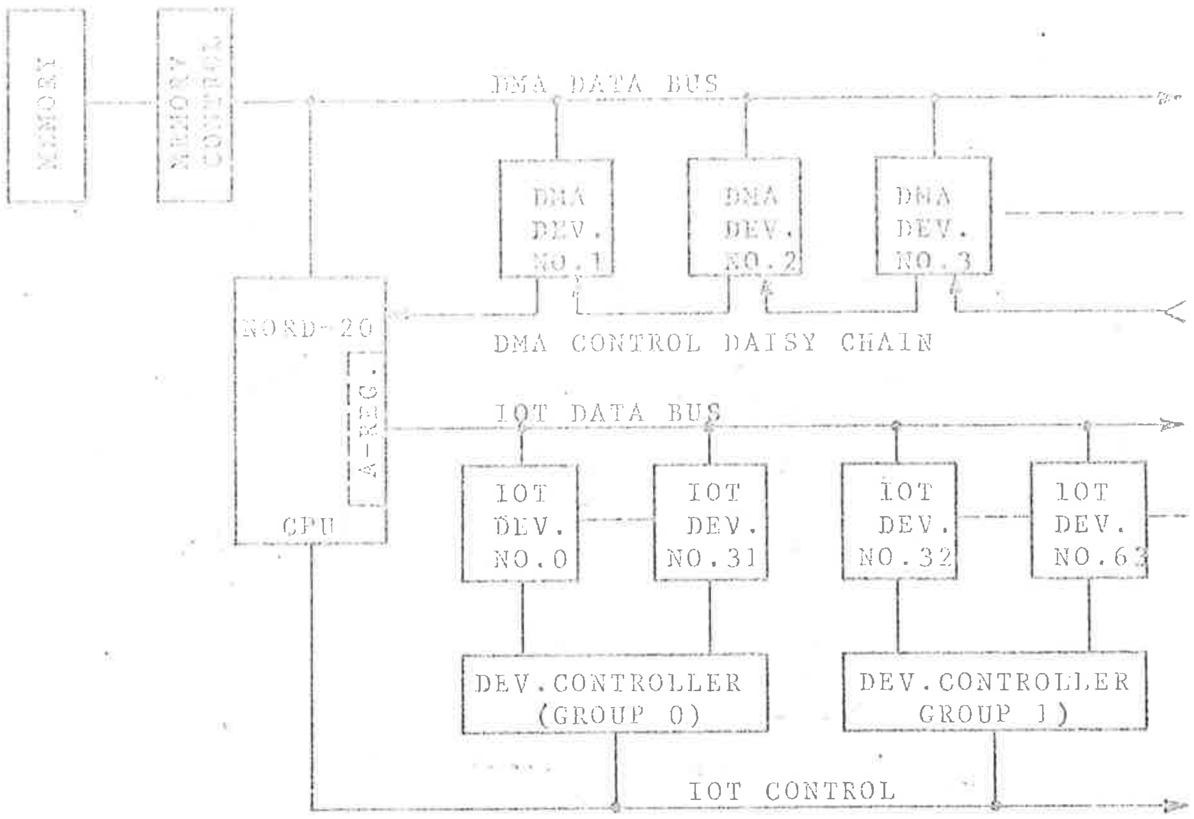


Figure 1.1

INPUT/OUTPUT BLOCK DIAGRAM

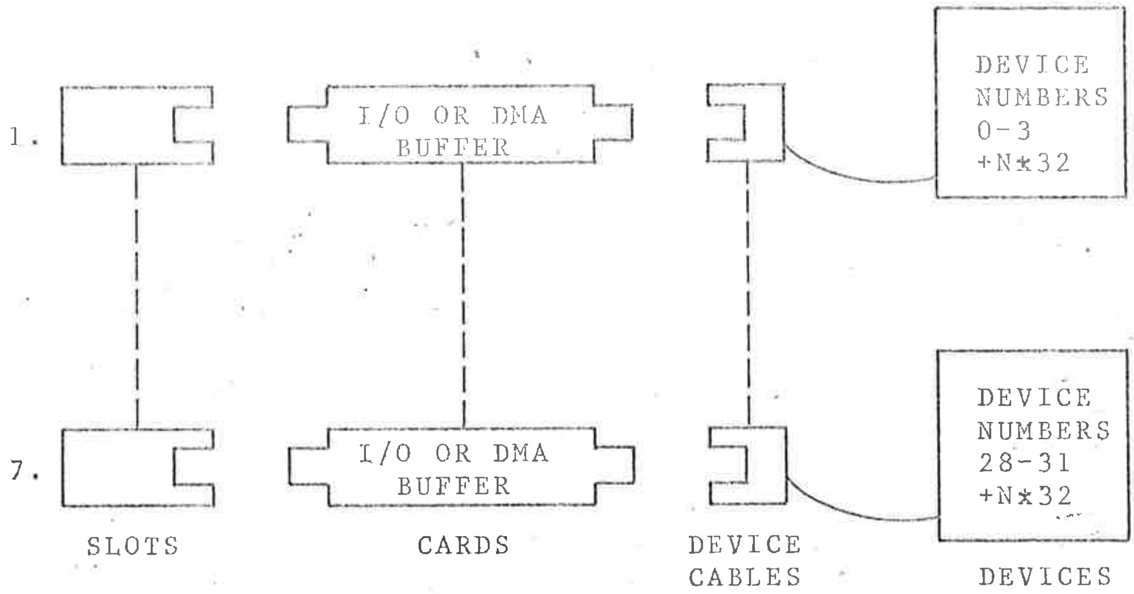
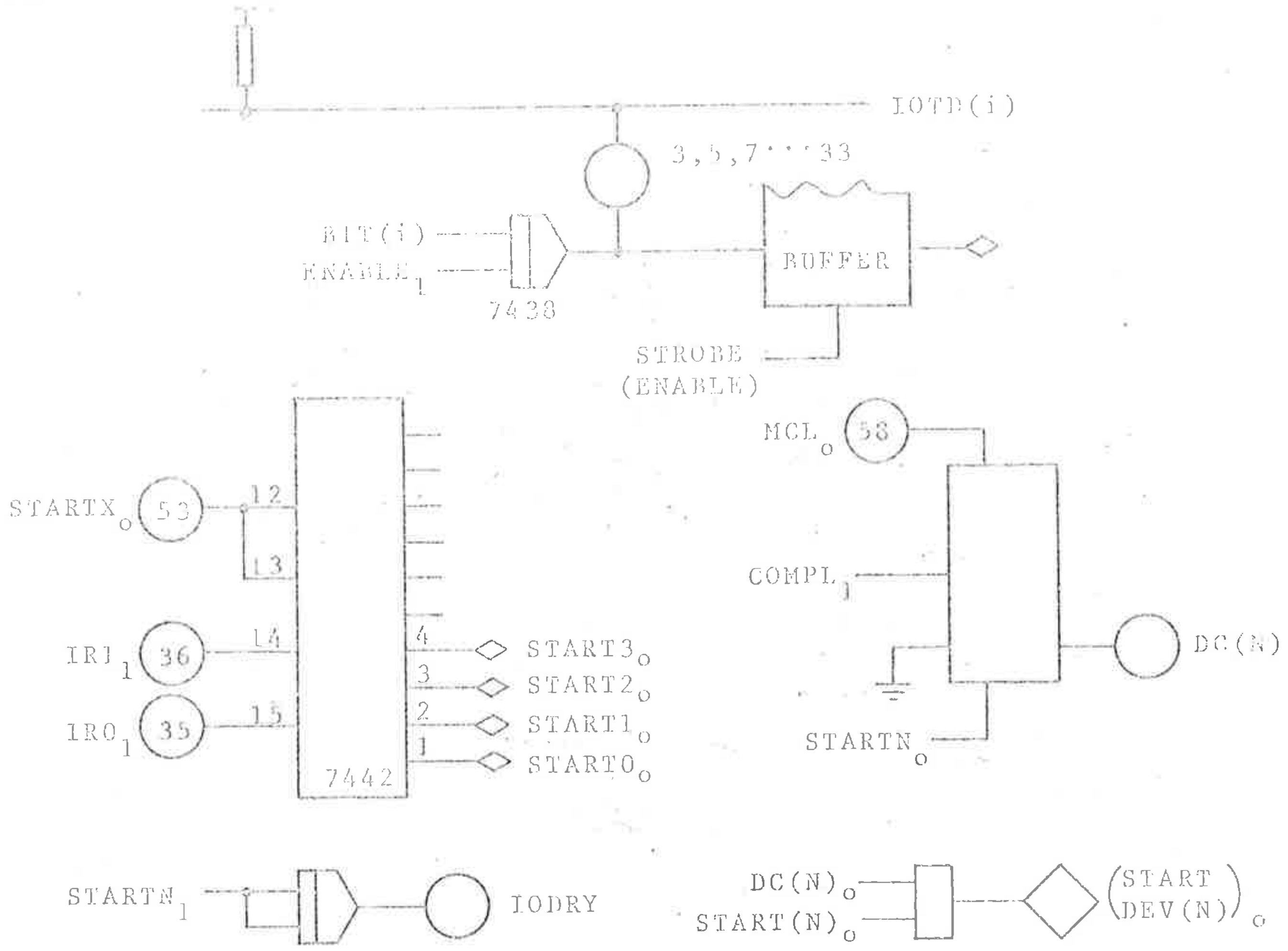


Figure 1.2

NORD-20 STANDARD I/O GROUP

DRAWN BY IB/eml	Remarks	Replacement for	Date
APPROVED BY		Replaced by	Date
DATE 2/3/72			



ABSOLUTE RATINGS:

- The "IOTD-BUS" and IODRY must be driven from a 30-fan-out wired-or gate. Ex.: SN 7438.

- The buffer(s) should not load the bus with more than two standard loads.

- MCL and all IR-bits should not be loaded with more than one standard load. All these requirements are relevant for one I/O-Bus position.

- Data direct from IOTD to cable must be buffered and enabled by START(N).

NB: - For devices with no response delay, the DCX may be grounded.

- For devices with response faster than (1-2)μs, the start to the device must be sent after the end of START-X if the standard DCX flip-flop is used.

DRAWN BY IB/eml	Remarks	Replacement for	Date
APPROVED BY		Replaced by	Date
DATE 3/3/72			