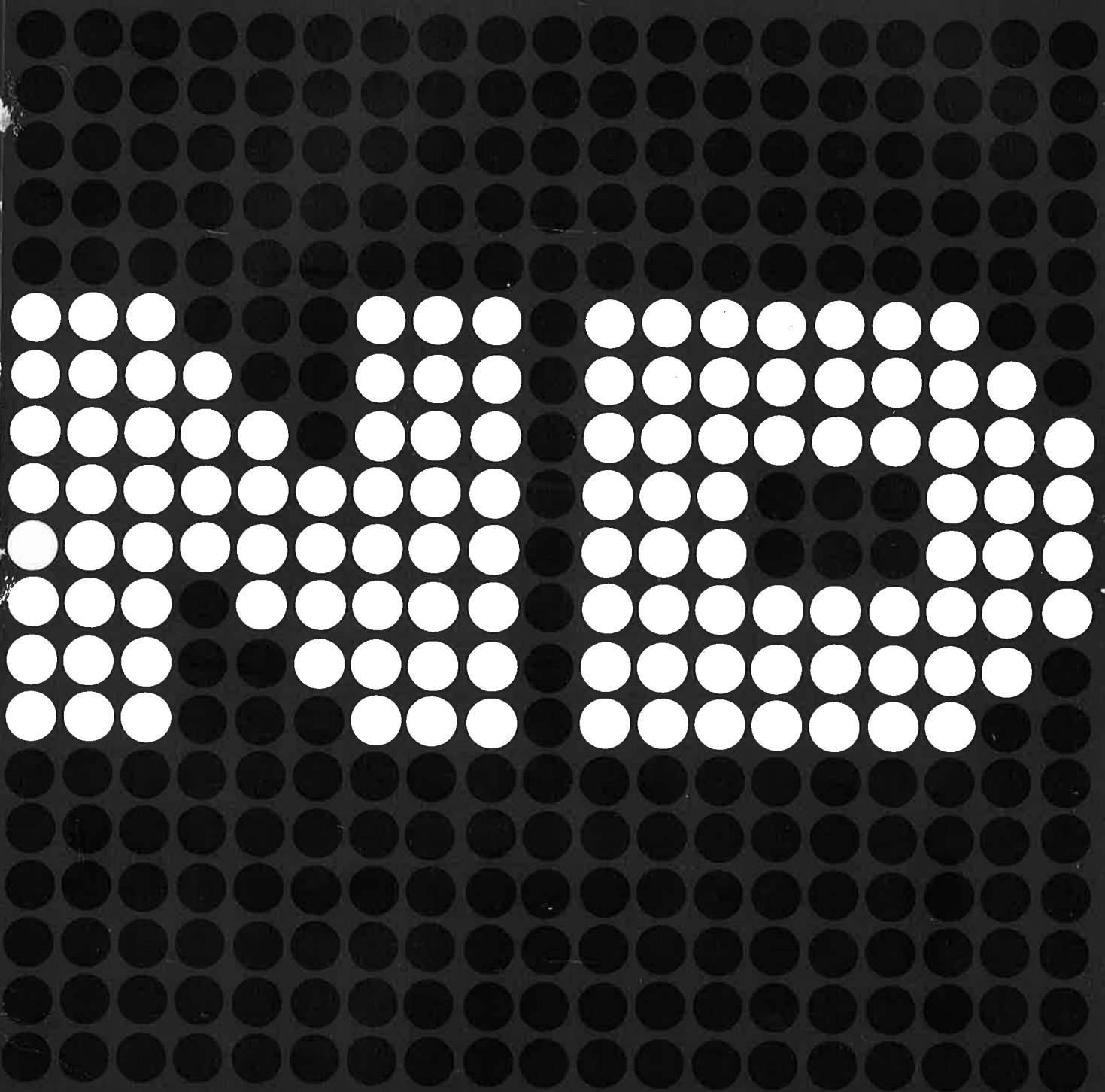
ND-100
Reference Manual

ND-06.014.02

NORSK DATA A.S



ND-100
Reference Manual

ND-06.014.02

NOTICE

The information in this document is subject to change without notice. Norsk Data A.S assumes no responsibility for any errors that may appear in this document. Norsk Data A.S assumes no responsibility for the use or reliability of its software on equipment that is not furnished or supported by Norsk Data A.S.

PRINTING RECORD

Printing	Notes
03/79	ORIGINAL PRINTING
01/82	SECOND PRINTING

NORSK DATA A.S
P.O. Box 4, Lindeberg gård
Oslo 10, Norway

Manuals can be updated in two ways, new versions and revisions. New versions consist of a complete new manual which replaces the old manual. New versions incorporate all revisions since the previous version. Revisions consist of one or more single pages to be merged into the manual by the user, each revised page being listed on the new printing record sent out with the revision. The old printing record should be replaced by the new one.

New versions and revisions are announced in the ND Bulletin and can be ordered as described below.

The reader's comments form at the back of this manual can be used both to report errors in the manual and to give an evaluation of the manual. Both detailed and general comments are welcome.

These forms, together with all types of inquiry and requests for documentation should be sent to the local ND office or (in Norway) to:

Documentation Department
Norsk Data A.S
P.O. Box 4, Lindeberg gård
Oslo 10

TABLE OF CONTENTS

+ + +

Section: *Page:*

Section: *Page:*

Section: *Page:*

Section: *Page:*

Section: *Page:*

Appendixes: *Page:*

1 INTRODUCTION TO ND-100

1.1 *GENERAL CHARACTERISTICS*

ND-100 is a general purpose computer and it is used in many applications like:

- Commercial data processing.
- Research.
- Education.
- Process control.

ND-100 is completely software compatible with NORD 10/S and runs the same operating system, SINTRAN III /VS.

The ND-100 Central Processing Unit (CPU) is placed on a single module. The word length is 16 bits in parallel.

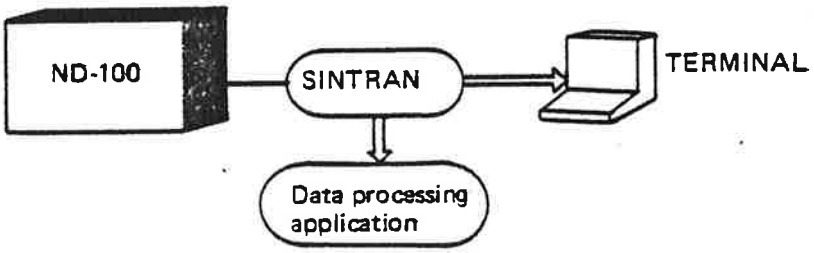


Figure 1.1: The Operating System SINTRAN III/VS allows the ND-100 to be used in many Applications.

1.2 *ND-100 FUNCTIONAL MODULES*

1.2.1 *General*

A standard ND-100 printed board module size is 366.8 mm x 280 mm.

The board size, together with the use of Large Scale Integrated (LSI) circuits, allows:

- Small physical dimensions.
- Closely related functions placed on the same module, thus reducing external wiring to a minimum.

Communication between ND-100 functional modules is done through an advanced high-speed bus, called ND-100 bus. The ND-100 bus is a printed back plane. The bus is available in two versions, one for connecting 12 modules and one for connecting 21 modules. The two versions are mounted in different card crates and different cabinets.

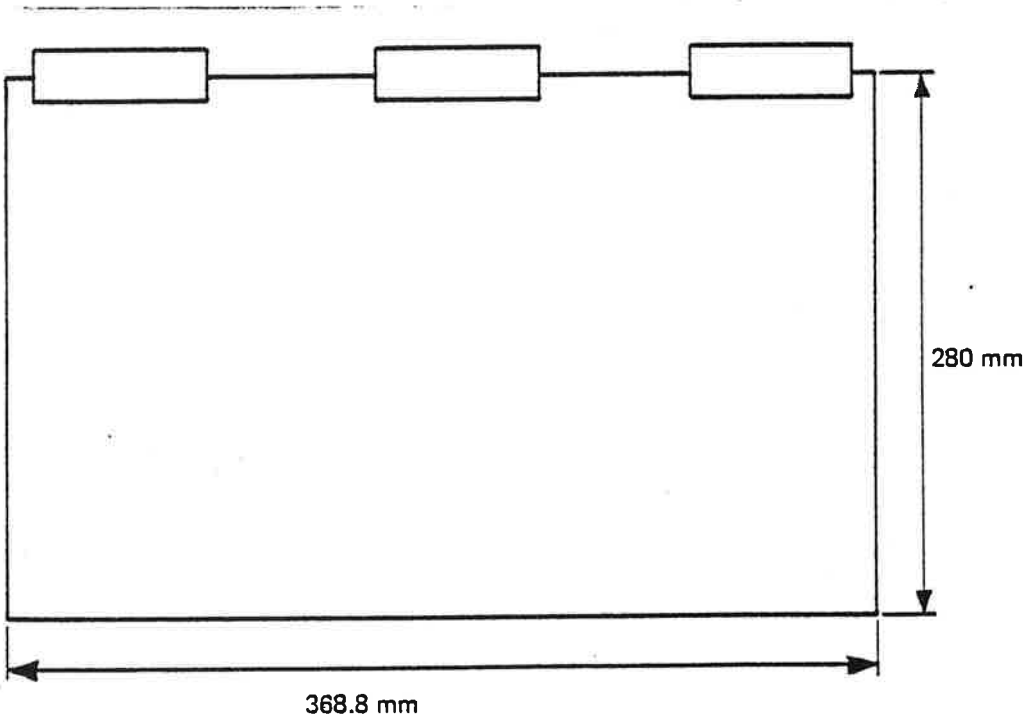


Figure 1.2: The Standard ND-100 Printed Board Module.

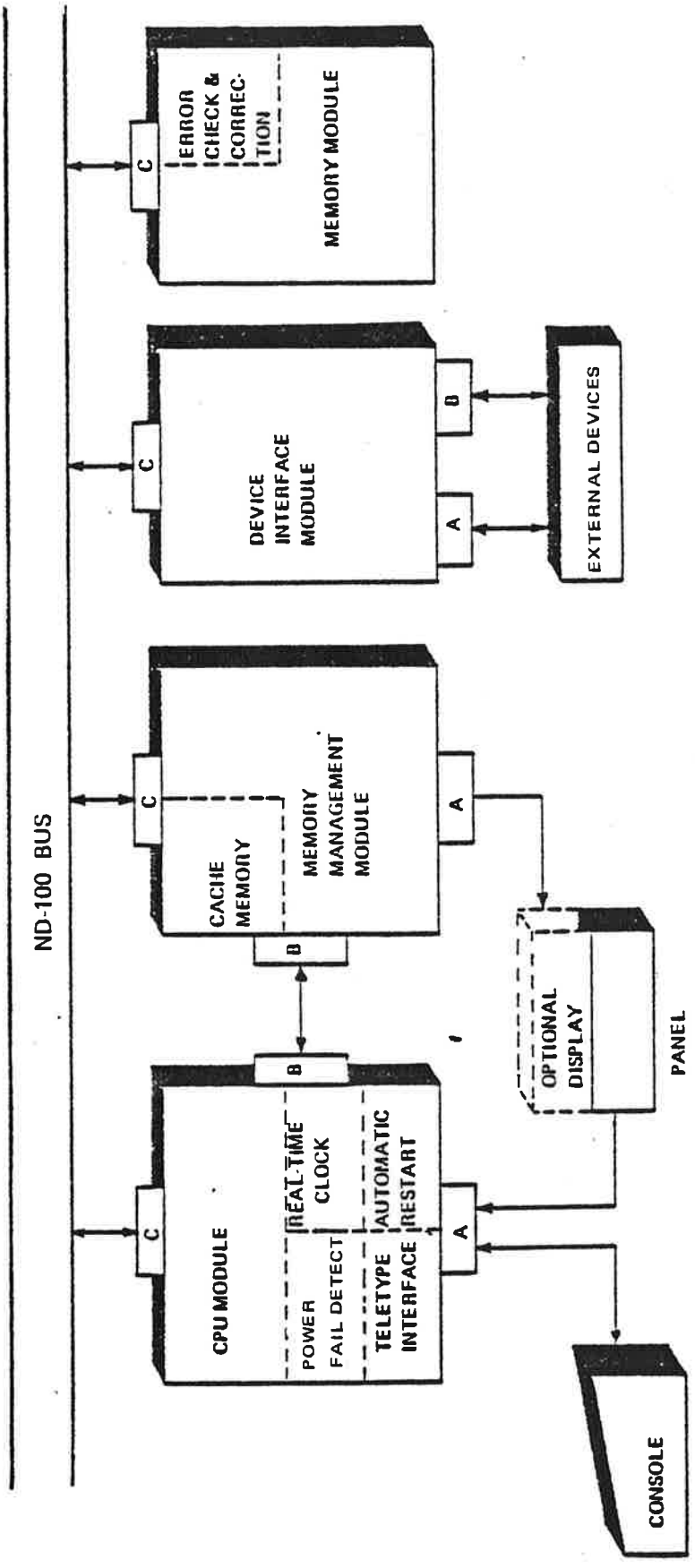


Figure 1.3: ND-100 Modules Connection (A, B and C are Plugs on the Modules).

1.2.2 *ND-100 Central Processing Unit (CPU) Module*

The CPU module contains, in addition to the CPU itself:

- A real-time clock.
- A current loop terminal interface with switch selectable speeds, 110 - 9600 baud/bps (bits per second).
- Power fail and automatic restart.

1.2.2.1 CPU Characteristics

The Processor

ND-100 CPU is a 16 bit parallel processor designed around the bit slice ALU (arithmetic logic unit) element.

The processor is controlled by a microprogram. The following is implemented in the microprogram:

- All instructions.
- Operator communication.
- Built-in test routines.
- Bootstrap loaders.

The microprogram is physically located in a 2k word by 64 bit Read Only Memory (ROM). One microinstruction is fetched and executed in the internal CPU cycle time. The cycle time is 150 ns for the fast CPU and 190 ns for the slow version.

Instruction Prefetch

A fast processor should not have to wait for instructions. In order to reduce instruction fetch waiting time, the ND-100 CPU will normally hold two instructions, the current executing instruction and the next one. This is accomplished by fetching the next instruction while executing the current instruction.

Special Feature

To allow dynamic microprogramming, a 256 word by 64 bit writeable control store is available as an option.

Instruction Set and Data Format

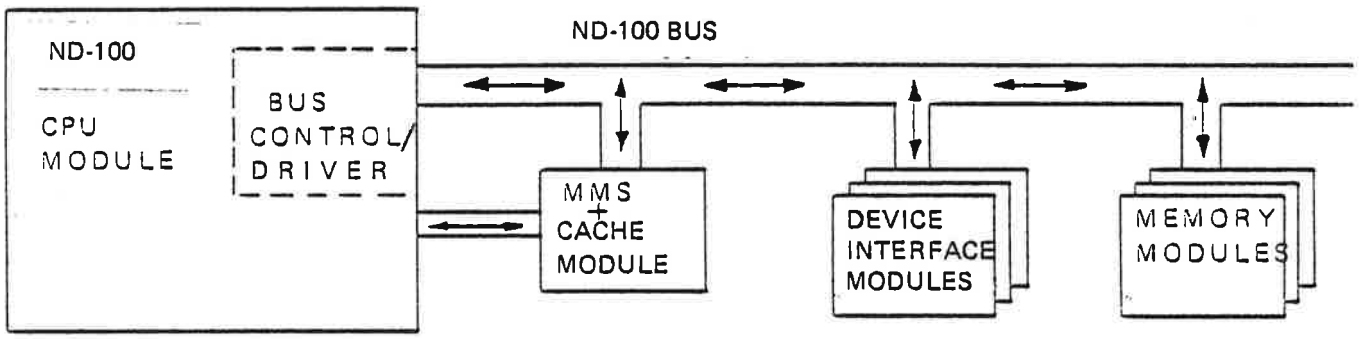
Although a standard ND-100 word is 16 bits, the computer has a comprehensive instruction set which includes operations on:

- Bits.
- Bytes.
- Single words.
- Double words.
- Triple words.
- Register file.
- Fixed or floating point arithmetic (32 - or 48 - bit word).

1.2.3 *ND-100 Architecture*

1.2.3.1 General

Figure 1.4 shows the ND-100 bus structure. The main highway for data and addresses in the system is the ND-100 bus. Data and address flow are shown by the arrows.



MMS = Memory Management System

Figure 1.4: ND-100 Bus Structure

Physically, the bus is organized as a printed backplane containing 12 or 21 "plug in" positions for module connection.

All communication between ND-100 modules except CPU, MMS and CACHE communication, is provided by this bus. That is, the ND-100 bus connects the:

- CPU to the memory system (including MMS and CACHE).
- CPU to the input/output system.
- DMA controllers to the memory system (DMA = Direct Memory Access). DMA controller is a special device interface module.

A bus control/driver, which is an integrated part of the CPU, controls the activity on the bus. This common bus architecture has several advantages:

- Uniform connection for all modules makes the system flexible and easy to expand.
- No external wiring of busses gives a more reliable system.
- No overhead in connecting several busses between source and destination makes a faster system (one crate system only).

1.2.3.2 ND-100 Configuration Examples

Figure 1.5 shows a typical medium sized ND-100 single processor system.

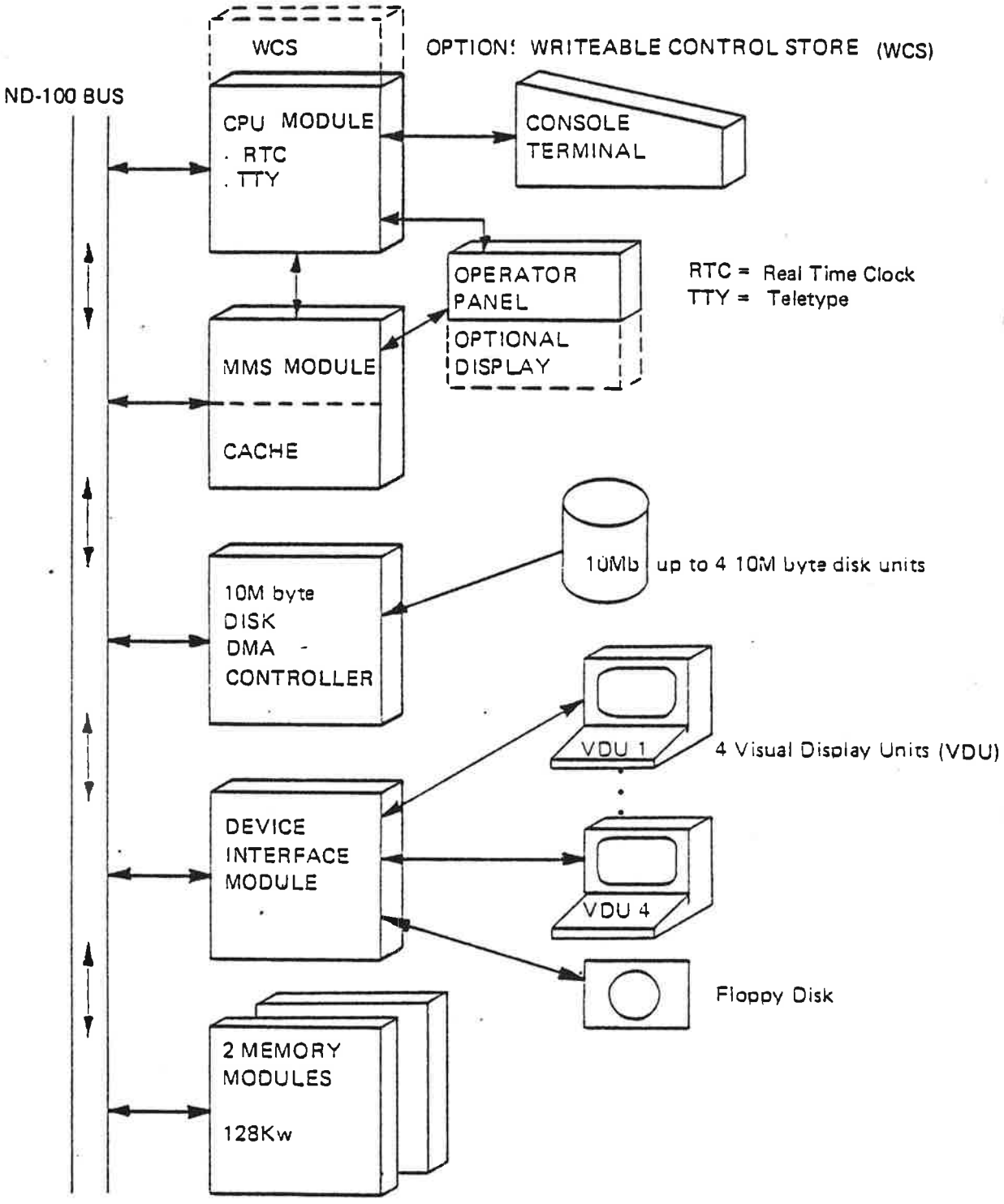


Figure 1.5: ND-100 Configuration Example

1.2.3.3 Multiprocessor Systems

For in-house communication between two ND-100s, between a ND-100 and NORD-10/S, or between a ND-100 and a NORD-50, a shared memory system could be used.

The shared memory system is available through the Big Multiport Memory System (BMPM) which allows up to four sources to access the same physical storage.

Example:

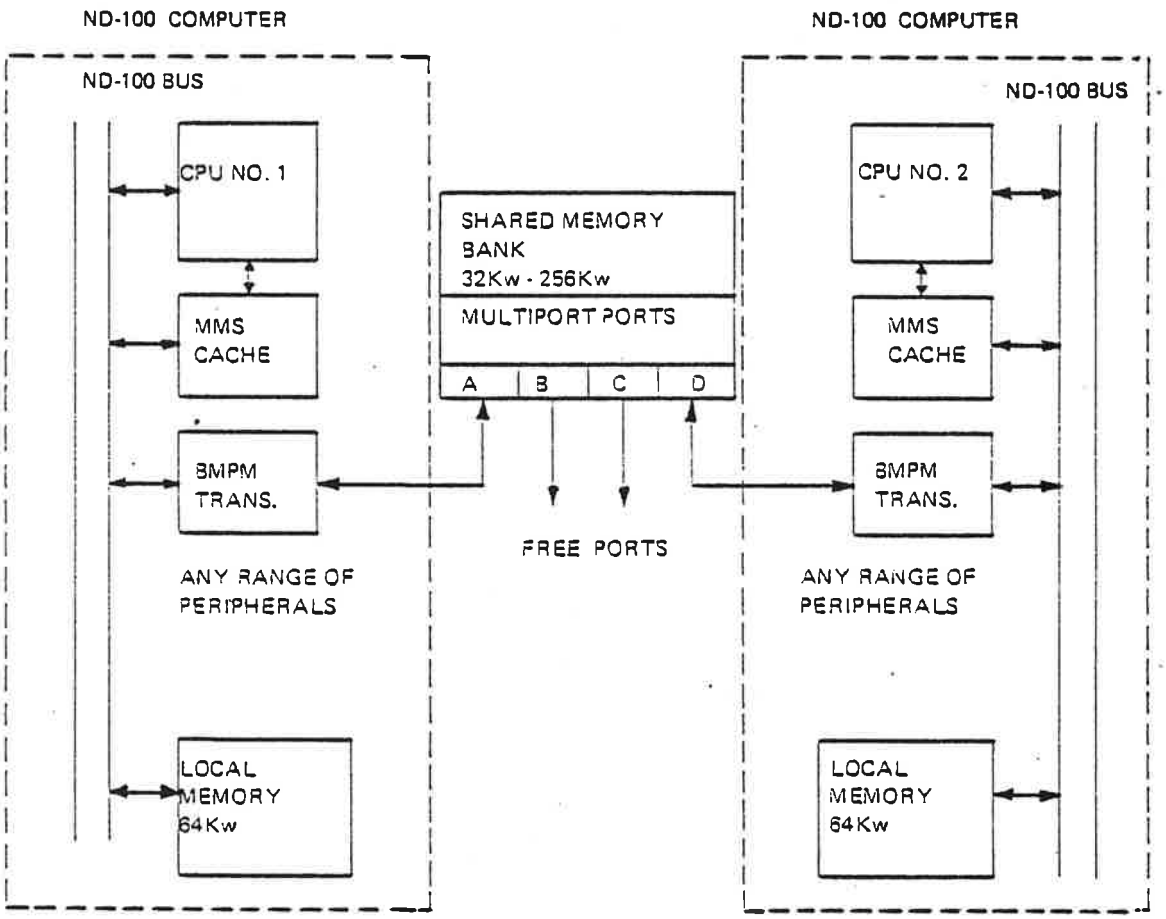


Figure 1.6: Communication between two ND-100 Computers using the Multiprocessor System.

1.2.3.4 Remote Operation

Remote operation in this context means one ND-100 being controlled by another ND-100. The two machines may be in the same room or connected via telephone lines using low or high speed modems.

The HDLC module is designed for this kind of operation, including DMA controlled communication. Figure 1.7 shows an example.

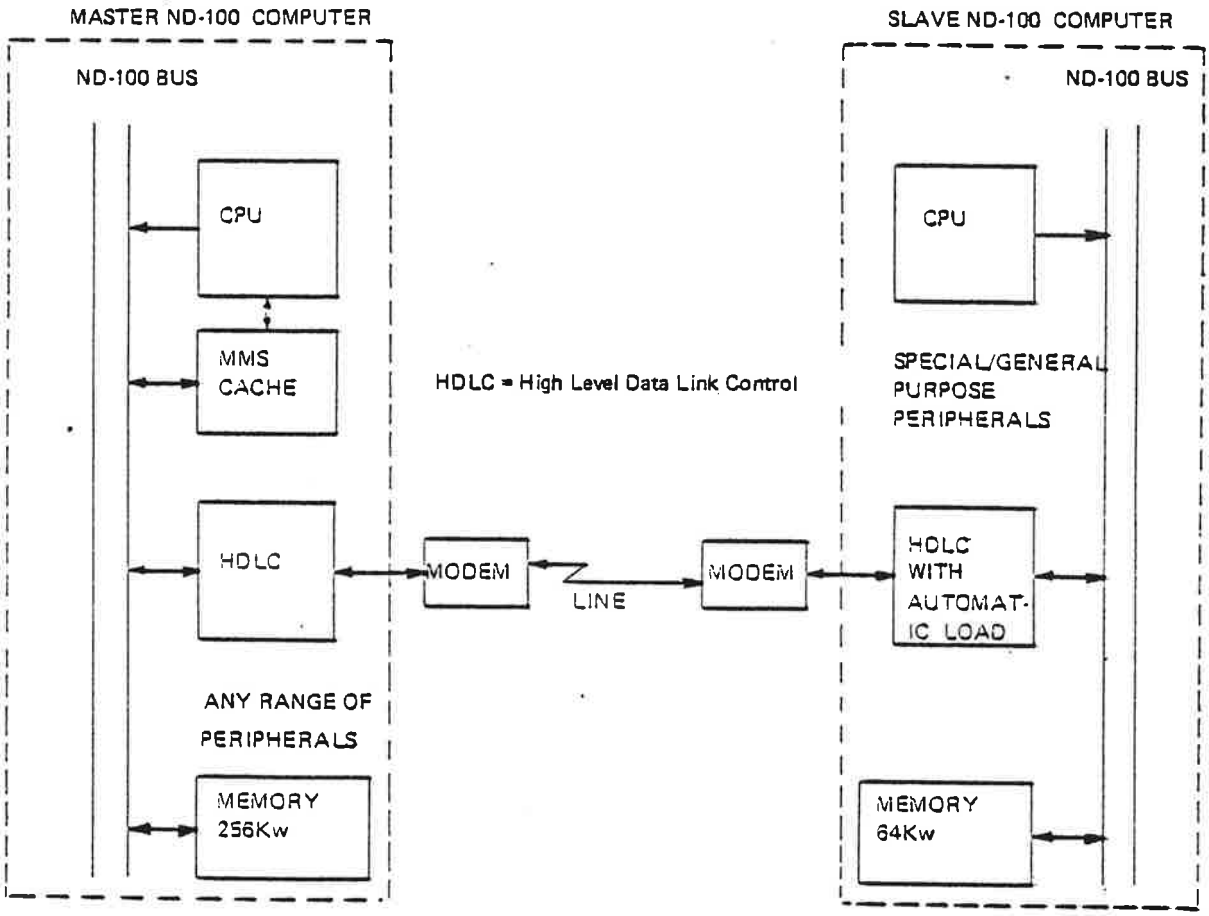


Figure 1.7: Connection between two ND-100 computers using a Telephone Line.

1.3 *THE INTERRUPT SYSTEM*

The ND-100 has a 16 level priority interrupt system, marked PL 0-15

To each level is assigned a complete set of all central registers: STS, A, D, T, L, X B and P

These registers plus eight scratch registers are located in a high speed register file close to the the CPU arithmetic both located on the CPU module.

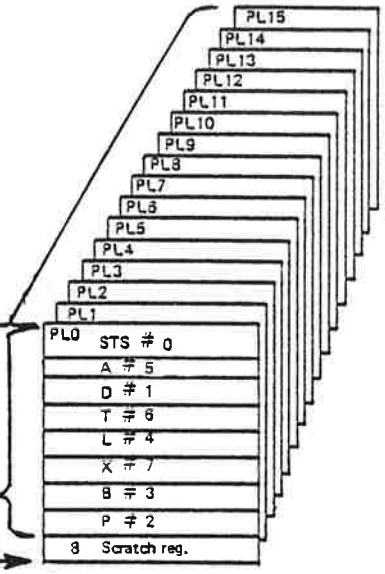


Figure 1.8: High Speed Register File.

With this architecture, switching between two program levels is reduced to selecting the working set of control registers. The time required for this operation is only 5 µs.

All program levels may be activated by software. In addition, each of the levels 10, 11, 12 and 13 may be activated by 512 vectored I/O interrupts. An IDENT instruction is used to identify the interrupting device.

Program level 14 is used by the Internal Interrupt System, which monitors error conditions or traps in the CPU. Program level 15 may only have one I/O interrupt source.

Program level 15 is not used by standard NORD equipment or software, but is available for users who need immediate access to the CPU.

The high speed register file is described in further detail later in this manual.

1.4 *THE MEMORY MANAGEMENT SYSTEM (MMS)*

The hardware memory management module is necessary for running the SINTRAN III/VS (Virtual Storage) operating system. The SINTRAN III/VS operating system includes:

— 64 K words (128 K bytes) virtual address range for each user independent of physical memory capacity.

— Dynamic allocation/relocation of programs in memory.

— Memory protection.

The implementation of the memory management system is based on two major subsystems:

— The Paging System.
— The Memory Protection System.

The *paging system* maps a 16 bit virtual address (describing a user's 64 K word virtual storage) into a 19 bit physical address, thus extending the physical address space to 512 K words. The paging system also has an extended mode which handles physical memory space up to 16 M words (32 M bytes). This mode gives a 24 bit physical address.

The implementation of paging is based on dividing physical memory into 1 K word pages which, under operating system control, are assigned to active programs.

Four page tables of 64 words each hold the physical page numbers assigned to an active program. These tables are located in high speed registers, reducing paging overhead to practically zero.

The *memory protection system* may be divided into two subsystems:

— The Page Protect System.
— The Ring Protect System.

The *page protect system* allows a page to be protected from read, write or instruction fetch accesses or any combination of these.

The *ring protect system* places each page and each user on one of four priority rings.

A page on one specific ring may not be accessed by a program that is assigned a lower priority ring number. This system is used to protect system programs from user programs, the operating system from its subprograms and the system kernel from the rest of the operating system.

1.5 *THE MEMORY SYSTEM*

The memory system has a flexible and hierarchial architecture. The memory system includes:

- 1K words (2K bytes) CACHE memory.
- Up to 16 M words main memory.
- Memory channel to the multiport memory system.

1.5.1 *Main Memory*

Main memory can have any size from:

32K words to 16 M words in steps of 32K words.

Each word in main memory is stored with a 6 bit error correction code which makes it possible to:

- Correct and log single bit errors.
- Detect and report all double errors and most multiple errors.

Seen from the program, memory access time depends on the effect of prefetch.

1.5.2 *Cache Memory*

Cache memory is optional and physically located on the memory management module.

The presence of cache memory will reduce average memory access time significantly. Cache is a high speed bipolar memory.

The purpose of cache memory is to hold the most recent data and instructions to be processed.

1.5.3 *Multiport Memory*

In order for the ND-100 to access the NORD-10/S Big multiport memory, a multiport memory transceiver is available.

1.6 *THE INPUT/OUTPUT SYSTEM*

The ND-100 input/output system is designed to be a flexible system providing communication between slow, character oriented devices as well as high speed, block oriented devices.

Depending on the speed, a device could be connected to ND-100 with:

- CPU controlled, Programmed Input/Output (PIO).
- With Direct Memory Access (DMA).

PIO is used for slow devices and DMA for fast devices.

1.6.1 *Programmed Input/Output — PIO*

Program controlled input/output always operates via the A register, which implies that each word of input/output has to be programmed via this register.

1.6.2 *Direct Memory Access — DMA*

A Direct Memory Access (DMA) channel is used to obtain high transfer rates to and from main memory. CPU activity and DMA transfers may be performed simultaneously, i.e., the DMA transfer is not controlled by the CPU as a PIO transfer is.

More than one DMA device may be active at the same time, sharing the total band width of the DMA channel. Total band width is 1.8 M words per second.

1.7 *ND-100 PERIPHERAL EQUIPMENT*

Most computer peripherals can be connected to ND-100. The range of standard peripherals includes:

Sequential Devices

- Terminals.
- Card readers.
- Line printers/plotters.

Mass Storage Devices

- Magnetic tapes.
- Disks from 10M bytes to 288M bytes per disk. Up to 4 disks may be connected to each input/output card.
- Floppy disks.

Computer networks

- Asynchronous modem controllers.
- Synchronous modem controllers including selectable frame format, HDLC or bisync.

In addition, ND-100 can be equipped with a NORD-10/S bus adapter which gives access to all NORD-10/S peripherals.

1.8 *ND-100 SOFTWARE*

1.8.1 *The Operating System*

The standard operating system for ND-100 computers is SINTRAN III, which may be delivered in two versions:

1. SINTRAN III/VS (Virtual Storage)

SINTRAN III/VS is a general purpose mass storage based operating system offering facilities for

- Real-time.
- Timesharing.
- Batch processing.

2. SINTRAN III core for machines without mass storage devices intended for real-time applications in process control and data communication.

1.8.2 *Supporting Software*

A number of programming languages and software systems complement the capabilities of the ND-100 SINTRAN III/VS.

- ND standard FORTRAN following ANSI standard for FORTRAN IV with certain extensions.
- ND COBOL system following the ANSI-74 COBOL standards.
- ND BASIC compiler, an extended version of the program generator for business oriented applications.
- PLANC, a high level system programming language.
- MAC assembler with macro expansions.
- QED, an interactive text editor.
- The SIBAS data base management system, designed in accordance with the Codasyl data base recommendations.
- ND TPS (Transaction Processing System) offering the necessary operational system software for development of transaction processing programs.
- ND Screen Handling System (NSHS), an interactive program to create, modify and use screen pictures.
- ND Data Entry System, a set of software modules designed to simplify terminal oriented data entry operations.

Other useful utility programs are ND SORT Package, Scientific Subroutine Library, Commercial Subroutine Library, ND PLOT Package.

For data communication with large scale computers, there are terminal emulator packages for: IBM 360/370, HB-6000, CDC CYBER, UNIVAC and others.

1.8.3 *Distributed Data Processing*

ND NET is a communication system for computer networks, enabling users to communicate with other computers in a network.

2 SYSTEM DESCRIPTION

2.1 *CENTRAL PROCESSOR*

2.1.1 *General*

ND-100 is microprogrammed and all instruction execution is in firmware using a 2K x 64 bit, fast Read Only Memory (ROM). To allow dynamic microprogramming, a 256 word by 64 bit writeable control store is optional. This gives the possibility of extending the ND-100 instruction set for special applications. The address arithmetic is also implemented in microprogram. This means that the addressing structure of ND-100 can be changed by rewriting the microprogram.

2.1.2 *Internal Communication*

The internal communication in the CPU is performed over the internal data bus (IDB). A bus is a highway for information, where only one word of information may travel at a time. The microprogram enables the information for the IDB from a certain source, and gives enable signals to the destination parts in the CPU where the information is needed.

Figure 2.1 shows how the IDB communicates with the central parts in the CPU. The memory management system and cache are connected directly to IDB and ND-100 bus for faster access. The bus control is implemented on the CPU module and controls the activity on the ND-100 bus.

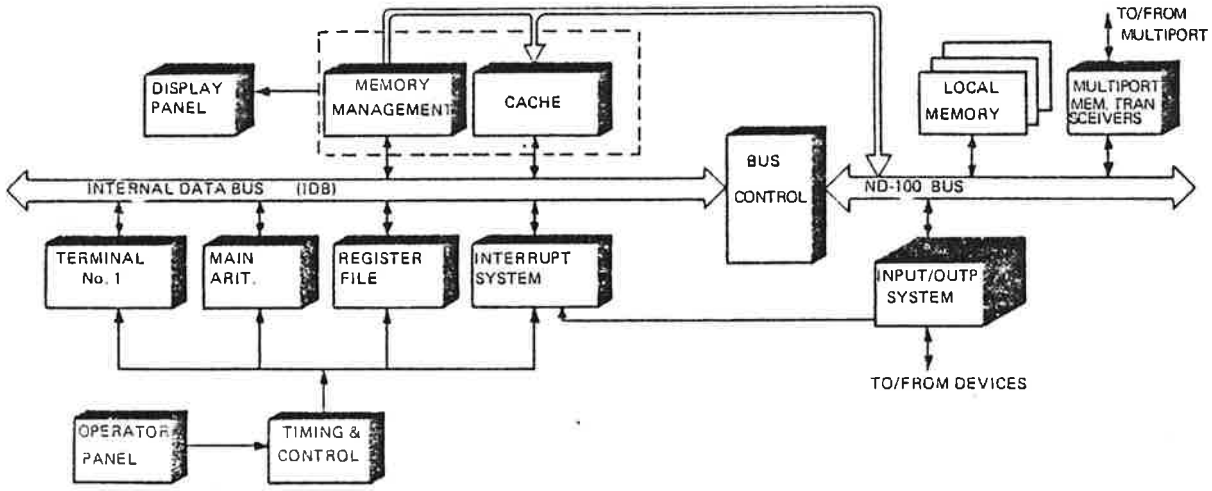


Figure 2.1: ND-100 Bus Structure

2.1.3 *The Address Arithmetic*

The address arithmetic in the ALU (arithmetic logic unit) forms a 16 bit address. The control of the address arithmetic is implemented in a microprogram. The 16 bit address goes to the memory management system. If the memory management module is not present, the address goes directly to the memory system via the ND-100 bus.

2.1.4 *Instruction Fetch*

The machine instructions to be executed reside in memory. The program counter, PC, is enabled for the ND-100 and a request is sent to memory. The instruction from memory is loaded into the prefetch register.

2.1.5 *Prefetch*

ND-100 uses prefetch. That is, the next instruction is fetched simultaneously with the execution of the current one. Consequently, an instruction fetch consists of copying the prefetch register to the instruction register.

The use of prefetch requires a strictly sequential program. In case of branch instructions or program change (interrupt), the prefetched instruction is skipped and a new instruction found.

Prefetch will not generate page fault if the last instruction before a page limit is a branch instruction.

Prefetch does not give any limitations in programming. For example, STA * +1 is legal but adds 1 *ms* to the execution time compared to STA < disp '1>.

2.1.6 *Instruction Execution*

The instruction to be executed will be loaded into the instruction register (IR) and the instruction map. Refer to Figure 2.2 The lower vector bits of the instruction are taken to IR and the upper operation code bits are taken care of by the map. This is a read only memory (ROM), where each different instruction gives a fixed program address to the microprogram sequencer. Since one machine instruction is executed by a number of instructions residing in microprogram control store, an instruction dependent address should be generated and this is the task of the microprogram sequencer.

This address is sent to the microprogram control store, which gives the logic control bits of the first microinstruction. These signals, together with the timing module, control the operation of the CPU. The operation specified by one microinstruction normally takes 150/190 ns (with cache/without cache). This time is referred to as a micro cycle. When a micro cycle is completed, the next microinstruction has already been read out from the microprogram control store.

2.1.7 *Main Arithmetic*

Refer also to Figure 2.2

From the A and B selector the arithmetic logic unit (ALU) receives the information about which A and B operand to select in the arithmetic operation. The ALU performs all the arithmetical and logical operations as specified in the instruction set. The bit slice, ALU, is completely controlled from the microprogram.

The ALU with its current registers has a two-way communication over IDB with the register file for loading and storing of the current register set.

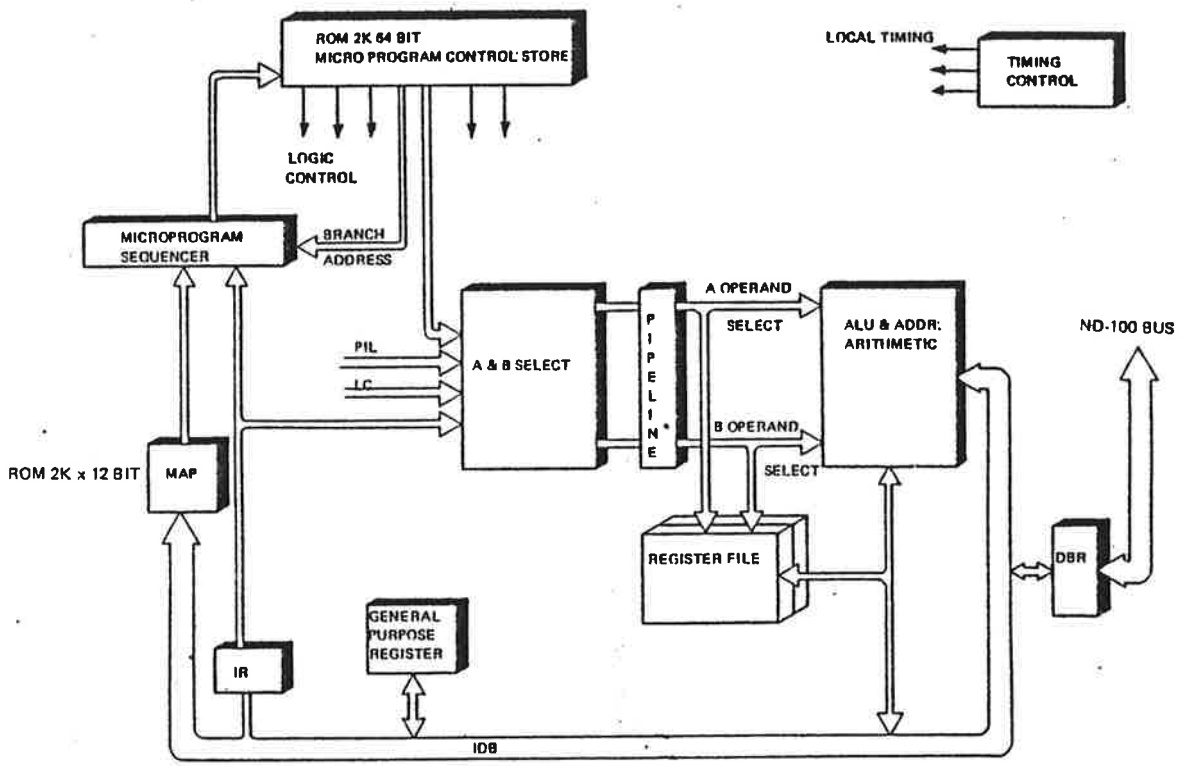


Figure 2.2: Instruction Execution

2.1.8 *The Register File*

Refer to Figure 2.3

There are 16 register sets in the ND-100, one for each of the 16 program levels. Each of the register sets consists of 8 general programmable registers and 8 scratch registers for microprogram use only. There is a total of 256 registers; these are referred to as the register file.

The 8 general registers are:

Status register (STS)

This register holds the indicators described in the status indicators section.

A register

This is the main register for arithmetic and logical operations directly with operands in memory. This register is also used for input/output communication.

D register

This register is an extension of the A register in double precision or floating point operations. It may be connected to the A register during double length shifts.

T register

Temporary register. In floating point instructions it is used to hold the exponent part. It is also used with the IOXT instruction to hold the device address.

L register

Link register. The return address after a subroutine jump is contained in this register.

X register

Index register. In connection with indirect addressing it causes post indexing.

B register

Base register or second index register. In connection with indirect addressing, it causes preindexing.

P register

Program counter, address of current instruction. This register is controlled automatically in the normal sequencing or branching mode. But it is also fully program controlled and its contents may be transferred to or from other registers.

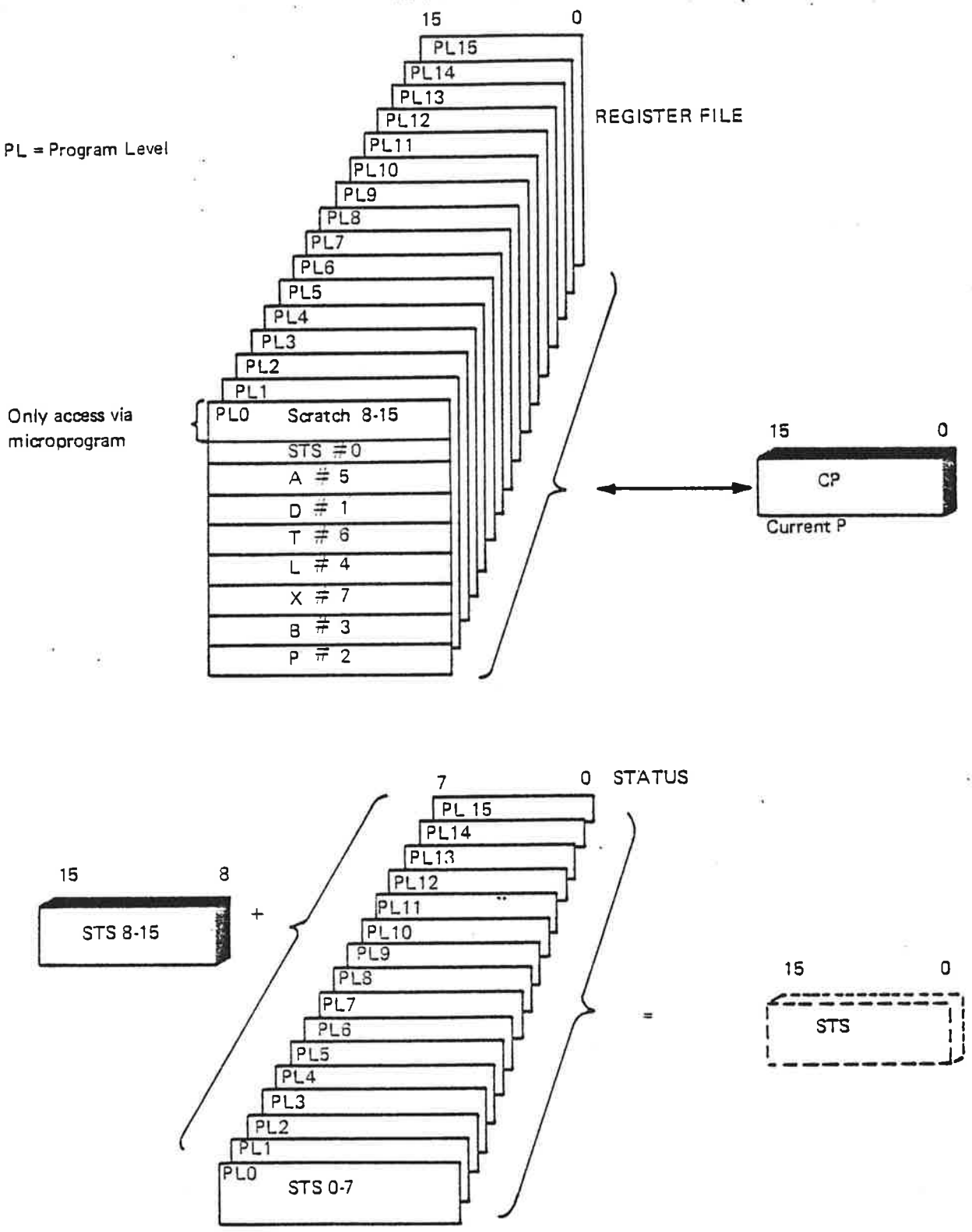


Figure 2.3: Register File

The current register set is held in the ALU and under level change this register set is stored in the register file. The register set for the new level is loaded to the ALU. Any registers or levels can be read or written by specifying register and level information.

2.1.9 *Status Indicators*

Eight indicators are accessible by programs. These 8 indicators are:

M — Multishift 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.

C — Carry indicator. The carry indicator is dynamic.

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

Q — Dynamic overflow indicator.

Z — Error indicator. This indicator is static and remains set until it is reset by program. The Z indicator may be internally connected to an interrupt level such that an error message routine may be triggered.

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

TG — Rounding indicator for floating point operations.

PTM — Page table modus. Enables use of the alternate page table.

These 8 indicators are fully program controlled either by means of the BOP instruction or by the TRA or TRR instructions where all indicators may be transferred to and from the A register. Refer to Figure 2.4.

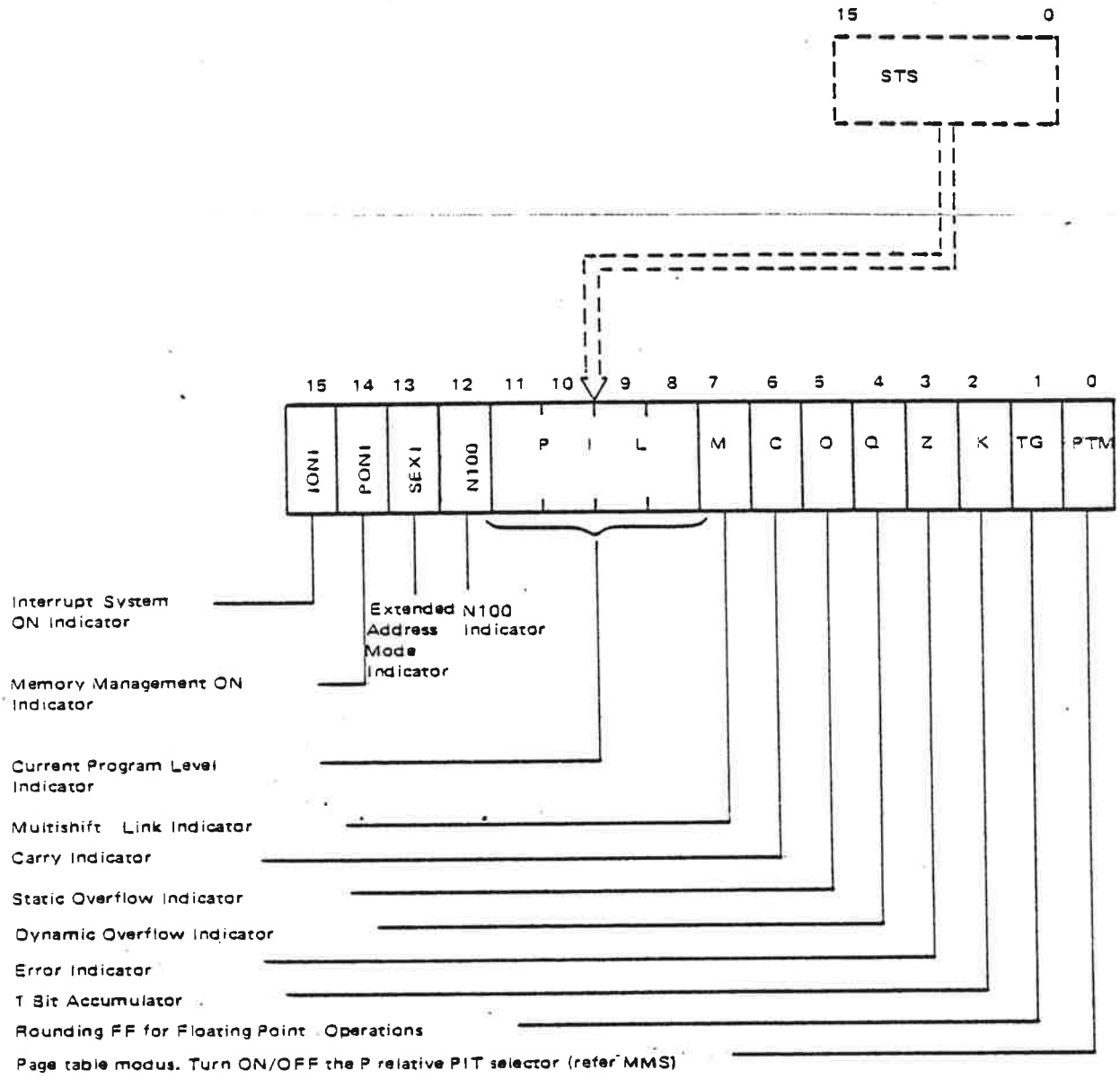


Figure 2.4: Status Register Assignment

The upper part (8 bits) is common for all program levels. This part gives us the following information:

IONI — Interrupt system ON indicator.

PONI — Memory management ON indicator.

SEXI — Extended indicator to show that MMS is in 24 bits extended addressing mode instead of the usual 19 bits addressing mode.

N100 — N100 indicator to tell the operating system that this is a ND-100 machine.

PIL — Current program level indicator.

2.2 *THE INTERRUPT SYSTEM*

2.2.1 *General*

The ND-100 interrupt system is designed to simplify programming and to allow high efficiency multiprogramming.

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

There are 16 program levels in ND-100 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, K, PTM and TG. There are also 9 registers that are only accessible from microprogram.

The context switching from one program level to another is completely automatic and requires only 5.0 μs; including the saving and unsaving of all registers and indicators.

The arrangement of the 16 program levels is as follows.

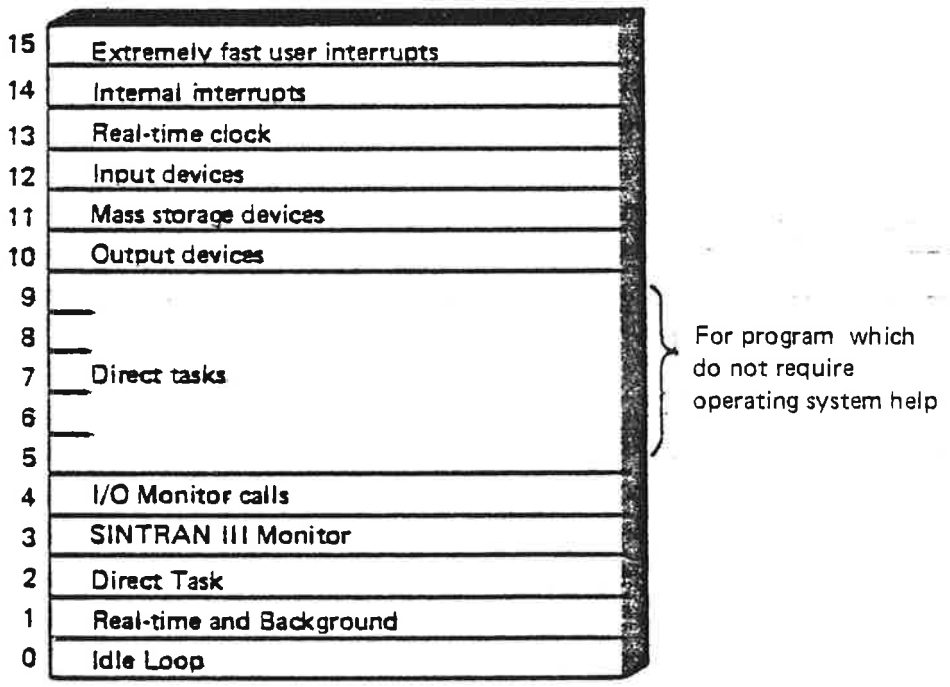


Figure 2.5: Level Assignments

The priority increases, program level 15 having the highest priority, program level 0, the lowest.

All program levels may be activated by software. In addition, the levels 10, 11, 12 and 13 may be activated by 512 external I/O interrupts. An IDENT instruction is used to identify the interrupting device. Program level 14 is used by the internal interrupt system, which monitors error conditions or traps in the CPU. Program level 15 may only have one I/O interrupt source.

Program level 15 is not used by standard ND equipment or software, but is available for users who need an immediate access to the CPU.

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.

For both internal hardware status interrupts and external interrupts there is an automatic priority identification mechanism which provides fast interrupt source detection.

2.2.2 *Functional Description*

Figure 2.6 shows the functional operation for the complete priority interrupt system.

There is one bit for each level in a detect register with 10 sources to cause a program level 14 interrupt, i.e., an internal interrupt. The detect register for program levels 0-9 are implemented in firmware which means that the micro-program takes care of the detection of interrupts on these levels.

The mask register is used to enable/disable the different program levels and conditions which may cause an internal interrupt. Program levels 0-9 are also taken care of by the microprogram.

When an interrupt comes, these two registers are ANDed together via an AND gate and the priority encoder gives a level value corresponding to the highest bit set in both the detect and mask registers.

This level indicator is compared with the current level to check if the new level is higher than the current one. If this is true, and the interrupt system is on, an interrupt will be generated.

The implementation of the ND-100 interrupt system is based on two registers: the detect register and the mask register. In both the detect and mask registers each interrupt level is assigned a bit position.

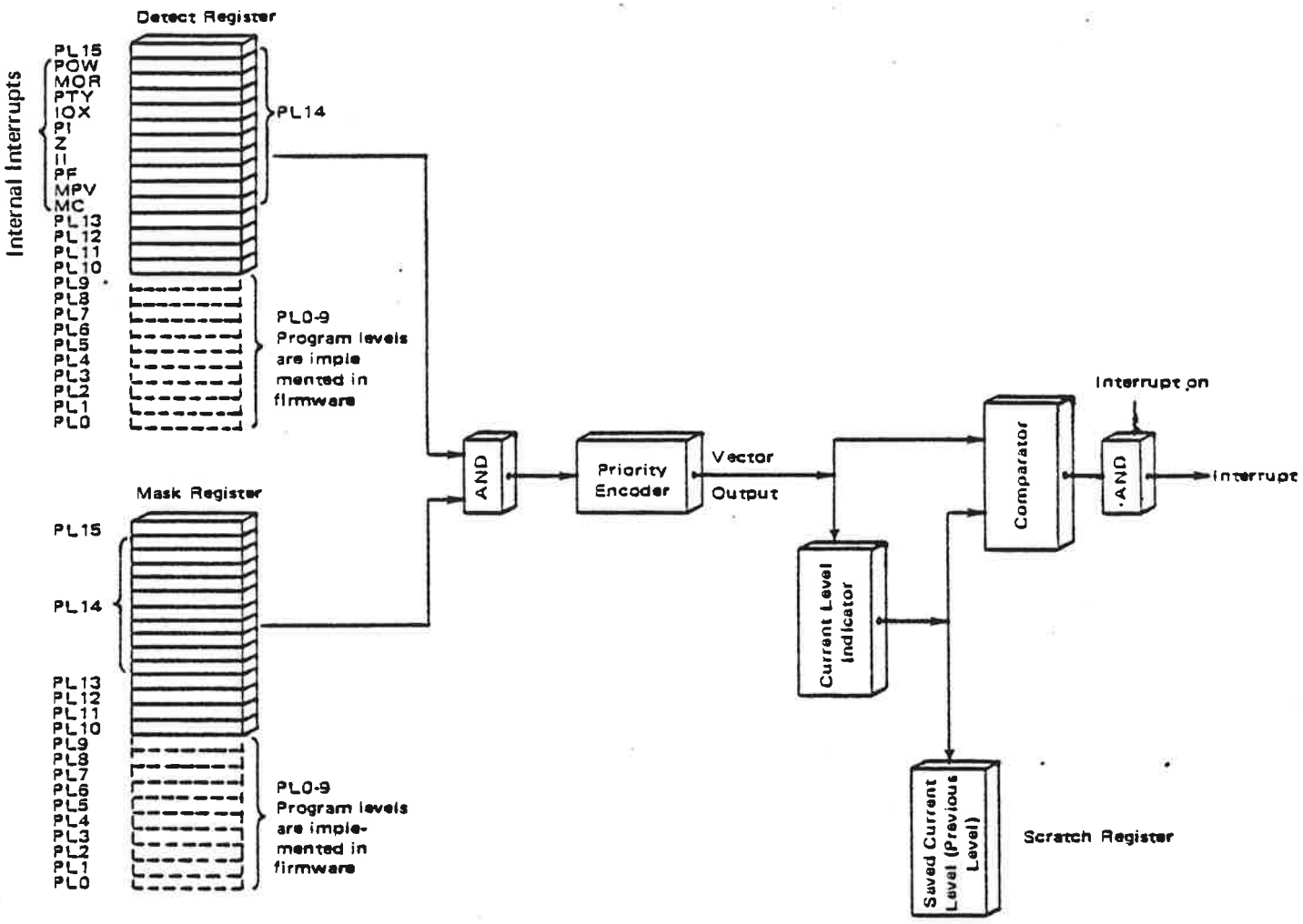


Figure 2.6: Priority Interrupt System

2.2.3 *The External Interrupt System*

Figure 2.7 gives a block diagram presentation of the external interrupt system.

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 mechanism for this is as follows.

The current program level is PIL (0 - 15). The 4 bit PIL register controls which register set (context block) to use.

The PIL number is constantly compared to a 4 bit code, PK. PK always contains the number of the highest program level which has its corresponding bits set in both PIE and PID. Whenever PK is unlike PIL, an automatic change of context block will take place through a short microprogram sequence.

The CPU will not ask for the next machine instruction but enter a microprogram that will change the program level to the PK. However, before the level change takes place, the program counter will be saved. The level change can be illustrated as follows:

1. The interrupt system is temporarily blocked to prevent false interrupts.

2. The program counter (CP) is copied to the saved program counter (P) on the current level.

3. The PIL (program level) register is copied into the PVL (previous program level) register.

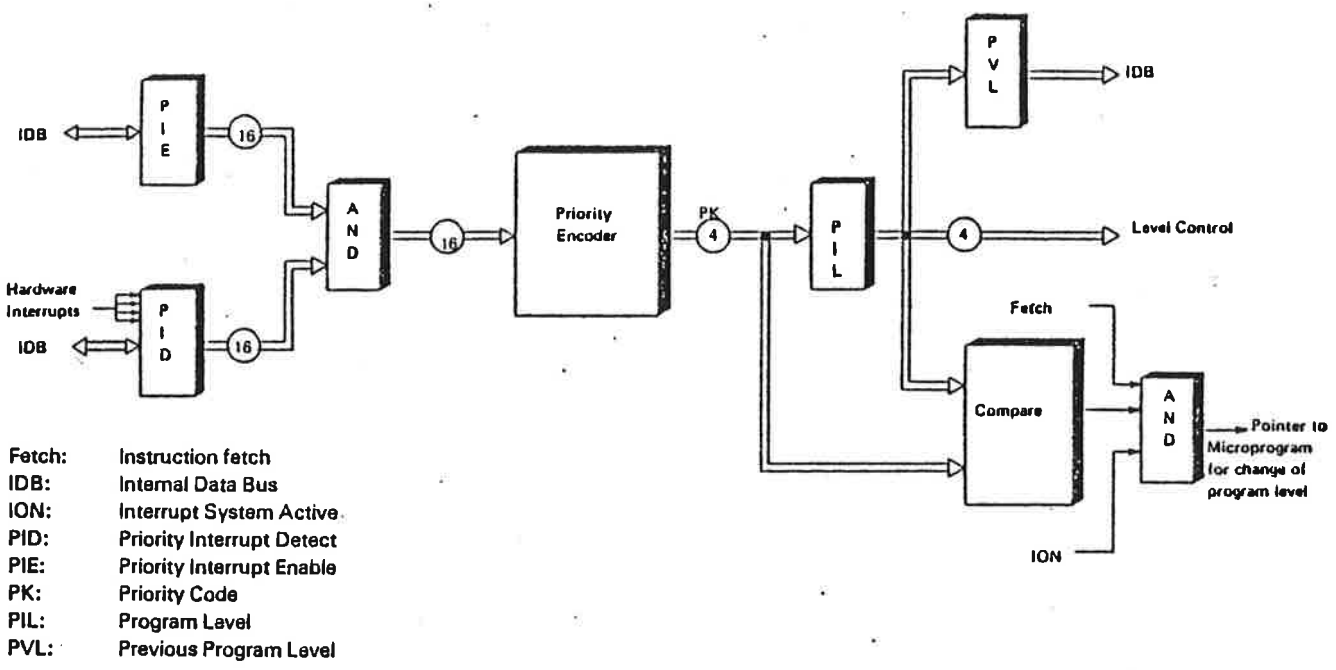
4. The PK (new level priority code) register is copied into the PIL (program level) register. (The CPU has, at this moment, changed level.)

5. The P (saved program counter) on the new level is copied to the CP (current program counter).

6. A fetch is issued, i.e., the first machine instruction on the new level is asked for.

This complete sequence requires only 5.0 μs from the completion of the instruction currently working when the interrupt took place, until the first instruction is started on the new level with its new set of registers and status.

External interrupts may set PID bits 15, 13, 12, 11, 10, and internal hardware status may set PID bit 14, because all internal interrupts are connected to this level.

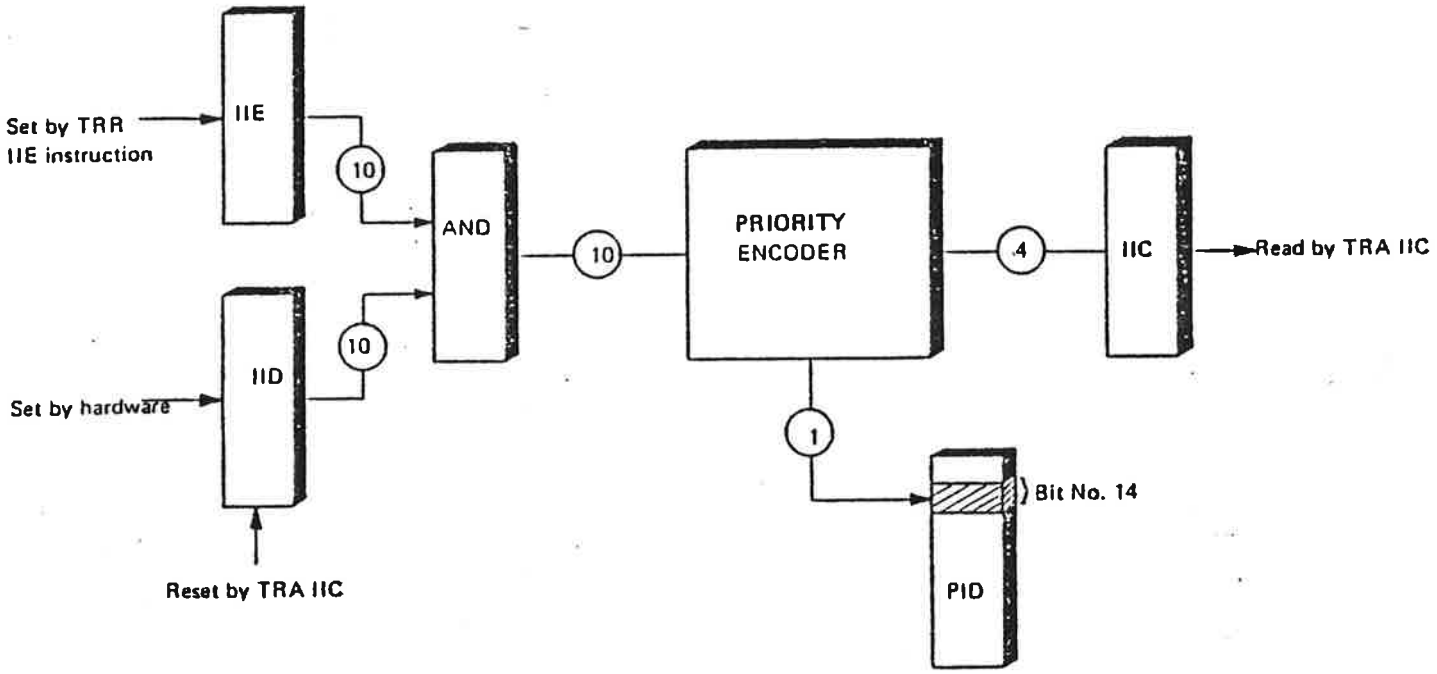


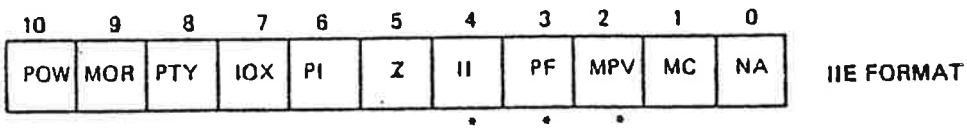
Fetch: Instruction fetch
IDB: Internal Data Bus
ION: Interrupt System Active
PID: Priority Interrupt Detect
PIE: Priority Interrupt Enable
PK: Priority Code
PIL: Program Level
PVL: Previous Program Level

Figure 2.7: External Interrup System

2.2.4 *The Internal Interrupt System*

The functional operation of the internal interrupt system is basically the same as the external one. Refer to Figure 2.8.





*Interrupts any micro-instruction.

IIC: Internal Interrupt Code
IID: Internal Interrupt Detect
IIE: Internal Interrupt Enable
TRR HE: Transfers the Content of the A-Register Into the IIE Register
TRA IIC: Transfers the Content of the IIC Register into the A-Register.

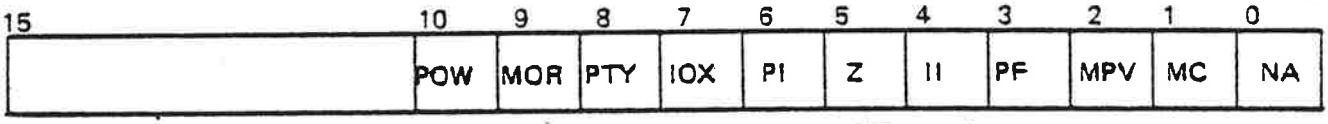
Figure 2.8: Internal Interrupt System, Block Diagram

2.2.4.1 The IIC and IIE Registers

As previously mentioned, the internal interrupt system is connected to level 14. Any internal interrupt condition will force the CPU to level 14. On this level the operating system will read the IIC — Internal Interrupt Code register. This register will hold a code between 0 - 12_8 which will identify the internal source for the interrupt.

Internal hardware status interrupts are individually enabled by an 11 bit register called IIE, Internal Interrupt Enable. IIE is set by the TRR IIE instruction. See Figure 2.8.

The internal hardware status interrupts are assigned to the IIE register in the following way:



The internal conditions which may cause internal interrupts and their associated vectors, the internal interrupt codes, are listed below:

	Bit No.:	IIC Code:	Cause
n/a	0	0	Not assigned
MC	1	1.	Monitor call
PV	2	2	Protect Violation. Page number is found in the Paging Status Register.
PF	3	3	Page fault. Page not in memory.
II	4	4	Illegal instruction. Not implemented instruction.
Z	5	5	Error indicator. The Z indicator is set.
PI	6	6	Privileged instruction
IOX	7	7	IOX error. No answer from external device.
PTY	8	10	Memory parity error
MOR	9	11	Memory out of range Addressing non-existent memory.
POW	10	12	Power fail interrupt
	11 - 15		Not assigned

2.2.4.2 Internal Hardware Status Interrupts

Monitor Call Interrupt

One of the internal interrupt sources is the monitor call instruction named MON. The monitor call instruction differs from the other internal interrupt sources in that the monitor call code or number is found in the T register on level 14.

The MON instruction may have up to 377_8 different codes (8 lower bits in the MON instruction) and the T_{14} register will be equal to this code with sign extention (bit 7 is sign).

Protect Violation Interrupt

A protect violation has occurred. Two types of violations are possible:

- Memory protect violation.

  This means that an illegal reference (read, write, fetch or indirect) has been attempted.

- Ring violation.

  This means that a program attempted to access an area with higher ring status.

Details regarding this interrupt are found in the paging status register.

Page Fault Interrupt

The program attempted to reference a page that is presently not in memory. Information regarding page number, etc. is found in the Paging Status register.

Illegal Instruction Interrupt

Attempted execution of an instruction that is not implemented causes this interrupt.

Error Indicator Interrupt

The Z indicator in the STS register has been set. This may be caused by several instructions:

- FDV with 0.0 (FDV = divide floating accumulator).
- EXR of an EXR instruction (EXR = execute register).
- DNZ overflow (DNZ = denormalize).
- RDIV overflow (RDIV = integer inter - register divide).
- Programmed setting of Z (BSET = bit set, MST = masked set or TRR = transfer to register).

The instructions are described in further detail in Section 3.

Note: Level 14 must always reset the Z indicator on the offending level, otherwise, a new interrupt will occur when the level is reentered.

Privileged Instruction Interrupt

Attempted execution of a privileged instruction causes this interrupt. The privileged instructions are listed below.

ION, IOF, PON, POF, PION, PIOF, WAIT, IOX, IOXT, IDENT, TRA, TRR, MCL, MST, LRB, SRB, IRR, IRW, SEX, REX, DEPO, EXAM, LWCS, OPCOM.

These instructions are described in further detail in Section 3.3.

IOX Error Interrupt

The addressed input/output device does not return a BDRY (Bus Data Ready) signal. This may be due to a malfunctioning or missing device or no device answering to an IDENT instruction.

Memory Parity Error Interrupt

A memory parity error has occurred. The least significant 16 bits of the failing address can be read from the PEA register using the TRA PEA instruction. PEA = Parity Error Address.

Further information may be read from the PES register (Parity Error Status).

Memory Out of Range Interrupt

This interrupt occurs when the program addresses nonexisting memory. The least significant 16 bits of the referenced address can be read from the PEA register.

Further information may be read from the PES register.

Power Fail Interrupt

This interrupt is triggered by the power sense unit. It is possible for this interrupt to occur simultaneously with some other internal interrupt. In this case, the power fail interrupt has priority.

2.2.4.3 Reset of the IIC Register

In order to optimize the processing of internal hardware status interrupts, the instruction TRA IIC will return the contents of IIC to the A register, bits 0-3, with bits 4 - 15 zero.

The instruction TRA IIC will automatically reset IIC.

Note that if the interrupt is caused by the error indicator Z, the Z indicator on that program level must be cleared by program control from program level 14. (Otherwise, another interrupt will occur.)

2.2.5 *Programming Control of the Interrupt System*

2.2.5.1 Programming the PID and PIE Registers

PID = Priority Interrupt Detect.
PIE = Priority Interrupt Enable.

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.

1. TRR PID and TRR PIE

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

2. 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.)

3. 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.

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

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

2.2.5.2 The WAIT, ION and IOF Instruction

The resetting of PID bits is also controlled by the WAIT instruction, which will reset PID on the 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 zeroed. This will cause a new program level to be entered and PK becomes different from PIL (PIL = 14, PK < 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 IIE, PIE and PIL, and the register set on program level zero will be used.

The ION instruction will continue 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 on 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 PIL 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 problems.

2.2.5.3 The Previous Level Register, PVL

In some cases after being forced to level 14_{10} it may be useful to know which level was the last one.

This might be the case when a MPV (Memory Protect Violation) has occurred. In this case one wishes to find the value of the SP (Saved Program) counter on the offending level and/or the offending instruction.

The PVL register holds the previous level information, and this could be read by the TRA PVL instruction.

2.2.5.4 Vectored Interrupts and the IDENT Instructions

In ND-100 there may be up to 2048 vectored interrupts. Usually, 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 using these levels 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, an IDENT instruction is used to identify the interrupt, since several devices may have interrupts on the same level. The instruction has the following format:

IDENT <program level>

When an IDENT instruction is executed, a hardware search on the indicated level is performed. The first interrupting device found will respond with its identification code and reset its interrupt condition.

The CPU will use the identification code (vector) as a branch address to the driver for the interrupting device.

If more than one device on the same level generates interrupts, the device interface located closest to the CPU has highest priority. If there is more than one device connected to the module, an internal priority on the module will determine which is to be treated first.

Programming Example:

```
LEV13,   WAIT              % Give up priority
         SAA      0        % Set content of A-reg. to 0
         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
```

2.2.6 *Initializing of the interrupt System*

Before use of the interrupt system it must be initialized. After switching power on, IIE, PIE and PIL will be zero. The registers on level zero will be in use. The interrupt initialization must include the following:

1. Enabling of the desired program levels by proper mask setting in PIE (Priority Interrupt Enable).

2. Enabling of the desired internal interrupt sources by proper mask setting in IIE — Internal Interrupt Enable register.

3. The P, saved program counters, on the levels to be used must be initialized, i.e., they must all point to the program to be executed on the different levels.

4. If the Z (error) indicator is enabled for interrupt (IIE bit number 5), care should be taken that this indicator is cleared in the status register (bit number 3) for all levels being initialized.

5. The IIC (Internal Interrupt Code) register, the PES (Parity Error Status) register and the PEA (Parity Error Address) register might be blocked after power up.

   By performing a TRA instruction for IIC and PES, all three registers will be unblocked and ready for use.

6. The interrupt system is turned ON.

Example:

```
LDA     (76032      % Enable for interrupts on level
TRR     PIE         % 1, 3, 4, 10, 11, 12, 13, and 14
LDA     (3736       % Enable for all internal
TRR     IIE         % Interrupt sources except for the Z indicator
LDA     (P1         % The saved program counters
IRW     10  DP      % on the enabled levels
LDA     (P3         % start value
IRW     30  DP      %
etc. for each P
in use
TRA     IIC         % Unlock IIC
TRA     PEA         % Unlock PEA and PES
ION                 % Turn on interrupt system
JMP     START       % Go to main program
```

2.3 *THE MEMORY MANAGEMENT SYSTEM*

2.3.1 *General*

The Memory Management System is designed to extend the ND-100 physical address space, and to provide a sophisticated memory and privileged instruction protection system. This system may be used for several purposes, such as:

- Dynamic memory allocation (paging).
- Program relocation.
- Expanding the maximum physical address space size to 16 M words.
- Memory protection of each individual page.
- Privileged instructions and ring structured program protection.

The Memory Management System includes two major subsystems:

- The paging system.
- The memory protection system.

The Paging System can work in two modes:

- Normal mode. A 16 bit virtual address is mapped into a 19 bit physical address. This extends the physical address space from 64 K to 512 K words. Four page tables of 64 entries each are used. This mode is compatible with the NORD-10/S.

- Extended mode. A 16 bit virtual address is mapped into a 24 bit physical address. This extends the physical address space from 64 K to 16 M words. Four page tables of 64 entries each are used.

For each mode the four page tables are located in high speed registers, directly connected to the internal data bus (IDB). This reduces paging overhead to practically zero. The page size is 1024 words.

The Memory Protection System may be divided into two subsystems:

- The page protection system.
- The ring protection system.

The page protection system protects each page from read, write or instruction fetch accesses or any combination of these.

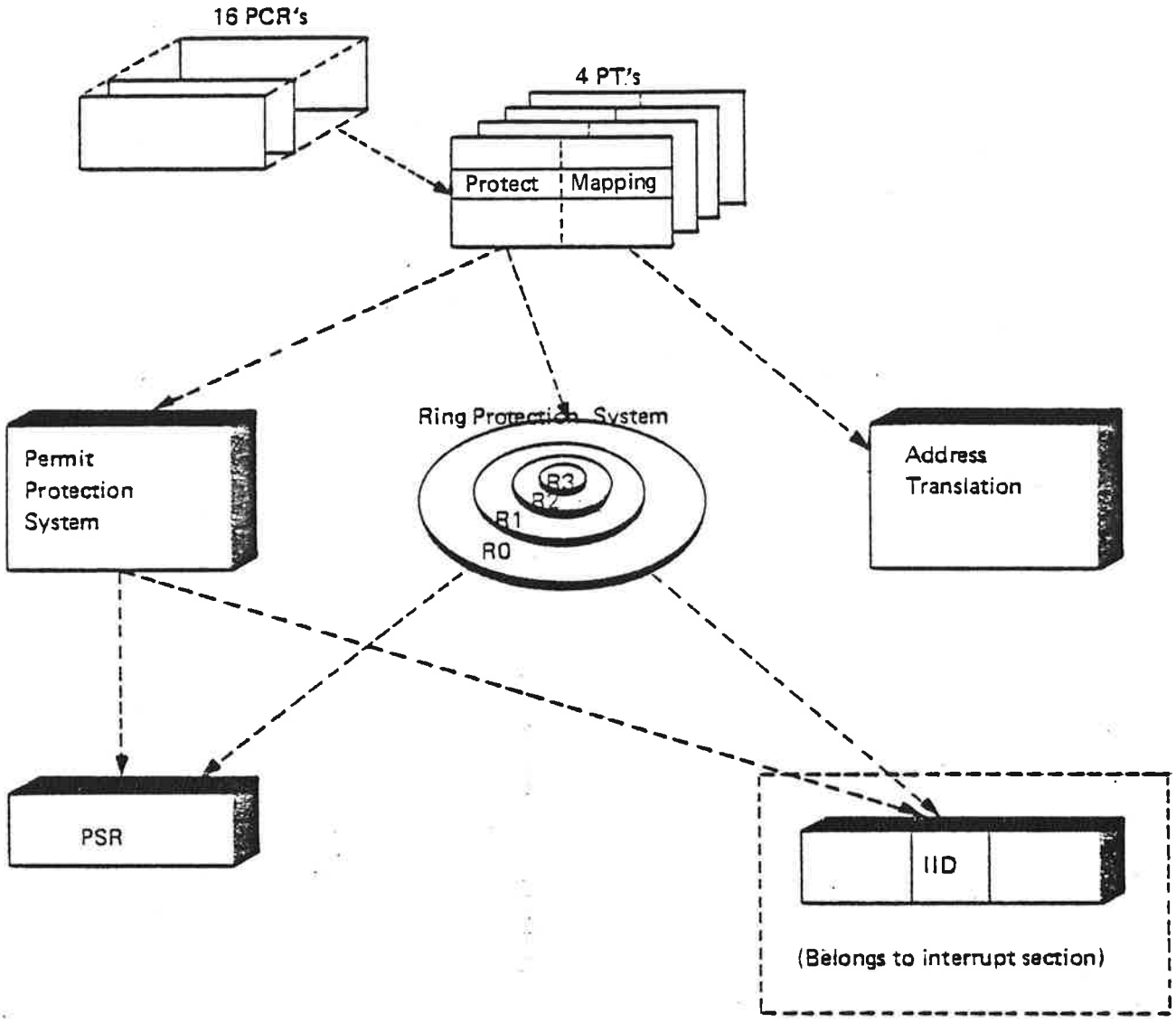
The ring protection system places each page on one of four priority rings. A page of memory that is placed on one specific ring may not be accessed by a program that resides in a page on a ring of lower priority. This system is used to protect system programs from user programs, the operating system from its subsystems, and the system kernel from the operating system.

2.3.2 *Memory Management Architecture*

Memory Management consists of:

- 4 page tables.
- 16 paging control registers.
- A paging status register.
- A permit protection system.
- A ring protection system.

The page size is fixed to 1K words, thus each page table will map the full 64K virtual address space of the ND-100.



IID: Internal Interrupt Detect
PCR: Paging Control Register
PSR: Paging Status Register
PT: Page Table

Figure 2.9: Memory Management Building Blocks

2.3.3 *The Paging System*

Number in parenthesis is valid for extended mode.
The Paging System is an automatic address interpretation system which maps a 16 bit virtual address, as seen from the program, into a 19 (24) bit physical address. This implies that the maximum memory size may be extended from 64K words to 512K (16 M) words. The system also allows programs to be written for 64K virtual memory with only parts of the program residing in physical memory at a given time, the rest being kept on mass storage.

The Paging System divides the memory into memory blocks or pages of 1024 words or 1K words. The pointers to these pages are found in the page tables. In ND-100, there are four page tables, each consisting of 64 entries, and each covering a full 64K address space. The tables are kept in high speed registers with a 32 bit word length.

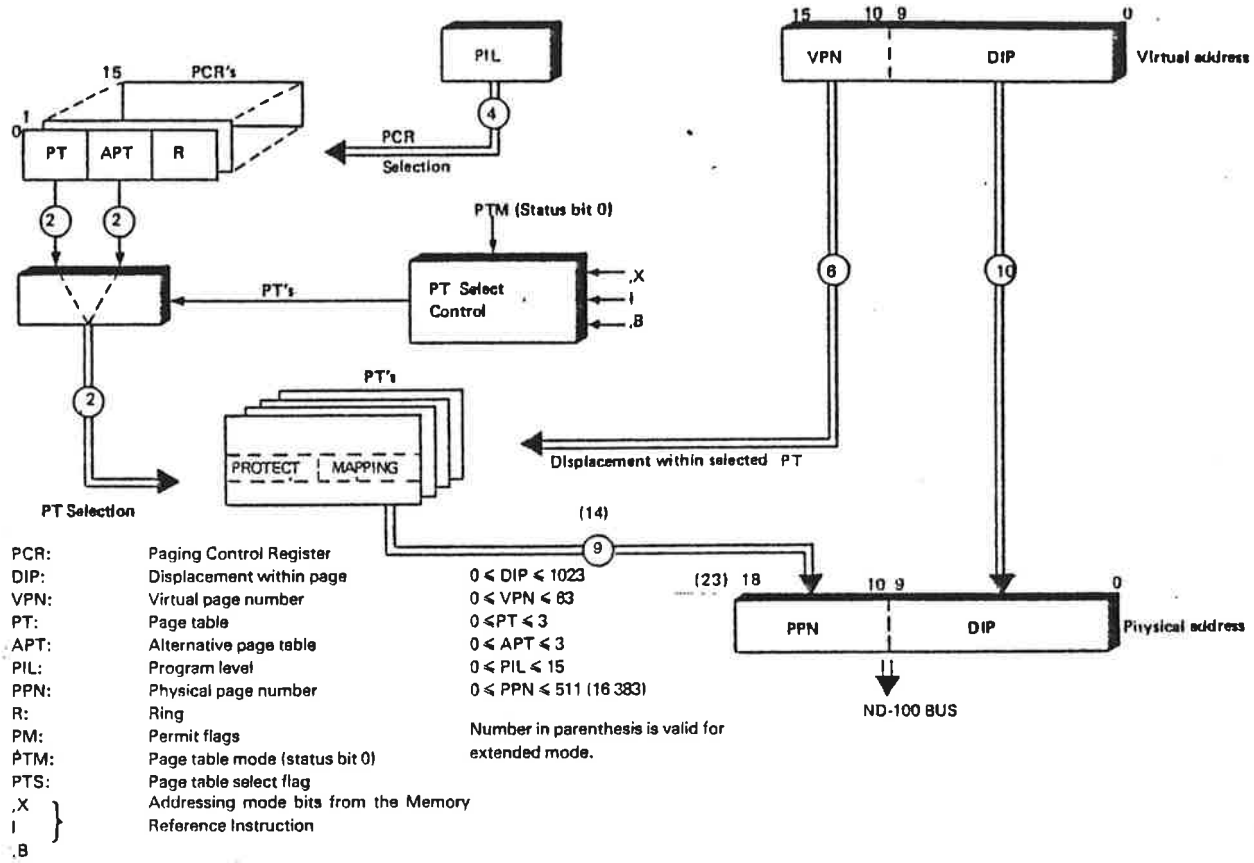
ND-100 uses 1K words per page. This implies that in order to map 64K words of virtual address space, 64 Page Table (PT) entries are required.

To address any location within a 1K address space, 10 address bits are required. These bits are the displacement within a page (DIP), and are transferred directly to the ND-100 bus. The most significant part of the virtual address (bits 10 - 15) are used as an address selecting one of 64 entries in PT. This address is referred to as Virtual Page Number (VPN).

The program level (PL) determines which paging control register (PCR) to use. The selected PCR determines which page index table to select, and VPN addresses an entry in the selected PT.

When a memory request is performed, the content of the 32 bits PT is looked up. 7 bits are used for protection, and are discussed later. 9 (14) bits are called Physical Page Number (PPN), and are transferred to the ND-100 bus. PPN can have values from 0 - 512 (16384). This makes it possible to access 512 (16384) pages. Since one page = 1024 words, it is possible to access
512 (16 384) x 1024 512 K (16 M) words.

Prior to program start, the operating system will set the PPN to the proper value in the PT. The address translation is therefore under control of the operating system.



PCR:	Paging Control Register	
DIP:	Displacement within page	0 ≤ DIP ≤ 1023
VPN:	Virtual page number	0 ≤ VPN ≤ 63
PT:	Page table	0 ≤ PT ≤ 3
APT:	Alternative page table	0 ≤ APT ≤ 3
PIL:	Program level	0 ≤ PIL ≤ 15
PPN:	Physical page number	0 ≤ PPN ≤ 511 (16 383)
R:	Ring	
PM:	Permit flags	
PTM:	Page table mode (status bit 0)	
PTS:	Page table select flag	
,X I ,B	Addressing mode bits from the Memory Reference Instruction	

Number in parenthesis is valid for extended mode.

Figure 2.10: Virtual to Physical Address Mapping

2.3.4 *The Shadow Memory*

The shadow memory is a number of reserved memory addresses. These memory addresses are used to access the page tables in the same way as the rest of the memory.

These reserved addresses are called shadow memory because it lies in the shadow of the main memory and is inaccessible for users on rings 0, 1 and 2. For ring 3 users or when paging is off however, main memory lies in the shadow and is inaccessible. Figure 2.11 shows the shadow memory layout.

The topmost locations in the 64 K virtual address space are reserved for page table access. In normal mode 1 x 64 x 4 = 256 locations are needed and in extended mode 2 x 64 x 4 = 512 locations are needed. The following octal addresses are hence reserved:

	Normal Mode:	Extended Mode:
Page table 0	177400 - 177477	177000 - 177177
Page table 1	177500 - 177577	177200 - 177377
Page table 2	177600 - 177677	177400 - 177577
Page table 3	177000 - 177777	177600 - 177777

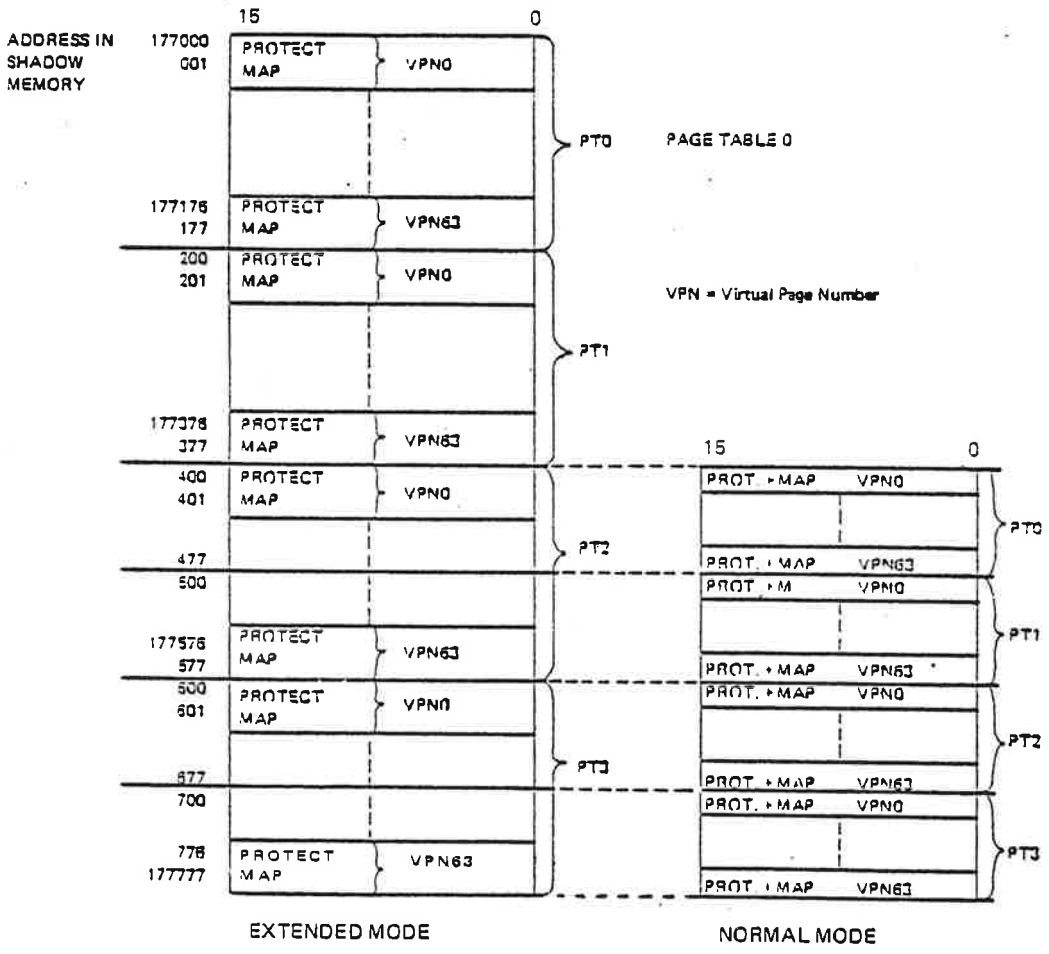
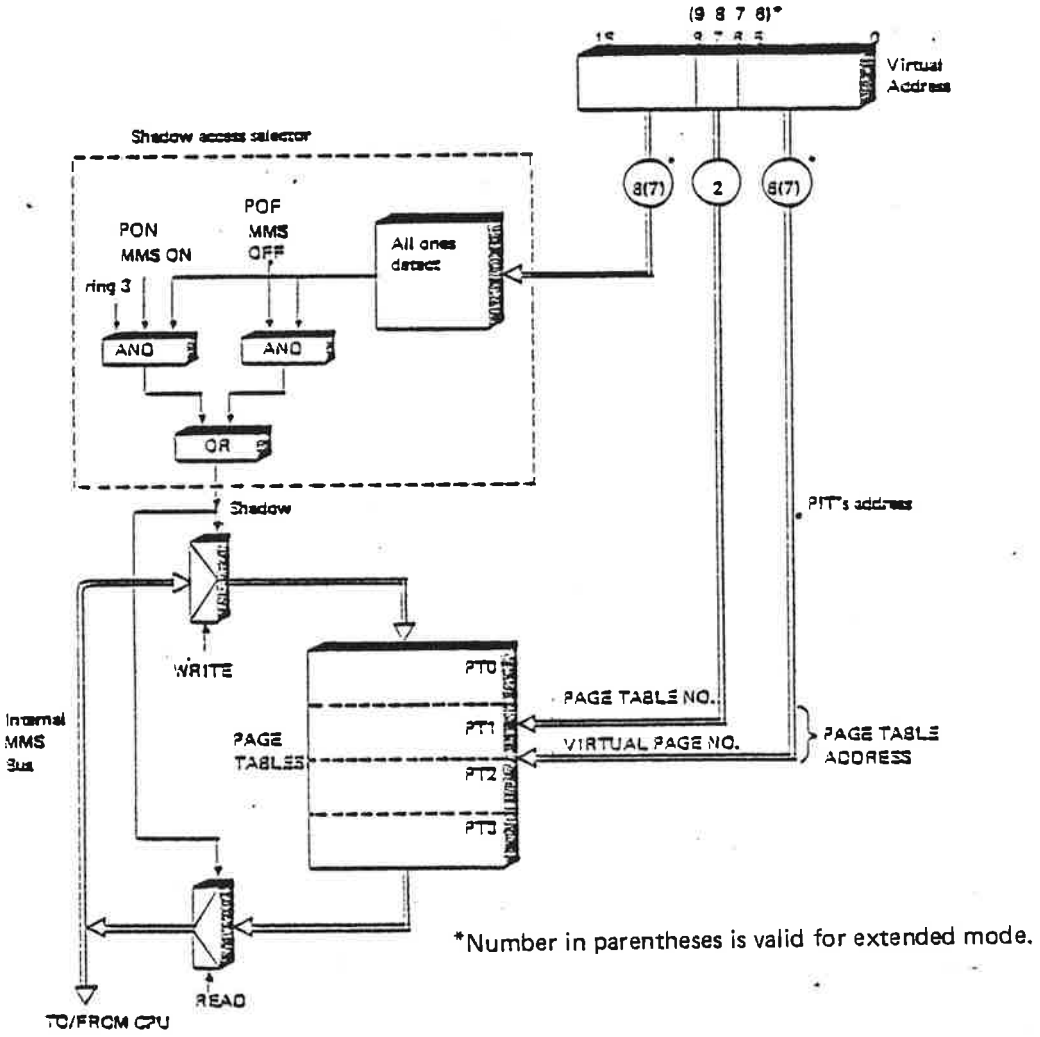


Figure 2.11: Shadow Memory Layout.

In normal mode only 16 of each page table entry's 32 bits are used. Therefore only one shadow memory word is read/written to fill one page table entry.

In extended mode 21 of the 32 page table entry's bits are used. This means that two shadow memory words are read/written to fill one page table entry.



*Number in parentheses is valid for extended mode.

Figure 2.12: Shadow Memory Addressing.

2.3.5 *The Page Tables*

In normal mode the map part requires 9 bits and the protect part requires 7 bits. Together the map and protect parts require 16 bits, which is the PT's 16 bit word length. The 9 PPN bits (Physical Page Number) in the map entry shown in figure 2.13 are used to select one of 512 pages in the memory.

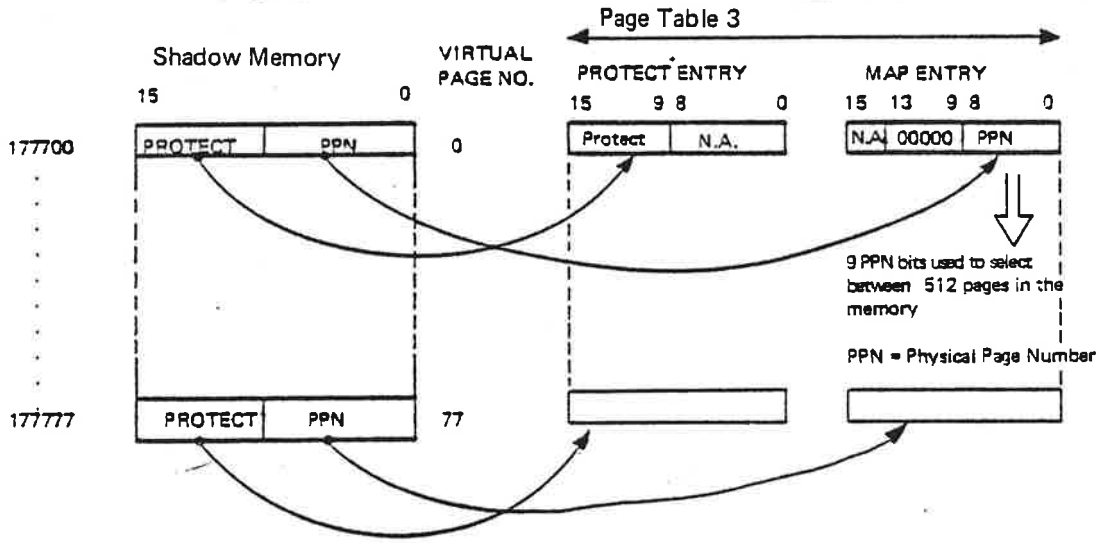


Figure 2.13: Reading Page Table 3 Entries as seen from Program in Extended Mode

In extended mode the map part requires 14 bits and the protect part requires 7 bits. Together the map and protect parts make 21 bits, which extend the PT's word length. Therefore we have to use two shadow memory locations for housing the map and protect parts. The 14 PPN bits in the map entry shown in figure 2.14 are used to select one of 16 384 pages in the memory.

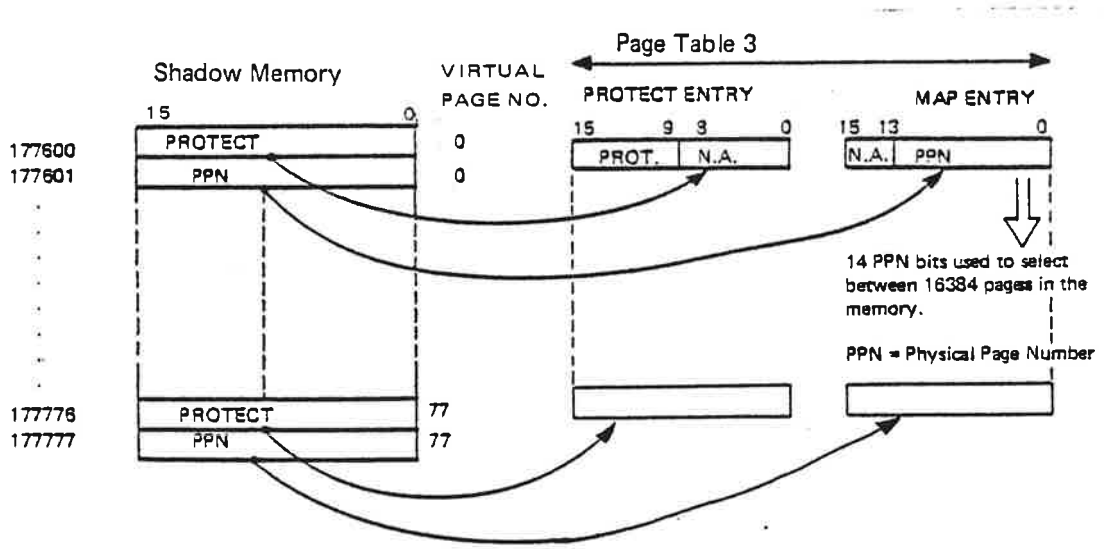
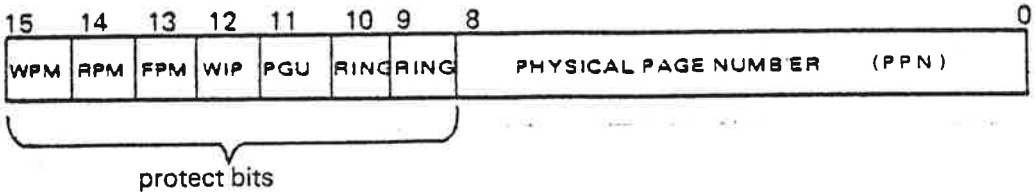


Figure 2.14: Reading Page Table 3 Entries as seen from Program in Extended Mode

The page table format:

In normal mode each entry has the following format:



Bits 13 - 15: Memory protection bits (WPM = Write Permitted, RPM = Read Permitted, FPM = Fetch Permitted).

Bit 12: Written in page (WIP)
This bit is automatically set by hardware.

Bit 11: Page used (PGU)
This bit is automatically set by hardware.

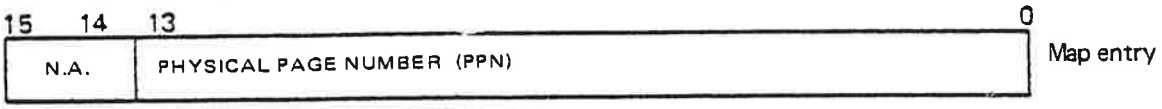
Bits 9 - 10: Ring bits
These bits decide which ring this page belongs to.

Bits 0 - 8: Physical page number
Nine bits addresses a maximum of 512 physical pages in memory.

The protect bits and the protection system are described in Section 2.3.5.

In extended mode the protect bits and the PPN bits require two entries, which have the following formats:

protect bits are identical in extended and normal mode



Bits 0 - 13: 14 bits address a maximum of 16 384 pages in memory.

2.3.5.1 Page Used and Written in Page

All entries in a page table are under program control only, except for the two bits PGU and WIP, which are also controlled automatically by the Memory Management System.

Bit 12: WIP - Written in Page

If this bit is set, the page has been written in, and it should be written back to mass storage. If it is zero, the page has not been modified and need not be rewritten. This bit is automatically set to one the first time a write occurs and then remains set. It is cleared by program (whenever a new page is brought from mass storage).

Bit 11: PGU - Page Used

If PGU = 1, the page has been used. The bit is automatically set whenever the page is accessed and it remains set. The bit is cleared by program. This bit may be used in operating systems to determine which page should be swapped.

2.3.5.2 Page Table Selection

ND-100 has 4 page tables. Which one to be used is selected by the Paging Control Register on the current program level. In PCR the information is either taken from the PT field or the APT field. One is to be selected. The alternative page table is used if the memory reference is *not* P relative and status bit 0 (PTM) is 1. The table below will help explain.

Addressing Mode				Address Mapping with PTM = 1	
,X	I	,B	Mnemonic	Via PT	Via APT
0	0	0		(P) + disp.	—
0	1	0	I	(P) + disp.	((P) + disp.)
0	0	1	,B	—	(B) + disp.
0	1	1	,B I	—	(B) + disp.; ((B) + disp.)
1	0	0	,X	—	(X) + disp.
1	0	1	,B ,X	—	(B) + (X) + disp.
1	1	0	I ,X	(P) + disp.	((P) + disp.) + (X)
1	1	1	,B I ,X	—	(B) + disp.; ((B) + disp.) + (X)

Note that indirect addressing involves 2 memory references where one or both go via the APT, as shown in the table.

Page Table Selection

The main principle is that all P relative memory references are mapped via PT and all other references are mapped via APT. This feature is used only by processes which require access to two segments with different virtual address spaces and gives one process access to 128K of virtual memory.

2.3.6 *Memory Protection System*

The memory management system employs two memory protection systems: a permit protection system and a ring protection system. The two systems together constitute an extensive memory protection, i.e., complete protection of system from user and user from user.

The memory protection system works on 1K pages. If a memory access violates any of the protection systems, an interrupt to program level 14 will occur with the internal interrupt code equal to 2 = MPV (memory protect violations).

2.3.6.1 Page Protection System

The page protection system is a protection system for each individual page of memory. Each individual page may be protected against:

- Read access.
- Write access.
- Instruction fetch access.

and any combination of these. Thus, there are 8 modes of memory protection for each page.

The read, write and fetch protect system is implemented by defining in bits 13 - 15 of the PT how the page may be used. In hardware, this information is compared with the instruction being executed, i.e., if it is load (read), store (write), instruction fetch or indirect address.

The three bits from the PT have the following meanings.

Bit 15: WPM — Write Permitted

WPM = 0. It is impossible to write into locations in this page regardless of the ring bits.

WPM = 1. Locations in this page may be written into if the ring bits allow it.

If an attempt is made to write into a write protected page, an internal interrupt to program level 14 will occur, and no writing will take place.

Bit 14: RPM — Read Permitted

RPM = 0. Locations in this page may not be read (they may be executed).

RPM = 1. Locations in this page may be read if the ring bits allow it.

If an attempt is made to read from a read protected page, an internal interrupt to program level 14 will occur.

Bit 13: FPM — Fetch Permitted

FPM = 0. Locations in this page may not be executed as instructions.

FPM = 1. Locations in this page may be used as instructions.

If an attempt is made to execute in fetch protected memory, an internal interrupt to program level 14 will occur and the execution is not started.

Indirect addresses may be taken both from pages which have FPM = 1 and from page which have RPM = 1.

All combinations of WPM, RPM and FPM are permitted. However, the combination where WPM, RPM and FPM are all zero is interpreted as *page not in memory* and will generate an internal interrupt with internal interrupt code, IIC, equal to page fault.

2.3.6.2 Ring Protection System

The ring protection system is a combined privileged instruction and memory protection system, where 64K virtual address space is divided into four different classes of programs or rings. Two bits (9 and 10) in each page table entry are used to specify which ring the page belongs to.

The ring bits have the following meaning:

Bit 10	Bit 9	
0	0	Ring 0: Programs executing from this page may not execute privileged instructions. The program may only access locations in ring zero. Locations outside ring 0 are completely inaccessible.
0	1	Ring 1: Programs executing from this page may not execute privileged instructions. The program may access locations in ring 1 and ring 0.
1	0	Ring 2: All instructions are permitted when executed from this page. The program may access locations in rings 2, 1 and 0.
1	1	Ring 3: All instructions are permitted and the whole address space is accessible if not protected by the RPM, WPM and FPM bits. The page tables may be accessed.

An illegal ring access or illegal use of privileged instructions will cause an internal hardware status interrupt to program level 14 and the instruction which caused the interrupt will not be executed.

The recommended way of using the ring bits is as follows:

Ring 0: User programs

Ring 1: Compilers, assemblers, data base systems

Ring 2: Operating system, File system, I/O system

Ring 3: Kernel of operating system

Associated with the ring bits in a PT entry are the two ring bits in the current program levels paging control register (PCR).

Before a program can start executing, the PCR on the relevant program level is loaded by the operating system with information about which PT, alternative PT and ring is to be used. The program's PT must also be loaded by the operating system prior to execution.

The ring bits of the appropriate PCR are compared with the ring bits of the appropriate page table entry. The PCR ring bits should always be greater than or equal to the PT ring bits. If not, an internal interrupt (MPV) will be generated.

The user's ring number is defined in the PCR-register, while the program's ring number is defined in the page tables.

Example:

If a user on ring no. 3 starts executing a program on ring no. 1, he is allowed to do so. However, he is forced to user ring no. 1 after program execution. Note that this happens only when executing programs on lower rings than the user's ring number. This does not happen when reading or writing operands on a lower ring.

One should note that the two protection systems are independent of each other and that both the individual memory protection mode and the ring mode must be satisfied before an operation is performed.

2.3.7 *Privileged Instructions*

In a multiuser multitask system, a user is not permitted to use all instructions in the instruction set. Some instructions may only be used by the operating system, and this category of instructions are called ***privileged instructions.***

Privileged Instructions:

— Input/output instructions

— All instructions which control the memory management and interrupt system

— Interprogram level communication instructions

Refer to the instruction repertoire for further information.

The only instruction the user has available for user/system communication is the monitor call Instruction — MON. The MON instruction may have up to 256 different parameters or calls. When the machine executes the MON instruction, it generates an internal interrupt.

The privileged instructions may only be executed on ring 2 and 3, i.e., only by the operating system. If programs on ring 0 and 1 try to execute any privileged instructions, a privileged instruction interrupt will be generated and the instruction will not be executed.

2.3.8 *Memory Management Control and Status*

2.3.8.1 The PON and POF Instructions

The memory management system is controlled by the two privileged instructions PON and POF.

PON — Turn on memory management system (paging on)

The instruction that is executed after the PON instruction will go through the address mapping (paging) mechanism, and the memory protection system will be active.

POF — Turn off memory management system (paging off)

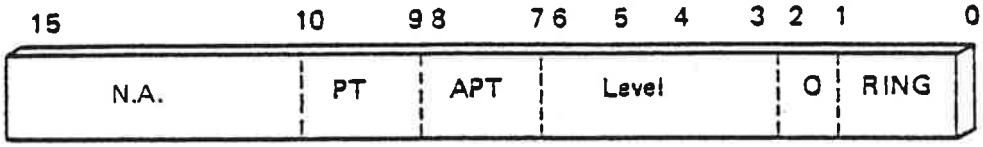
The instruction will turn off the memory management system and the next instruction will be taken from a physical address in the lower 64K, the address following the POF instruction.

The machine will then be in an unrestricted mode without any hardware protection feature, i.e., all instructions are legal and all memory "available".

2.3.8.2 Paging Control Registers

There is one PCR (paging control register) for each level. The setting of the PCRs is done by the operating system prior to the program execution. Only one PCR may be written into at a time by the instruction TRR PCR.

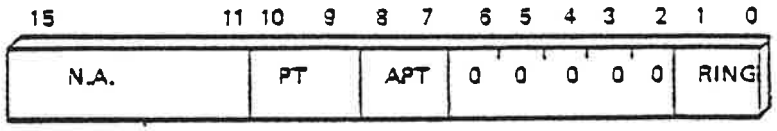
This instruction uses the contents of the A register. The A register has the following format:



Bits 11 - 15: Not assigned
Bits 9 - 10: Page table number (0-3)
Bits 7 - 8: Alternative page table number (0-3)
Bits 3 - 6: Program level (PCR number) (0-15)
Bit 2: Equals zero
Bits 0 - 1: Ring number (0-3)

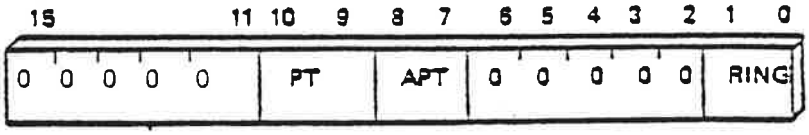
Transferring the A-reg. to the PCR:

The instruction TRR PCR transfers the A-reg. to the PCR. After executing this instruction, PCR has the following format.



Transferring the PCR to the A-reg:

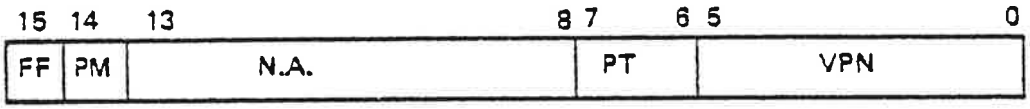
For maintenace purposes it may be desirable to read back the content of the 16 PCRs to the A-register. This is done by executing the TRA PCR instruction. After executing this instruction, the A-register has the following format.



2.3.8.3 Paging Status Register

Whenever the memory management system reports any errors (page fault, memory protection violations), the operating system is alerted through an internal interrupt with the interrupt code equal to the error source. Next, the operating system will read the paging status register for further information. The paging status register is used for further specifications when a page fault or a memory protection violation occurs.
The instruction TRA PSR is used to read this register. Errors lock the PSR register, TRA PSR unlocks it again.

The bits in PSR have the following meaning:



PSR Format

Bit 15: FF = Fetch Fault.

Memory management interrupt occurred during an instruction fetch.

Bit 14: PM = Permit violation.

1 = permit violation interrupt (read, write, fetch protect system).

0 = ring protection violation interrupt.

Permit violation has priority if both conditions occur.

Bits 6-7: PT = Page Table.

Page table number.

Bits 0-5: VPN = Virtual Page Number.

Virtual page number.

Note that bits 0 - 7 are the 8 least significant bits of the physical page table entry in normal mode.

If bit 15 is a one, the page fault or protection violation occurred during the fetch of an instruction. In this case, the P register has not been incremented and the instruction causing the violation (and the restart point) is found from the P register on the program level which caused the interrupt.

If bit 15 is zero, the page fault or protection violation occurred during the data cycles of an instruction. In this case, the P register points to the instruction after the instruction causing the internal hardware status interrupt. When the cause of the internal hardware status interrupt has been removed, the restart point will be found by subtracting one from the P register.

2.3.9 *The SEX and REX Instructions*

The address mode for the page mapping system is controlled by the two privileged instructions SEX and REX.

SEX — Set extended address mode

The SEX inatruction will set the paging system in a 24 bit address mode instead of a 19 bit address mode. A physical address space up to 16 M words will then be available.

Bit number 13 in the status register (STS) set to one, indicating the extended address mode.

REX — Reset extended address mode

The REX instruction will reset the extended address mode (24 bits) to normal address mode (19 bits). This implies that 512 K words of physical address space is now available.

Bit number 13 in the status register is reset, indicating normal address mode.

Note that after change of mode, the page tables must be initialized.

2.4 *ND-100 MEMORY SYSTEM*

2.4.1 *General*

Computer performance is, to a great extent given by the efficiency of the memory system. General requirements are:

- Low access time.
- Low storage cost.
- Large capacity.

These requirements are usually conflicting.

In the ND-100 system, a compromise is achieved through the implementation of a multilevel hierarchial memory system. Figure 2.15 shows the major building blocks in this system.

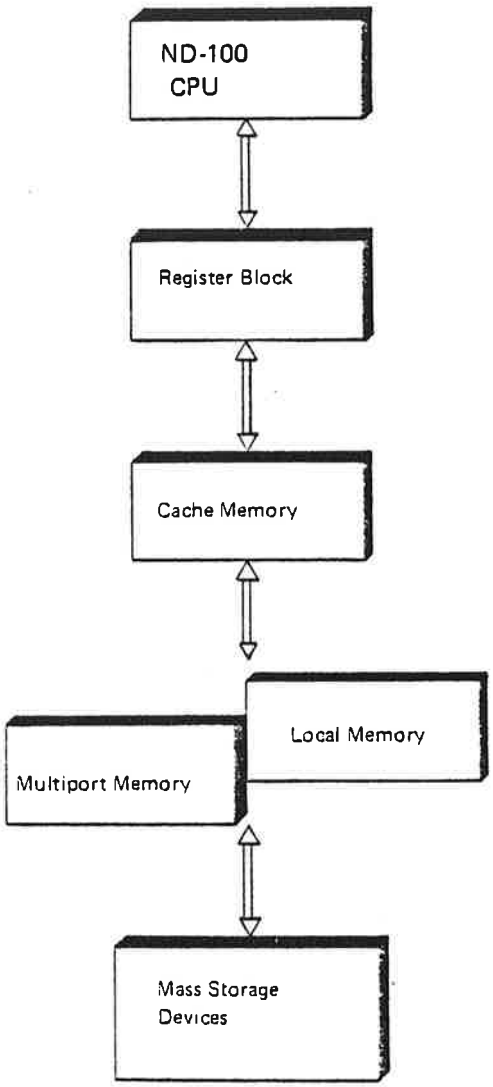


Figure 2.15: Multilevel Storage System

The concept is to hold the most frequently used information as near the CPU as possible. In other words, the average access time for instructions and data should be close to main memory access time. At the same time, most of the information resides on mass storage. That is, price per stored bit approaches the mass storage device cost.

The memory system includes (ordered by access time):

- 8 programmable registers associated to each program level.
- 1K words CACHE memory (optional)
- Up to 16M words local memory on each module.
- Up to 2M words multiport memory address space.
- Disk storage.

Here we will discuss local memory, multiport memory and cache.

Note! One ND-100 CPU can only access 512K words if used in normal address mode. That means the sum of local memory and multiport memory for one CPU cannot exceed 512K words.

If used in extended address mode ND-100 can access up to 16 words.

2.4.2 *ND-100 Memory Architecture*

Figure 2.16 shows the storage interconnection.

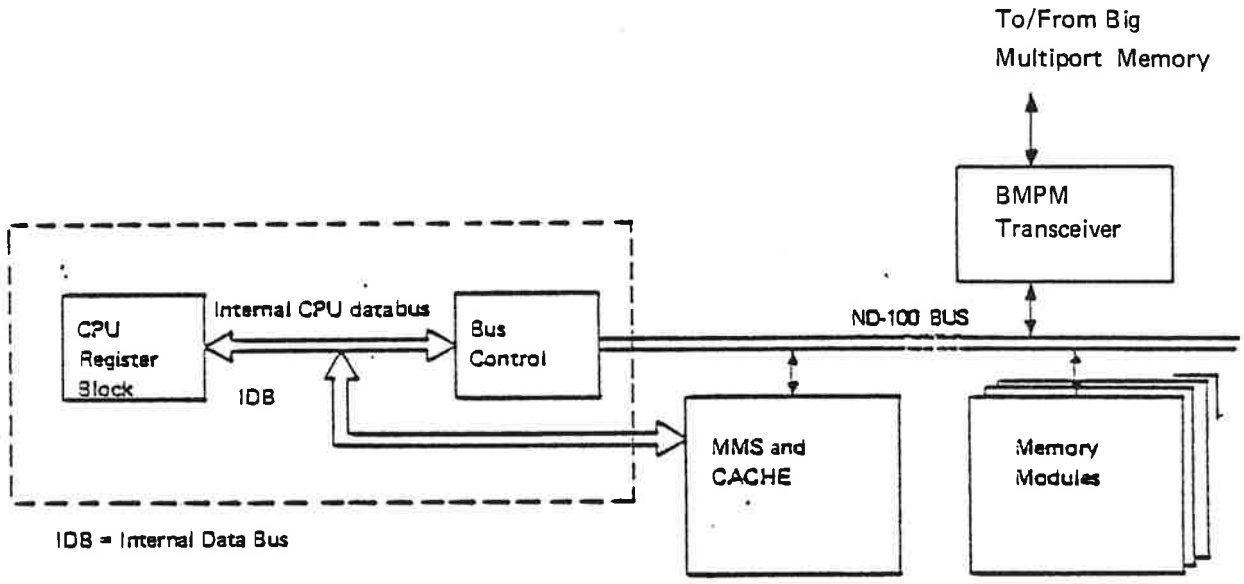


Figure 2.16: Storage Interconnection

CACHE memory is physically located on the memory management module and connected directly to the internal CPU data bus (IDB).

Local memory may consist of several modules plugged directly into the ND-100 bus.

Multiport memory is accessed through a Big Multiport Memory (BMPM) transceiver in the ND-100 bus connected to one port in a separate card crate.

2.4.2.1 Local (Main) Memory

Local memory facts:

- Memory size from 32K words in steps of 32K words up to 512K words (normal address mode), 16M words (extended address mode).
- 32K, 64K or 256K words per memory module.
- Direct connection to ND-100 bus for low access time.
  Typical: 320 ns measured on CPU bus control.
- Error correction of single bit failures and reporting double bit failures.

2.4.2.2 Memory Module Placement in ND-100 Bus

Memory modules should be placed in the right-most position (position 12 or 21) in the ND-100 bus and expanded to the left.

Module address range may be defined in two different ways:

- Prewired position code in each bus slot.
- Thumbwheel setting of a module address area.

2.4.2.3 The Position Code

The position code defines a module placed in position 12 to have the address range 0 - 64K words, 64K words to 128K words in position 11 and so on. In other words, there is a resolution of 64K words per position, expanding to the left.

2.4.2.4 The Thumbwheel Setting

It is possible to mix module sizes of 16K words, 32K words and 64K words in the same memory system. In this case the position code can not be used.

The thumbwheel setting allows an address resolution of 16K words per position and should be used in cases where module sizes are mixed.

The thumbwheels are physically located at the top of the memory module and define lower address limit for each module.

The module itself knows its size, which is added to the lower limit and presented on a display giving lower limit to the next module.

Examples:

General

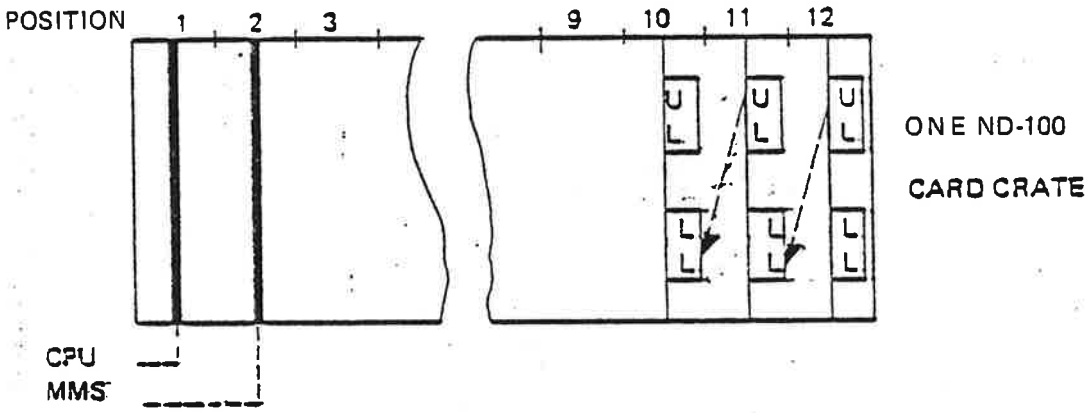


Figure 2.17: Memory Module placement in the Card Crate

LL: Lower limit is set by two hexadecimal thumbwheels or given by module placement (the position code). Lower limit defines the lower address to access the module.

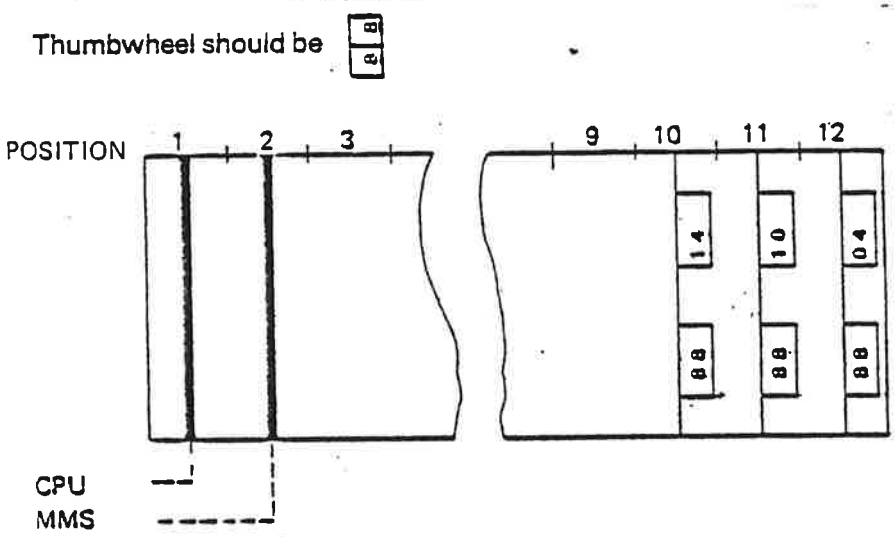
UL: Upper limit is displayed as two octal digits and defines the highest address to access the memory module. Upper limit is generated internally on the memory module as an addition of lower limit and the size of the memory module. The upper limit is displayed in steps of 16K.

As indicated in the above figure, upper limit on a memory module covering one part of the address range, should be equal to lower limit on the next memory module covering the following higher addresses.

CASE 1:

Lower limit defined by the position code.

Thumbwheel should be

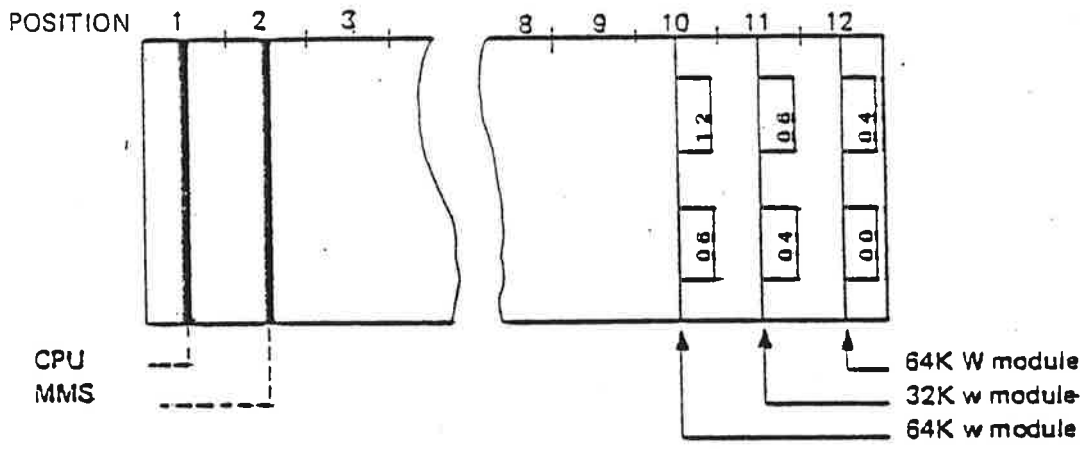


Position 12: Address range 0 - 64K words
Position 11: Address range 64 - 128K words
Position 10: Address range 128 - 192K words
and so on

Requirement: All memory modules must be 64K words.

CASE 2:

Lower limit defined by thumbwheel.



Resolution on thumbwheel is 16K words per digit. Only digits below 8 are legal.

2.4.3 *Memory Error Correction*

To each 16 bit word stored in memory, a 6 bit error correction code (ECC) is generated. That is, each word is stored as 22 bits.

When reading from memory, a new ECC is generated and compared with the stored one. This comparison allows the memory system to:

- Accept good data (no errors).
- Detect, correct and log single bit errors.
- Detect double bit errors and interrupt the CPU for uncorrectable memory failure.
- In most cases, interrupt the CPU for memory failures on multiple errors (certain unfortunate combinations of multiple errors could be bypassed).

Error Codes (PES bits 8-13) Decoding Table:

Error Code	Syndrome Bits Fatal S4	S3	S2	S1	S0	No Error	Single code Error	Single data Error	
0	0	0	0	0	0	Good			
1	0	0	0	0	1		EC0		
2	0	0	0	0	1	0		EC1	
3	0	0	0	0	1	1			E0
4	0	0	0	1	0	0		EC2	
5	0	0	0	1	0	1			E1
6	0	0	0	1	1	0			
7	0	0	0	1	1	1			E2
10	0	0	1	0	0	0		EC3	
11	0	0	1	0	0	1			E3
12	0	0	1	0	1	0			
13	0	0	1	0	1	1			E4
14	0	0	1	1	0	0			E5
15	0	0	1	1	0	1			E6
16	0	0	1	1	1	0			E7
17	0	0	1	1	1	1			
20	0	1	0	0	0	0		EC4	
21	0	1	0	0	0	1			E8
22	0	1	0	0	1	0			E9
23	0	1	0	0	1	1			E10
24	0	1	0	1	0	0			E11
25	0	1	0	1	0	1			
26	0	1	0	1	1	0			E12
27	0	1	0	1	1	1			
30	0	1	1	0	0	0			E13
31	0	1	1	0	0	1			
32	0	1	1	0	1	0			E14
33	0	1	1	0	1	1			
34	0	1	1	1	0	0			E15
35	0	1	1	1	0	1			
36	0	1	1	1	1	0			
37	0	1	1	1	1	1		EC5	
40	1	0	0	0	0	0			
"	"								
52	1	0	1	0	1	0	All 22 bits are zero		Special cases
65	1	1	0	1	0	1	All 22 bits are one		
"	"								
74	1	1	1	1	0	0			
75	1	1	1	1	0	1	Lower byte parity error		For 2 bit parity check memory
76	1	1	1	1	1	0	Upper byte parity error		
77	1	1	1	1	1	1	Upper + lower byte par. error		

Multiple Errors

Figure 2.18: Error Codes (EC) as Reported in the PES Register

2.4.4 *Memory Control and Status*

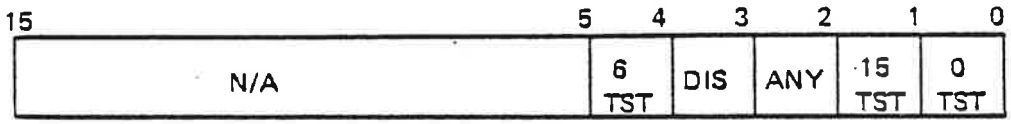
2.4.4.1 Error Correction Control Register (ECCR)

This register controls the error correction network.

The error correction control register is loaded by executing the instruction:

TRR ECCR

The format is as follows:



Description:

Bit 0, 1, 3 and 4 are used by maintenance only to test the error correction network.

Bit 0: set to "1" simulates memory error in bit 0. TST = Test

Bit 1: set to "1" simulates memory error in bit 15. TST = Test

Bit 2: interrupt condition control bit.

"0" = only multiple errors will generate parity error interrupt.
"1" = all errors will generate parity errors.

This bit is turned on and off by an RT program logging single bit errors.

Bit 3: Disable. (DIS)

When this bit is set, error correction and parity error interrupt are disabled.

Bit 4: Set to "1" simulates memory error in bit 6. TST = Test

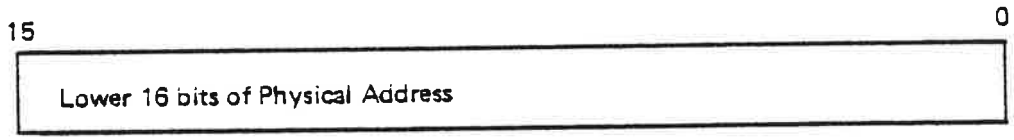
2.4.4.2 Memory Status Registers (PEA and PES)

Feedback information from the memory system is given in two status registers:

- PEA (Parity Error Address).
- PES (Parity Error Status).

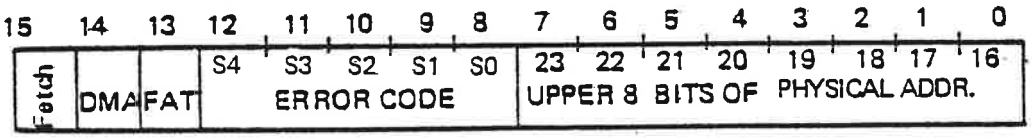
Both registers are read to the A register by the TRA instruction.

Format of PEA: (A register after TRA PEA)



A PEA register holds the 16 least significant address bits of the last memory reference.

Format of PES: (A register after TRA PES)



Bits 0-7: Most significant address bits of the last memory reference.

Bits 8-12: Error code (0-4) which points out the failing and corrected bit if a single bit error has occurred (see bit 13). Refer to the table below for decoding of the error code.

Bit 13: Fatal

If fatal is set 1, a multiple error has occurred and the error code does not contain relevant information. Fatal not set (''0'') means single bit error (bit number found in error code) or good data (error code = 0).

Bit 14: DMA; error occurred during DMA reference.

Bit 15: Fetch — error occurred during instruction fetch or during an examine (EXAM) or a deposit (DEPO) instruction.

When the error condition occurs, the content of PES and PEA is locked and not released until TRA PEA is executed. These registers do not contain correct information unless an internal interrupt with code 10 or 11 (parity error and memory out of range) is detected.

2.4.5 *Multiport Memory*

Two multiport memory systems are available. These two systems are called:

— Big Multiport Memory (BMPM).
— Multiport Memory 4 (MPM4).

2.4.5.1 Big Multiport Memory (BMPM)

ND-100 can be equipped with a multiport memory transceiver to access the big multiport memory system.

The BMPM system allows up to four sources to access the same physical memory area.

One source is connected to one of four BMPM ports through a multiport channel. All devices meeting the multiport channel specification are allowed access to this memory system.

Typical applications of the BMPM system are:

— Multiprocessor communication through a shared memory system.
— Shared memory between CPU and high speed DMA devices.

The BMPM system is physically located in a separate card crate.

One card crate can hold 384K words, and 8 crates can be connected.

2.4.5.2 Multiport Memory 4 (MPM4)

The MPM4 combines the features from the big multiport memory (BMPM) and the bus extender (BEX). The MPM4 extends the ND-100 bus to new card racks. In these racks you can install memory modules, DMA modules and ordinary I/O modules. The memory modules may be shared with other ND-100s, ND-500s and DMA devices. By using the MPM4 system you are able to build a big and flexible computer system.

2.4.6 *Cache Memory*

Cache is an optional high speed memory buffer.

The presence of cache will reduce average memory access time significantly.

2.4.6.1 Cache Memory Architecture

Location

Cache memory is physically located on the memory management module and has direct (through special wiring) connection to the internal CPU data bus (IDB).

Placement/Replacement Algorithm

The cache memory should hold the most recent data and instructions to be processed. The algorithm used for this purpose is called "Write Through" (WT).

This algorithm ensures that all information in cache is also held as backup in main memory. That is, cache memory does not need standby power during a power break.

The algorithm concept is as follows:

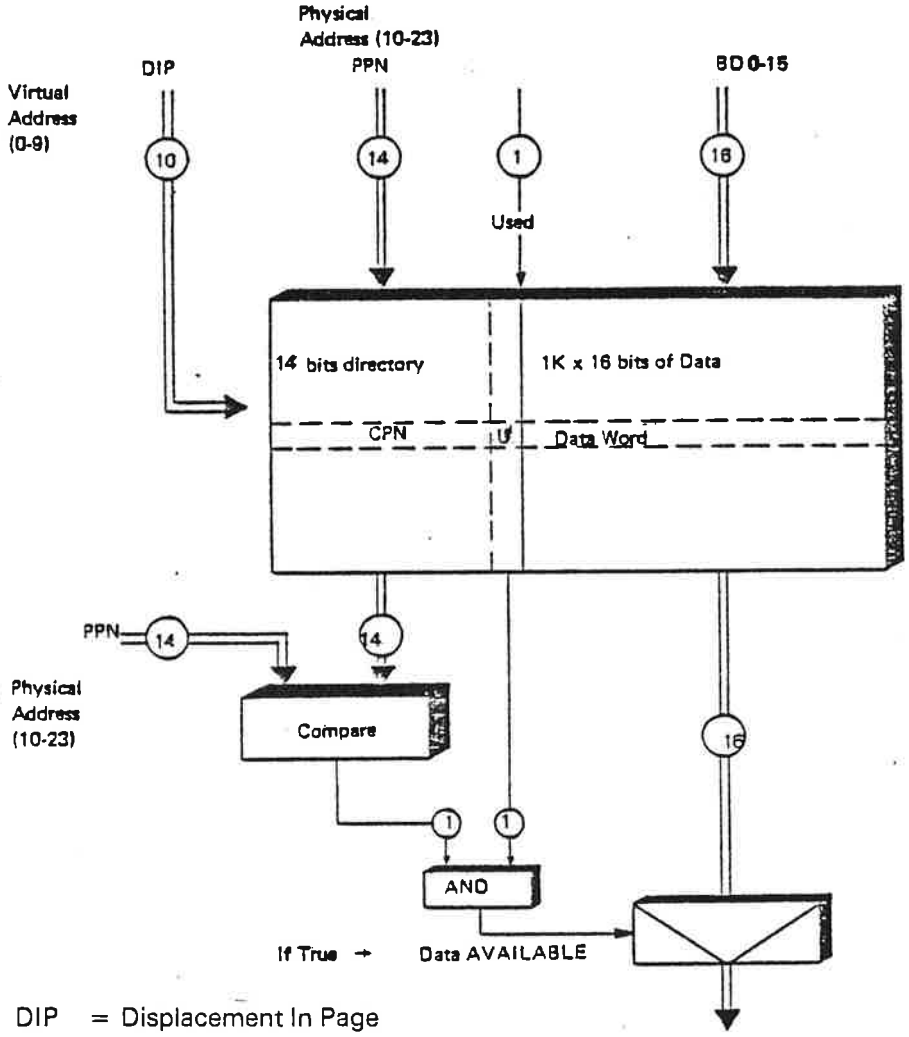
- A write operation goes to cache memory as well as main memory.
- During a read operation data is taken from cache memory if found there. Otherwise, it is taken from main memory and written into CPU and also into the cache memory (for probable later use).

2.4.6.2 Cache Memory Organization

The cache memory is organized as a 1K word, by 31 bit look up table. Each word in cache is a copy of a word on one of the physical pages in main memory and there is a one to one connection between displacement in cache and displacement in the page.

In order to associate each cache word with one physical page, a directory is used. The directory is 15 bits to each word telling which page this word belongs to. During write the directory is updated to the Physical Page Number (PPN) written into.

During read, the directory is compared with the accessed PPN. If they are equal, it was a cache hit, if not, the displacement was equal, but the cache word belongs to another page than the one accessed. Refer to Figure 2.19 for illustration.



DIP = Displacement In Page
PPN = Physical Page Number
BD = Bus Data
CPN = Cache Page Number
U = Used

Figure 2.19: Cache Operation Principles

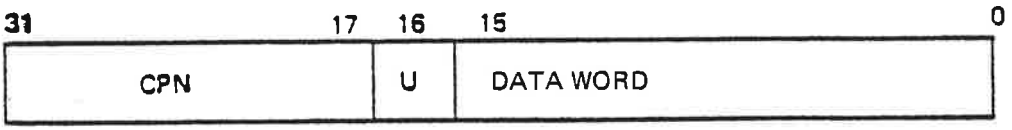


Figure 2.20: Format of One Cache Word

CPN: Cache Page Number defines what PPN (Physical Page Number) the CPU word belongs to.

U: "1" — this cache location contains valid information.

"0" — this cache location does not contain valid information.

The U bit is only used by hardware and will be "0" after a cache clear.

DATA WORD: This is a copy of a word in main memory.

2.4.6.3 Cache Control and Status

Cache memory contains:

- 3 registers for control
- 1 status register for feedback information

2.4.6.3.1 CACHE CONTROL

Clearing Cache

ND-100 cache concept requires that all changes in main memory should be updated in cache. This is done automatically when the CPU writes to memory. A DMA transfer will not be mapped through cache, however, so that a DMA transfer would result in different data in cache and memory. To avoid this, the operating system will execute the instruction

```
TRR CCLR        % Clear cache
```

when a DMA transfer is initiated.

Setting of Cache Inhibit Limits

Assume that all external sources to memory (DMA, etc.) could use a predefined address area.

Note that data is not *removed* from cache when the cache inhibit area is expanded. Therefore, expansion of the cache inhibit area must always be accompained by clear cache. Note that the whole address range is inhibited after master clear.

lower limit ⩽ PPN ⩽ upper limit

The limit setting is included to define a CPU private area, thus avoiding the clear cache operation for each DMA transfer.

The limit registers are set by the instructions:

```
LDA <lower limit>       % lower limit page number
TRR LCIL                % set lower limit
```

and

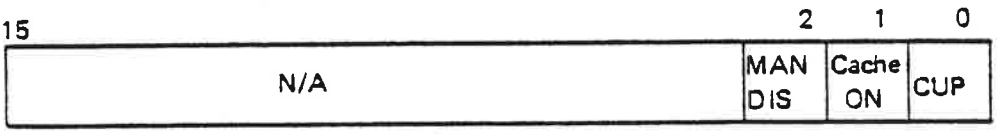
```
LDA <upper limit>       % upper limit page number
TRR UCIL                % set upper limit
```

2.4.6.3.2 CACHE STATUS REGISTER

The cache status register is used by diagnostic programs and loaded to the A register by

TRA CSR % Cache status → A register

The format of CSR:



Bit 0: CUP

Cache updated — CUP is "1" if the next memory reference (i.e., the instruction readout for the following TRA CSR) causes writing in cache. (Before TRA CSR is executed the next instruction is prefetched!)

Bit 1: CACHE ON

Cache on is "1" if cache is present, except during a 60 μs period, following cache clear and master clear. If bit 2, MAN DIS is "1"; cache on will be "0".

Bit 2: MAN DIS

Manual Disable of cache.

"1" if disabled
"0" if not disabled

This bit is controlled by a switch on the memory management system module.

The cache status register is 1XX if the cache option is not installed.

2.5 *ND-100 INPUT/OUTPUT SYSTEM*

2.5.1 *General*

The Input/Output system (abbreviated to I/O system) provides a two-way communication between the CPU and its peripherals. General requirements for an I/O system are:

- Reliability.
- Flexibility. The I/O system should be able to handle slow devices as well as high speed devices.
- Modularity. The I/O system should be easy to expand as the customer requires. I/O configuration should be easy to change.

The requirements mentioned above depend, of course, on the system's architecture.

2.5.2 *ND-100 I/O Architecture*

The ND-100 bus provides the communication betwen functional blocks in ND-100.

All ND-100 modules are made to a common standard to allow identical connection to this bus. This convention also includes I/O device controllers.

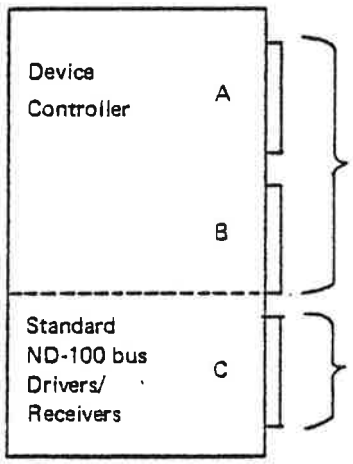
The ND-100 bus is controlled completely by the bus control/driver which is an integrated part of the CPU. This arrangement includes the following features:

- The I/O device controller is directly connected to the same printed back-plane as the CPU.
  - no external wiring
  - increased reliability
- There is no connection of external buses.
  - a faster system
  - easy to maintain
- I/O modules can be plugged into the bus.
  - easy to expand
  - easy to reconfigure

It is also possible to extend the ND-100 bus by using Bus Extenders (BEX).
The BEX system extends the ND-100 bus to a maximum of 8 card crates (both 12 and 21 position crates).
However, this system slows down the execution time.

2.5.3 *ND-100 Card Crate — Physical Layout*

The ND-100 card crate is available in two versions. One version takes a maximum of 12 modules and the other a maximum of 21 modules. Each module has one 96 pins contactor for direct contact to the ND-100 bus when plugged into the crate. Refer to Figure 2.21. Figure 2.22 and 2.23 show the layout of the two card crates.



Two 64 pins contactors for device connection.

96 pins contactor for direct contact to the ND-100 bus.

Figure 2.21: ND-100 Module and Connectors

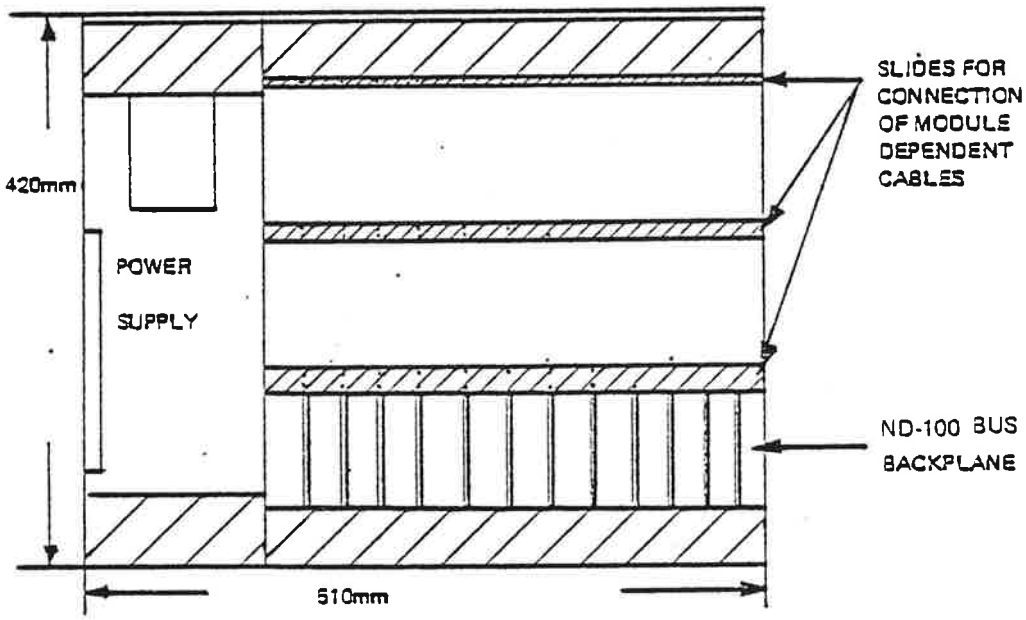


Figure 2.22: 12 Position ND-100 Crate Layout (Top View).
In the 12 position version, the required power is supplied by a power supply located within the card crate. This approach leads to a very compact system.

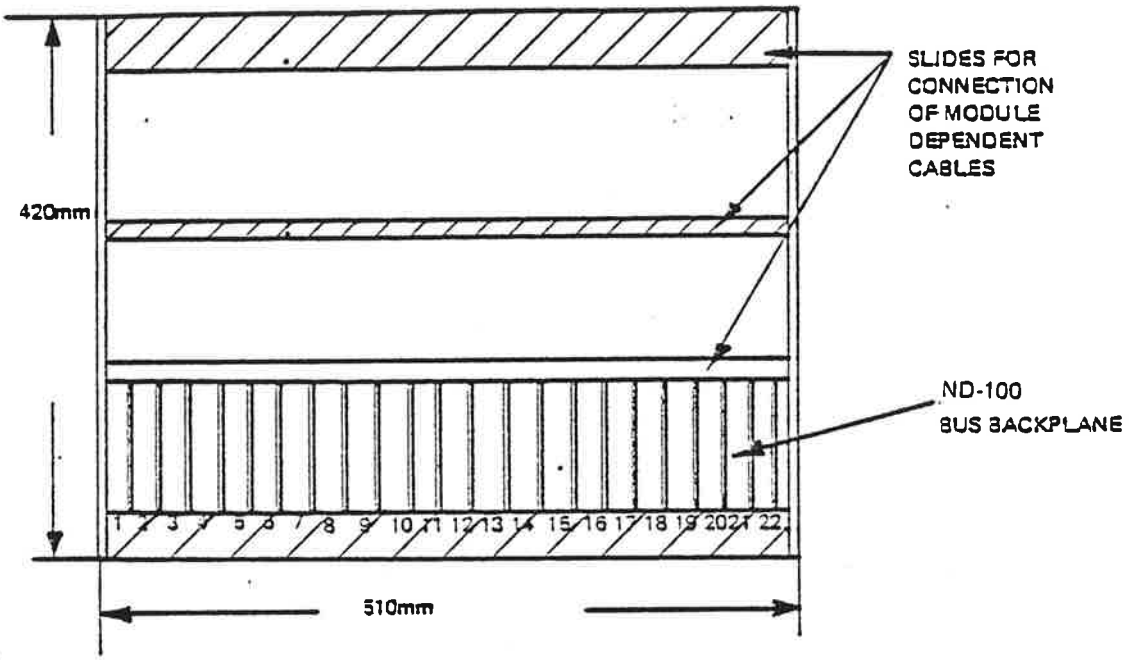


Figure 2.23: 21 Position ND-100 Crate Layout (Front View)
In the 21 position version, the power supply is removed from the card crate and located in the upper part of the cabinet. Thus, the cabinet is bigger than the cabinet for a 12 position crate.

Figure 2.24 shows the recommended placement of modules in a card crate. The placement rules are equal for both the 12 and 21 position crate.

If the memory management and cache module is present, the first I/O module should be placed in position 3, the next in position 4 and so on, expanding to the right.

If the MMS and cache module is *not* present, move all I/O modules one position left.

RULE: There should never be empty positions between the CPU and the last I/O module. Expansion is from left to right.

If the 12 or 21 position crates are not enough, new card crates can be added, thus expanding the ND-100 to a maximum of 8 crates. This is done by using Bus Extenders as described later in this manual.

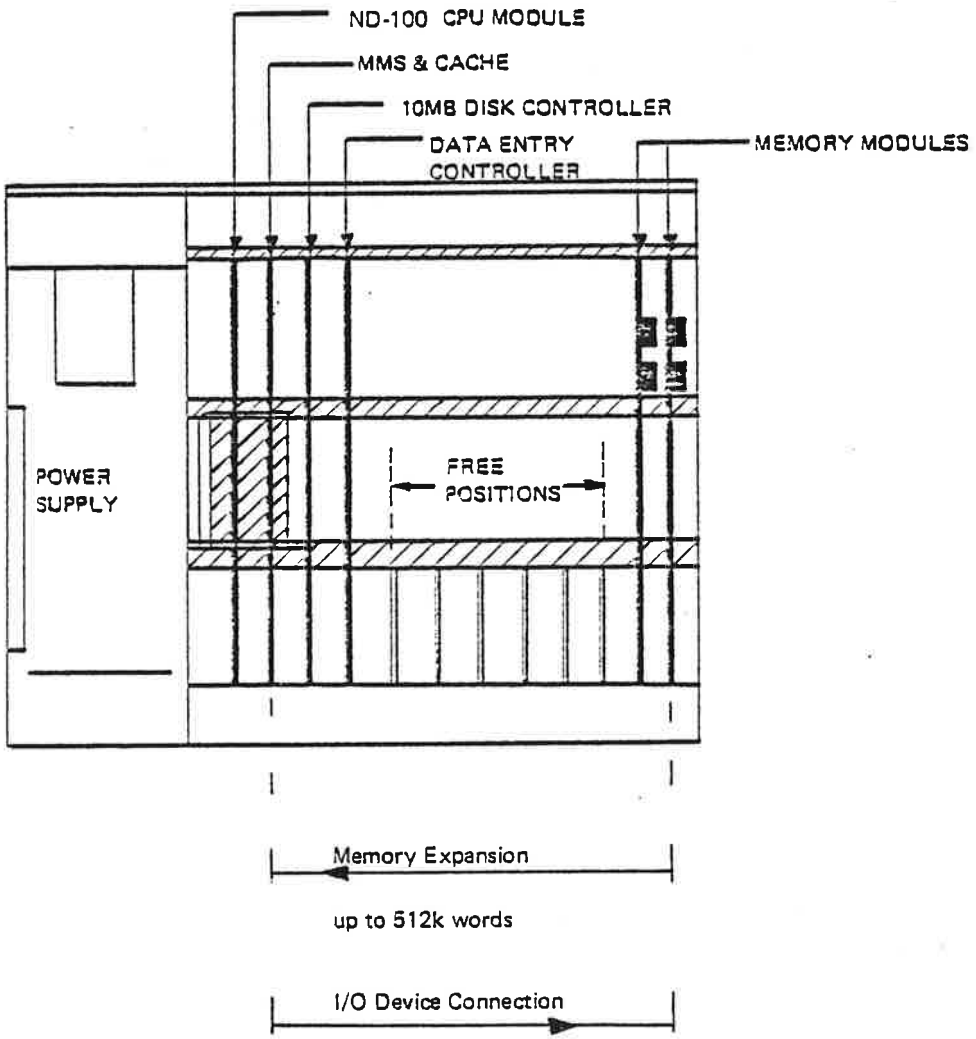


Figure 2.24: Recommended Placement of Modules in a ND-100 crate.

2.5.4 *The ND-100 Bus*

The ND-100 bus has been frequently mentioned due to its importance as a system highway.

Although the bus is physically one printed backplane, it may be divided up into two logical parts:

- Multiplexed address/data bus.
- Control lines.

ND-100 bus facts:

- The multiplexed address/data bus is 24 bits wide, supporting a physical address space of 16M words.
- No loss in memory access time due to multiplexed bus.
- Precise balance and termination give typically 20 ns address/data set up time.

All modules connected to the system are presented the same information simultaneously and are continuously "listening" to the bus activity.

The control lines are used to define the valid information on the bus (addresses or data) and to connect one source to one destination.

2.5.5 *ND-100 I/O System Functional Description*

External devices may be classified as:

- Slow character/word oriented input/output devices (example: terminals.)
- High speed block oriented mass storage devices (example: disk, magnetic tape).

ND-100 handles these device classes in different ways.

The first class is completely controlled by the CPU. This is called Programmed Input/Output (PIO).

The mass storage device controller operates directly on memory. This is called Direct Memory Access (DMA).

The program that controls a peripheral device is called a device driver. These drivers are subroutines delivered by Norsk Data together with the complete hardware/software configuration.

2.5.6 *Programmed Input/Output — PIO*

A PIO interface is always designed to handle slow byte/word oriented devices and is completely controlled by the CPU.

All exchange of data, control and status between the CPU and a device is programmed via the A register.

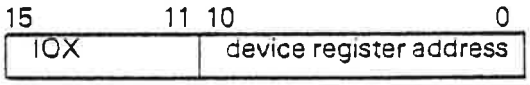
2.5.6.1 The Input/Output Instruction — IOX

The IOX instruction is a privileged machine instruction used in information exchange between the I/O system and the A register.

The I/O system usually contains several device controllers, each of them associated with a device register address. The lower 11 bits of the IOX instruction contains the address to the device that is to be accessed.

IOX instruction format:

IOX <device register address>



IOX Instruction Format

2.5.6.2 Interface Channels and Registers

An I/O interface is said to have two channels if it can handle both input and output transfers. This means one input channel and one output channel.

Examples:

— A terminal interface has two channels, one for input from the terminal's keyboard, one for output to the terminal's screen.

— A paper tape punch has only one channel, the output channel.

At least three registers are assigned to each channel for each device. Norsk Data's standard assignment of registers for a two channel device is:

Input Channel

— Input control register.
— Input status register.
— Input data register.

Output Channel

— Output control register.
— Output status register.
— Output data register.

Each of the above mentioned registers has a number in the device. In the IOX instruction the three least significant bits are used to select one register in the selected device.

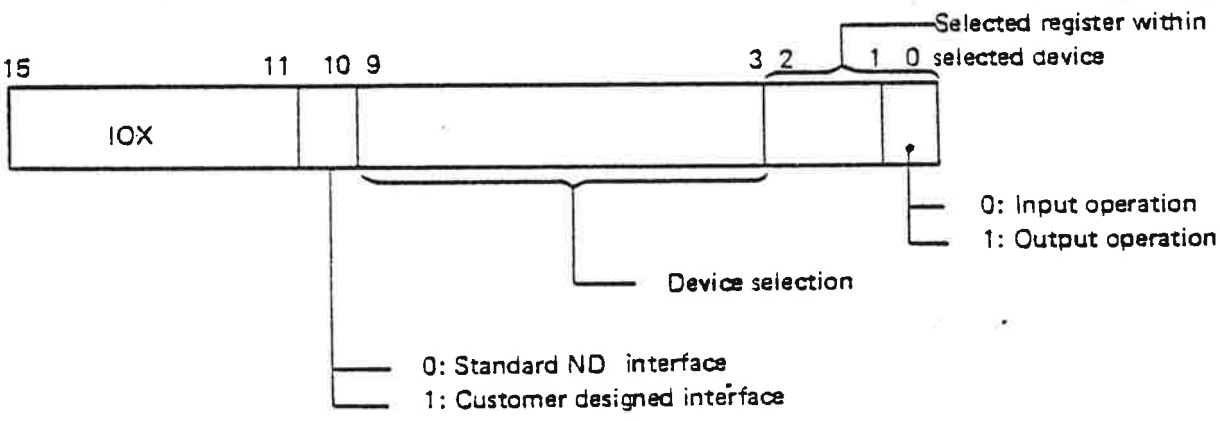


Figure 2.25: IOX Instruction Decoding Details

The IOX instruction is used for both input and output.

IOX Output

— Odd device register address (bit 0 = "1").
— Content of A register is written into register specified in "device register address".

IOX Input

- Even device register address (bit 0 = "0").
- Content of register specified in "device register address" is loaded into A register.

Device Register Address Range

Standard interfaces delivered by Norsk Data use addresses from 0 - 1777_8 (bit 10 is always zero).

Customer designed interfaces can use the address range from 2000_8 - 3777_8 (bit 10 is one).

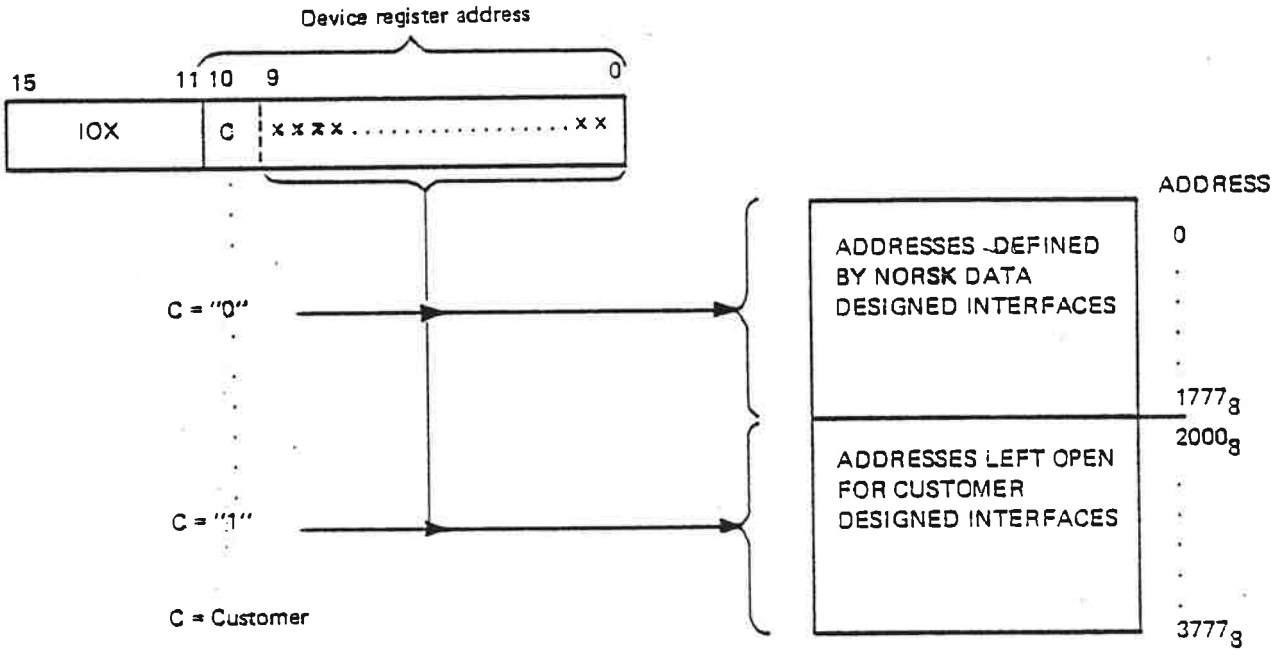


Figure 2.26: IOX Address Range

Special Feature

For future extension, of device addresses the T register can hold the device register address. The IOX instruction then has the format:

IOXT % T = <device register address>

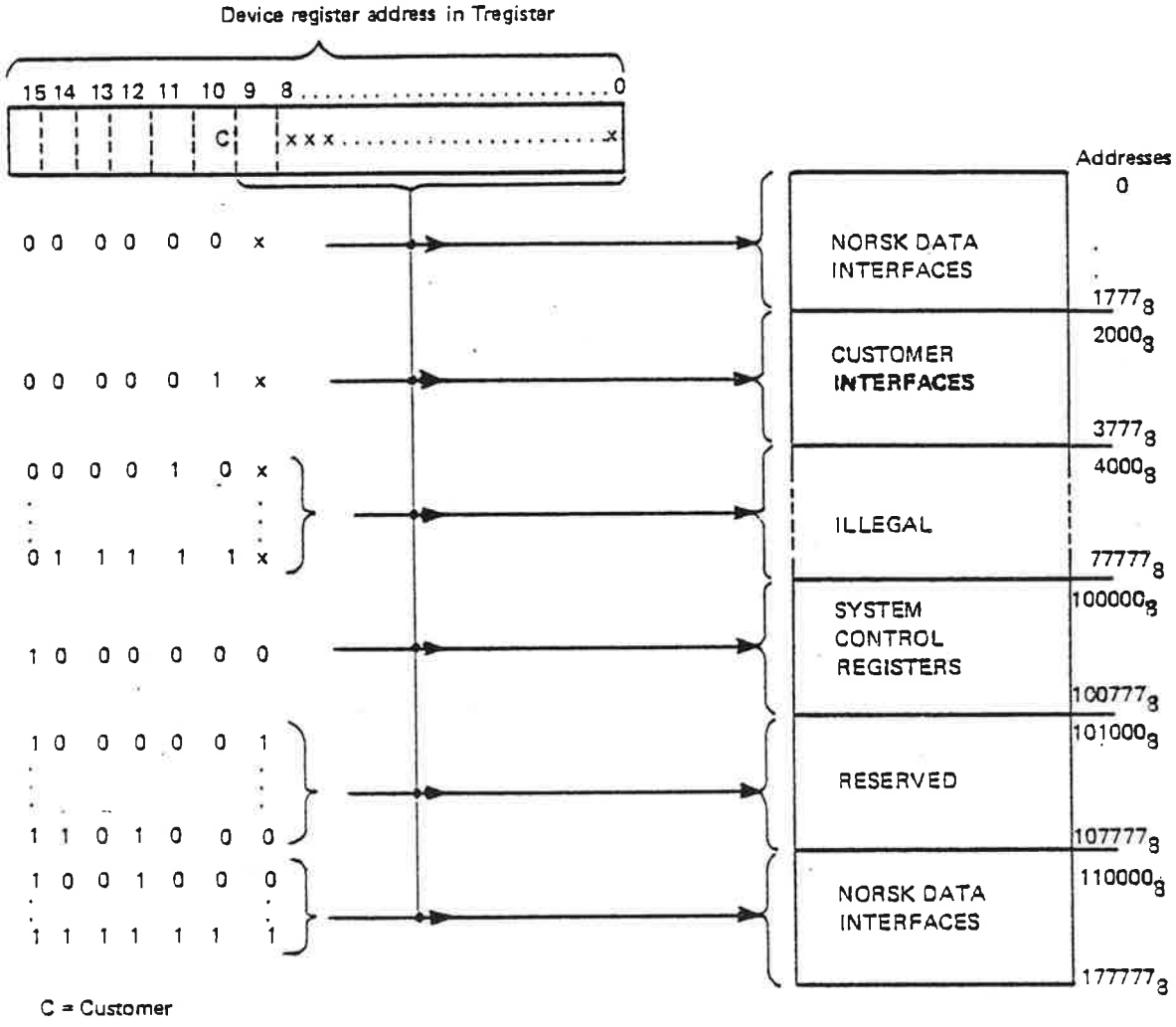


Figure 2.27: IOXT Address Range

2.5.6.3 Control and Status Registers

Commands to a device are given through the control register.

```
LDA <command>          % Initiate A register with command
IOX <dev. addr. + CR>  % Write control register from A register
                       % (CR = control register)
```

Device feedback goes through the status register:

```
IOX <dev. addr. + SR>  % Read status register to A register
                       % (SR = status register)
```

The formats of these registers are device dependent and found in the hardware programming specifications for each device type.

2.5.7 *Direct Memory Access (DMA)*

2.5.7.1 General

Direct Memory Access is used to obtain high transfer rates to and from memory.

Instead of using IOX for each word via the A register, a DMA controller is connected directly to the main memory via the ND-100 bus. This connection is called a DMA channel.

More than one DMA device may be active on the DMA channel at the same time, sharing the channel's total band width (1.8 M words/sec.).

Typical DMA devices are:

- Disks.
- Magnetic tapes.
- High speed serial/parallel intercomputer links.

After activation, a DMA transfer runs completely independently of the CPU. That means that CPU and DMA activity may be performed in parallel. CPU and DMA controllers operate simultaneously and independently of each other.

Conflicts are avoided by the bus control/driver in the CPU. If the CPU requests the bus (instruction fetch, I/O, access, etc.) simultaneously with a DMA controller, the bus is given to the DMA transfer. This effect is called cycle steal.

A HAWK disk, for example, will steal one cycle of 550 ns per each 6.4 *m*s transfer time which occupies less than 10% of the bus band width. The effect of cycle steal in this example is close to zero due to prefetch of instructions and the average distribution of bus requests within the instructions.

2.5.7.2 DMA Controller Operation

A DMA transfer may be divided into three steps:

- Initialization.
- Transfer.
- Termination and status check.

The bus is also fast enough to handle both DMA activity and CPU activity at the same time without slowing down the CPU. A CPU memory reference will hold the bus for typically 320 ns, a DMA transfer typically 550 ns.

A disk transfer, consequently, will use 550 ns of bus time for each 6.4 *n* of transfer time. That is less than 10% of the bus band width. This does not mean that there is 10% less CPU activity. The use of instruction prefetch and normal distribution of memory references reduces. DMA activity to practically zero overhead.

2.5.7.2.1 INITIALIZATION

A DMA controller has to be initialized before a transfer can be started. The initialization is done by a device driver activated by the operating system when a DMA transfer is needed.

The driver program accesses the DMA controller by means of IOX instructions. Through different transfer parameters, the driver tells the DMA interface what to do.

Typical parameters are:

- Memory Address Register (MAR) holds the first memory address to read from (DMA output) or write into (DMA input).
- Block Address Register (BAR) holds the first address to read or write from on the physical device.
- Word Count Register holds the number of words to be transferred.
- Control Register gives device function (read, write, etc.) and start.

The formats of the registers are given in the hardware programming specifications for each device.

2.5.7.2.2 TRANSFER

After initialization and start is given, the data transfer takes place. Data is exchanged between the DMA controller and memory at the speed determined by the device.

In order to reduce the possibility for overrun on input and underrun on output, each device controller contains a buffer for at least 16 words between device and memory.

2.5.7.2.3 TERMINATION AND STATUS CHECK

The DMA transfer is completed when the word counter is zero. A DMA controller tells this to the CPU through an interrupt on level 11. The device driver is again activated to read the device status which gives information on the status of the transfer.

2.5.7.2.4 GENERAL CONSIDERATIONS

In ND-100 all DMA controllers have a buffer for at least 16 words between device and memory. That is, if the DMA channel for some reason is occupied, the buffer will prevent underrun on output and overrun on input.

If there is a high load on the DMA channel, i.e., several DMA controllers that can be active at the same time, some general considerations should be taken.

- The DMA controller with the smallest buffer should be placed closest to the CPU.
- If several DMA controllers have the same buffer space, the fastest should be placed closest to the CPU.

These rules are related to hardware priority associated to placement relative to the CPU.

2.5.8 *The I/O System and the Interrupt System*

2.5.8.1 General

Under a running system (SINTRAN III), all I/O devices connected to the ND-100 will be prepared for operation and then allowed to operate asynchronously with respect to the CPU. That means that the I/O controllers activate themselves through an interrupt to the CPU if a status change occurs.

Possible status changes in the I/O system are:

— End of operation interrupt.

  If output this means data is transmitted, can accept next

  If input this means data is available, please read it (before overrun)

— Error interrupt.

2.5.8.2 Interrupt Level Usage

Interrupt levels 10, 11, 12, 13 and 15 are available to the I/O system as physical lines in the ND-100 bus. These lines go directly to the interrupt detect controller in the CPU.

The Level Assignment

— All output interrupts use level 10.

— All DMA controllers use level 11.

— All input interrupts use level 12.

— Real-time clocks and special devices such as HDLC input use level 13.

— Level 15 is not used by Norsk Data equipment but is available for special purposes.

2.5.8.3 Device Interrupt Identification

As indicated above, more than one device may use the same interrupt line. In order to find the interrupting device an IDENT instruction is executed.

The IDENT <PL> will return a vector (called ident code) from the interrupting device to the A register.

The ident code is unique for each device and is used to find that device driver. The driver will read the status register to find the reason for the interrupt and take proper action.

The IDENT <PL> instruction will only search for interrupts on the level specified in PL (10 - 13).

Example:

> The instruction IDENT PL12 will only search for interrupt in the input channel. A possible existing interrupt on level 10 or level 11 is ignored and handled later by IDENT PL10 and IDENT PL11 respectively.

2.5.9 *Programming Specifications for I/O Devices on the CPU Board*

The real-time clock (device register address range10-13) is always located on the CPU board. The terminal with device register address range 300-307 is located on the CPU board unless a strap on the CPU board is removed.

Since these devices are included in every CPU, their programming specifications are given here. Programming specifications for other devices are given in separate manuals.

2.5.9.1 The Real-time Clock

The real-time clock on the CPU board has device register address range10-13.

IOX 10: Returns 0 in the A register and has no other effect.

IOX 11: Clear real-time clock counter. This instruction will cause the next clock pulse to occur exactly 20 ms later. If this instruction is executed repeatedly, the counter will never be incremented, and no clock pulses will occur. This may affect the execution of operator's communication on console terminal.

IOX 12: Read real-time clock status.

Bit 0 - 1: The clock will give interrupt when next clock pulse arrives.

Bit 3 = 1: The clock is ready for transfer, i.e., a clock pulse has occurred.

Bits 1-2 and 4-15 are always zero.

IOX 13: Set real-time clock status.

Bit 0 = 1: Enable interrupt if ready for transfer occurs.

Bit 13 = 1: Clear ready for transfer.

2.5.9.2 The Current Loop Interface

The current loop interface located on the CPU board has device register address range 300 - 307.

IOX 300: Read input data (according to input control word setting). The last inputted character is transferred to the A register. The data available signal is reset if the micro programmed operator communication (MOPC) is not active.

IOX 301: No operation.

IOX 302: Read input status.

Bit 0 = 1: Data available will give interrupt when it occurs.

Bit 3 = 1: Data is available (ready for transfer). Is never given if MOPC is active.

Bit 4 = 1: Inclusive or of error bits 5-7.

Bit 5 = 1: Framing error.

Bit 6 = 1: Parity error.

Bit 7 = 1: Overrun.

Bits 1-2 and 8-15 are always zero.

IOX 303: Set input control.

Bit 0 = 1: Enable interrupt if data available (ready for transfer) occurs.

Bit 11 and Bit 12:
Bit 11 = 1 and Bit 12 = 1 signifies 5 bits code.
Bit 11 = 0 and Bit 12 = 1 signifies 6 bits code.
Bit 11 = 1 and Bit 12 = 0 signifies 7 bits code.
Bit 11 = 0 and Bit 12 = 0 signifies 8 bits code.

Bit 13 = 1 signifies 1 stop bit.
Bit 13 = 0 signifies 2 (1.5 for 5 bits) stop bits.

Bit 14 = 1: A parity bit is added to the number of bits mentioned above.

Bit 14 = 0: No extra bit is added to the bits mentioned above.

IOX 304: Returns 0 in the A register and has no other effect.

IOX 305: Write data (according to input control word setting).

IOX 306: Read output status.

Bit 0 = 1: Ready for transfer will given interrupt when it occurs.
Bit 3 = 1: Ready for transfer.
Bits 1-2 and 4-5 are always zero.

IOX 307: Set output control.

Bit 0 = 1: Enable interrupt if ready for transfer occurs.

2.6 *ND-100 BUS EXTENDER (BEX)*

2.6.1 *General*

Although 21, or often less than 12, modules are sufficient for most systems, some configurations require more space than even the 21 position card crate can offer.

This space problem is solved by using the ND-100 Bus Extender (BEX) system. The BEX system makes it possible to extend the ND-100 bus by linking together card crates. The maximum number of card crates is 8. Using 21 position card crates this give 168 positions for card connection. Note that only one CPU module and one Memory Management System (MMS) module may be connected to the system. The rest of the positions is free for Input/Output modules and Memory modules.

2.6.2 *Bus Extender Architecture*

The BEX system consists of Bus Extender (BEX) modules and crate interconnection cables. One BEX module is located in each crate. Two crates are physically connected via two interconnection cables between the BEX modules. Refer to Figure 2.28.

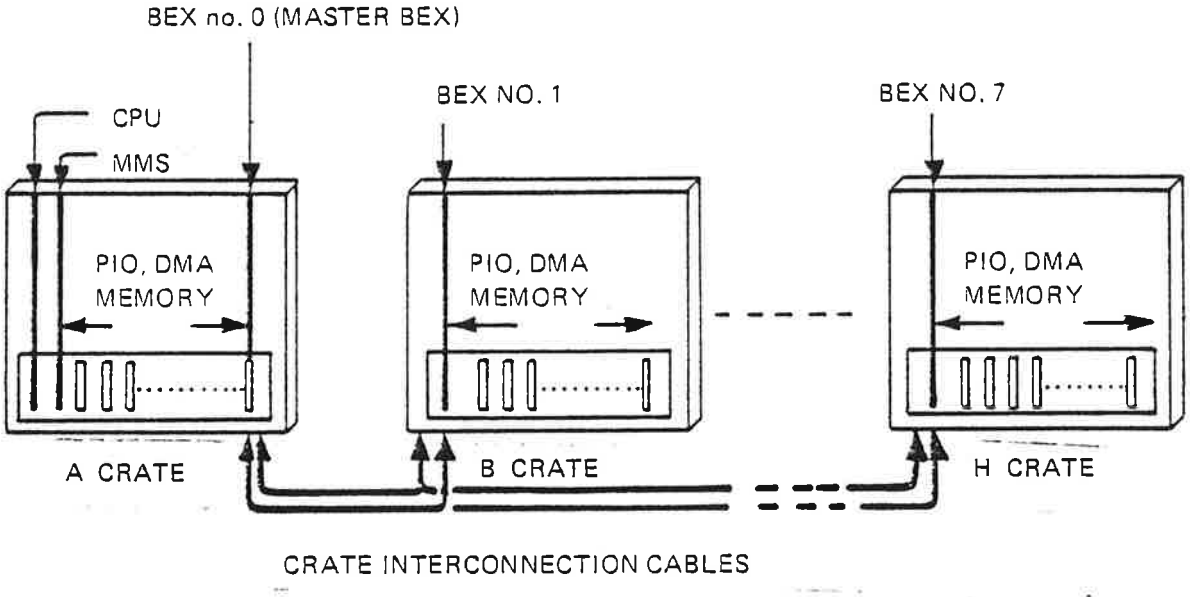


Figure 2.28: ND-100 Bus Extender System

The crate where the CPU is located is called the A crate. The BEX module located in the A crate is named BEX no. 0 (also MASTER BEX).

It is posssible to mix Programmed Input/Output (PIO) modules, Direct memory Access (DMA) modules and memory modules in all the crates.

3 ND-100 INSTRUCTIONS

3.1 *INTRODUCTION TO THE INSTRUCTION REPERTOIRE*

3.1.1 *General*

In the ND-100 all instructions occupy a single word, 16 bits, yielding an efficient use of memory and high speed code. Floating point arithmetic operations and floating/integer conversions are included in the standard instruction set.

The instruction set of ND-100 is divided into the following 5 classes:

- Memory reference instructions.
- Register instructions.
- Input/Output control instructions.
- System control instructions.
- Customer specified instructions

Each instruction is given a short description. This includes its mnemonic as used in the assembly language, the 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.

When a register is mentioned in this chapter, it refers to the register set on the current program level. For example, "the A register" means the A register on the current program level.

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 PTM, TG, K, Z, Q, O, C, M

Status Word:

Bit

0	PTM	Page table mode
1	TG	Rounding indicator for floating point operations
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	Multi-shift link indicator
8-11	PL	Program level indicator
12	N-100	ND-100 Indicator
13	SEXI	Extended address mode
14	PONI	Memory Management On Indicator
15	IONI	Interrupt System On Indicator

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
	Logical AND
V	Logical inclusive OR
V	Logical exclusive OR
( )	The contents of
μs	Microsecond
ns	Nanosecond

3.1.2 *Instruction and Data Formats*

The ND-100 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.

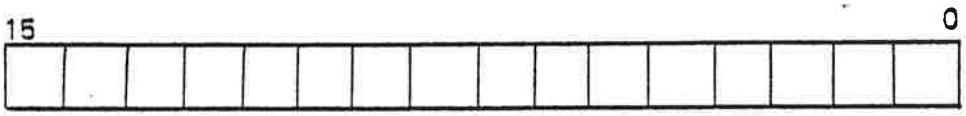


Figure 3.1: ND-100 Bit Numbering Convention

The content of a ND-100 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 contents of a word with all bits set to one is represented as 177777.

The standard ND-100 instruction set provides instructions for the following 6 different data formats:

1. Single bit
2. 8 bit byte
3. 16 bit word
4. 32 bit double word
5. 48 bit floating point word
6. 32 bit floating point word (optional, instead of 48 bit floating point)

3.1.2.1 Single Bit

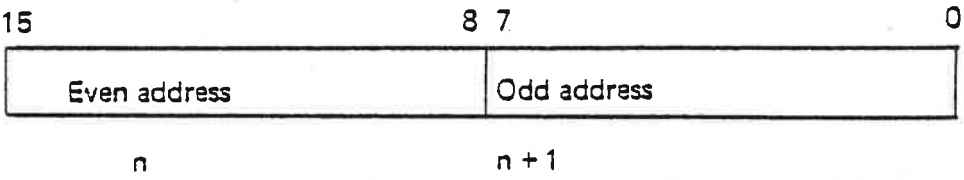
A single bit data word is typically used for a logical variable; the bit instructions are used for manipulation of single bit variables. The bit instructions specify operations on any bit in any of the general registers, as well as the accumulator indicator K.

3.1.2.2 8 Bit Byte

Two instructions are available in the standard ND-100 instruction set for byte manipulation, i.e., load byte and store byte.

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

The byte addressing 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

3.1.2.3 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 also contains instructions to treat numbers as unsigned (absolute magnitude) numbers.

Range

$-32768 \leqslant X \leqslant 32767$

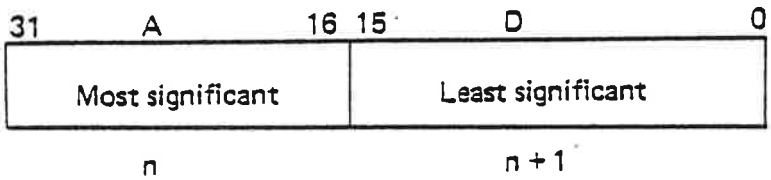
or

$0 \leqslant X \leqslant 65535$

3.1.2.4 32 Bit Double Word

Two instructions are available to handle double word formats, load double and store double.

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:

$-2\,147\,483\,648 \leqslant X \leqslant 2\,147\,483\,647$

3.1.2.5 48 Bit Floating Point Word

The standard ND-100 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.

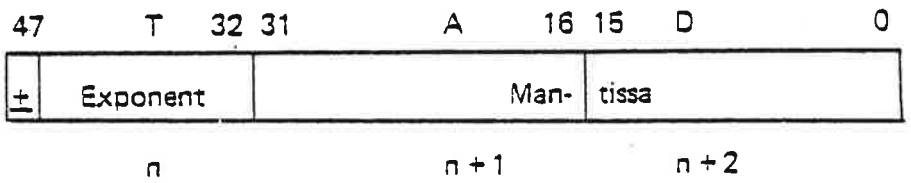
The data format of floating point words uses 32 bits for the mantissa, one bit for sign and 15 bits for biased exponent.

The mantissa is always normalized, $0.5 \leqslant \text{mantissa} < 1$. 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

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

or

$$10^{-4920} < X < 10^{4920}$$

Examples (octal format):

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

3.1.2.6 32 Bit Floating Point Word

As an option, the ND-100 may be equipped with microprogram for 32 bit floating point format instead of the standard 48 bit format described in the previous section. The instructions affected are:

FAD	Floating Point Add
FSB	Floating Point Subtract
FMU	Floating Point Multiply
FDV	Floating Point Divide
NLZ	Convert Integer to Floating Point
DNZ	Convert Floating Point to Integer

The data format of 32 bit floating words uses 23 bits for the mantissa, one bit for sign and 9 bits for a biased exponent. These 32 bits are packed in two 16 bit words by omitting the most significant bit of the mantissa, which is always a one in non-zero numbers.

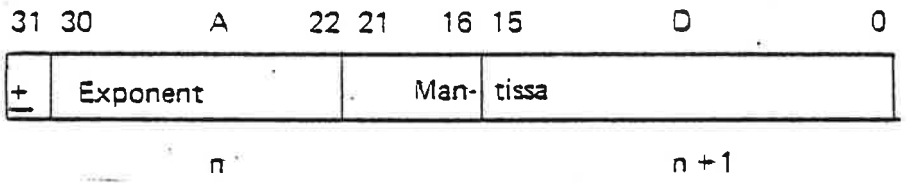
The mantissa is always normalized, $0.5 \leqslant \text{mantissa} \leqslant 1$. The exponent base is 2, the exponent is biased with 2^8.

A standarized floating zero contains zero in all 32 bits.

In main memory, one 32 bit floating point data word occupies two 16 bit memory locations, which are addressed by the address of the exponent part.

n	exponent, sign and mantissa bits 16-21
n + 1	mantissa bits 0-15

In CPU registers, bits 0 - 15 of the mantissa are in the D register, bits 16 - 21 and exponent and sign are in the A register. These two registers together are defined as the 32 bit floating accumulator. The T registrer is not affected by 32 bit Floating Point operators.



32 Bit Floating Point Word Format

The accuracy is 23 bits or approximately 7 decimal digits. Any integer up to 2^{23} has an exact floating point representation.

The range is

$$2^{-256} \cdot 0.5 \leqslant X < 2^{255} \cdot 1 \text{ or } X = 0$$

or

$$10^{-76} < X < 10^{76}$$

Examples (octal format):

	A	D
0:	0	0
+1.0:	040100	0
−1.0:	140100	0
+3.0:	040240	0

NOTE: The instruction times are given in Appendix A.2.

3.2 *THE INSTRUCTION REPERTOIRE*

3.2.1 *Memory Reference Instructions*

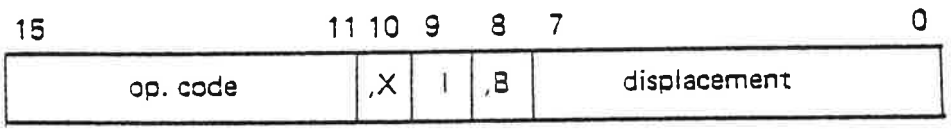
Memory reference instructions specify operations on words in memory. For all the memory reference instructions in ND-100, the addressing mode is the same with the exception of the conditional jump, the byte and the register block instructions. The addressing structure for these memory reference instructions is given under the specific instruction specification.

The ND-100 has the following groups of memory reference instructions:

- Store instructions.
- Load instructions.
- Arithmetic and logical instructions.
- Sequencing instructions.
- Byte instructions.
- Register block instructions.

3.2.1.1 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 an 8 bit signed displacement using 2's complement for negative numbers and sign extension. (Note that excluded from this is the conditional jump, the byte and the register block instructions.)



ND-100 uses a relative addressing scheme, 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" and ",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 take place after a possible indirect addressing.

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 range for directly addressing 128 locations backwards and 127 locations forward.

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 one extra memory cycle to the listed execution time.

The address computation is summarized in the table below. 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
( )	Contents of a 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) + displ.)
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)

Addressing Mode Table

P relative Addressing (,X = 0 I = 0 ,B = 0)

The *P relative* addressing mode is specified 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, which here we shall represent by STA * 2, and this instructiuon 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 limitation to this mode of addressing (locations more than 128_{10} 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.

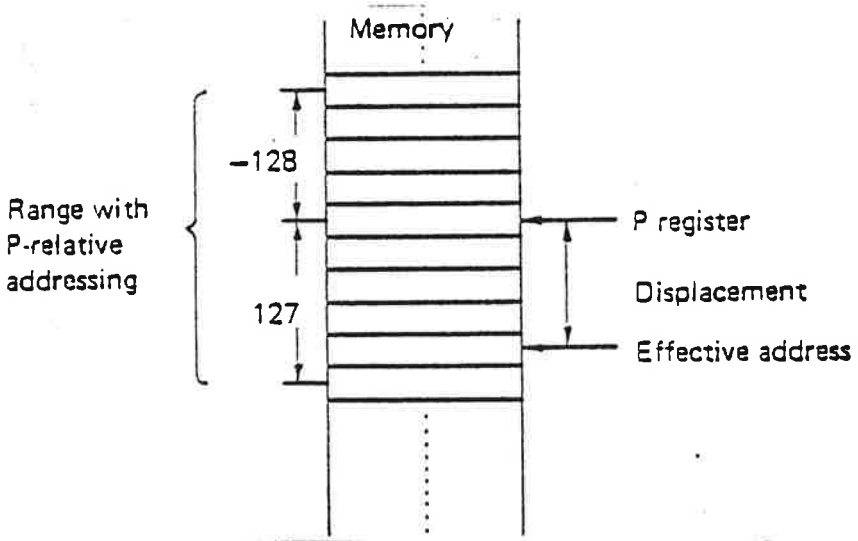


Figure 3.2: Schematic Illustration of P relative Addressing

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 the 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 another memory location which is accessed instead.

Example:

Suppose location 405 contains the instruction LDA I * 2 (045002_8) and that this instruction is executed. Let us also suppose memory location 16003 contains the value 17 and that 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 is used as an address and the contents of this address (17) is finally loaded into the A register.

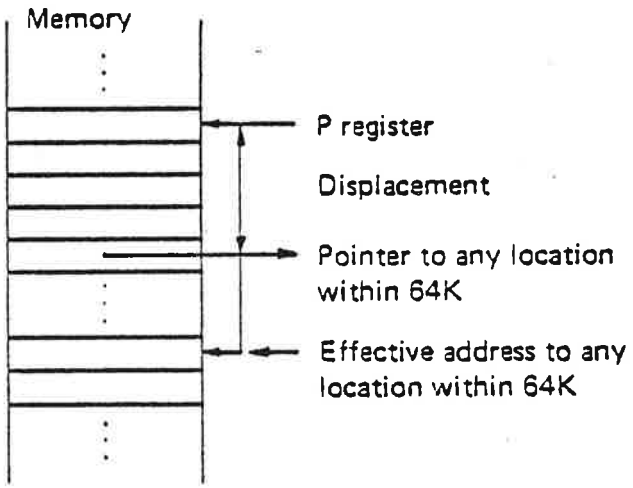


Figure 3.3: Schematic Illustration of Indirect P relative Addressing

B relative Addressing (,X = 0 I = 0 ,B = 1)

The above two addressing modes are theoretically quite sufficient. However, if the ND-100 provided only 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 relative 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 and the resulting sum is used to specify the memory location accessed.

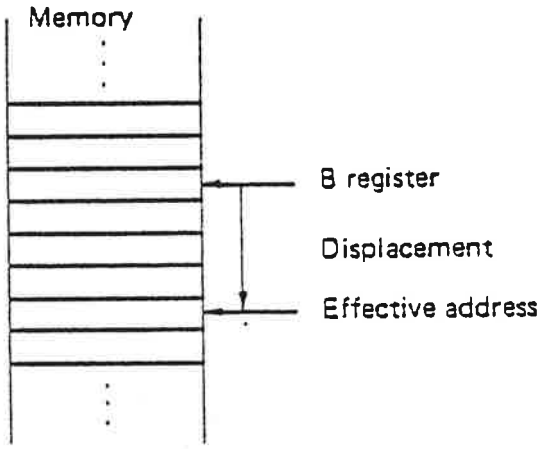


Figure 3.4: Schematic Illustration of B relative Addressing

Example:

Let location 405 contain the instruction LDA −4,B (044774_8) and the B register contain 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 the sum of 10031, and the contents of the 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 contain 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 location 403, 2000, is then used as a pointer to the subroutine.

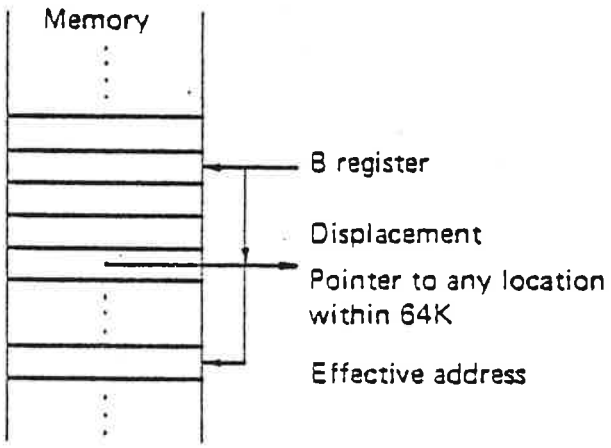


Figure 3.5: Schematic Illustration of Indirect B relative Addressing

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 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 push down 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
        ...
        ...
        ...
```

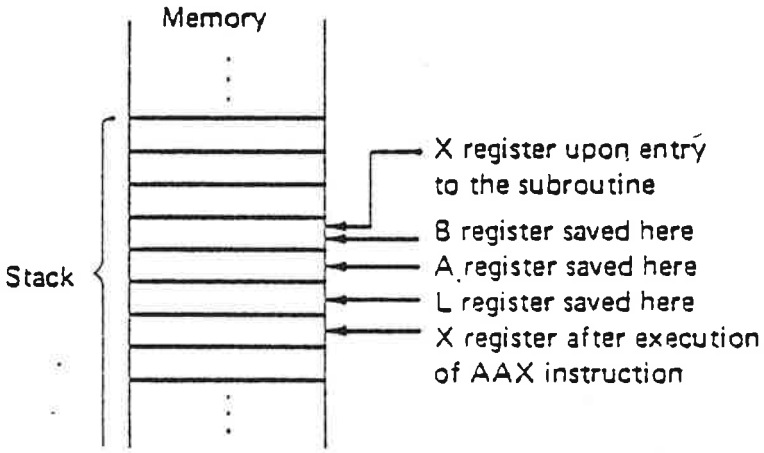


Figure 3.6: Illustration of the Effect of the Stack Code

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

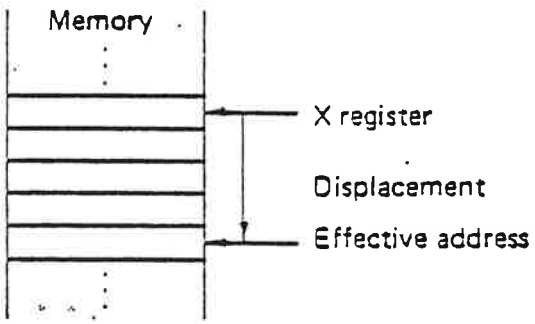


Figure 3.7: Schematic Illustration of X relative Addressing

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.

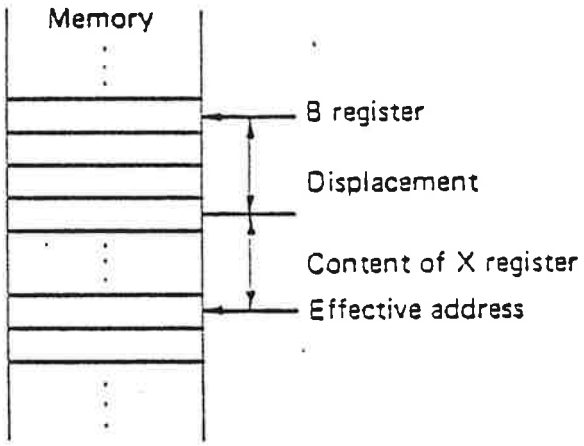


Figure 3.8: Schematic Illustration of B relative Indexed Addressing

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

The last two addressing modes are 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.

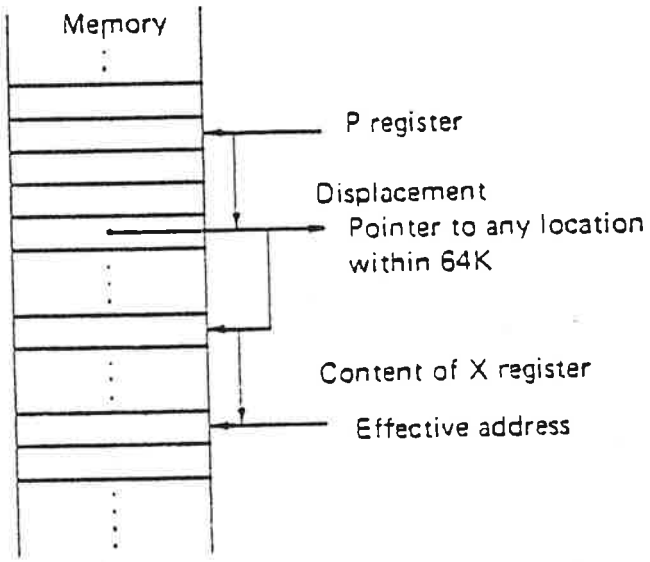


Figure 3.9: Schematic Illustration of Indirect P relative Indexed Addressing

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 is used instead of the contents of the P register in the effective address computation. This mode can therefore be used to step through arrays pointed to from a data area common to several subprograms.

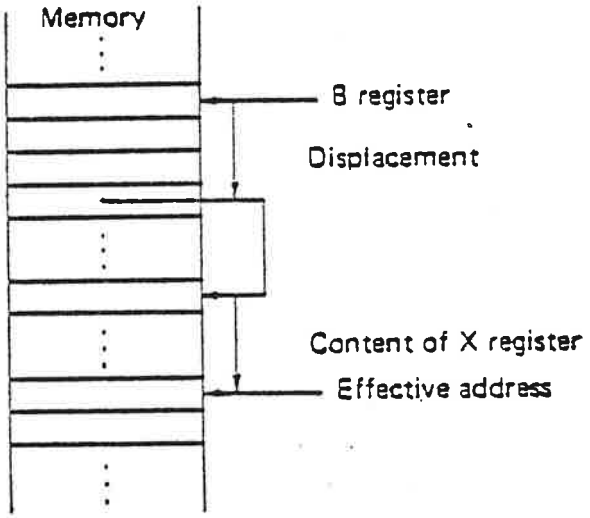


Figure 3.10: Schematic Illustration of Indirect B relative Indexed Addressing

3.2.1.2 Store Instructions

STZ — Store zero — Code: 000 000

Format: STZ <address mode> <disp.>

The effective location is cleared.
Affected: (EL)

STA — Store A register — Code: 004 000

Format: STA <address mode> <disp.>

The contents of the A register are stored in the effective location.
Affected: (EL)

STT Store T register Code: 010 000

Format: STT <address mode> <disp.>

The contents of the T register is stored in the effective location.
Affected: (EL)

STX Store X register Code: 014 000

Format: STX <address 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)

STD Store double word Code: 020 000

Format: STD <address mode> <disp.>

The contents of the A register are stored in the effected location, and the contents of the D register are stored in the effective location plus one.
Affected: (EL), (EL + 1)

STF Store floating accumulator Code: 030 000

Format: STF <address mode> <disp>

The contents of the floating accumulator is stored in three memory locations, starting with exponent part in effective location.
Affected: (EL), (EL + 1), (EL + 2)

MIN Increment memory and skip if zero Code: 040 000

Format: MIN <address mode> <disp.>

Effective word is read and incremented by one and then stored in the effective location. If the result becomes zero, the next instruction is skipped.
Affected: (EL), (P)

3.2.1.3 Load Instructions

LDA — Load A register — Code: 044 000

Format: LDA <address mode> <disp.>

The effective word is loaded into the A register.
Affected: (A)

LDT — Load T register — Code: 050 000

Format: LDT <address mode> <disp.>

The effective word is loaded into the T register.
Affected: (T)

LDX — Load X register — Code: 054 000

Format: LDX <address mode> <disp.>

The effective word is loaded into the T register.
Affected: (X)

LDD — Load double word — Code: 024 000

Format: LDD <address 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)

LDF — Load floating accumulator — Code: 034 000

Format: LDF <address 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)

3.2.1.4 Arithmetical and Logical Instructions

ADD — Add to A register — Code: 060 000

Format: ADD <address 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 positions of the adder, otherwise the carry indicator is reset to 0. If 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

SUB — Subtract from A register — Code: 064 000

Format: SUB <address 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 indicatttors.
Affected: (A), C, O, Q

AND — Logical AND — Code: 070 000

Format: AND <address 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)

ORA — Logical inclusive OR — Code: 074 000

Format: OR <address 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)

MPY — Multiply integer — Code: 120 000

Format: MPY <address 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

FAD — Add to floating accumulator — Code: 100 000

Format: FAD <address 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.
Affected: (T), (A), (D), TG

FSB — Subtract from floating accumulator — Code: 104 000

Format: FSB <address 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.
Affected: (T), (A), (D), TG

FMU Multiply floating accumulator Code: 110 000

Format: FMU <address mode> <disp.>

The contents of the floating accumulator are multiplied with the number in the effective floating word locations with the result in the floating accumulator.
Affected: (T), (A), (D), TG

FDV Divide floating accumulator Code: 114 000

Format: FDV <address 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 attempted, the error indicator Z is set to one. The error indicator Z may be sensed by a BSKP instruction (see BOP).
Affected: (T), (A), (D), Z, TG

3.2.1.5 Sequencing Instructions

JMP — Jump — Code: 124 000

Format: JMP <address mode> <disp.>

The next instruction is taken from the effective address of the JMP instruction (the effective address is loaded into the program counter).
Affected: (P)

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

Format: JPL <address mode> <disp.>

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

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 conditions, but is the same for all instructions.

A conditional jump instruction must be specified by means of the 8 mnemonics listed below. It is illegal to specify CJP or any combinations of ,B, I and ,X.

The 8 jump conditions are as follows:

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. 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 X 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.

3.2.1.6 Byte Instructions

To facilitate the handling of character strings, the ND-100 provides two instruction for byte handling, load byte, LBYT and store byte, SBYT.

Because of the requirements of full 64K addressing, the LBYT and SBYT use an addressing scheme different from the normal ND-100 addressing.

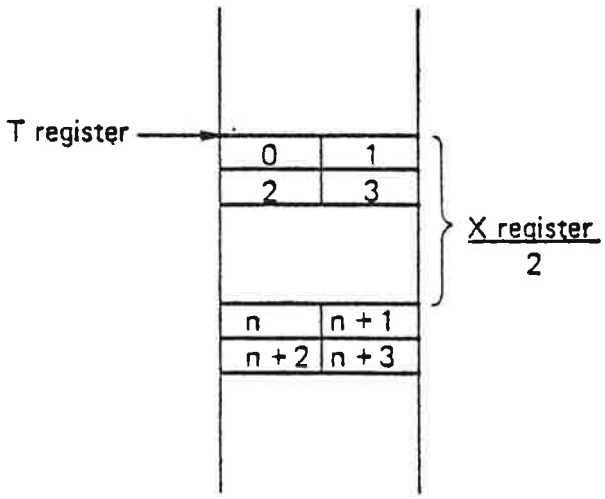
For byte addressing, two of the ND-100 registers, the T and X registers are used for addressing the byte.

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) + ½ (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 8 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)

SBYT Store byte Code: 142 600

Format: SBYT

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)

3.2.1.7 Extended BYTE-instructions

Byte operands occupy fields in the memory that may start and end at any byte address. A byte operand is specified by a two word descriptor, giving start address and field length:
The descriptor's words have the following format:

D1: Bit 0-16 Give the byte operand's word address in the memory.

D2: Bit 15. This bit specifies whether the operand starts in the left byte or the right byte.
Bit 15=0, left byte
Bit 15=1, right byte

Bit 14. Page table mode (bit 14=1 selects the alternativepage table).

Bit 13. This bit should be 0 when the instruction is started.

Bit (0-11). Field length (number of bytes).

The descriptor of the source operand is contained in the A, and D registers; The descriptor of the destination operand is in the X, and T registers (for D1, D2 respectively).

Field length may be of any size less than 4K-1 bytes. Sufficient interruptability is taken care of during execution.

The extended byte-instructions may run on all program levels.

BFILL Byte Fill Code: 140 130

Format: BFILL

This instruction has only one operand. The destination operand is specified in the X, and T registers. The right-most byte in the A-reg. (bits 0-7) is filled into the destination field.

After execution, the X-register and T-register bit 15 point to the end of the field (after the last byte). The T-register bits (0-11) equal zero.

The instruction will always have a skip return (no error condition).

MOVB Move bytes Code: 140 131

Format: MOVB

This instruction moves a block of bytes from the location specified for the source operand to the location specified for the destination operand.

The move operation takes care of source- and destination-field overlap.

The number of bytes moved is determined by the shortest field length of the operands.

After execution, the A,D and X,T registers (bit 15 in D and T) point to the end of the field that is moved (after the last byte). D-reg. bits 0-11 equal zero and T-reg. bits 0-11 contain the number of bytes moved.

The T-reg. bits 12-13 and the D-reg. bit 12 are used during the execution, and are left cleared. Bit 13 must be zero before execution (used as an interrupt mark).

Tne instruction will always have a skip return (no error condition).

MOVBF Move bytes forward Code 140 132

Format: MOVBF

This instruction moves a block of bytes from the location specified as the source operand to the location specified as the destination operand.

The move operation always starts with the first byte (lower address). The number of bytes moved is determined by the shortest field length of the operands. Forbidden overlap exists when the source data to be moved, will be destroyed. That happens when a byte is stored in a word before that word is read from memory. This is reported by an error return (no skip).

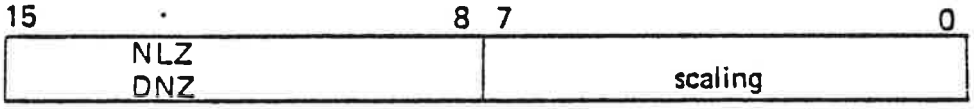
After successful execution, the A,D and X,T registers (bit 15 in D and T) point to the end of the fields that are moved (after the last byte). The numbers initially contained in the D- and T-registers, bits 0-11, are decremented by the number of bytes moved.

The T-reg. bits 12-13 and the D-reg. bit 12 are used during the execution and are left cleared. Bit 13 must be zero before execution (used as an interrupt mark).

The instruction will have a skip-return when no illegal overlap exists.

3.2.2 *Register Instructions*

3.2.2.1 Floating Point Conversion Instructions



Two instructions are available. A single precision fixed point number may be converted to a 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 subinstructions are described in Section 3.2.2.1.1 for the standard 48 bit floating point format, and in Section 3.2.2.1.2 for the alternative optional 32 bit floating point format.

3.2.2.1.1 STANDARD 48 BIT FLOATING POINT CONVERSION

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, a scaling factor of $+16_{10}$ will give a floating point number with the same value as the integer. 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)

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, a scaling factor of -16_1, $_0$ will give a fixed point number with the same value as the integer part of the floating point number. 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 zeros.

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

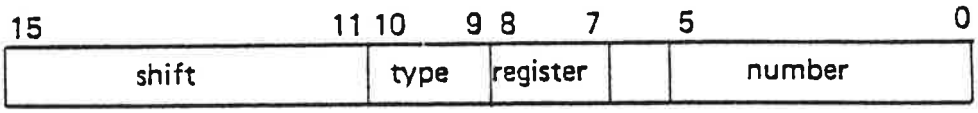
* When converting an exact floating point zero, scaling factors more negative than −16 will give erroneous results.

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

3.2.2.1.2 OPTIONAL 32 BIT FLOATING POINT CONVERSION

The normalize and denormalize operations for 32 bit floating point use the same instruction codes as for 48 bit floating point operations, but do not affect the T register. For the 32 bit DNZ operations, the scaling factor should *always* be −16. Other scaling factors will not cause a different result but will affect the test for overflow.

3.2.2.2 Shift Instructions



Shift instructions operate on registers. A shift instruction consists of three parts:

- The register to be shifted (specified by the shift register 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 the form:

<shift register> <type> <number>

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

Note that bit 6 in the instruction is ignored.

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

SHT — Shift the T register (register field 00) — Code: 154 000

Format: SHT <type> <number>

The T register is shifted as specified by the <type> and <number>.
Affected: (T), M

SHD — Shift the D register (register field 01) — Code: 154 200

Format: SHD <type> <number>

The D register is shifted as specified by the <type> and <number>.
Affected: (D), M

SHA — Shift the A register (register field 10) — Code: 154 400

Format: SHA <type> <number>

The A register is shifted as specified by the <type> and <number>.
Affected: (A), M

SAD — Shift the A and D registers connected (register 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

Type Field

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

Mnemonic		Type field	
nil	Arithmetic shift. During right shifts, the sign bit (bit 15) is extended during the shifting, in left shifts zeros are fed into vacated bit positions.	0	0 Code: 000 000
ROT	Rotational shift. In single register shifts 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> in the number field of the instruction is a signed number, 5 bits plus sign, which specifies the shift direction (positive or negative shift) and the number of shifts.

N > 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 75_8
SHA $100-3_8$
SHA SHR 3_8

Note that SHA −3 *cannot* be used.

In a right shift, nothing should be written between the SHR mnemonic and the number of shifts (this is peculiar for the assembler). 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 10_8

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 (the SWAP is faster).

SAD ROT 20

Example 4:

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

SHD ZIN SHR 2

3.2.2.3 Register Operations

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

<register operation> <sub-instruction> <sr> <dr>

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

ROP register operations (see Section 3.2.2.3.1)
EXTended register operation instructions (see Section 3.2.2.3.2)

In addition, there are two instructions for accessing single registers outside current program level (see Section 3.3.3) and two instructions for accessing a whole register block outside current program level (see Section 3.3.2).

Only the ROP instructions have sub-instructions.

The ROP register instructions are:

RADD	Register addition, dr ← + sr	Code: 146 000
RSUB	Register subtraction, dr ← dr — sr	Code: 146 600
RAND	Register logical AND, dr ← dr sr	Code: 144 400
RORA	Register logical OR, dr ← dr V sr	Code: 145 500
REXO	Register logical exclusive OR, dr ← dr V sr [V REXO]	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) and 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:

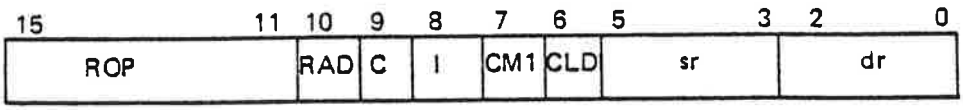
SD	D register	as source	Code: 10
SP	Program counter	as source	Code: 20
SB	B register	as source	Code: 30
SL	L register	as source	Code: 40
SA	A register	as source	Code: 50
ST	T register	as source	Code: 60
SX	X register	as source	Code: 70

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

3.2.2.3.1 ROP — REGISTER OPERATION INSTRUCTIONS



The instruction decodes bits 0-10 as:
Source and Destination Register (bits 0-5):

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 instruction.

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

Bits 3-5 specify one out of seven registers containing the value to be used as the source register operand.

sr = 0: Produces a source value equal to zero.

Subinstructions (bits 6-10):

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 to both add previous carry and to add 1 in the same ROP instruction. (If this is attempted, 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
C, I = 0, 1: Logical and, RAND
C, I = 1, 0: Logical exclusive or, REXO
C, I = 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.

Exclusive ROP Mnemonics

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 specifed.

(ADC, AD1)

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

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

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

(RSUB, CM1, ADC, AD1)

RSUB uses CM1 and AD1.

Specifying ROP Instructions

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 logical exclusive OR
COPY,	dr ← sr	Register transfer

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

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 specifications.

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 ∧ 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				<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		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[1]	AD1		CLD	<sr><dr>	dr ← sr + 1
1 0 1 1 0	RADD[2]	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						
1 1 1 0 1	} not applicable					
1 1 1 1 0						
1 1 1 1 1						

The ROP Instruction Table

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

Some examples of use of the ROP instruction.

Example 1:

Add the contents of the A and X registers 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:

Subract 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 DA 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
```

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 word length 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.

3.2.2.3.2 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 mutiplied (represented as two's complement integers), the codes are the same as for ROP.

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), C, O, Q

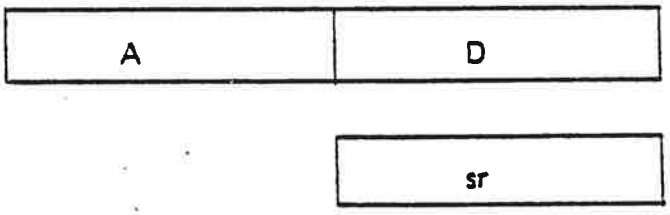
RDIV Integer inter-register divide Code: 141 600

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 → A ← (quotient) and D ← (remainder).

The sign of the remainder is always equal to the sign of 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 rightmost position.



Affected: (A), (D), Z, C, O, Q

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> is 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> is a new EXR <sr >. If this is attempted, the error indicator Z is set to one.
Affected: (IR), registers changed by the specified instruction.

MIX 3 — Multiply index by 3 — Code: 143 200

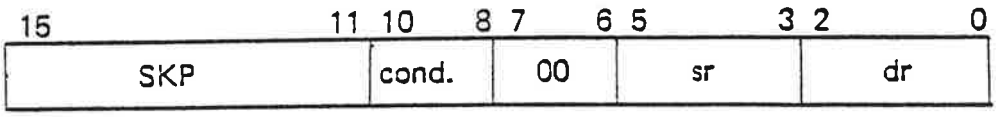
Format: MIX3

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)

3.2.2.4 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.

Note that bits 6 and 7 are both zero. Otherwise, the instruction would belong to the EXTended instructions. See Section 3.2.2.3.2.

The SKP conditions test the result of the arithmetic expression (dr) — (sr) which sets the four indicators:

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

The eight SKP conditions are as follows: (next page)

Mnemonic:	Condition 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) ⩾ 0. The contents of the source and destination registers are treated as signed numbers. Overflow is not taken care of.
GRE	0 1 0	S V o = 0	Greater or equal to. (dr) — (sr) ⩾ 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) ⩾ 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) — (sr) ≠ 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.
LST	1 1 0	s V o = 1	Less than. (dr) — (sr) < 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) < 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
⩽ Less than or equal

The programmer is advised to use the formats in the following examples when specifying a skip instruction. (The mnemonic IF and the number 0, which both have the value zero, are used for easy readability. They are not required.)

Comparing a register with zero:

SKP IF	DL	UEQ	0	Skip if L register ≠ 0
SKP IF	DX	GRE	0	Skip if X register ⩾ 0
SKP IF	DB	LSS	0	Skip if B register < 0
SKP IF	0	LSS	ST	Skip if T register > 0
SKP IF	0	GRE	SD	Skip if D register < 0

Comparing the arithmetic value of the contents of two registers:

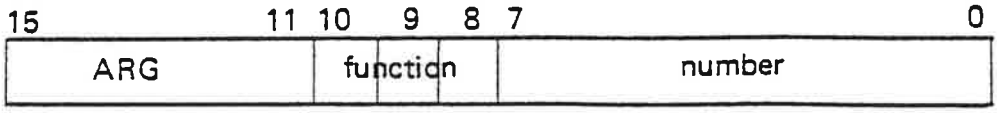
SKP IF	DD	EQL	SL	Skip if D register = L register
SKP IF	DT	UEQ	SX	Skip if T register ≠ X register
SKP IF	DB	LSS	SA	Skip if B register < A register or Skip if A register > B register
SKP IF	DX	GRE	SB	Skip if X register ⩾B register or Skip if B register ⩽ X register

Comparing two magnitude numbers:

SKP IF	DL	MGRE	ST	Skip if L register ⩾ T register or Skip if T register ⩽ L register
SKP IF	DB	MLST	SX	Skip if B register < X register or Skip if 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.

3.2.2.5 Argument Instructions

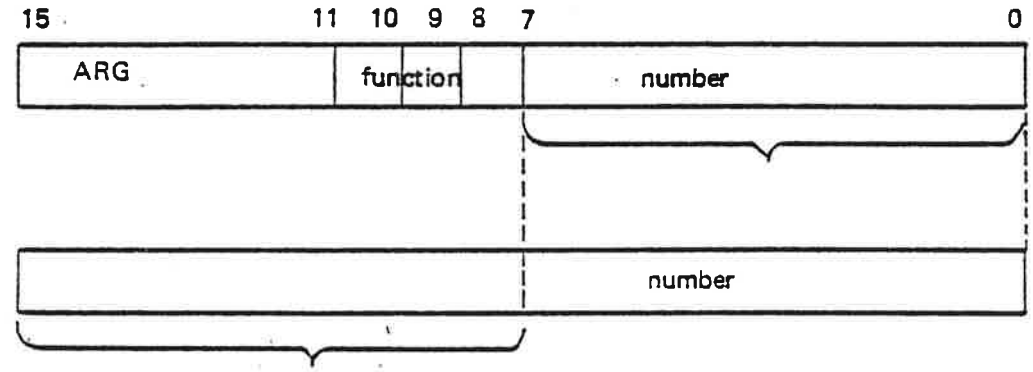


Argument instructions operate on registers. The function field is used to specify one out of eight argument instructions. The number field is used to specify the argument, a signed number ranging from —128 to 127.

Negative numbers are represented in 2's complement. The 8 argument number bits are extended to 16 bits using sign extension. The 8 argument number bits remain the 8 least significant bits of the 16 bits. The 8 most significant bits are extended with ones or zeros. When the number is positive, we extended with zeros. When the number is negative, we extend with ones.

When we have a set argument instruction all of the 16 bits are copied into the specified register.

When we have an add argument instruction all of the 16 bits are added to the 16 bits already in the specified register. See Figure 3.11.



these bits are extended with ones or zeros.
- Ones if the number is negative
- Zeros if the number is positive.

The extended argument number is set or added into one of the register B, A, T or X.

Figure 3.11: Sign Extension of the Argument Instruction.

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

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

Examples of argument instructions follow.

Example 1:

Set the contents of the T register equal to 13_8. Bits 8-15 becomes zero because of the sign extension:

```
SAT   13B
```

Example 2:

The contents of the B register becomes 177752_8 after execution of this instruction. Bits 8-15 becomes one because of the sign extension:

```
SAB   -26B
```

Example 3:

Add 3 to the contents of the X register. The contents of bits 8-15 depend on the previous content of the X-register:

```
AAX   3
```

Example 4:

Subtract 6 from the contents of the A register. The contents of bits 8-15 depend on the previous content of the X-register.

```
AAA   -6
```

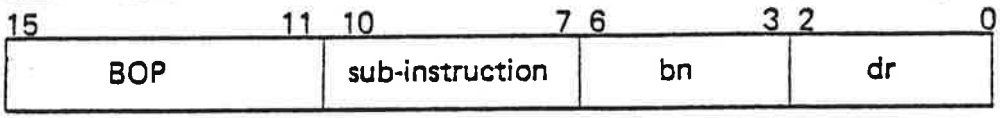
Example 5:

The contents of the A register will be $177\ 640_8$ after the execution of this instruction.Bits 8-15 becomes one because of the sign extension:

```
SAA   -140B
```

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

3.2.2.6 Bit Operation Instructions



BOP Bit Operation

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

The specified 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, *except* if dr = 0 the status register is specified.

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

Sixteen different sub-instructions 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 number 0, 10, 20, 30, ..., 170 because <bn> is contained in bits 3-6 of the BOP instruction.

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

SSPTM	Page table mode
SSTG	Rounding indicator for floating point operations
SSK	One bit accumulator indicator
SSZ	Error indicator
SSQ	Dynamic overflow indicator
SSO	Static overflow indicator
SSC	Carry indicator
SSM	Multi-shift link indicator

3.2.2.6.1 BIT SKIP INSTRUCTIONS

Four sub-instructions are available to test the setting of the specified bit.

BSKP ZRO <bn> <dr>	Skip next instruction if bit = 0.
BSKP ONE <bn> <dr>	Skip next instruction if bit = 1
BSKP BCM <bn> <dr>	Skip next instruction if bit_0 = K
BSKP BAC <bn> <dr>	Skip next instruction if bit = K

3.2.2.6.2 BIT SET INSTRUCTIONS

Four sub-instructions are available to set the specified bit.

BSET ZRO <bn> <dr>	bit ← 0
BSET ONE <bn> <dr>	bit ← 1
BSET BCM <bn> <dr>	bit ← bit_0, complement bit
BSET BAC <bn> <dr>	bit ← K

3.2.2.6.3 ONE BIT ACCUMULATOR INSTRUCTIONS

Eight sub-instructions 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
BSTC <bn> <dr>	bit ← K_0, K ← 1	Store complement and set
BLDA <bn> <dr>	K ← bit	Load
BLDC <bn> <dr>	K ← bit_0	Load complement
BANC <bn> <dr>	K ← bit_0 K	Logical AND complement
BORC <bn> <dr>	K ← bit_0 V K	Logical OR complement
BAND <bn> <dr>	K ← bit K	Logical AND
BORA <bn> <dr>	K ← bit V K	Logical OR

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 170B DT
```

Example 4:

Set bit 6 in the X register to one.

```
BSET ONE 60B DX
```

Example 5:

Copy A register bit 14 into X register bit 13.

```
BLDA 160B DA          % K ← A bit 14
BSET BAC 150B DX      % X bit 13 ← K
```

3.2.3 *System Control Instruction*

3.2.3.1 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.

3.3 *PRIVILEGED INSTRUCTIONS*

3.3.1 *General*

The instructions termed privileged instructions are available only to:

- programs running in system mode (rings 2 and 3)
- programs running in stop mode

3.3.2 *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 0-7. 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 from or written into.

The specification 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. The SRB instruction is privileged.

If the current program level is specified, the stored P register pointer to the instruction following SRB.
Affected:(EL), + 1 + 2 + 3 + 4 + 5 + 6 + 7
P · X T A D L STS B

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. The LRB instruction is privileged.

Affected: All the registers on specified program level are affected. Note: if the current level is specified, the P register is *not* affected.

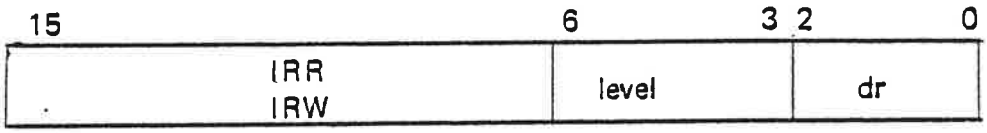
3.3.3 *Inter-level Register Instructions*

In the ND-100 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.

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

IRR Inter Register Read Code: 153 600

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

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

Example:

The instruction IRR 160 DP will copy the contents of the program counter on program level 14 into the A register on the 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 the current program level into one of the general registers on any level, including the current level. If the current level P register is specified, the IRW instruction will be a dummy instruction. If bits 0-2 are zero, the A register bits 0-7 are written into the status register on the specified level. The IRW instruction is privileged.

Example:

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

3.3.4 *Accumulator Transfer Instructions*

The internal registers in ND-100 which cannot be reached by the register instructions are controlled by the following four privileged instructions:

TRA	transfer to A register
TRR	Transfer from A register
MCL	Masked clear
MST	Masked set

The internal registers controlled by these instructions are described in Appendix D.

Transfer to A register:

TRA Transfer to A register Code: 150 000

Format: TRA <register name>

The registers which may be transferred to the A register with the TRA instruction are shown in the following table. 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. The TRA instruction is privileged.

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 zeros 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 in the register specfiied by <register name>. The registers which TRR may operate on are shown in the following table. The TRR instruction is privileged.

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 registers which MCL may operate on are shown in the following table. The MCL instruction is privileged.

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 the following table. The MST instruction is privileged.

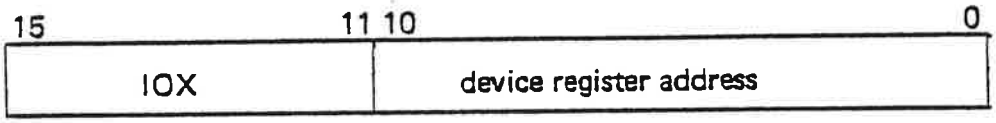
Register Name	$Code_8$	TRA	TRR	MCL	MST
PANS	0	X			
PANC	0		X		
STS	1	X	X	X	X
OPR	2	X			
LMP	2		X		
PSR	3	X			
PCR	3		X		
PVL	4	X			
IIC	5	X			
IIE	5		X		
PID	6	X	X	X	X
PIE	7	X	X	X	X
CSR	10	X			
CCL	10		X		
LCIL	11		X		
ACTL	11	X			
ALD	12	X			
UCILR	12		X		
PES	13	X			
PCR	14	X			
PEA	15	X			

PANS = Panel Status
PANC = Panel Control
STS = Status
OPR = Operator's Panel Switch Register
LMP = Operator's Lamp Register
PSR = Paging Status Register
PCR = Paging Control Register
PVL = Previous Program Level
IIC = Internal Interrupt Code
IIE = Internal Interrupt Enable
PID = Priority Interrupt Detect
PIE = Priority Interrupt Enable
CSR = Cache Status Register
CCL = Cache Clear
LCIL = Lower Cache Inhibit Limit Register
ACTL = Active Level
ALD = Automatic Load Descriptor
UCILR = Upper Cache Inhibit Limit Register
PES = Memory Error Status
PCR = Paging Control Register
PEA = Memory Error Address

3.3.5 *Input/Output Control Instructions*

IOX Input/Output Execute Code: 164 000

Format: IOX <device register address>



All transfers between the ND-100 and 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. Different devices will, however, require different numbers of device register addresses. Thus, the maximum number of physical devices that may be connected will depend on the specified configuration of devices.

Simple devices will usually require at least three different instructions (device register 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.

The IOX instruction is privileged.

Programming specifications and device register addresses for the different devices are found in separate manuals.

3.3.5.1 Extension of the Device Register Address

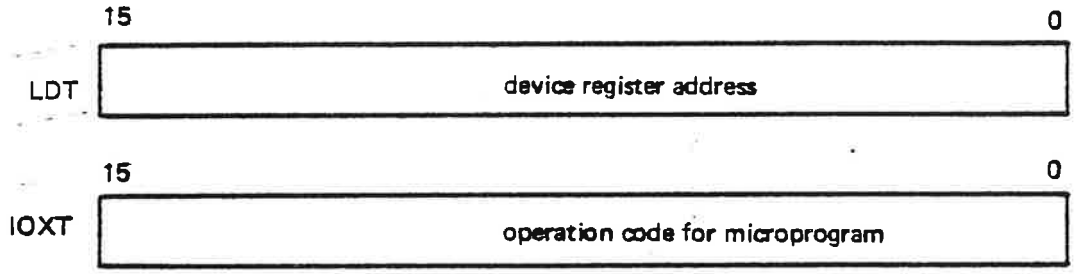
Since the number of peripheral devices delivered by Norsk Data is increasing, there is need for an extension of the device register address. That is done by the instruction:

Format: IOXT Code: 150 415

where the T register contains the 16 bits <device register address>. These 16 bits define an upper limit of 65536 device register addresses to the number of registers that may be adressed.

The device register address must be loaded into the T register before executing this instruction.

IOXT is privileged.



3.3.6 *System Control Instructions*

The following 11 instructions are denoted as the system control instructions:

ION	Interrupt system on
IOF	Interrupt system off
IDENT	Identify input/output interrupt
PON	Memory management on
POF	Memory management off
MON	Monitor call
WAIT	Wait or give up priority
SEX	Set extended address mode
REX	Reset extended address mode
PION	Memory management and interrupt system on
PIOF	Memory management and interrupt system off

Except for the MON instruction, all the system control instructions belong to the class of privileged instructions.

3.3.6.1 Interrupt Control Instructions

A full description of the interrupt system is presented in Section 2.2. A short summary is given here.

The ND-100 computer has a priority interrupt system with 16 program levels. Each program level has its own set of registers and status indicators. The priority increases — program level 15 has the highest priority, program level 0 the lowest.

The arrangement of the 16 program levels is as follows:

15	Reserved for extremely fast user interrupts
14	Internal hardware status interrupts
13 - 10	Vectored interrupts, maximum 2048 vectored interrupts
9 - 0	System programming and 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 by 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, i.e., the contents of the PL register.

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

When power is turned on, the power-up sequence will reset PIE and PID 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 IOX instruction will be the last instruction executed at the old program level, and the old program level will point to the instruction after ION. The interrupt indicator on the operator's display is lighted by the ION. The ION instruction is privileged.

IOF Interrupt system off Code: 150 401

Format: IOF

The IOF instruction turns off the interrupt system, i.e., the mechanisms 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 display is reset by the IOF instructions. The IOF instruction is privileged.

In addition, the following three registers are available for interrupt programming:

IIE Internal Interrupt Enable
IIC Internal Interrupt Code
PVL Previous Level causing internal hardware status interrupt

In ND-100 there are possibilities for 2048 vectored input/output interrupts where each physical input/output 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 service the input/output device causing the interrupt. Actually, there are four IDENT instructions, one to identify and serve 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 in bits 0 - 8 of the A register with bits 9 - 15 all zeros.

If the IDENT instruction is executed, but there is no device to serve, the A register is unchanged. An IOX error interrupt to level 14 will occur if enabled. Refer to the Interrupt System.

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. The IDENT instruction is privileged.

For ND-100 the identification codes are standarized for input/output devices delivered from Norsk Data.

3.3.6.2 Memory Management Control Instructions

A full description of memory management is given in Section 2.3. The paging system is controlled by the following privileged instructions:

PON — Memory management on — Code: 150 410

Format: PON

This instruction should only be used with the interrupt system on and with the necessary internal hardware status interrupts enabled. The page index tables and the PCR registers should be initialized before PON is executed. The PON instruction is privileged.

The instruction executed after the PON instruction will use the page index table specified by PCR.

POF — Memory management off — Code: 150 404

Format: POF

This instruction is a privileged instruction and may only be executed if the ring bits are 11 (3) or 10 (2).

The instruction will turn off the memory management system, and the next instruction will be taken from a physical address in lower 64K, the address following the POF instruction.

The CPU will be in an unrestricted mode without any hardware protection features, i.e., all instructions are legal and all memory "available". POF is privileged.

PION — Memory management and interrupt system on — Code: 150 412

Format: PION

The PION instruction will turn on both the memory management system and the interrupt system. Refer to ION and PON. PION is privileged.

PIOF — Memory management and interrupt system off — Code: 150 412

Format: PIOF

The PIOF instruction will turn off both the memory management and interrupt systems. Refer to IOF and POF. PIOF is privileged.

SEX — Set extended address mode — Code: 150 406

Format: SEX

The SEX instruction will set the paging system in a 24 bit address mode instead of a 19 bit address mode. A physical address space up to 16 M words will then be available.

Bit number 13 in the status register is set to one, indicating the extended address mode. SEX is privileged.

REX — Reset extended address mode — Code: 150 407

Format: REX

The REX instruction will reset the extended address mode (24 bits) to normal address mode (19 bits). This implies that 512K words of physical address space is now available.

Bit number 13 in the status register is reset, indicating normal address mode. REX is privileged.

OPCOM — Operator's Communication — Code: 150 400

Format: OPCOM

The OPCOM instruction has the same function as pushing the OPCOM button on the front panel. OPCOM is privileged.

3.3.6.3 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 instruction after the WAIT.

In this programmed wait, the RUN lamp on the front panel is switched off. To start the program in the instruction after the WAIT, type ! (exclamation mark) on the console terminal.

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 not 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.

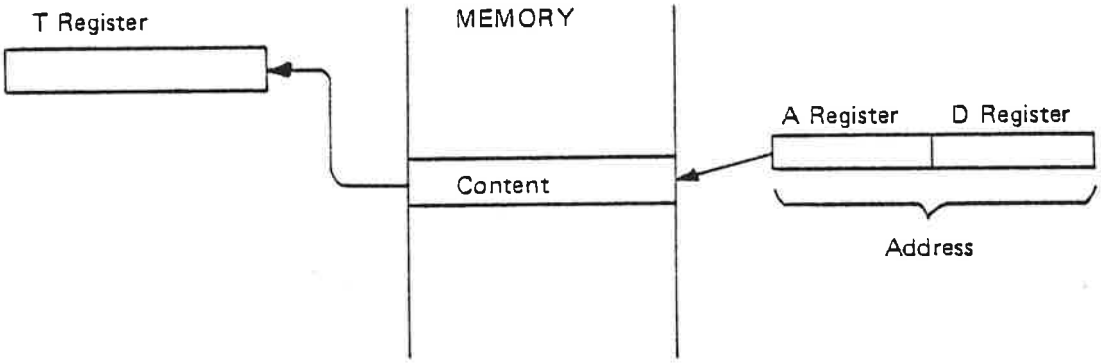
Note that it is legal to specify WAIT followed by a number less than 400_8. This may be useful to detect in which location the program stopped. The WAIT instruction is displayed at the operator's panel, IR register. The WAIT instruction is privileged.

3.3.7 *Examine and Deposit*

EXAM Examine Code: 150 416

Format: EXAM

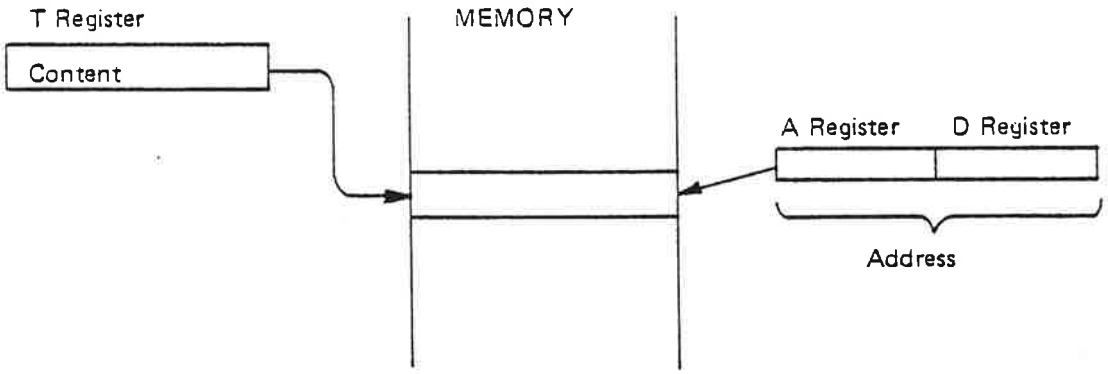
After execution of this instruction, the T register will be loaded with the content of the physical memory location, pointed to by the A and D register. EXAM is privileged.



DEPO Deposit Code: 150 417

Format: DEPO

This instruction will store the content of the T register into the physical memory location, pointed to by the A and D register. DEPO is privileged.

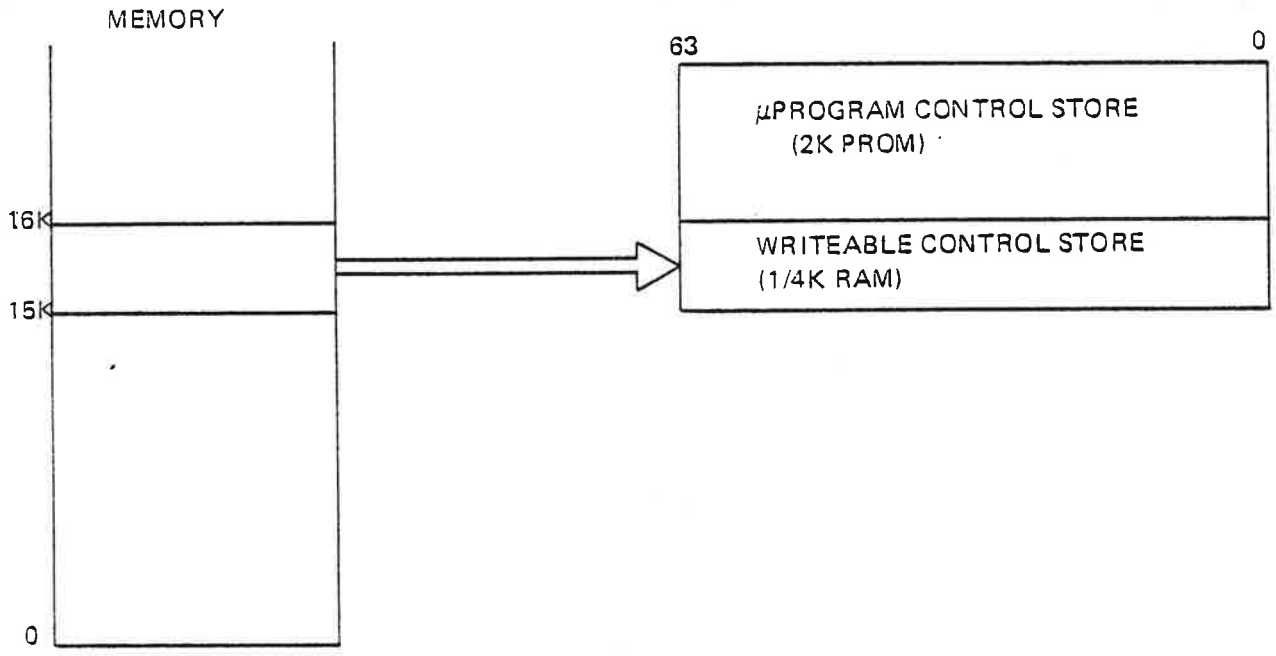


3.3.8 *Load Writeable Control Store*

LWCS Load Writeable Control Store Code: 143 500

Format: LWCS

By executing this instruction, the 256 word by 64 bit RAM writeable control store will be loaded with the content from memory locations with address from 15 - 16 K in main memory. Microprogram addresses from 4000_8 to 4377_8 will then be accessible. When the instruction is finished, all microprogram addresses are legal and illegal instruct. ROM out of range interrupt will never occur. LWCS is privileged.



Four ordinary 16 bit memory locations are required to make one 64 bit location in Writeable Control Store. Therefore, 1K is needed from main memory.

3.3.9 *Customer Specified Instructions*

The remaining free codes may be used to extend the ND-100 instruction set. The codes that can be used for customer specified instructions are as follows:

1402XX	1403XX	1405XX	1407XX
1411XX	1413XX	1415XX	1417XX
1421XX	1423XX	1425XX	

These 11 instructions have the following entry points in writeable control store:

1402XX	CUST 1	Entry point in μ program	4001_8
1403XX	CUST 2	Entry point in μ program	4002_8
1405XX	CUST 3	Entry point in μ program	4003_8
1407XX	CUST 4	Entry point in μ program	4004_8
1411XX	CUST 5	Entry point in μ program	4005_8
1413XX	CUST 6	Entry point in μ program	4006_8
1415XX	CUST 7	Entry point in μ program	4007_8
1417XX	CUST 10	Entry point in μ program	4010_8
1421XX	CUST 11	Entry point in μ program	4011_8
1423XX	CUST 12	Entry point in μ program	4012_8
1425XX	CUST 13	Entry point in μ program	4013_8

If these instructions are not implemented, they will cause an internal hardware status interrupt to level 14 (illegal instruction).

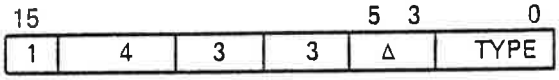
All micro instruction codes are available for new customer specified instructions. For further information about programming in WCS, contact Norsk Data.

3.3.10 *Physical Memory Read/Write Instructions*

When the extended address mode (controlled by the instructions SEX and REX) is used, 7 special, privileged instructions are useful to read/write physical memory locations independent of whether paging is ON or OFF. They will affect

the page tables if the address is within the page table range.

3.3.10.1 Format of Instructions:



Δ is the displacement (bit 3-5) added to the X-reg. to give the effective location (EL).

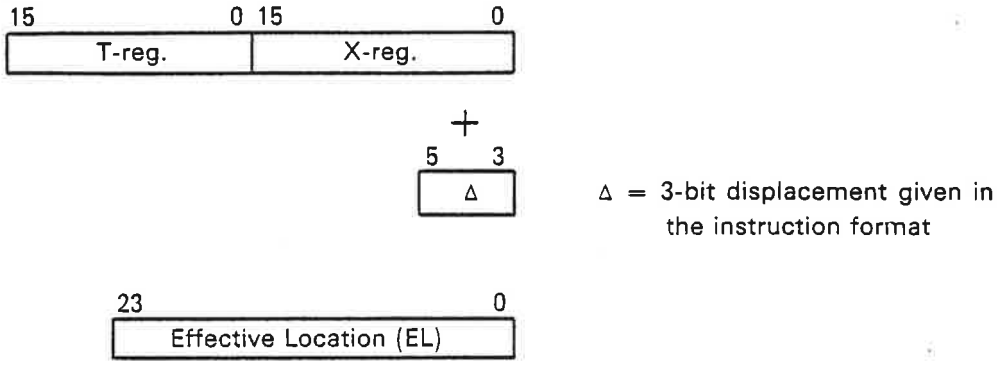
Type:	Name:	Effect:
0:	LDATX.	A: = (EL)
1:	LDXTX.	X: = (EL)
2:	LDDTX.	A: = (EL), D: = (EL + 1)
3:	LDBTX.	B: = 177000 V ((EL) + (EL)) (V = inclusive OR)
4:	STATX.	(EL): = A
5:	STZTX.	(EL): = 0
6:	STDTX.	(EL): = A, (EL + 1): = D

In computers with microprogram versions 015xx A-J (48-bit) or 026xx A-F (32-bit), the LDBTX-instruction must be followed by a word containing 177777.

In later versions (015xx K→ or 026xx G→) the 177777-instruction is not necessary.

3.3.10.2 Addressing:

All the 7 instructions generate a 24-bit effective location (EL). The effective location is calculated from the T- and X-register plus a 3-bit displacement contained in the instruction.



The 3-bit displacement is added to the X-register. If the X-register plus the displacement give a carry, the carry is dropped and *not* added to the T-register. This means that the T-register always determines which 64 K memory area to address.

3.4 *INSTRUCTIONS IN THE «COMMERCIAL EXTENDED» (CE) OPTION*

By expanding the microprogram PROM of the ND-100 CPU, a number of instructions are introduced. The instructions are collectively known as the «Commercial Extended» option.

3.4.1 *Decimal Instructions*

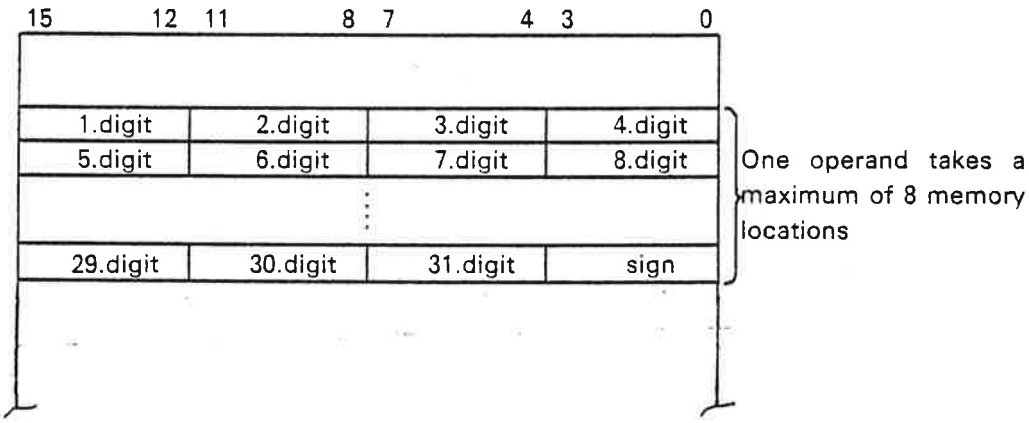
3.4.1.1 DATA FORMATS FOR DECIMAL INSTRUCTIONS

3.4.1.1.1 Packed Decimal Number (BCD-Coded Numbers)

One decimal digit is represented by 4 binary digits (bits). Two decimal digits are placed next to each other to form a byte (8 bits). Two such bytes are placed in each memory location.

The decimal digits form operands. Maximum length of an operand is 31 digits plus a sign byte. This occupies eight 16 bit words in the memory.

Memory



Each decimal digit is represented by the following binary digit:

Decimal Digit	Binary Digit
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

The codes 1010 - 1111 do not represent digits. These codes are used to represent the decimal digit's sign (plus or minus). This is done in the following way:

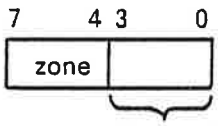
1010, 1100 and 1110 represent plus.
1011 and 1101 represent minus.
1111 represents unsigned (treated as plus).

All sign codes are allowed, but only 1100 (for plus) and 1101 (for minus) are used in the instructions.

3.4.1.1.2 ASCII Coded Decimal Number

In the ASCII format each decimal digit occupies one byte. The four high-order bits of this byte are called the zone. The four low-order bits, the numeric are occupied by the decimal digit, and are encoded the same way as a packed decimal digit. The most significant bit in the byte is the parity-bit. This bit is neither tested nor set in the instruction.

Decimal Digit	ASCII CODE
0	0 0 1 1 0 0 0 0
1	0 0 1 1 0 0 0 1
2	0 0 1 1 0 0 1 0
3	0 0 1 1 0 0 1 1
4	0 0 1 1 0 1 0 0
5	0 0 1 1 0 1 0 1
6	0 0 1 1 0 1 1 0
7	0 0 1 1 0 1 1 1
8	0 0 1 1 1 0 0 0
9	0 0 1 1 1 0 0 1



The parity bit (bit 7) is always 0.

These bits have the same value as the binary digits in Section 3.4.1.1.1.

A decimal operand in this format may have four different sign representations:

1. Separate trailing, the byte to the right of the last significant digit contains the sign. Sign is represented by the ASCII code of + (53 octal) or — (55 octal).

2. Separate leading, ASCII code of the sign occupies the first byte (left-most).

3. Embedded trailing, the right-most byte occupies both the least significant digit and the sign.

4. Embedded leading, the first byte (left-most) in the number contains both the sign and the left-most digit.

   When the sign is embedded, the following codes are used to represent the right-most / left-most digit and sign:

Positive operand:		Negative operand:	
0 = 173	0 1 1 1 1 0 1 1	0 = 175	0 1 1 1 1 1 0 1
1 = 101	0 1 0 0 0 0 0 1	1 = 112	0 1 0 0 1 0 1 0
2 = 102	0 1 0 0 0 0 1 0	2 = 113	0 1 0 0 1 0 1 1
3 = 103	0 1 0 0 0 0 1 1	3 = 114	0 1 0 0 1 1 0 0
4 = 104	0 1 0 0 0 1 0 0	4 = 115	0 1 0 0 1 1 0 1
5 = 105	0 1 0 0 0 1 0 1	5 = 116	0 1 0 0 1 1 1 0
6 = 106	0 1 0 0 0 1 1 0	6 = 117	0 1 0 0 1 1 1 1
7 = 107	0 1 0 0 0 1 1 1	7 = 120	0 1 0 1 0 0 0 0
8 = 110	0 1 0 0 1 0 0 0	8 = 121	0 1 0 1 0 0 0 1
9 = 111	0 1 0 0 1 0 0 1	9 = 122	0 1 0 1 0 0 1 0

A decimal operand in ASCII format has maximum 32 digits, maximum field length is 16 words, 32 bytes.

3.4.1.2 The Decimal Instructions

The decimal operands reside in main memory only. They occupy fields that may start at any byte address. The decimal operands must be right adjusted, i.e. the least significant digits (and sign) are placed right adjusted from the last byte of the field.

A decimal operand is specified by a two words descriptor, D1 and D2.
The two words have the following formats:

D1: Bit 0-16 give the decimal operand's word address in the memory.

D2: Bit 15. This bit specifies whether the operand start in the left byte or the right byte.
Bit 15=0, left byte
Bit 15=1, right byte

Bit 14. Not used.

Bit 11-13. These bits specify the sign representation when the operand is in ASCII format.

13 12 11
0 0 0 : embedded trailing (default)

0 0 1 : separate trailing

0 1 0 : embedded leading

0 1 1 : separate leading

1 0 0 : unsigned

Bit 13 is also used to represent an unsigned number in BCD-representation (the sign-code is 1111).

Bit 10. This bit is used to specify rounding. If the least significant digits are lost during shift, and the last digit shifted out of the fields is ⩾ 5, a one is added to the shifted operand.
Bit 10=0, rounding on
Bit 10=1, rounding off

Bit 5-9. These bits give the position of the decimal point in the field. The number in these bits can be a positive decimal number from 0 to 31. Zero means that the decimal position is to the right of the least significant digit. The number has to be less than the field length. (It is not legal to specify a point outside the field.) The decimal point position is used to compute the shift count in the shift instruction (SHDE).

Bit 0-4. These bits give the field length of the operand in nibbles (4 bits) or bytes. The field length includes sign. The field length is in nibbles when you have packed decimal number (BCD) and in bytes when you have ASCII coded decimal numbers. The field length is a maximum of 32 nibbles/bytes.

Specification of the operands for all the decimal instructions:

Descriptor of the first operand in A,D registers.
Descriptor of the second operand in X,T registers.

Before any operation is performed, the operands are read into the CPU's registerfile. Then the operation is performed, before the result is written back to memory. This is why overlap is not tested in the ND-100 CIS (Commercial Instruction Set).

ADDD Add decimal Code:140 120

Format:ADDD

The second operand is added to the first operand and the sum is placed in the first operand's location. If necessary, high-order zeroes are applied for either operand.

When the first operand field is too short to contain all significant digits of the sum, a decimal overflow occurs.

Overflow has two possible causes:

a) A carry from of the most significant digit position in the result field.

b) Oversized result, which occurs when the second operand field is larger than the first operand field and significant result digits are lost. The field sizes alone are not an indication of overflow.

This instruction does not give automatic scaling as in N10-CIS, so the operands have to be aligned before entering this instruction, for examle, by the shift instruction (SHDE).

If bit 13 in D2 in the destination descriptor is set, the sign in the result field is 1111 (unsigned).

An empty operand (field-length equal 0) is treated as a positive zero.

Error indication:

```
ADDD        % instruction
ERROR       % error return, overflow
CONTINUE    % skip, OK return
```

SUBD Subtract decimal Code:140 121

Format:SUBD

The second operand is subtracted from the first operand and the difference is placed in the first operand's location.

The subtract decimal is similar to add decimal, except that the sign of the second operand is changed from positive to negative, or from negative to positive after the operand is read from memory, but before the arithmetic operation. A zero difference can have both positive and negative sign.

Error (overflow) is indicated by error return (see ADDD).

COMD Compare decimal Code: 140 122

Format:COMD

The first operand is compared with the second operand. The result is placed in the A-register. If the operands are unequal in length, the shorter is extended with zeroes. None of the operands are changed as a result of the operation.

The positions of the decimal points are not taken into account when the two digits are compared. Therefore the operands must be aligned before the operation, as in ADDD/SUBD. Use the instruction SHDE to align the operands.

One of the two fields is extended with zeroes so the two fields have the same number of digits.

An unsigned number is treated as positive, positive and negative zeroes are equal. An empty operand is treated as a positive zero.

Result in A-reg.:

operands equal	0
first operand greater	1
second operand greater	—1

This instruction will always have a skip return (no error condition).

PACK Convert to packed decimal Code:140 124

Format:PACK

The format of the first operand is changed from ASCII Coded Decimal Number (unpacked) to Packed Decimal Number (packed), and the result put in the second operand location. The right four bits in the ASCII code (the numeric) are used for the digits. The specified sign representation in the unpacked format is converted to 14 octal (+) or 15 octal (-), unsigned is converted to plus, unless bit 13 in D2 of the destination descriptor is set. If so the sign code in the destination field will always be 17 octal (unsigned).

The conversion is done one digit at a time, and the destination is filled from the least significant position (sign position).

The sign and digits of the first operand are checked for valid codes, and illegal codes are reported.

If necessary, the second operand field is extended with high-order zeroes. If the second operand field is too short to contain all significant digits of the first operand, the remaining digits are ignored, causing overflow.

Both illegal code and overflow are reported by an error-return (no skip), with an error code placed in D-register bits (0-4). The first detected error is reported.

Error codes: (1 forbidden overlap - not used in the ND-100 CIS.)
2 illegal code
3 overflow

After error return caused by illegal code, both A-reg. and D-reg. bit 15 point to the byte containing the illegal code.

UPACK Convert to unpacked decimal Code:140 125

Format:UPACK

The format of the first operand is changed from Packed Decimal Number (packed) to ASCII Coded Decimal (unpacked), and the result is placed in the second operand's location.

The digits of the packed operand are tested for illegal codes and supplied with zones with coding 0011 (no parity set). The sign of the packed operand is not tested for legal code, but is treated as plus if bit 0 is 0, and minus if bit 0 is 1 (except for the code 1111, which is unsigned and treated as plus). The sign is then converted to the specified representation in the unpacked format.

If necessary, the second operand is extended with high-order zeroes (ASCII). The conversion starts in the least significant postion (sign) and the fields are prosessed one word at the time. If the second operand field is too short to contain all significant digits of the first operand, the remaining digits are ignored. This is detected as overflow. The error-code reported back is the one detected first, and the same as in PACK. After error return caused by illegal code, the A-reg. and D-reg. bit 15 point to the byte containing the illegal code, also as in PACK.

SHDE Decimal shift Code:140126

Format: SHDE

Operand one is moved to the operand two field with its digits offset (shifted) to the left or right.

The shift count is computed as the difference in decimal position of the two operands.

When shifting left the second operand is generated from left to right; for right shift the second operand field is generated from right to left.

Shift input will always be zero. The sign is set to either + (14 octal) or — (15 octal) depending on what sign the source operand have, or 17 octal (unsigned) if bit 13 in D2 is set. Digits shifted out of the operand field are lost. If high order digits different from zero are lost during left shift, this is indicated by an error return (no skip).

Rounding is performed if bit 10 in D2 of the destination operand is set. This means that a 1 is added to the operand if the last digit shifted out of the field is $\geqslant 5$.

3.4.2 *Stack Handling Instructions*

Programs written in high level languages such as FORTRAN and PLANC, execute faster if they use the specially provided stack handling instructions in the

CE-option.

3.4.2.1 Data Structure Operated Upon by the Instructions

The B-register will always point to a «stack-frame» containg the following information.

B-reg. -200: LINK points to the next instruction in case a LEAVE-instruction is executed.

B-reg. -177: PREVB points to the previous stack frame on the stack.

B-reg. -176: STP points to the next stack frame on the stack.

B-reg. -175: SMAX points to the top of the stack. This is used to detect stack overflow.

B-reg. -173: ERRCODE is filled with the A-register's content each time an ELEAV-instruction is executed.

In addition to these addresses which are used by the microprogram, the stack will usually contain a number of addresses accessed by other instructions.

INIT Initialize stack Code:140 134

Usage:

INIT:
Next address: Stack demand (words)
Next address: Address of stack start (words)
Next address: Maximum stack size (words)
Next address: Flag
Next address: Not used by the microprogram

Error return address
Normal return address

Effect:
L + 1 = = > Address of stack start (LINK)
Address of stack start + 200 = = > B-reg.
Old B-reg. = = > B - 177 (PREVB)
Address of stack start + Maximum stack size = = > B - 175 (SMAX)
Stack demand - 172 + B = = > B - 176 (STP)

If the Flag-word bit 0 is different from the Status register bit 0, you have error return.
If you have stack overflow, you have error return.
In all other cases you have normal return.

ENTR Enter stack Code 140 135

Usage:

ENTR
Next address: Stack demand (words)

Error return address
Normal return address

Effect:
(B - 176) + 200 = = > B
L + 1 = = > B - 200 (LINK)
Old B = = > B - 177 (PREVB)
(Old B - 175) = = > B - 175 (SMAX)
Stack decimal - 172 + B = = > B - 176 (STP)

If you have stack overflow, you have error return.
In all other cases you have normal return.

LEAVE Leave stack Code:140 136

Format: LEAVE

Effect:
(B - 200) = = > P (LINK)
(B - 177) = = > B (PREVB)

ELEAV Error leave stack Code:140 137

Format: ELEAV

Effect:
(B - 200) - 1 = = > B - 200 (LINK)
A = = > B - 173 (ERRCODE)
(B - 200) = = > P (LINK)
(B - 177) = = > B (PREVB)

4 OPERATOR'S INTERACTION

4.1 *CONTROL PANEL PUSH BUTTONS*

When the panel key is unlocked, the panel push buttons are active and have the following effect:

MCL — This is the MASTER CLEAR button used to force the computer system into a defined initialized state. First, the red and green indicator lamps on the CPU board will light up. Then the microprogram is forced to execute the master clear routine. This will also be executed when the MACL command is given to MOPC (refer to Section 4.2.2.1.1), when the CPU goes through the power up sequence, or when the bus line called MCL is activated by an interface.

The master clear routine turns off the green indicator lamp, then the PIE register is cleared. The paging and interrupt systems are turned off. The paging system is set in REX mode. Subsequent memory examine functions with MOPC are set to 24 bit physical examine mode. The CPU self test microprogram is executed. If no errors are found, the green indicator lamp is lit, and the terminal interface on the CPU board (the MOPC terminal) is initialized to receive and transmit 7 bits and even parity. Parity is not checked by MOPC on input. An interrupt level change to level 0 is then executed. After this the CPU will be in stop mode.

STOP — This push button has the same effect as giving the STOP command to MOPC. The CPU will enter stop mode and MOPC will be active.

LOAD — This push button has the same effect as writing $ or & to MOPC. Its exact effect is determined by the setting of the ALD thumb-wheel switch on the CPU board.

OPCOM — OPCOM is always operative in stop mode. When the machine is running, pressing this button will allow the operator to use the CPU board terminal for operator communication. When the CPU is running, it will enable MOPC to read input from the terminal interface located on the CPU board. It will also inhibit input interrupts from this terminal, and disable the transfer of data from the terminal interface to any macro program (main memory program). The terminal interface will be in this state until the escape character is typed, or the CPU is stopped and restarted.

When MOPC is entered a # is printed at the beginning of each line.

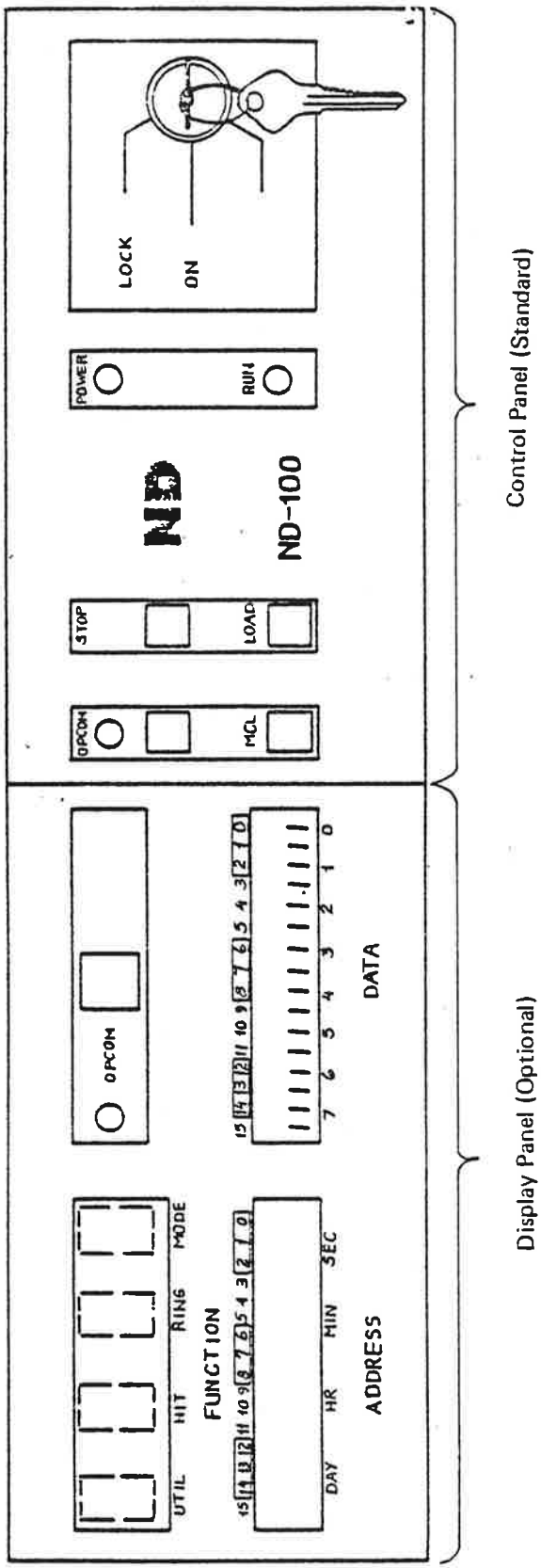


Figure 4.1: ND-100 Front Panel

4.1.1 *The Panel Lock Key*

The Panel Lock Key has three positions:

1. LOCK

   When placed in this position, the operator's panel control switches are disabled. This is the normal position for an operating machine. Main power is applied to the computer.

   Note: Automatic restart may be initiated after power failure only if the lock key is switched in this position.

2. ON

   In this position the panel switches can be operated. Main power is applied to the computer.

3. STAND-BY

   In this position the main power is disabled. Stand-by voltage is applied to memory and display. This position will not be present (or valid) on machines delivered from January 1980.

4.1.2 *Status Indicators*

POWER ON

Indicates that +5V is present in the rack.

RUN

Indicates that the CPU is running.

OPCOM — Operator Communication.

Indicates that the operators communication microprogram is running. This light may also be lit in RUN mode by pressing the OPCOM button. (OPCOM and RUN are lit at the same time). The OPCOM light will always be lit when the computer is not running.

Note: When OPCOM and RUN are lit at the same time, input from the console terminal will only interact with the OPCOM microprogram. Output to console may come from OPCOM or the active program.

4.2 *MICROPROGRAMMED OPERATOR'S COMMUNICATION*

4.2.1 *General Considerations*

The ND-100 has a microprogram in the read only memory for communication between the operator and the machine. This program is called MOPC (Microprogrammed Operator's Communication) and is used for operational control of the ND-100. It includes such functions as memory and register examine and deposit, breakpoint control, bootstrap loading, etc.

Whenever entered, MOPC will perform the necessary communication with the terminal connected to the current loop interface on the CPU printed circuit board. This terminal will be shared as output device between MOPC and other possible programs. As input device MOPC will receive input from the terminal as long as the OPCOM lamp on the operator's panel is lit.

MOPC will never wait if the terminal is not ready for the transmission of characters. Instead, it will start executing the STOP routine or the running program. MOPC will then be dormant until next time it is entered, and continue with the tasks it had to postpone. The maximum time spent in MOPC is 20 μs. If MOPC does not have any activity to sustain on the terminal, it will use 6 μs every time it is entered.

The ND-100 operator's communication includes bootstrap 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 characters:

Characters legal in STOP or RUN:

Character:	*Use:*
0 - 7	Octal digits used to specify addresses and data.
A - Y	Letters used to specify commands and register names. Letters typed in succession are acted upon when CR (carriage return) or / is typed. Different letter combinations may have the same effect because of a scrambling algorithm used to pack the letters.

@ or (space)	All characters written before this character are ignored (break character).
<	Used to separate lower and upper bounds in dump commands.
/	Specifies memory or register examine.
↵ (carriage return)	Ends a line. Used to terminate commands or to perform a register or memory deposit function.
*	This character will cause the address of the last examined memory address to be printed.
"escape"	Terminates the communcation between the CPU board terminal and MOPC. This character has no effect if the CPU is in STOP mode.

Characters only legal in STOP:

Character:	*Use:*
!	Start program in main memory command.
Z	Single instruction command.
& or $	Bootstrap load command.
.	Breakpoint command.
"	Manual instruction command.
#	Start microprogrammed memory test.

All other characters are answered with a ?, and characters written before the erroneous character will be forgotten (as if "space" had been typed).

4.2.2 *Control Functions (Does not affect display)*

4.2.2.1 System Control

4.2.2.1.1 MASTER CLEAR

When MACL ↓ is written to MOPC, the CPU microprogram will execute the master clear routine. The effect of this routine is described in the section on Panel Pushbuttons - 4.1.

4.2.2.1.2 STOP

When STOP ↓ is written to MOPC, the CPU will stop execution of the program in main memory. No level change will be performed and program execution can be continued by typing the exclamation mark character.

4.2.2.1.3 ALD LOAD

In the following table the different columns signify:

ALD — Setting of the ALD thumbwheel switch on the CPU module.

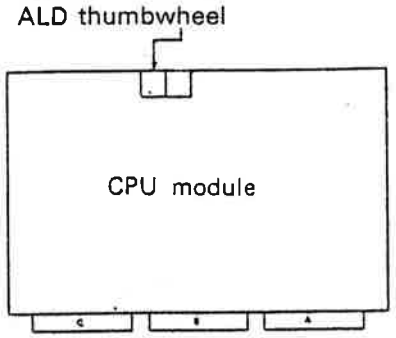
I12 — Corresponding value of the internal register number 12.

POW OK — Indicates the action performed when the panel key is locked and power comes on (or hardware master clear is finished), and standby power has been on all the time since power last went off.

POW NOK — Indicates the action performed when the panel key is locked and power comes on (or hardware master clear is finished), and standby power has been missing for some time since power last went off.

LOAD — Indicates the action performed if the load button is pressed, or $ & is written to MOPC.

ALD	I12	STB POW OK	STB POW NOK	LOAD
15	0	Start in address 20	Stop	Nothing
14	1560	Start in address 20	Binary load from 1560	Binary load from 1560
13	20500	Start in address 20	Mass storage load from 500	Mass storage load from 500
12	21540	Start in address 20	Mass storage load from 1540	Mass storage load from 1540
11	400	Start in address 20	Binary load from 400	Binary load from 400
10	1600	Start in address 20	Binary load from 1600	Binary load from 1600
9		Start in address 20		
8		Start in address 20		
7	100000	Stop	Stop	Nothing
6	101560	Binary load from 1560	Binary load from 1560	Binary load from 1560
5	120500	Mass storage from 500	Mass storage load from 500	Mass storage load from 500
4	121540	Mass storage from 1540	Mass storage load from 1540	Mass storage load from 1540
3	100400	Binary load from 400	Binary load from 400	Binary load from 400
2	101600	Binary load from 1600	Binary load from 1600	Binary load from 1600



position of the ALD thumberwheel on the CPU module

4.2.2.1.4 GENERAL LOAD

Binary load is started by typing:

<physical device address> & or <physical device address> $

Loading will take place from the specified device. This device must conform with the programming specifications of either Teletype or tape reader. The device address is the lowest address associated with the device. Binary load will be performed if & or $ is written (or the LOAD button is pressed) and the switch selected ALD has bit 13 equal to "0".

4.2.2.1.5 LEAVE MOPC

ESCAPE

If the ESCAPE key is pressed and the CPU is running, MOPC will be left, and subsequent input from the terminal will be routed to main memory programs. MOPC will be entered again by pushing the OPCOM button on the panel or by executing the instruction 150400 (OPCOM).

4.2.2.2 Program Execution

4.2.2.2.1 START PROGRAM

Format:

xxxxxx !

The machine is started in the address given by the octal number. The address will be physical or virtual depending on whether the paging system is on or off.

4.2.2.2.2 CONTINUE A PROGRAM

!

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 the MACL command typed).

4.2.2.2.3 SINGLE INSTRUCTION

xxxxxxZ

A single Z character will cause one main memory instruction (or one interrupt level change) to be executed. If an octal argument is specified, the specified number of instructions are executed, after which stop mode is entered again. Page faults, protect violations and interrupt level changes are executed correctly, but are counted as extra instructions. An extra overhead of approximately 3 μs is introduced between each instruction when the CPU is in this semi-RUN mode.

4.2.2.2.4 INSTRUCTION BREAKPOINT

xxxxxx.

This command starts execution in the same semi-RUN mode as described in Section 4.2.2.2.3. When the program address xxxxxx is reached, execution stops before that address is executed, and a "." is printed. If the specific address is never reached, the semi-RUN mode continues until a character other than 0-7 or A-Y is typed.

4.2.2.2.5 MANUAL INSTRUCTION

xxxxxx''

This command starts continuous execution of the instruction specified as argument. The execution stops when a character other than 0-7 or A-Y is typed.

Example:

150410'' is an easy way to turn on the paging system.

4.2.2.2.6 SINGLE I/O INSTRUCTION FUNCTION

xxxxxxIO/

This function exeuctes an IOX instruction with xxxxxx as device number. The output data is taken from the operator's register OPR (see Section 4.2.3.2.5). Returned data is printed after the slash and not stored anywhere. No working registers are affected.

4.2.2.3 Miscellaneous Functions

4.2.2.3.1 INTERNAL MEMORY TEST

xxx#

When the # character is typed, memory test of the addresses between the B register (lower limit) and the X register (upper limit) is performed in segment xxx. If the test is successful, # is typed when finished. If the test is unsuccessful, ? is typed and the test stops at the failing address. The registers then contain the following information:

T: Failing bits
P: Failing address
D: Error pattern
L: Test pattern
B: Start address
X: Stop address

4.2.2.3.2 DELETE ENTRY

When @ or (space) is typed, all characters written before this character are ignored.

4.2.2.3.3 CURRENT LOCATION COUNTER

*

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

4.2.3 *Monitor Functions (Also shown on Display)*

4.2.3.1 Memory Functions

4.2.3.1.1 PHYSICAL EXAMINE MODE

E↵

Subsequent examine will be in physical memory with a 24 bit address. Default mode after master clear.

4.2.3.1.2 VIRTUAL EXAMINE MODE

nE↵

This command will change the xamine mode for subsequent memory examine functions. n is in the range 0-3 and specifies the page table via which the examine address shall be mapped. Page fault and memory protect violation are ignored and physical page 0 used instead.

4.2.3.1.3 MEMORY EXAMINE

Format:

xxxxxx /

The octal number before the character "/" specifies the memory address.

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

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

If the paging system is used, examine mode may be selected by an E command (see Section 4.2.3.1.1 and 4.2.3.1.2). If virtual examine is specified page faults and protect violations are ignored. In this case, <octal number> specifies a virtual address. If physical examine is specified, <octal number> may contain up to 24 bits of physical address.

Example:

```
717/003456                  % Examine address 717

717/003456 ↵                % Examine address 717
003450 ↵                    % and 720
000013                      % and 721
```

4.2.3.1.4 MEMORY DEPOSIT

Format:

xxxxxx ↵

After a memory examine, the contents of the memory cell may be changed by typing an octal number terminated by CR. If the CPU is running, "DEP" must be written between the number and CR.

Example:

```
717/003456   3475 ↵       % The contents of
003450   1700 ↵            % address 717 is changed
000123 ↵                   % From 3456 to 3475 and 720
123456                     % is changed from 3450 to 1700.
                           % 721 contains 123 and remains
                           % unchanged.
```

4.2.3.1.5 DEPOSIT RULES

Content is only changed by zzzzzz↵ in STOP mode and by zzzzzzDEP↵ in STOP or RUN mode.

Content is unchanged by ↵in STOP or RUN mode and zzzzzz↵ in RUN mode (? is answered).

4.2.3.1.6 MEMORY DUMP

xxxxxx < yyyyyy↵

The contents of the memory addresses between xxxxxx and yyyyyy are printed out, with 8 addresses per line. The dump is taken from the 64K area last addressed by a preceding memory examine function. A memory examine function should always be done before a memory dump. The dumping will stop if any key is pressed.

4.2.3.2 Register Functions

4.2.3.2.1 REGISTER EXAMINE

Format:

xx Ry/

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

The second octal number (y) specifies which register to examine on that level. The following codes apply:

0 Status register, bits 0-7
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.

Example:

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

Instead of the notation Ry, it is possible to address registers by their names. The names are single letter names, namely: S, D, P, B, L, A, T, X corresponding to R0-R7 respectively.

4.2.3.2.2 REGISTER DEPOSIT

Format:

xxxxxx↵

After a register examine, the contents of the register may be changed by typing an octal number terminated by CR. If the CPU is running, "DEP" must be written between the number and CR.

Examples:

A/ 123456 54321↵	% Contents of A register on level 0 % is changed to 054321
7P/ 000044 55↵	% Contents of P register on level 7 % is changed to 000055

4.2.3.2.3 REGISTER DUMP — RD

xx < yy RD ↵

The contents of the working registers in register blocks xx to yy are printed out, with one register block per line. The registers are printed in the following order: STS, D, P, B, L, A, T, X.

If only one reegister block should be printed, xx must be equal to yy.

Note the case: <RD↵ dump register block on level 0.

4.2.3.2.4 USER REGISTER — U

U/

The last value written by TRR LMP, is selected as display source.

4.2.3.2.5 OPERATOR PANEL SWITCH REGISTER — OPR

OPR/

This selects a scratch register where a code to be read by TRA OPR can be deposited. Content of OPR can be read and changed from the console.

4.2.3.3 Internal Register Functions

4.2.3.3.1 INTERNAL REGISTER EXAMINE

Format:

I xx /

The octal number (xx) specifies which internal register is examined. The following codes apply:

0	PANS	Operator's Panel Status, used by operator's panel micro-program.
1	STS	Status register.
2	OPR	Operator's panel switch register, simulated by a scratch register.
3	PSR	Paging status register
4	PVL	Previous program level
5	IIC	Internal interrupt code
6	PID	Priority interrupt detect
7	PIE	Priority interrupt enable
10	CSR	Cache status register, for maintenance only.
11	ACTL	Current level, decoded.
12	ALD	Automatic load descriptor
13	PES	Memory error status
14	PCR	Paging control register. The examined register belongs to the program level controlled by bits 3-6 of the A register.
15	PEA	Memory error address
16	Spare	Do not use.
17	Spare	Do not use.

Example:

```
I7/ 030013          % Present content of PIE is 030013
```

4.2.3.3.2 INTERNAL REGISTER DEPOSIT

Format:

xxxxx ↵

After an internal register examine the contents of the internal register with the same internal register code may be changed by typing an octal number terminated by CR. If the CPU is running, "DEP" must be written between the number and CR. For deposit, the following internal register codes apply:

0	PANC	Operator's panel control, used by operator's panel microprogram.
1	STS	Status register. Only bits 0-7 will be changed.
2	LMP	Writes into a scratch register that may be displayed by writing U/ to MOPC.
3	PCR	Paging control register.
4	Spare	Do not use.
5	IIE	Internal interrupt enable.
6	PID	Priority interrupt detect.
7	PIE	Priority interrupt enable.
10	CCL	Cache Clear.
11	LCIL	Lower cache inhibit limit register.
12	UCILR	Upper cache inhibit limit register.
13	Spare	Do not use.
14	Spare	Do not use.
15	ECCR	Error correction control register.
16	Spare	Do not use.
17	Spare	Do not use.

Examples:

```
I7/ 030013   0↵                  % Examine PIE and change to 000000

I12/ 021540   20044↵             % Examine ALD and change UCILR
                                 % to 020044
```

4.2.3.3.3 INTERNAL REGISTER DUMP — IRD

IRD↵

The 16 internal registers are printed out. This function is only allowed when the CPU is in STOP mode. This restriction avoids the unintentional unlocking of PEA, PES and IIC when the CPU is running.

4.2.3.3.4 SCRATCH REGISTER DUMP — RDE

xx < yy RDE↵

The contents of the 8 scratch registers (only microprogram accessible) in the register blocks xx to yy are printed out, with one register block per line. This function is useful for microprogram debugging only.

4.2.4 *Display Functions (Affects only display)*

4.2.4.1 Displayed Format

uuzzyx F↙

This command will define the display format when the optional display unit is included in the system. uuzzyx are octal digits and define the chosen format. F, without argument, (or with argument equal to zero) will set the default display format which is octal format. The parts of the argument have the following effect:

x — Number representation code.

- x = 0 — Displayed data is in octal representation. zz have no effect.
- x = 1 — Displayed data is in unary representation, i.e., 4 of the bits in the displayed data are used to light one out of 16 indicators. zz indicates which 4 bits to decode.
- x = 2 — Displayed data is in binary representation. zz has no effect.

y — Afterglow code.

- y = 0 — No afterglow in display.
- y = 1 — Zeros are stretched.
- y = 2 — Ones are stretched.
- y = 3 — Zeros and ones are stretched.

zz — Lower start bit for binary display.

- zz = 0-24 $_8$ — Position of lowest bit position to be represented in binary representation.

uu — Display processor maintenance codes (4 bits).

- uu = 1 — Display year and month.
- uu = 2 — Inhibit message.
- uu = 4 — Initialize panel processor.
- uu = 10 — Abort message.

Example:

1421F↙
After this format specification, bits 14 $_8$ - 17$_8$ will be shown in unary representation with afterglow on ones.

4.2.4.2 Display Memory Bus

xy BUS/

This command is only useful when the optional display is included in the system. The memory bus is displayed, and depending on the argument xy, various types of bus information can be sampled and displayed. Read from cache is not displayed.

x = 0 = CD	CPU Data is displayed
x = 1 = DD	DMA Data is displayed
x = 2 = CA	CPU Address is displayed
x = 3 = DA	DMA Address is displayed
y = 0	nothing is displayed
y = 1 = R	only read accesses are displayed
y = 2 = W	only write accesses are displayed
y = 3 = WR	both read and write accesses are displayed

Example:

23 BUS/

All addresses sent from the CPU to memory will be displayed in the DATA field and "CAWR" is shown in the FUNCTION field.

4.2.4.3 Display Activity

ACT/

With this display mode active levels (ACT), clock and indicator functions are displayed.

4.2.5 *Bootstrap Loaders*

The ND-100 has bootstrap loaders for both mass storage and character oriented devices. There are two different load formats:

- Binary format load.
- Mass storage load.

Octal load is not implemented in ND-100.

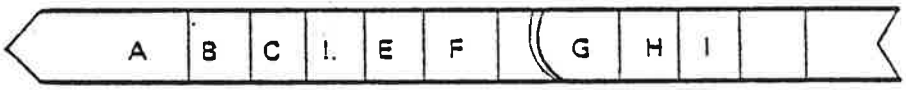
4.2.5.1 Binary Format Load

Binary load is started by typing:

<physical device address> & or <physical device address> $

Loading will take place from the specified device. This device must conform with the programming specifications of either Teletype or tape reader. The device address is the lowest address associated with the device. Binary load will be performed if & or $ is written (or the LOAD button is pressed) and the switch selected ALD has bit 13 equal to "0".

The binary information must obey the following format:



A Any characters not including ! (ASCII 41_8).

B (Optional) octal number (any number of digits) terminated with a CR (line feed is ignored).

C (Optional) octal number terminated with the character ! (see below).

! Indicates 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 are 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 (see above). If I is not a blank, then control is returned to the operator's communication. (The number B will be found in the P register.)

If no device address precedes the & command, then the & is equivalent to pushing the LOAD button on the operator's panel.

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.

4.2.5.2 Mass Storage Load

Mass storage load is started in the same way as binary format load, except that bit 13 in the device address should be a "1".

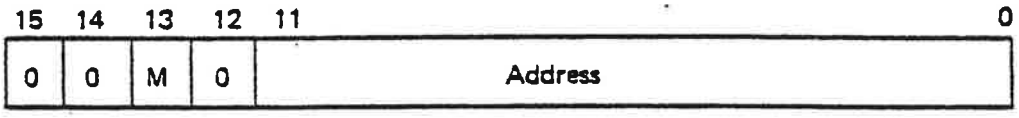
When loading from mass storage, 1K words will be read from mass storage address 0 into main memory starting in address 0. After a successful load, the CPU is started in main memory address 0.

The mass storage device must conform with either drum or disk programming specifications.

4.2.5.3 Automatic Load Descriptor

The ND-100 has a thumbwheel switch called the Automatic Load Descriptor (ALD) (CPU card). This switch selects a 16 bit value to use when the LOAD button is pushed or when a single $ or & is typed.

The 16 bit value has the following meaning:



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 bits 0-10 (unit 0).

If bit 13 is 0, binary load is taken from the device whose (lowest) address is found in bits 0-10.

4.3 *THE DISPLAY*

4.3.1 *General*

The optional display part of the panel is present if the machine has the memory management module installed. This module contains, in addition to the memory management system and cache memory, a display processor. The display processor controls the activity on the display.

There is one button on the display part, the "OPCOM" button. This button allows the operator to use the CPU board terminal for operator communication. This button has the same function as the "OPCOM" button on the operator's panel. The display part of the panel may be placed outside the cabinet (in another room, etc.). Therefore, it is practical to have an "OPCOM" button on this part of the panel.

4.3.2 *The Different Display Functions*

Figure 4.1 shows the normal activity on the display when the machine is running.

The DATA field displays information in binary or octal format (see Section 4.2.4.1). The possible contents are:

- *Active levels (only binary)*

  The active levels in the computer will be shown. There are 16 positions (0-15), one for each level. A one ( ) is set in one of these positions, indicating the active level. The display is provided with afterglow so that it is possible to observe a single instruction on a program level.

- *Register contents.*

  If a register examine is done, the content of the register is shown here.

- *Memory contents.*

  When a memory examine is done, the content of the examined cell will be shown here.

- *Bus information.*

  If the BUS command is given to display memory accesses on the ND-100 bus, the data present on the bus will be shown here and updated continually. When binary format is selected, the address field is used as extension for bit 16-23.

The ADDRESS Field:

— *Calendar clock.*

A clock that tracks the operating system clock is shown here displaying day, hour, minute and second. This clock is adjusted by the "UPDATE" command under SINTRAN III. Under the load procedure this clock will be read by the operating system and taken as system clock. The clock is also connected to the stand-by power and will stay correct even in case of a power failure.

— *Year and month.*

Year and month from the system clock is also shown here by giving the specific F command to MOPC (see Section 4.2.4.1). For example, 1979:10 means October 1979.

— *Current program counter.*

During a register examine, the current program counter is shown here. For example, PC:10153.

— *Memory address.*

If a memory examine is done, the address of the memory location examined is shown here.

The FUNCTION Field:

— *Indicator functions.*

UTIL, utility of the machine, is shown here. That is, how much time the machine spends on level 0 (idle). The more utility, the less the time spent on level 0 and more segments on the display are lit up.

Example:

— *No activity.*

HIT, tells the hits rate in cache memory. The higher the cache hit rate , the more segments are lit up on the display.

Example:

— RING, indicates the user ring taken from the PCR.

Example:

Paging off Ring 1 Ring 2 Ring 3

— MODE, tells if the interrupt system and/or the paging system is turned on.
Example:

Both the interrupt system and the paging system is on.

Only the interrupt system is on.

— *Register name.*

If a register examine is done, the name of the register, eventually also the level for the register, is shown.

Example:

5A, OPR, etc.
5A = A register on level 5
OPR = Operator's Register

— *Memory examine mode.*

When a memory examine is done, the examine mode; virtual or physical, will be shown.

Example:

PEXM — physical examine
2EXM — virtual examine mapped through page table 2.

— *Bus examine type.*

What kind of bus information to be sampled and displayed by the BUS command is displayed here.

Example:

DC R — data under a CPU read from memory operation.

APPENDIX A

ND-100 INSTRUCTIONS

A.1 *ND-100 INSTRUCTION CODES*

Instruction formats and explanations found in the ND-100 Reference Manual.

Δ = displacement
∧ = logical AND
∨ = inclusive OR
∨ = exclusive OR

MEMORY REFERENCE INSTRUCTIONS

Effective Address:

	000000	Address relative to P;	EL-P+ Δ
,X	002000	Address relative to X;	EL=X+ Δ
I	001000	Indirect address;	EL=(P+ Δ)
,B	000400	Address relative to B;	EL=B+ Δ

Store Instructions:

STZ	000000	Store zero;	(EL):=0
STA	004000	Store A;	(EL):=A
STT	010000	Store T;	(EL):=T
STX	014000	Store X;	(EL):=X
MIN	040000	Mem.incr, skip if zero	(EL):=(EL)+1

Load Instructions:

LDA	044000	Load A;	A: =(EL)
LDT	050000	Load T;	T: =(EL)
LDX	054000	Load X;	X: =(EL)

Arithmetical and Logical Instructions:

ADD	060000	Add to A (C, O and Q may also be affected);	A:=A+(EL)
SUB	064000	Subtract from A (C, O and Q may also be affected);	A:=A—(EL)
AND	070000	Logical AND to A;	A:=A ∧(EL)
ORA	074000	Logical inclusive OR to A;	A:=A ∨(EL)
MPY	120000	Multiply integer (O and Q may also be affected);	A:=A*(EL)

Double Word Instructions:

STD	020000	Store double word;	(DW):=AD
LDD	024000	Load double word;	AD:=(DW)

Floating Instructions:

STF	030000	Store floating accum.;	(FW):=TAD
LDF	034000	Load floating accum.;	TAD:=(FW)
FAD	100000	Add to floating accum. (C may also be affected);	TAD:=TAD+(FW)
FSB	104000	Subtract from floating accum. (C may also be affected);	TAD:=TAD—(FW)
FMU	110000	Multiply floating accum. (C may also be affected);	TAD:=TAD*(FW)
FDV	114000	Divide floating accum. (Z and C may also be affected);	TAD:=TAD/(FW)

Byte Instructions:

Addressing: EL=(T)+(X)/2 X=1: Right byte
X=0: Left byte

SBYT	142600	Store byte
LBYT	142200	Load byte
BFILL	140130	Byte fill
MOVBF	140131	Move bytes
MOVB	140132	Move bytes forward

REGISTER OPERATIONS

Arithmetic Operations, RAD=1:

C, O and Q may be affected by the following instructions:

RADD	146000	Add source to destination;	(dr):=(dr)+(sr)
RSUB	146600	Subtract source from destination;	(dr):=(dr)—(sr)
COPY	146100	Register transfer;	(dr):=(sr)
AD1	000400	Also add one to destination;	(dr):=(dr)+1
ADC	001000	Also add old carry to destination;	(dr):=(dr)+C

Logical Operations, RAD = 0:

SWAP	144000	Register exchange;	(sr): = (dr); (dr): = (sr)
RAND	144400	Logical AND to destination;	(dr): = (dr) ∧(sr)
REXO	145000	Logical exclusive OR;	(dr): = (dr) ∨(sr)
RORA	145400	Logical inclusive OR;	(dr): = (dr)V(sr)
CLD	000100	Clear destination before op.;	(dr) = 0
CM1	000200	Use one's complement of source;	(sr) = (sr)^0

Combined Instructions:

EXIT	146142	COPY SL DP,	Return from sub-routine
RCLR	146100	COPY,	Register clear
RINC	146400	RADD AD1,	Register increment
RDCR	146200	RADD CM1,	Register decrement

Extended Arithmetic Operations:

RMPY	141200	Multiply source with destination. Result in double accumulator	AD: = (sr)*(dr)
RDIV	141600	Divide double accumulator with source register. Quotient in A, remainder in D (AD = A*(sr) + D)	A: = AD/(sr)

EXECUTE INSTRUCTION

EXR	140600	Execute instruction found in specified register.

BIT INSTRUCTIONS

BSKP	175000	Skip next location if specified condition is true;	P:=P+1
BSET	174000	Set specified bit equal to specified condition;	
BSTA	176200	Store and clear K;	(B):=K; K:=0
BSTC	176000	Store complement and set K;	(B):=K_0; K:=1
BLDA	176600	Load K;	K:=(B)
BLDC	176400	Load bit complement to K;	K:=$(B)_0$
BANC	177000	Logical AND with bit compl;	K:=K∧$(B)_0$
BORC	177400	Logical OR with bit compl.;	K:=K∨$(B)_0$
BAND	177200	Logical AND to K;	K:=K∧(B)
BORA	177600	Logical OR to K;	K:=K∨(B)

SHIFT INSTRUCTIONS

SHT	154000	Shift T register
SHD	154200	Shift D register
SHA	154400	Shift A register
SAD	154600	Shift A and D registers connected Arithmetic shift. During right shift, bit 15 is extended. During left shift, zeros are shifted in from right.
ROT	001000	Rotational shift. Most and least significant bits are connected.
ZIN	002000	Zero end input
LIN	003000	Link end input. The last vacated bit is fed to M after every shift instruction.
SHR	000200	Shift right; gives negative shift counter.

FLOATING CONVERSION

NLZ	151400	Convert the number in A to a floating number in FA.
DNZ	152000	Convert the floating number in FA to a fixed point number in A.
NLZ+20	151420	Integer to floating conversion.
DNZ-20	152360	Floating to integer conversion.

SEQUENCING INSTRUCTIONS

Unconditional Jump:

JMP	124000	Jump;	P=EL
JPL	134000	Jump to subroutine;	L=P; P=EL

Conditional Jump:

JAP	130000	Jump if A is positive; P= + Δ if:	A≥0
JAN	130400	Jump if A is negative;	A<0
JAZ	131000	Jump if A is zero;	A=0
JAF	131400	Jump if A is nonzero;	A0
JXN	133400	Jump if X is negative;	X<0
JPC	132000	Increment X and jump if positive; X=X+1; P=P+ Δ if	X≥0
JNC	132400	Increment X and jump if negative; X=X+1; P=P+ Δ if	X<0
JXZ	133000	Jump if X is zero;	X=0

Skip Instructions:

SKP	140000	Skip next location if specified condition is true;	P=P+1

Specified Condition:

EQL	000000	Equal to
UEQ	002000	Unequal to
GRE	001000	Signed greater or equal to
LST	003000	Signed less than
MLST	003400	Magnitude less than
MGRE	001400	Magnitude greater or equal to
IF	000000	May be used freely to obtain
O	000000	easy readability

TRANSFER INSTRUCTIONS

Load Independent Instructions:

TRA	150000	Transfer specified internal register to A
TRR	150100	Transfer A to specified internal register

Inter-level Instructions:

IRR	153600	Inter-register Read A:= Specified register on specified level
IRW	153400	Inter-register Write Specified register on specified level := A

MEMORY EXAMINE/DEPOSIT INSTRUCTIONS

EXAM	150416	Memory examine T:= memory location pointed to by AD register
DEPO	150417	Memory deposit Move T to memory location pointed to by AD register

SYSTEM CONTROL INSTRUCTIONS

IOF	150401	Turn off interrupt system
ION	150402	Turn on interrupt system
LWCS	143500	Load writeable control store
MON	153000	Monitor call instruction
PIOF	150405	Turn off paging and interrupt
PION	150412	Turn on page and interrupt
POF	150404	Turn off paging system
PON	150410	Turn on paging system
REX	150407	Reset extended address mode
SEX	150406	Set extended address mode
WAIT	151000	Halt the program/ Give up priority
OPCOM	150400	Start MOPC

PRIVILEGED INSTRUCTIONS

The instructions available only to programs running in system mode (ring 2 or 3) are termed privileged instructions, which are:

IOF	150401	Turn off interrupt system
ION	150402	Turn on interrupt system
PIOF	150405	Turn off paging and interrupt
PION	150412	Turn on page and interrupt
POF	150404	Turn off memory management system
PON	150410	Turn on memory management system
LWCS	143500	Load writeable control store
WAIT	151000	Give up priority, reset current PID bit
IDENT	143600	Identify interrupt
IOX	164000	Input/Output
IOXT	150415	Input/Output
TRA	150000	Transfer internal register to A
TRR	150100	Transfer internal register from A
MCL	150200	Masked clear of register
MST	150300	Masked set of register
LRB	152600	Load registerblock
SRB	152402	Store register block
IRW	153400	Inter-register write
IRR	153600	Inter-register read
REX	150407	Reset extended address mode
SEX	150406	Set extended address mode
EXAM	150416	Memory examine T = memory location pointed to by AD register
DEPO	150417	Memory deposit Memory location pointed to by AD register
OPCOM	150400	Set in OPCOM mode

PHYSICAL MEMORY READ/WRITE INSTRUCTIONS

LDATX	143300	A:=(EL)
LDXTX	143301	X:=(EL)
LDDTX	143302	A:=(EL), D:=(EL+1)
LDBTX	143303	B:=177000 V((EL)+(ED))
STATX	143304	(EL):=A
STZTX	143305	(EL):0
STDTX	143306	(EL):=A, (EL+1):=D

INPUT/OUTPUT CONTROL

IOXT	150415	Transfer data to/from specified device
IOX	164000	Transfer data to/from specified device
IDENT	1436PL	Transfer IDENT code of interrupting device with highest priority on the specified level to A register.
PL10	000004	Level 10
PL11	000011	Level 11
PL12	000022	Level 12
PL13	000043	Level 13

ARGUMENT INSTRUCTIONS

SAA	170400	Set argument to A;	A:=ARG
AAA	172400	Add argument to A;	A:=A+ARG
SAX	171400	Set argument to X;	X:=ARG
AAX	173400	Add argument to X;	X:=X+ARG
SAT	171000	Set argument to T;	T:=ARG
AAT	173000	Add argument to T;	T:=T+ARG
SAB	170000	Set argument to B;	B:=ARG
AAB	172000	Add argument to B;	B:=B+ARG

REGISTER BLOCK INSTRUCTIONS

Addressing: (EL) + 1 + 2 + 3 + 4 + 5 + 6 + 7
P X T A D L STS B

LRB	152600	Load register block
SRB	152402	Store register block

INSTRUCTIONS IN THE CE-OPTION

(CE = Commercial Extended)

ADDD	140120	Add decimal
SUBD	140121	Subtract decimal
COMD	140122	Compare decimal
PACK	140124	Convert to packed decimal
UPACK	140125	Convert to unpacked decimal
SHDE	140126	Decimal shift
INIT	140134	Initialize stack
ENTR	140135	Enter Stack
LEAVE	140136	Leave stack
ELEAV	140137	Error leave stack

ND-100 MNEMONICS AND THEIR OCTAL VALUES

AAA	: 172400	FMU	: 110000	MON	: 153000
AAB	: 172000	FSB	: 104000	MPY	: 120000
AAT	: 173000	GEQ	: 000400	MST	: 150300
AAX	:173400	GRE	: 001000	NLZ	: 151400
ADC	: 001000	I	: 001000	ONE	: 000200
ADD	: 060000	IDENT	: 143600	OPCOM	: 150400
ADDD	: 140120	IF	: 000000	OPR	: 000002
AD1	: 000400	IIC	: 000005	ORA	: 074000
ALD	: 000012	IIE	: 000005	PACK	: 140124
AND	: 070000	INIT	: 140134	PCR	: 000003
,B	: 000400	IOF	: 150401	PEA	: 000015
BAC	: 000600	ION	: 150402	PES	: 000013
BANC	: 177000	IOX	: 164000	PGS	: 000003
BAND	: 177200	IOXT	: 150415	PID	: 000006
BCM	: 000400	IRR	: 153600	PIE	: 000007
BLDA	: 176600	IRW	: 153400	PIOF	: 150405
BLDC	: 176400	JAF	: 131400	PION	: 150412
BORA	: 177600	JAN	: 130400	PL10	: 000004
BORC	: 177400	JAP	: 130000	PL11	: 000011
BSET	: 174000	JAZ	: 131000	PL12	: 000022
BSKP	: 175000	JMP	: 124000	PL13	: 000043
BSTA	: 176200	JNC	: 132400	POF	: 150404
BSTC	: 176000	JPC	: 132000	PON	: 150410
CCLR	: 000010	JPL	: 134000	PVL	: 000004
CILR	: 000012	JXN	: 133400	RADD	: 146000
CLD	: 000100	JXZ	: 133000	RAND	: 144400
CM1	: 000200	LBYT	: 142200	RCLR	: 146100
CM2	: 000600	LCIL	: 000011	RDCR	: 146200
COMD	: 140122	LDA	: 044000	RDIV	: 141600
COPY	: 146100	LDATX	: 143300	REX	: 150407
CSR	: 000010	LDBTX	: 143303	REXO	: 145000
DA	: 000005	LDD	: 024000	RINC	: 146400
DB	: 000003	LDDTX	: 143302	RMPY	: 141200
DD	: 000001	LDF	: 034000	RORA	: 145400
DEPO	: 150417	LDT	: 050000	ROT	: 001000
DL	: 000004	LDX	: 054000	RSUB	: 146600
DNZ	: 152000	LDXTX	: 143301	SA	: 000050
DP	: 000002	LEAVE	: 140136	SAA	: 170400
DT	: 000006	LIN	: 003000	SAB	: 170000
DX	: 000007	LMP	: 000002	SAD	: 154600
ECCR	: 000015	LRB	: 152600	SAT	: 171000
ELEAV	: 140137	LSS	: 002400	SAX	: 171400
ENTR	: 140135	LST	: 003000	SB	: 000030
EQL	: 000000	LWCS	: 143500	SBYT	: 142600
EXAM	: 150416	MCL	: 150200	SD	: 000010
EXIT	: 146142	MGRE	: 001400	SEX	: 150406
EXR	: 140600	MIN	: 040000	SHA	: 154400
FAD	: 100000	MIX3	: 143200	SHD	: 154200
FDV	: 114000	MLST	: 003400	SHDE	: 140126

```
SHR     : 000200
SHT     : 154000
SKP     : 140000
SL      : 000040
SP      : 000020
SRB     : 152402
SSC     : 000060
SSK     : 000020
SSM     : 000070
SSO     : 000050
SSQ     : 000040
SSTG    : 000010
SSZ     : 000030
ST      : 000060
STA     : 004000
STATX   : 143304
STD     : 020000
STDTX   : 143306
STF     : 030000
STS     : 000001
STT     : 010000
STX     : 014000
STZ     : 000000
STZTX   : 143305
SUB     : 064000
SUBD    : 140121
SWAP    : 144000
SX      : 000070
TRA     : 150000
TRR     : 150100
UCIL    : 000012
UEQ     : 002000
UPACK   : 140125
WAIT    : 151000
,X      : 002000
ZIN     : 002000
ZRO     : 000000
```

A.2 *ND-100 INSTRUCTION EXECUTION TIMES*

NOTE: The instruction times are measured for a program running on a standard ND-100. That is, all references are in local memory. Two models of the ND-100 are available and the instruction times are given first for the slower model without cache and then for the faster model with cache.

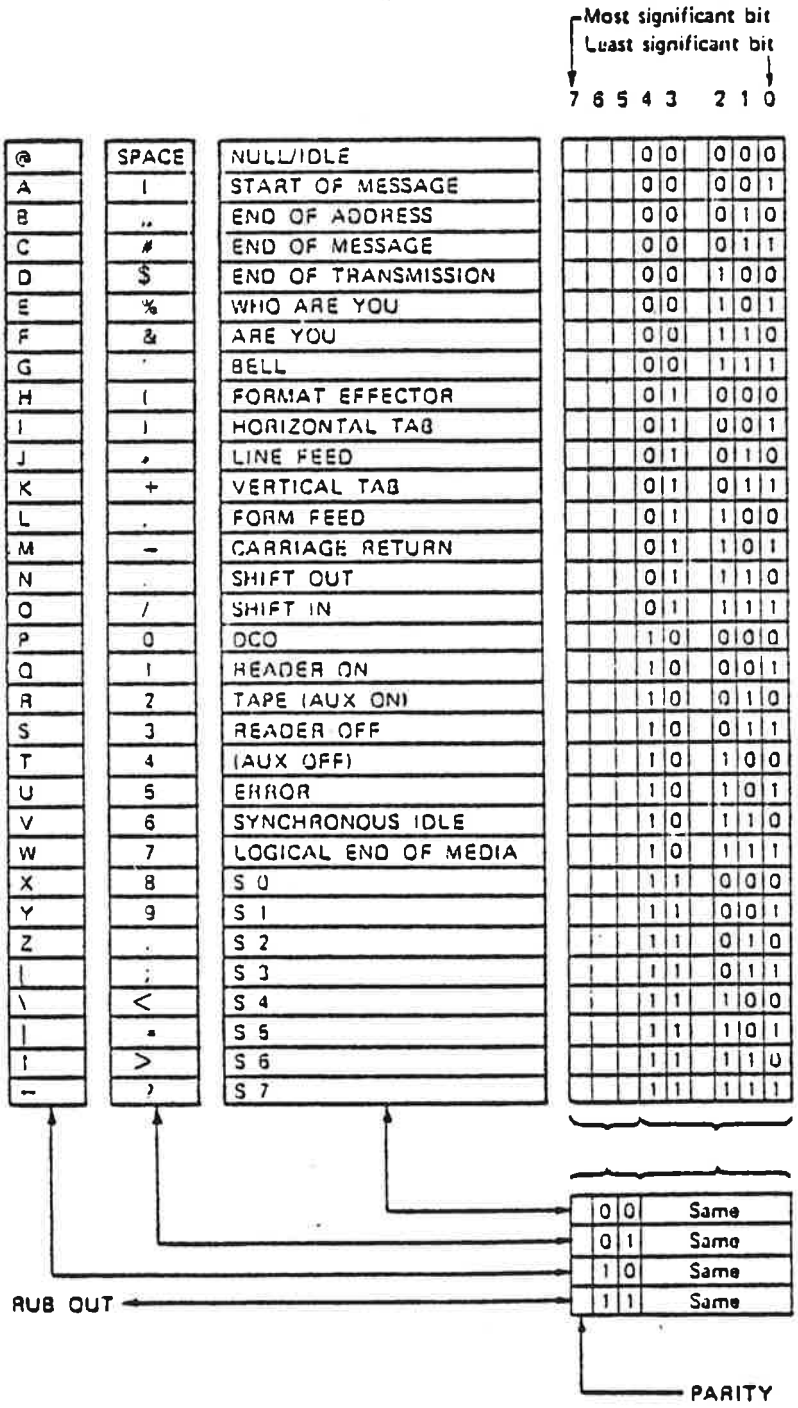
Instruction	*Standard 190ns cycle time (Not Cache)*	*Fast CPU 150ns cycle time (Cache)*
JMP *1	1.84	0.99
SAA 5	0.73	0.46
AAA 2	0.73	0.46
COPY SA DD	0.73	0.46
RADD SB DA	0.73	0.46
RSUB ST DX	0.73	0.46
SWAP SA DB	0.94	0.74
RAND SA DT	0.73	0.46
REXO ST DT	0.73	0.46
RORA SD DA	0.73	0.46
BSET ONE 20 DX	1.14	0.89
BSET BAC 30 DX	1.72	1.32
BSTA 40 DX	2.31	1.76
BLDA 20 DX	1.33	1.03
SHA 1	1.33	1.03
SHA 13	3.29	2.48
SHT 1	1.33	1.03
SAD 1	1.53	1.17
LDA *16	1.65	0.95
STA *16	1.52	1.20
LDD *16	2.39	1.29
STD *16	2.14	1.80
LDF *16	3.12	1.66
STF *16	2.72	2.41
IRW 50 DX	2.14	1.61
IRR 50 DB	2.14	1.61
TRA PIE	0.94	0.84
TRR PCR	3.44	2.83
MCL PID	8.38	8.05
SRB 40	6.71	6.24
LRB 40	7.57	4.11
LBYT % LEFT	2.22	1.38
SBYT % LEFT	2.87	2.22
LBYT % RIGHT	2.04	1.24
SBYT % RIGHT	2.48	1.93
LDA I *16	2.39	1.29
STA I *16	2.26	1.55
IOX 302	5.02	3.64
ION	6.71	6.13
MON 0	0.77	0.59

Instruction	*Standard 190ns cycle time (Not Cache)*	*Fast CPU 150ns cycle time (Cache)*
Conditional JUMP-SKIP Instructions Condition TRUE		
JAN *2	1.49	0.94
JPC *2	1.49	0.94
SKP DX EQL SA	1.49	0.94
BSKP ONE 10 DA	1.95	1.30
BSKP BCM 10 DA	2.60	1.65
Condition FALSE		
JAN *2	0.87	0.71
JPC *2	0.87	0.71
SKP DX EQL SA	0.87	0.71
BSKP ONE 10 DA	1.95	1.30
BSKP BCM 10 DA	1.95	1.42
Instructions with Data Dependent Execution Times		
MPY *5	7.49	5.57
MIX3	0.78	0.60
RMPY SX DT	4.86	3.62
RDIV ST % POS. NO.	8.40	6.22
RDIV SB % NEG. NO.	7.75	5.82
RDIV SX % OVERFLOW	2.28	1.74
FAD *7	4.55	3.13
FAD *12	8.17	5.57
FAD *15	13.46	9.55
FSB *7	4.74	3.24
FSB *12	11.00	7.70
FSB *15	16.22	11.62
FMU *7	18.82	13.89
FMU *12	18.84	14.13
FMU *20	18.82	13.90
FDV *7	19.98	14.62
FDV *12	20.15	14.78
FDV *20	4.34	3.24
NLZ 20 % 0	0.94	0.73
NLZ 20 % 1	5.77	4.37
NLZ 20 % 40000 (8)	3.08	2.31
DNZ —20 % 0	1.88	1.45
DNZ —20 % 1	5.79	3.27
DNZ —20 % 40000 (8)	2.07	1.59
MIN *3 % SKP FALSE	2.25	1.56
MIN *3 % SKP TRUE	2.82	1.79
EXR SA	0.91	0.69
WAIT	6.99	6.69

APPENDIX B

MODEL 33 ASR/KSR TELETYPE CODE (ASCII) IN BINARY FORM

HOLE PUNCHED = MARK = 1
NO HOLE PUNCHED = SPACE = 0



APPENDIX C

STANDARD ND-100 DEVICE REGISTER ADDRESSES AND IDENT CODES

In the following only the most frequently used Device Names are listed.

Two Device Names may use the same Device Register Address range. In these cases only the most common Device Name is listed.

Definitions:

Device Register Address = Device Number + Register Number.

Device Number = The lowest Device Register Address for each Device Name.

Register Number = Register Number within the Device (see the manual ND-100 Input/Output System, Appendix B).

Interrupt Level 10 = Output Devices (PIO).
Interrupt Level 11 = Mass Storage Devices (DMA).
Interrupt Level 12 = Input Devices (PIO).
Interrupt Level 13 = Real Time Clock.

SINTRAN III Logical Device Number is a unique number for the Device Name.

Ident Code is a code sent from the device interface. The Ident Code tells the CPU which device asked for an interrupt. The Ident Code is unique for the Device Number on a specified Interrupt Level.

STANDARD ND-100 DEVICE NUMBERS AND IDENT CODES*

Device Reg. Address Range (octal)	*Interrupt Level*	*SINTRAN III Logical Device Numbers* (octal)*	*Ident Code (octal)*	*Device Name*
4- 7			4	Memory Parity N-12/N-42
10- 13	13		1	Real Time Clock 1
14- 17	13		2	Real Time Clock 2
20- 23	13		6	Real Time Clock 3
24- 27	13		7	External Interrupt
30- 33	12		16	NORD-50/1
34- 37	10		16	ACM 5
40- 43	10		15	ACM 1
44- 47	10		25	ACM 2
50- 53	10		40	ACM 3
54- 57	10		41	ACM 4
60- 77				NORD-50/1 Regs.
100-107	10/12	6	4	Sync. Modem 1
110-117	10/12	16	14	Sync. Modem 2
120-127	10/12	30	20	Sync. Modem 3
130-137	10/12	31	24	Sync. Modem 4
140-147	10/12	26	30	Sync. Modem 5
150-157	10/12	27	34	Sync. Modem 6
160-167	10/12		40	Sync. Modem 7
170-177	10/12		10	Sync. Modem 8
200-207	10/12	7	60	Terminal 17
210-217	10/12	17	61	Terminal 18
220-227	10/12	52	62	Terminal 19
230-237	10/12	53	63	Terminal 20
240-247	10/12	54	64	Terminal 21
250-257	10/12	55	65	Terminal 22
260-267	10/12	56	66	Terminal 23
270-277	10/12	57	67	Terminal 24
300-307**	10/12	1	1(120)***	Terminal 1
310-317**	10/12	11	5(121)***	Terminal 2/TET 15
320-327**	10/12	42	6(122)***	Terminal 3/TET 14
330-337**	10/12	43	7(123)***	Terminal 4/TET 13
340-347	10/12	44	44	Terminal 5/TET 12
350-357	10/12	45	45	Terminal 6/TET 11
360-367	10/12	46	46	Terminal 7/TET 10
370-377	10/12	47	47	Terminal 8/TET 9

* A complete list of SINTRAN III Logical Device Numbers is found in SINTRAN III Reference Manual (ND-60.125).

** Terminal no. 1 is implemented on the CPU module. Terminals with device numbers 310-317, 320-327 and 330-337 are normally not used.

*** Number in parenthesis is valid for 4 current loop modules.

Device Reg. Address Range (octal)	*Interrupt Level*	*SINTRAN III Logical Device Numbers* (octal)*	*Ident Code (octal)*	*Device Name*
400- 403	12	2	2	Paper Tape Reader 1
404- 407	12	12	22	Paper Tape Reader 2
410- 413	10	3	2	Paper Tape Punch 1
414- 417	10	13	22	Paper Tape Punch 2
420- 423	12	4	3	Card Reader 1
424- 427	12	14	23	Card Reader 2
430-433	10	5	3	Line Printer 1
434- 437	10	15	23	Line Printer 2
440- 443	10	10	11	Calcomp Plotter 1
444- 447	10	50	12	Card Punch 1
450- 453	10	35	21	Card Punch 3/Calc. 2
454- 457	10	51	13	Card Punch 2
460- 467	10/12		31	E & Pict. Syst. I/O
470- 477	12			Graphical Pen
500- 507	11		1	Disk System 1
510- 517	11		5	Disk System 2
520- 527	11		3	Mag.Tape 1
530- 537	11		7	Mag. Tape 2
540- 547	11		2	Drum 1
550- 557	11		6	Drum 2
560- 577	12/13	1006	156	HDLC HASP 1
600- 607	11	22	4	Versatec 1
610- 617	11		11	Core -to-Core 1
620- 637	11	36	10	CDC I/O Link
640- 647	10/12	1040	124	Terminal 33
650- 657	10/12	1041	125	Terminal 34
660- 667	10/12	1042	126	Terminal 35
670- 677	10/12	1043	127	Terminal 36
700- 707	12	20	11	CATSY 1
710- 717	12	21	21	CATSY 2
720- 727	11		23	E & S Pict. Syst. DMA
730- 737	10		10	D/A- Converter
750- 753	13		5	BIG MPM LOG Module
754- 757	12		13	Process Input 5
760- 767	10-11-12-13		100	Test Card
770- 773	12		17	Dig. Reg. 1 Input
774- 777	10		17	Dig. Reg. 1 Output
1000-1003	12		26	Dig. Reg. 2 Input
1004-1007	10		26	Dig. Reg. 2 Output
1010-1013	12		27	Dig. Reg. 3 Input
1014-1017	10		27	Dig. Reg. 3 Output
1020-1023	12		43	Dig. Reg. 4 Input
1024-1027	10		43	Dig. Reg. 4 Output
1030-1033	12		116	NORD 50/2
1034				Watch Dog
1035				Process Output 1
1036				Process Output 2

Device Reg. Address Range (octal)	*Interrupt Level*	*SINTRAN III Logical Device Numbers (octal)*	*Ident Code (octal)*	*Device Name*
1037				Process Output 3
1040-1043	12		15	Process Input 1
1044-1047	12		25	Process Input 2
1050-1053	12		40	Process Input 3
1054-1057	12		12	Process Input 4
1060-1077				NORD-50/2 Reg.
1100-1107	10/12	1044	130	Terminal 37
1110-1117	10/12	1045	131	Terminal 38
1120-1127	10/12	1046	132	Terminal 39
1130-1137	10/12	1047	133	Terminal 40
1140-1147	10/12	1050	134	Terminal 41
1150-1157	10/12	1051	135	Terminal 42
1160-1167	10/12	1052	136	Terminal 43
1170-1177	10/12	1053	137	Terminal 44
1200-1207	10/12	70	70	Terminal 25
1210-1217	10/12	71	71	Terminal 26
1220-1227	10/12	72	72	Terminal 27
1230-1237	10/12	73	73	Terminal 28
1240-1247	10/12	74	74	Terminal 29/PHOTOS. 1
1250-1257	10/12	75	75	Terminal 30/PHOTOS.2
1260-1267	10/12	76	76	Terminal 31/PHOTOS.3
1270-1277	10/12	77	77	Terminal 32/PHOTOS. 4
1300-1307	10/12	60	50	Terminal 9
1310-1317	10/12	61	51	Terminal 10
1320-1327	10/12	62	52	Terminal 11
1330-1337	10/12	63	53	Terminal 12
1340-1347	10/12	64	54	Terminal 13
1350-1357	10/12	65	55	Terminal 14
1360-1367	10/12	66	56	Terminal 15
1370-1377	10/12	67	57	Terminal 16
1400-1407	10/12	1054	140	Terminal 45
1410-1417	10/12	1055	141	Terminal 46
1420-1427	10/12	1056	142	Terminal 47
1430-1437	10/12	1057	143	Terminal 48
1440-1443	12		101	A/D Converter 1
1444-1447	12		102	A/D Converter 2
1450-1453	12		103	A/D Converter 3
1454-1457	12		104	A/D Converter 4
1460-1463	12		105	A/D Converter 5
1464-1467	12		106	A/D Converter 6
1470-1473	12		107	A/D Converter 7
1474-1477	12		110	A/D Converter 8
1500-1507	10/12	1060	144	Terminal 49

Device Reg. Address range (octal)	*Interrupt Level*	*SINTRAN III Logical Device Numbers (octal)*	*Ident Code (octal)*	*Device Name*
1510-1517	10/12	1061	145	Terminal 50
1520-1527	10/12	1062	146	Terminal 51
1530-1537	10/12	1063	147	Terminal 52
1540-1547	11		17	Big Disk System 1
1550-1557	11		20	Big Disk System 2
1560-1567	11	1000- 1002	21	Floppy Disk 1 (Unit 0, 1, 2)
1570-1577	11	1003- 1005	22	Floppy Disk 2 (Unit 0, 1, 2)
1600-1603	11		14	Versatec 2
1604-1607				HDLC Remote Load 1
1610-1613				HDLC Remote Load 2
1614-1617				HDLC Remote Load 3
1620-1623				HDLC Remote Load 4
1624-1627				HDLC Remote Load 5
1630-1633				HDLC Remote Load 6
1634-1637				HDLC Remote Load 7
1640-1657	12/13		150	HDLC NORD-NET 1
1660-1677	12/13		151	HDLC NORD-NET 2
1700-1717	12/13		152	HDLC NORD-NET 3
1720-1737	12/13		153	HDLC NORD-NET 4
1740-1757	12/13		154	HDLC NORD-NET 5
1760-1777	12/13		155	HDLC NORD-NET 6
100000-100003				Bus Expander 0
100004-100007				Bus Expander 1
100010-100013				Bus Expander 2
100014-100017				Bus Expander 3
100020-100023				Bus Expander 4
100024-100027				Bus Expander 5
100030-100033				Bus Expander 6
100034-100037				Bus Expander 7
100115				ECCR
100200-100203	13/13		20	Bus Controller 1
100204-100207	13/13		21	Bus Controller 2
100210-100213	13/13		22	Bus Controller 3
100214-100217	13/13		23	Bus Controller 4
100220-100223	13/13		24	Bus Controller 5
100224-100227	13/13		25	Bus Controller 6
100230-100233	13/13		26	Bus Controller 7
100234-100237	13/13		27	Bus Controller 8
100240-100243	13/13		30	Bus Controller 9
100244-100247	13/13		31	Bus Controller 10
100250-100253	13/13		32	Bus Controller 11
100254-100257	13/13		33	Bus Controller 12
100260-100263	13/13		34	Bus Controller 13

Device Reg. Address range (octal)	*Interrupt Level*	*SINTRAN III Logical Device Numbers (octal)*	*Ident Code (octal)*	*Device Name*
100264-100267	13/13		35	Bus Controller 14
100270-100273	13/13		36	Bus Controller 15
100274-100277	13/13		37	Bus Controller 16
100300-100303	13/13		40	Bus Controller 17
100304-100307	13/13		41	Bus Controller 18
100310-100313	13/13		42	Bus Controller 19
100314-100317	13/13		43	Bus Controller 20
100320-100323	13/13		44	Bus Controller 21
100324-100327	13/13		45	Bus Controller 22
100330-100333	13/13		46	Bus Controller 23
100334-100337	13/13		47	Bus Controller 24
100340-100343	13/13		50	Bus Controller 25
100344-100347	13/13		51	Bus Controller 26
100350-100353	13/13		52	Bus Controller 27
100354-100357	13/13		53	Bus Controller 28
100360-100363	13/13		54	Bus Controller 29
100364-100367	13/13		55	Bus Controller 30
100370-100373	13/13		56	Bus Controller 31
100374-100377	13/13		57	Bus Controller 32

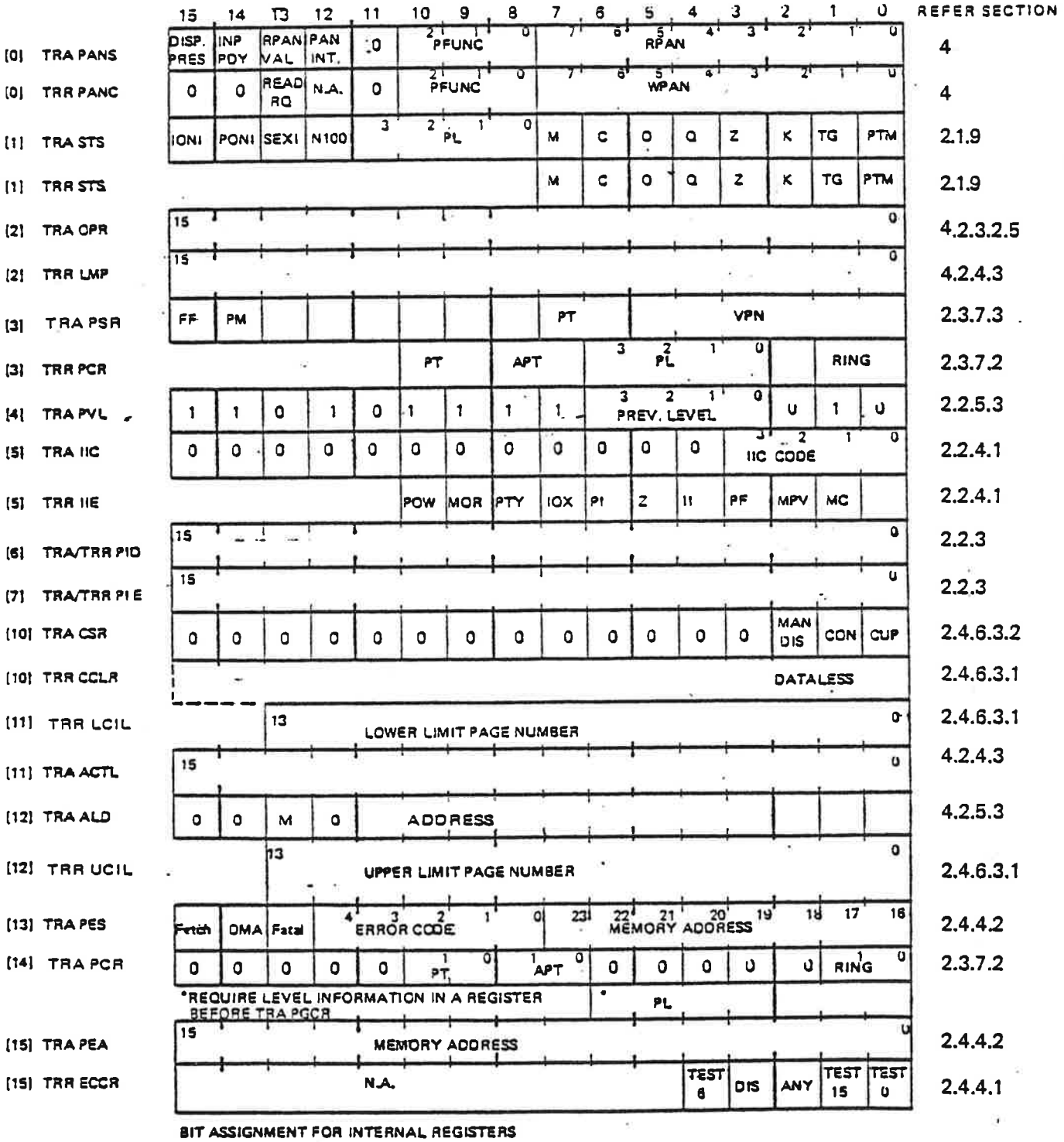
APPENDIX D

INTERNAL REGISTERS

The following internal registers are implemented for internal control and status of the CPU. Format is given in the following table. Detailed descriptions are found in the sections specified.

Register Name:	*No.:*	*Description:*
PANS	0	Panel status register. Gives information to the microprogram about the display status. Also used by microprogram.
PANC	0	Panel Control. Controls the state of the display from the microprogram. Also used by microprogram.
STS	1	Status Register. Bits 0-7 are level dependent and accessible from user programs while bits 8-15 are system dependent and only accessible by system (TRA/TRR).
OPR	2	Operator's register. Implemented in firmware.
LMP	2	Display register. Implemented in firmware.
PSR		Paging status register.
PCR	3	Paging control register.
PVL	4	Previous level. The content of the register is: IRR <previous level * 10^8> DP.
IIC	5	Internal interrupt code.
IIE	5	Internal interrupt enable.
PID	6	Priority interrupt detect.
PIE	7	Priority interrupt enable.
CSR	10	Cache status.
CCLR	10	Clear cache
LCILR	11	Lower cache inhibit limit register
ACTL	11	Active level

ALD	12	Automatic load descriptor
UCILR	12	Upper cache inhibit limit register
PES	13	Parity error status
PCR	14	Paging control register read on specified level
PEA	15	Parity error address



BIT ASSIGNMENT FOR INTERNAL REGISTERS

APPENDIX E

SWITCH SETTINGS FOR THE DIFFERENT ND-100 MODULES

E.1 *SWITCHES ON THE CPU MODULE (3002)*

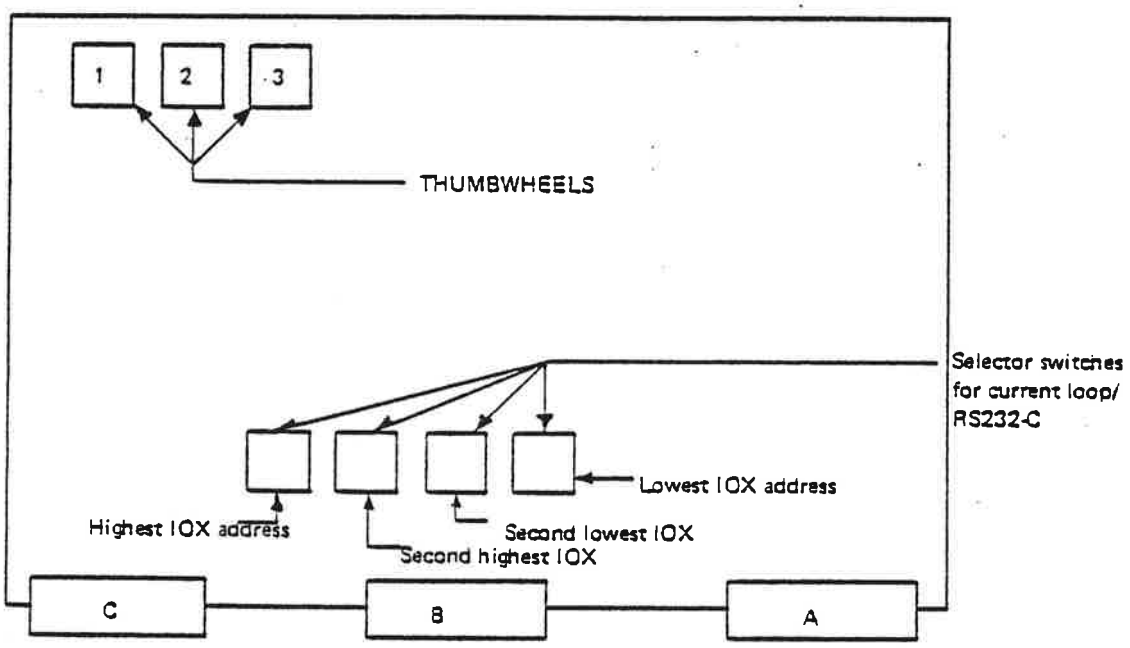
E.1.1 *ALD — Automatic Load Descriptor*

ALD	I12	LOCK and Standby Power OK	LOCK and Standby Power not OK	UNLOCK and Load
15	0	Start in address 20	Stop	Nothing
14	1560	Start in address 20	Binary load from 1560	Binary load from 1560
13	20500	Start in address 20	Mass storage load from 500	Mass storage load from 500
12	21540	Start in address 20	Mass storage load from 1540	Mass storage load from 1540
11	400	Start in address 20	Binary load from 400	Binary load from 400
10	1600	Start in address 20	Binary load from 1600	Binary load from 1600
9		Start in address 20		
8		Start in address 20		
7	100000	Stop	Stop	Nothing
6	101560	Binary load from 1560	Binary load from 1560	Binary load from 1560
5	120500	Mass load from 500	Mass storage load from 500	Mass storage load from 500
4	121540	Mass load from 1540	Mass storage load from 1540	Mass storage load from 1540
3	100400	Binary load from 400	Binary load from 400	Binary load from 400
2	101600	Binary load from 1600	Binary load from 1600	Binary load from 1600

E.1.2 *Console: Speed Setting for Console Terminal.*

0	110 baud
1	150 baud
2	300 baud
3	2400 baud
4	1200 baud
5	1800 baud
6	4800 baud
7	9600 baud
8	2400 baud
9	600 baud
10	200 baud
11	134.5 baud
12	75 baud
13	50 baud
14	—
15	—

E.2 *SWITCHES ON FLOPPY AND 4 TERMINALS MODULE (3010)*



1 = Floppy disk system
2 = terminal group
3 = initial band rate for terminals

E.2.1 *1 Floppy Disk System*

0 = floppy system no. 1 (IOX 1560 - 1567, IDENT = 21)
1 = floppy system no. 2 (IOX 1570 - 1577, IDENT = 22)

2-15 are unused, will answer on IOX 0-7.

E.2.2 *2 Terminal Group:*

Each group consists of 4 terminals with consecutive IOX addresses and ident codes.

0 = terminals 1-4	(IOX 300 - 337,	IDENT 120-123)
1 = terminals 5-8	(IOX 340 - 377,	IDENT 44-47)
2 = terminals 9-12	(IOX 1300 - 1337,	IDENT 50-53)
3 = terminals 13-16	(IOX 1340 - 1377,	IDENT 54-57)
4 = terminals 33-36	(IOX 640 - 677,	IDENT 124-127)
5 = terminals 37-40	(IOX 1100 - 1137,	IDENT 130-133)
6 = terminals 41-44	(IOX 1140 - 1177,	IDENT 134-137)
7 = terminals 45-48	(IOX 1400 - 1437,	IDENT 140-143)
8 = terminals 49-52	(IOX 1500 - 1537,	IDENT 144-147)
9 = terminals 53-46	(IOX 1640 - 1677,	IDENT 150-153)
10 = terminals 57-60	(IOX 1700 - 1737,	IDENT 154-157)
11 = terminals 61-64	(IOX 1704 - 1777,	IDENT 160-163)
12 = terminals 17-20	(IOX 200 - 230,	IDENT 60-63)
13 = terminals 21-24	(IOX 240 - 270,	IDENT 64-67)
14 = terminals 25-28	(IOX 1200 - 1237,	IDENT 70-73)
15 = terminals 29-32	(IOX 1240 - 1277,	IDENT 74-77)

E.2.3 *3 Initial Baud Rate for Terminals*

0 = 110 baud
1 = 150 baud
2 = 300 baud
3 = 2400 baud
4 = 1200 baud
5 = 1800 baud
6 = 4800 baud
7 = 9600 baud
8 = 2400 baud
9 = 600 baud
10 = 200 baud
11 = 134.5 baud
12 = 75 baud
13 = 50 baud
14 = unused
15 = unused

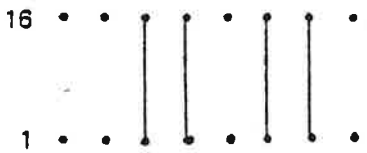
Selector switches for current loop/RS232-C.
Switch set to 0 selects current loop.
Switch set to 1 selects RS232-C.
Switch settings under Terminal Group and Initial baud rate for terminals are also valid for the 8 terminal modules (3013).

This description is correct for the 2 position switch.

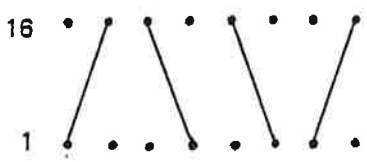
If a hexadecimal switch is used:

0 = current loop
F = RS232-C

If component houses are used:

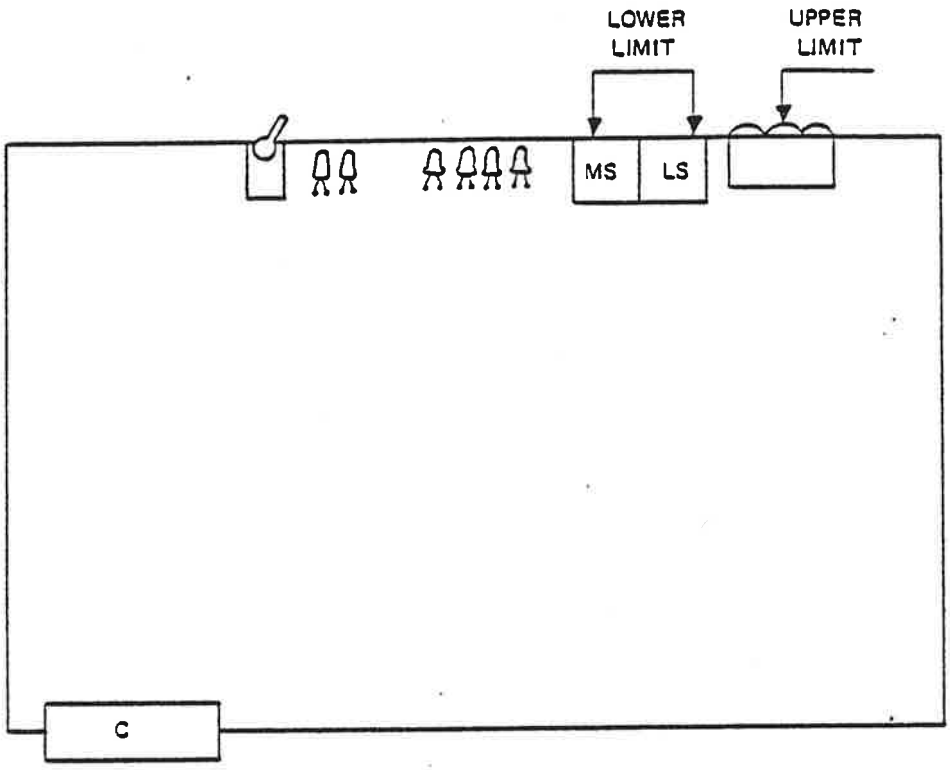


Selects current loop



Selects RS-232-C

E.3 *SWITCHES ON MEMORY MODULES (3005)*



Lower limit is a 2 digit octal number, determining lower memory address.

Upper limit is automatically displayed according to actual memory size.

The limit address increments are 16K units, such that:

Octal Address = 40,000 x limit.

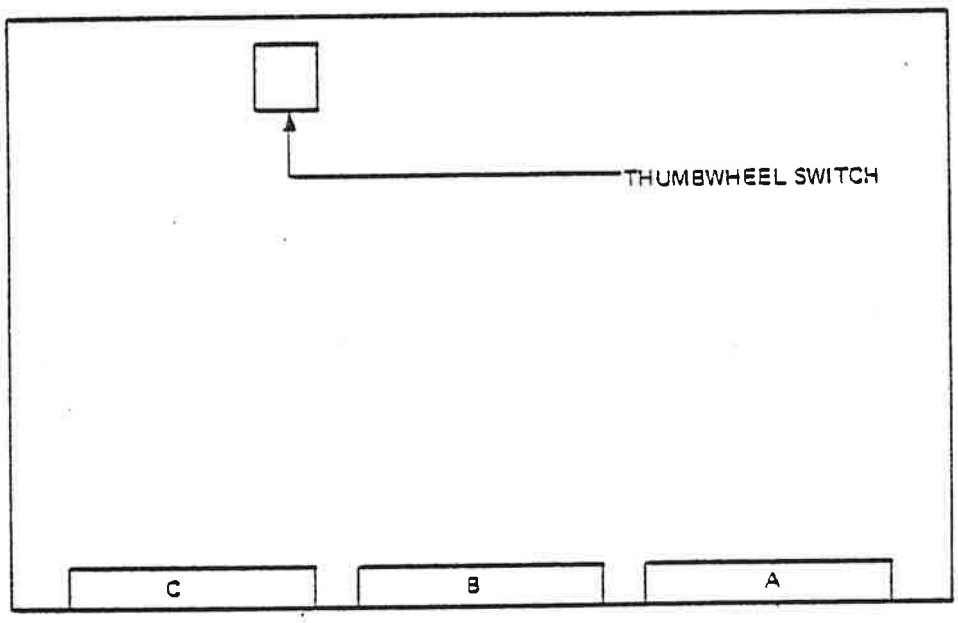
Lower limit	Size	Upper limit	Address range
0 0	16K	0 1	0 - 16K
0 0	32K	0 2	0 - 32K
0 0	64K	0 4	0 - 64K
0 3	32K	0 5	48 - 80K
0 3	64K	0 7	48 - 112K
3 4	64K	4 0	448 - 512K

The switches can be set to numbers 8 - 15, but only 0-7 normally have a meaning. It is, however, possible to allow the slot position code to determine the address range by putting lower limit to 88. (This is only meaningful for 64K). Then the address ranges will be:

Slot	Lower limit	Size	Upper limit	Address Range
12	88	64K	04	0-64K
11	88	64K	10	64-128K
10	88	64K	14	. .
9	88	64K	20	. .
8	88	64K	24	. .
7	88	64K	30	. .
6	88	64K	34	384-448K
5	88	64K	40	448-512K

etc.

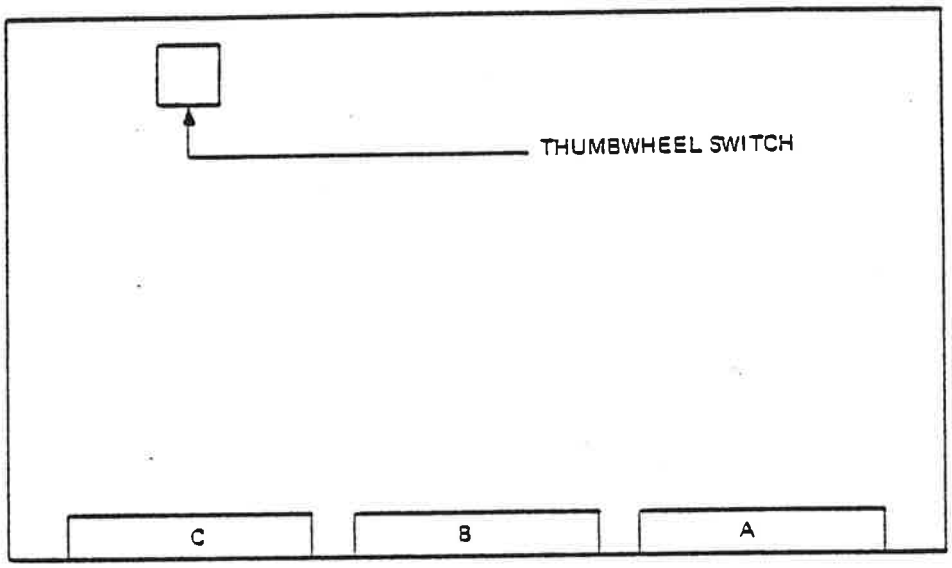
E.4 *SWITCHES ON THE 10MB DISK MODULE (3004)*



Switch pos 0: device no. 500-507, ident no. 1 (disk system 1)
Switch pos 1: device no. 510-517, ident no. 5 (disk system 2)
Switch pos 2-15: not used

IOX address bit 15 active will inhibit this card.

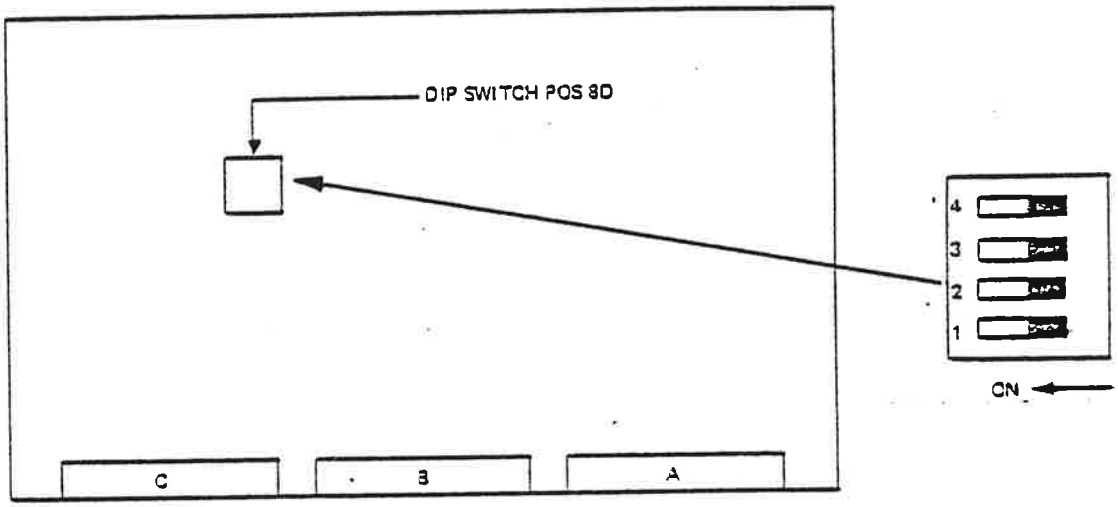
E.5 *SWITCH SETTING ON THE PERTEC MAGNETIC TAPE MODULE(3006)*



Switch pos 2: device no. 520-527, ident no. 3 (mag. tape 1)
Switch pos 3: device no. 530-537, ident no. 7 (mag. tape 2)
Switch pos 2-15: not used.

IOX address bit 15 active will inhibit this card.

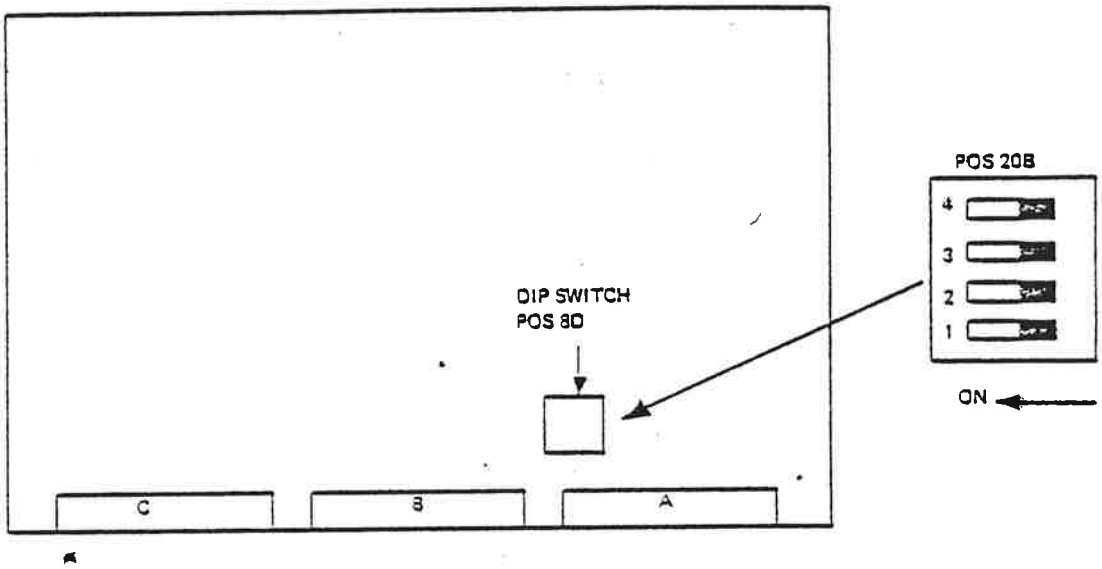
E.6 *SWITCH SETTING ON ND-100 BUS ADAPTER (3008)*



Switch 1 = off: 0 ⩾ device no. ⩾ 3777, 0 ⩾ident no. ⩾ 377
Switch 1 = on: 2000 ⩾ device no. ⩾ 3777, 400 ⩾ ident no. ⩾ 777
Switch 2 = off: normal
Switch 2 = on: block all interfaces on this bus
Switches 3 and 4: not used.

IOX address bit 15 active will inhibit this card.

E.7 *SWITCH SETTING ON LOCAL I/O Bus (3009)*



Switch 1 = off: 0 ⩾ device no. ⩾ 1777, 0 ⩾ ident no. ⩾ 377
Switch 1 = on: 2000 ⩾ 3777, 400 ⩾ ident no. ⩾ 777
Switch 2 = off: normal
Switch 2 = on: block all interfaces on this bus
Switches 3 and 4: not used

IOX address bit 15 active will block this card.

APPENDIX F

OPERATOR'S COMMUNICATION INSTRUCTION SURVEY

F.1 *CONTROL FUNCTIONS (DOES NOT AFFECT DISPLAY)*

System Control

OPCOM ☐	Enter Operator's Communication mode
ESC key	Leave Operator's Communication mode
MCL ☐ MACL ⌟	Generate Master Clear
STOP ☐ STOP ⌟	Stop Program and enter OPCOM Mode
LOAD ☐ & or $	Load according to ALD code (read by I12/)
xxxxxx& or xxxxxx$	Load from device x

Program Control

!	Continue Program from address of program counter
xxxxxx!	Start Program from address x
Z	Execute a Single Instruction according to program counter
xxxxxxZ	Execute x Instructions from address of program counter
xxxxxx▪	Execute Program until program counter = x and stop
xxxxxx''	Execute Instruction Code x repeatedly
xxxxxxIO/nnnnnn	Execute IOX instruction with device number x OPR = Output Data; n = Returned Data

Miscellaneous Functions

xxx#	Do Memory Test in segment x from address of B register to address of X register. P = Fail Address, T = Fail Bits, D = Fail Pattern, L = Test Pattern.
space or @	Delete entry
*nnnnnn	Current Location of memory examine is n (16 least sign. bits)
OPR/nnnnnn zzzzzz ⌟	Change Operators Panel "Switches" from n to z

F.2 *DISPLAY FUNCTIONS (AFFECTS ONLY DISPLAY)*

uuzzyxF ⌟ Define Format of Displayed Information (F ⌟ is default)

x (3 bits): 0 = Octal; 1 = Decoded according to z; 2 = Binary

y (3 bits): 0 = Normal; 1 = Stretch Zeros; 2 = Stretch Ones; 3 = Stretch Zeros and Ones

z (6 bits): Decode the 4 bits z to z+3 to a ONE among ZEROs.

u (4 bits): for Display Processor Maintenance

1 = Display Year and Month
2 = Inhibit message
4 = Initialize panel processor
10 = Abort message

yxBUS/ Display Memory Accesses on NORD-100 Bus

x (3 bits): 0 = Undefined; 1 = Read Access; 2 = Write Access; 3 = Write or Read Access

y (3 bits): 0 = CPU Data; 1 = DMA Data; 2 = CPU Address; 3 = DMA Address

ACT/ Display Computer Activity (default after MACL)

F.3 *MONTIOR FUNCTIONS (ALSO SHOWN ON DISPLAY)*

Memory

E ⌟	Set Physical Examine mode (default after MACL)
xE ⌟	Set Virtual Examine mode. Map via page table x.
xxxxxxxx/ nnnnnn zzzzzz ⌟	Examine and Change Content of memory address x from n to z. x is 24 bits at Physical and 16 bits at Virtual Examine.
xxxxxx < yyyyyy ⌟	Dump Content of memory from address x to address y. Select 64K area of last Examine.

Registers

xxRy/ nnnnnn zzzzzz ⌟	Examine and Change Content of register Ry on level xx from n to z. Ry may be written as R0=S, R1=D, R2=P, R3=B, R4=L, R5=A, R6=T, R7=X.
xx < yyRD ⌟	Dump Registers R0 to R7 from level x to level y.
U/ nnnnnn	Content of User Register is n
OPR/nnnnnn zzzzzz ⌟	Change Operators Panel "Switches" from n to z

Internal Registers

Ixx/ nnnnnn	Content of Internal Register No. x is n x (4 bits): 0 = PANS 1 = STS 2 = OPR 3 = PSR 4 = PVL 5 = IIC 6 = PID 7 = PIE 10 = CSR 11 = ACTL 12 = ALD 13 = PES 14 = PCR 15 = PEA
Iyy/ nnnnnn zzzzzz ⌟	Deposit z in Internal Registers No. y (n is dummy) y (4 bits): 0 = PANC 1 = STS 2 = LMP 3 = PCR 5 = IIE 6 = PID 7 = PIE 10 = CCLR 11 = LCIL 12 = UCIL 15 = ECCR
IRD ⌟	Dump Internal Registers 0 - 15 (only in STOP)
xx < yyRDE ⌟	Dump Scratch Registers from level x to level y

Deposit Rules

Content is only changed by zzzzzz ⌟ in STOP mode and by zzzzzzDEP ⌟ in STOP or RUN mode.

Content is unchanged by ⌟ in STOP or RUN mode and by zzzzzz ⌟ in RUN mode (? is answered).

Explanations:

☐ = Control Panel Button
⌟ = Carriage Return
n = computer answer

All other characters are typed by Operator.

APPENDIX G

ND-100 TECHNICAL SPECIFICATIONS

G.1 *SPECIFICATIONS*

Processor:

Microprocessor cycle time:	190 ns/150 ns (fast option)
CACHE memory size:	1K/31 bits
Paging overhead with CACHE:	0
Paging overhead without CACHE:	50 ns

Memory:

Maximum virtual memory address space:	64 K words
Maximum physical memory address space:	512 K words normal address mode 16 M words extended address mode
Access time for Local Memory:	read 320 ns write 200 ns Add 40 ns if correction
Error checking and correcting memory:	22 bits, single bit detection and correction All double bit errors are detected
Battery stand-by power for memory:	Minimum 18 minutes

Interrupt System:

16 priority interrupt levels each with 8 registers

Context block switching time:	Min. 5.0 µs. Typical 7.5 µs
External interrupt identification time:	3.3 µs typical

I/O System:

Maximum DMA rate/channel to local memory:	1.8 M words

G.2 *PHYSICAL*

ND-100 CPU Crate, Rack Mountable:

Dimensions

Height:	400 mm
Width:	482 mm
Depth:	505 mm

Can be mounted in 19 inch cabinets of various heights, depending on configuration.

Power:	230V, range 198 - 264V
	(115V, range 90-132V)
	45-440 Hz
	Max. 2 Amp. 230V

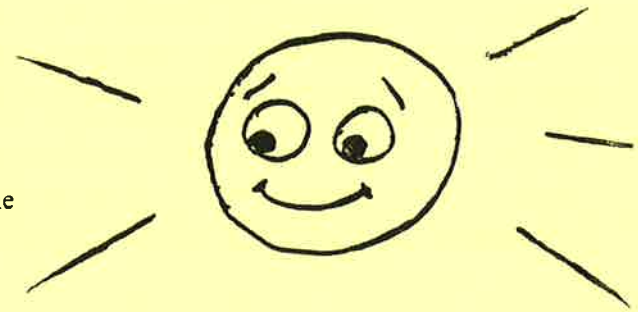
* * * * * * * * * * SEND US YOUR COMMENTS!!! * * * * * * * * * *

Are you frustrated because of unclear information in this manual? Do you have trouble finding things? Why don't you join the Reader's Club and send us a note? You will receive a membership card - and an answer to your comments.

Please let us know if you

* find errors
* cannot understand information
* cannot find information
* find needless information

Do you think we could improve the manual by rearranging the contents? You could also tell us if you like the manual!!



* * * * * * * * * HELP YOURSELF BY HELPING US!! * * * * * * * * *

Manual name: Manual number:

What problems do you have? (use extra pages if needed) _____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Do you have suggestions for improving this manual? _____

_____

_____

_____

_____

_____

_____

_____

_____

Your name: _____ _____ _____ Date: _____

Company: _____ _____ _____ Position: _____

Address: _____

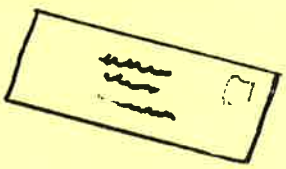
_____

What are you using this manual for? _____

_____

_____

Send to: Norsk Data A.S.
Documentation Department
P.O. Box 4, Lindeberg Gård
Oslo 10, Norway



Norsk Data's answer will be found on reverse side

Answer from Norsk Data _____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

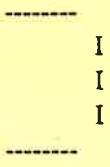
_____

_____

_____

_____

Answered by _____ _____ _____ _____ Date _____



Norsk Data A.S.

Documentation Department

P.O. Box 4, Lindeberg Gård

Oslo 10, Norway