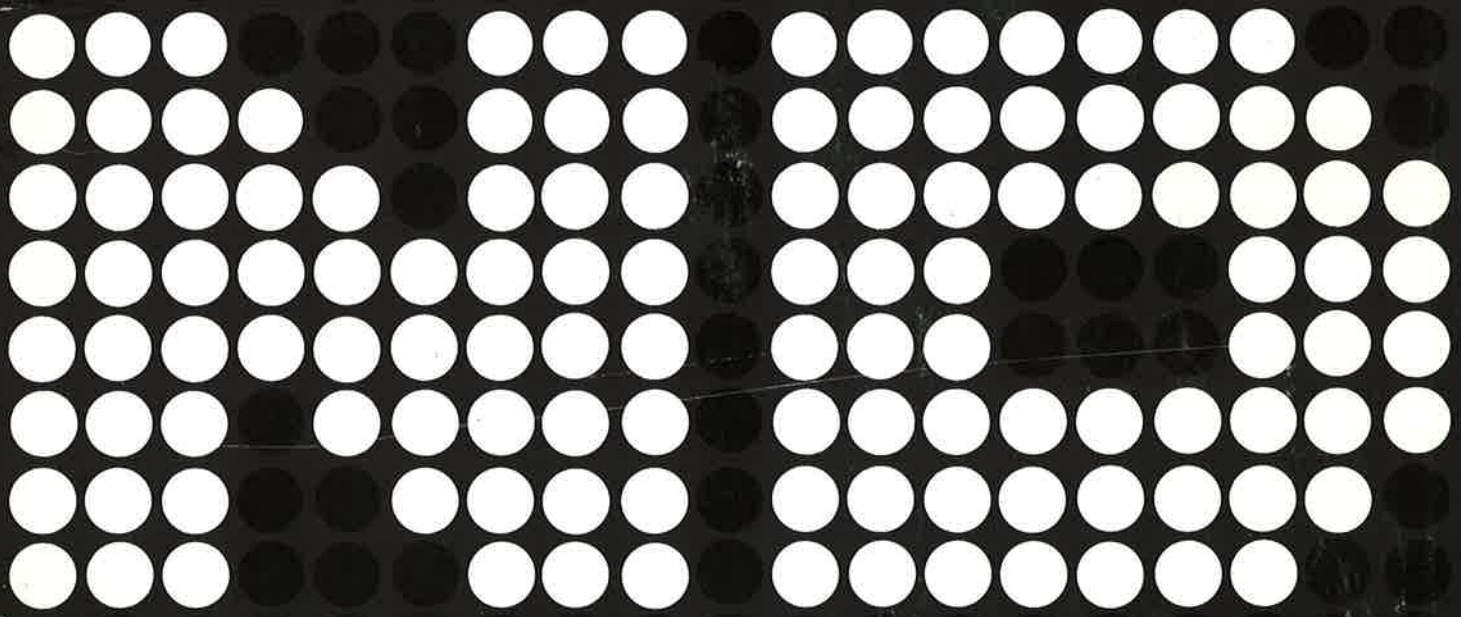
NORD-12
REFERENCE MANUAL

A/S NORSK DATA-ELEKTRONIKK



NORD-12
REFERENCE MANUAL

TABLE OF CONTENTS

+‡+

Chapters :		Page:
1	INTRODUCTION	1-1
1.1	General Characteristics	1-1
1.2	Peripheral Equipment	1-2
1.3	Software	1-2
2	SYSTEM ARCHITECTURE	2-1
2.1	Introduction	2-1
2.2	Central Processor	2-3
2.2.1	Indicators	2-5
2.3	Instruction and Data Formats	2-5
2.3.1	Instruction Formats	2-6
2.3.2	Data Formats	2-6
2.3.2.1	Single Bit	2-6
2.3.2.2	8-bit Byte	2-6
2.3.2.3	16-bit Word	2-7
2.3.2.4	32-bit Double Word	2-7
2.3.2.5	48-bit Floating Point Word	2-8
2.4	Interrupt System	2-9
3	INSTRUCTION REPERTOIRE	3-1
3.1	Memory Reference Instructions	3-3
3.1.1	Addressing Structure	3-3
3.1.2	Store Instructions	3-14
3.1.3	Load Instructions	3-15
3.1.4	Arithmetical and Logical Instructions	3-16
3.1.5	Sequencing Instructions	3-19
3.1.6	Byte Instructions	3-21
3.1.7	Register Block Instructions	3-22
3.2	Operate Instructions	3-24
3.2.1	Floating Point Conversion Instructions	3-24
3.2.2	Shift Instructions	3-25
3.2.3	Register Operations	3-29
3.2.3.1	ROP Register Operation Instructions	3-30
3.2.3.2	EXTended Register Operation Instructions	3-37
3.2.3.3	Inter Level Register Instructions	3-39
3.2.4	Skip Instructions	3-40
3.2.5	Argument Instructions	3-43
3.2.6	Bit Operation Instructions	3-45
3.2.6.1	Bit Skip Instructions	3-46
3.2.6.2	Bit Setting Instructions	3-46
3.2.6.3	One Bit Accumulator Instructions	3-47
3.2.7	Accumulator Transfer Instruction	3-48
3.2.7.1	Transfer to A register	3-49
3.2.7.2	Transfer from A register	3-50

Chapters:		Page
3.3	Input/Output Control Instructions	3-52
3.3.1	Recommended Device Addresses	3-52
3.3.2	Format of Status and Control Word	3-56
3.4	System Control Instructions	3-57
3.4.1	Interrupt Control Instructions	3-57
3.4.2	Monitor Call Instruction	3-60
3.4.3	Wait or give up Priority	3-61
3.5	Customer Specified Instructions	3-62
4	THE INPUT/OUTPUT SYSTEM	4-1
4.1	Input/Output Hardware	4-1
4.1.1	General Description	4-1
4.1.2	Vectored Interrupt Identification	4-2
4.2	Input/Output Programming	4-4
4.2.1	Programming Examples	4-4
4.2.2	Input/Output Interrupt Programming	4-5
4.2.3	Design of an Input/Output Handler Routine	4-5
5	THE INTERRUPT SYSTEM	5-1
5.1	Control of Program Levels	5-2
5.1.1	Program Level Activation	5-4
5.2	Initialization of Interrupt System	5-4
5.3	Interrupt Program Organization	5-5
5.4	Internal Interrupts	5-6
5.4.1	Monitor Call Interrupt	5-6
5.5	Vectored Interrupts	5-7
6	CONTROL PANEL	6-1
6.1	NORD-12 Control Panel	6-1
6.1.1	Power On/Off	6-1
6.1.2	Master Clear	6-1
6.1.3	Restart	6-2
6.1.4	Load	6-2
6.1.5	Continue	6-2
6.1.6	Stop	6-2
7	NORD-10 OPERATOR'S PANEL	7-1
7.1	Panel Elements	7-1
7.2	18-bit Switch Register	7-1
7.3	18-bit Light emitting Diode Register	7-1
7.4	16 Selector Pushbuttons and 16 associated Light emitting Diodes	7-1
7.5	Display Level Select	7-3

Chap ters: Page

7.6	Control Buttons	7-3
7.6.1	Master Clear	7-3
7.6.2	Restart	7-3
7.6.3	Load	7-3
7.6.4	Decode Address	7-4
7.6.5	Set Address	7-4
7.6.6	Deposit	7-4
7.6.7	Enter Register	7-4
7.6.8	Single Instruction	7-4
7.6.9	Continue	7-5
7.6.10	Stop	7-5
7.7	Mode Indicators	7-5
8	OPERATOR'S COMMUNICATION	8-1
8.1	Functions	8-2
8.1.1	Start a Program	8-2
8.1.2	Memory Examine	8-2
8.1.3	Memory Deposit	8-3
8.1.4	Register Examine	8-3
8.1.5	Internal Register Examine	8-4
8.1.6	Current Location Counter	8.4
8.1.7	Break Function	8-4
8.1.8	Bank Number	8-5
8.2	Bootstrap Loaders	8-5
8.2.1	Octal Format Load	8-5
8.2.2	Binary Format Load	8-6
8.2.3	Mass Storage Load	8-7
8.2.4	Automatic Load Descriptor	8-7
8.2.5	Examples	8-9
APPENDIX	A	A-1
APPENDIX	B	B-1
APPENDIX	C	C-1

1 INTRODUCTION

1.1 General Characteristics

The NORD-12 computer system is a compact mini-computer system with an unusually large instruction set.

The NORD-12 belongs to the NORD-10 family of 16-bit computers where program compatibi lity with the NORD-10 is assured by the fact that both the instruction set and the Read Only Memory which controls the instruction execution is common to both the NORD-10 and the NORD-12. For readers familiar with the NORD-10, we recommend you read Appendix C which lists all differences between NORD-12 and NORD-10.

A basic instruction set is common to all NORD-12 machines, and this set is highly optimized to produce effective code; hardware floating point arithmetic is standard as are the instructions to manipulate individual bits at high speed.

The register structure and addressing scheme facilitate the processing of structured data with high efficiency.

The NORD-12 is micro-programmed, and all NORD-12 instructions are executed by means of a micro-program located in a very fast (65 ns) read only memory. Micro-programming gives the NORD-12 computer flexibility and a very large growth potential. New instructions may be added to the NORD-12, and instructions for special applications may be optimized for a particular use.

NORD-12 provides up to 1024 customer-specified instructions. These instructions are micro-programmed in a programmable read-only memory, which is added onto the standard read-only memory.

Micro-programming in NORD-12 is also used to control the operator's panel and to perform operator communication between the operator and the console Teletype or display.

Bootstrap loaders both for character-oriented devices and mass storage devices are also controlled by a micro-program.

The NORD-12 uses MOS type memories with memory size from 4K to 64K words; memory increment size is 4K. Memory parity is an option in which case the word-length is 18 bits with one parity bit for each 8-bit byte.

Another option is a power fail/auto restart system which also provides 30 minutes of memory stand-by power.

The NORD-12 standard processor executes at a speed of 490 ns for each micro-instruction. This manual gives complete timing figures for all instructions.

The input/output and interrupt systems of NORD-12 are designed for ease of use and very high speed. NORD-12 has 16 program levels each with its own set of registers, making possible a complete context switching from one program level to another in only 2.0 μs. In addition 2048 vectored priority input/output interrupts are standard.

1.2 Peripheral Equipment

A complete range of peripheral equipment is available for the NORD-12. The I/O system is common for both the NORD-12 and the NORD-10, and all interfaces for the NORD-10 are immediately available also for the NORD-12. When upgrading from a NORD-12 to a NORD-10 all peripherals and interfaces may be moved from the NORD-12 to the NORD-10.

Most peripherals to the NORD-12 are offered with a range of different performances. The range of peripherals include several types of console typewriters, teletypes or display terminals, paper tape equipment, line printers, card equipment, high speed electrostatic printer/plotters, magnetic cassette tape, 9-track magnetic tape also including high performance 90 ips 1600 bpi tape, fixed head drums, moving head cartridge disc, A/D - D/A equipment, transmission line interfaces and a CAMAC interface.

1.3 Software

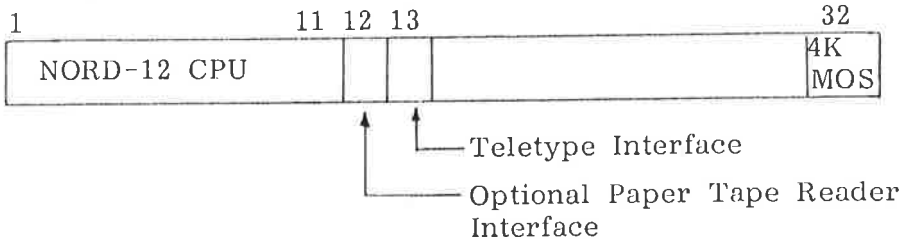
Based on 7 years of experience with NORD-1 and NORD-10 a wide range of system software is available including a SINTRAN III/12 operating system.

For further information, please contact A/S Norsk Data-Elektronikk's Sales Offices.

2 SYSTEM ARCHITECTURE

2.1 Introduction

The NORD-12 in its minimum size has 4K MOS memory, a Teletype interface and a small control panel. From this initial configuration it is possible to expand to a very large computer system.



The NORD-12 initial configuration provides 20 unused slots, a slot may either be used for a 4K memory module (max 64K) or for an I/O interface (max 12 interfaces in the CPU crate).

If more I/O slots are required an External Bus Driver which takes 3 slots in the CPU crate, provides a full NORD-10/NORD-12 bus on differential line driver/line receiver signal levels.

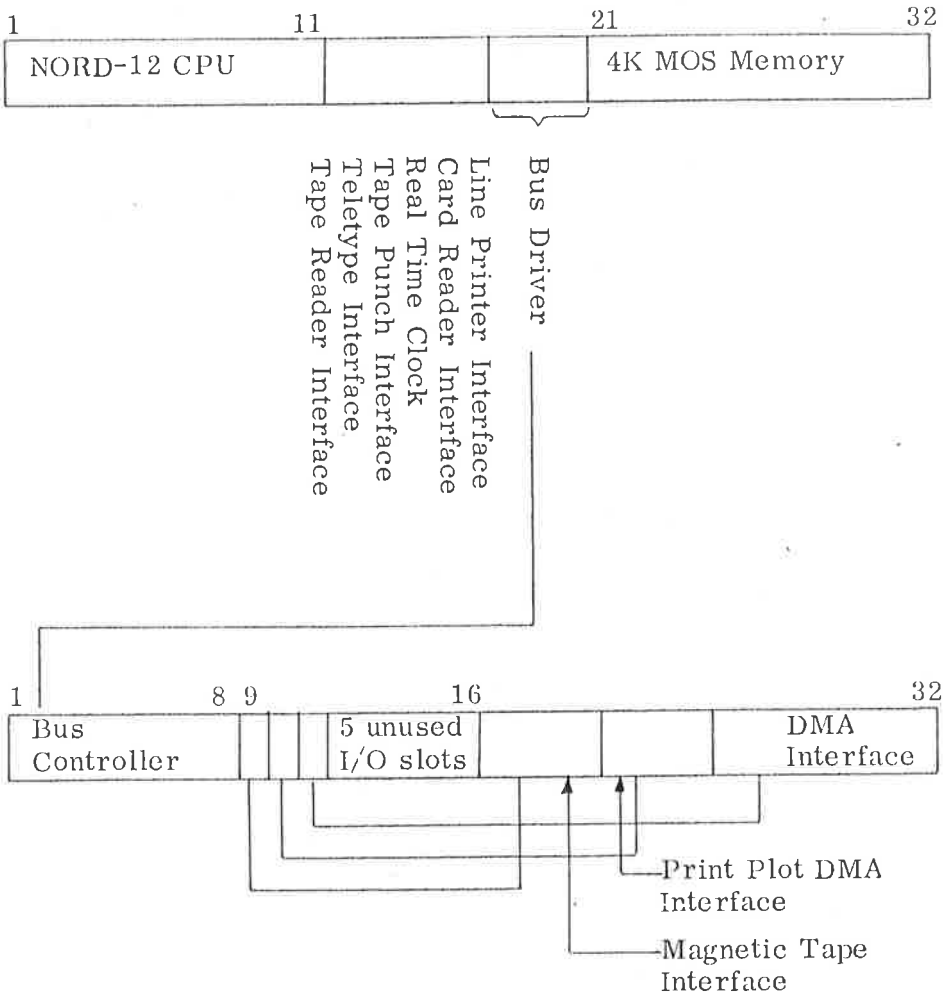
This I/O bus may then be connected to one or more of the following units:

1. Bus Controller, which provides core address registers for DMA type interfaces, and another 8 or 16 I/O slots.

2. Bus Switch, making it possible to switch peripherals between different NORD-10/NORD-12 computers.

3. CAMAC crate controller, providing access to the wide range of CAMAC equipment available.

Figure 2.1: Example of a larger system.



In this example a 48K with 6 interfaces for programmed I/O and 3 DMA interfaces for a 10 Mbyte Cartridge disc (expandable to 40 Mbytes), a 90 ips 1600 bpi Magnetic tape and a 1000 lines/min clextrostatic printer/plotter will fit into two 7" high standard 19" crates.

2.2 Central Processor

The connection of main modules in the CPU is through the common data bus. BD. and common address bus. BA. as shown in Figure 2.2. For simplicity control lines and inter-register buses are omitted in this figure.

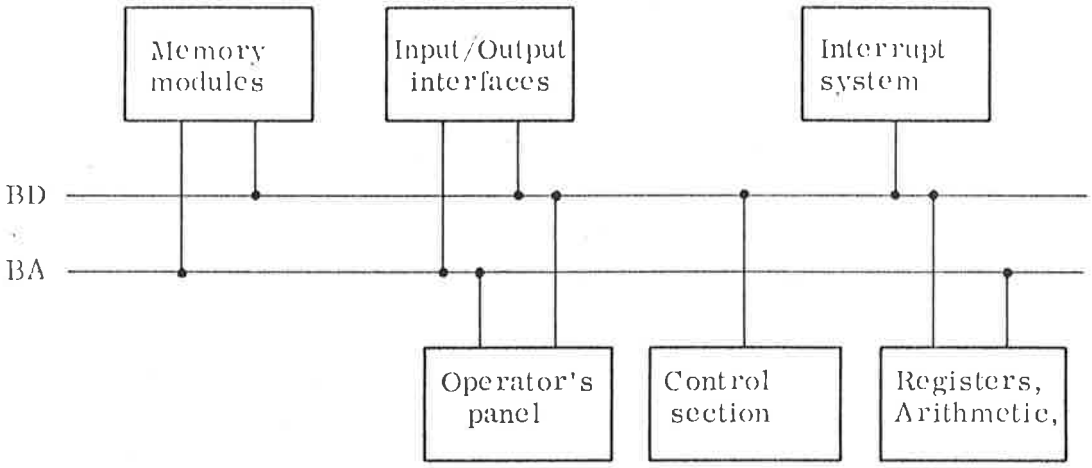


Figure 2.2: NORD-12 CPU Bus Structure

A more detailed diagram of the control section and register block is given in Figure 2.3.

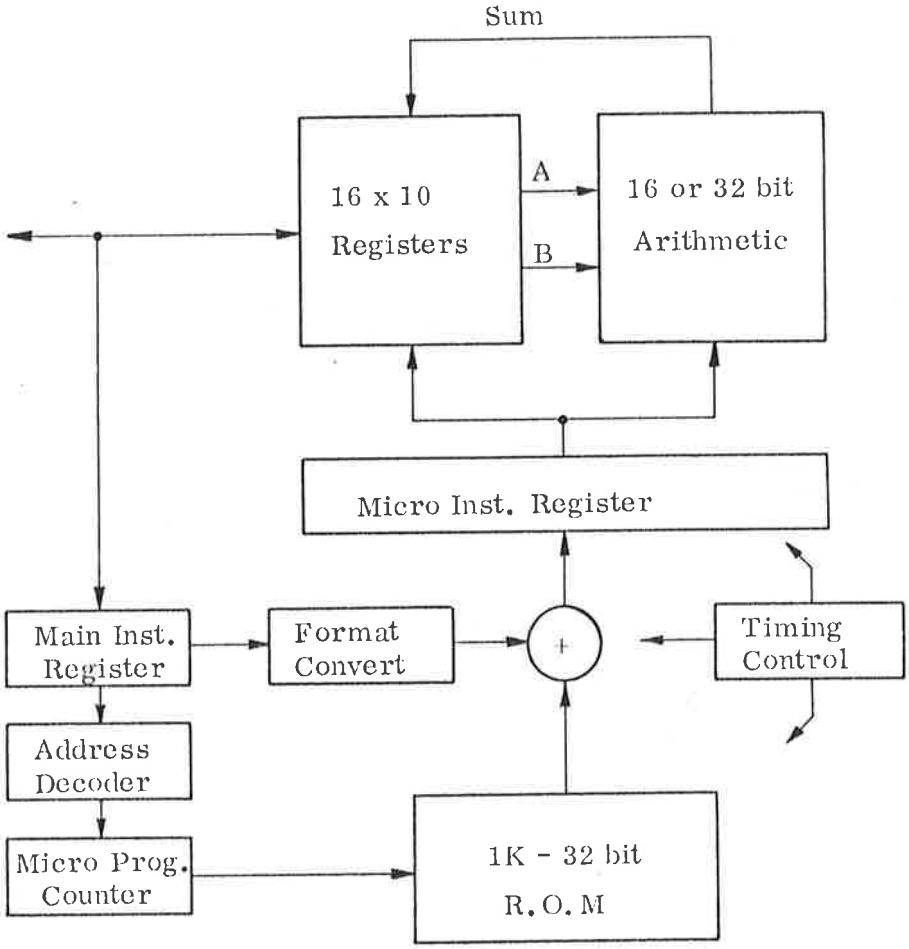


Figure 2.3 CPU Block Diagram

The register block contains 8 general registers for each program level and two scratch registers for each level to be used by the micro-processor.

The arithmetic unit is normally operated in a 16-bit format. The 32-bit format is used for floating point and double precision operations. The arithmetic unit contains the necessary buffer registers to do the complete inner loop in the floating point micro-programs using only 490 ns. 32-bit format is achieved by two 16 bit operations in sequence.

Some instructions in the NORD-12 instruction set are general two-address inter-register instructions. Due to the generality of these instructions, 2048 inter-register instructions (see Section 3.2.3) are converted directly to the three-address format of the micro-instruction and fed directly into the micro-instruction register. The remaining bits, i.e. cycle control etc. are read from the read-only memory.

2.2.1 Indicators

Six indicators are accessible by program. These six indicators are:

C — Carry indicator. The carry indicator is dynamic.

Q — Dynamic overflow indicator.

O — Static overflow indicator. This indicator remains set after an overflow condition until it is reset by program.

Z — Error indicator. This indicator is static and remains set until it is reset by program.

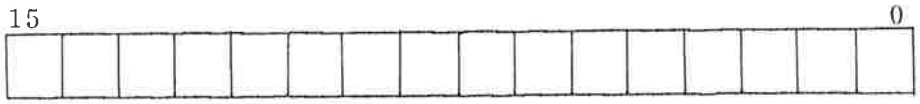
K — One bit accumulator. This indicator is used by the BOP bit operations, instructions operating on one-bit data.

M — Multi-shift link indicator. This indicator is used as temporary storage for discarded bits in shift instructions in order to ease the shifting of multiple precision words.

These six indicators are fully program controlled either by means of the BOP instructions or by the TRA or TRR instructions where all indicators may be transferred to and from the A register.

2.3 Instruction and Data Formats

The NORD-12 has a 16-bit word format. The bits are conventionally numbered 0 to 15 with the most significant bit numbered 15 and the least significant bit numbered 0.



16-bit NORD-12 word

Figure 2.4: NORD-12 Bit Numbering Convention

The content of a NORD-12 word is conventionally represented by a 6-digit octal number. Thus, the content of a word with all 16 bits set to zero is represented as 000000 while the content of a word with all bits set to one is represented as 177777.

2.3.1 Instruction Formats

All NORD-12 instructions are contained in one single 16 bit word.

The instruction set is divided into the following five subclasses:

Memory Reference Instructions

Operate Instructions

Input/Output Control Instructions

System Control Instructions

Customer Specified Instructions.

In Chapter 3 each instruction is given a short description. This includes a diagram showing the instruction format.

2.3.2 Data Formats

The standard NORD-12 instruction set provides instructions for the following five different data formats:

a) Single bit

b) 8-bit byte

c) 16-bit word

d) 32-bit double word

e) 48-bit floating point word

2.3.2.1 Single Bit

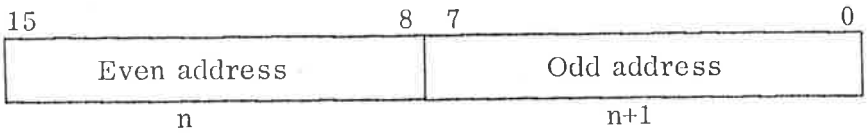
A single bit data word is typically used for a logical variable; the bit instructions (see Section 3.2.6) are for manipulation of single bit variables. The bit instructions specify operations on any bit in any of the general registers, as well as the accumulator indicator K.

2.3.2.2 8-bit Byte

Two instructions are available in the standard NORD-12 instruction set for byte manipulations, i.e. load byte and store byte, see Section 3.1.6.

A byte consists of 8 bits giving a range of $0 \leq X \leq 255$.

The byte addressing, see Section 3.1.6. is such that when two bytes are packed into a word the even byte address points to the left half of the word



Byte Format.

2.3.2.3 16-bit Word

The most common data word format is the 16-bit word contained in one memory location or one register.

Representation of negative numbers is in 2's complement. The skip instruction, see Section 3.2.4, also contains instructions to treat numbers as unsigned (magnitude) numbers.

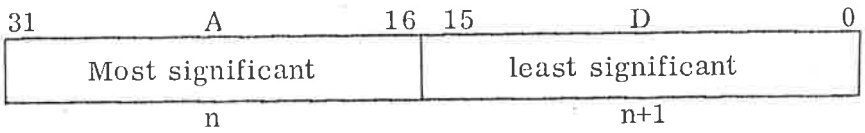
Range $-32768 \leq X \leq 32767$

or $0 \leq X \leq 65535$

2.3.2.4 32-bit Double Word

Two instructions are available to handle double word formats, load double and store double, see Sections 3.1.2 and 3.1.3.

A double word is a 32-bit number which occupies two consecutive locations (n, n+1) in memory, and where negative numbers are in 2's complement.



Double Word Format.

A double word is always referred to by the address of its most significant part. Normally a double word is transferred to the registers so that the most significant part is contained in the A register and the least significant in the D register. Range as integers:

$$-2\ 147\ 483\ 648 \leq X \leq 2\ 147\ 483\ 647$$

2.3.2.5 48-bit Floating Point Word

The standard NORD-12 instruction set provides full floating point hardware arithmetic instructions, load floating, store floating, add subtract, multiply and divide floating, convert floating to integer and convert integer to floating.

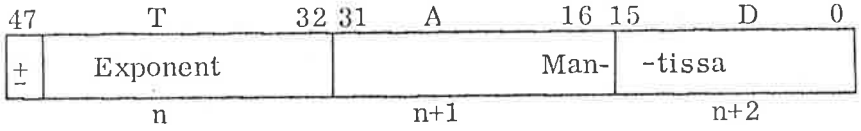
The data format of floating point words is 32 bits mantissa magnitude, one bit for sign and 15 bits for a biased exponent.

The mantissa is always normalized, $0.5 \leq$ mantissa < 1. The exponent base is 2. the exponent is biased with 2^{14}. A standardized floating zero contains zero in all 48 bits.

In main memory one floating point data word occupies three 16-bit core locations, which are addressed by the address of the exponent part.

n	exponent and sign
n+1	most significant part of mantissa
n+2	least significant part of mantissa

In CPU registers bits 0-15 of the mantissa are in the D register, bits 16-31 in the A register and bits 32-47, exponent and sign, in the T register. These three registers together are defined as the floating accumulator.



Floating Word Format

The accuracy is 32 bits or approximately 10 decimal digits; any integer up to 2^{32} has an exact floating point representation.

The range is

$$2^{-16384} \cdot 0.5 \leq X < 2^{16383} \cdot 1 \text{ or } X = 0$$

or $$10^{-4920} < x < 10^{4920}$$

Examples (octal format):

	T	A	D
0	0	0	0
+1:	040001	100000	0
-1:	140001	100000	0

2.4 Interrupt System

The NORD-12 Interrupt System allows priority interrupt handling at extremely high speed. The interrupt system consists of 16 program levels in hardware, each program level with its own complete set of general registers and status indicators. The program levels are numbered from 0-15 with increasing priority; program level 15 has the highest priority, program level 0 the lowest. The context switching from one program level to another is completely automatic and requires only 2.0 μs.

All program levels can be activated by program. In addition program levels 10-13 and 15 can be activated by external devices. Level 14 is used for monitor calls.

As many as 2048 vectored interrupts may be connected.

By using these program levels large programming systems may be greatly simplified. Independent tasks may be organized at different program levels with all priority decisions determined by hardware and with almost no overhead because of the rapid context switching.

The program level to run is controlled from the two 16-bit registers:

PIE - Priority Interrupt Enable

PID - Priority Interrupt Detect

Each program level is controlled by the corresponding bits in these registers. The PIE register is program controlled, and the PID register is controlled by both program and vectored interrupts.

At any time, the highest program level which has its corresponding bits set in both PIE and PID is running. This level is called PL.

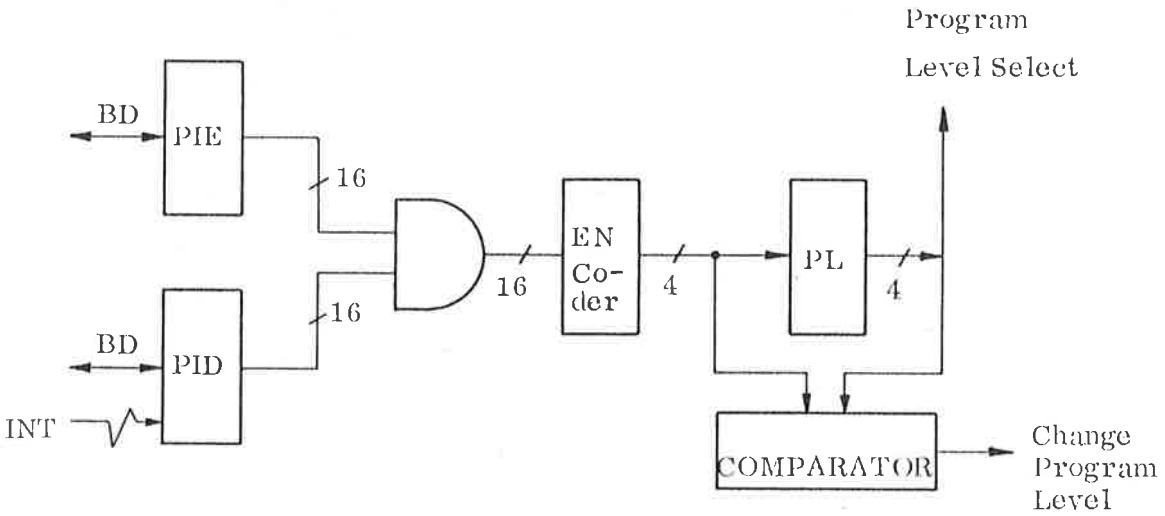


Figure 2.10: Program Level Control

A change from a lower to a higher program level is caused by an interrupt request. A change from a higher program level to a lower takes place when the program on the higher program level gives up its priority.

3 INSTRUCTION REPERTOIRE

In NORD-12 all instructions occupy a single word, 16-bits, yielding a very efficient use of memory, and also producing code with unusual efficiency with regard to speed. 48 bits floating point arithmetic operations and floating integer conversions are standard.

Note that in this chapter one is always referring to the register set on current program level, for example "the A register" means " the A register on current program level".

In this manual the instruction set of NORD-12 is divided into the following five subclasses:

3.1 Memory Reference Instructions

3.2 Operate Instructions

3.3 Input/Output Control Instructions

3.4 System Control Instructions

3.5 Customer Specified Instructions

Each instruction is given a short description. This includes its mnemonic as used in the assembly language, octal code, a diagram showing its format, timing information and special comments. For each instruction the systems and indicators that can be affected by the instruction are listed.

The definitions used in the descriptions are as follows:

General Registers

A	A register
D	D register
T	T register
L	L register
X	X register
B	B register
P	Program counter
STS	Status register containing K, Z, Q, O, C, M

Status Word

Bit		
2	K	One bit accumulator
3	Z	Error indicator
4	Q	Dynamic overflow indicator
5	O	Static overflow indicator
6	C	Carry indicator
7	M	Multishift link indicator
8-11	PL	Program level indicator
15	IONI	Interrupt System On indicator

Special Registers

OPR	Operator's panel switch register
LMP	Lamp register
PVL	Previous level register
PID	Priority interrupt detect
PIE	Priority interrupt enable
ALD	Automatic load descriptor
IR	Instruction register

Abbreviations

EL	Effective location
EW	Effective word
AD	Double accumulator
FA	Floating accumulator
DW	Double word
FW	Floating word
sr	Source register
dr	Destination register
$\wedge$	Logical AND
V	Logical inclusive OR
$\forall$	Logical exclusive OR
()	The contents of
μs	Microsecond
ns	Nanosecond

The NORD-12 is offered with dynamic MOS memories, these memory chips have the following specifications (as measured on the chip-level)
access-time 300 ns.
cycle-time 490 ns

The instruction times specified in this manual are as measured from a program running in a standard NORD-12, with standard MOS memory.

3.1 Memory Reference Instructions

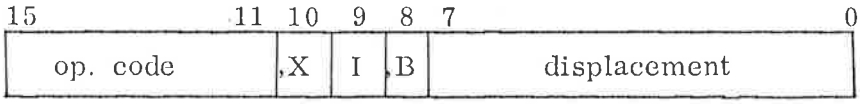
Memory reference instructions specify operations on words in memory. For all the memory reference instructions in NORD-12 the addressing mode is the same, with the exception of the conditional jump, the byte and the register block instructions. The addressing structure for these memory reference instructions is given under the specific instruction specification.

The NORD-12 has the following groups of memory reference instructions:

3.1.2 Store Instructions
3.1.3 Load Instructions
3.1.4 Arithmetic and logical Instructions
3.1.5 Sequencing Instructions
3.1.6 Byte Instructions
3.1.7 Register Block Instruction

3.1.1 Addressing Structure

In memory reference instruction words, 11 bits are used to specify the address of the desired word(s) in memory, 3 address mode bits and 8-bit signed displacement using 2's complement for negative numbers and sign extension.*



NORD-12 uses a relative addressing system, which means that the address is specified relative to the contents of the program counter, or relative to the contents of the B and/or X registers.

The three addressing mode bits called ",X" "I" ",B" provide eight different addressing modes.

The addressing mode bits have the following meaning:

- The I bit specifies indirect addressing
- The ,B bit specifies address relative to the contents of the B register, pre-indexing. The indexing by ,B takes place before a possible indirect addressing.
- The ,X bit specifies address relative to the contents of the X register, post-indexing. The indexing by ,X takes place after a possible indirect addressing.

---

* Excepted from this is the conditional jump, the byte, and the register block instructions.

If all the ,X, I and ,B bits are zero, the normal relative addressing mode is specified. The effective address is equal to the contents of the program counter plus the displacement, (P) + disp.

The displacement may consist of a number ranging from -128 to +127. Therefore this addressing mode gives a dynamic range for directly addressing 128 locations backwards and 127 locations forwards.

Generally, a memory reference instruction will have the form:

<operation code> <addressing mode> <displacement>

Note that there is no addition in execution time for relative addressing, pre-indexing, post-indexing or both. Indirect addressing, however, adds 0.9 μs to the listed execution time.

The address computation is summarized in Table 3.1. The symbols used are defined as follows:

,X	Bit 10 of the instruction
I	Bit 9 of the instruction
,B	Bit 8 of the instruction
disp.	Contents of bits 0-7 of the instruction (displacement)
(X)	Contents of the X register
(B)	Contents of the B register
(P)	Contents of the P register
()	Means contents of the register or word.

The effective address is the address of that memory location which is finally accessed after all address modifications (pre- and post-indexing) have taken place in the memory address computation.

,X	I	,B	Mnemonic	Effective Address
0	0	0		(P) + disp.
0	1	0	I	((P) + disp.)
0	0	1	,B	(B) + disp.
0	1	1	,B I	((B) + disp.)
1	0	0	,X	(X) + disp.
1	0	1	,B ,X	(B) + disp. + (X)
1	1	0	I ,X	((P) + disp.) + (X)
1	1	1	,B I ,X	((B) + disp.)+ (X)

Table 3.1 Addressing Modes

Wise and competent use of the NORD-12 addressing modes will result in efficient programs. Advanced readers may wish to skip the rest of this section after perusing Table 3.1, which summarizes the addressing structure.

P-relative Addressing ,X = 0 I=0 ,B=0

The P-relative addressing mode is specifed by setting the ,X I and ,B bits all to zero. In this mode the displacement bits (bits 0-7) specify a positive or negative 7-bit address relative to the current value of the program counter (P register).

Example:

Suppose memory location 403 contains the instruction 004002_8, which in this chapter we shall represent by STA * 2, and this instruction is executed. The ,X I and ,B bits are all set to zero indicating P-relative addressing, and a positive displacement of 2 is given; the contents of the A register will therefore be stored in memory location 405. If, instead location 403 contains the instruction JMP* -2 and it is executed, the next instruction to be executed will be taken from location 401. While there is an obvious limitiation to this mode of addressing (locations more than 128_8 words away from the instruction being executed cannot be accessed). this mode of addressing is still quite useful for doing local jumps and accessing nearby constants and variables.

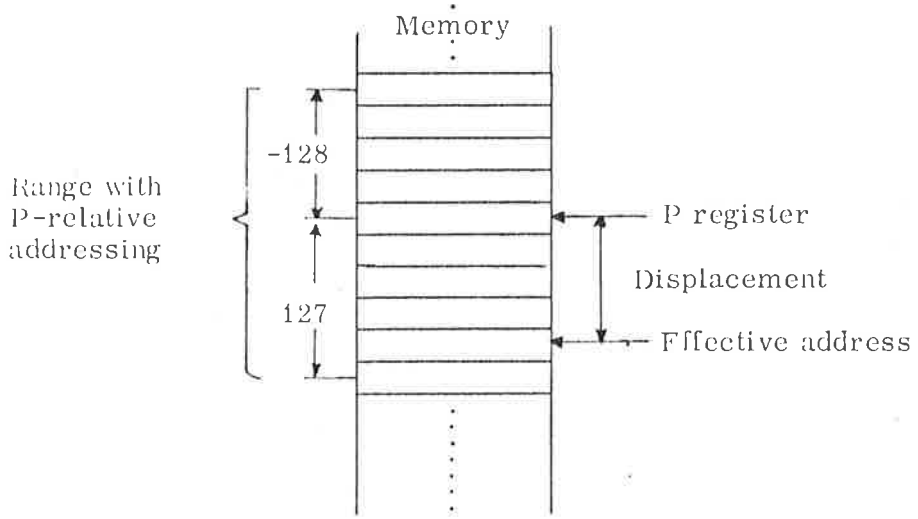


Figure 3.1: Schematic Illustration of P-relative Addressing

Indirect P-relative Addressing ,X=0 I=1 ,B=0

Since one must be able to access memory locations more than 128_{10} words away from the instruction being executed, the simplest method of doing this is to use the indirect P-relative addressing mode, specified by setting the I bit to one and the ,X bit and ,B bit to zero in memory address instructions. In this mode an address relative to program counter is computed, exactly as for P-relative addressing, by adding the displacement to the value of the program counter; but, rather than the addressed location actually being accessed the contents of the addressed location are used as a 16-bit address of a memory location which is accessed instead.

Example:

Suppose location 405 contains the instruction LDA I*2 (045002_8), and this instruction is executed. Furthermore, suppose memory location 16003 contains the value 17, and the memory location 407 contains 016003. The net result of executing the instruction in location 405 is to load the value 17 into the A register. First the displacement, 2, of the LDA instruction is added to the value of the location counter, 405, giving the result 407; then the contents of location 407, 16003. are used as an address and the contents of this address 17, are finally loaded into the A register.

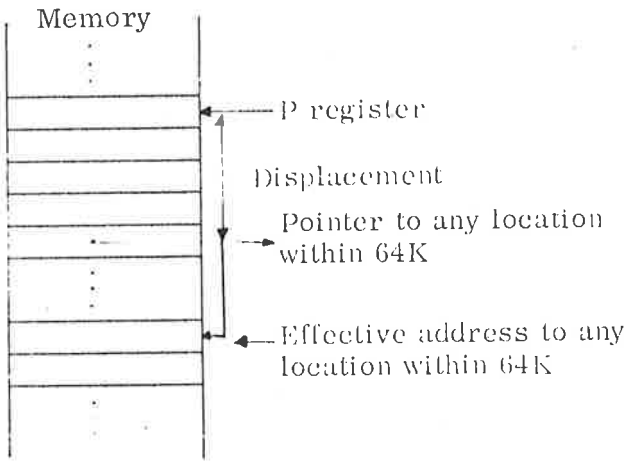


Figure 3.2: Schematic Illustration of indirect P-relative addressing

B-relative Addressing ,X=0 I=0 ,B=1

The above two addressing modes are quite sufficient; in fact theoretically, either one alone is sufficient. However, if the NORD-12 provided only one or both of the two addressing modes already described, it would not be particularly convenient for program efficiency. For instance, suppose that two subprograms each a couple of hundred words long, need to communicate, Within each subprogram memory accesses are commonly made using P-rcaltive addressing, or occasionally, indirect P-relative addressing. But between the subprograms indirect P-relative addressing would have to be used almost exclusively since, in general, locations in one subprogram, which instructions in the other subprogram must access, will not be less than 128 words apart. But this is very inefficient since both subprograms must contain indirect pointers to data and instructions local to the other subprogram.

To overcome this difficulty another addressing mode is available B-relative addressing, which permits both subprograms to directly address a common data area. B-register relative addressing is specified by setting the ,X and I bits to zero and the ,B bit to one in memory address instructions. This addressing mode is quite closely related to P-relative addressing, but instead the displacement is added to the current value of the B register, the resultant sum is used to specify the memory location accessed.

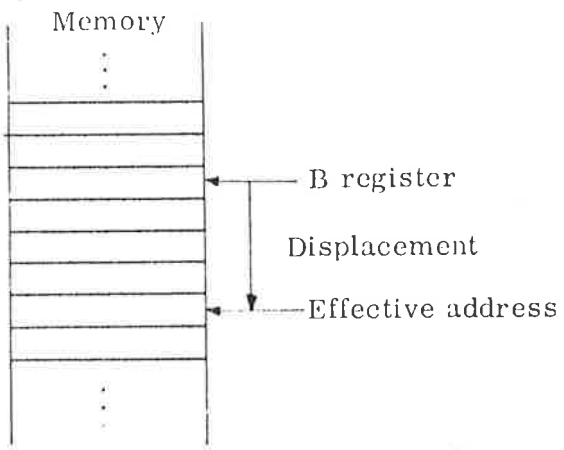


Figure 3.3: Schematic illustration of B-relative addressing

Example:

Let location 405 contain the instruction LDA -4 ,B (044774_8) and the B register contains the value 10035. Execute the instruction in location 405. This causes the contents of location 10031 to be loaded into the A register. The minus 4 in the displacement field of the LDA instruction in location 405 is added to the contents of the B register, 10035, giving a sum of 10031, and the contents of location 10031 are loaded into the A register.

Indirect B-relative Addressing ,X=0 I=1 ,B=1

Naturally, there is also an indirect B-relative addressing mode which is specified by setting the ,B and I bits to one and the ,X bit to zero in memory reference instructions. This mode has the same relationship to B-relative addressing that indirect P-relative addressing has to P-relative addressing. This permits a subprogram to access data or locations in other subprograms indirectly via pointers in an area common to several subprograms. This address mode is used extensively for calling library routines.

Example:

Let location 10031 contain the instruction JPL I 3 ,B (135403_8) and the B register contains 400, a pointer to an area common to several subprograms. Furthermore, let location 403 contain the value 2000. If the instruction in location 10031 is executed, the subroutine beginning at location 2000 will be called. The displacement, 3, in the JPL instruction is added to the contents of the B register, 400, giving a result of 403. The contents of locations 403, 2000, are then used as a pointer to the subroutine.

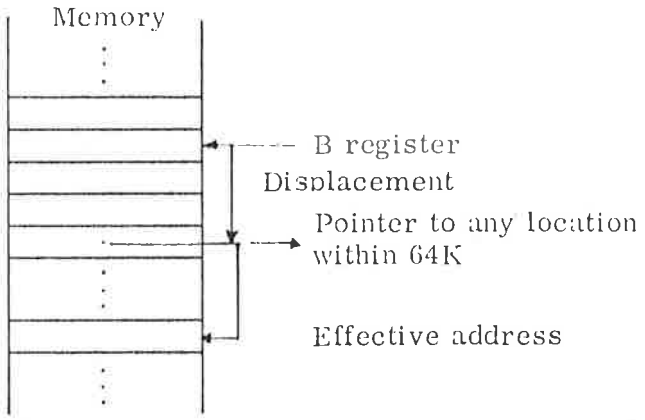


Figure 3.4: Schematic illustration of indirect B-relative addressing

X-relative (or Indexed) Addressing , X=1 I=0 , B=0

The other four addressing modes all involve use of the X register. The simplest of these is X-relative addressing which works like P- and B-relative addressing, but the displacement is added to the X register's contents during the address calculation instead of to the contents of the P or B register. This addressing mode is often used for randomly accessing the elements of a block of data.

Example:

Let a recursive subroutine when being called save the contents of the L, A and B registers in a three word block on a pushdown stack, and the X register point to the first free register in the stack. The following code might then be found at the beginning of the recursive subroutine:

```
SUB,    STA  1, X
        COPY SL DA
        STA  2, X
        COPY SB DA
        STA  0, X
        AAX  3
        ...
        ...
        ...
```

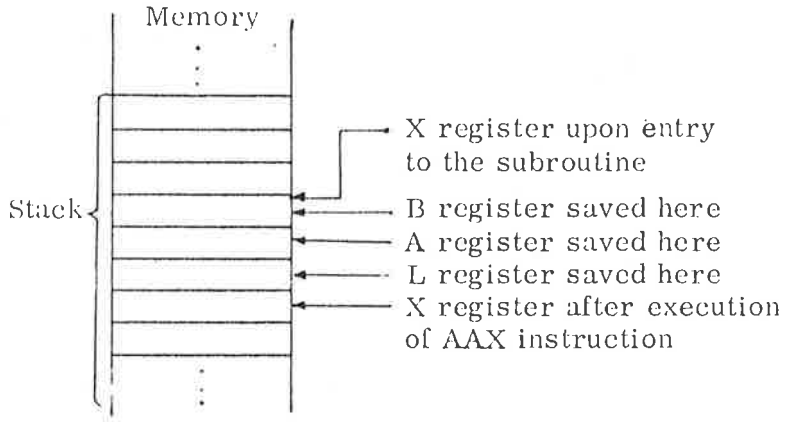


Figure 3.5: The effect of this code is illustrated in the figure

For another example reread B-relative addressing mentally substituting "X register" for "B register".

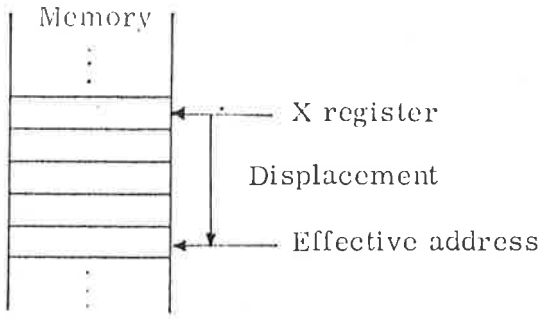


Figure 3.6: Schematic illustration of X-relative addressing

B-relative Indexed Addressing ,X=1 I=0 ,B=1

When the ,X and ,B bits are set to one and the I bit to zero in memory reference instructions, the mode is called B-relative indexed addressing. In this mode the contents of the X and B registers and the displacement are all added together to form the effective address.

B-relative indexed addressing is often very useful; for instance, when accessing row by row elements of a two-dimensional array stored column by column.

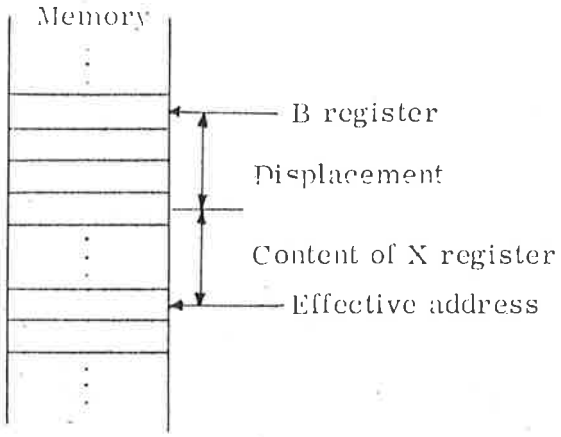


Figure 3.7: Schematic illustration of B-relative indexed addressing

Indirect P-relative Indexed Addressing ,X=1 I=1 ,B=0

The last two addressing modes are rather difficult to describe, but very useful. Indirect P-relative indexed addressing is selected by setting the ,X and I bits to one and the ,B bit to zero in the memory address instruction. This mode allows successive elements of an array arbitrarily placed in memory to be accessed in a convenient manner.

The address calculation in the mode takes place as follows:
The contents of the P register, say 4002, are added to the displacement say -1, and produce a sum, 4001. The contents of the location 4001, say 10100, are added to the contents of the X register, say -100, to produce a new sum, 10000, the effective address, By incrementing the X register, successive locations may be accessed, For instance: using the above example, locations 10000 through 10100 can be successively accessed by stepping the contents of the X register from -100 to zero.

Readers are advised to go over this example carefully: Stepping through an array in this fashion is done very often.

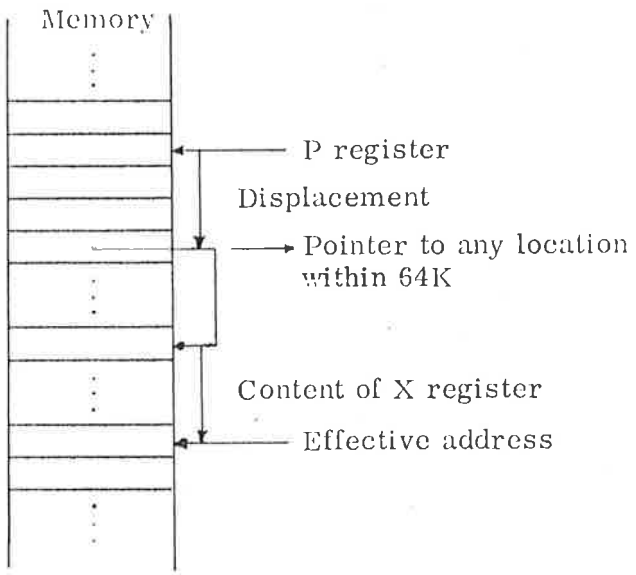


Figure 3.8: Schematic illustration of indirect P-relative indexed addressing

Indirect B-relative Indexed Addressing ,X=1 I=1 ,B=1

The final addressing mode, indirect B-relative indexed addressing, is identical to indirect P-relative indexed addressing except that the contents of the B register are used in place of the contents of the P register in the effective address computation. This mode can therefore by used to step through arrays pointed to from a data area common to several subprograms.

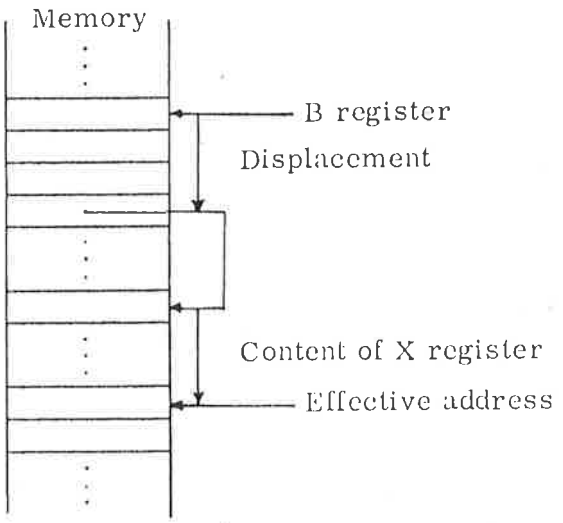


Figure 3.9: Schematic illustration of indirect B-relative indexed addressing

As an example of efficient use of different addressing techniques, we will write a general program which moves an array starting in location ABEG and has a length which is stored in location LONG, to a location starting in BBEG.

```
INIT,    LDA  (ABEG
         ADD LONG
         COPY SA DB          % (B) = ABEG + (LONG)
         LDA  (BBEG
         ADD  LONG
         STA  TEMP           % (TEMP) = BBEG + (LONG)
         LDX  LONG
         COPY CM2 SX  DX     % (X) = -(LONG)

LOOP,    LDA  ,X  ,B
         STA  I  TEMP  ,X
         JNC    LOOP         % INCREMENT,
                             % TEST AND JUMP
                             % FINISHED
```

```
DONE,   -
        -
        -

TEMP,   0
```

As is shown, the innerloop consists only of 3 instructions (or 64 bits when the indirect address is also counted).

The X register is used to step through both arrays, and it is initialized to contain the two's complement of the length of the arrays. This makes it possible to combine the incrementing of X, the test for completion and the jump into one instruction JNC (see Section 3.1.5). Because the X register is now reserved, we use the B-register to compensate for the correct start address in the LDA instruction, and because both the X and B register are now reserved, we have to use an indirect address for the STA instruction. (Note that this example gives the shortest program, not the fastest!)

3.1.2 Store Instructions

STZ — Store zero — Code 000 000

Format: STZ <adr. mode><disp.>

The effective location is cleared.

Affected: (EL) — Time: 2.2 μs

STA — Store A register — Code: 004 000

Format: STA <adr. mode><disp.>

The contents of the A register are stored in the effective location.

Affected: (EL) — Time: 2.2 μs

STT — Store T register — Code: 010 000

Format: STT <adr. mode><disp.>

The contents of the T register are stored in the effective location.

Affected: (EL) — Time: 2.2 μs

STX — Store X register — Code: 014 000

Format: STX <adr. mode><disp.>

The contents of the X register are stored in the effective location. The address of this instruction may be modified by the contents of the X register.

Affected: (EL) — Time: 2.2 μs

STD — Store Double word — Code: 020 000

Format: STD <adr. mode><disp.>

The contents of the A register are stored in the effective location, and the contents of the D register are stored in the effective location plus one.

Affected: (EL) , (EL+1) — Time: 3.5 μs

STF — Store floating accumulator — Code: 030 000

Format: STF <adr. mode><disp.>

The contents of the floating accumulator are stored in three memory locations, starting with exponent part in effective location.

Affected: (EL), (EL+1), (EL+2) — Time: 4.3 μs

MIN — Increment memory and skip if zero — Code: 040 000

Format: MIN <adr. mode><disp.>

Effective word is read and incremented by one and then restored in the effective location. If the result becomes zero, the next instruction is skipped.

Affected (EL), (P) — Time: 5.0 μs

3.1.3 Load Instructions

LDA — Load A register — Code: 044 000

Format: LDA <adr. mode> <disp.>

The effective word is loaded into the A register.

Affected: (A) — Time: 2.3 μs

LDT — Load T register — Code: 050 000

Format: LDT <adr. mode><disp.>

The effective word is loaded into the T register.

Affective: (T) — Time: 2.3 μs

LDX — Load X register — Code: 054 000

Format: LDX <adr. mode><disp.>

The effective word is loaded into the X register. The address of this instruction may be modified by the previous contents of the X register.

Affected: (X) — Time: 2.3 μs

LDD — Load double word — Code: 024 000

Format: LDD <adr. mode><disp.>

The contents of the effective location are loaded into the A register, and the contents of the effective location plus one are loaded into the D register.

Affected: (A), (D) — Time: 3.6 μs

LDF — Load floating accumulator — Code: 034 000

Format: LDF <adr. mode><disp.>

The contents of the effective location and the two following locations are loaded into the floating accumulator, i.e. T, A and D registers.

Affected: (T), (A), (D) — Time: 4.5 μs

3.1.4 Arithmetical and Logical Instructions

ADD — Add to A register — Code: 060 000

Format: ADD <adr. mode><disp.>

The effective word is added to the A register with the result in the A register. The carry indicator is set to 1 if a carry occurs from the sign bit position of the adder, otherwise the carry indicator is reset to 0. If the signs of the two operands are equal but the sign of the result is different, overflow has occurred, and both the dynamic and static overflow indicators are set to one. If the condition for overflow does not exist, the dynamic overflow indicator is reset to 0, while the static overflow indicator is left unchanged.

Affected: (A), C, O, Q — Time: 2.3 μs

SUB — Subtract from A register — Code: 064 000

Format: SUB <adr. mode><disp.>

The 2's complement of the effective word is formed and added to the contents of the A register with the result in the A register. The same rules as for ADD apply for the setting of the overflow and carry indicators.

Affected (A), C, O, Q — Time: 2.3 μs

AND — Logical and — Code: 070 000

Format: AND <adr. mode> < disp.>

The logical product of the effective word and the contents of the A register are formed, with the result in the A register. The logical product contains a one in each bit position for which there is a corresponding one in both the A register and the effective word, otherwise the bit position contains a zero.

Affected: (A) — Time: 2.3 μs

ORA — Logical inclusive or — Code: 074 000

Format: ORA <adr. mode> < disp.>

Logical inclusive or is formed between the effective word and the contents of the A register, with the result in the A register. Logical inclusive or contains a zero in each bit position for which there is a corresponding zero in both the A register and the effective word, otherwise the bit position contains a one.

Affected: (A) — Time: 2.3 μs

MPY — Multiply integer — Code: 120 000

Format: MPY <adr. mode><disp.>

The effective word and the A register are multiplied and the result is placed in the A register. Both numbers are regarded as signed integers and the result as a 16-bit signed integer. If the result in absolute value is greater than 32767, overflow has occurred and the static and dynamic overflow indicators are set to one.

Affected: (A), O, Q — Time: 16.7 μs

FAD — Add to floating accumulator — Code: 100 000

Format: FAD <adr. mode> <disp.>

The contents of the effective location and the two following locations are added to the floating accumulator with the result in the floating accumulator. The previous setting of the carry and overflow indicators is lost.

Affected: (T), (A), (D), C, O, Q — Time: 7.5 - 32.8 μs

FSB — Subtract from floating accumulator — Code: 104 000

Format: FSB <adr. mode> <disp.>

The contents of the effective location and the two following locations are subtracted from the floating accumulator with the result in the floating accumulator. The previous setting of the carry and overflow indicators is lost.

Affected: (T), (A), (D), C, O, Q — Time: 8.0 - 33.3 μs

FMU — Multiply floating accumulator — Code: 110 000

Format: FMU <adr. mode> <disp.>

The contents of the floating accumulator are multiplied with the number of the effective floating word locations with the result in the floating accumulator. The previous setting of the carry and overflow indicators is lost.

Affected: (T), (A), (D), O, Q — Time: 27.8-29.3 μs

FDV — Divide floating accumulator — Code: 114 000

Format: FDV <adr. mode> <disp.>

The contents of the floating accumulator are divided by the number in the effective floating word locations. Result in floating accumulator. If division by zero is tried, the error indicator Z is set to one. The error indicator Z may be sensed by a BSKP instruction (see BOP). The previous setting of the carry and overflow indicators is lost.

Affected: (T), (A), (D), Z, C, O, Q — Time: 11.2-31.7 μs

3.1.5 Sequencing Instructions

JMP — Jump — Code: 124 000

Format: JMP <adr. mode><disp.>

The effective address is loaded into the program counter, and the next instruction is taken from the effective address of the JMP instruction.

Affected: (P) — Time: 2.3 μs

JPL — Transfer P to L and jump — Code: 134 000

Format: JPL <adr. mode><disp.>

The contents of the program counter are transferred to the L register, the effective address is loaded into the program counter, and the next instruction is taken from the effective address of the JPL instruction. Note that the program counter points to the instruction after the jump (it has been incremented before transfer to the L register).

Affected: (P), (L) — Time 2.3 μs

CJP — Conditional jump

Instruction bits 8-10 are used to specify one of 8 jump conditions. If the specified condition becomes true, the displacement is added to the program counter and a jump relative to current location takes place. The range is 128 locations backwards and 127 locations forwards. If the specified condition is false, no jump takes place.
Execution time depends on condition, but is the same for all instructions.

A conditional jump instruction must be specified by means of the eight mnemonics listed below. It is illegal to spedify CJP followed by any combination of ,B I and ,X.

The eight jump conditions are:

JAP — Jump if A register is positive or zero, A bit 15=0 — Code: 130 000

Format: JAP <disp.>

JAN — Jump if A register is negative, A bit 15=1 — Code: 130 400

Format: JAN <disp.>

JAZ — Jump if A register is zero — Code: 131 000

Format: JAZ <disp.>

JAF — Jump if A register is filled (not zero) — Code: 131 400

Format: JAF <disp.>

JXN — Jump if X register is negative, i.e., X bit 15=1. — Code: 133 400

Format: JXN <disp.>

JXZ — Jump if X register is zero — Code: 133 000

Format: JXZ <disp.>

JPC — Count and jump if register is positive or zero. — Code: 132 000

Format: JPC <disp.>

X is incremented by one, and if the X bit 15 equals zero after the incrementation, the jump takes place.

JNC — Count and jump if X register is negative. — Code: 132 400

Format: JNC <disp.>

X is incremented by one; if then the X bit 15 equals one, the jump takes place.

Affected: (P) and (X) for JPC and JNC.

Time:
Condition false: 1.8 μs
Condition true: 2.3 μs

3.1.6 Byte Instructions

To facilitate the handling of character strings, the NORD-12 provides two instructions for byte handling, load byte, LBYT, and store byte, SBYT.

Because of the requirement of full 64K addressing, the LBYT and SBYT use an addressing scheme different from the normal NORD-12 addressing.

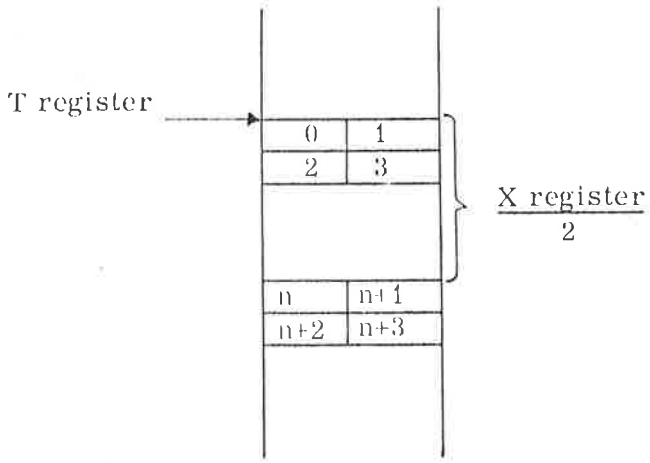
For byte addressing, two of the NORD-12 registers, the T and X registers are used for addressing the byte.

The contents of the T register point to the beginning of the character string, and the contents of the X register point to a byte within this string, Thus the address of the word which contains the byte equals

$$(T) + \frac{1}{2}(X)$$

If the X register is even (, $X_0 = 0$,) the byte is in the left part of the word, if $X_0 = 1$, the byte is in the right part of the word.

A byte consists of eight bits.



The specifications for the two byte instructions are then as follows:

LBYT Load byte — Code: 142 200

Format: LBYT

The 8-bit byte specified by the contents of the T and X registers is loaded into the A register bits 0-7, with the A register bits 8-15 cleared.

Affected: (A)

Time:
Left byte: 11.4 μs
Right byte: 6.8 μs

SBYT Store byte

Format: SBYT — Code: 142 600

The byte contained in the A register bits 0-7 is stored in one half of the effective location pointed by the T and X registers, the second half of this effective location being unchanged. The contents of the A register are unchanged.

Affected: (EL)

Time:
Left byte: 16.5 μs
Right byte: 12.0 μs

3.1.7 Register Block Instructions

To facilitate the programming of registers on different program levels, two instructions, SRB and LRB, are available for storing and loading of a complete register block to and from memory.

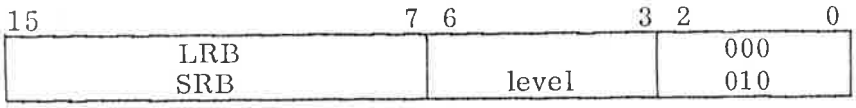
A register block always consists of the following registers in this sequence:

P	Program counter
X	X register
T	T register
A	A register
D	D register
L	L register
STS	Status register bits 2-7, bit 0-1 and bits 8-15 are zero
B	B register

The addressing for these two instructions is as follows:

The contents of the X register specify the effective memory address from where the register block is read or written into.

The specifications for the two instructions are as follows:



SRB Store Register Block Code: 152 402

Format: SRB <level$_8$ * 10$_8$>

The instruction SRB<level$_8$ * 10$_8$>stores the contents of the register block on the program level specified in the level field of the instruction. The specified register block is stored in succeeding memory locations starting at the location specified by the contents of the X register.

If the current program level is specified, the stored P register points to the instruction following SRB.

Affected: (EL), (EL+1) ... (EL+7) Time: 16.5 μs

Example:

Let the contents of the X register be 042562, then the instruction

SRB 140$_8$

stores the contents of the register block on program level 12 into the memory addresses 042562, 042563,....., 042571.

LRB Load Register Block Code: 152 600

Format: LRB <level$_8$ * 10$_8$>

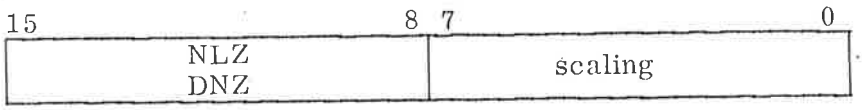
The instruction < LRB level$_8$ * 10$_8$> loads the contents of the register block on program level specified in the level field of the instruction. The specified register block is loaded by the contents of succeeding memory locations starting at the location specified by the contents of the X register. If the current program level is specified, the P register is not affected.

Affected: All the registers on specified program level are affected. Note: If the current level is specified, the P register is <u>not</u> affected.

Time: 15.8 μs

3.2 Operate Instructions

3.2.1 Floating Point Conversion Instructions



Two instructions are available. A single precision fixed point number may be converted to a standard form floating point number. A floating point number may be converted to a fixed point single precision number For both instructions the scaling factor is specified in the displacement part of the instruction. The range of the scaling factor is from -128 to +127, which gives a conversion range from approximately 10^{-39} to 10^{39}. The execution time depends on the scaling factor and the argument to convert.

The two sub-instructions are:

NLZ Normalize Code: 151 400

Format: NLZ <scaling>

Converts the number in the A register to a standard form floating number in the floating accumulator, using the scaling of the NLZ instruction as a scaling factor. For integers the scaling factor should be +16, a larger scaling factor will result in a higher floating point number. Because of the single precision fixed point number, the D register will be cleared.

Affected: (T), (A), (D), Time: 4.5-14.5 μs

DNZ Denormalize Code: 152 000

Format: DNZ <scaling>

Converts the floating number in the floating accumulator to a single precision fixed point number in the A register, using the scaling of the DNZ instruction as a scaling factor. When converting to integers, the scaling factor should be -16, a greater scaling factor will cause the fixed point number to be greater. After this instruction the contents of the T and D registers will all be zeroes.

If the conversion causes underflow, the T, A and D registers will all be set to zero.

If the conversion causes overflow,* the error indicator Z is set to one. Overflow occurs if the resulting integer in absolute value is greater than 32767.

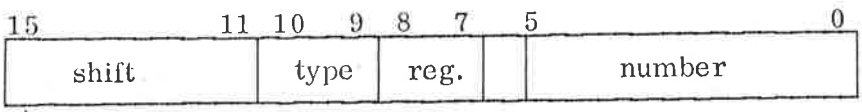
The conversion will truncate, and negative numbers are converted to positive numbers before conversion. The result will again be converted to a negative number.

Some examples:

T-A-D before conversion (in decimal)		A-after conversion
0.9	DNZ -20_8	0
3.141592	DNZ -20_8	3
3.141592	DNZ -17_8	6
3.141592	DNZ -16_8	12
3.7	DNZ -20_8	3
3.7	DNZ -17_8	7
3.7	DNZ -21_8	1
-3.141592	DNZ -20_8	-3
-3.7	DNZ -20_8	-3
32768.0	DNZ -20_8	Overflow
-32768.0	DNZ -20_8	Overflow

Affected: (A), (T), (D), Z Time: 5.5-15.5 μs

3.2.2 Shift Instructions



Shift instructions operate on registers. A shift instruction consists of three parts: the register to be shifted, specified by the shift reg. fields, type of shift to be performed, specified by the type field; and the number of shifts to be performed, specified by the number field. A shift instruction will have to form:

<shift register> <type> <number>

---

* The overflow test is fail-proof for a scaling constant of -20_8 only.

Every shift instruction causes the last bit which is discarded to be contained in the M; the multi shift link indicator. This may then be used as end input for the next shift instruction.

Note that bit 6 in the instruction is ignored.

The time of a shift instruction is independent of the type of shift.

The following four specifications of the <shift register> are available:

SHT — Shift the T register — reg. field 00 — Code: 154 000

Format: SHT<type><number>

The T register is shifted as specified by the <type> and <number>.

Affected: (T), M — Time: $2.9+0.5 \cdot N_0$

SHD — Shift the D register — reg. field 01 — Code: 154 200

Format: SHD<type><number>

The D register is shifted as specified by the <type> and <number>.

Affected: (D), M — Time: $2.9+0.5 \cdot N_0$

SHA — Shift the A register — reg. field 10 — Code: 154 400

Format: SHA<type><number>

The A register is shifted as specified by the <type> and <number>

Affected: (A), M — Time: $2.9+0.5 \cdot N_0$

SAD — Shift the A and D registers connected — reg. field 11 — Code: 154 600

Format: SAD<type><number>

Bit 0 of the A register is connected to bit 15 of the D register.

Affected: (A), (D), M — Time: $4.4+0.5 \cdot N_0$

type field

For each shift instruction the following four types of shift can be specified, one at a time:

Mnemonic		type field	
nil	Arithmetical shift. During right shifts the sign bit (bit 15) is extended during the shifting, in left shift zeroes are fed into vacated bit positions.	0 0	Code: 000 000
ROT	Rotational shift. In single register shift bit 0 is connected to bit 15, in double shifts bit 0 of the D register is connected to bit 15 of the A register.	0 1	Code: 001 000
ZIN	Zero end input.	1 0	Code: 002 000
LIN	Link end input. The contents of the M indicator will be shifted into the vacated bit(s).	1 1	Code: 003 000

number field

The <number> of the instruction in the number field is a signed number, 5 bits plus sign, which specifies the shift direction (positive or negative shift) and the number of shifts.

$N \geq 0$, i.e., if bit 5 = 0 then shift left

$N < 0$, i.e., if bit 5 = 1 then shift right

The maximum number of shifts is 31 left shifts and 32 right shifts.

Only the A, T and D registers may be shifted, If any other register is to be shifted, its contents must first be placed in the A, T or D register.

If no shift direction is specified, left shift is assumed.

The number of shifts is interpreted by the assembler as an octal number.

A right shift may be specified either by the correct 6 bit negative shift count or by writing the mnemonic code SHR followed by the positive number of right shifts. A shift instruction to shift the accumulator 3 positions to the right may be specified by one of the following identical instructions:

```
SHA     75₈
SHA  100-3₈
SHA SHR  3₈
```

In a right shift, nothing should be written between the SHR mnemonic and the number of right shifts * (a space to distinguish between SHR and the number is necessary). SHR must be the last mnemonic used in the instruction.

Some examples of correctly specified shift instructions:

Example 1

Shift the A and D registers connected 8 positions (octal 10) left.

```
SAD 10₈
```

Example 2

Rotate the T register 6 places to the left.

```
SHT ROT  6
```

Example 3

Shift the connected A and D registers 16 positions to the left. Rotate shift is specified, which in this case will cause the contents of the A and D registers to be exchanged. The same effect may be obtained by means of a SWAP SA DD instruction.

```
SAD ROT  20
```

Example 4:

Shift the D register two places to the right. Feed zeroes into the left end during the shifting. Bits 15 and 14 in the D register will become zero.

```
SHD ZIN SHR  2
```

* This is an assembler peculiarity.

3.2.3 Register Operations

The register operation instructions specify operations between any two general registers; a source register, sr, and a destination register, dr. Any instruction may consist of the parts:

⩤register operation><subinstruction><sr><dr>

There are ten basic register operations belonging to the two groups:

ROP register operations	Section 3.2.3.1
EXTended register operation instructions	Section 3.2.3.2

In addition there are two instructions for accessing single registers outside current program level, see Section 3.2.3.3, and two instructions for accessing a whole register block outside current program level, see Section 3.1.7.

Only the ROP instructions have subfields.

The ROP register instructions are:

RADD	Register addition, dr ← dr + sr	Code: 146 000
RSUB	Register subtractions, dr ← dr - sr	Code: 146 600
RAND	Register logical AND, dr ← dr ∧ sr	Code: 144 400
RORA	Register logical OR, dr ← dr ∨ sr	Code: 145 500
REXO	Register logical exclusive OR dr ← dr ∀ sr	Code: 145 000
SWAP	Register exchange, sr ← dr and dr ← sr	Code: 144 000
COPY	Register transfer, dr ← sr	Code: 146 100

The EXTended register instructions are:

RMPY	Integer inter-register multiply, AD ← dr * sr	Code: 141 200
RDIV	Integer inter-register divide AD/ < sr > ⇒ A ← Quotient D ← Remainder	Code: 141 600
EXR	Execute register, Instruction register ← sr	Code: 140 600
MIX3	Multiply index by 3, (X) ← ((A)-1) * 3	Code: 143 200

The source registers <sr> are specified as follows:

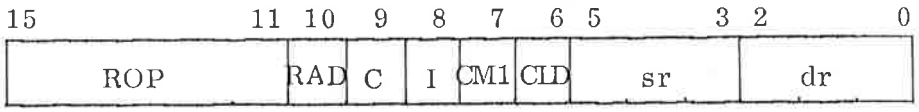
SD	D register	as source	Code: 10
SP	Program counter	as source	Code: 20
SB	B register	as source	Code: 30
SL	L register	as source	Code: 40
SA	A register	as source	Code: 50
ST	T register	as source	Code: 60
SX	X register	as source	Code: 70

If no source register is specified, zero will be taken as the source register.

The destination registers <dr> are specified as follows:

DD	D register	as destination	Code: 1
DP	Program counter	as destination	Code: 2
DB	B register	as destination	Code: 3
DL	L register	as destination	Code: 4
DA	A register	as destination	Code: 5
DT	T register	as destination	Code: 6
DX	X register	as destination	Code: 7

3.2.3.1 ROP Register Operation Instructions



The instruction decodes bits 0-10 as follows:

Bits 0-2 specify one out of seven registers to be the destination register. The destination register will be loaded with the result of the ROP instructions.

dr = 0 Normally a no - operation instructions, except that the carry indicator will be reset if RAD = 1.

Bits 3-5 specify the one out of eight registers which contains the value to be used as the source register operand.

sr = 0 Produces a source value equal to zero

CLD = 1 Clear destination register before operation. If the source and the destination register are the same, the register as source is not cleared.

CM1 = 1 Use complement (one's complement) of source register as operand. The source register remains unchanged.

Bits 8 and 9 are decoded in two different ways, depending on whether the RAD bit is zero or one.

RAD = 1 Add source to destination.

When RAD = 1, bits C and I are decoded as follows:

C = 1,
I = 0 Also add old carry to destination, ADC.

C= 0,
I = 1 Also add 1 to destination, AD1.

It is not possible both to add previous carry and to add 1 in the same ROP instruction. (If it is tried, 1 will be added regardless of the status of the carry indicator.)

RAD = 0 Binary register operations.

The C and I bits are decoded as follows:

C, I=0, 0 Register swap, destination and source exchanged, SWAP
0, 1 Logical and , RAND
1, 0 Logical exclusive or, REXO
1, 1 Logical inclusive or, RORA

If RAD = 1, the overflow and carry indicators are set according to the same rules as apply for ADD: if RAD = 0, the overflow and carry indicators remain unchanged.

The following groups of ROP mnemonics are mutually exclusive, i.e. only one may be used in a ROP instruction.

(SD, SP, SB, SL, SA, ST, SX)

Only one source register must be specified.

(DD, DP, DB, DL, DA, DT, DX)

Only one destination register must be specified.

(ADC, AD1)

Both 1 and old carry cannot be added in the same instruction

(RADD, RSUB, SWAP, RAND, REXO, RORA, COPY)

Only one type of operation must be specified.

(ADC, AD1, SWAP, RAND, REXO, RORA)

Add 1 or add carry may not be used together with the binary register operations.

(RSUB, CM1, ADC, AD1)

RSUB uses CM1 and AD1.

The recommended way to specify ROP instructions is to use the following mnemonics which will be correctly translated by the assembly language.

RADD,	dr ← dr + sr	Register addition
RSUB,	dr ← dr − sr	Register subtraction
RAND,	dr ← dr ∧ sr	Register logical AND
RORA,	dr ← dr ∨ sr	Register logical OR
REXO,	dr ← dr ∀ sr	Register logical exclusive OR
SWAP,	dr ↔ sr	Register exchange
COPY,	dr ← sr	Register transfer

Note that the ROP instruction is included in the above mentioned mnemonics.

Time: RADD, RSUB, RAND, REXO, RORA : 1.4 μs

Time: SWAP : 2.9 μs

If the P register is used as destination (DP), an additional micro cycle, 490 ns, will be required.

Decoding of RAD	C	I	CM1	CLD	Instructions					Result of Instructions
0	0	0	0	0	SWAP				<sr><dr>	sr ↔ dr
0	0	0	0	1	SWAP			CLD	<sr><dr>	dr ← sr, sr ← 0
0	0	0	1	0	SWAP	CM1			<sr><dr>	dr ← $\overline{sr}$, sr ← dr
0	0	0	1	1	SWAP	CM1		CLD	<sr><dr>	dr ← $\overline{sr}$, sr ← 0
0	0	1	0	0	RAND				<sr><dr>	dr ← dr ∧ sr
0	0	1	0	1	RAND			CLD	<sr><dr>	dr ← 0
0	0	1	1	0	RAND	CM1			<sr><dr>	dr ← dr ∧ $\overline{sr}$
0	0	1	1	1	RAND	CM1		CLD	<sr><dr>	dr ← 0
0	1	0	0	0	REXO				<sr><dr>	dr ← dr ∨ sr
0	1	0	0	1	REXO			CLD	<sr><dr>	dr ← sr
0	1	0	1	0	REXO	CM1			<sr><dr>	dr ← dr ∨ $\overline{sr}$
0	1	0	1	1	REXO	CM1		CLD	<sr><dr>	dr ← $\overline{sr}$
0	1	1	0	0	RORA				<sr><dr>	dr ← dr ∨ sr
0	1	1	0	1	RORA			CLD	<sr><dr>	dr ← sr
0	1	1	1	0	RORA	CM1			<sr><dr>	dr ← dr ∨ $\overline{sr}$
0	1	1	1	1	RORA	CM1		CLD	<sr><dr>	dr ← $\overline{sr}$
1	0	0	0	0	RADD[1)]				<sr><dr>	dr ← dr + sr
1	0	0	0	1	RADD[1)]			CLD	<sr><dr>	dr ← sr
1	0	0	1	0	RADD		CM1		<sr><dr>	dr ← dr + $\overline{sr}$
1	0	0	1	1	RADD[1)]		CM1	CLD	<sr><dr>	dr ← $\overline{sr}$
1	0	1	0	0	RADD	AD1			<sr><dr>	dr ← dr + sr + 1
1	0	1	0	1	RADD[2)]	AD1		CLD	<sr><dr>	dr ← sr + 1
1	0	1	1	0	RADD	AD1	CM1		<sr><dr>	dr ← dr − sr
1	0	1	1	1	RADD[1-2)]	AD1	CM1	CLD	<sr><dr>	dr ← − sr
1	1	0	0	0	RADD	ADC			<sr><dr>	dr ← dr + sr + c
1	1	0	0	1	RADD[1)]	ADC		CLD	<sr><dr>	dr ← sr + c
1	1	0	1	0	RADD	ADC	CM1		<sr><dr>	dr ← dr + $\overline{sr}$ + c
1	1	0	1	1	RADD[1)]	ADC	CM1	CLD	<sr><dr>	dr ← $\overline{sr}$ + c
1	1	1	0	0	Not applicable					
1	1	1	0	1						
1	1	1	1	0						
1	1	1	1	1						

Table 3.2 The ROP Instruction

This table shows all possible combinations of the ROP instructions and their results.

dr : destination register
sr : source register
$\overline{sr}$: one's complement of sr
c : old carry

---

1) RADD CLD is equal to COPY

2) RADD AD1 CM1 is equal to RSUB

The assembly language will also permit use of the following combined mnemonics:

CM2	CM1 AD1	Two's complement
EXIT	COPY SL DP	Return from subroutine
RCLR	COPY 0	Register clear
RINC	RADD AD1	Register increment
RDCR	RADD CM1	Register decrement

The mnemonics RCLR, RINC and RDCR should be followed only by the destination register specification.

Some examples of use of the ROP instruction:

Example 1:

Add the contents of the A and X register with the result in the X register:

```
RADD SA  DX
```

Example 2:

Complement (two's complement) the A register:

```
COPY CM2 SA  DA
```

Example 3:

Subtract the contents of the T register from the contents of the B register, with the result in the B register:

```
RSUB ST  DB
```

Example 4:

Increment the X register by one:

```
RINC DX
```

Example 5:

Decrement the L register by one. (One's complement of zero equals -1 in two's complement.)

```
RDCR DL
```

Example 6:

Clear the T register:

```
RCLR DT
```

Example 7:

Set the X register equal to one:

```
RCLR AD1 DX
```

Example 8:

Set the B register equal to minus one:

```
RCLR CM1 DB
```

Example 9:

Copy the contents of the X register into the T register:

```
COPY SX DT
```

Example 10:

Exchange the contents of the A and D registers:

```
SWAP SA DD
```

Example 11:

Form logical AND between the contents of the L and X registers with the result in the X register:

```
RAND SL DX
```

Example 12:

Copy the contents of the A register into the X register, and clear the A register (the CLD code causes a destination register of zero to be swapped).

```
SWAP CLD SA DX
```

Some short programs using ROP instructions:

Example 13:

Form the two's complement of the 32 bit double word in A and D:

```
        COPY CM2 SD   DD
        COPY CM1 ADC  SA  DA
```

Example 14:

Add together the two double wordlength numbers N1 and N2 with the result in the A and D registers:

```
        LDD    N1
        SWAP   SA    DD
        ADD    N2+1
        SWAP   SA    DD
        RADD   ADC   DA
        ADD    N2
```

Example 15:

Subroutine jump, and return from subroutine to main program:

```
                JPL     SUBR            % ERROR STOP
ERR.            WAIT
NORM,
-------------------------------
SUBR,           LDA     OLA
                SUB     PER
                SKP     IF      DA  EQL 0
                        EXIT            % ERROR EXIT
                        EXIT    AD1
```

The JPL instruction will place the address of the WAIT instruction into the L register. (When JPL is executed, the program counter points to the address after this instruction.)

The subroutine SUBR has two exits, one to the location immediately following the jump (EXIT), which in this case is an error exit, and one to the location two addresses after the jump.

Note: If the P register is used as source (SP), the P register has already been incremented and points to the next instruction.

3.2.3.2 EXTended Register Operation Instructions

RMPY — Integer inter-register multiply — Code: 141 200

Format: RMPY <sr><dr>

The sr and dr fields are used to specify the two operands to be multiplied (represented as two's complement integers), the codes are the same as for ROP, see Section 3.2.3.

The result is a 32-bit signed integer which will be placed in the A and D registers with the 16 most significant bits in the A register and the 16 least significant bits in the D register.

Affected: (A), (D) — Time: 15.9 μs

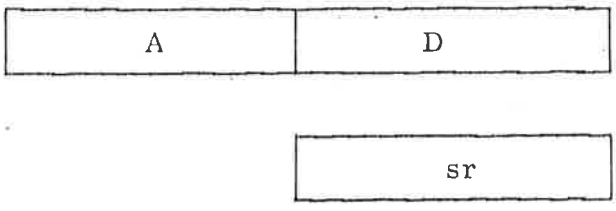
RDIV — Integer inter-register divide — Code: 141 600

Format: RDIV <sr>

The 32-bit signed integer contained in the double accumulator AD is divided by the contents of the register in the sr field, with the quotient in the A register and the remainder in the D register, i.e., AD/sr $\Rightarrow$ A $\leftarrow$ quotient, D $\leftarrow$ remainder.

The sign of the remainder is always equal to the dividend (AD), The destination field of the instruction is not used. If the division causes overflow, the error indicator Z is set to one.

The numbers are considered as fixed point integers with the fixed point after the right-most position.



Affected: (A), (D), Z, C, O, Q — Time: 6.0 - 24.0 μs

Example:

Before division		After division:		
Double accumulator	Divisor	A	D	Z
22	4	5	2	0
-22	4	-5	-2	0
378452	-16	-23653	-4	0
32767	1	32767	0	0
32768	1			1
65535	2	32762	1	0

EXR — Execute register — Code: 140 600

Format: EXR <sr>

The contents of the register specified in the <sr> field of the instruction are transferred to the instruction register, and the contents are then executed as an instruction.

Note: If the instruction specified by the contents of <sr> in a memory reference instruction with relative addressing, the address will be relative to the EXR <sr> instruction. If the instruction specified by the contents of <sr> is a JPL instruction, the L register will point to the instruction after the EXR <sr>.
Note also that it is illegal to have an EXR <sr>, where the contents of <sr> are a new EXR <sr>, if it is tried, the error indicator Z is set to one.

Affected: (IR), affections of the specified instructions.

Time: 3.8 μs

MIX3 — Multiply index by 3. — Code: 143 200

Format: MIX3

The X register is set equal to the contents of the A register minus one multiplied by three, i.e.,

(X) $\leftarrow$ [(A) - 1] * 3

Affected: (X)

Time: 2.0 μs

3.2.3.3 Inter Level Register Instructions

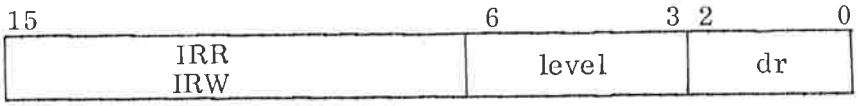
In the NORD-12 there are 16 complete sets of registers and status indicators, one set for each level,

The access to and from registers outside the current program level is by two instructions:

IRR Inter Register Read

IRW Inter Register Write

The format of this instruction is as follows:



Bits 0-2 specify the register to be read, using the same codes and mnemonics as are used for specifying destination registers for the register operations, see Section 3.2.3.

Bits 3-6 specify the program level number. It is possible to read the current program level as well as all outside program levels.

IRR Inter register read Code: 153 600

Format: IRR $\angle level_8 * 10_8 > <dr>$

This instruction is used to read into the A register on current program level one of the general registers inside/outside current program level. If bits 0-2 are zero, the status register on specified program level will be read into the A register bits 1-7, with bits 8-15 and bit 0 cleared.

Time: 2.9 μs

Example:

The instruction IRR 160 DP will copy the contents of the program counter on program level 14 into the A register on current program level.

IRW Inter register write Code: 153 400

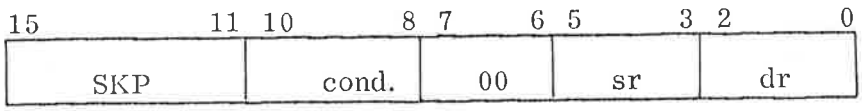
Format: IRW <level$_8$ * 10$_8$><dr>

This instruction is used to write the A Register on current program level into one of the general registers. It is also possible to write into the registers on current level. Then, if the P register is specified, the IRW instruction will be a dummy instruction. If bits 0-2 are zero, the A register bits 1-7 are written into the status register on specified level. Time: 2.9 μs

Example:

The instruction IRW 110 will copy the bits 0-7 of the A register on current program level into the status register on program level 9.

3.2.4 Skip Instructions



SKP Skip next instruction if specified condition is true. Code: 140 000

Format: SKP <dr><cond.><sr>

The cond. field specifies one of eight conditions between the registers dr and sr. If the specified condition is true, the next instruction is skipped. If not, the next instruction is not skipped. The registers dr, destination register, and sr, source register, are specified as for register operation registers, see Section 3.2.3.

Note that bits 6 and 7 are both zero. Otherwise, the instruction would belong to the EXTended instruction, see Section 3.2.3.2.

The SKP conditions test upon the result of the arithmetic expressions (dr) - (sr) which set the four indicators:

s - sign
z - result zero
c - carry
o - overflow

Time:
No skip: 1.8 μs
Skip: 2.3 μs

The eight SKP conditions are a s follows:

Mnemonic	Cond. field	Condition true if:	
EQL	0 0 0	z=1	Equal. The condition tests for equality between the source and destination registers (dr)=(sr)=0.
GEQ	0 0 1	s=0	Greater or equal to. (dr)-(sr) $\geq$0. The contents of the source and destination registers are treated as signed numbers. Overflow is not taken care of.
GRE	0 1 0	sVo=0	Greater or equal to. (dr)-(sr) $\geq$0. The contents of the source and destination registers are treated as signed numbers. Overflow is taken care of.
MGRE	0 1 1	c=1	Magnitude greater or equal to. (dr)-(sr) $\geq$ 0. The contents of the source and destination registers are treated as unsigned magnitudes, where 000 000 is the lowest and 177 777 the highest number. Overflow is taken care of.
UEQ	1 0 0	z=0	Unequal to. The condition tests for equality between the source and destination registers (dr) $\neq$ (sr) $\neq$ 0
LSS	1 0 1	s=1	Less than (dr)-(sr) $<$ 0. The contents of the source and destination registers are treated as signed numbers. Overflow is not taken care of

Mnemonic	Cond. field	Condition true if:	
LST	1 1 0	sVo=1	Less than (dr)-(sr) $\prec$ 0. The contents of the destination and source registers are treated as signed numbers. Overflow is taken care of.
MLST	1 1 1	c=0	Magnitude less than (dr)-(sr) $\prec$ 0. The contents of the source and destination registers are treated as unsigned magnitudes, where 000 000 is the lowest number and 177 777 is the highest number. Overflow is taken care of.

By swapping the register code in the sr and dr fields and inverting the relationship code, it is also possible to test these relationships.

$>$ Greater than

$\leq$ Less than or equal

The programmer is advised to use the same format as in these examples when specifying a skip instruction. (The mnemonic IF and the number 0, which both have the value zero, are used for easy readability).

Comparing a register with zero:

SKP IF DL UEQ	0	Skip if L register $\neq$ 0
SKP IF DX GRE	0	Skip if X register $\geq$ 0
SKP IF DB LSS	0	Skip if B register $\prec$ 0
SKP IF 0 LSS	ST	Skip if T register $>$ 0
SKP IF 0 GRE	SD	Skip if D register $\prec$ 0

Comparing the arithmetic value of the contents of two registers:

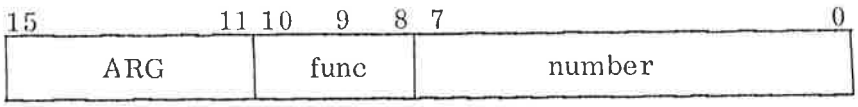
SKP IF DD EQL	SL	Skip if D register = L register
SKP IF DT UEQ	SX	Skip if T register $\neq$ X register
SKP IF DB LSS	SA	Skip if B register $\prec$ A register (or A register $>$ B register)
SKP IF DX GRE	SB	Skip if X register $\geq$ B register (or B register $\leq$ X register)

Comparing two magnitude numbers:

SKP IF DL MGRE	ST	Skip if L register $\geq$ T register (or T register $\leq$ L register)
SKP IF DB MLST	SX	Skip if B register $<$ X register (or X register $>$ B register)

The magnitude tests are especially useful when comparing the relationship between memory addresses which are represented as magnitude numbers in a computer with more than 32K memory.

3.2.5 Argument Instructions



Argument instructions operate on registers. The function field is used to specify one out of eight argument instructions. The number field is used to specify the argument, a signed number ranging from -128 to 127.

Bits 8 and 9 in the function field specify one out of four registers, B, A, T or X, and bit 10 one of the operations: set argument to or add argument to.

The eight argument instructions are:

SAA	Set argument to A register Format: SAA <number>	Code: 170 400
AAA	Add argument to A register Format: AAA <number>	Code: 172 400
SAX	Set argument to X register Format: SAX <number>	Code: 171 400
AAX	Add argument to X register Format: AAX <number>	Code: 173 400
SAT	Set argument to T register Format: SAT <number>	Code: 171 000
AAT	Add argument to T register Format: AAT <number>	Code: 173 000

SAB	Set argument to B register	Code: 170 000
	Format: SAB <number>	
AAB	Add argument to B register	Code: 172 000
	Format: AAB <number>	Time: 1.4 μs

An argument instruction should be specified by means of one of the eight mnemonics listed above.

Examples of argument instructions:

Example 1:

Set the contents of the T register equal to 13_8. Bits 8-15 will become zero:

SAT 13_8

Example 2:

Set the contents of the B register equal to -28_8. Bits 8-15 will become one. sign extension:

SAB -26_8

Example 3:

Add 3 to the contents of the X register. The addition is modulo 2^{15}.

AAX 3

Example 4:

Subtract 6 from the contents of the A register (modulo 2^{15}).

AAA -6

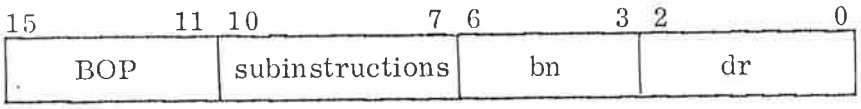
Example 5:

The contents of the A register will be $177\,640_8$ after the execution of this instruction (sign extension).

SAA -140_8

In an add argument instruction the carry and overflow indicators are set according to the same rules as apply for the ADD instruction. see Section 3.1.4.

3.2.6 Bit Operation Instructions



BOP Bit Operation

The BOP instruction specifies operations on single bits in one of the seven general registers, or the status register.

The specific bit to be manipulated is specified by the <dr> and <bn> fields in the instruction. The <dr> field specifies the particular register and the <bn> field the particular bit in that register.

The register dr is specified by means of the same mnemonics as used for destination registers in the ROP and SKP instructions, see Section 3.2.3, except if dr = 0 the status is specified.

The BOP instruction may use a one bit accumulator register, K, to hold temporary results.

16 different subinstructions are available in the BOP instruction.

In the following description "bit" means the bit specified by destination register dr and bit number bn. Note that bn is specified by octal numbers and the "bits" are numbered 0, 10, 20, 30, ...170.

The six control indicators of the status register which may be operated upon by means of the BOP instruction should be specified with the following mnemonics:

(Subscript $_0$ signifies the complement of the specified bit.)

SSK	One bit accumulator indicator.
SSZ	Error indicator
SSQ	Dynamic overflow indicator.
SSO	Static overflow indicator.
SSC	Carry indicator.
SSM	Multi shift link indicator.

3.2.6.1 Bit Skip Instructions

Four subinstructions are available to test the setting of the specified bit.

BSKP ZRO <bn><dr>	Skip next instruction if bit = 0	Time: 2.3-2.8 μs
BSKP ONE <bn><dr>	Skip next instruction if bit = 1	Time: 2.3-2.8 μs
BSKP BCM <bn><dr>	Skip next instruction if bit_0 = K	Time: 3.3-3.8 μs
BSKP BAC <bn><dr>	Skip next instruction if bit = K	Time: 3.3-3.8 μs

3.2.6.2 Bit Setting Instructions

Four subinstructions are available to set the specified bit.

BSET ZRO <bn><dr>	bit $\leftarrow$ 0	Time: 1.4 μs
BSET ONE <bn><dr>	bit $\leftarrow$ 1	Time: 1.4 μs
BSET BCM <bn><dr>	bit $\leftarrow$ bit_0, complement bit	Time: 1.4 μs
BSET BAC <bn><dr>	bit $\leftarrow$ K	Time: 3.0 μs

3.2.6.3 One Bit Accumulator Instructions

Eight subinstructions are available to specify operations between the specified bit and the one bit accumulator, K.

BSTA	<bn><dr>	bit ← K, K ← 0	Store and clear Time: 3.3 μs
BSTC	<bn><dr>	bit ← K_0, K ← 1	Store complement and set Time: 3.3 μs
BLDA	<bn><dr>	K ← bit	Load Time: 2.9 μs
BLDC	<bn><dr>	K ← bit_0	Load complement Time: 2.9 μs
BANC	<bn><dr>	K ← bit_0 ∧ K	Logical AND complement Time: 2.9 μs
BORC	<bn><dr>	K ← bit_0 V K	Logical OR complement Time: 2.9 μs
BAND	<bn><dr>	K ← bit ∧ K	Logical AND Time: 2.9 μs
BORA	<bn><dr>	K ← bit V K	Logical OR Time: 2.9 μs

Some examples of correctly specified bit operation instructions.

Example 1:

Skip next instruction if the carry indicator is set.

```
BSKP ONE SSC
```

Example 2:

Reset the static overflow indicator.

```
BSET ZRO SSO
```

Example 3:

Complement the sign bit in the T register (complement a floating point number).

```
BSET BCM 170₈ DT
```

Example 4:

Set bit 6 in the X register to one.

```
BSET ONE 60₈ DX
```

Example 5:

Copy A register bit 14 into X register bit 13.

```
BLDA 160₈ DA          % K ←A bit 14
BSET BAC 150₈ DX      % X bit 13←K, K←0
```

3.2.7 Accumulator Transfer Instructions

The internal registers in NORD-12 which cannot be reached by the register instructions are controlled by the following four instructions:

TRA	Transfer to A register, Section 3.2.7.1
TRR	Transfer from A register, Section 3.2.7.2
MCL	Masked clear, Section 3.2.7.2.
MST	Masked set, Section 3.2.7.2.

The registers which are read and/or controlled by these instructions are:

Name	$Code_8$	Description
STS	1	Status register. Bits 2-7 may be read or set, while bits 8-11 (PL) bit 14 (PONI) and bit 15 (IONI) may only be read.
OPR	2	Operator's panel switch register, see Section 7.2
LMP	2	Operator's panel lamp register, see Section 7.3.
PVL	4	Previous level. The contents of the register are: IRR < previous level * 10_8 > DP, see Section 5.4
PID	6	Priority interrupt detect, see Section 5.1.
PIE	7	Priority interrupt enable, see Section 5.1
ALD	12	Automatic load descriptor, see Section 8.2.4.

Table 3.3: Survey of Registers controlled by accumulator Transfer Instructions. Codes not shown should not be used. See also Table 3.4.

There are also two instructions for accessing single registers outside current program level, see Section 3.2.3.3.

3.2.7.1 Transfer to A register

TRA Transfer to A register Code: 150 000

Format: TRA <register name>

The register which may be transferred to the A register with the TRA instruction is shown in Table 3.4. The contents of the register specified by the register name are copied into the A register. The operator's panel and the paging systems are optional, and without these options a TRA instruction, which tries to read a non-implemented register, will cause the A register to be cleared.

Time: 4.3 μs

3.2.7.2 Transfer from A Register

The transfer from the A register may be either an ordinary transfer of all 16 bits or a selective setting of zeroes and ones.

The three subinstructions are:

TRR — Transfer to register — Code: 150 100

Format: TRR ⟨register name⟩

The contents of the A register are copied into the register specified by ⟨register name⟩. The registers which TRR may operate on are shown in Table 3.4.

Time: 4.8 μs

MCL — Masked clear — Code: 150 200

Format: MCL ⟨register name⟩

For each bit which is a one in the A register the corresponding bit specified by ⟨register name⟩ will be set to zero. The register which MCL may operate on is shown in Table 3.4.

Time: 5.8 μs

MST — Masked set — Code: 150 300

Format: MST ⟨register name⟩

For each bit which is a one in the A register the corresponding bit in the register specified by ⟨register name⟩ will be set to one. The registers which MST may operate on are shown in Table 3.4.

Time: 5.8 μs

Register Name	$Code_8$	TRA	TRR	MCL	MST
STS	1	x	x	x	x
OPR	2	x			
LMP	2		x		
PVL	4	x			
PID	6	x	x	x	x
PIE	7	x	x	x	x
ALD	12	x			

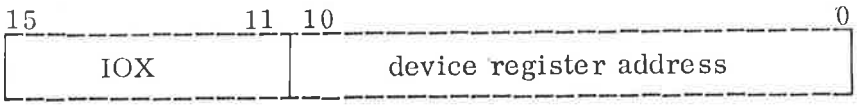
Table 3.4. Accumulator Transfer Instructions

3.3 Input/Output Control Instructions

IOX Input/Output execute Code: 164 000

Format: IOX device register address

Time: 2.7 μs



All program controlled transfers between the CPU A register and the external devices are controlled by using the IOX instruction. The IOX instruction is loaded into the instruction register, IR, of the CPU. The CPU in its turn generates the Input/Output timing and enables the selection of the appropriate device, which is specified by its device register address, <device register address> , bits 0-10. These 11 bits define an upper limit of 2048 device register addresses to the number of registers that may be addressed. Some registers may require two device register addresses, one for reading and one for writing. Different devices will, however, require different number of device register addresses, Thus, the maximum number of physical devices that may be connected will depend on the specific configuration of devices.

Simple devices will usually require at least three different instructions (addresses), write control register, read status register, and read or write data buffer register. More complex devices like magnetic tape units may need up to eight instructions. Instructions for the same device are assigned successive device register addresses.

3.3.1 Recommended Device Addresses

Device addresses used for A/S Norsk Data-Elektronikk produced equipment on a standard Input/Output bus follow a preset assignment. The standard address formats for the different groups of devices are shown in Figure 3.5.

Device Group	Standard Group Address	Address Bits 10	9	8	7	6	5	4	3	2	1	0
Directly controlled registers	e 000	0	e	0	0	0	register no.					
Syncronous, Modems	e 100	0	e	0	0	1	modem no.			channel	control	transfer
Asyncronous Modems	e 200	0	e	0	1	0	display no.			channel	control	transfer
Teletypes	e 300	0	e	0	1	1	Tele-type no.			channel	control	transfer
Paper tape devices line printers, etc.	e 400	0	e	1	0	0	device type			channel	control	transfer
Mass storage device	e 500	0	e	1	0	1	mass storage no.			reg. no.		transfer
Plotters, intercore other DMA devices	e 600	0	e	1	1	0	device type+ register no.				control	transfer
Miscellaneous	e 700	0	e	1	1	1						transfer
		2000	1000	400	200	100	40	20	10	4	2	1

Table 3.5 Standard Device Addresses for ND produced Equipment.

The e bit is used for extension of the groups, extension: e=1
The e is normally equal zero.

channel { 0 input channel, i.e. input devices
1 output channel, i.e. output devices

control { 0 data register
1 status or control register

transfer { 0 input transfer
1 output transfer

Bit 10 is used to distinguish between ND produced and customer produced equipment, bit 10 equals zero: ND produced equipment.

In the following some examples are given of device addresses. For further programming specifications a NORD-10 Input/Output manual should be consulted.

Example 1:

Teletype Addresses:

The codes below are relevant for the first Teletype, Teletype number 0. The codes for the first eight Teletypes are found by adding 10_8 * N for the codes given, where N is the specific Teletype number.

Input Channel,

IOX 300	Read Data Register
IOX 302	Read Status Register
IOX 303	Write Control Register

Output Channel,

IOX 305	Write Data Register
IOX 306	Read Status Register
IOX 307	Write Control Register

Example 2:

Paper Tape Reader Addresses

IOX 400	Read Data Register
IOX 402	Read Status Register
IOX 403	Write Control Register

Example 3:

Paper Tape Punch Addresses

IOX 411	Write Data Register
IOX 412	Read Status Register
IOX 413	Write Control Register

Example 4:

Line Printer Addresses

IOX 431	Write Data Register
IOX 432	Read Status Register
IOX 433	Write Control Register.

Example 5:

The standard device addresses for the mass storage devices are as follows:

500	Disk I with four units
510	Disk II with four units
520	Magnetic tape I with four units
530	Magnetic tape II with four units
540	Drum I
550	Drum II
560	Drum III
570	Drum IV

and the standard register addresses within each device.

0	Core Address Register
2	Sector Block Address Register
4	Status Control Register
6	Word Count Register

Example 6:

Drum Addresses

The codes below are relevant for drum I

IOX 540	Read Core Address
IOX 541	Load Core Address
IOX 542	Read Sector Counter
IOX 543	Load Block Address
IOX 544	Read Status Register
IOX 545	Load Control Register
IOX 547	Load Word Count Register

Example 7:

Real Time Clock Addresses

IOX 10	Read Data
IOX 11	Write Data
IOX 12	Read Status
IOX 13	Write Control

3.3.2 Format of Status and Control Word

The format of status and control word may be assigned by the designer of each device controller. The following standard is used by ND for its own device control cards (when applicable) and is recommended for customer use.

Status Word

Bit 0	Ready for transfer, interrupt enabled
1	Error interrupt enabled
2	Device active
3	Device ready for transfer
4	Inclusive OR of errors
5	Error indicator
6	Error indicator
7	Error indicator
8	Error indicator
9	Selected unit
10	Selected unit
11	Operational mode of device
12	Operational mode of device
13	Operational mode of device
14	Operational mode of device
15	Operational mode of device

Control Word

Bit 0	Enable interrupt on device ready for transfer
1	Enable interrupt on errors
2	Activate device
3	Test mode
4	Device clear
5	Address bit 16
6	Address bit 17
7	Not assigned
8	Not assigned
9	Unit
10	Unit
11	Device operation
12	Device operation
13	Device operation
14	Device operation
15	Device operation

3.4 System Control Instructions

The following five instructions are denoted as the system control instructions:

ION	Interrupt system on
IOF	Interrupt system off
IDENT	Identify Input/Output interrupt
MON	Monitor call
WAIT	Wait or give up priority

3.4.1 Interrupt Control Instructions*

The NORD-12 computer has a priority interrupt system with 16 program levels. Each program level has its own set of registers and status indicators. The priority is increasing: program level 15 has the highest priority, program level 0 the lowest.

The arrangement of the 16 program levels are as follows:

15	Reserved extremely fast user interrupts.
14	Reserved monitor calls
13-10	Vectored interrupts, maximum 2048 vectored interrupts.
9-8	System programming.
7-0	User programming levels.

All 16 program levels can be activated by program control. In addition, program level 15, 13, 12, 11 and 10 may also be activated from external devices.

The program level to run is controlled from the two 16-bit registers:

PIE	Priority interrupt enable
PID	Priority interrupt detect

Each bit in the two registers is associated with the corresponding program level. The PIE register is controlled by program only.

---

* A complete description of the NORD-12 Interrupt System is found in Chapter 5.

The PID register is controlled both by program and hardware interrupts. At any time, the highest program level which has its corresponding bits set in both PIE and PID is running, i.e. the contents of the PL register.

The PIE and PID are controlled by the TRA, TRR, MST and MCL instructions, see Section 3.2.7.

When power is turned on, the power-up sequence will reset PIE. and the register set on program level zero will be used.

Two instructions are used to control the on-off function of the interrupt system.

ION — Interrupt system on — Code: 150 402

Format: ION

The ION instruction turns on the interrupt system. At the time the ION is executed, the computer will resume operation at the program level with highest priority. If a condition for change of program levels exists, the ION instruction will be the last instruction executed at the old program level, and the P register at the old program level will point to the instruction after ION. The interrupt indicator on the operator's panel is lighted by the ION.

Time: 2.3 μs

IOF — Interrupt system off — Code: 150 401

Format: IOF

The IOF instruction turns off the interrupt system, i.e. the machanisms for changing of program levels are disabled. The computer will continue operation at the program level at which the IOF instruction was executed, i.e. the PL register will remain unchanged. The interrupt indicator on the operator's panel is reset by the IOF instructions.

Time: 2.3 μs

Initialization of the interrupt system is treated in Section 5.2.

In addition the following register is available to ease the interrupt programming:

PVL	Previous level causing internal hardware status interrupt.

Its use is described in Section 5.4. In NORD-12 there are possibilities for 2048 vectored Input/Output interrupts where each physical Input/Output unit will have its own unique identification code and priority. The IDENT instruction is used to distinguish between vectored interrupts.

IDENT — Identify vectored interrupt — Code: 143 600

Format: IDENT <program level number>

When a vectored interrupt occurs, the IDENT instruction is used to identify and serviced the actual Input/Output device causing the interrupt. Actually, there are four IDENT instructions, one to identify and service Input/Output interrupts on each of the four levels 10, 11, 12 and 13. The particular level to serve is specified by the program level number.

The four instructions are:

IDENT	PL10	Identify Input/Output interrupt on level 10	Code: 143 604
IDENT	PL11	Identify Input/Output interrupt on level 11	Code: 143 611
IDENT	PL12	Identify Input/Output interrupt on level 12	Code: 143 622
IDENT	PL13	Identify Input/Output interrupt on level 13.	Code: 143 643

The identification code of the Input/Output device is returned to bits 0-8 on the A register with bits 9-15 all zeroes.

If the IDENT instruction is executed, but there is no device to serve, the A register is unchanged.

If several devices on the same program level have simultaneous interrupts, the priority is determined by which Input/Output Slot the device is plugged into, and the interrupt line to the corresponding PID bit will remain active until all devices have been serviced. When a device responds to an IDENT, it turns off its interrupt signal

Time: 3.3 μs

For NORD-12 the identification codes are standardized for Input/Output devices delivered from ND.

Table 3.6 on Page 3-63 shows the IOX addresses and IDENT codes used in standard software.

3.4.2 Monitor Call Instruction

MON Monitor Call Code: 153 000

Format: MON <number>

The instruction is used for monitor calls, and causes an internal interrupt to program level 14. The parameter, <number>, following MON, must be specified between -200_8 and 177_8. This provides for 256 different monitor calls. This parameter, sign extended, is also loaded into the T register on program level 14.

Time: 3.3 μs

3.4.3 Wait or give up Priority

WAIT Wait Code: 151 000

Format: WAIT <number$_8$>

The WAIT instruction will cause the computer to stop if the interrupt system is not on. The program counter will point to the instructions after the WAIT.

In this programmed wait the STOP button on the operator's panel is lighted. To start the program in the instruction after the WAIT, push the button CONTINUE or type! on the console TTY.

If the interrupt system is on, WAIT will cause an exit from the program level now operating (the corresponding bit in PID is reset), and the program level with the highest priority will be entered, which normally will then have a lower priority than the program level which executes the WAIT instruction. Therefore, the WAIT instruction means "Give up priority".

If there are no interrupt requests on any program level when the WAIT instruction is executed, program level zero is entered. A WAIT instruction on program level zero is ignored.

Note that it is legal to specify WAIT followed by a number less than 377_8. This may be useful to detect in which location the program stopped. The WAIT instruction is displayed at the operator's panel, IR register.

Time: 5.0 μs

3.5 Customer Specified Instructions

The remaining free codes on the skip instruction may be used to augment the NORD-12 instruction set. The codes to be used for customer specified instructions are as follows:

1401XX	1403XX	1405XX	1407XX
1411XX	1413XX	1415XX	1417XX
1421XX	1423XX	1425XX	1427XX
1431XX	1433XX	1435XX	1437XX

These 16 instructions have provisions for 16 new entry points in a Read-only-memory outside the address space in the 1K standard Read-only-memory.

If these instructions are not implemented, they will cause the CPU to enter STOP mode.

All the 16 customer specified instruction have the source (sr) and destination (dr) fields available for further specifications.

These fields may either be used to let the customer specified instruction operate on the general registers, or used to augment the number of customer specified instructions.

If the sr and dr fields are used to increase the number of customer specified instructions, up to 1024 instructions may be added.

A/S Norsk Data-Elektronikk should be contacted for further information on specifications and programming rules for the NORD-12 microprocessor.

DEVICE		STANDARD		EXTENTION	
	Level	IDENT	IOX	IDENT	IOX
Tape punch	10	2, 22	410-413, 414-417	32	+1000
Tape reader	12	2, 22	400-403, 404-407	32	+1000
Line printer	10	3, 23	430-433, 434-437	33	+1000
Card reader	12	3, 23	420-423, 424-427	33	+1000
Sync. modem	10, 12	4, 14	100-107, 110-117	20, 24	120-127, 130-137
N-1 I/O channel	11	10	-	-	-
Cassette tape	12	11, 21	700-707, 710-717		
Digital registers	10, 12	17	770-777	27	+1000
Teletype	10, 12	1, 5, 6, 7, 44-47	300-377	50-57	+1000
Async. modem	10, 12	60-67	200-277	70-77	+1000
Analog/digital converter		-	720-727		+1000
Digital/analog converter		-	730-737		+1000
Versatec plotter	11	4	600-607	14	+1000
Disk	11	1, 5	500-507, 510-517	15, 25	+1000
Drum	11	2, 6	540-547, 550-557	12, 16	560-567, 570-577
Mag. tape	11	3, 7	520-527, 530-537	13	+1000
Real time clock	13	1	10-13	2	14-17

Table 3.6: Standard IOX addresses and IDENT codes

4 THE INPUT/OUTPUT SYSTEM

4.1 Input/Output Hardware

4.1.1 General Description

In NORD-12 all Input/Output device interface cards are made to a common standard. The CPU module contains a pre-wired bus with a number of identical interface slots permitting any mixture of devices without changing the backwiring and plug panel. Device plugs are also made to a common standard.

This system permits the use of printed backplane wiring for all wiring within one module. Cable connectors are plugged directly into the backplane.

The direct memory access channel, DMA, has a transfer capacity of 1.2M word/second. There may be a single very high-speed device requiring this speed, or several different slower devices sharing the channel. In the latter case, there will be no channel time overhead in switching between devices. Thus, several devices using the channel simultaneously, will be given a total throughput equivalent to the maximum speed of the channel.

An optional controller which permits control of the devices from two different CPU's, multi-machine environment, is also available.

Both modules in Figure 4.1 are standard 19" modules, each with 32 card positions.

The CPU module contains:

- The CPU, consisting of Registers, Arithmetic, Microprogrammed Control Section, Interrupt System, and Operator's Panel Driver. (11 card positions.)

- 13 card positions for Input/Output device buffers. Each program controlled device, such as Teletypes, Paper Tape Reader, Paper Tape Punch, Card Reader, Line Printer, etc. requires one card position in this bus. The bus loading may not exceed that of 12 normal Input/Output device buffers. When more buffers are needed (or the space is used for memory extension as described below). External Bus Driver is used. This Bus Driver occupies 3 of the 13 card positions, but it only represents a load of one device buffer.

- 8 card positions for 4K by 16 bits semiconductor memory cards, giving a standard capacity of 32K words (64K bytes). Memory may be expanded to maximum 64K words by a small amount of optional wiring in 8 of the 13 card positions for Input/Output device buffers.

The External Bus Driver is used to drive a differential Main I/O Bus with a maximum cable length of 50 meters. Several Bus Controller Modules may be connected to this bus, to drive Local Input/Output buses where Input/Output device buffers may be plugged in.

The Bus Controller Modules drives 8 or 16 card positions on a Local I/O Bus, and contains 8 or 16 card positions for one or two complex device controllers (drum, disc, magnetic tape, etc.).

The position of the device interface in the modules determines the interrupt priority of the device. If several devices within one modules are connected to the same program level, the device closes to the controller has the highest priority within that level. Also if two devices in the same module compete for a direct memory access, the device closest to the controller has the highest priority and will win the first access.

4.1.3 Vectored Interrupt Identification

The NORD-12 has a multiprogram system with 16 program levels. Each program level has a complete set of registers. Out of these 16 program levels five different program levels may be triggered by hardware interrupts. These levels are: 15, 13, 12, 11, and 10.

Several different interrupt sources may be connected to the program levels 10, 11, 12, and 13, while program level 15 is reserved for extremely fast user Input/Output.

To identify which device is interrupting a "who are you" type of instruction is used. This returns a 9-bit identification from the interrupting device to the A register. The instruction has the format:

IDENT program level number

and is described in Section 3.4.1.

For program level 15, which is exclusively reserved user Input/Output, there is no identification system, and identification is obtained by reading a status word.

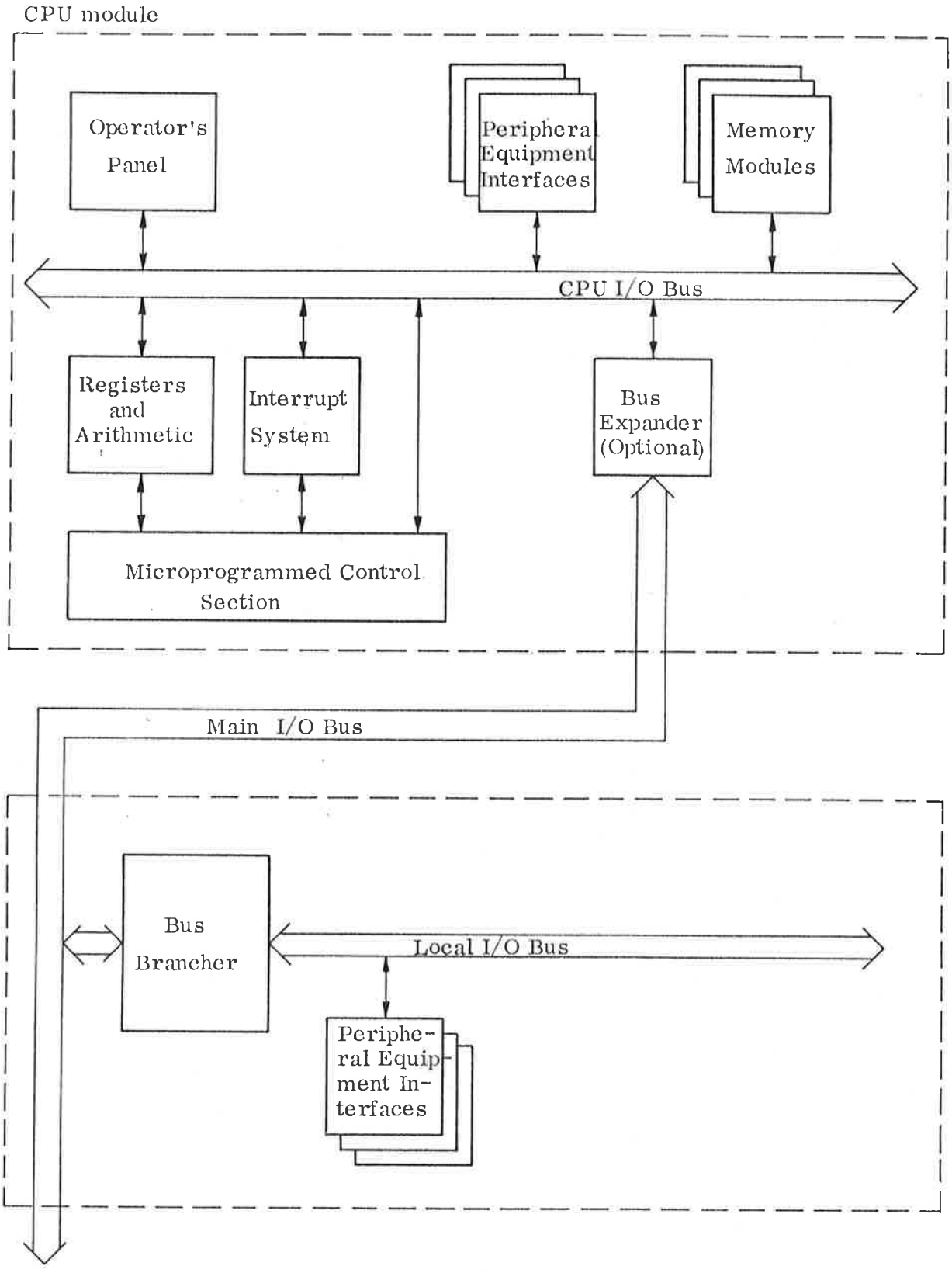


Figure 4.1: NORD-12 Bus System.

4.2 Input/Output Programming

The recommended way to perform Input/Output in a software system is to use standard input/output subroutines. Input/Output subroutines and drivers for all standard devices are available from A/S Norsk Data Elektronikk.

Data transfer between the A register and an external device will be controlled by IOX instructions containing an 11-bit "Device register address" - DRA.

For direct memory access devices like disks, drums, magnetic tape, etc., the IOX instruction is used to write or read control information to or from the controller of the specific device. Complex devices like those mentioned may need several DRA's. A device like a punch, reader, Teletype input, Teletype output will require at least three DRA instructions.

The three instructions are:

IOX	load device control register
IOX	read device status register
IOX	read device data buffer register or write device data buffer register

ND's standard for use of the bits in device status and device control register is shown in Section 3.3.1.

The Input/Output system makes it possible for the programmer to control external devices in a tight and flexible manner.

Detained information about DRA, status, control, etc., for different devices are found in the "Programming Specification" for each device type.

4.2.1 Programming Examples

The following example shows a simple subroutine which reads a byte from the tape reader:

```
INPUT,   SAA  4
         IOX DEVC + RDR        %SET CONTROL (ACTIVATE DEVICE)
         IOX DEVS + RDR        %READ DEVICE STATUS
         BSKP ONE 30 DA        %DEVICE READY?
         JMP* - 2              %NO
         IOX RDEVB + RDR       %READ DEVICE BUFFER
         EXIT

RDR      = 400                 % 1. DRA FOR TAPE READER
DEVC     =   3
DEVS     =   2
RDEVB    =   0
```

Programming examples for complex devices may be found in the appropriate programming manuals.

4.2.2 Input/Output Interrupt Programming

Input/Output via waiting loops as shown in the previous section is very ineffective due to the fact that most of the computer time will be spent in the Input/Output loops. This may be avoided by utilizing the interrupt system. An interrupt will occur every time the device is ready for transfer.

The necessary software will normally be:

- Input/Output subroutines which will put a byte into a device buffer. (Software buffers.)
- Interrupt identification sequences on the program levels which the devices are connected to. (Using IDENT instructions.)
- Interrupt drivers fro each device type. The identification sequence will branch to the driver of the interrupting device. The driver will fetch a byte from the device buffer and output it to the device (output device) or read a byte from the device and put it into the device buffer (input device). The user of such a system, however, will only "see" the Input/Output subroutines and does not have to bother about details.

4.2.3 Design of an Input/Output Handler Routine

This is an example of a simple Input/Output driver system:

```
%PROGRAM ON LEVEL 12
RET,            SAA 0
INT12,          WAIT
                IDENT  PL12     %GET INTERRUPT IDENTIFICATION
                RADD SA DP      %ADD NUMBER TO
                                %P REGISTER
                JMP ERROR       %IDENT 0 MEANS I/O
                                %SYSTEM ERROR
                JMP DRIVER1     %GO TO 1.  DRIVER
                JMP DRIVER2
                -
                JMP DRIVERN
                JMP RET
%DRIVER FOR AN INPUT DEVICE

DRIVER1,        IOX STATUS      %READ DEVICE STATUS
                BSKP  ZRO 40 DA
                JMP ERRORD      %DEVICE  ERROR
                IOX RBUF        %READ DEVICE BUFFER
                -
                -
                -
                PUT BYTE INTO BUFFER ETC.
                ENABLE AND ACTIVATE DEVICE FOR NEXT
                TRANSFER

                JMP RET
```

5 THE INTERRUPT SYSTEM

The NORD-12 interrupt system is designed to simplify programming, and to allow multiprogramming at extremely high efficiency.

This is achieved by use of a complete set of registers and status indicators for each program level.

There are 16 program levels in NORD-12 and therefore 16 sets of registers and status indicators. Each set consists of: A, D, T, L, X and B registers, program counter, and each of the status indicators O, Q, Z, C, M, and K.

The context switching from one program level to another is completely automatic, and requires only 2.0 μs; the saving and unsaving of all registers and status indicators are included.

In addition to the 16 program levels, there is a maximum of 512 vectored interrupts connected to each of the program levels 13, 12, 11 and 10.

For the vectored interrupts there is an automatic priority identification mechanism, thus no polling of interrupts is necessary.

The arrangement of the 16 program levels are as follows:

15	Reserved extremely fast user interrupts.
14	Monitor call
13-10	Vectored interrupts, up to 2048 vectored interrupts.
9-8	System programming
7-0	Programming levels.

The priority is increasing, program level 15 has the highest priority, program level 0 the lowest.

The structure of a large programming system may be greatly simplified by the use of these program levels where independent tasks may be organized at different program levels with all priority decisions determined by hardware, and with almost no overhead because of the rapid context switching.

All 16 program levels can be activated by program control. In addition program levels 15, 13, 12, 11 and 10 may also be activated from external devices.

5.1 Control of Program Levels

The program level to run is controlled from the two 16-bit registers:

PIE	Priority interrupt enable
PID	Priority interrupt detect

Each bit in the two registers is associated with the corresponding program level. The PIE register is controlled by program only. The PID register is controlled both by program and hardware interrupts. At any time, the highest program level which has its corresponding bits set in both PIE and PID is running.

The actual hardware mechanisms for this are as follows:

The number of the current program level is called PL ($0 \leq PL \leq 15$), and this 4-bit PL register controls which register set (context block) to use.

All the time the PL number is compared to a 4-bit register PIK. At any time, PIK contains the number of the highest program level which has its corresponding bits set in both PIE and PID. Whenever PIK becomes different from PL, an automatic change of context block will take place through a short micro-program sequence. This sequence will do the following:

1) Read PL and store it in the PVL register, previous program level.

2) Read PIK and store it into PL.

3) Resume operation with a new register set determined by PL.

This complete sequence requires only 2.0 μs, from the completion of the instruction currently working when the interrupt took place, and until the first instruction is started on the new level with its new set of register and status.

The programming control of the interrupt system is as follows:

PID and PIE may be read to the A register with the instructions

TRA PID and TRA PIE

Three instructions are available for the setting of these registers

TRR PID and TRR PIE

The TRR instruction will copy the A register into the specified register.

MST PID and MST PIE

The MST, masked set, instruction will set the bits in the specified register to one where the corresponding bits in the A register are ones. (The A register is used as a mask for selection of which bit to set.)

MCL PID and MCL PIE

The MCL, masked clear, instruction will reset to zero the bits in the specified register where the corresponding bits in the A register are ones.

In addition to TRA, TRR, MCL and MST the PID register is also controlled in the following ways:

External interrupts may set PID bits 15, 13, 12, 11, 10.

The resetting of PID bits is also controlled by the WAIT instruction, which will reset PID on current program level. (The WAIT instruction is also called "Give up Priority".)

For example a program on program level 14, which issues a WAIT instruction, will cause PID bit 14 to be zero, which again will cause a new program level to be entered because PIK became different from PL (= 14).

The interrupt system is also controlled by the two instructions,

ION	Turn on interrupt system
IOF	Turn off interrupt system.

When power is turned on, the power-up sequence will reset PIE and PL and the register-set on program level zero will be used.

The ION instruction will resume operation at the highest program level at the time ION is executed, if a condition for change of program levels exists the ION instruction will be the last instruction executed at the old program level, and the P register at the old program level will point to the instruction after ION.

The IOF instruction will turn off the mechanisms for changing of program level, and PL will remain unchanged.

IOF and ION may also be used to disable the interrupt system for short periods, for example in order to prevent software timing hazards.

5.1.1 Program Level Activation

All program levels may be activated by program, by setting the appropriate bits in PIE and PID.

Example:

If program level 9 is already enabled, bit 9 in PIE is set, then the program level is activated from a lower program level by setting bit 9 in PID.

```
        SAA  0
        BSET ONE 110  DA        %SET BIT 9 TO ONE
        MST PID                 %SET PID BIT 9
NEXT,   .........
```

The MST PID will be the last instruction executed, and the P register at the lower program level will point to the NEXT instruction.

Note that it is not possible to program-activate a program level which already is activated (i.e., has its PID bit set to one), if it is tried, the program level will only be entered once.

5.2 Initialization of Interrupt System

The initialization of the NORD-12 interrupt system is simple. After power-up, PIE and PL will be zero and register block zero is in use. The initialization sequence must include the following:

1) Enabling of the desired program levels by setting PIE.

2) The program counter on all program levels used have to be initialized. The program counter must point to the entry point of that particular program level.

   The remaining initialization of registers may be performed either at program level itself at the time of the first entry, or together with the initialization of the program counter.

3) The last instruction in the initialization sequence is ION.

5.3 Interrupt Program Organization

A program at a program level will typically be organized as a loop, which is executed once each time the program level is activated:

```
ENTRX,  -                       %FIRST ENTRY POINT
        -
        -
        -
        WAIT                    %GIVE UP PRIORITY
        JMP ENTRX
```

If response time is important the following organization is better:

```
        WAIT                    %GIVE UP PRIORITY
ENTRX,  -                       %FIRST ENTRY POINT
        -
        -
        -
        JMP ENTRX - 1
```

Note that a WAIT instruction on program level zero will reset PID bit zero, but since there are no program levels with lower priority, the program on program level zero will be re-entered at the instruction following the WAIT.

5.4 Internal Interrupts

Program level 14 is reserved for internal interrupts. On NORD-12 these internal interrupts are caused by the MON, Monitor Call instruction, see also Section 3.4.2.

To speed internal interrupt processing an instruction for reading the previous level is provided. This is done with the instruction

```
TRA PVL
```

which reads the PVL register, previous program level (level causing internal interrupt) into bits 3-6 in the A register, with remaining bits in the A register being equal to the code for inter-register read the P register, i.e., the contents of the A register:

IRR <previous level$_8$ * 10_8> DP

This technique gives a very fast access to the P register of the program level causing the internal interrupt.

Example:

```
TRA     PVL         %A: = IRR <level> DP
EXR     SA          %A: = P register on interrupting
                          level
COPY    SA DX
LDA     -1 ,X       % A: = Interrupting instruction
```

Note: PVL is only set when enterin level 14 from a level with lower priority. Care should be taken so that programs on level 14 and level 15 do not cause internal interrupts.

5.4.1 Monitor Call Interrupt

A monitor call has been executed. The level may be found as previously explained. The number of the call is automatically set to the T register on level 14.

Note that this number is 8-bit with sign-extension, (i.e. in the range -200_8 to 177_8). See Section 3.4.2.

5.5 Vectored Interrupts

In NORD-12 there may be up to 2048 vectored interrupts: typically each physical input/output unit will have its own unique interrupt response code and priority.

These vectored interrupts must be connected to the four program levels 13, 12, 11 and 10.

The standard way of connecting is as follows:

Level 13 Real time clock

Level 12 Input devices

Level 11 Mass storage devices

Level 10 Output devices

The vectored interrupts are connected to the corresponding bits in the PID register.

When a vectored interrupt occurs, the IDENT instruction is used to find out which device gave interrupt on this program level, if several devices have simultaneous interrupt. The priority is determined by which Input/Output slot the device is plugged into. For further information, see Section 4.1.3, or the Input/Output Manual.

The IDENT instruction provides a very fast response time, and no polling of devices is required.

Programming example:

```
RETURN,  SAA  0
                 WAIT        % GIVE UP PRIORITY
LEV13,   IDENT   PL13        % IDENTIFY DEVICE ON
                             % LEVEL 13
         RADD    SA  DP      % COMPUTED GO TO
         JMP     ERR13       % CODE 0,  ERROR
         JMP     DRIV1       % CODE 1,
         JMP     DRIV2       % CODE 2
         -
         -
         JMP     DRIVN       % CODE N
```

Note that only three instructions are required from time of the interrupt until the specific Input/Output driver is entered.

The IDENT instruction will turn off the interrupt signal of the device which gave interrupt. If several devices have their interrupt signals on, the interrupt line to the corresponding PID bit will remain active, and as soon as the WAIT instruction has reset one bit in PID, this bit will be set again, and the WAIT instruction will have no effect.

6 CONTROL PANEL

The NORD-12 is controlled from the same micro-program which controls the NORD-10. Therefore the same control panel as developed for the NORD-10 may also be used for the NORD-12. This panel is described in Chapter 7. In addition the complete micro-programmed operator communication as developed for the NORD-10 is also available for the NORD-12. This communication which requires a Teletype or visual display unit is described in Chapter 8.

6.1 NORD-12 Control Panel

To reduce costs a special panel for the NORD-12 has been developed. This panel consists of six pushbuttons. The function of each of the buttons is given below. The panel buttons may be locked by means of a key.

6.1.1 Power On/Off

The power button may only be operated if the panel is unlocked. With the panel locked, none of the control buttons may be operated.

6.1.2 MASTER CLEAR

Pushing this button will generate a hardware master clear signal. This signal sets the control logic in the CPU and the Input/Output system to a defined state and the micro-programmed operator's communication (MOPC) is started. If the CPU is running when "MASTER CLEAR" is pushed, the program cannot be restarted by pushing the CONTINUE button, because the contents of the P and A registers are lost. The PIE register is reset by the master clear signal.

Light in the MASTER CLEAR button indicates an error input to the CPU from the operator's communication program or one of the load programs. The light is reset when the MASTER CLEAR button is pushed.

6.1.3 RESTART

This button generates a restart signal. When this signal is detected by the micro-program in stop mode, the CPU will start in address 20. The RESTART button has no effect when the CPU is running. IF the CPU is running, the STOP button must be pushed before the RESTART. To be sure that the program has been started on level zero, the MASTER CLEAR button should also be pushed.

6.1.4 LOAD

The LOAD button starts automatic program load from a device. The device may be an Input/Output device or a mass storage device, depending on the setting of a switch register (ALD) on the Panel Driver Card. The use of this register is explained in Section 8.2.4.

When a load program is active, the LOAD button lights.

6.1.5 CONTINUE

When this button is pressed, the machine starts running from the address specified by the P register. The level is given by the contents of the PIE and PID registers. If the MASTER CLEAR is first pressed, PIE is cleared and the program is started on level zero.

6.1.6 STOP

Pushing this button stops the machine i.e., the micro-program running in stop mode is started. The stop mode is indicated by light in the STOP button.

7 NORD-10 OPERATOR'S PANEL

7.1 Panel Elements

The operator's panel for the NORD-10 computer has the following elements:

1. An 18-bit switch register
2. An 18-bit light diode register
3. 16 selector pushbuttons and 16 associated light emitting diodes.
4. 8 mode indicators
5. A two-digit display and two pushbuttons
6. 10 control buttons
7. Power on/off button

7.2 18-bit Switch Register

This register is used to present 18-bit data to the CPU. Normally only 16 of these are used, the switches may be read from program with the "TRA OPR" instruction. In installations with big memory, more than 64K, 18 switches and lamps may be needed to represent the possible 18-bit addresses.

7.3 18-bit Light emitting Diode Register

This is used to display 16-bit data or 18-bit addresses from the CPU. Register contents, addresses and contents of memory locations may be displayed in this register. The register 16-bits can be set with the "TRR LMP" instruction (the user register (see below) must be selected).

7.4 16 Selector Pushbuttons and 16 associated Light emitting Diodes.

These pushbuttons are used to select one of 16 possible registers to be displayed in the data display register. When one button is pushed, (a register selected) this is indicated with light in the associated diode above the button.

The possible register selections are:

ACTIVE LEVELS : When this button is pushed, the data display (described above) will show the active program levels. 16 diodes (0-15) are used, one for each of the 16 levels. In this mode the lamps are provided with after-glow so that it is possible to observe a single instruction on a program level.

DMA
ADR : If this button is pushed, the data display will show the active DMA (direct memory access) address. (See also Section 7.6.4.)

ADR : This register shows the actual memory address being referenced, excluding DMA references and instruction (program) addresses.

P
ADR : This is the memory address each time an instruction is read (fetch cycle). Effectively the data display will show the program address.

U : This is the user register set by the "TRR LMP" instruction.

Note: If the U register is set from program by "TRR LMP" and the U is NOT selected, the setting of U will disturb the displaying of the selected register. The degree of disturbance will depend on the frequency of the U updating related to the panel interrupt frequency.

DATA : Displays data going to and from memory and on the I/O-bus.

EXM : This selection has two uses:

CPU in STOP

The data display will show the contents of the memory location whose address is set in the switch register when the SET ADDRESS button was last pushed (see below). When the CPU stops, this address is preset to zero. (The selected address is always zero after pushing the SINGLE INSTR button).
Use of the '/' (see Section 8.1.2) in MOPC will also set the memory address displayed.

CPU runs

The data display will show the contents of the memory location whose address is set in the switch register. The memory location is sampled after each panel interrupt, (about every 20 ms).

IR : This selection will display the CPU instruction register.

STS, P, L, B, X, T, A, D.

: If one of these is selected, the data display will show the contents of that register. The register is sampled at each panel interrupt. There is a complete set of these registers on each of the 16 interrupt levels, so one has to select the appropriate level when one of these registers is examined, see Section 7.5.

7.5 Display Level select

This consists of two pushbuttons, "+" and "-", and a two-digit display. By means of the two buttons, the level may be stepped up or down. The contents of the display show the selected level. If the display is stepped outside the limits 0-15, the 2 digit display will show the active program level and the selected register (STS, P, L, B, X, T, A or D) is taken from the active level.

7.6 Control Buttons

These 10 pushbuttons are used to control the CPU and to modify registers and memory. The function of each of the buttons is given below.

7.6.1 MASTER CLEAR

Pushing this button will generate a hardware master clear signal. This signal sets the control logic in the CPU and the Input/Output system to a defined state and the micro-programmed operator's communication (MOPC) is started. If the CPU is running when "MASTER CLEAR" is pushed, the program cannot be restarted by pushing the CONTINUE button, because the contents of the P and A registers are lost. The PIE register is reset by the master clear signal.

Light in the MASTER CLEAR button indicates an error input to the CPU from the operator's communication program or one of the load programs. The light is reset when the MASTER CLEAR button is pushed.

7.6.2 RESTART

This button generates a restart signal. When this signal is detected by the micro-program in stop mode, the CPU will start in address 20. The RESTART button has no effect when the CPU is running. If the CPU is running, the STOP button must be pushed before the RESTART. To be sure that the program has been started on level zero, the MASTER CLEAR button should also be pushed.

7.6.3 LOAD

The LOAD button starts automatic program load from a device. The device may be an Input/Output device or a mass storage device, depending on the setting of a switch register (ALD) on the Panel Driver Card. The use of this register is explained in Section 8.2.4.

When a load program is active, the LOAD button lights.

7.6.4 DECODE ADDRESS

This button is used in connection with the displaying of addresses (DMA ADR, ADR or P ADR selected). When this button is pushed, the address is not displayed directly. The address space is divided into 4K segments and each bit in the display register represents one segment. Bit 0 is lighted if addresses 0 - 7777_8 are used, etc. Light in the button indicates the state of the address display register.

7.6.5 SET ADDRESS

When the machine is in stop mode and a memory examine is wanted, the address must be set up in the panel switch register and the SET ADDRESS button pushed. The address is now saved and is not changed before the SET ADDRESS button is pushed again with a new content in the switch register. This address is also changed when a memory examine is executed from the console device (character"/" used).

Note that this button is used in stop mode only. When the machine is running, the address in the switch register is used directly.

When the machine enters stop mode, the register used by the set address function is set to zero. This means that after a single instruction the examined address is zero.

7.6.6 DEPOSIT

When an address is selected with the SET ADDRESS button, the contents of this cell may be changed with the DEPOSIT button. The new contents are set up in the switch register and the DEPOSIT button pushed. The display selection must be EXM.

7.6.7 ENTER REGISTER

This button is used to load a register. One of the registers STS, P, L, B, X, T, A or D is selected with the register selection switches. Level is selected with the level selector. The contents of the switch register are now stored in the selected register when the ENTER REGISTER button is pushed.

7.6.8 SINGLE INST.

Pushing the SINGLE INSTRUCTION button causes a program to advance one instruction. The address is taken from the P register and the CPU goes back to stop mode after execution of one instruction. The instruction is executed on the level given by the PIE and PID registers.

7.6.9 CONTINUE

When this button is pressed, the machine starts running from the address specified by the P register. The level is given by the contents of the PIE and PID registers. If the MASTER CLEAR is first pressed, PIE is cleared and the program is started on level zero.

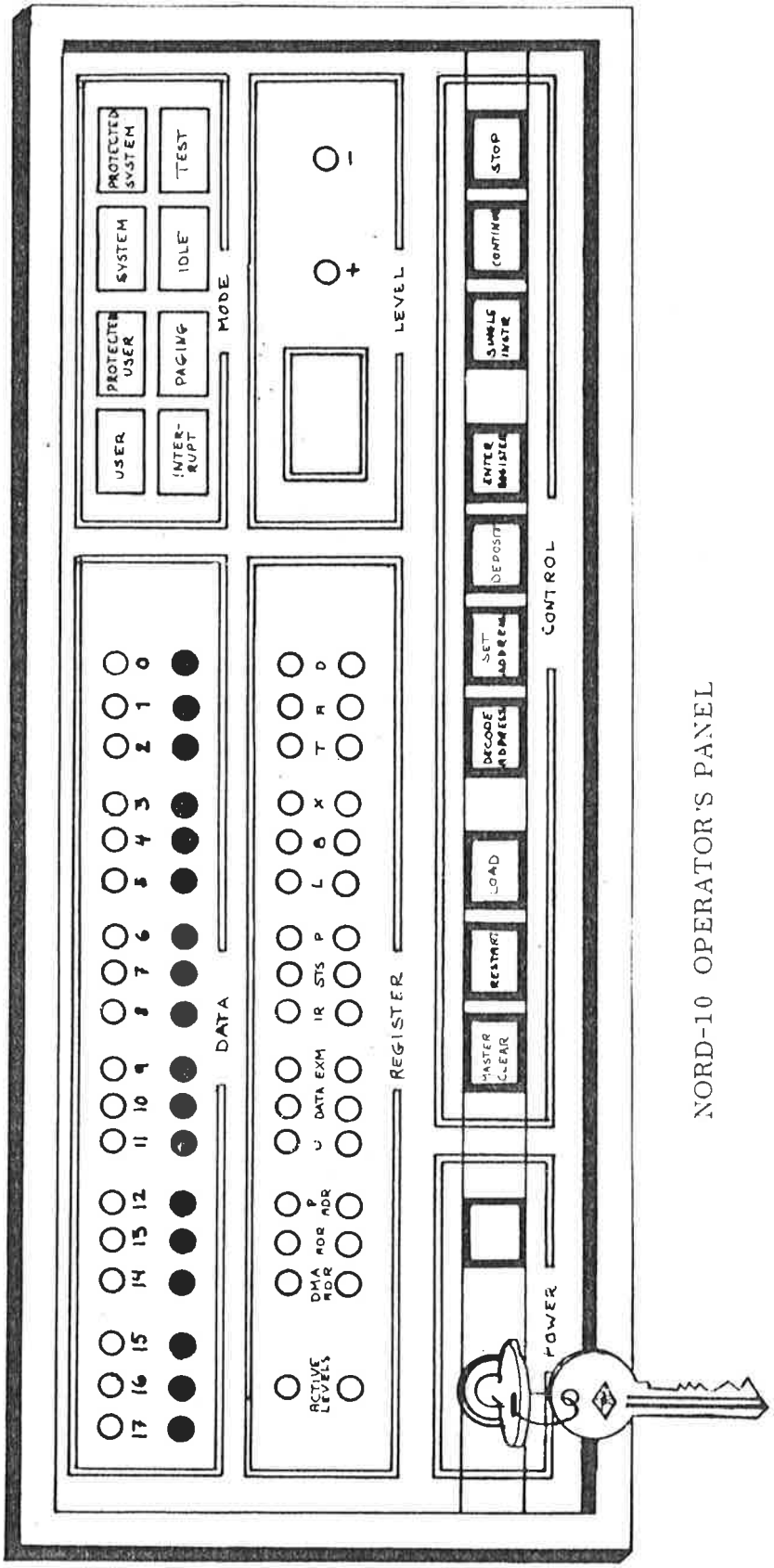
Light in the CONTINUE button indicates that the CPU is running.

7.6.10 STOP

Pushing this button stops the machine i.e., the micro-program running in stop mode is started. The stop mode is indicated by light in the STOP button.

7.7 Mode Indicators

USER	Indicates that the program is running on ring 0 (see Section 6.5).
PROTECTED USER	Indicates that the program is running on ring 1 (see Section 6.5).
SYSTEM	Indicates that the program is running on ring 2 (see Section 6.5).
PROTECTED SYSTEM	Indicates that the program is running on ring 3 (see Section 6.5).
INTERRUPT	Indicates that the interrupt system is turned on i.e., an ION instruction has been executed.
PAGING	Indicates that the paging system is turned on, i.e., PON instruction has been executed.
IDLE	Indicates interrupt system on, and that the CPU is running on level zero.
TEST	Not used.



NORD-10 OPERATOR'S PANEL

8 OPERATOR'S COMMUNICATION

The NORD-10/NORD-12 has a micro program in the read only memory for communication between the operator and the machine. This program is called MOPC (Micro programmed Operator's Communication).

MOPC is always running when the machine is in stop mode, or the state of the machine, when MOPC is running, is defined as the stop mode.

To control a NORD-10 or a NORD-12 two different operator's panels are available. The panel specifically developed for the NORD-10 is described in Chapter 7. The smaller panel developed for the NORD-12 is described in Chapter 6, in the NORD-12 Reference Manual. Both panels are available for NORD-10 and NORD-12, and one of these panels must be selected.

When in STOP mode the NORD-10/NORD-12 micro-program is designed in such a way that either the large operator's panel or the console device (Teletype or Visual display unit) may completely control the NORD-10 or NORD-12. (For special applications the console device may be omitted).

The NORD-10/NORD-12 operator's communication includes bootstraping programs and automatic hardware load from both character oriented devices and mass storage devices.

When communicating with the MOPC program, the following characters are legal input chatacters:

	Characters:	Use:
	0, 1, 2, 3, 4, 5, 6, 7	Octal digits use to specify addresses and data
	(D	Restart MOPC, clear PIE
	Ø	Octal load
	&	Binary load
	!	Start program in main memory
	/	Specifies register or memory cell examine
	CR (carriage return)	Terminator of line
	LF (line feed)	Echoes, no other effect
	␣ (space)	Octal number before the space is ignored
*)	B	Used to specify bank number
	I	Internal register examine

*) Does not apply for NORD-12.

Characters:	Use:
R	Specifies operation on one of the eight registers STS, D, P, B, L, A, T, X on a specified level
*	Current location counter

All other characters are ignored and followed by "?".

8.1 Functions

8.1.1 Start a Program

Format:

< octal number > !

The machine is started in the address given by the octal number. If the octal number is omitted, the P register is used as start address, i.e., this is a "continue function". The program level will be the same as when the computer was stopped (if Master Clear has not been pushed or (J) typed).

8.1.2 Memory Examine

Format:

< octal number > /

The octal number before the chatacter "/" specifies the physical memory address.

When the "/" is typed, the contents of the specified memory cell are printed out as an octal number.

If a CR (carriage return) is given, the contents of the next memory cell are printed out.

Example:

```
717/003456                  % EXAMINE ADDRESS 717

717/003456 (CR)             % EXAMINE ADDRESS 717
003450 (CR)                 % EXAMINE ADDRESSES 720
000013                      % AND 721
```

8.1.3 Memory Deposit

Format:

octal number (CR)

After a memory examine the contents of the memory cell mat be changed by typing an octal digit terminated by CR.

Example:

```
717/003456 3475  (CR)      % THE CONTENTS OF
003450 1700 (CR)           % ADDRESS 717 IS CHANGED
000123  (CR)               % FROM 3456 TO 3475 AND 720
123456                     % IS CHANGED FROM 3450 TO
                           % 1700.  721 CONTAINS 123 AND
                           % REMAINS UNCHANGED
```

8 1.4 Register Examine

Format:

< octal number > R < octal number > /

The first octal number specifies the program level (0-17), if this number is omitted, program level zero is assumed.

The second octal number specifies which register on that level to examine, the following codes apply:

0	Status register, bits 1-7
1	D register
2	P register
3	B register
4	L register
5	A register
6	T register
7	X register

After the "/" is typed, the contents of the register are printed out.

Examples:

R5/	A register level 0
7R2/	P register level 7

8.1.5 Internal Register Examine

Format:

I octal number /

The octal number specifies which internal register is examined, the following codes apply:

	0		Maintenance only
	1	STS	Status register, program level is contained in bits 8-11, bit 14 = PONI and bit 15 = IONI
	2	OPR	Operator's panel switch register
*	3	PGS	Paging status register
	4	PVL	Previous program level
*	5	IIC	Internal interrupt code
	6	PID	Priority interrupt detect
	7	PIE	Priority interrupt enable
	10		Maintenance only
	11		Maintenance only
	12	ALD	Automatic load descriptor
*	13	PES	Memory error status
	14		Maintenance only
*	15	PEA	Memory error address
	16		Maintenance only
	17		Maintenance only

8.1.6 Current Location Counter

When * is typed, an octal number is printed indicating the current address on which a memory examine or memory deposit will take place. The current location counter is set by the memory examine command /, and it is also incremented for each time carriage return is typed.

8.1.7 Break Function

When à is typed, the MOPC is restarted. This function is also used to terminate an octal load. PIE is set to zero.

8.1.8 Bank Number

Format: *

octal number B

This command is used when the computer has more than 64K memory. The memory is divided into 64K banks (0-3).

This command has to be used to specify the bank number when a memory examine/deposit has to be done.

8.2 Bootstrap Loaders

The NORD-10/NORD-12 has bootstrap loaders for both mass storage and character oriented devices. Three different load formats are standard:

Octal format load

Binary format load

Mass storage load

8.2.1 Octal Format Load

Octal load is (normally) started by typing:

physical device address $

The operator's communication will start taking its input from the device with the specified device address. The actual device must conform with the programming specification of either Teletype or tape reader. The device address is the lowest address associated with the device.

During octal load there is no echoing of characters. All legal operators commands are accepted. Illegal commands terminate the loading and "?" is typed** on the console. Normally a or ! is used to terminate an octal load.

If no device address precedes the $ command, then $ is nearly equivalent to pushing the LOAD button on the operator's panel. (See Section 8.2.4).

---

* Does not apply for NORD-12

** In installations without console an attention lamp is turned on.

8.2.2 Binary Format Load

Binary load is (normally) started by typing:

physical device address &

Loading will take place from the specified device. This device must conform with the programming specification of either Teletype or tape reader. The device address is the lowest address associated with the device.

The binary information must obey the following format:

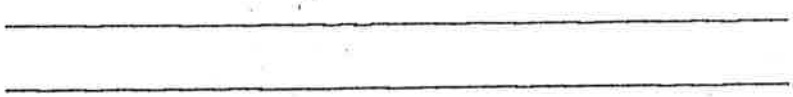


Figure 8.1: Binary load format

A	Any bytes ot including ! (ASCII 41_8)
B	(Optional) octal number (any number of digits) terminated with a non-octal character*:
C	(Optional) octal number terminated with the character ! (see below).
!	Signals start of binary information. (ASCII 41_8).
E	Block start address. Presented as two bytes (16 bits), most significant byte first.
F	Word count. Presented as two bytes (16 bits), most significant byte first. (E, F and H is not included in F).
G	Binary information. Each word (16 bits) presented as two bytes, most significant byte first.
H	Checksum. Presented as two bytes (16 bits), most significant byte first. The checksum is the 16-bit arithmetic sum of all words in G.
I	Action code. If I is a blank (zero), then the program is started in the address previously found in the octal number B (see above). If B is not specified, B=0 is assumed. If I is not a blank, then control is returned to the operator's communication, which decodes I. (The number B will be found in the P register on level 0.)

* Line feed (ASCII 12_8) is ignored within octal numbers.

If no device address precedes the & command, then the & is nearly equivalent to pushing the LOAD button on the operator's panel. (See Section 8.2.4.)

If a checksum error is detected, "?" is typed * on the console and control is returned to the operator's communication.

Note that the binary loader does not require any of the main memory.

The binary load will change the registers on level 0.

The binary load format is compatible with the format dumped by the)BPUN command in the MAC assembler.

8.2.3 Mass Storage Load

When loading from mass storage, 1K words will be read from mass storage address 0 into main memory starting in address 0. After a successful load, the CPU is started in main memory address 0.

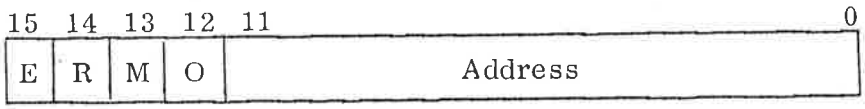
If an error occurs, a new load is tried, If it is never possible to load, Master Clear must be pushed to get the micro-program out of the load sequence.

The actual mass storage must conform with either drum or disc programming specification.

Mass storage load must be started by typing $ or &, or pushing the LOAD button on the operator's panel. If ALD is not set for mass storage load the appropriate code for mass storage load must precede &.

8.2.4 Automatic Load Descriptor

The NORD-10/NORD-12 has a 16-bit switch register called Automatic Load Descriptor (ALD)**. This register specifies the load prodedure to use when the LOAD button is pushed or when a single $ or & is typed.
T
The ALD format is as follows



Automatic Load Descriptor (ALD) Format

---

* In installations without console an attention lamp is turned on.

** Situated on the Panel Driver Card.

E — Extensions

If this bit (bit 15) is 1, then the load function is extended. Effectively the micro program jumps to the micro address found in ALD, bits 0-11.

Note: This bit is active even if the $ and & commands are preceded by a device address.

(The E bit is used when starting micro-programmed diagnostic programs. The start address is put in ALD bits 0-11.

R — Restart*

If this bit (bit 14) is 1, the load junction degenerates to a jump to main memory address:

Address = 4 * (ALD bits 0-13)

This bit is used when the bootstrap program is held in read only main memory. Note: E=0).

M — Mass storage load

If this bit (bit 13) is 1, mass storage load is taken from the device whose (lowest) address is found in ALD bits 0-10 (unit 0).
Note: E=R=0.)

O — Octal format load

If this bit (bit 12) is set, octal format load will take place from the device whose (lowest) address is found in ALD bits 0-10.

Note: & will override this bit, a single & will start a binary format load from the device whose (lowest) address is found in ALD bits 0-10.

If bit 12 is not set, binary format load will take place from the device whose (lowest) address is found in ALD bits 0-10.

Note: $ will override this bit, a single $ will start an octal format load from the device whose (lowest) address is found in ALD bits 0-10. (Note: E=R=M=0.)

---

* Not to be confused with the RESTART button on the operator's panel.

8.2.5 Examples

Following is a table showing possible use of the ALD setting:

Command / ALD	<n> &	&	<n> &	&	Pushing LOAD or < > &
000300	Octal load from <n>	Octal load from 300	Binary load from <n>	Binary load from 300	Binary load from 300
010400	Octal load from <n>	Octal load from 400	Binary load from <n>	Binary load from 400	Octal load from 400
020540	Octal load from <n>	Mass storage load from 540	Binary load from <n>	Mass storage load from 540	Mass storage load from 540
077760	Octal load from <n>	Start in address 177700	Binary load from <n>	Start in address 177700	Start in address 177700
103000	Jump to μ address 3000	Jump to μaddress 3000	Jump to μ address 3000	Jump to μ address 3000	Jump to μ address 3000

Table 8.1

APPENDIX A

NORD-12 MNEMONICS AND THEIR OCTAL VALUES

AAA	: 172400	DP	: 000002
AAB	: 172000	DT	: 000006
AAT	: 173000	DX	: 000007
AAX	: 173400	EQL	: 000000
ACT	: 000400	EXIT	: 146142
ADC	: 001000	EXR	: 140600
ADD	: 006000	FAD	: 100000
AD1	: 000400	FDV	: 114000
ALD	: 000012	FMU	: 110000
AND	: 070000	FSB	: 104000
,B	: 000400	GRE	: 001000
BAC	: 000600	I	: 001000
BANC	: 177000	IDENT	: 143600
BANC	: 177200	IF	: 000000
BCM	: 000400	IOF	: 150401
BLDA	: 176600	ION	: 150402
BLDC	: 176400	IOX	: 164000
BORA	: 177600	IRR	: 153600
BORC	: 177400	IRW	: 153400
BSET	: 174000	JAF	: 131400
BSKP	: 175000	JAN	: 130400
BSTA	: 176200	JAP	: 130000
BSTC	: 176000	JAZ	: 131000
CLD	: 000100	JMP	: 124000
CM1	: 000200	JNC	: 132400
CM2	: 000600	JPC	: 132000
COPY	: 146100	JPL	: 134000
DA	: 000005	JXN	: 133400
DB	: 000003	JXZ	: 133000
DD	: 000001	LBYT	: 142200
DL	: 000004	LDA	: 044000
DNZ	: 152000	LDD	: 024000

LDF	: 034000	RMPY	: 141200
LDT	: 050000	RORA	: 145400
LDX	: 054000	ROT	: 001000
LIN	: 003000	RSUB	: 146600
LMP	: 000002	SA	: 000050
LRB	: 152600	SAA	: 170400
LST	: 003000	SAB	: 170000
MCL	: 150200	SAD	: 154600
MGRE	: 001400	SAT	: 171000
MIN	: 040000	SAX	: 171400
MIX3	: 143200	SB	: 000030
MLST	: 003400	SBYT	: 142600
MON	: 153000	SD	: 000010
MPY	: 120000	SHA	: 154400
MST	: 150300	SHD	: 154200
NLZ	: 151400	SHR	: 000200
ONE	: 000200	SHT	: 154000
OPR	: 000002	SKA	: 001000
ORA	: 074000	SKP	: 140000
PID	: 000006	SL	: 000040
PIE	: 000007	SP	: 000020
PIN	: 002000	SRB	: 152402
PL10	: 000004	SSC	: 000060
PL11	: 000011	SSK	: 000020
PL12	: 000022	SSM	: 000070
PL13	: 000043	SSO	: 000050
PVL	: 000004	SSQ	: 000040
RADD	: 146000	SSZ	: 000030
RAND	: 144400	ST	: 000060
RCLR	: 146100	STA	: 004000
RDCR	: 146200	STD	: 020000
RDIV	: 141600	STF	: 030000
REXO	: 145000	STS	: 000001
RINC	: 146400	STT	: 010000

STX	: 014000	TRR	: 150100
STZ	: 000000	UEQ	: 002000
SUB	: 064000	WAIT	: 151000
SWAP	: 144000	,X	: 002000
SX	: 000070	ZIN	: 002000
TRA:	: 150000	ZRO	: 000000

APPENDIX B

NORD-10/NORD-12 INSTRUCTION CODE

			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	000.000	STZ	0	0	0	0	0											
	004.000	STA	0	0	0	0	1											
	010.000	STT	0	0	0	1	0											
	014.000	STX	0	0	0	1	1											
1	020.000	STD	0	0	1	0	0											
	024.000	LDD	0	0	1	0	1											
	030.000	STF	0	0	1	1	0											
	034.000	LDF	0	0	1	1	1											
2	040.000	MIN	0	1	0	0	0	X	I	B	Displacement Δ							
	044.000	LDA	0	1	0	0	1											
	050.000	LDT	0	1	0	1	0											
	054.000	LDX	0	1	0	1	1											
3	060.000	ADD	0	1	1	0	0											
	064.000	SUB	0	1	1	0	1											
	070.000	AND	0	1	1	1	0											
	074.000	ORA	0	1	1	1	1											
4	100.000	FAD	1	0	0	0	0											
	104.000	FSB	1	0	0	0	1											
	110.000	FMU	1	0	0	1	0											
	114.000	FDV	1	0	0	1	1											
5	120.000	MPY	1	0	1	0	0											
	124.000	JMP	1	0	1	0	1											
	130.000	CJP	1	0	1	1	0	Subin.										
	134.000	JPL	1	0	1	1	1											
6	140.000	SKP+EXT	1	1	0	0	0						S			D		
	144.000	ROP	1	1	0	0	1											
	150.000	MIS	1	1	0	1	0	Subin.										
	154.000	SHT	1	1	0	1	1						No. of shifts					
7	160.000	IOT	1	1	1	0	0				Device number							
	164.000	IOX	1	1	1	0	1	Device address										
	170.000	ARG	1	1	1	1	0	Fncn.			Argument							
	174.000	BOP	1	1	1	1	1	Function					Bit no.			D		
			100.000	40.000	20.000	10.000	4.000	2.000	1.000	400	200	100	40	20	10	4	2	1

APPENDIX C

DIFFERENCES BETWEEN NORD-10 AND NORD-12

This appendix is provided for those comparing NORD-10 with NORD-12.

1. Maximum memory size of the NORD-12 is 64K words. The Memory Management System option is not available for the NORD-12. With this option the maximum memory size for NORD-10 is 256K words.

2. The speed of the NORD-10 is 300 ns per micro-instruction for NORD-12 it is 500 ns per micro-instruction. This gives a 5/3 ratio for floating point, for all other instructions consult the NORD-10 and NORD-12 Reference Manuals.

3. Internal hardware error interrupts are not connected to level 14 on the NORD-12, and there is no IIE (Internal Interrupt Enable) register and no IIC (Internal Interrupt Code) However, on the NORD-12 it is possible to hardwire interrupts to level 14.

4. NORD-12 is only available with dynamic MOS memories (4096 bit/chip), while the NORD-10 is offered with a range of different memories.

5. The rounding algorithm for floating point differs between NORD-10 and NORD-12. On NORD-12 there is no TG (Rounding) flip-flop in the Status Register (bit 1 in Status on NORD-10), and for NORD-12 all floating point results are truncated. On the NORD-10 the least significant bit in the result is forced to one if the result could not be exactly represented. For both the NORD-10 and NORD-12 all integers up to $2^{32}-1$ will be exactly represented in floating point format, and all results to this limit will also be exact.

6. Only NORD-10 may have a NORD-1 Input/Output Channel as option.

7. The Memory Parity system differs between NORD-10 and NORD-12. On NORD-10 TRA instructions are used for reading memory parity error information, on NORD-12 the parity mechanism is an Input/Output device which plugs into an I/O Slot. For both NORD-10 and NORD-12 Memory Parity is an option.

8. On NORD-10 the Bus Transceiver or Bus Extender is standard, this is an option on NORD-12.

– we want bits of the future

A/S NORSK DATA-ELEKTRONIKK ØKERNVEIEN 145 OSLO 5 NORWAY PHONE: 21 73 71 TELEX: 18284