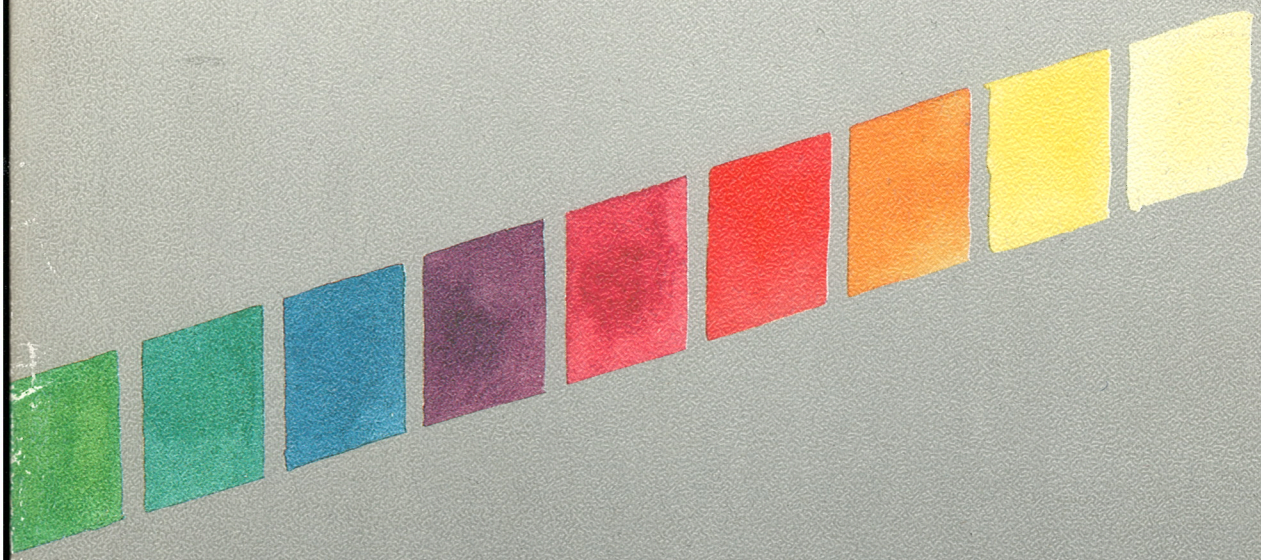


ND-110 and ND-120 Microprogrammer's Guide

ND-06.031.1 EN

Norsk Data

ND-110 and ND-120 Microprogrammer's Guide

ND-06.031.1 EN

Version 1 November 1987

Send all documentation requests to:

Norsk Data A.S
Graphic Centre
P.O. Box 25 – Bogerud
N-0621 Oslo 6
NORWAY

The product

In the ND-100 family of computers from Norsk Data, the ND-110 Standard and ND-110/CX central processor unit (CPU) cards are the successor CPU developments to the ND-100, ND-100/CE and ND-100/CX CPUs.

The ND-120/CX CPU is a high-speed version (approximately 1.9 times faster) of the ND-110/CX CPU.

The ND-110 CPU is upwards compatible with the ND-100 CPU. Similarly, the ND-120 CPU is upwards compatible with the ND-100 and the ND-110 CPUs. Note however that the ND-120/CX CPU's on-board memory requires that other memory boards in a ND-100 series card crate be set up with the correct memory address ranges.

The manual

This manual describes the microinstruction word format used in the ND-110 and ND-120 CPUs, and how the various fields within the microinstruction word control the operation of the CPU hardware.

The manual is designed as a reference document, although it may be read sequentially if an overview is required of how the microcode controls the operation of the CPU.

The reader

The information in this manual is intended for designers and service personnel. It is assumed that the reader has a good understanding of the basic functional elements in computers.

Related manuals

ND-06.018	NORD-100 Microprogramming Description. Describes the microprogram of the ND-100 CPU, including the format of the ND-100 microinstruction word.
ND-06.030	ND-110 and ND-120 Functional Description. Describes the functionality of the ND-110 and the ND-120 CPU cards.
ND-30.080	ND-120 Installation Guide. Describes how to install the ND-120/CX CPU card into an ND-100 series card crate.
ND-99.058	ND-110 Installation Guide. Describes how to install the ND-110/CX CPU card into an ND-100 series card crate.

List of figures

CHAPTER 1 SCOPE OF THE MANUAL

CHAPTER 1 SCOPE OF THE MANUAL

This manual describes how the microcode controls the operation of the ND-110 and ND-120 CPUs.

- Chapter 2 gives an overview of the CPU's functional hardware elements that interface to the microinstruction words issued from control store.

- Chapter 3 gives a description of the structure of the microinstruction word, together with an explanation of the function(s) performed by each of its fields. This description relates the function(s) of each field to its effect on the CPU hardware.

- Chapter 4 gives an appreciation of the activities involved in writing and assembling microcode, and describes the use of the TRR CS instruction to patch the existing contents of the control store.

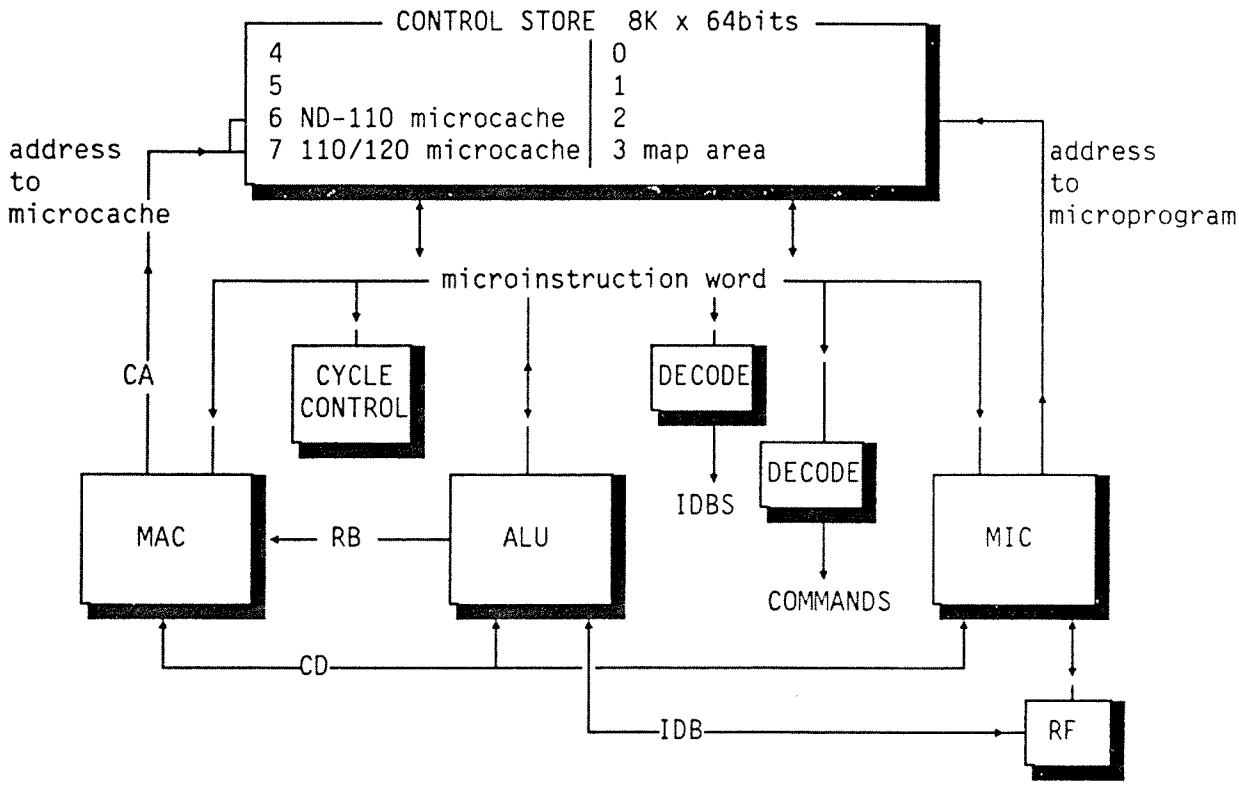
For ease of reference, the list of mnemonics for the ND-110 and ND-120 microinstruction sets are included in Appendices A and B respectively.

CHAPTER 2 CPU FUNCTIONAL INTERFACE TO THE MICROCODE

CHAPTER 2 CPU FUNCTIONAL INTERFACE TO THE MICROCODE

2.1 FUNCTIONAL ELEMENTS

The CPU includes functional elements of hardware which interface directly to various bits of the 64-bit microinstruction word output from the control store. These are represented in figure 1.



ALU : 16-bit arithmetic and logic unit
CA : cache address bus
CD : cache data bus
COMMAND : command field decode
IDB : internal data bus
IDBS : internal data bus source
MAC : macroinstruction address controller
MIC : microprogram instruction controller
RB : B-operand
RF : register file

Figure 1. Microcode interface to CPU hardware functions

The following description provides a summary of the functions performed by these elements in response to microinstruction words issued from the control store. In doing so, it introduces the terminology used in chapter 3 (Microinstruction word format) when referring to functions that the microinstruction word is operating on in the CPU hardware.

For a full functional description of the CPU, refer to the ND-110/ND-120 Functional Description manual (ND-06.030).

2.2 CONTROL STORE

This is an 8K x 64-bit random access memory area which stores the microcode. In normal operation it outputs successive microinstruction words in the sequence defined by the microinstruction controller (MIC).

The whole of the control store is loaded from Control Store PROM when the control panel Master Clear button is pressed, or when the OPCOM command LCS (ND-120 only) is issued.

Figure 1 illustrates how the 8K address range in the control store is assigned.

- The map area contains the first microinstruction for every macroinstruction in the CPU's repertoire.

- Microinstruction cache is addressed in parallel with the macroinstruction cache, via the CA bus.

  Every time the instruction cache is updated with a new entry, the first microinstruction for that macroinstruction is also written into the microcache area of control store. Then, if an instruction fetch operation gives a cache hit, the macroinstruction is obtained from cache and at the same time its first microinstruction is also read from microcache. This cuts out any introduced by the MIC computing the control store map address plus the access time to then obtain this first microinstruction.

  In the ND-110, the instruction cache is 2K long, and the corresponding microcache address area uses the top 2K in control store.

  In the ND-120, the instruction cache is only 1K long, and the corresponding microcache address area occupies the top 1K in control store.

2.3 MIC

The MIC computes control store addresses, to ensure that the CPU executes microinstructions in the correct sequence.

Fetch operations in the microcode obtain the next macroinstruction, from the memory address specified by the MAC. The MIC then generates the "map" (microprogram address pointer) entry points for the fetched macroinstruction. by adding 6000_8 to the op code (i.e. the most significant 10 bits) of the macroinstruction itself. This defines the starting point in microprogram for every macroinstruction fetched into the CPU.

If a cache hit occurred for the instruction fetch, the first microinstruction for this macroinstruction is obtained directly from microcache (see Control store above), and so the MIC's operation to compute the map address is bypassed, and the result of its computation is ignored.

One or more microinstructions are required to execute a macroinstruction. The MIC controls the sequence required. This may specify a jump condition, or the current address + 1 (next), or a return condition (from an internal 4-deep push/pop stack which can nest up to four microprogram subroutine levels). The microinstruction control may include conditional branching in the microcode sequence (involving selection from a variety of test objects), and vectored jump sequencing.

The MIC incorporates the loop counter function. This is a 6-bit register, with the most significant bit as a sign bit. It has an auto-increment/decrement facility, which enables it to be counted towards zero from either direction. On reaching zero, further counting is inhibited. The loop counter is used to repeat one (or several) microinstructions a predetermined number of times, or to count (for example) the number of shifts needed to normalise a floating point number.

2.4 MAC

This provides the logical (virtual) 16-bit memory address, to the Memory Management System (MMS). It includes arithmetic operations to speed the computation of memory addresses, avoiding reference to the ALU for such purposes. Microinstruction command decodes specify the memory reference computation required.

2.5 ALU

This performs arithmetic and logic functions. Two 4-bit pointers, the A and B operands, select sources for operations to be performed. Nine ALU function bits (ALUI) select operations within the ALU. Data input and output ports are directly connected to the CPU's internal bus system.

2.6 REGISTER FILE

The register file stores the register blocks for the 16 program levels of the CPU. Each block comprises 16 registers. Of these, 8 hold the registers P, B, X, L, T, D, A and STS. The other 8 are used by the microprogram as scratch registers, for MOPC (Microprogram Operators Panel Control), and the interrupt system. The register file is addressed using the A operand to indicate the level, and the B-operand to indicate the register within that level.

2.7 COMMAND DECODE

This generates control signals from the COMM and MIS fields of the microinstruction word. These define the CPU control operation that is to be performed.

2.8 IDBS DECODE

This specifies which source of data is to be enabled onto the internal data bus (IDB) of the CPU. These sources are identified as functonal registers in the CPU.

2.9 CYCLE CONTROL

The cycle controller functions as a finite-state machine which defines the timing sequence in executing every microinstruction. Each microinstruction takes one microcycle to complete. A microcycle is in turn subdivided into a number of nanocycles. A nanocycle is a fixed period of time (25.6ns in ND-110/CX and ND-120/CX).

The number of nanocycles required depends on what tasks the microinstruction specifies, and on external events, including memory access, ND-100 bus activity, and the trap and interrupt systems.

CHAPTER 3 MICROINSTRUCTION WORD FORMAT

CHAPTER 3 MICROINSTRUCTION WORD FORMAT

3.1 CPU CONTROL OPERATORS

Control of the CPU in the ND-110 and the ND-120 is effected by the microcode. Sequencing through successive microinstructions is determined by a combination of external events (e.g. interrupts, DMA cycles), and the microcode itself. Fetch operations in the microcode obtain the next macroinstruction, which then maps the CPU to the appropriate address in control store, to implement that macroinstruction by microcode.

Each microcycle is subdivided into nanocycles to implement a synchronous machine. External events affect the CPU through the trap and interrupt systems and the nanocycle controller. These events are synchronised to the CPU before they are used.

3.2 ND-110 MICROINSTRUCTION WORD FORMAT

ND-110 MICROINSTRUCTION CHART

The definition of fields in the ND-110's microinstruction word, and their functions, is summarised in the chart in figure 3 on page 17.

SUMMARY OF CHANGES FROM ND-100 TO ND-110

For ease of comparison, the ND-100 microinstruction word format chart is also included here, in figure 2 on page 16.

Bit 50	name changed from EN EXT to XREF, same function (Extended Register File).
Bits 42-43	MISO-1 used more widely to extend the repertoire of command decodes (see COMMO-4 field).

Bit 41	name changed from SEXT to IDBS4.
Bits 37-41	IDBS0-4 fully decoded, giving extended list of IDB source selections.
Bits 32-36	repertoire of command decodes extended.
Bits 25-27	branch addressing changed, to use only VECT bit 25 with MISO bit 42.
Bits 26-27	$SLWCS_0$ (bit 26) and $MAPJ_0$ (bit 27) renamed $DLY0_0$ and $DLY1_0$ respectively, and used in microcycle length control.
Bits 20-21	cycle timing control changed, to use only bit 21, and its name is accordingly changed from TC1 to DELAY.
Bit 20	name changed from TC0 to BIT20, and now used for miscellaneous addressing functions, in particular as an extension to the branch address field (bits 0-11).
Bits 4-7	condition test objects 0, 6, 8 and 9 are changed.

3.3 ND-120 MICROINSTRUCTION WORD FORMAT

ND-120 MICROINSTRUCTION CHART

The definition of fields in the ND-120 microinstruction word, and their functions, is summarised in figure 4 on page 18.

SUMMARY OF CHANGES FROM ND-110 TO ND-120

Bits 37-41	IDBS decodes 33, 34, 36 and 37 changed.
Bits 32-36	command decodes 5, 36.2 and 36.3 changed.
Bits 26-27	name changed from DLY to DELAY to accord with revised timing delay function, and logic sense inverted.

Bit 25	$VECT_1$ logic sense inverted, same function.
Bit 24	$SCOND_1$ logic sense inverted, same function.
Bit 23	$ECOND_1$ logic sense inverted, same function.
Bit 22	$LOOP_1$ logic sense inverted, same function.
Bit 21	$DELAY_1$ name changed from DELAY to DLY to accord with revised timing delay function, and logic sense inverted.

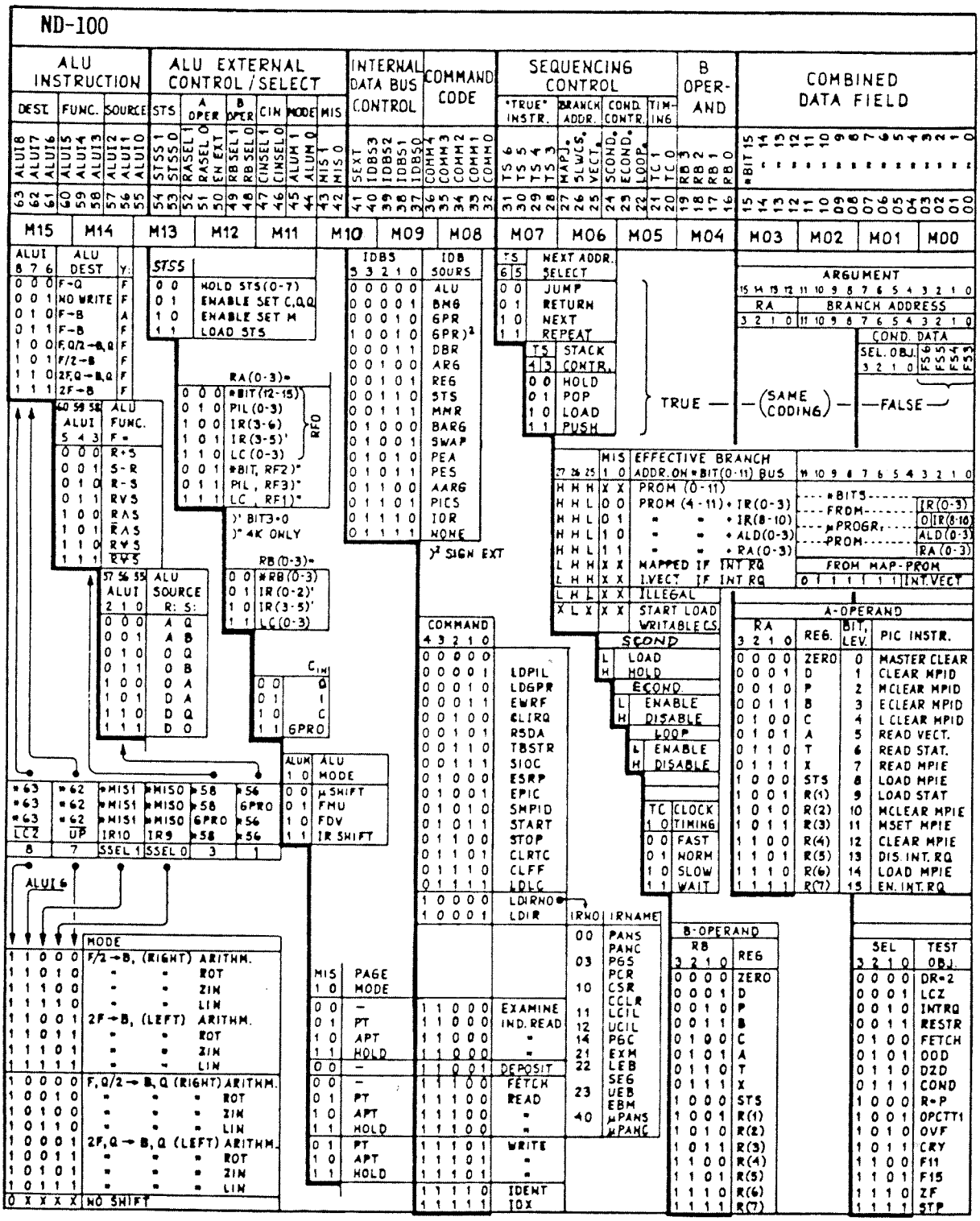


Figure 2. ND-100 microinstruction word format & functions chart

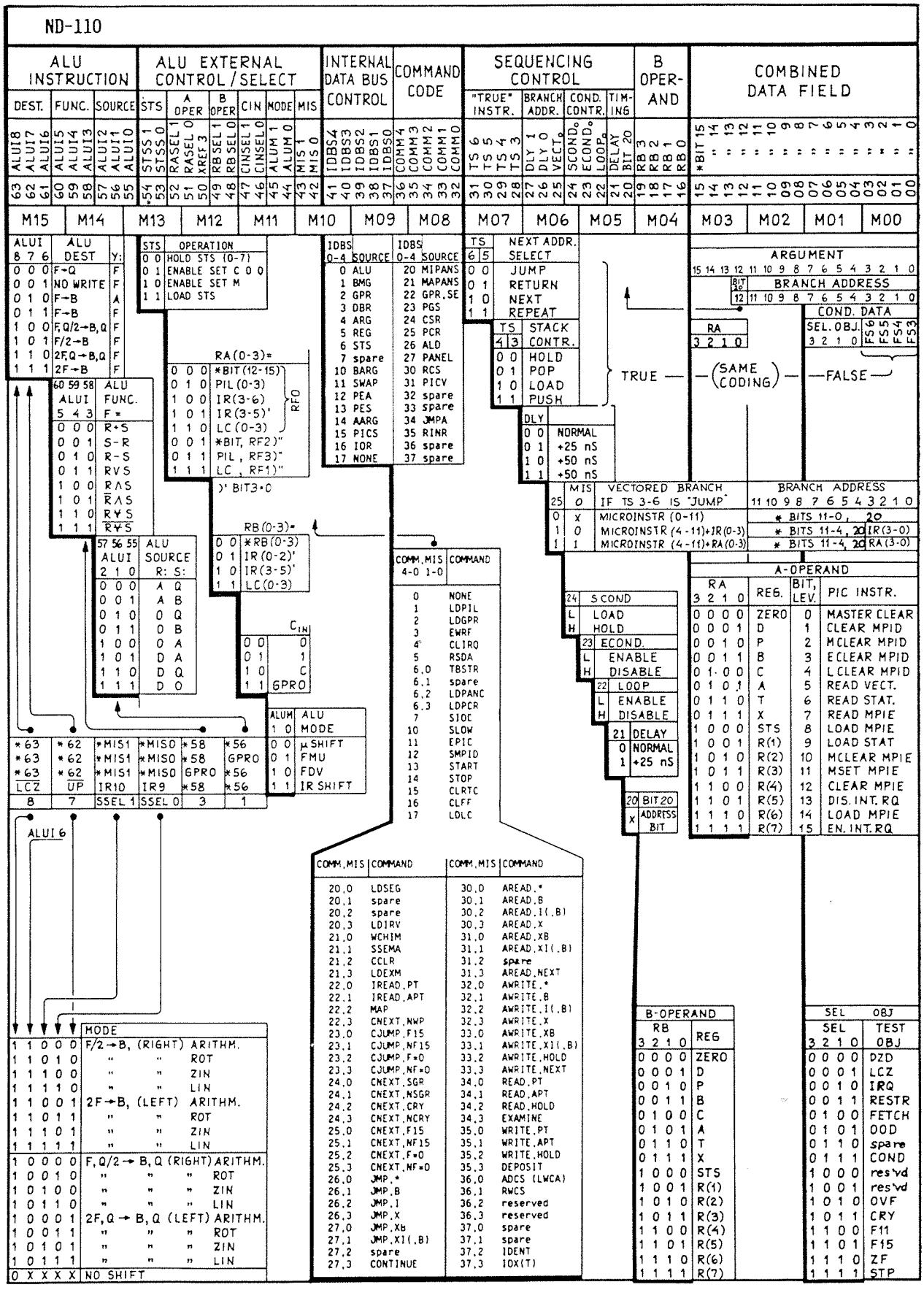


Figure 3. ND-110 microinstruction word format & functions chart

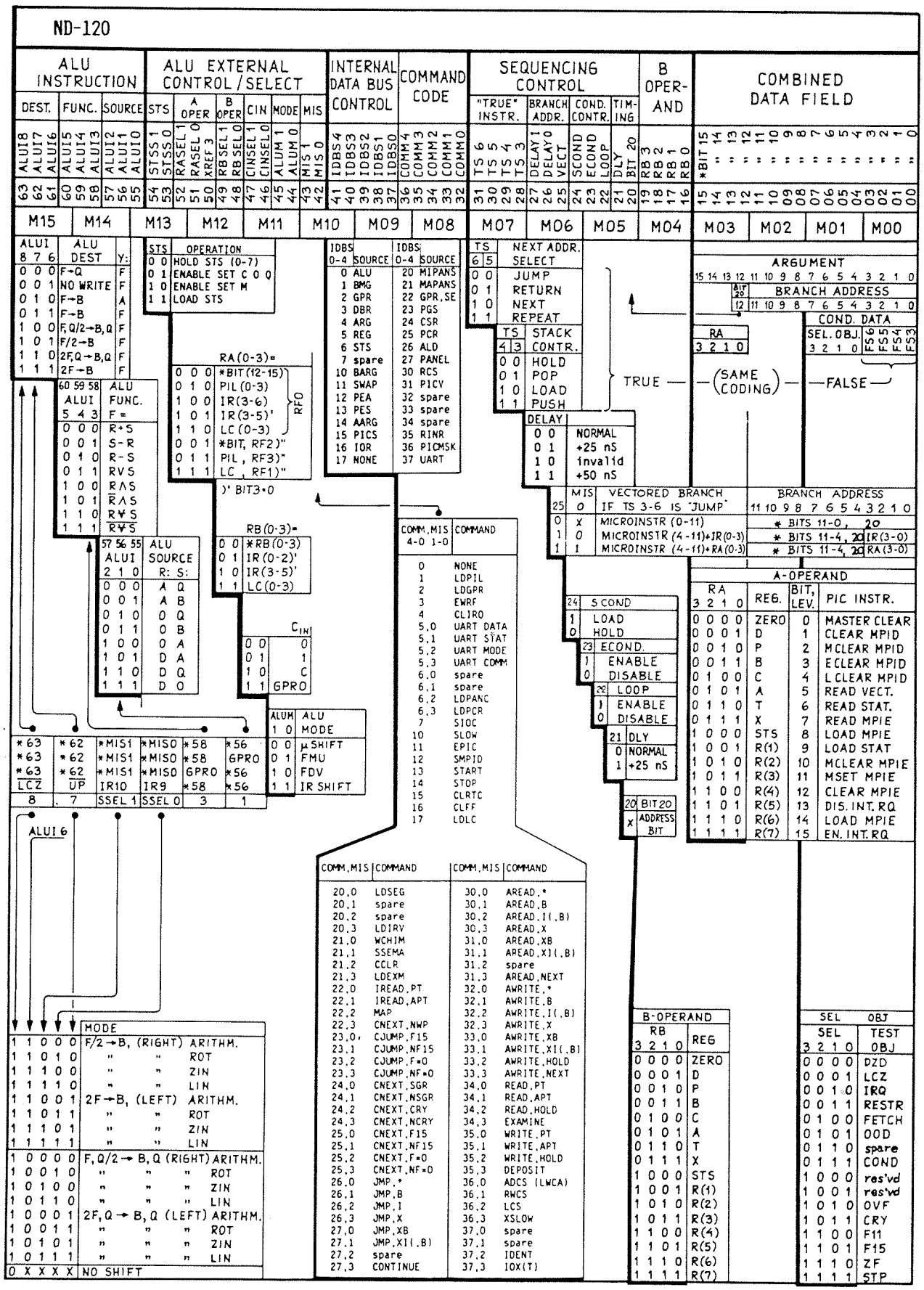


Figure 4. ND-120 microinstruction word format & functions chart

3.4 MICROINSTRUCTION WORD: FIELD DEFINITIONS

Field descriptions This section covers the field definitions for both the ND-110 and the ND-120. Where a definition differs between ND-110 and ND-120, the ND-110 definition is given first in italics, then the ND-120 definition follows in non-italics.

In these definitions:

"spare" means available to a microprogrammer, for future use.

"reserved" means not used and not available to a microprogrammer.

Bits 0-3 : This selects the false sequence to be latched by SCOND (bit 24). It specifies the sequencer instruction (jump, return, next, repeat) and stack control (hold, pop, load, push), in case a later test ECOND (bit 23) gives a false result.

Bits 4-7 : This selects the test object to be latched by SCOND (bit 24). If a test object changes state, one microinstruction will be executed before the change is detected. This is caused by the pipelining of ALU, and sequencing calculations.

The 16 test objects are:

0	DZD	double zero detect.
1	LCZ	loop counter is 0. Used to terminate looping on an instruction.
2	IRQ	used to test the interrupt request flip-flop, to see if interrupts are pending.
3	RESTR	used to test if the current program is running in restricted mode. Instructions that are privileged should not execute when this condition is true.
4	FETCH	indicates whether the last memory access was a fetch of new macroinstruction. Used to generate the correct P value in case of page fault.
5	OOD	the one out detect flip-flop is used to set TG (floating point rounding indicator) correctly in FAD, FSB and FMU instructions.
6	-	spare.

7	COND	this condition latches the true/false result of the last microinstruction, so that it can be tested later if that is more convenient than testing it now. It will postpone the test as long as desired.
8	-	reserved (always false). In ND-100, it was used to detect if a single or multiple store instruction has affected *+1, and if true, the prefetched instruction must be discarded.
9	-	reserved.
10	OVF	overflow from ALU. Indicates overflow during arithmetic operations.
11	CRY	carry out from ALU.
12	F11	bit 11 from ALU.
13	F15	bit 15 (sign bit) from ALU.
14	ZF	result = 0 from ALU.
15	STP	indicates the Stop flip-flop is set (i.e. no program running).

Bits 0-11 : This field may contain the branch address in jump microinstructions, in which case Bit 20 is available as the 13th bit of the branch address (see Bit 20). The jump addresses may also be generated by the MIC, and partly by the vectorized jump mechanism. When the vectorised jump mechanism is in use the microinstruction bits 0-3 will be discarded.

Bits 0-15 : These are enabled as an argument, onto the IDB if IDB source (IDBS bits 37-41) is ARG (4).

Bits 12-15 : These select the A-operand if RASEL (bits 51-52) are 0. The A-operand controls the BMG, as well as being a register file and ALU operator.

Bits 16-19 : These select the B-operand if RBSEL (bits 48-49) are 0.

Bit 20 : For microinstruction branches (see Bits 0-11), this is issued as the 13th address bit, along with bits 0-11. This covers the 8K address range of the ND-110 and ND-120 control store.

Bit 21 : *In ND-110, this is the delay function DELAY, which extends the length of the previous microcycle because of the function of the current microcycle.*

In ND-120, this is the delay function, which extends the length of the current microcycle because of the function of the current microcycle.

Bits 22-24 : These are condition control bits:

- *In ND-110, these are true when = 0 (low)*
- In ND-120, these are true when = 1 (high).

Bit 22 (LOOP) increments or decrements the loop counter (always towards zero). If the selected condition is false, RETURN/HOLD is executed by the MIC. If it is true, TS3-6 (bits 28-31) are executed. The new value of the loop counter will not be available as an A-operand or test object until after the next microinstruction.

Bit 23 (ECOND) enables testing (against the previously set test condition) of the result of the previous microinstruction. If the result is true, the MIC sequencer executes TS3-6 (bits 28-31) to find the next microinstruction. If the result is false, the MIC sequencer executes the false sequence instruction which was previously output to the hold register by SCOND (bit 24).

Bit 24 (SCOND) sets a condition to be tested on a later occasion. A 4-bit code indicating the test condition, and a 4-bit code indicating the sequence instructions in case of a false test, are output to the Hold register.

Bit 25 : This specifies a vectored jump if the true sequence is "jump":

- *In ND-110, it is true when = 0 (low)*
- In ND-120, it is true when = 1 (high).

If true, the vector address is the microinstruction branch address (Bits 0-11 plus Bit 20). If false, the lower 4 bits of the vector address are determined by MISO (bit 42) to be either IR(0-3) or the A-operand.

Bits 26-27 : *In ND-110, these are the delay function DLY(1,0). They extend the length of the current microcycle because of the function of the current microcycle. They are true when = 0 (low).*

In ND-120, these are the delay function(1,0). They extend the length of the previous microcycle because of the function of the current microcycle. They are true when = 1 (high).

Bits 28-31 : These control the microprogram sequence "next address source" and "push-pop stack operation" if the condition result is true. They will only be in control as long as LOOP (bit 22) and ECOND (bit 23) are true.

Bits 32-36 : This is the Command Field.

COMM ,MIS	COMMAND	DESCRIPTION
00	NONE	No command executed.
01	LDPIL	Load the upper byte of the STS register from the upper byte of IDB. The new PIL will not be available as the A-operand until the next microinstruction has been completed.
02	LDGPR	Load IDB into GPR. If a shift is specified in the same microinstruction, GPR will not be shifted, only loaded.
03	EWRF	Write IDB into the register file specified by the A and B operands.
04	CLIRQ	Force interrupt request to the cleared condition. This is used by the interrupt handler.
05	*RSDA*	*In ND-110, reset the data available signal from the UART.*
05,x	CEUART	In ND-120, enable the UART: • If IDB source (bits 37-41) =37, the operation is READ. • If IDB source ≠37, the operation is WRITE. The COMM,MIS commands are: 05,0: UART data 05,1: UART status 05,2: UART mode 05,3: UART comm See the data sheet on the 2661 UART device for further details.
06,0	*TBSTR*	*In ND-110, Transmit Data Strobe command to the UART on the CPU card.*
06,0	-	In ND-120, spare.
06,1	-	spare.
06,2	LDPANC	load lower byte of IDB into the panel processor FIFO.
06,3	LDPCR	load paging control register.

07 SIOC

load I/O control register. The control register bits are:

Bit	Description
7	Reset real time clock.
6	Set terminal #1 in OPCOM (console), as opposed to normal
5	Enable master clear: red LED ON[1]
4	Initialisation completed: green LED ON[1]
3	clock interrupt generated from RTC trap handler.
2	enable output interrupt on level 10, from terminal #1 (UART transmit buffer empty).
1	enable input interrupt on level 12, from terminal #1 (UART data available).
0	enable clock interrupt on level 13.

Note: [1] When CLEAR is held, both red and green LEDs are illuminated. When CLEAR is released, red LED only stays illuminated while control store is loaded from EPROM, hardware is initialised, and self-test runs. After successful completion, the red LED is extinguished and the green LED is illuminated.

10 SLOW

Command a "slow" microcycle. In ND-120, a "slow" microcycle is a minimum (no wait states) of 204.8ns. In ND-110, it is 256ns.

11 EPIC

Enable priority interrupt controller. The A-operand then specifies a command to the PIC. The commands are:

0	MCLR	clear regs and enable ints
1	CLRMPID	clear all interrupts
2	MCLRMPID	clear ints from M-bus
3	ECLRMPID	clear ints from Mask reg
4	LCLRMPID	clear int for last vect rd
5	RDVECT	read vector
6	RDSTAT	read status register
7	RDMPIE	read mask register
10	LDMPIE	set mask reg: inh all ints
11	LDSTAT	load status
12	MCLRMPIE	bit clear mask register
13	MSETMPIE	bit set mask register
14	CLRMPIE	clr mask reg: en all ints
15	DISINTRQ	disable interrupt request
16	LDMPIE	load mask register
17	ENINTRQ	enable interrupt request.

12	SMPID	Set bits in the micro-PID (Priority Interrupt Detect) register in the PIC. Bits can only be set to 1 by this command. (The PIC command CLRMPID resets bits to 0.)
13	START	Reset the Stop flip-flop. The CPU will start executing a macro (main memory) program.
14	STOP	Set the Stop flip-flop. When the next FETCH is performed, the microprogram is forced to microaddress 16 (panel interrupt).
15	CLRTC	Clear real-time interrupt. Every 20ms the real time clock will generate a panel interrupt, thereby entering the clock routine. This routine sends out the CLRTC command.
16	CLFF	Clear all flip-flops having special functions regarding the floating point rounding indicator (TG). The DZD (double zero detect used in FDV) and the OOD (one out detect used in FAD, FSB and FMU) flip-flops are cleared by CLFF.
17	LDLC	Load the loop counter with the 6 lower bits of the IDB. The modified loop counter will not be available as an A-operand until the next microinstruction has been completed.
20,0	LDSEG	Load segment register. This selects a physical 64K memory address area (address bits 16-23). The segment register is located in the MAC.
20,1	-	Spare.
20,2	-	Spare.
20,3	LDIRV	Load instruction register (MIC).
21,0	WCIHM	Write cache inhibit map (bit 15 of IDB, see TRR CILP macroinstruction).
21,1	SSEMA	Set semaphore flag (prepare semaphore request).
21,2	CCLR	Cache clear.
21,3	LDEXM	Load examine mode in MAC function.

22,0	IREAD,PT	Indirect address read, relative to page table.
22,1	IREAD,APT	Indirect address read, relative to alternative page table.
22,2	MAP	Address control store mapped as when FETCH. Used in Execute Register. instruction (IDB contains instruction).
22,3	CNEXT,NWP	Conditional "next", if P is not changed in the last cycle.
23,0	CJMP,F15	Conditional jump if F15 is true (ALU result is negative).
23,1	CJMP,NF15	Conditional jump if F15 is false (ALU result is positive).
23,2	CJMP,F=0	Conditional jump if F=0 is true (ALU result is zero).
23,3	CJMP,NF=0	Conditional jump if F=0 is false (ALU result is not zero).
24,0	CNEXT,SGR	Conditional next if SGR true (greater or equal).
24,1	CNEXT,NSGR	.. SGR false (less).
24,2	CNEXT,CRY	.. carry true (o'flow)
24,3	CNEXT,NCRY	.. carry false.
25,0	CNEXT,F15	Conditional next if F15 is true (ALU result is negative).
25,1	CNEXT,NF15	Conditional next if F15 is false (ALU result is positive).
25,2	CNEXT,F=0	Conditional next if F=0 is true (ALU result is zero).
25,3	CNEXT,NF=0	Conditional next if F=0 is false (ALU result is not zero).
26,0	JMP,*	Macro jump, P-relative.
26,1	JMP,B	Macro jump, B-relative.
26,2	JMP,I	Macro jump, indirect.
26,3	JMP,X	Macro jump, X-relative.
27,0	JMP,XB	Jump, post-indexed, B-relative.
27,1	JMP,XI (,B)	Jump, post-indexed, indirect.
27,2	-	Spare.
27,3	CONTINUE	Fetch relative to P.
30,0	AREAD,*	Read, P-relative.
30,1	AREAD,B	Read, B-relative.
30,2	AREAD,I (,B)	Read, indirect, may be B-indexed.
30,3	AREAD,X	Read, X-relative.
31,0	AREAD,XB	Read, post-indexed after B-relative.
31,1	AREAD,XI (,B)	Read, post-indexed after indirect.
31,2	-	Spare.
31,3	AREAD,NEXT	Read next location.

32,0	AWRITE,*	Write, P-relative.
32,1	AWRITE,B	Write, B-relative.
32,2	AWRITE,I (,B)	Write, indirect, may be B-indexed.
32,3	AWRITE,X	Write, X-relative.
33,0	AWRITE,XB	Write, post-indexed after B-relative.
33,1	AWRITE,XI(,B)	Write, post-indexed after indirect, may be B-indexed.
33,2	AWRITE,HOLD	Write in last-used address (used by "MIN").
33,3	AWRITE,NEXT	Write in next location.
34,0	READ,PT	Read, page table-relative, address from IDB.
34,1	READ,APT	Read, alt PT-relative, address from IDB
34,2	READ,HOLD	Read from last-used PT, addr from IDB.
34,3	EXAMINE	Read, physical addressing using segment register.
35,0	WRITE,PT	Write, page table-relative, address from IDB.
35,1	WRITE,APT	Write, alt PT-relative, addr from IDB.
35,2	WRITE,HOLD	Write to last-used PT, address from IDB
35,3	DEPOSIT	Write, physical addressing using segment register.
36,0	ADCS	Load control store address from IDB.
36,1	RWCS	Read/write control store as addressed by ADCS.
36,2	-	*In ND-110, reserved.*
36,2	LCS	In ND-120, load control store from PROM and perform a master clear (i.e. same function as MCL button on front panel).
36,3	-	*In ND-110, reserved.*
36,3	XSLOW	In ND-120, force current microcycle to the maximum length of time (435.2ns). This is used for very slow I/O devices, e.g. the UART.
37,0	-	Spare (available for future use).
37,1	-	Spare (available for future use).
37,2	IDENT	IDENT instruction request.
37,3	IOX(T)	IOX or IOXT instructon request.

Bits 37-41 : This controls which source is selected onto the IDB.

IDBS 0-4	SOURCE	DESCRIPTION
0	ALU	ALU result or register pointed to by the A-operand (dependent on bits 60-63).
1	BMG	Bit-mask-generator.
2	GPR	General purpose register.
3	DBR	DB/cache read register.
4	ARG	Microcode argument register.
5	REG	Register file, selected by A & B operands

6	STS	Macro status register.
7	-	Not used.
10	BARG	B-operand as argument on bits 0-3 (values 0-17).
11	SWAP	Byte swap register (previous IDB contents are byte-swapped).
12	PEA	Parity error address.
13	PES	Parity error status & address.
14	AARG	A-operand as argument on bits 3-6 (with bits 0-2 =0) giving range of values 0-170
15	PICS	Internal status of the PIC. A read status command must be issued to the PIC. simultaneously (command decode 11 - EPIC)
16	IOR	I/O register from UART etc.
17	none	Nothing enabled onto the IDB by IDBS field
20	MIPANS	Panel status register (EPANS).
21	MAPANS	Panel status register (EPANS), and also panel status read bit (12) is reset.
22	GPR,SEXT	General purpose register, sign extended.
23	PGS	Paging status extended.
24	CSR	Cache status register.
25	PCR	Paging control register.
26	ALD	Automatic load descriptor and print-status
27	PANEL	Panel interrupt vector.
30	RCS	Read control store (see also command decode 36,1 -RWCS).
31	PICV	PIC interrupt vector (see also command decode 11 - EPIC).
32	-	Spare.
33	-	Spare.
34	*JMPA*	*In ND-110, LA(10-15), CA(0-9) update P in ALU (jump).*
34	-	In ND-120, spare.
35	RINR	Read installation number.
36	-	*In ND-110, spare.*
36	PICMASK	In ND-120, read PIC mask register.
37	-	*In ND-110, spare.*
37	UART	In ND-120, read UART (see also command decode 5,x - UART).

Bits 42-43 : Miscellaneous bits (MIS 0-1). These bits are used:

- to control the shift linkage in the ALU.
- as an extension to the command field (see bits 32-36).

Bits 44-45 : These modify bits 55-63, before they reach the ALU. See bits 55-63.

Bits 46-47 : These control the carry-in line to the ALU:

00	Carry is 0.
01	Carry is 1.
10	Carry is the C flip-flop of the STS register.
11	Carry is the GPR bit 0 (used in division).

Bits 48-49 : These control the source of the B-operand to the CPU:

00 The B-operand is bits 16-19 of the microinstruction. It is used to address a register in register file or ALU WRF.

01 The B-operand is bits 0-2 of the current macroinstruction. This is used to address the destination register, in macroinstructions.

10 The B-operand is bits 3-5 of the current macroinstruction. This is used to address the source register, in macroinstructions.

11 The B-operand is bits 0-3 of the loop counter. This is used when performing a level change, to provide fast scanning of addresses in register file and working register file. It is also used to read the loop counter from MIC onto BA0-3, using BARG in the ALU to steer it onto the IDB.

Bits 50-52 : These control the source of the A-operand to the ALU.

000 The A-operand is bits 12-15 of the microinstruction.

010 The A-operand is the PIL register (used to address the current level register block in the register file).

100 The A-operand is bits 3-6 of the current macroinstruction. This is used in bit operation instructions, and in instructions specifying a level in the register file).

101 The A-operand is bits 3-5 of the current macroinstruction. This is used to address the source register, in macroinstructions.

110 The A-operand is bits 0-3 of the loop counter.

001 Extended register file RF2 addressed by *BIT12-15 on the A-operand. This is used to store global scratch registers, accessible by microcode only.

011 Extended register file RF3 addressed by program level bits PIL0-3 on the A-operand. This is used to store the PCRs and other level-specific scratch registers that must be retained.

011 Extended register file RF1 addressed by loop counter bits LC12-15 on the A-operand. This is used for temporary storage in loop-context operations.

Bits 53-54 : These control the behaviour of the 8 lower bits of the STS register:

00 does not affect the STS register.

01 updates the C, O and Q flip-flops, to the result of the current ALU instruction.

10 updates the M flip-flop to the disappearing bit in a shift microinstruction. If there is ALUM IRSHIFT, M will be updated, regardless of the setting of bits 53-54.

11 loads the 8 lower bits of the STS register, from the IDB.

Bits 55-63 : Together with bits 44-45, these control the behaviour of the ALU. Bits 44-45 modify bits 55-63 before they reach the ALU.

ALUM0 = 0 (bit 44)
ALUM1 = 0 (bit 45): Micro Controlled Mode (μSHIFT)

The ALU behaviour is controlled entirely from the microprogram.

ALUM0 = 1 (bit 44)
ALUM1 = 0 (bit 45): Multiply Mode (FMU)

The ALU behaviour is controlled from the microprogram, except that GPR bit 0 controls the addition of the A-operand register or 0. This is only used in muliplication instructions.

ALUM0 = 0 (bit 44)
ALUM1 = 1 (bit 45): Divide Mode (FDV)

The ALU behaviour is controlled from the microprogram, except that GPR bit 0 controls whether or not the A-operand register should be added or subtracted. This is only used in divide instructions.

ALUM0 = 1 (bit 44)
ALUM1 = 1 (bit 45): Shift Instruction (IR SHIFT)

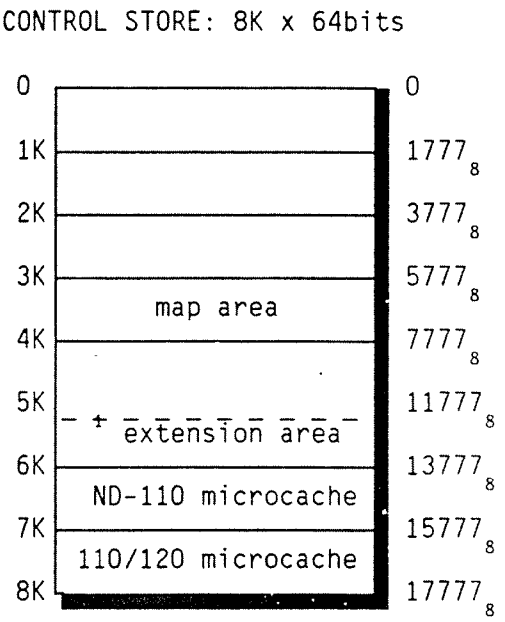
The shift direction and shift mode is controlled from the loop counter together with bits 9-10 of the macroinstruction. The extra microinstruction executed after the loop counter reaches 0 does not shift the ALU result.

CHAPTER 4 WRITING MICROPROGRAM

CHAPTER 4 WRITING MICROPROGRAM

4.1 CONTROL STORE AREA

The ND-110 and ND-120 have 8K of 64-bit control store. This area is assigned as shown in figure 5.



[1] Precise area of extension space in control store varies according to the version of microcode used.

Figure 5. Control store address assignments

- map

  The map entry area contains the first microinstruction for every macroinstruction fetched into the CPU.

- microinstruction cache

  ND-110 has 2K of microinstruction cache, occupying address range 6-8K.

  ND-120 has 1K of microinstruction cache, occupying address range 7-8K.

- extension area

  A part of the control store area, within the address range 5-6K, is available for extension of the microcode, for example to write new or customised machine instructions. The actual address area available for this varies according to the version of microcode supplied with the CPU. Before writing into this extension area, refer to ND to determine the precise address range that is available.

  In ND-120, the address range 6-7K is not required for microcache, and may be used as a further 1K of extension area. However, any new instructions written into this area could not then be ported into the same address area in a ND-110.

CAUTION

All locations in control store may be modified using the TRR CS instruction.

Severe system malfunction will result if a location that is already assigned by the system is overwritten using TRR CS.

4.2 GENERATING MICROCODE

Generating microcode involves the following operations:

1. specifying the complete functionality of the new macroinstruction that is to be implemented by the new microprogram.

2. designing and coding the necessary microinstructions to perform the new macroinstruction.

3. assembling the symbolic microprogram into binary format, using the microcode assembler program.

4. writing the binary representation of the WCS contents into main memory, then loading it into WCS.

4.3 DESIGN OF MICROCODE

Designing microcode to implement a machine instruction, requires detailed understanding of the structure of the microinstruction word, and how the various fields and combinations of fields in this word stimulate the hardware, to perform the required operations.

It also requires a detailed knowledge of all the microinstruction mnemonics (Appendix A for ND-110; Appendix B for ND-120), which provide the tools to code the microprogram.

4.4 MICROCODE ASSEMBLER

Language syntax

The microcode assembler program creates a binary file from the symbolic microcode. The following general points apply:

- if "/" is found in a character group, the current location counter (CLC) is updated to the octal number in front of /.
- if ":" is found in a character group, the rest of the group is regarded as a label.
- the character ";" terminates each microinstruction, and CLC is then incremented. Each microinstruction may contain an unspecified number of lines.
- the termination to a macroinstruction is the label CONT. This sets the Z register to zero, and executes a COMM,CONTINUE T,JMP which fetches the next macroinstruction relative to P. This command can be executed elsewhere.
- the characters "space" and "tab" are accepted by the microcode assembler as delimiters between mnemonics, labels and numbers.
- the character "%" is used to indicate that the rest of the line is a comment.

Any other characters are used to form character groups, which may be mnemonics, labels or commands. If a number between 0-7 is found in a character group, it is regarded as an octal number. Each mnemonic in the list is translated by the assembler into its corresponding octal value. All the octal values of the mnemonics in one microinstruction are "ORed" together. If an octal number is encountered, it is placed in bits 0-15 of the microinstruction word and considered as an argument. If a symbolic name is encountered which is not found in the mnemonic list, it is regarded as a label, which is or will be defined.

Microinstruction branching uses the standard format, whereby the condition and false sequence must be set up at least one microinstruction before CONDENABL (condition enable) is executed. The test is on the result of the set condition in the previous microinstruction.

- When there is no more input to the assembler, all labels are replaced by their values, which are filled into bits 0-11 and 20, in the correct microaddresses. Note that for ND-110 only, bits 22-27 of the microinstruction word are inverted after the word is assembled.

Assembling µcode

Microinstructions are generated using the Microassembler program. This program is internal to Norsk Data. For further information, the ND-500 Microprogramming Guide (ND-05.012) is a useful reference.

Five output files are generated from the assembler, with the name *-F32 or *-F48. Their filename extensions are:

:LIST	listing file - complete
:LABE	labels and their addresses
:ERR	error listing
:UDEF	undefined entries
:DATA	the assembled output for ?-PROM-PROGRAM.

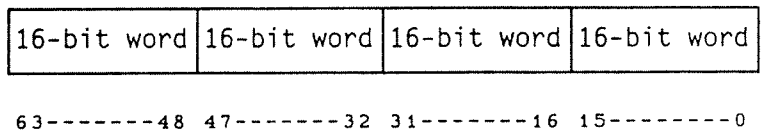
The control store PROM-program runs on the ND-500 under FORTRAN, and generates eight output files of the form Fnn-xx-yy:PROM, where nn is 32 or 48, and xx-yy is the microcode bit range (e.g. 0-7, 8-15, etc.).

For further information on writing of microcode, refer to Norsk Data.

4.5 UPDATING THE CONTROL STORE

Any 16-bit part of a location in any address in the control store may be patched using the TRR CS instruction, or may be inspected using the TRA CS instruction.

Since these instructions implement 16-bit data transfers, they need to be performed four times to write or read a complete 64-bit microinstruction word:



Because of the possible severe consequences that may arise as a result of a user modifying the contents of control store, the TRR CS and TRA CS instructions are privileged.

The TRA CS and TRR CS instructions are described in the ND-110/ND-120 Instruction Set manual (ND-06.029).

Reloading CS

Control store is reloaded (from the control store EPROM on the CPU) every time the machine is powered up, or whenever it is initialised by pressing the Master Clear button or issuing the LCS (ND-120 only) command from OPCOM. A control store reload operation overwrites any patches that were made to control store by the TRR CS instruction, so if those patches were a required feature they must then be reinstalled by repatching. Note that the MACL command in OPCOM initialises the machine without reloading control store.

Alternatively, permanently required patches may be blown into the control store EPROM. If this is required, refer to Norsk Data for assistance.

APPENDIX A MNEMONICS LIST FOR THE ND-110

The following pages present the list of mnemonics for the ND-110 microinstruction set.

The list includes the op-code for the microinstruction (in octal), and a one-line description of it.

Note that the microcode assembler for ND-110 inverts bits 22-27.

```
 1 A,Z          000000 000000 000000 000000 A-OP. IS 0. USED TO ADDR. ZERO REG., STATUS OR SCRATCH
 2 A,D          000000 000000 000000 010000 A-OP. IS 1. USED TO ADDR. THE D-REGISTER
 3 A,P          000000 000000 000000 020000 A-OP. IS 2. USED TO ADDR. THE P-REGISTER
 4 A,B          000000 000000 000000 030000 A-OP. IS 3. USED TO ADDR. THE B-REGISTER
 5 A,L          000000 000000 000000 040000 A-OP. IS 4. USED TO ADDR. THE L-REGISTER
 6 A,A          000000 000000 000000 050000 A-OP. IS 5. USED TO ADDR. THE A-REGISTER
 7 A,T          000000 000000 000000 060000 A-OP. IS 6. USED TO ADDR. THE T-REGISTER
 8 A,X          000000 000000 000000 070000 A-OP. IS 7. USED TO ADDR. THE X-REGISTER
 9 A,STS        000000 000000 000000 100000 A-OP. IS 10. USED TO ADDR. REG. 10, STATUS OR SCRATCH
10 A,R1         000000 000000 000000 110000 A-OP. IS 11. USED TO ADDR. REG. 11, ADDRESS OR SCRATCH
11 A,R2         000000 000000 000000 120000 A-OP. IS 12. USED TO ADDR. REGISTER 12, SCRATCH
12 A,R3         000000 000000 000000 130000 A-OP. IS 13. USED TO ADDR. REGISTER 13, SCRATCH
13 A,R4         000000 000000 000000 140000 A-OP. IS 14. USED TO ADDR. REGISTER 14, SCRATCH
14 A,R5         000000 000000 000000 150000 A-OP. IS 15. USED TO ADDR. REGISTER 15, SCRATCH
15 A,R6         000000 000000 000000 160000 A-OP. IS 16. USED TO ADDR. REGISTER 16, SCRATCH
16 A,R7         000000 000000 000000 170000 A-OP. IS 17. USED TO ADDR. REGISTER 17, SCRATCH
17 A,PIL        000010 000000 000000 000000 A-OP. IS VALUE IN "PIL" REGISTER, LOADED BY "COMM,LDPIL"
18 A,REG        000020 000000 000000 000000 A-OP. IS VALUE OF "INSTR. BITS 3-6" LOADED BY LAST "T,MAPJ"
19 A,SRCE       000024 000000 000000 000000 A-OP. IS VALUE OF "INSTR. BITS 3-5" LOADED BY LAST "T,MAPJ"
20 A,LC         000030 000000 000000 000000 A-OP. IS VALUE OF "LOOP COUNTER" TWO MICROINSTRUCTIONS AGO
21 PIC,RSTS     000000 000000 000000 060000 A-OP. IS 6. USED TO GIVE "READ STATUS" CMD. TO INTSYS
22 PIC,MCL      000000 000000 000000 000000 A-OP. IS 0. USED TO GIVE "MASTER CLEAR" CMD. TO INTSYS
23 PIC,MCLPID   000000 000000 000000 020000 A-OP. IS 2. USED TO GIVE "MASKED CLEAR PID" CMD. TO INTSYS
24 PIC,RMSK     000000 000000 000000 070000 A-OP. IS 7. USED TO GIVE "READ MASK" CMD. TO INTSYS
25 PIC,LOSTS    000000 000000 000000 110000 A-OP. IS 11. USED TO GIVE "LOAD STATUS" CMD. TO INTSYS
26 PIC,LMSK     000000 000000 000000 160000 A-OP. IS 16. USED TO GIVE "LOAD MASK" CMD. TO INTSYS
27 PIC,MCLMSK   000000 000000 000000 120000 A-OP. IS 12. USED TO GIVE "MASKED CLEAR MASK" CMD TO INTSYS
28 PIC,IOF      000000 000000 000000 150000 A-OP. IS 15. USED TO GIVE "IOF" CMD. TO INTSYS
29 PIC,ION      000000 000000 000000 170000 A-OP. IS 17. USED TO GIVE "ION" CMD. TO INTSYS
30 PIC,MSTMSK   000000 000000 000000 130000 A-OP. IS 13. USED TO GIVE "MASKED SET MASK" CMD. TO INTSYS
31 PIC,RVECT    000000 000000 000000 050000 A-OP. IS 5. USED TO GIVE "READ-VECTOR" COMMAND TO INTSYS
32 A,0          000000 000000 000000 000000 A-OP. IS 0
```

```
33 A,1          000000 000000 000000 010000 A-OP. IS 1
34 A,2          000000 000000 000000 020000 A-OP. IS 2
35 A,3          000000 000000 000000 030000 A-OP. IS 3
36 A,4          000000 000000 000000 040000 A-OP. IS 4
37 A,5          000000 000000 000000 050000 A-OP. IS 5
38 A,6          000000 000000 000000 060000 A-OP. IS 6
39 A,7          000000 000000 000000 070000 A-OP. IS 7
40 A,10         000000 000000 000000 100000 A-OP. IS 10
41 A,11         000000 000000 000000 110000 A-OP. IS 11
42 A,12         000000 000000 000000 120000 A-OP. IS 12
43 A,13         000000 000000 000000 130000 A-OP. IS 13
44 A,14         000000 000000 000000 140000 A-OP. IS 14
45 A,15         000000 000000 000000 150000 A-OP. IS 15
46 A,16         000000 000000 000000 160000 A-OP. IS 16
47 A,17         000000 000000 000000 170000 A-OP. IS 17
48 B,Z          000000 000000 000000 000000 B-OP. IS 0. USED TO ADDR. ZERO REG., STATUS OR SCRATCH
49 B,D          000000 000000 000001 000000 B-OP. IS 1. USED TO ADDR. THE D-REGISTER
50 B,P          000000 000000 004002 000000 B-OP. IS 2. USED TO ADDR. THE P-REGISTER
51 B,B          000000 000000 004003 000000 B-OP. IS 3. USED TO ADDR. THE B-REGISTER
52 B,L          000000 000000 000004 000000 B-OP. IS 4. USED TO ADDR. THE L-REGISTER
53 B,A          000000 000000 000005 000000 B-OP. IS 5. USED TO ADDR. THE A-REGISTER
54 B,T          000000 000000 000006 000000 B-OP. IS 6. USED TO ADDR. THE T-REGISTER
55 B,X          000000 000000 004007 000000 B-OP. IS 7. USED TO ADDR. THE X-REGISTER
56 B,STS        000000 000000 000010 000000 B-OP. IS 10. USED TO ADDR. REG. 10, STATUS OR SCRATCH
57 B,R1         000000 000000 000011 000000 B-OP. IS 11. USED TO ADDR. REG. 11, ADDR. OR SCRATCH
58 B,R2         000000 000000 000012 000000 B-OP. IS 12. USED TO ADDR. REG. 12, SCRATCH
59 B,R3         000000 000000 000013 000000 B-OP. IS 13. USED TO ADDR. REG. 13, SCRATCH
60 B,R4         000000 000000 000014 000000 B-OP. IS 14. USED TO ADDR. REG. 14, SCRATCH
61 B,R5         000000 000000 000015 000000 B-OP. IS 15. USED TO ADDR. REG. 15, SCRATCH
62 B,R6         000000 000000 000016 000000 B-OP. IS 16. USED TO ADDR. REG. 16, SCRATCH
63 B,R7         000000 000000 000017 000000 B-OP. IS 17. USED TO ADDR. REG. 17, SCRATCH
64 B,DEST       000001 000000 000000 000000 B-OP. IS VALUE OF "INSTR. BITS 0-2" LOADED BY LAST "T,MAPJ"
```

```
65 B,SRCE          000002 000000 000000 000000 B-OP. IS VALUE OF "INSTR. BITS 3-5" LOADED BY LAST "T,MAPJ"
66 B,LC            000003 000000 000000 000000 B-OP. IS VALUE OF THE "LOOP COUNTER" TWO INSTRUCTIONS AGO
67 B,0             000000 000000 000000 000000 B-OP. IS 0
68 B,1             000000 000000 000001 000000 B-OP. IS 1
69 B,2             000000 000000 000002 000000 B-OP. IS 2
70 B,3             000000 000000 000003 000000 B-OP. IS 3
71 B,4             000000 000000 000004 000000 B-OP. IS 4
72 B,5             000000 000000 000005 000000 B-OP. IS 5
73 B,6             000000 000000 000006 000000 B-OP. IS 6
74 B,7             000000 000000 000007 000000 B-OP. IS 7
75 B,10            000000 000000 000010 000000 B-OP.. IS 10
76 B,11            000000 000000 000011 000000 B-OP. IS 11
77 B,12            000000 000000 000012 000000 B-OP. IS 12
78 B,13            000000 000000 000013 000000 B-OP. IS 13
79 B,14            000000 000000 000014 000000 B-OP. IS 14
80 B,15            000000 000000 000015 000000 B-OP. IS 15
81 B,16            000000 000000 000016 000000 B-OP. IS 16
82 B,17            000000 000000 000017 000000 B-OP. IS 17
83 ALUF,ANDAQ      010000 000000 000000 000000 A /\ Q -> F
84 ALUF,ANDAB      010200 000000 000000 000000 A /\ B -> F
85 ALUF,ANDDA      011200 000000 000000 000000 D /\ A -> F
86 ALUF,ANDDQ      011400 000000 000000 000000 D /\ Q -> F
87 ALUF,ORAQ       006000 000000 000000 000000 A \/ Q -> F
88 ALUF,ORAB       006200 000000 000000 000000 A \/ B -> F
89 ALUF,ORDA       007200 000000 000000 000000 D \/ A -> F
90 ALUF,ORDQ       007400 000000 000000 000000 D \/ Q -> F
91 ALUF,XORAQ      014000 000000 000000 000000 A XOR Q -> F
92 ALUF,XORAB      014200 000000 000000 000000 A XOR B -> F
93 ALUF,XORDA      015200 000000 000000 000000 D XOR A -> F
94 ALUF,XORDQ      015400 000000 000000 000000 D XOR Q -> F
95 ALUF,XNORAQ     016000 000000 000000 000000 NOT(A XOR Q) -> F
96 ALUF,XNORAB     016200 000000 000000 000000 NOT(A XOR B) -> F
```

```
 97 ALUF,XNORDA      017200 000000 000000 000000 NOT(D XOR A) -> F
 98 ALUF,XNORDQ      017400 000000 000000 000000 NOT(D XOR Q) -> F
 99 ALUF,ZERO        010400 000000 000000 000000 0 -> F
100 ALUF,INVQ        016400 000000 000000 000000 NOT(Q) -> F
101 ALUF,INVB        016600 000000 000000 000000 NOT(B) -> F
102 ALUF,INVA        017000 000000 000000 000000 NOT(A) -> F
103 ALUF,INVD        017600 000000 000000 000000 NOT(D) -> F
104 ALUF,PASSQ       014400 000000 000000 000000 Q -> F
105 ALUF,PASSB       014600 000000 000000 000000 B -> F
106 ALUF,PASSA       015000 000000 000000 000000 A -> F
107 ALUF,PASSD       015600 000000 000000 000000 D -> F
108 ALUF,MASKAQ      012000 000000 000000 000000 NOT(A) /\ Q -> F
109 ALUF,MASKAB      012200 000000 000000 000000 NOT(A) /\ B -> F
110 ALUF,MASKDA      013200 000000 000000 000000 NOT(D) /\ A -> F
111 ALUF,MASKDQ      013400 000000 000000 000000 NOT(D) /\ Q -> F
112 ALUF,A+Q         000000 000000 000000 000000 A + Q -> F
113 ALUF,A+B         000200 000000 000000 000000 A + B -> F
114 ALUF,D+A         001200 000000 000000 000000 D + A -> F
115 ALUF,D+Q         001400 000000 000000 000000 D + Q -> F
116 ALUF,A+Q+1       000000 040000 000000 000000 A + Q + 1 -> F
117 ALUF,A+B+1       000200 040000 000000 000000 A + B + 1 -> F
118 ALUF,D+A+1       001200 040000 000000 000000 D + A + 1 -> F
119 ALUF,D+Q+1       001400 040000 000000 000000 D + Q + 1 -> F
120 ALUF,-Q-1        004400 000000 000000 000000 -Q - 1 -> F
121 ALUF,-B-1        004600 000000 000000 000000 -B - 1 -> F
122 ALUF,-A-1        005000 000000 000000 000000 -A - 1 -> F
123 ALUF,-D-1        003600 000000 000000 000000 -D - 1 -> F
124 ALUF,Q-A-1       002000 000000 000000 000000 Q - A - 1 -> F
125 ALUF,B-A-1       002200 000000 000000 000000 B - A - 1 -> F
126 ALUF,A-D-1       003200 000000 000000 000000 A - D - 1 -> F
127 ALUF,Q-D-1       003400 000000 000000 000000 Q - D - 1 -> F
128 ALUF,A-Q-1       004000 000000 000000 000000 A - Q - 1 -> F
```

```
129 ALUF,A-B-1      004200 000000 000000 000000 A - B - 1 -> F
130 ALUF,D-A-1      005200 000000 000000 000000 D - A - 1 -> F
131 ALUF,D-Q-1      005400 000000 000000 000000 D - Q - 1 -> F
132 ALUF,Q          000400 000000 000000 000000 Q -> F
133 ALUF,B          000600 000000 000000 000000 B -> F
134 ALUF,A          001000 000000 000000 000000 A -> F
135 ALUF,D          001600 000000 000000 000000 D -> F
136 ALUF,Q+1        000400 040000 000000 000000 Q + 1 -> F
137 ALUF,B+1        000600 040000 000000 000000 B + 1 -> F
138 ALUF,A+1        001000 040000 000000 000000 A + 1 -> F
139 ALUF,D+1        001600 040000 000000 000000 D + 1 -> F
140 ALUF,Q-1        002400 000000 000000 000000 Q - 1 -> F
141 ALUF,B-1        002600 000000 000000 000000 B - 1 -> F
142 ALUF,A-1        003000 000000 000000 000000 A - 1 -> F
143 ALUF,D-1        005600 000000 000000 000000 D - 1 -> F
144 ALUF,-Q         004400 040000 000000 000000 -Q -> F
145 ALUF,-B         004600 040000 000000 000000 -B -> F
146 ALUF,-A         005000 040000 000000 000000 -A -> F
147 ALUF,-D         003600 040000 000000 000000 -D -> F
148 ALUF,Q-A        002000 040000 000000 000000 Q - A -> F
149 ALUF,B-A        002200 040000 000000 000000 B - A -> F
150 ALUF,A-D        003200 040000 000000 000000 A - D -> F
151 ALUF,Q-D        003400 040000 000000 000000 Q - D -> F
152 ALUF,A-Q        004000 040000 000000 000000 A - Q -> F
153 ALUF,A-B        004200 040000 000000 000000 A - B -> F
154 ALUF,D-A        005200 040000 000000 000000 D - A -> F
155 ALUF,D-Q        005400 040000 000000 000000 D - Q -> F
156 ALUD,Q          000000 000000 000000 000000 F -> Q ; F -> Y
157 ALUD,NONE       020000 000000 000000 000000 F -> Y
158 ALUD,B,YA       040000 000000 000000 000000 F -> B ; A -> Y
159 ALUD,B          060000 000000 000000 000000 F -> B ; F -> Y
160 ALUD,SRD        100000 000000 000000 000000 F -> Y ; (F,Q)/2 -> (B,Q)
```

```
161 ALUD,SRB        120000 000000 000000 000000 F -> Y ; F/2 -> B
162 ALUD,SLD        140000 000000 000000 000000 F -> Y ; (F,Q)*2 -> (B,Q)
163 ALUD,SLB        160000 000000 000000 000000 F -> Y ; F*2 -> B
164 STS,LO          000140 000000 000000 000000 IDB-BITS 0-7 -> STATUS-BITS 0-7
165 STS,EA          000040 000000 000000 000000 CRY -> STS C ; OVF -> STS Q ; OVF \/ STS O -> STS O
166 STS,ES          000100 000000 000000 000000 ALU SHIFT OUTPUT -> STATUS M
167 CRY,C           000000 100000 000000 000000 STATUS C -> CARRY IN
168 CRY,GPR         000000 140000 000000 000000 GPR BIT 0 -> CARRY IN
169 ALUM,FMU        000000 010000 000000 000000 MULTIPLY ALU MODE. GPR 0 -> ALU-INSTR 1 ; RIGHT GPR-SHIFT
170 ALUM,FDV        000000 020000 000000 000000 DIVISION ALU MODE. GPR 0 -> ALU-INSTR 3 ; LEFT GPR-SHIFT
171 ALUM,IR         000000 030000 000000 000000 SHIFT INSTR. MODE. SH MODE FROM IR-BITS. M IS SET AUTOM.
172 ALUM,MIC        000000 000000 000000 000000 MICROPROGRAM CONTR. SHIFT. SHIFT MODE FROM "MIS-BITS"
173 MIS,ROT         000000 002000 000000 000000 SPECIFIES ROTATIONAL SHIFT IF "ALUM,MIC"
174 MIS,ZIN         000000 004000 000000 000000 SPECIFIES ZERO-END-INPUT SHIFT IF "ALUM,MIC"
175 MIS,LIN         000000 006000 000000 000000 SPECIFIES LINK-END-INPUT SHIFT IF "ALUM,MIC"
176 LCOUNT          000000 000000 000100 000000 COUNT LOOP-COUNTER ; IF FALSE "RETURN"&"HOLD"
177 CONDENABL       000000 000000 000200 000000 ENABLE COND. SEQ., USE "FALSE" SPECS IF CONDITION FALSE
178 IDBS,ALU        000000 000000 000000 000000 ARITHMETIC-LOGIC-UNIT -> IDB
179 IDBS,BMG        000000 000040 000000 000000 BIT-MASK-GENERATOR -> IDB
180 IDBS,GPR        000000 000100 000000 000000 GENERAL-PURPOSE-REGISTER -> IDB
181 IDBS,DBR        000000 000140 000000 000000 DATA-BUS-REGISTER -> IDB
182 IDBS,ARG        000000 000200 000000 000000 ARGUMENT (MICRO-INSTRUCTION-BITS 0-15) -> IDB
183 IDBS,REG        000000 000240 006000 000000 REGISTER-FILE -> IDB
184 IDBS,STS        000000 000300 000000 000000 STATUS -> IDB
185 IDBS,BARG       000000 000400 000000 000000 B-OPERAND-ARGUMENT (0-17) -> IDB
186 IDBS,SWAP       000000 000440 000000 000000 BYTE-SWAP OF LAST IDB -> IDB
187 IDBS,PEA        000000 000500 006000 000000 PEA-REGISTER -> IDB
188 IDBS,PES        000000 000540 006000 000000 PES-REGISTER -> IDB
189 IDBS,AARG       000000 000600 000000 000000 A-OPERAND-ARGUMENT*10 (0-170) -> IDB
190 IDBS,PIC        000000 000640 006000 000000 PRIORITY-INTERRUPT-CONTROL STATUS REGISTER -> IDB
191 IDBS,IOR        000000 000700 006000 000000 UART DATA AND STATUS -> IDB
192 IDBS,DSABL      000000 000740 000000 000000 DISABLE IDBS (USED TO READ "PIC"-INFO, EXCEPT PIC-STS)
```

```
193 IDBS,MIPANS      000000 001000 006000 000000 IR 0 USED BY MOPC
194 IDBS,MAPANS      000000 001040 006000 000000 IR 0 USED BY TRA-INSTRUCTION
195 IDBS,GPR,SEXT    000000 001100 000000 000000 GPR BITS 0-7 (WITH BITS 8-15 EQUAL TO BIT 7) -> IDB
196 IDBS,PGS         000000 001140 006000 000000 PAGING STATUS ACCORDING TO LAST REQUEST (NEVER LOCKED)
197 IDBS,CSR         000000 001200 006000 000000 CACHE STATUS REGISTER
198 IDBS,PCR         000000 001240 006000 000000 READ PAGING CONTROL REGISTER (NUMBER SELECTED BY PIL)
199 IDBS,ALD         000000 001300 006000 000000 AUTOMATIC LOAD DESCRIPTOR & PRINT STATUS
200 IDBS,PANEL       000000 001340 006000 000000 PANEL VECTOR
201 IDBS,RCS         000000 001400 000000 000000 READ CONTROL STORE
202 IDBS,PICVC       000000 001440 006000 000000 INTERRUPT VECTOR
203 IDBS,LBR         000000 001500 006000 000000 READ LBR FROM LA BUS (MUST HAVE COMM,PLBR)
204 NOT USED         000000 000000 000000 000000 NOT USED
205 COMM,LDPIL       000000 000001 000000 000000 IDB BITS 8-15 -> STATUS BITS 8-15
206 COMM,LDGPR       000000 000002 000000 000000 IDB -> GENERAL-PURPOSE-REGISTER
207 COMM,EWRF        000000 000003 004000 000000 IDB -> REGISTER FILE WORD ADDRESSED BY A-OP AND B-OP
208 COMM,CLIRQ       000000 000004 000000 000000 PREVENT JUMP TO INTR VECT, REMOVE "PANEL"-EFF ON IRQ-TEST
209 COMM,RSDA        000000 000005 000000 000000 RESET DATA AVAILABLE IN "UART"
210 COMM,TBSTR       000000 000006 000000 000000 "TRANSMIT BUFFER STROBE" TO "UART"  IDB 0-7 -> "UART"
211 NOT USED         000000 000000 000000 000000 NOT USED
212 COMM,LDPANC      000000 004006 000000 000000 SEND ONE BYTE TO IR 0
213 COMM,LDPCR       000000 006006 000000 000000 LOAD PCR (NUMBER DETERMINED BY PIL)
214 COMM,SIOC        000000 000007 000000 000000 LOAD "I/O-CONTROL" REGISTER (TERMINAL-1, CLK-INTS, ETC.)
215 COMM,SLOW1       000000 000010 000000 000000 SLOW CYCLE
216 COMM,SLOW2       000000 000010 000000 000000 SLOW CYCLE
217 COMM,EPIC        000000 000011 000000 000000 A-OPERAND IS AN INSTRUCTION TO "PIC" (PRIORITY INT CONTR)
218 COMM,SMPID       000000 000012 006000 000000 SET MICRO-PID. PID-BITS WHERE IDB IS "1" ARE FORCED TO "1"
219 COMM,START       000000 000013 000000 000000 RESET THE "STOP" FLIP-FLOP
220 COMM,SSTOP       000000 000014 000000 000000 SET THE "STOP" FLIP-FLOP
221 COMM,CLRTC       000000 000015 000000 000000 CLEAR THE 20 MS CLOCK FLIP-FLOP
222 COMM,CLFF        000000 000016 000000 000000 CLEAR THE "OOD" AND THE "DZD" FLIP-FLOPS
223 COMM,LDLC        000000 000017 004000 000000 LOAD THE "LOOP COUNTER" WITH THE 6 LOWER IDB-BITS
224 COMM,LDSEG       000000 000020 004040 000000 LOAD SEGMENT REGISTER (SELECTS PHYSICAL 64K AREA)
```

```
225 COMM,LDDOMI      000000 002020 004040 000000 LOAD SINTRAN-4 DOMAIN REGISTER
226 COMM,LDPS        000000 004020 004040 000000 LOAD SINTRAN-4 PS REGISTER
227 COMM,LDIRV       000000 006020 004040 000000 LOAD INSTRUCTION REGISTER FOR OR-LOGIC USE
228 COMM,WCIHM       000000 000021 004040 000000 WRITE CACHE INHIBIT MAP (BIT 15 =1 IS NORMAL, =0 IS INH)
229 COMM,SSEMA       000000 002021 000040 000000 THE NEXT REQUEST IS A SEMAPHOR REQUEST
230 COMM,CCLR        000000 004021 000040 000000 CLEAR CACHE (BIT 3 OF CSR MUST BE 0 BEFORE START)
231 COMM,LDEXM       000000 006021 004040 000000 LOAD EXAMINE MODE
232 COMM,IREAD,PT    000000 000022 000040 000000 INDIRECT ADDRESS READ (PT-RELATIVE)
233 COMM,IREAD,APT   000000 002022 000040 000000 INDIRECT ADDRESS READ (APT-RELATIVE)
234 COMM,MAP         000000 004022 000040 000000 USE IDB AS INSTRUCTION (AS IN EXR)
235 COMM,CNEXT,NWP   000000 006022 004040 000000 EXECUTE NEXT INSTRUCTION IF P NOT CHANGED IN LAST CYCLE
236 COMM,CJMP,F15    000000 000023 004040 000000 JUMP IF F15 =1
237 COMM,CJMP,NF15   000000 002023 004040 000000 JUMP IF F15 =0
238 COMM,CJMP,F=0    000000 004023 004040 000000 JUMP IF F =0
239 COMM,CJMP,NF=0   000000 006023 004040 000000 JUMP IF F  <> 0
240 COMM,CNEXT,SGR   000000 000024 004040 000000 SKIP LST
241 COMM,CNEXT,NSGR  000000 002024 004040 000000 SKIP GRE
242 COMM,CNEXT,CRY   000000 004024 004040 000000 SKIP MLS
243 COMM,CNEXT,NCRY  000000 006024 004040 000000 SKIP MGR
244 COMM,CNEXT,F15   000000 000025 004040 000000 SKIP GEQ
245 COMM,CNEXT,NF15  000000 002025 004040 000000 SKIP LSS
246 COMM,CNEXT,F=0   000000 004025 004040 000000 SKIP UEQ
247 COMM,CNEXT,NF=0  000000 006025 004040 000000 SKIP EQL
248 COMM,JMP,*       000000 000026 002040 000000 JMP P-RELATIVE
249 COMM,JMP,B       000000 002026 002040 000000 JMP ,B
250 COMM,JMP,I       000000 004026 002040 000000 JMP I & I,B
251 COMM,JMP,X       000000 006026 002040 000000 JMP ,X
252 COMM,JMP,XB      000000 000027 002040 000000 JMP ,X,B
253 COMM,JMP,XI      000000 002027 002040 000000 JMP I,X & I,B,X
254 COMM,LBCONT      000000 004027 000040 000000 USED IN PREX-INSTRUCTION ONLY
255 COMM,CONTINUE    000000 006027 000040 000000 FETCH NEW INSTRUCTION RELATIVE TO P
256 COMM,AREAD,*     000000 000030 000040 000000 READ P-RELATIVE
```

```
257 COMM,AREAD,B      000000 002030 000040 000000 READ ,B
258 COMM,AREAD,I      000000 004030 000040 000000 READ I & I,B
259 COMM,AREAD,X      000000 006030 000040 000000 READ ,X
260 COMM,AREAD,XB     000000 000031 000040 000000 READ ,B,X
261 COMM,AREAD,XI     000000 002031 000040 000000 READ I,X & I,B,X
262 COMM,IREAD2       000000 004031 000040 000000 READ FIRST PART OF 32-BIT INDIRECT ADDRESS
263 COMM,AREAD,NEXT   000000 006031 000040 000000 READ NEXT ADDRESS
264 COMM,AWRITE,*     000000 000032 000040 000000 WRITE P-RELATIVE
265 COMM,AWRITE,B     000000 002032 000040 000000 WRITE ,B
266 COMM,AWRITE,I     000000 004032 000040 000000 WRITE I & I,B
267 COMM,AWRITE,X     000000 006032 000040 000000 WRITE ,X
268 COMM,AWRITE,XB    000000 000033 000040 000000 WRITE ,X,B
269 COMM,AWRITE,XI    000000 002033 000040 000000 WRITE I,X & I,X,B
270 COMM,AWRITE,HOLD  000000 004033 000040 000000 WRITE IN LAST ADDRESS
271 COMM,AWRITE,NEXT  000000 006033 000040 000000 WRITE IN NEXT ADDRESS
272 COMM,RDRQ,PT      000000 000034 000040 000000 READ PAGE-TABLE-RELATIVE, ADDRESS FROM IDB
273 COMM,RDRQ,APT     000000 002034 000040 000000 READ ALTERNATIVE-PT-RELATIVE, ADDRESS FROM IDB
274 COMM,RDRQ,HOLD    000000 004034 000040 000000 READ FROM LAST USED PT, ADDRESS FROM IDB
275 COMM,EXRQ         000000 006034 000040 000000 EXAMINE
276 COMM,WRRQ,PT      000000 000035 000040 000000 WRITE PT-RELATIVE, ADDRESS FROM IDB
277 COMM,WRRQ,APT     000000 002035 000040 000000 WRITE APT-RELATIVE, ADDRESS FROM IDB
278 COMM,WRRQ,HOLD    000000 004035 000040 000000 WRITE IN LAST USED PT, ADDRESS FROM IDB
279 COMM,DERQ         000000 006035 000040 000000 DEPOSIT
280 COMM,ADCS         000000 000036 004040 000000 SET CONTROL STORE ADDRESS
281 COMM,RWCS         000000 002036 000040 000000 READ OR WRITE IN CONTROL STORE
282 COMM,WLBR         000000 004036 004040 000000 WRITE LOGICAL BANK REGISTER
283 COMM,WALBR        000000 006036 004040 000000 WRITE ALTERNATIVE LOGICAL BANK REGISTER
284 NOT USED          000000 000000 000000 000000 NOT USED
285 COMM,IDENT        000000 004037 000040 000000 IDENT BUS REQUEST
286 COMM,IOX          000000 006037 000040 000000 IOX BUS REQUEST
287 T,JMP             000000 000000 000000 000000 TRUE SEQUENCE IS JUMP
288 T,JMP0-3          000000 000000 001000 000000 TRUE SEQ IS JUMP. IR0-3 CONTROLS THE 4 LOWER JUMP ADDR BITS
```

```
289 T,JMPAOPR       000000 006000 001000 000000 TRUE SEQ IS JUMP. A-OP CONTROLS THE 4 LOWER JUMP ADDR BITS
290 T,RETURN        000000 000000 040000 000000 TRUE SEQUENCE IS RETURN
291 T,NEXT          000000 000000 100000 000000 TRUE SEQUENCE IS NEXT
292 T,HOLD          000000 000000 000000 000000 TRUE STACK OPERATION IS HOLD
293 T,POP           000000 000000 010000 000000 TRUE STACK OPERATION IS POP
294 T,LOAD          000000 000000 020000 000000 TRUE STACK OPERATION IS LOAD
295 T,PUSH          000000 000000 030000 000000 TRUE STACK OPERATION IS PUSH
296 F,JMP           000000 000000 000000 000000 FALSE SEQUENCE IS JUMP. USED WITH A CONDITION SETTING
297 F,RETURN        000000 000000 000000 000004 FALSE SEQUENCE IS RETURN. USED WITH A CONDITION SETTING
298 F,NEXT          000000 000000 000000 000010 FALSE SEQUENCE IS NEXT. USED WITH A CONDITION SETTING
299 F,HOLD          000000 000000 000000 000000 FALSE STACK OPERATION IS HOLD. USED WITH A COND SETTING
300 F,POP           000000 000000 000000 000001 FALSE STACK OPERATION IS POP. USED WITH A COND SETTING
301 F,LOAD          000000 000000 000000 000002 FALSE STACK OPERATION IS LOAD. USED WITH A COND SETTING
302 F,PUSH          000000 000000 000000 000003 FALSE STACK OPERATION IS PUSH. USED WITH A COND SETTING
303 COND,STP        000000 000000 000400 000360 CONDITION FOR TESTING IS  "STOP"
304 COND,PXM        000000 000000 000400 000220 CONDITION FOR TESTING IS PREFIX-MODE
305 COND,OVF        000000 000000 000400 000240 CONDITION FOR TESTING IS "OVERFLOW" FROM ALU
306 COND,CRY        000000 000000 000400 000260 CONDITION FOR TESTING IS "CARRY" FROM ALU
307 COND,F11        000000 000000 000400 000300 CONDITION FOR TESTING IS BIT 11 FROM ALU
308 COND,F15        000000 000000 000400 000320 CONDITION FOR TESTING IS BIT 15 FROM ALU
309 COND,F=0        000000 000000 000400 000340 CONDITION FOR TESTING IS "ALL ZEROS" FROM ALU
310 COND,LC=0       000000 000000 000400 000020 CONDITION FOR TESTING IS "LOOP-COUNTER" CONTENT =0
311 COND,FETCH      000000 000000 000400 000100 CONDITION FOR TESTING IS LAST MEMORY REQUEST WAS FETCH
312 COND,IRQ        000000 000000 000400 000040 COND FOR TEST IS "IRQ". CHECKS LEV 10-15 IF "COMM,CLIRQ"
313 COND,RESTR      000000 000000 000400 000060 COND FOR TESTING IS "RESTRICTED-MODE". TRUE IF RING 0-1
314 COND,OOD        000000 000000 000400 000120 CONDITION FOR TESTING IS THE "ONE-OUT-DETECT" FLIP-FLOP
315 COND,DZD        000000 000000 000400 000000 COND FOR TESTING IS THE "DOUBLE-ZERO-DETECT" FLIP-FLOP
316 COND,COND       000000 000000 000400 000160 COND FOR TEST IS OUTCOME OF LATEST TEST (DELAYS TESTS)
317 AB,CDIGI        000000 000000 000011 170000 COUNTER CONTROLLING NUMBER OF DIGITS IN AN OCTAL NUMBER
318 AB,UPPNR        000000 000000 000012 130000 SCRATCH WORD KEEPING UPPER ADDR LIMIT IN MOPC DUMP CMND
319 AB,CURNR        000000 000000 000013 130000 SCRATCH WORD KEEPING CURRENT ADDR IN A MOPC DUMP COMMAND
320 AB,CNT10        000000 000000 000017 170000 COUNTER CONTROLLING NO. OF OCTAL NOS/LINE IN MOPC DUMP CMND
```

```
321 AB,PRCHR        000000 000000 000016 160000 SCRATCH WORD : THE NEXT CHARACTER TO BE WRITTEN BY MOPC
322 AB,TXT1         000000 000000 000014 160000 SCRATCH WORD : DISPLAY TEXT
323 AB,TXT2         000000 000000 000013 160000 SCRATCH WORD : DISPLAY TEXT
324 AB,SCRAM        000000 000000 000015 140000 SCR WD : SCRAMBLED REPRESENTATION OF LETTERS IN MOPC-INPUT
325 AB,OCTNR        000000 000000 000015 150000 SCRATCH WORD : OCTAL NUMBER ASSEMBLED FROM MOPC-INPUT
326 AB,DISPL        000000 000000 000015 160000 TYPE OF RUNNING DISPLAY
327 AB,OCTAD        000000 000000 000013 150000 ADDRESS OF RUNNING DISPLAY
328 AB,OCTA2        000000 000000 000012 160000 WORD TO EXTEND ADDRESS IN OCTAD TO 24 BITS
329 AB,DEPOS        000000 000000 000017 150000 SOME OCTAL DIGIT WAS WRITTEN SINCE LAST COMMAND TERMINATED
330 AB,DUMPF        000000 000000 000016 150000 SCRATCH WORD INDICATING THAT A DUMP IS IN PROGRESS
331 AB,RONLY        000000 000000 000014 150000 THE EXAMINED REGISTER IS READ-ONLY
332 AB,WRTYP        000000 000000 000012 150000 TYPE OF VARIABLE IN CASE OF DEPOSIT
333 AB,WRADR        000000 000000 000011 150000 ADDRESS OF VARIABLE IN CASE OF DEPOSIT
334 AB,ACTLV        000000 000000 000017 140000 SCRATCH WORD HOLDING "ACTIVE LEVELS"
335 AB,PVL          000000 000000 000015 170000 SCRATCH WORD HOLDING "PREVIOUS LEVEL"
336 AB,IIE          000000 000000 000012 170000 SCR WORD HOLDING "PIC" REPRESENTATION OF LAST IIE SETTING
337 AB,PID          000000 000000 000014 170000 SCR WORD HOLDING MICROPROGRAM-KNOWN BITS OF PID REGISTER
338 AB,PIE          000000 000000 000013 170000 SCRATCH WORD HOLDING THE "PIE" REGISTER
339 AB,STATUS       000000 000000 000017 160000 SCRATCH WORD HOLDING THE LATEST "COMM,SIOC" INFORMATION
340 AB,OCTN2        000000 000000 000015 130000 SCRATCH WORD EXPANDING THE "AB,OCTNR" TO 24 BITS
341 AB,NUMBR        000000 000000 000012 120000 SCRATCH WORD HOLDING AN OCTAL NUMBER BEING PRINTED BY MOPC
342 AB,OPR          000000 000000 000011 160000 SCRATCH WORD HOLDING THE "OPR" REGISTER VALUE
343 AB,BRKPT        000000 000000 000016 140000 SCRATCH WORD HOLDING BREAKPOINT ADDRESS
344 AB,SINGL        000000 000000 000014 140000 SCRATCH WORD COUNTING SINGLE-INSTRUCTION
345 AB,BPFLG        000000 000000 000013 140000 SCRATCH WORD INDICATING THAT BREAKPOINT IS ON
346 AB,MACL         000000 000000 000011 140000 SCRATCH WORD HOLDING RETRY COUNTER FOR LOAD AFTER MACL
347 AB,LMP          000000 000000 000012 140000 SCRATCH WORD HOLDING THE "LMP" REGISTER VALUE
348 AB,EXMOD        000000 000000 000011 130000 SCRATCH WORD HOLDING THE "EXM" REGISTER
349 AB,MANIR        000000 000000 000014 130000 SCRATCH WORD HOLDING FLAG FOR MANUAL IR
350 AB,SSAVE        000000 000000 000016 130000 SCRATCH WORD HOLDING STS DURING DECIMAL INTRUCTIONS
351 XRF             000004 000000 000000 000000 SELECT EXTENDED REGISTER FILE
352 AB,UCIL         000000 000000 000014 120000 SCRATCH WORD HOLDING CACHE INHIBIT UPPER LIMIT
```

```
353 AB,LCIL          000000 000000 000013 120000 SCRATCH REGISTER HOLDING CACHE INHIBIT LOWER LIMIT
354 IDBS,LA          000000 001600 000000 000000 LOGICAL ADDRESS USED BY "ALU" WHEN JUMP
355 AB,PGS           000000 000000 000015 120000 COPY OF PGS FROM LAST PF/PV
356 IDBS,INR         000000 001640 006000 000000 READ INSTALLATION NUMBER
357 AB,STBNK         000000 000000 000011 120000 GLOBAL BANK POINTER TO THE SEGMENT TABLE BANK
358 AB,STSRT         000000 000000 000016 120000 GLOBAL POINTER TO THE SEGMENT TABLE WITHIN THE BANK
359 AB,CMBNK         000000 000000 000017 120000 GLOBAL POINTER TO THE CORE MAP TABLE BANK
```

APPENDIX B MNEMONICS LIST FOR THE ND-120

The following pages present the list of mnemonics for the ND-120 microinstruction set.

The list includes the op-code for the microinstruction (in octal), and a one-line description of it.

```
 1 A,Z          000000 000000 000000 000000 A-OP. IS 0. USED TO ADDR. ZERO REG., STATUS OR SCRATCH
 2 A,D          000000 000000 000000 010000 A-OP. IS 1. USED TO ADDR. THE D-REGISTER
 3 A,P          000000 000000 000000 020000 A-OP. IS 2. USED TO ADDR. THE P-REGISTER
 4 A,B          000000 000000 000000 030000 A-OP. IS 3. USED TO ADDR. THE B-REGISTER
 5 A,L          000000 000000 000000 040000 A-OP. IS 4. USED TO ADDR. THE L-REGISTER
 6 A,A          000000 000000 000000 050000 A-OP. IS 5. USED TO ADDR. THE A-REGISTER
 7 A,T          000000 000000 000000 060000 A-OP. IS 6. USED TO ADDR. THE T-REGISTER
 8 A,X          000000 000000 000000 070000 A-OP. IS 7. USED TO ADDR. THE X-REGISTER
 9 A,STS        000000 000000 000000 100000 A-OP. IS 10. USED TO ADDR. REG. 10, STATUS OR SCRATCH
10 A,R1         000000 000000 000000 110000 A-OP. IS 11. USED TO ADDR. REG. 11, ADDRESS OR SCRATCH
11 A,R2         000000 000000 000000 120000 A-OP. IS 12. USED TO ADDR. REGISTER 12, SCRATCH
12 A,R3         000000 000000 000000 130000 A-OP. IS 13. USED TO ADDR. REGISTER 13, SCRATCH
13 A,R4         000000 000000 000000 140000 A-OP. IS 14. USED TO ADDR. REGISTER 14, SCRATCH
14 A,R5         000000 000000 000000 150000 A-OP. IS 15. USED TO ADDR. REGISTER 15, SCRATCH
15 A,R6         000000 000000 000000 160000 A-OP. IS 16. USED TO ADDR. REGISTER 16, SCRATCH
16 A,R7         000000 000000 000000 170000 A-OP. IS 17. USED TO ADDR. REGISTER 17, SCRATCH
17 A,PIL        000010 000000 000000 000000 A-OP. IS VALUE IN "PIL" REGISTER, LOADED BY "COMM,LDPIL"
18 A,REG        000020 000000 000000 000000 A-OP. IS VALUE OF "INSTR. BITS 3-6" LOADED BY LAST "T,MAPJ"
19 A,SRCE       000024 000000 000000 000000 A-OP. IS VALUE OF "INSTR. BITS 3-5" LOADED BY LAST "T,MAPJ"
20 A,LC         000030 000000 000000 000000 A-OP. IS VALUE OF "LOOP COUNTER" TWO MICROINSTRUCTIONS AGO
21 PIC,RSTS     000000 000000 000000 060000 A-OP. IS 6. USED TO GIVE "READ STATUS" CMD. TO INTSYS
22 PIC,MCL      000000 000000 000000 000000 A-OP. IS 0. USED TO GIVE "MASTER CLEAR" CMD. TO INTSYS
23 PIC,MCLPID   000000 000000 000000 020000 A-OP. IS 2. USED TO GIVE "MASKED CLEAR PID" CMD. TO INTSYS
24 PIC,RMSK     000000 000000 000000 070000 A-OP. IS 7. USED TO GIVE "READ MASK" CMD. TO INTSYS
25 PIC,LOSTS    000000 000000 000000 110000 A-OP. IS 11. USED TO GIVE "LOAD STATUS" CMD. TO INTSYS
26 PIC,LMSK     000000 000000 000000 160000 A-OP. IS 16. USED TO GIVE "LOAD MASK" CMD. TO INTSYS
27 PIC,MCLMSK   000000 000000 000000 120000 A-OP. IS 12. USED TO GIVE "MASKED CLEAR MASK" CMD TO INTSYS
28 PIC,IOF      000000 000000 000000 150000 A-OP. IS 15. USED TO GIVE "IOF" CMD. TO INTSYS
29 PIC,ION      000000 000000 000000 170000 A-OP. IS 17. USED TO GIVE "ION" CMD. TO INTSYS
30 PIC,MSTMSK   000000 000000 000000 130000 A-OP. IS 13. USED TO GIVE "MASKED SET MASK" CMD. TO INTSYS
31 PIC,RVECT    000000 000000 000000 050000 A-OP. IS 5. USED TO GIVE "READ-VECTOR" COMMAND TO INTSYS
32 A,0          000000 000000 000000 000000 A-OP. IS 0
```

```
33 A,1          000000 000000 000000 010000 A-OP. IS 1
34 A,2          000000 000000 000000 020000 A-OP. IS 2
35 A,3          000000 000000 000000 030000 A-OP. IS 3
36 A,4          000000 000000 000000 040000 A-OP. IS 4
37 A,5          000000 000000 000000 050000 A-OP. IS 5
38 A,6          000000 000000 000000 060000 A-OP. IS 6
39 A,7          000000 000000 000000 070000 A-OP. IS 7
40 A,10         000000 000000 000000 100000 A-OP. IS 10
41 A,11         000000 000000 000000 110000 A-OP. IS 11
42 A,12         000000 000000 000000 120000 A-OP. IS 12
43 A,13         000000 000000 000000 130000 A-OP. IS 13
44 A,14         000000 000000 000000 140000 A-OP. IS 14
45 A,15         000000 000000 000000 150000 A-OP. IS 15
46 A,16         000000 000000 000000 160000 A-OP. IS 16
47 A,17         000000 000000 000000 170000 A-OP. IS 17
48 B,Z          000000 000000 000000 000000 B-OP. IS 0. USED TO ADDR. ZERO REG., STATUS OR SCRATCH
49 B,D          000000 000000 000001 000000 B-OP. IS 1. USED TO ADDRESS THE D-REGISTER
50 B,P          000000 000000 004002 000000 B-OP. IS 2. USED TO ADDRESS THE P-REGISTER
51 B,B          000000 000000 004003 000000 B-OP. IS 3. USED TO ADDRESS THE B-REGISTER
52 B,L          000000 000000 000004 000000 B-OP. IS 4. USED TO ADDRESS THE L-REGISTER
53 B,A          000000 000000 000005 000000 B-OP. IS 5. USED TO ADDRESS THE A-REGISTER
54 B,T          000000 000000 000006 000000 B-OP. IS 6. USED TO ADDRESS THE T-REGISTER
55 B,X          000000 000000 004007 000000 B-OP. IS 7. USED TO ADDRESS THE X-REGISTER
56 B,STS        000000 000000 000010 000000 B-OP. IS 10. USED TO ADDR. REG. 10, STATUS OR SCRATCH
57 B,R1         000000 000000 000011 000000 B-OP. IS 11. USED TO ADDR. REG. 11, ADDR. OR SCRATCH
58 B,R2         000000 000000 000012 000000 B-OP. IS 12. USED TO ADDRESS REGISTER 12, SCRATCH
59 B,R3         000000 000000 000013 000000 B-OP. IS 13. USED TO ADDRESS REGISTER 13, SCRATCH
60 B,R4         000000 000000 000014 000000 B-OP. IS 14. USED TO ADDRESS REGISTER 14, SCRATCH
61 B,R5         000000 000000 000015 000000 B-OP. IS 15. USED TO ADDRESS REGISTER 15, SCRATCH
62 B,R6         000000 000000 000016 000000 B-OP. IS 16. USED TO ADDRESS REGISTER 16, SCRATCH
63 B,R7         000000 000000 000017 000000 B-OP. IS 17. USED TO ADDRESS REGISTER 17, SCRATCH
64 B,DEST       000001 000000 000000 000000 B-OP. IS VALUE OF "INSTR. BITS 0-2" LOADED BY LAST "T,MAPJ"
```

```
65 B,SRCE    000002 000000 000000 000000 B-OP. IS VALUE OF "INSTR. BITS 3-5" LOADED BY LAST "T,MAPJ"
66 B,LC      000003 000000 000000 000000 B-OP. IS VALUE OF THE "LOOP COUNTER" TWO INSTRUCTIONS AGO
67 B,0       000000 000000 000000 000000 B-OP. IS 0
68 B,1       000000 000000 000001 000000 B-OP. IS 1
69 B,2       000000 000000 000002 000000 B-OP. IS 2
70 B,3       000000 000000 000003 000000 B-OP. IS 3
71 B,4       000000 000000 000004 000000 B-OP. IS 4
72 B,5       000000 000000 000005 000000 B-OP. IS 5
73 B,6       000000 000000 000006 000000 B-OP. IS 6
74 B,7       000000 000000 000007 000000 B-OP. IS 7
75 B,10      000000 000000 000010 000000 B-OP. IS 10
76 B,11      000000 000000 000011 000000 B-OP. IS 11
77 B,12      000000 000000 000012 000000 B-OP. IS 12
78 B,13      000000 000000 000013 000000 B-OP. IS 13
79 B,14      000000 000000 000014 000000 B-OP. IS 14
80 B,15      000000 000000 000015 000000 B-OP. IS 15
81 B,16      000000 000000 000016 000000 B-OP. IS 16
82 B,17      000000 000000 000017 000000 B-OP. IS 17
83 ALUF,ANDAQ 010000 000000 000000 000000 A /\ Q -> F
84 ALUF,ANDAB 010200 000000 000000 000000 A /\ B -> F
85 ALUF,ANDDA 011200 000000 000000 000000 D /\ A -> F
86 ALUF,ANDDQ 011400 000000 000000 000000 D /\ Q -> F
87 ALUF,ORAQ  006000 000000 000000 000000 A \/ Q -> F
88 ALUF,ORAB  006200 000000 000000 000000 A \/ B -> F
89 ALUF,ORDA  007200 000000 000000 000000 D \/ A -> F
90 ALUF,ORDQ  007400 000000 000000 000000 D \/ Q -> F
91 ALUF,XORAQ 014000 000000 000000 000000 A XOR Q -> F
92 ALUF,XORAB 014200 000000 000000 000000 A XOR B -> F
93 ALUF,XORDA 015200 000000 000000 000000 D XOR A -> F
94 ALUF,XORDQ 015400 000000 000000 000000 D XOR Q -> F
95 ALUF,XNORAQ 016000 000000 000000 000000 NOT(A XOR Q) -> F
96 ALUF,XNORAB 016200 000000 000000 000000 NOT(A XOR B) -> F
```

```
 97 ALUF,XNORDA      017200 000000 000000 000000 NOT(D XOR A) -> F
 98 ALUF,XNORDQ      017400 000000 000000 000000 NOT(D XOR Q) -> F
 99 ALUF,ZERO        010400 000000 000000 000000 0 -> F
100 ALUF,INVQ        016400 000000 000000 000000 NOT(Q) -> F
101 ALUF,INVB        016600 000000 000000 000000 NOT(B) -> F
102 ALUF,INVA        017000 000000 000000 000000 NOT(A) -> F
103 ALUF,INVD        017600 000000 000000 000000 NOT(D) -> F
104 ALUF,PASSQ       014400 000000 000000 000000 Q -> F
105 ALUF,PASSB       014600 000000 000000 000000 B -> F
106 ALUF,PASSA       015000 000000 000000 000000 A -> F
107 ALUF,PASSD       015600 000000 000000 000000 D -> F
108 ALUF,MASKAQ      012000 000000 000000 000000 NOT(A) /\ Q -> F
109 ALUF,MASKAB      012200 000000 000000 000000 NOT(A) /\ B -> F
110 ALUF,MASKDA      013200 000000 000000 000000 NOT(D) /\ A -> F
111 ALUF,MASKDQ      013400 000000 000000 000000 NOT(D) /\ Q -> F
112 ALUF,A+Q         000000 000000 000000 000000 A + Q -> F
113 ALUF,A+B         000200 000000 000000 000000 A + B -> F
114 ALUF,D+A         001200 000000 000000 000000 D + A -> F
115 ALUF,D+Q         001400 000000 000000 000000 D + Q -> F
116 ALUF,A+Q+1       000000 040000 000000 000000 A + Q + 1 -> F
117 ALUF,A+B+1       000200 040000 000000 000000 A + B + 1 -> F
118 ALUF,D+A+1       001200 040000 000000 000000 D + A + 1 -> F
119 ALUF,D+Q+1       001400 040000 000000 000000 D + Q + 1 -> F
120 ALUF,-Q-1        004400 000000 000000 000000 -Q - 1 -> F
121 ALUF,-B-1        004600 000000 000000 000000 -B - 1 -> F
122 ALUF,-A-1        005000 000000 000000 000000 -A - 1 -> F
123 ALUF,-D-1        003600 000000 000000 000000 -D - 1 -> F
124 ALUF,Q-A-1       002000 000000 000000 000000 Q - A - 1 -> F
125 ALUF,B-A-1       002200 000000 000000 000000 B - A - 1 -> F
126 ALUF,A-D-1       003200 000000 000000 000000 A - D - 1 -> F
127 ALUF,Q-D-1       003400 000000 000000 000000 Q - D - 1 -> F
128 ALUF,A-Q-1       004000 000000 000000 000000 A - Q - 1 -> F
```

```
129 ALUF,A-B-1       004200 000000 000000 000000 A - B - 1 -> F
130 ALUF,D-A-1       005200 000000 000000 000000 D - A - 1 -> F
131 ALUF,D-Q-1       005400 000000 000000 000000 D - Q - 1 -> F
132 ALUF,Q           000400 000000 000000 000000 Q -> F
133 ALUF,B           000600 000000 000000 000000 B -> F
134 ALUF,A           001000 000000 000000 000000 A -> F
135 ALUF,D           001600 000000 000000 000000 D -> F
136 ALUF,Q+1         000400 040000 000000 000000 Q + 1 -> F
137 ALUF,B+1         000600 040000 000000 000000 B + 1 -> F
138 ALUF,A+1         001000 040000 000000 000000 A + 1 -> F
139 ALUF,D+1         001600 040000 000000 000000 D + 1 -> F
140 ALUF,Q-1         002400 000000 000000 000000 Q - 1 -> F
141 ALUF,B-1         002600 000000 000000 000000 B - 1 -> F
142 ALUF,A-1         003000 000000 000000 000000 A - 1 -> F
143 ALUF,D-1         005600 000000 000000 000000 D - 1 -> F
144 ALUF,-Q          004400 040000 000000 000000 -Q -> F
145 ALUF,-B          004600 040000 000000 000000 -B -> F
146 ALUF,-A          005000 040000 000000 000000 -A -> F
147 ALUF,-D          003600 040000 000000 000000 -D -> F
148 ALUF,Q-A         002000 040000 000000 000000 Q - A -> F
149 ALUF,B-A         002200 040000 000000 000000 B - A -> F
150 ALUF,A-D         003200 040000 000000 000000 A - D -> F
151 ALUF,Q-D         003400 040000 000000 000000 Q - D -> F
152 ALUF,A-Q         004000 040000 000000 000000 A - Q -> F
153 ALUF,A-B         004200 040000 000000 000000 A - B -> F
154 ALUF,D-A         005200 040000 000000 000000 D - A -> F
155 ALUF,D-Q         005400 040000 000000 000000 D - Q -> F
156 ALUD,Q           000000 000000 000000 000000 F -> Q ; F -> Y
157 ALUD,NONE        020000 000000 000000 000000 F -> Y
158 ALUD,B,YA        040000 000000 000000 000000 F -> B ; A -> Y
159 ALUD,B           060000 000000 000000 000000 F -> B ; F -> Y
160 ALUD,SRD         100000 000000 000000 000000 F -> Y ; (F,Q)/2 -> (B,Q)
```

```
161 ALUD,SRB        120000 000000 000000 000000 F -> Y ; F/2 -> B
162 ALUD,SLD        140000 000000 000000 000000 F -> Y ; (F,Q)*2 -> (B,Q)
163 ALUD,SLB        160000 000000 000000 000000 F -> Y ; F*2 -> B
164 STS,LO          000140 000000 000000 000000 IDB BITS 0-7 -> STATUS BITS 0-7
165 STS,EA          000040 000000 000000 000000 CRY -> STS C ; OVF -> STS Q ; OVF \/ STS 0 -> STS 0
166 STS,ES          000100 000000 000000 000000 ALU SHIFT OUTPUT -> STATUS M
167 CRY,C           000000 100000 000000 000000 STATUS C -> CARRY IN
168 CRY,GPR         000000 140000 000000 000000 GPR BIT 0 -> CARRY IN
169 ALUM,FMU        000000 010000 000000 000000 MULTIPLY ALU MODE. GPR 0 -> ALU-INSTR 1 ; RIGHT GPR-SHIFT
170 ALUM,FDV        000000 020000 000000 000000 DIVISION ALU MODE. GPR 0 -> ALU-INSTR 3 ; LEFT GPR-SH
171 ALUM,IR         000000 030000 000000 000000 SHIFT INSTR. MODE. SH MODE FROM IR-BITS. M IS SET AUTOM.
172 ALUM,MIC        000000 000000 000000 000000 MICROPROGRAM CONTR. SHIFT. SHIFT MODE FROM "MIS-BITS"
173 MIS,ROT         000000 002000 000000 000000 SPECIFIES ROTATIONAL SHIFT IF "ALUM,MIC"
174 MIS,ZIN         000000 004000 000000 000000 SPECIFIES ZERO-END-INPUT SHIFT IF "ALUM,MIC"
175 MIS,LIN         000000 006000 000000 000000 SPECIFIES LINK-END-INPUT SHIFT IF "ALUM,MIC"
176 LCOUNT          000000 000000 000100 000000 COUNT LOOP-COUNTER ; IF FALSE "RETURN"&"HOLD"
177 CONDENABL       000000 000000 000200 000000 ENABLE COND. SEQ., USE "FALSE" SPECS IF CONDITION FALSE
178 IDBS,ALU        000000 000000 000000 000000 ARITHMETIC-LOGIC-UNIT -> IDB. CGA.
179 IDBS,BMG        000000 000040 000000 000000 BIT-MASK-GENERATOR -> IDB. CGA.
180 IDBS,GPR        000000 000100 000000 000000 GENERAL-PURPOSE-REGISTER -> IDB. CGA.
181 IDBS,DBR        000000 000140 000000 000000 DATA-BUS-REGISTER -> IDB. CGA.
182 IDBS,ARG        000000 000200 000000 000000 ARGUMENT (MICROINSTRUCTION BITS 0-15) -> IDB. CGA.
183 IDBS,REG        000000 000240 000040 000000 REGISTER-FILE -> IDB
184 IDBS,STS        000000 000300 000000 000000 STATUS -> IDB. CGA.
185 NOT,USED        000000 000340 000000 000000 NOT USED
186 IDBS,BARG       000000 000400 000000 000000 B-OPERAND-ARGUMENT (0-17) -> IDB. CGA.
187 IDBS,SWAP       000000 000440 000000 000000 BYTE-SWAP OF LAST IDB -> IDB. CGA.
188 IDBS,PEA        000000 000500 000040 000000 PEA-REGISTER -> IDB
189 IDBS,PES        000000 000540 000040 000000 PES REGISTER -> IDB
190 IDBS,AARG       000000 000600 000000 000000 A-OPERAND-ARGUMENT*10 (0-170) -> IDB. CGA.
191 IDBS,PIC        000000 000640 000000 000000 PRIORITY-INTERRUPT-CONTROL STATUS REG. -> IDB. CGA.
192 IDBS,IOR        000000 000700 000040 000000 UART-STATUS -> IDB
```

```
193 IDBS,DSABL        000000 000740 000000 000000 DISABLE IDBS (USED TO READ "PIC" INFO, EXCEPT PIC-STS)
194 IDBS,MIPANS       000000 001000 000040 000000 IR 0 USED BY MOPC
195 IDBS,MAPANS       000000 001040 000040 000000 IR 0 USED BY TRA-INSTRUCTION
196 IDBS,GPR,SEXT     000000 001100 000000 000000 GPR BITS 0-7 (WITH BITS 8-15 = BIT 7) -> IDB. CGA.
197 IDBS,PGS          000000 001140 000000 000000 PAGING STATUS ACCORDING TO LAST REQUEST. CGA.
198 IDBS,CSR          000000 001200 000040 000000 CACHE STATUS REGISTER
199 IDBS,PCR          000000 001240 000000 000000 READ PAGING CONTROL REG.(NUMBER SELECTED BY PIL) CGA.
200 IDBS,ALD          000000 001300 000040 000000 AUTOMATIC LOAD DESCRIPTOR & PRINT STATUS
201 IDBS,PANEL        000000 001340 000040 000000 PANEL VECTOR
202 IDBS,RCS          000000 001400 000000 000000 READ CONTROL STORE
203 IDBS,PICVC        000000 001440 000000 000000 INTERRUPT VECTOR. CGA.
204 NOT,USED          000000 001500 000000 000000 NOT USED
205 NOT,USED          000000 001540 000000 000000 NOT USED
206 IDBS,LA           000000 001600 000000 000000 LOGICAL ADDRESS USED BY "ALU" WHEN JUMP.
207 IDBS,INR          000000 001640 000040 000000 READ INSTALLATION NUMBER FROM BACKWIRING.
208 IDBS,PICM         000000 001700 000000 000000 INTERRRUPT MASK REGISTER. CGA.
209 IDBS,UART         000000 001740 000000 000000 READ FROM THE REG. ADDRESSED BY COMM,UART
210 COMM,LDPIL        000000 000001 000000 000000 IDB BITS 8-15 -> STATUS BITS 8-15
211 COMM,LDGPR        000000 000002 000000 000000 IDB -> GENERAL-PURPOSE-REGISTER
212 COMM,EWRF         000000 000003 000040 000000 IDB -> REGISTER FILE WORD ADDRESSED BY A-OP AND B-OP
213 COMM,CLIRQ        000000 000004 000000 000000 PREVENT JUMP TO INTR VECT, REMOVE "PANEL"-EFF ON IRQ-TEST
214 COMM,UART,DATA    000000 000005 000000 000000 CONSOLE DATA-REGISTERS
215 COMM,UART,STATUS  000000 002005 000000 000000 CONSOLE STATUS-REGISTER
216 COMM,UART,MODE    000000 004005 000000 000000 CONSOLE MODE-REGISTERS 1/2
217 COMM,UART,COM     000000 006005 000000 000000 CONSOLE COMMAND-REGISTER
218 NOT,USED          000000 000006 000000 000000 "TRANSMIT BUFFER STROBE" TO "UART". IDB 0-7 -> "UART"
219 NOT,USED          000000 002006 000000 000000 NOT USED
220 COMM,LDPANC       000000 004006 000000 000000 SEND ONE BYTE TO IR 0
221 COMM,LDPCR        000000 006006 000040 000000 LOAD PCR (NUMBER DETERMINED BY PIL)
222 COMM,SIOC         000000 000007 000040 000000 LOAD "I/O-CONTROL" REGISTER
223 COMM,SLOW         000000 000010 000000 000000 SLOW CYCLE (200 NS)
224 COMM,EPIC         000000 000011 000000 000000 A-OPERAND IS AN INSTRUCTION TO "PIC"
```

```
225 COMM,SMPID       000000 000012 000000 000000 SET MICRO-PID. PID-BITS WHERE IDB IS "1" ARE FORCED TO "1"
226 COMM,START       000000 000013 000000 000000 RESET THE "STOP" FLIP-FLOP
227 COMM,SSTOP       000000 000014 000000 000000 SET THE "STOP" FLIP-FLOP
228 COMM,CLRTC       000000 000015 000000 000000 CLEAR THE 20 MS CLOCK FLIP-FLOP
229 COMM,CLFF        000000 000016 000000 000000 CLEAR THE "OOD" AND THE "DZD" FLIP-FLOP
230 COMM,LDLC        000000 000017 000040 000000 LOAD THE "LOOP COUNTER" WITH THE 6 LOWER IDB-BITS
231 COMM,LDSEG       000000 000020 000040 000000 LOAD SEGMENT REGISTER (SELECTS PHYSICAL 64K AREA)
232 NOT,USED         000000 002020 004040 000000 LOAD SINTRAN-4 DOMAIN REGISTER
233 NOT,USED         000000 004020 004040 000000 LOAD SINTRAN-4 PS REGISTER
234 COMM,LDIRV       000000 006020 000000 000000 LOAD INSTRUCTION REGISTER FOR OR-LOGIC USE
235 COMM,WCIHM       000000 000021 000000 000000 WRITE CACHE INHIBIT MAP (BIT 15 =1 IS NORMAL, =0 IS INH)
236 COMM,SSEMA       000000 002021 000000 000000 THE NEXT REQUEST IS A SEMAPHOR REQUEST
237 COMM,CCLR        000000 004021 000000 000000 CACHE CLEAR
238 COMM,LDEXM       000000 006021 000040 000000 LOAD EXAMINE MODE
239 COMM,IREAD,PT    000000 000022 002040 000000 INDIRECT ADDRESS READ (PT-RELATIVE)
240 COMM,IREAD,APT   000000 002022 006040 000000 INDIRECT ADDRESS READ (APT-RELATIVE)
241 COMM,MAP         000000 004022 000040 000000 USE IDB AS INSTRUCTION (AS IN EXR)
242 COMM,CNEXT,NWP   000000 006022 002040 000000 EXECUTE NEXT INSTRUCTION IF P NOT CHANGED IN LAST CYCLE
243 COMM,CJMP,F15    000000 000023 002040 000000 JUMP IF F15 =1
244 COMM,CJMP,NF15   000000 002023 002040 000000 JUMP IF F15 =0
245 COMM,CJMP,F=0    000000 004023 002040 000000 JUMP IF F =0
246 COMM,CJMP,NF=0   000000 006023 002040 000000 JUMP IF F  <> 0
247 COMM,CNEXT,SGR   000000 000024 000040 000000 SKIP LST
248 COMM,CNEXT,NSGR  000000 002024 000040 000000 SKIP GRE
249 COMM,CNEXT,CRY   000000 004024 000040 000000 SKIP MLS
250 COMM,CNEXT,NCRY  000000 006024 000040 000000 SKIP MGR
251 COMM,CNEXT,F15   000000 000025 000040 000000 SKIP GEQ
252 COMM,CNEXT,NF15  000000 002025 000040 000000 SKIP LSS
253 COMM,CNEXT,F=0   000000 004025 000040 000000 SKIP UEQ
254 COMM,CNEXT,NF=0  000000 006025 000040 000000 SKIP EQL
255 COMM,JMP,'       000000 000026 002040 000000 JMP P-RELATIVE
256 COMM,JMP,B       000000 002026 006040 000000 JMP ,B
```

```
257 COMM,JMP,I          000000 004026 000040 000000 JMP I & I,B
258 COMM,JMP,X          000000 006026 006040 000000 JMP ,X
259 COMM,JMP,XB         000000 000027 006040 000000 JMP ,X,B
260 COMM,JMP,XI         000000 002027 006040 000000 JMP I,X & I,B,X
261 NOT,USED            000000 004027 000040 000000 USED IN PREX-INSTRUCTION ONLY
262 COMM,CONTINUE       000000 006027 002040 000000 FETCH NEW INSTRUCTION RELATIVE TO P
263 COMM,AREAD,*        000000 000030 002040 000000 READ P-RELATIVE
264 COMM,AREAD,B        000000 002030 006040 000000 READ ,B
265 COMM,AREAD,I        000000 004030 006040 000000 READ I & I,B
266 COMM,AREAD,X        000000 006030 006040 000000 READ ,X
267 COMM,AREAD,XB       000000 000031 006040 000000 READ ,B,X
268 COMM,AREAD,XI       000000 002031 006040 000000 READ I,X & I,B,X
269 NOT,USED            000000 004031 000040 000000 READ FIRST PART OF 32-BIT INDIRECT ADDRESS
270 COMM,AREAD,NEXT     000000 006031 000040 000000 READ NEXT ADDRESS
271 COMM,AWRITE,*       000000 000032 002000 000000 WRITE P-RELATIVE
272 COMM,AWRITE,B       000000 002032 006000 000000 WRITE ,B
273 COMM,AWRITE,I       000000 004032 000000 000000 WRITE I & I,B
274 COMM,AWRITE,X       000000 006032 006000 000000 WRITE ,X
275 COMM,AWRITE,XB      000000 000033 006000 000000 WRITE ,X,B
276 COMM,AWRITE,XI      000000 002033 006000 000000 WRITE I,X & I,X,B
277 COMM,AWRITE,HOLD    000000 004033 000000 000000 WRITE IN LAST ADDRESS
278 COMM,AWRITE,NEXT    000000 006033 000000 000000 WRITE IN NEXT ADDRESS
279 COMM,RDRQ,PT        000000 000034 000040 000000 READ PAGE-TABLE-RELATIVE, ADDRESS FROM IDB
280 COMM,RDRQ,APT       000000 002034 000040 000000 READ ALTERNATIVE-PT-RELATIVE, ADDRESS FROM IDB
281 COMM,RDRQ,HOLD      000000 004034 000040 000000 READ FROM LAST USED PT, ADDRESS FROM IDB
282 COMM,EXRQ           000000 006034 000040 000000 EXAMINE
283 COMM,WRRQ,PT        000000 000035 000000 000000 WRITE PT-RELATIVE, ADDRESS FROM IDB
284 COMM,WRRQ,APT       000000 002035 000000 000000 WRITE APT-RELATIVE, ADDRESS FROM IDB
285 COMM,WRRQ,HOLD      000000 004035 000000 000000 WRITE IN LAST USED PT, ADDRESS FROM IDB
286 COMM,DERQ           000000 006035 000000 000000 DEPOSIT
287 COMM,ADCS           000000 000036 000000 000000 SET CONTROL STORE ADDRESS
288 COMM,RWCS           000000 002036 000000 000000 READ OR WRITE IN CONTROL STORE
```

```
289 COMM,MACL       000000 004036 000000 000000 HARD RESET. MASTER CLEAR WITH LOAD CONTROL STORE
290 COMM,XSLOW      000000 006036 000000 000000 MAKE EXTRA SLOW CYCLE (425 NS)
291 COMM,IDENT      000000 004037 000000 000000 IDENT BUS REQUEST
292 COMM,IOX        000000 006037 000000 000000 IOX BUS REQUEST
293 T,JMP           000000 000000 000000 000000 TRUE SEQUENCE IS JUMP
294 T,JMP0-3        000000 000000 001000 000000 TRUE SEQ IS JUMP. IR0-3 CONTROLS THE 4 LOWER JUMP ADDR BITS
295 T,JMPAOPR       000000 006000 001000 000000 TRUE SEQ IS JUMP. A-OP CONTROLS THE 4 LOWER JUMP ADDR BITS
296 T,RETURN        000000 000000 040000 000000 TRUE SEQUENCE IS RETURN
297 T,NEXT          000000 000000 100000 000000 TRUE SEQUENCE IS NEXT
298 T,HOLD          000000 000000 000000 000000 TRUE STACK OPERATION IS HOLD
299 T,POP           000000 000000 010000 000000 TRUE STACK OPERATION IS POP
300 T,LOAD          000000 000000 020000 000000 TRUE STACK OPERATION IS LOAD
301 T,PUSH          000000 000000 030000 000000 TRUE STACK OPERATION IS PUSH
302 F,JMP           000000 000000 000000 000000 FALSE SEQUENCE IS JUMP. USED WITH A CONDITION SETTING
303 F,RETURN        000000 000000 000000 000004 FALSE SEQUENCE IS RETURN. USED WITH A CONDITION SETTING
304 F,NEXT          000000 000000 000000 000010 FALSE SEQUENCE IS NEXT. USED WITH A CONDITION SETTING
305 F,HOLD          000000 000000 000000 000000 FALSE STACK OPERATION IS HOLD. USED WITH A COND SETTING
306 F,POP           000000 000000 000000 000001 FALSE STACK OPERATION IS POP. USED WITH A COND SETTING
307 F,LOAD          000000 000000 000000 000002 FALSE STACK OPERATION IS LOAD. USED WITH A COND SETTING
308 F,PUSH          000000 000000 000000 000003 FALSE STACK OPERATION IS PUSH. USED WITH A COND SETTING
309 COND,STP        000000 000000 000400 000360 CONDITION FOR TESTING IS  "STOP"
310 COND,PXM        000000 000000 000400 000220 CONDITION FOR TESTING IS PREFIX-MODE
311 COND,OVF        000000 000000 000400 000240 CONDITION FOR TESTING IS "OVERFLOW" FROM ALU
312 COND,CRY        000000 000000 000400 000260 CONDITION FOR TESTING IS "CARRY" FROM ALU
313 COND,F11        000000 000000 000400 000300 CONDITION FOR TESTING IS BIT 11 FROM ALU
314 COND,F15        000000 000000 000400 000320 CONDITION FOR TESTING IS BIT 15 FROM ALU
315 COND,F=0        000000 000000 000400 000340 CONDITION FOR TESTING IS "ALL ZEROS" FROM ALU
316 COND,LC=0       000000 000000 000400 000020 CONDITION FOR TESTING IS "LOOP-COUNTER" CONTENT = 0
317 COND,FETCH      000000 000000 000400 000100 CONDITION FOR TESTING IS LAST MEMEMORY REQUEST WAS FETCH
318 COND,IRQ        000000 000000 000400 000040 COND FOR TEST IS "IRQ". CHECKS LEV 10-15 IF "COMM,CLIRQ"
319 COND,RESTR      000000 000000 000400 000060 COND FOR TESTING IS "RESTRICTED-MODE". TRUE IF RING 0-1
320 COND,OOD        000000 000000 000400 000120 CONDITION FOR TESTING IS THE "ONE-OUT-DETECT" FLIP-FLOP
```

```
321 COND,DZD        000000 000000 000400 000000 COND FOR TESTING IS THE "DOUBLE-ZERO-DETECT" FLIP-FLOP
322 COND,COND       000000 000000 000400 000160 COND FOR TEST IS THE OUTCOME OF THE LATEST TEST
323 AB,CDIGI        000000 000000 000011 170000 COUNTER CONTROLLING NUMBER OF DIGITS IN AN OCTAL NUMBER
324 AB,UPPNR        000000 000000 000012 130000 THE UPPER ADDRESS LIMIT IN A MOPC DUMP CMD.
325 AB,CURNR        000000 000000 000013 130000 THE CURRENT ADDRESS IN A MOPC DUMP CMD.
326 AB,CNT10        000000 000000 000017 170000 COUNTER CONTROLLING THE NUMBER OF OCTAL DIGITS PER LINE
327 AB,PRCHR        000000 000000 000016 160000 THE NEXT CHARACTER TO BE WRITTEN BY MOPC
328 AB,TXT1         000000 000000 000014 160000 SCRATCH WORD CONTAINING DISPLAY TEXT
329 AB,TXT2         000000 000000 000013 160000 SCRATCH WORD CONTAINING DISPLAY TEXT
330 AB,SCRAM        000000 000000 000015 140000 SCRAMBLED REPRESENTATION OF LETTERS IN MOPC-INPUT
331 AB,OCTNR        000000 000000 000015 150000 OCTAL NUMBER ASSEMBLED FROM MOPC-INPUT
332 AB,DISPL        000000 000000 000015 160000 TYPE OF RUNNING DISPLAY
333 AB,OCTAD        000000 000000 000013 150000 ADDRESS OF RUNNING DISPLAY
334 AB,OCTA2        000000 000000 000012 160000 WORD TO EXTEND ADDRESS IN OCTAD TO 24 BITS
335 AB,DEPOS        000000 000000 000017 150000 SOME OCTAL DIGIT WAS WRITTEN SINCE LAST COMMAND TERMINATED
336 AB,DUMPF        000000 000000 000016 150000 SCRATCH WORD INDICATING THAT A DUMP IS IN PROGRESS
337 AB,RONLY        000000 000000 000014 150000 THE EXAMINED REGISTER IS READ-ONLY
338 AB,WRTYP        000000 000000 000012 150000 TYPE OF VARIABLE IN CASE OF DEPOSIT
339 AB,WRADR        000000 000000 000011 150000 ADDRESS OF VARIABLE IN CASE OF DEPOSIT
340 AB,ACTLV        000000 000000 000017 140000 SCRATCH WORD HOLDING "ACTIVE LEVELS"
341 AB,PVL          000000 000000 000015 170000 SCRATCH WORD HOLDING "PREVIOUS LEVEL"
342 AB,IIE          000000 000000 000012 170000 A "PIC" REPRESENTATION OF THE LAST "IIE" SETTING
343 AB,PID          000000 000000 000014 170000 THE MICROPROGRAM-KNOWN BITS OF THE "PID" REGISTER
344 AB,PIE          000000 000000 000013 170000 SCRATCH WORD HOLDING THE "PIE" REGISTER
345 AB,STATUS       000000 000000 000017 160000 SCRATCH WORD HOLDING THE LATEST "COMM,SIOC" INFORMATION
346 AB,OCTN2        000000 000000 000015 130000 SCRATCH WORD EXPANDING THE "AB,OCTNR" TO 24 BITS
347 AB,NUMBR        000000 000000 000012 120000 AN OCTAL NUMBER BEING PRINTED BY MOPC
348 AB,OPR          000000 000000 000011 160000 SCRATCH WORD HOLDING THE "OPR" REGISTER VALUE
349 AB,BRKPT        000000 000000 000016 140000 SCRATCH WORD HOLDING BREAKPOINT ADDRESS
350 AB,SINGL        000000 000000 000014 140000 SCRATCH WORD COUNTING SINGLE-INSTRUCTION
351 AB,BPFLG        000000 000000 000013 140000 SCRATCH WORD INDICATING THAT BREAKPOINT IS ON
352 AB,MACL         000000 000000 000011 140000 SCRATCH WORD HOLDING RETRY COUNTER FOR LOAD AFTER MACL
```

```
353 AB,LMP          000000 000000 000012 140000 SCRATCH WORD HOLDING THE "LMP" REGISTER VALUE
354 AB,EXMOD        000000 000000 000011 130000 SCRATCH WORD HOLDING THE "EXM" REGISTER
355 AB,MANIR        000000 000000 000014 130000 SCRATCH WORD HOLDING FLAG FOR MANUAL IR
356 AB,SSAVE        000000 000000 000016 130000 SCRATCH WORD HOLDING STS DURING DECIMAL INTRUCTIONS
357 XRF             000004 000000 000000 000000 SELECT EXTENDED REGISTER FILE
358 AB,UCIL         000000 000000 000014 120000 SCRATCH WORD HOLDING CACHE INHIBIT UPPER LIMIT
359 AB,LCIL         000000 000000 000013 120000 SCRATCH REGISTER HOLDING CACHE INHIBIT LOWER LIMIT
360 AB,PGS          000000 000000 000015 120000 COPY OF PGS FROM LAST PF/PV
361 AB,STBNK        000000 000000 000011 120000 GLOBAL BANK POINTER TO THE SEGMENT TABLE BANK
362 AB,STSRT        000000 000000 000016 120000 GLOBAL POINTER TO THE SEGMENT TABLE WITHIN THE BANK
363 AB,CMBNK        000000 000000 000017 120000 GLOBAL POINTER TO THE CORE MAP TABLE BANK
364 AB,BAUD         000000 000000 000010 010000 BAUD RATE THUMBWHEEL VALUE
365 AB,OLD303       000000 000000 000015 050000 SAVED IOX 303 A-REG TO SPOT UART DIFFS
```

APPENDIX C GLOSSARY

Bit The smallest unit of data in a digital computer. A bit may have the value 0 (zero) or 1 (one).

Cache An area of fast-access memory, closer to the CPU than main memory. Cache is used to store the program instructions and data that are most recently used by the computer, and which by implication are most likely to be used again. This fast availabliity significantly reduces the time that the CPU would otherwise need to access the same program/data.

Console The terminal used by the operator to communicate directly with the CPU, via OPCOM (q.v.). The console is the only terminal that can communicate directly with the microprogram. (See also Terminal no. 1, and OPCOM).

Data Numbers, letters, symbols and the codes that are used to represent them, regarded as objects to be stored or processed by a computer.

Field A portion of an expression (e.g. a computer word) that is designated as having a specific significance or function within that word. The 64-bit microinstruction word used in the ND-100 series of computers is organised into fields: see the Microinstruction word format & function charts on pages 16, 17 and 18.

IDBS Internal data bus source. Within the CPU, data is moved around over several high-speed busses, which are collectively referred to in functional terms as the internal data bus system. The IDBS field of a microinstruction defines which of several possible sources will drive the internal data bus.

Information The interpretation given to data when understood in a specific context. Thus for example, depending on the context, the 16-bit value 143304_8 may be understood as:

- The unsigned integer 50884_{10}
- The ASCII string "FD"
- The ND-110/ND-120 instruction STATX

Instruction A code which can be interpreted by the CPU (or ALU) as a command to perform one or more logical or arithmetical operations. The ND-110 and ND-120 have two levels of instructions: macroinstructions and microinstructions. (See also macroinstruction and microinstruction.)

Macroinstruction A single software instruction written as part of a source program language. Sometimes referred to as machine instruction or MAC instruction. In the ND-110 and ND-120, each macroinstruction is executed by a sequence of microinstructions. (See also microinstruction.)

Microcode The name given to a set of microinstructions.

Microinstruction In the ND-100 family of computers, this is a 64-bit instruction word which acts directly within the central processing unit to control processor activity. A sequence of one or more microinstructions is required to execute a machine (macro-) instruction.

Microprogram The set of very low level instructions that together define the functions of the CPU. (See also microinstruction)

MMS Memory Management System: that part of the ND computer which handles memory page addressing and memory (paging and ring) protection.

MOPC Acronym from Microprogram OPerator Communication. This is the part of the microprogram that communicates with the console (terminal no. 1) when the CPU is in the OPCOM mode.

ND-100 family The family of 16-bit general-purpose computers from Norsk Data, currently consisting of the following machines:

- ND-100
- ND-100/CE
- ND-100/CX
- ND-110 Standard
- ND-110/CX
- ND-120/CX.

On-board memory Memory that is available for access by the CPU in a ND-100 bus computer system, and which is "on-board" the CPU card itself, as is the case in the ND-120 CPU. See also "Primary memory". Note that memory on-board the CPU card is functionally a part of the primary addressable memory in the ND-100 bus system. As such, it is also accessible to DMA Controllers on the same ND-100 bus.

OPCOM Acronym from OPerator COMmunication mode. In this mode the CPU console (terminal no. 1) communicates directly with the microprogram. This mode is used for service and maintenence only. It involves running the "microprogrammed operator's communication" (MOPC) program under SINTRAN. Terminal no. 1 (also called "console") is the user's interface to OPCOM.

Page A contiguous area of memory consisting of 1024 words (2K bytes). By extension, a page may also refer to 1024 words on a disk, tape or other storage medium.

Page table An area of high speed memory, within the memory management part of the CPU, which contains the information needed for the memory management system to convert virtual (16-bit) addresses to physical (24-bit) addresses.

Primary memory This is random access memory (RAM) in the ND-100 system that is accessible to both the CPU and to DMA controllers on that bus. It may be contributed from a variety of cards in the system (e.g. ND-120 on-board memory, memory cards, PIOC cards, Ethernet cards, Multiport memory). In functional terms, it must be strung together in consecutive address ranges (e.g. blocks of 64K), to provide a continuous addressable memory space in the computer.

Panel The control panel of an ND-100 series computer.

PIC Priority Interrupt Controller.

Terminal no. 1 See OPCOM.

Trap A Trap is an unprogrammed conditional jump that is automatically activated in the CPU when it detects certain conditions that require immediate attention. Traps can be generated by hardware or by the microprogram.

Word A computer word is a basic unit of information that a particular computer handles.

The ND-110 and ND-120 belong to the ND-100 family of computers, which at the MACROinstruction level use 16-bit words, and at the MICROinstruction level use 64-bit words.

Index

Order Information

Additional copies of this manual may be ordered.

Please send your order to your local ND office,

or in Norway to:

Documentation Department
Norsk Data A.S
Olaf Helsets Vei 5
Post Box 25, Bogerud
0621 Oslo 6
Norway

or in UK to:

Documentation Department
Norsk Data Ltd
Benham Valence
Newbury
Berkshire RG16 8LU
England

quoting your name, company and address, and specifying the ND part number for this manual.

Updates

This manual may be updated in two ways:

- complete new issues
- revised pages.

New issues consist of a complete new manual which replaces the earlier issue of the manual. A new issue incorporates all the revisions since the previous issue.

Revisions consist of one or more pages which replace pages in the existing issue of the manual. Each revised page is listed on tne new Printing Record which accompanies the revision pages. The previous Printing Record should be replaced by the new one.

New issues and revisions are announced in the ND Bulletin, and can be ordered through the normal ND channels.

Reader's comments

The Reader's Comments form at the end of this manual can be used both to report errors in the manual and to give an evaluation of it. All comments are welcome, both detailed and general.