NORD-100
Functional Description

NORSK DATA A.S

NORD-100
Functional Description

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
08/80	Original Printing — Version 01

NORD-100 FUNCTIONAL DESCRIPTION
Publication No. ND-06.015.01

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

PREFACE

To The Reader

The NORD-100 Functional Description describes the NORD-100 computer architecture and principles from a hardware standpoint. The main building blocks and their functions will be described. Since this is a hardware manual, it should be of interest to all technical and maintenance personnel who wish to gain detailed information about the NORD-100 computer system. Since the NORD-100 computer system is designed as an integrated system of hardware and software, this manual should also be of interest to software personnel.

Prerequisite Knowledge

Some knowledge of digital techniques and computer systems in general is recommended.

It is also assumed that the reader of this manual has some knowledge of the NORD-100 instruction set and assembly programming.

The Manual

This manual is meant to be read from beginning to end since some sections assume knowledge of the previous sections. But one section may be read independently. After an introduction in Section I, the manual starts in Section II with a description of the central processor. In Section III the part of the system located closest to the central processor, the memory management system, is described. The part that connects the different building blocks of the system together, the NORD-100 bus, is described in Section IV. The storage system is discussed in Section V. The communication with peripherals, the input/output system, is described in Section VI. The operator's interaction is taken care of in Section VII, while the power supply is given a short description in Section VIII. Section IX gives more detailed and specified information of certain parts. Appendixes are mostly references for special information while working with the NORD-100.

Related manuals containing more information of the NORD-100 computer system are:

NORD-100 Reference Manual (ND-06.014)
NORD-100 Input/Output System (as yet not released)
NORD-100 Microprogramming Description (06.018)

TABLE OF CONTENTS

\+ + +

Section: *Page:*

Section: *Page:*

Section: *Page:*

SECTION III — THE MEMORY MANAGEMENT SYSTEM

Section: *Page:*

SECTION IV — NORD-100 BUS SYSTEM

SECTION V — THE NORD-100 STORAGE SYSTEM

Section: *Page:*

Section: *Page:*

SECTION VI — THE INPUT/OUTPUT SYSTEM

Section: *Page:*

Section: *Page:*

Section: *Page:*

SECTION VIII — NORD-100 POWER SUPPLY

SECTION IX — MISCELLANEOUS

Section: *Page:*

APPENDIXES

SECTION I

NORD-100 ARCHITECTURE

I.1 *INTRODUCTION*

NORD-100 is a 16 bit general purpose single board computer. It is general purpose in the sense that it has both software and hardware available for most computer applications. The maximum address space is 64K words (128K bytes) without the memory management system, and 16M words (32M bytes) with the memory management system. It is completely software compatible with NORD-10/S and runs the same operating system, SINTRAN III/VS.

I.2 THE INSTRUCTION SET

Although a standard NORD-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 blocks

- fixed or floating point arithmetic

The 48 bit floating point instructions add, subtract, multiply and divide and have a 32 bit mantissa and a 16 bit exponent. As an option, the NORD-100 may be equipped with a 32 bit floating point format.

For efficient system control, specially tailored privileged instructions are included, such as loading and storing of the register blocks and interprogram level read/write operations.

The NORD-100 is microprogrammed. All instructions are executed by firmware residing in a 2K by 64 bit read only memory (ROM) called the microprogram control store.

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

This makes it possible for software to extend the NORD-100 instruction set for special applications.

I.3 ADDRESSING MODES

A variety of addressing modes may be used:

- Program counter relative addressing
- Indirect addressing
- Pre-indexed addressing
- Post-indexed addressing
- Combinations of the above mentioned modes

The address arithmetic is implemented as microprogram routines. This implies that the addressing structure of NORD-100 can be changed by rewriting the micro-program.

I.4 THE BUS STRUCTURE

The main highway for addresses and data in the system is the NORD-100 BUS, a multiplexed address and data bus. All communication between NORD-100 modules is provided by this bus, except communication between the CPU and the memory management module (including the CACHE memory).

Since both memory and device interfaces are connected to the NORD-100 bus, the CPU has the same easy access to peripherals as it has to memory.

I.5 FUNCTIONAL MODULES

I.5.1 *GENERAL*

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

Communication between NORD-100 functional modules is done through an advanced high-speed bus called NORD-100 bus. The NORD-100 bus is arranged as a printed backplane containing 12 positions for module connections. The bus can be extended to any number of other NORD-100 busses by a driver and a receiver module.

Figure I.5.1 shows the NORD-100 module connection.

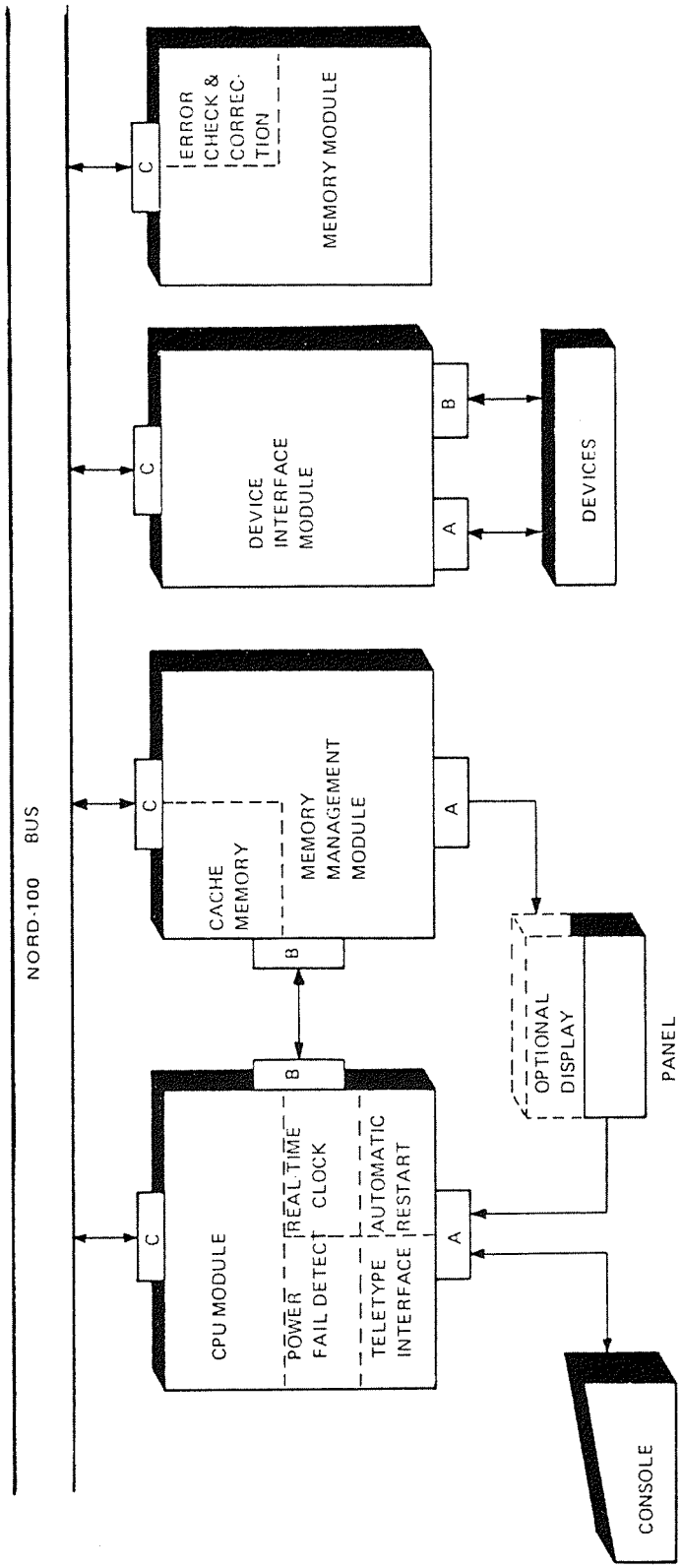


Figure I.5.1: NORD-100 Module Connection

I.5.2 *THE 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 bauds/bps (bits per second)
- power fail and automatic restart

THE PROCESSOR:

NORD-100 CPU is a 16 bit parallel processor controlled by a microprogram. The following is implemented in the microprogram:

- all instructions
- operator communication
- built-in test routines
- bootstrap loaders
- the address arithmetic

One microinstruction is fetched and executed in a certain time, which is the internal CPU cycle time. The NORD-100 CPU is delivered in two versions, with either 190 ns or 150 ns internal CPU cycle time.

INSTRUCTION PREFETCH:

A fast processor should not have to wait for instructions. In order to reduce instruction fetch waiting time, the NORD-100 CPU will fetch the next instruction at the same time as the current instruction is executed.

THE INTERRUPT SYSTEM:

The NORD-100 has a 16 level priority interrupt system, which permits the processor to be interrupted by conditions outside or inside the system. To each level is assigned a complete set of working registers and these registers are located in a high-speed register file on the CPU module. With this architecture, context switching consists of selecting another set of working registers. This is done by the microprogram.

I.5.3 *THE MEMORY MANAGEMENT SYSTEM*

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

- 64K words (128K bytes) virtual address range for each users 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:

The *paging system* can work in two modes:

The normal mode, which is compatible with the NORD-10/S paging system. This maps a 16 bit virtual address (describing a 64K word virtual memory) into a 19 bit physical address. In this mode the physical address space can be extended up to 512K words (1M byte).

The extended mode, which is special for NORD-100, maps the 16 bit virtual address into a 24 bit physical address. The NORD-100 can then cover an address range of 16M words (32M bytes).

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

Four page tables hold the physical page numbers assigned to active programs. These tables are located in high speed registers, reducing paging overhead to practically zero.

THE MEMORY PROTECTION SYSTEM:

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 program 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.

I.5.4 *THE MEMORY SYSTEM*

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

- 1K word (2K byte) CACHE memory
- up to 16M word (32M byte) main memory
- memory channel to the NORD-10/S big multiport memory system

MAIN MEMORY:

Main memory can have any size from:

32K words to 16M 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:

- detect and correct single bit errors
- detect and report all double errors and most multiple errors.

One memory module contains maximum 64K words (128K bytes), and also includes error checking and correcting for this module.

CACHE MEMORY:

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

The presence of cache memory will reduce average memory access time significantly. Cache is a high speed bipolar memory of 150 ns cycle time.

Access time is virtually zero because it works in parallel with the microprogram.

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

MULTIPORT MEMORY:

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

If direct memory access (DMA) devices with high transfer rate are to be used, the multiport memory system should be employed to avoid cycle stealing from the CPU.

I.5.5 *THE INPUT/OUTPUT SYSTEM*

The NORD-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 the NORD-100 with:

- CPU controlled, programmed input/output (PIO)
- direct memory access (DMA)

PROGRAMMED INPUT/OUTPUT – PIO

Program controlled input/output always operates via the A register. Each word or byte of input/output has to be done by programming.

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.

The DMA channel is connected to main memory by the CPU bus control on cycle steal basis.

To avoid cycle steal, and for higher performance, the DMA channel can be connected to a separate port on the multiport memory system.

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 1.8M words per second.

I.6 THE CABINET

Figure I.6.1 shows a standard NORD-100 cabinet, which is 54 cm wide, 96 cm high and 84 cm deep.

At the top, a dual floppy disk is located. The next module is for the standard panel and the optional display. The panel is the right half, containing four push buttons and a key for locking of the panel.

The left half contains the removable display, which is driven by an independent microprocessor, located on the memory management module. The microprocessor receives data to be displayed from the CPU microprogram and from a digital clock on the memory management module. Cache hit ratio and degree of utilization of the CPU are also monitored by the display microprocessor.

Behind the two next modules the NORD-100 card crate, including power supply is located. By removing the front covers, the card crate can be pulled out on runners.

How the rest of the cabinet is to be used, depends on the size of the system (e.g., more than one card crate), and what kind of peripherals the system is equipped with (e.g., mag. tape, big disks, etc.). At the bottom there is space for a disk drive. This is sufficient for a small standard NORD-100 system. For big systems you may need more than one cabinet.

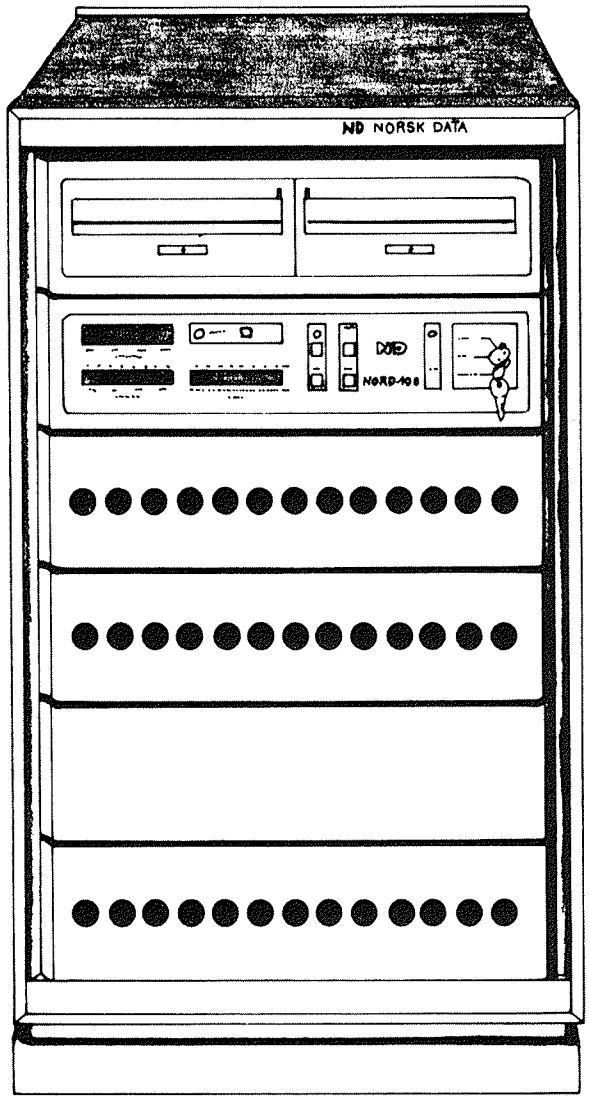


Figure I.6.1: Standard NORD-100 Cabinet

SECTION II

CENTRAL PROCESSOR

II.1 GENERAL

NORD-100 is based upon a microprogrammed CPU architecture. A micro-programmed architecture means:

- a large portion of system control is performed by a read only memory (ROM)
- this ROM contains long words called microinstructions
- each microinstruction contains bits to control each of the main elements in the system
- changes in the machine's instruction set are simple to make by rewriting the microprogram
- The hardware package count is reduced giving smaller computers

The CPU performs arithmetic and logic operations, instruction decode and execution, and allocation control of the NORD-100 bus.

II.2 INTERNAL COMMUNICATION

The communication inside the CPU is performed by means of the internal data bus (IDB). A bus is a highway for information, where only one word of information may travel at a time. Instructions, operands, addresses and data are transmitted on the internal data bus under control of the microprogram. The microprogram enables information onto the IDB from a certain source, and gives enable signals to the destination in the CPU where the information is needed.

Figure II.2.1 shows how the IDB communicates with the central parts in the CPU, the memory management and cache, and with the sources connected to the NORD-100 bus.

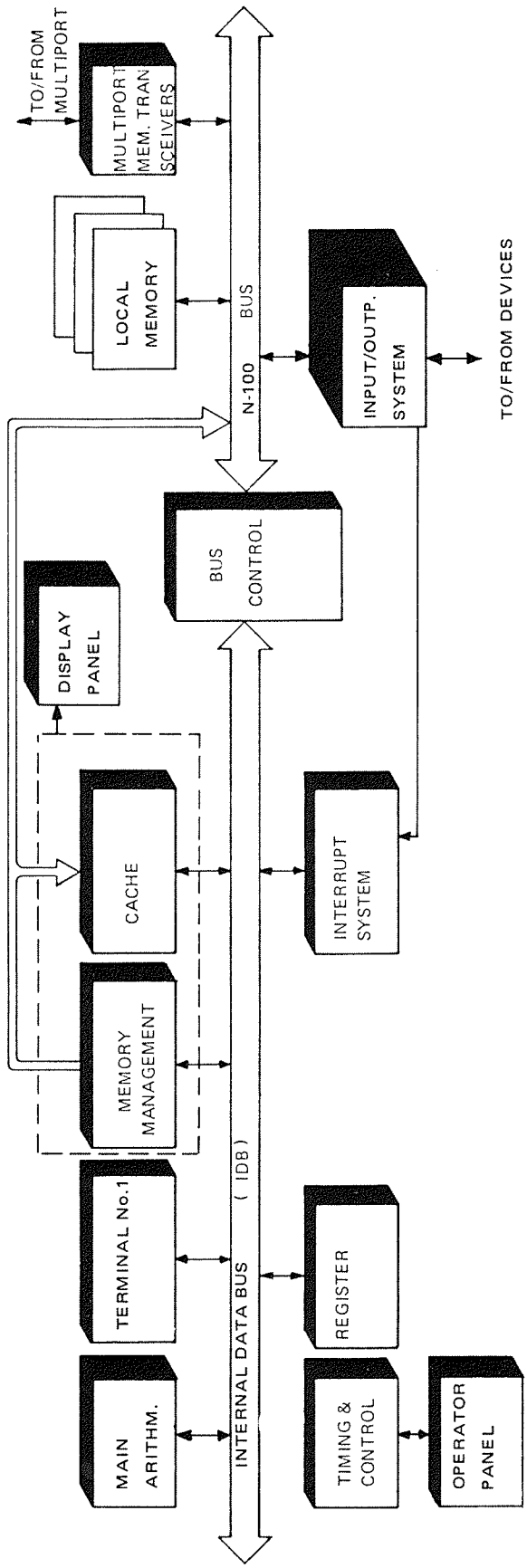


Figure II.2.1: NORD-100 Bus Structure

The main parts communicating with the IDB and with the NORD-100 bus are the following:

— Timing and Control

This part controls the sequence of events occurring during the execution of one microinstruction. It generates the correct timing signals from the clock circuitry.

— Main Arithmetic

This is where the arithmetic and the logic functions are performed, i.e., the part in the processor that computes. It also contains the current register block.

The 16 bit virtual addresses are also calculated here under control of the microprogram. They are sent to the memory management system (MMS). If no MMS is present, the addresses will be sent out as physical address to the memory system via the NORD-100 bus directly.

— Register File

The working register blocks for levels not currently running are stored here. The register file has two-way communication with the IDB, for save and unsave of the current register block.

— Operator's Panel

This is the communication between the push buttons on the front panel and the control part in the CPU.

— Interrupt System

The interrupt system handles interrupts by continually comparing the current priority level of the processor with the level of any interrupting devices. It identifies the device with the highest level above the processor's priority level, and controls the change from one level to another. This system communicates over the IDB with the other CPU parts.

— Terminal no. 1

The terminal interface no. 1 communicates directly with IDB.

— Memory Management

The main arithmetic presents the 16 bit virtual address for the memory management system, which maps this into a physical memory address on the NORD-100 bus, with the help of the page tables. This requires a two-way communication with the IDB.

— Cache

Cache memory is connected directly to the IDB. This gives faster access of the addresses, and faster presentation of data back to the CPU if it is contained in the cache memory.

— Display Panel

The display panel is controlled by a special display microprocessor on the memory management module.

— Bus Control

The NORD-100 bus is controlled by the CPU board. All requests for the NORD-100 bus and allocation of this bus are handled here.

— Input/Output System

CPU controlled programmed input/output devices have two-way communication with the NORD-100 bus. Direct memory access devices, which operate directly on memory, communicate over the NORD-100 bus.

— Local Memory

This contains memory modules with up to 64K words on each module. Local memory is directly connected to the NORD-100 bus to keep the access time low. To accomplish error checking and correction six extra bits are added to each 16 bit word in local memory.

— Multiport Memory

A multiport memory transceiver is necessary for communication between the NORD-100 bus and the big multiport memory system.

II.3 FUNDAMENTAL BUILDING BLOCKS

II.3.1 *General Considerations*

A microprogrammed CPU architecture consists of two parts: the microprogram control store and the control decode. The microprogram control store is a 2K by 64 bit, fast read only memory (ROM). Here the microprogram which controls the operation of the computer is stored. The microprogram mainly performes the execution of the machine instructions. However, some part of the microprogram performs other functions:

- operator's communication
- built-in test routines
- bootstrap loaders

To allow dynamic microprogramming, a 256 word by 64 bit writable control store is optional. This gives the possibility of extending the NORD-100 instruction set for special applications. The address arithmetic is also implemented in microprogram. This means that the addressing structure of NORD-100 can be changed by rewriting the microprogram.

A pipeline register is placed at the output of the microprogram control store. This register contains the microinstruction currently being executed. The information enters the control decode circuitry, where the microinstruction is decoded to activate a set of specific control lines to perform the given function.

Figure II.3.1 shows the fundamental building blocks.

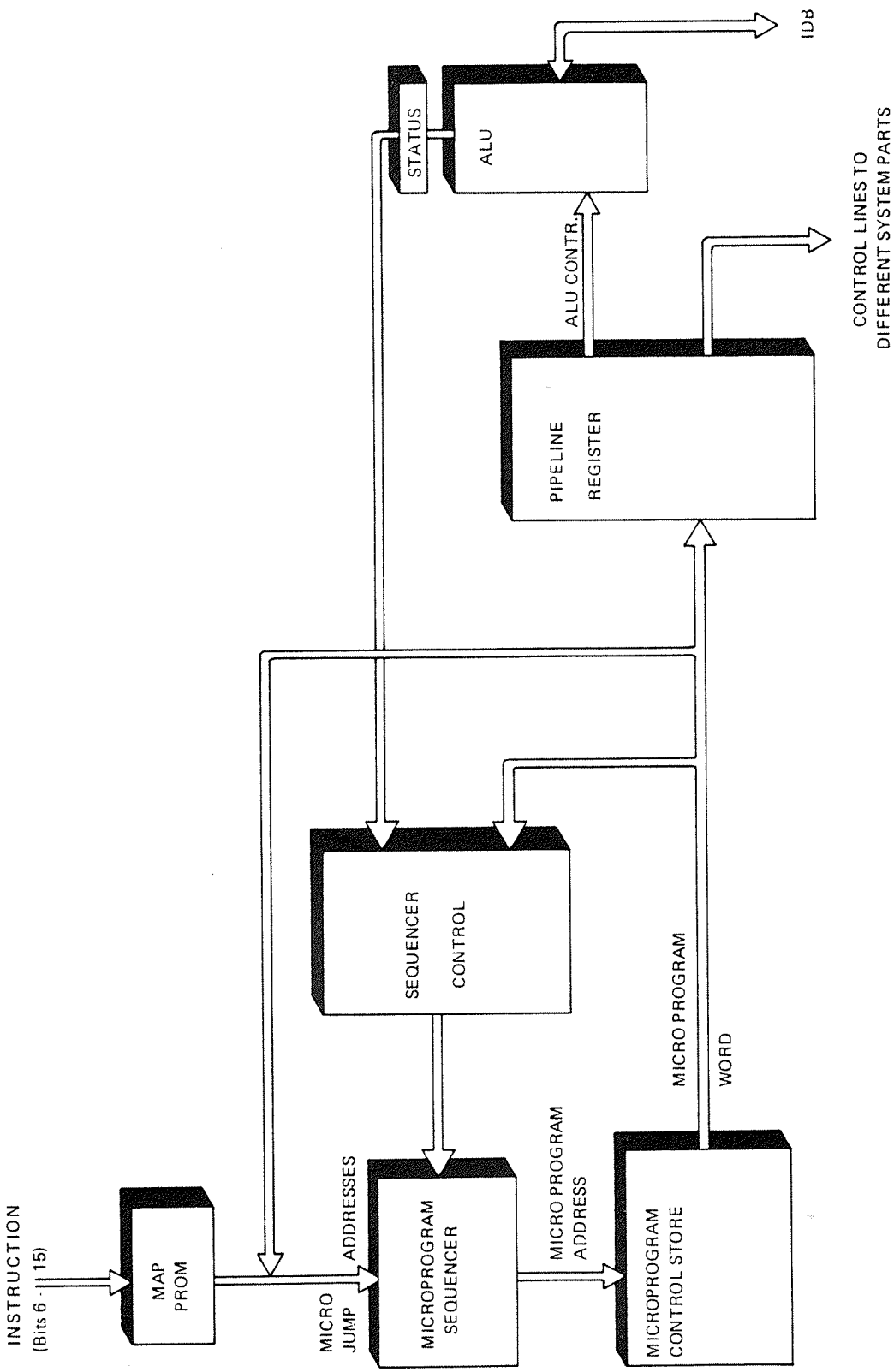


Figure II.3.1: Fundamental Building Blocks

In the next paragraphs, the working principles of the CPU will be described.

II.3.2 *INSTRUCTION EXECUTION OVERVIEW*

To find the microprogram entry point of an instruction, the upper 10 bits (bits 6-15) of the instruction itself are used as address to a special PROM, called the map (or the mapping PROM). The map output is the entry point to the microprogram for this machine instruction. It is loaded into the microprogram sequencer, which uses it as address to the microprogram control store.

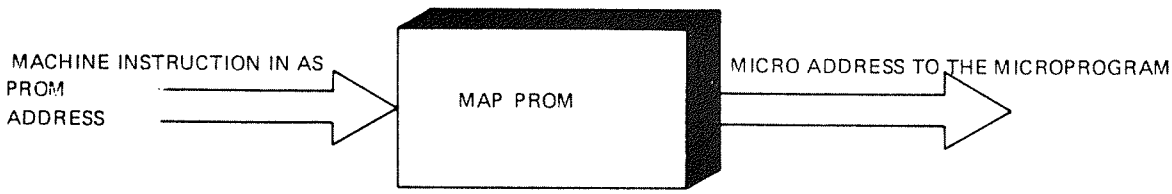


Figure II.3.2: The Mapping PROM

Several microinstructions may be needed to execute one machine instruction. The microprogram sequencer controls the sequence of these micro instructions. After the access time, delay of the PROM, the microinstruction will be present at the output of the microprogram control store. Each microinstruction contains bits to control the different elements in the system.

One part of the microinstruction controls the microprogram sequencer, together with status information from the execution of the last micro instruction. In this way, the sequence of microinstructions may be dependent on status information, etc. Subroutine linkage in the microprogram is also taken care of by the sequencer, it has a built-in stack for that purpose.

Branch addresses in the microprogram are fed directly back and to the microprogram sequencer, to be used as the next control store address. All the micro instruction bits not controlling the sequencer go to the pipeline register. The pipeline register bits are either used directly as control lines for the data hardware or are decoded to set specific control lines. So the output of the pipeline register is a set of control lines, containing the data hardware as in the ALU (arithmetic logic unit) for execution of the current microinstruction.

As indicated above, each microinstruction not only contains bits to control the data hardware, but also bits to define the address of the next microinstruction to be executed. This is done by the sequencer control bits and by the branch address bits.

The pipeline register allows the next microinstruction fetch to occur in parallel with the data operation of the current microinstruction. The parts of the micro instruction which are fed back to the sequencer to determine the next address is clocked into the microprogram sequencer at the same time as in the pipeline register is clocked. The fetch of the next microinstruction can therefore be performed while executing the current one in the data hardware.

The time interval from the pipeline register gets the current microinstruction until a new microinstruction is ready at the input of the pipeline register, constitutes a single clock cycle. During this time, the operations in the other CPU elements occur.

Most of the CPU elements are LSI (large scale integration) and MSI (medium scale integration) chips, which are controlled directly by the bits from the pipeline register.

II.4 INSTRUCTION FETCH AND EXECUTION

II.4.1 *INSTRUCTION READOUT*

Refer to Figure II.4.1.

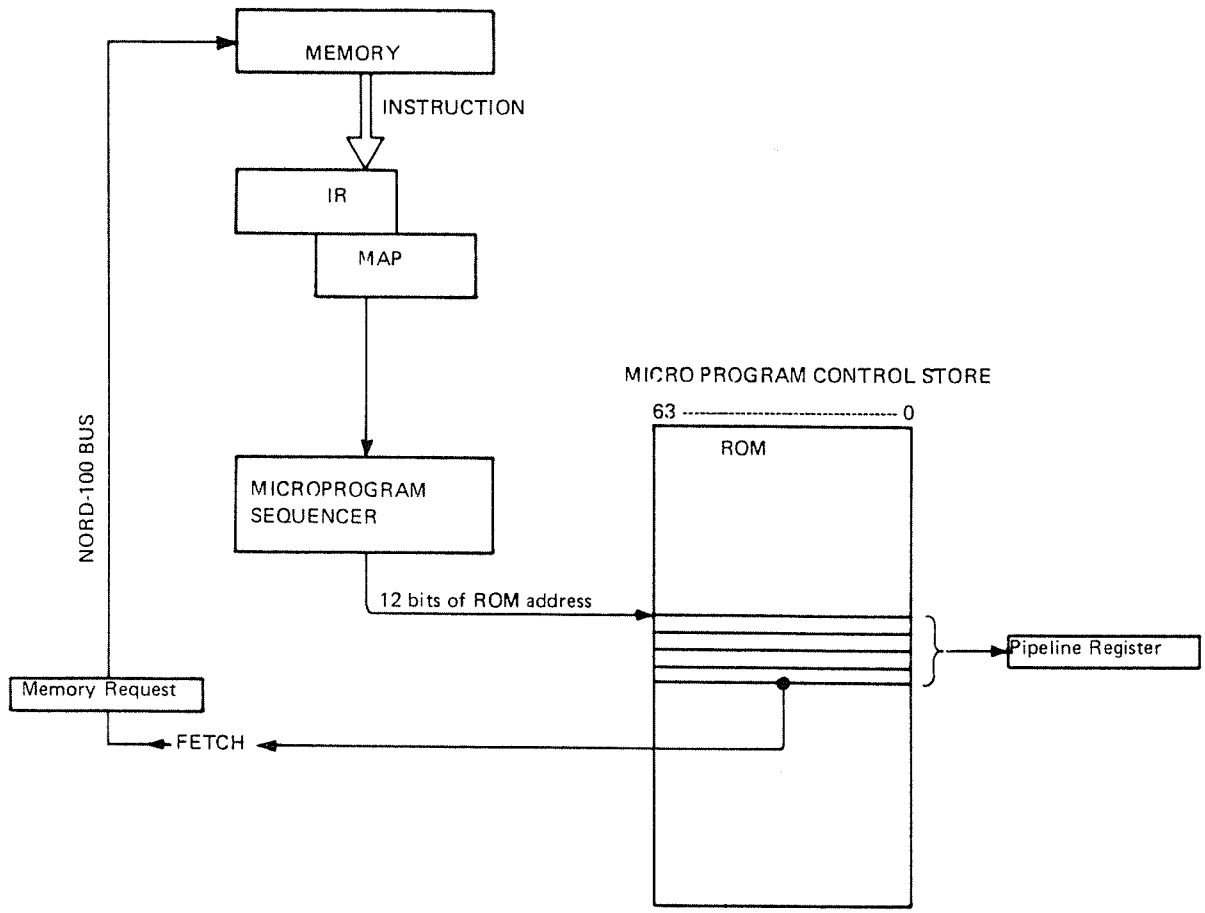


Figure II.4.1: Instruction Readout

The machine instructions to be executed reside in memory. When the micro-instruction sequence of one machine instruction is finished, a fetch for a new machine instructiuon is issued. The program counter is enabled onto the NORD-100 bus, and a read request is sent to memory. After the access in memory, the instruction is loaded into the instruction register (IR) and also into the 2K by 12 bit mapping prom (MAP), as an address. The MAP then supplies the address of the first microinstruction to execute the machine instruction. The address is loaded into the microprogram sequencer which addresses the micro-program control store. The output of the microprogram control store, together with the timing circuitry, controls the operation of the CPU. The microprogram sequencer manages the microprogram control store address and their sequence. The map contains two almost equal entry point maps, one containing all NORD-100 instructions, and one containing only the unprivileged instruction set. Which map to use is selected by the most significant ring bit of the active PCR (Paging Control Register).

The operations specified by one microinstruction take normally either 190 ns or 150 ns, depending on whether it is a slow or a fast version of the NORD-100. This time it is referred to as a micro cycle, or the internal CPU cycle time. When a micro cycle is completed, the next microinstruction has already been read out from the microprogram control store. When the sequence of microinstructions is finished, a new fetch will be issued, and the computer is ready for execution of a new machine instruction.

II.4.2 *PREFETCH*

NORD-100 uses prefetch. That is, the next instruction is fetched simultaneously with the execution of the current one. A fetch request for the next instruction will always be issued when NORD-100 starts executing an instruction. In this way the instruction fetch waiting time will be reduced, so that the processor does not have to wait for the instructions fetched from memory. The speed of the processor will be increased considerably. When the prefetched instruction shall be executed, it will most often be available.

Figure II.4.2 shows the sequence under instruction fetch.

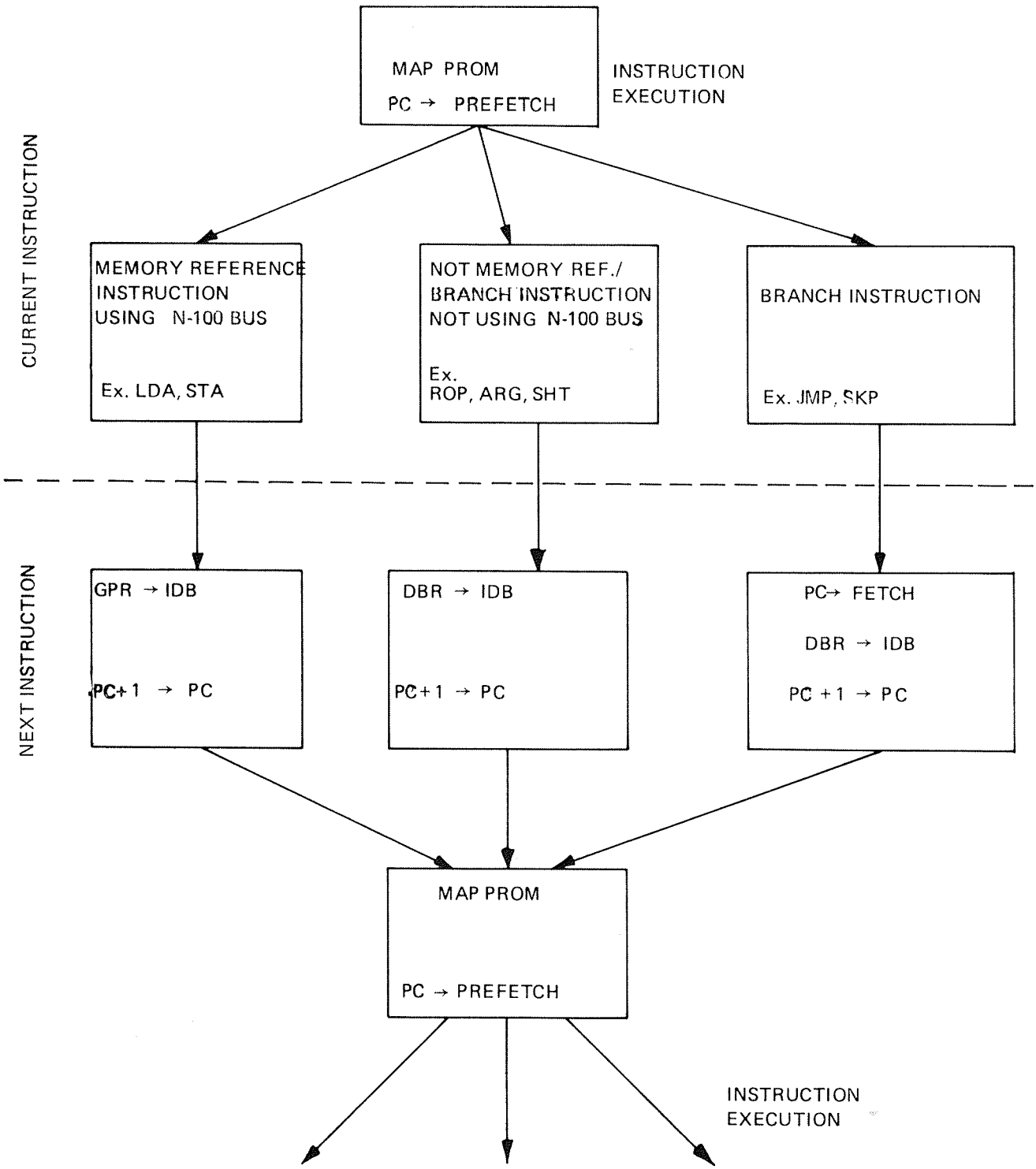


Figure II.4.2: Instruction Prefetch

The current machine instruction to be executed uses the MAP to start execution. At the same time, the next instruction will be fetched from memory in a prefetch cycle.

The instruction can be divided into three main groups according to the "prefetch strategy" they use:

1. Memory Reference Instructions

   These instructions specify operations on words in memory, and therefore use the NORD-100 bus.

   Example:

   LDA, STA, ADD

   These instructions will save the next instruction in a CPU register (GPR – general purpose register).

2. Branch Instructions

   These instructions change the subsequent sequence of instructions. The program counter (PC) will not become PC + 1, but a value dependent on on the instruction operands.

   Example:

   JMP, SKP, BSKP

   These instructions will never cause the prefetched instruction to be read from the NORD-100 bus. A new fetch request is issued instead.

3. Other Instructions

   These are instruction which do not make memory references, i.e., do not use the NORD-100 bus and don't change the sequence of the instructions. These instructions are mostly operations inside the CPU itself.

   Example:

   RADD, SHT, SAA

   After these instructions the next instruction can be read from the NORD-100 bus.

If the current instruction is a memory reference instruction, the next instruction, the prefetched one, will be loaded from memory and into the GPR by the instruction itself. Refer to Figure II.4.2.

When the current instruction is finished, the next instruction found in the GPR. When execution of that instruction starts, the incremented program counter is used to perform a new prefetch request for the next instruction and the same sequence of events will be repeated.

If the current instruction is one in group three, the prefetched instruction will be loaded (automatically by hardware) into the DBR (Data Bus Read register). Because these instructions do not use the NORD-100 bus, like the memory reference instructions, the instruction doesn't have to be saved in the GPR.

When the current instruction is finished, the prefetched instruction is found in the DBR. The PC is normally incremented by one. The instruction is then executed, and a prefetch is issued simultaneously.

If the current instruction is a branch instruction, an unnecessary prefetch has been made. The prefetched instruction will then not be used. The new program counter will be used for an ordinary fetch, and the fetched instruction will be enabled from DBR to the internal data bus for start of execution. At the same time, the program counter will be incremented by one, and when the execution starts, this program counter will be used for a prefetch request.

So, in the case of branch instructions the prefetched instruction is skipped and a new instruction is found. Therefore, the maximum benefit of prefetch is gained only in a strictly sequencial program.

Prefetch will not generate a 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 1μs to the execution time compared to an ordinary STA.

II.4.3 *INSTRUCTION EXECUTION*

Refer to Figure II.4.3.

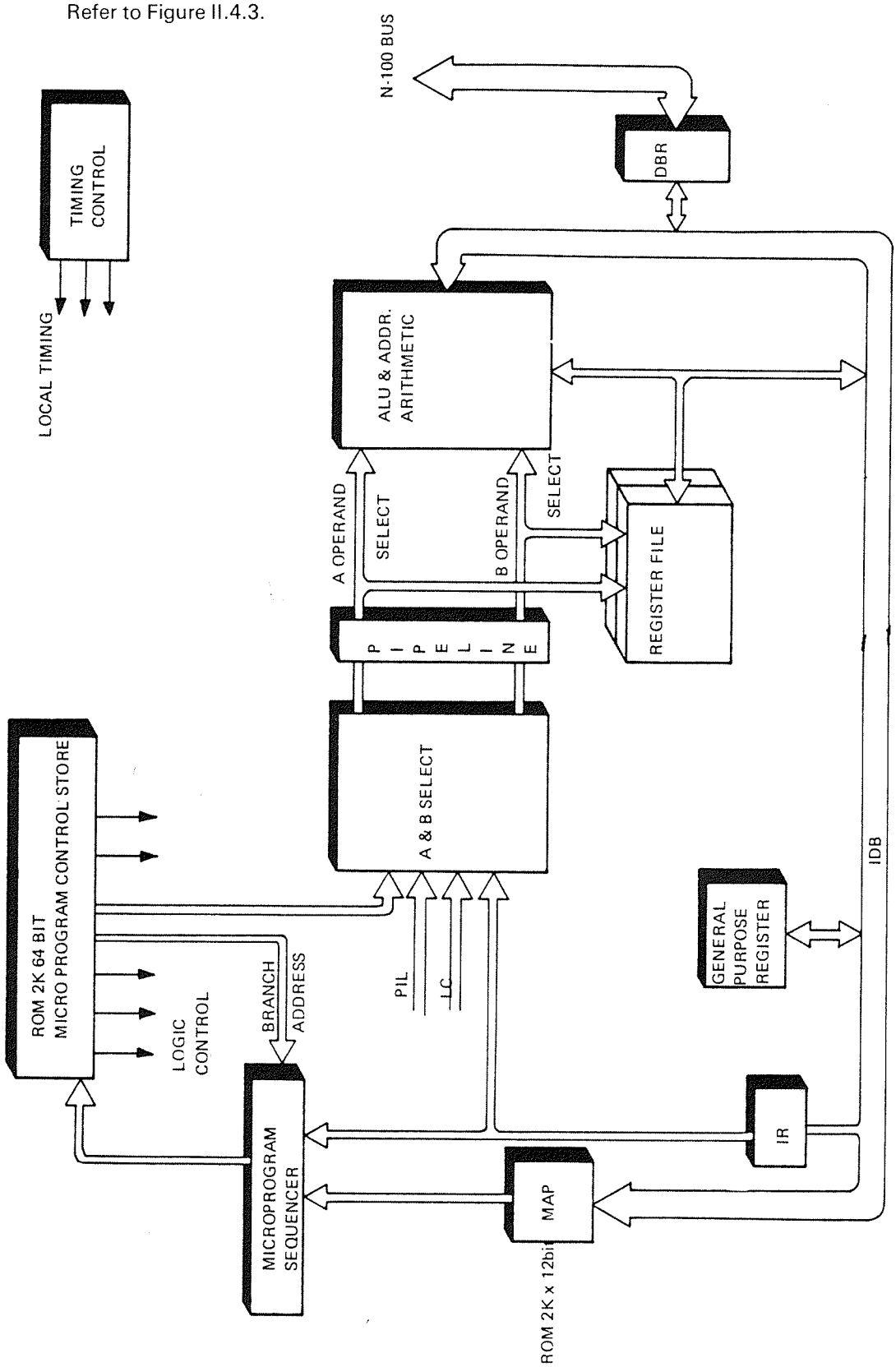


Figure II.4.3: The Instruction Part

An instruction to be executed will be found prefetched in the GPR or the DBR, or by an ordinary fetch after branch instructions. These two registers may be enabled onto the internal data bus (IDB). Bits 6-15 of the instruction (the op. code) will be the address to the MAP and bits 0-10 of the instruction will be loaded into the instruction register (IR).

The output of the MAP is the address of the first microinstruction which must be executed to perform the function of the machine instruction. The microprogram sequencer executes a jump to this address. The number of micro instructions to be executed varies with the different machine instructions, and the sequence and the microprogram addresses are determined by the microprogram sequencer.

The microprogram control store words are divided into fields, and each field controls parts of the processor, such as ALU, register file, I/O control, priority interrupt control, etc. Within each word, a field controls the sequencer, telling it how to generate the next microprogram address. The function performed by one word in microprogram control store is called a microinstruction and the time required to execute a micro instruction is called a micro cycle.

The microinstructions may be executed to fetch data from memory (for example, under an LDA instruction), perform ALU operations, shift registers, and so forth. Together with the timing, they will control the operation of the CPU. During these operations, the information from the IR may be needed to determine source/ destination registers in register operations, which address modus in memory reference instructions, the kind of shift mode in shift instruction, etc. This information may affect the microprogram sequencer, the A and B select, the shift linkage circuitry and the loop counter.

After the completion of each machine instruction, a branch will be made back to the instruction fetch part of the microprogram.

From here, there may be branches to other parts of the micro program. For example, the machine may receive an interrupt and be routed to an interrupt service routine address.

The ALU contains the working register set (refer to the register file) for the current level and performs all arithmetic, logic and shift operations on data within the CPU.

When changing from one program level to another after an interrupt, the register block in the ALU is saved in the register file. The working register block of the new level is then read into the ALU. This two-way communication is done over the IDB.

The ALU is controlled from the microprogram and information concerning which operands to use in ALU operations, are received from the A and B select inputs to the ALU.

These A and B operands are controlled by the microprogram, and are also used to address a specific word in the register file if a read or write operation in the register file is specified by the microinstruction. The A operand will then select the interrupt level to be affected, and the B operand will select one register on that level. The A operand can also control a bit mask generator, which generates a single bit among zeros on the IDB.

The A operand can have its value from:

1. The microinstruction directly, used for several purposes.

2. IR bits 3-5, used to select source register during register operations (COPY, etc.).

3. IR bits 3-6, used to generate single bits in bit instructions, and to select interrupt level in inter register read/write or register block instructions.

4. PIL register, used to address the current level in interrupt handling microprogram parts.

5. The loop counter, used for special microprogram purposes, mainly to control the bit mask generator.

The B operand can have its value from:

1. The microprogram directly.

2. IR bits 3-5, used to write in source register in SWAP.

3. IR bits 0-2, used to write in destination register in register operations.

4. The loop counter, used to scan through one register block during interrupt level change.

Only some typical uses of the A and B operands are outlined above. For a complete understanding, the reader has to examine the microprogram or its flow charts (in the NORD-100 Microprogram Description manual).

II.5 THE REGISTER FILE

Refer to Figure II.5.1.

There are 16 register sets in the NORD-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 are a total of 256 registers; these are referred to as the register file.

II.5.1 *THE 8 WORKING REGISTERS*

Status register

This register holds the 8 indicators described in the status indicator 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.

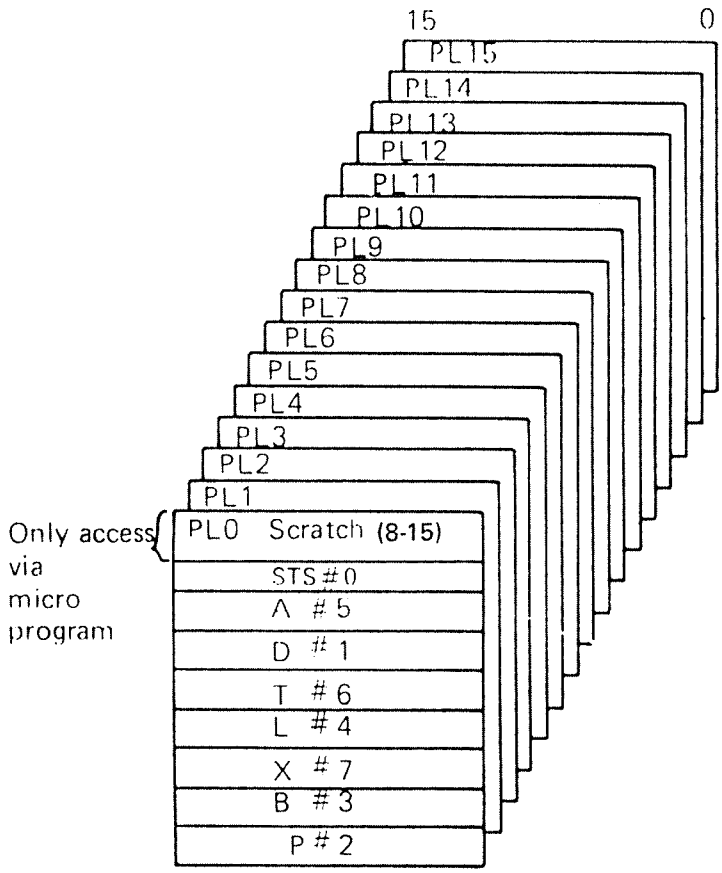


Figure II.5.1: The Register File

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

II.5.2 *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 a program.

Q — Dynamic overflow indicator.

Z — Error indicator. This indicator is static and remains set until it is reset by a 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. Note that TRR STS only writes bits 0-7 of the status register, not the whole register. Refer to Figure II.5.2.

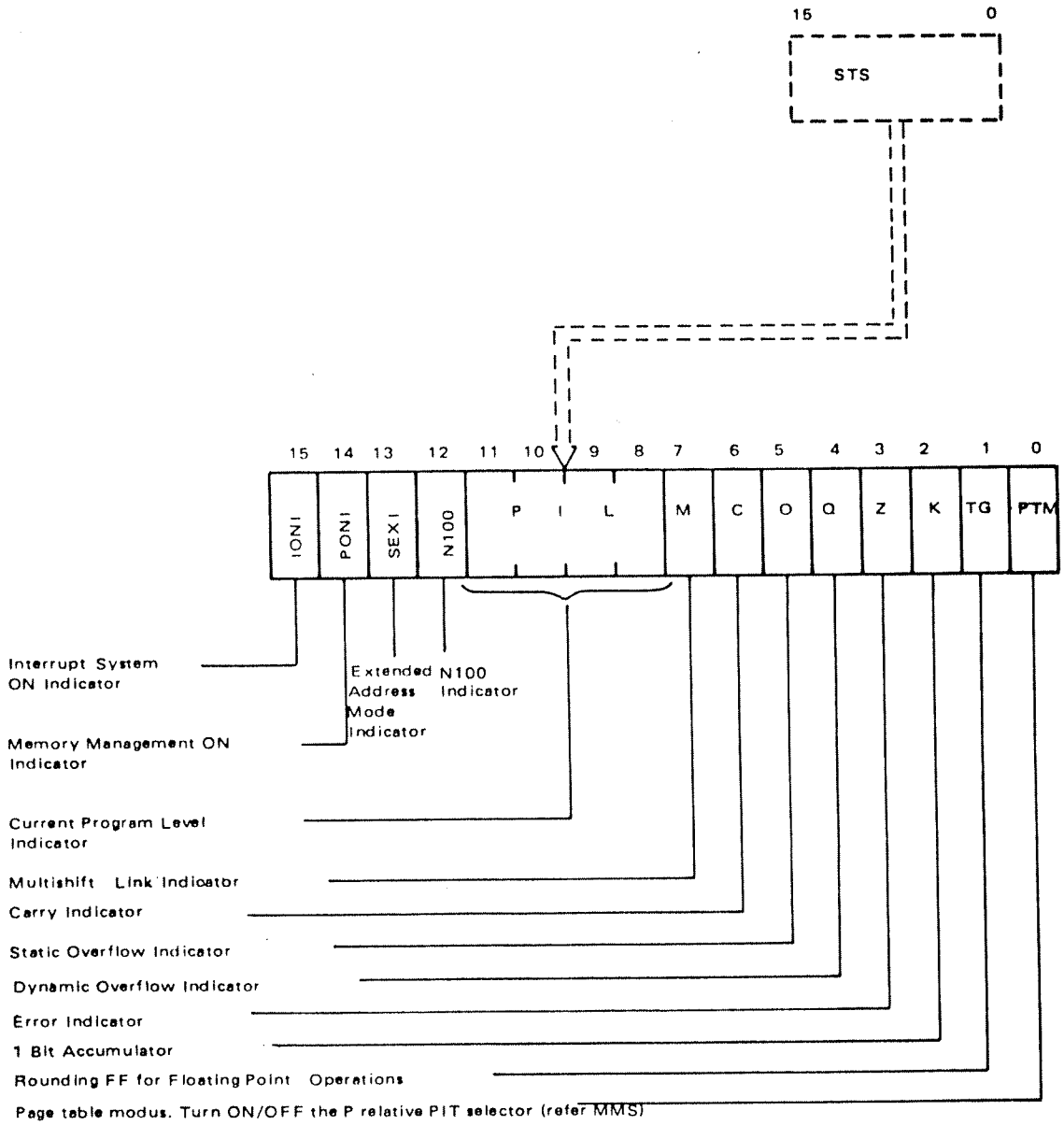


Figure II.5.2: Status Register Assignment

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

IONI Interrupt system ON indicator.

PONI Memory management ON indicator.

SEXI Extended indicator to show that the memory management system is in 24 bit extended addressing mode instead of the usual 19 bit addressing mode.

N100 N100 indicator to tell the operating system that this is a N100 machine.

PIL Current program level indicator.

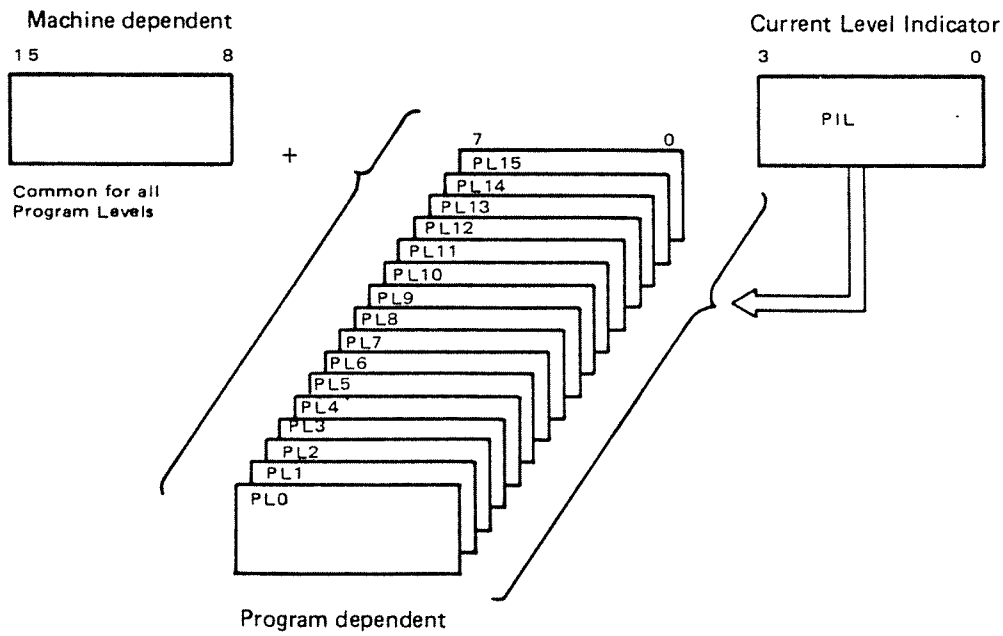


Figure II.5.3: The Status Register

Figure II.5.3 shows that STS bit 0-7, as described, has one copy on each level, while STS bit 8-15 is common for all levels.

II.6 MICRO PROGRAM SEQUENCER

II.6.1 *GENERAL*

The use of an advanced microprogram sequencer with a built-in stack has made it possible to take advantage of the latest microprogramming techniques: micro branching, micro subroutines and repetitive microinstruction execution.

II.6.2 *THE MICRO PROGRAM SEQUENCER*

The purpose of the microprogram sequencer is to generate the address to the microprogram control store, making it possible to fetch and execute a micro-instruction. The microprogram sequencer contains a micro address register with multiplexed input, a push/pop stack and an incrementer. Control lines provide the information needed to select the source of the next microinstruction address. It is possible to make branches in the microprogram, execute subroutines in the microprogram and carry out repetitive microinstruction execution.

There are two sets of control lines to the sequencer. One controls the input multiplexer for the micro address register. This line provides the micro address register with access to either the current microinstruction (REPEAT), an address stored in the push/pop stack (RETURN), the output of the incrementer (NEXT) or a direct address in (JUMP).

The push/pop stack (FIL0) is used to provide return address linkage when executing micro subroutines. It can be used for up to four return (link) addresses. A set of control lines from the sequencer control, controls the push/pop stack and determines whether the function being performed is a jump to a subroutine (PUSH), or a return from a subroutine (POP). It is also possible to hold the stack information (HOLD) or to load the top word (LOAD).

After a subroutine has been completed, a return to the address immediately following the jump to the subroutine instruction may be accomplished by selecting the stack as the source address (RETURN) and executing a POP at the same time.

If the incrementer is selected as the source address (NEXT), the sequencer will step through the microprogram.

The sequencer control bits are divided into three fields:

- sequencer control field
- branch address field
- conditional field

The sequencer control field selects the source of the next micro address, and controls the sequencer stack if a conditional sequence is not specified. These bits indicate the source of the next micro address:

1. The stack (RETURN)
2. A direct branch address (JUMP)
3. From an incrementer (NEXT)
4. The current one (REPEAT)

The operation of the built-in stack can be:

1. HOLD the stack
2. POP the stack
3. PUSH the stack
4. LOAD the current micro address + 1 into the stack

A direct branch address comes from the branch address field in the micro-instruction (bits 0-11). When this input is selected, a direct branch in the micro-program will be performed.

II.6.3 *SEQUENCING*

A microinstruction may specify two different sets of next address select control bits. Which set to use depends on the ALU (arithmetic logic unit) result of the last executed microinstruction, or on a number of other test objects originating in the CPU. This makes conditional branching possible. There is a special condition enable bit in each microinstruction that makes this two-way branch occur.

Prior to the testing microinstruction, one of the test objects must be selected by the microprogram. If the test object is true, one set of select control bits is used. If it is false, the other set is used.

II.6.4 *FUNCTIONAL FLOW*

Refer to Figure II.6.4.

Branch address to the microprogram sequencer, when JUMP is specified, can be generated by the mapping PROM, the interrupt hardware, the vector select or the microprogram word itself.

The execution of a machine instruction will always start with a micro address found in the map. When the map is selected as input to the microprogram sequencer, a jump will be made to the first microinstruction executing this machine instruction. The interrupt hardware (interrupt address buffer) and the vector select will be disabled under this operation.

Under the following sequence of microinstructions, further sequencing information may be needed about the instruction. This information is supplied as a 4 bit vector from the vector selector. These bits, together with 8 bits from the microprogram control store, give a 12 bits jump address. The micro program specifies the 8 most significant bits (MSB) in the branch address, while the 4 least significant bits (LSB) are taken from the vector selector. Input to the vector select are the following:

- IR (instruction register) bits 8 - 10 cause the branch address bits to depend on the address mode of the machine instruction.

- IR bits 0 - 3 are used to distinguish between different TRA and TRR instructions.

- RA (the A operand) is a vector used inside the execution of MOPC. It can be used to branch into a vector depending on a calculated variable in the microprogram.

- ALD (automatic load descriptor), four bits from the ALD switch setting, indicate what kind of automatic load is to be performed.

Which of these 4 inputs to be selected is controlled from the microprogram.

An interrupt may occur at any time, but is only paid attention to at the time of mapping jumps. At the time the interrupt is allowed, the 5 bit priority encoded interrupt vector, together with a fixed code, will make a vectored branch in the microprogram. The micro address is generated by the interrupt address buffer, which will force the micro address to the interrupt service vector. Under this operation the map prom and the vector select are disabled.

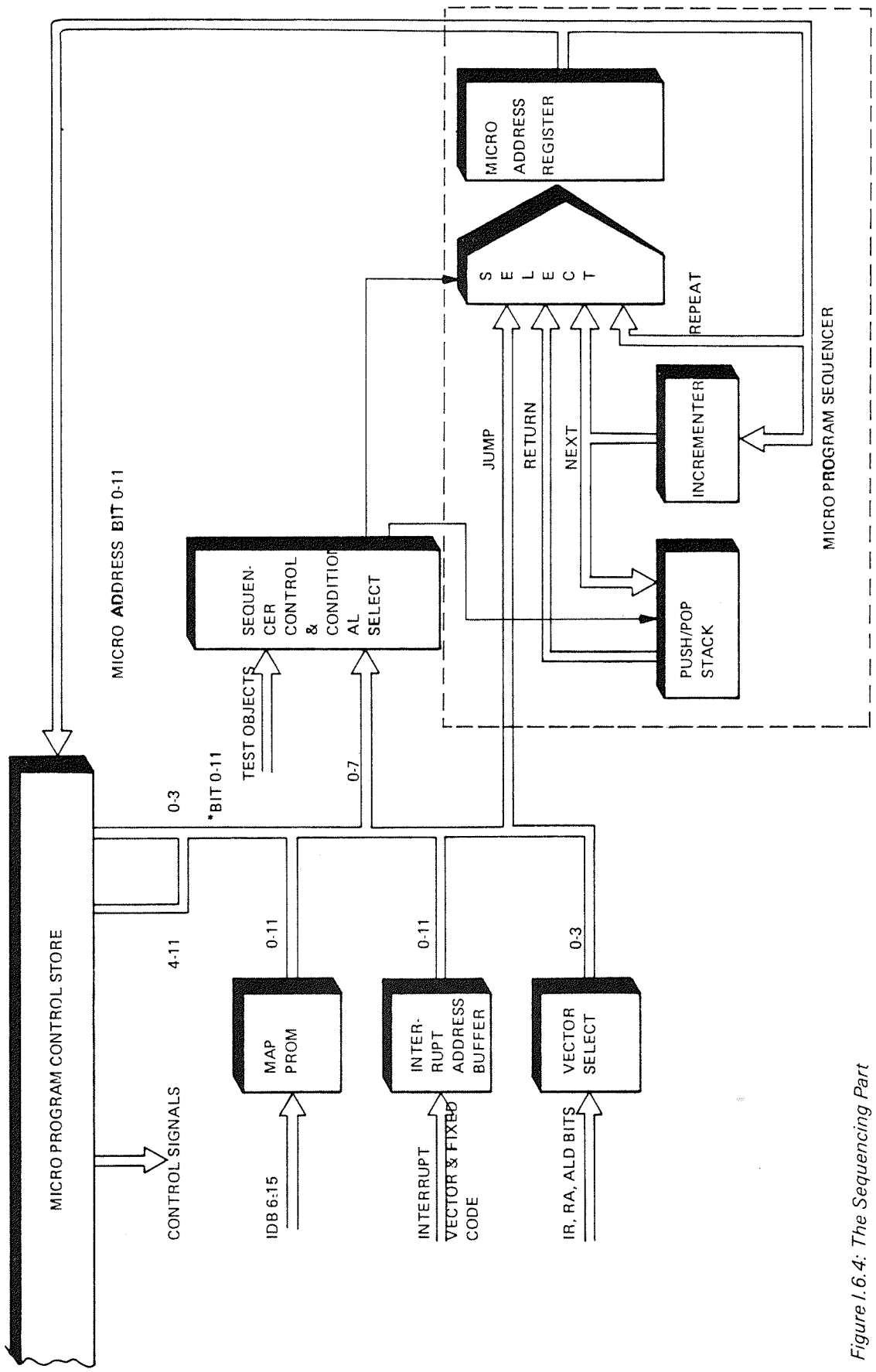


Figure I.6.4: The Sequencing Part

Example:

The LDA, B, X instruction is to be executed.

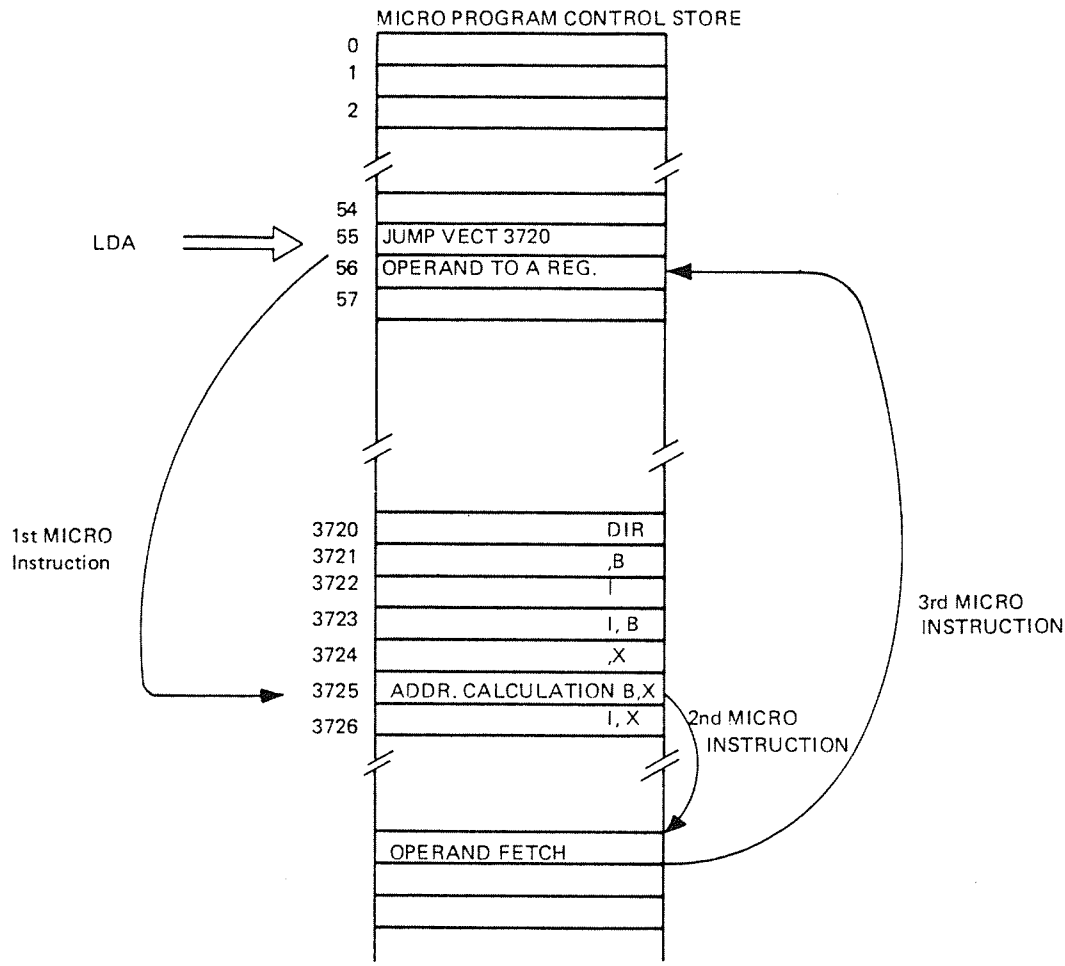


Figure II.6.5: LDA, B, X Execution

The output of the map PROM will give, for instance 55, as entry point. This number enters the microprogram sequencer and is used as address to the microprogram control store. This will be the entry point for all LDAs. The microinstruction will execute a vectorized jump, with IR bits 8 - 10 as the 3 lower branch address bits, and 3720 as branch address bits 4 - 11 given by the microprogram. This microinstruction (in 3725) will calculate the operand address by adding the displacement from the LDA instruction and the content of the B register and the X register. It also makes a branch to the microinstruction which will generate a read request for the operand. This micro instruction will make a RETURN back to location 56 in the microprogram, and when the operand is ready from memory, this will be loaded into the A register by the microinstruction located in location 56.

During this sequence of microinstructions, the next instruction, which has been prefetched, has been loaded into the GPR (general purpose register).

II.7 PIPELINE

Refer to Figure II.7.1.

The pipeline register is placed on the output of the microprogram control store.

When the execution of a machine instruction starts, the first microinstruction will be present at the output of the microprogram control store, i.e., at the input of the pipeline and microprogram sequencer. The bits to the micro program sequencer are for determination of the next microinstruction to be executed. The part of the microinstruction that goes to the pipeline register are the data manipulation control bits to the different system elements, for control of the data hardware.

When the clock pulse arrives, both the pipeline register and the microprogram sequencer are clocked at the same time.

The pipeline register then contains the microinstruction currently being executed because the microinstruction execution is performed by the activation of the different control lines from the pipeline register.

While the execution of the current microinstruction occurs, the address of the next microinstruction to be executed is sent to the microprogram control store from the microprogram sequencer. After the PROM access delay, the next micro instruction arrives at the input of the pipeline register and the microprogram sequencer. When the clock pulse now arrives, the execution of the next micro instruction is performed and the address for the next micro instruction will be sent to the microprogram control store.

The pipeline register allows the address to the microprogram control store to be changed, while the current microinstruction is being executed. In this way a microinstruction fetched in the PROM in one microinstruction cycle is executed in the next micro cycle. While one micro instruction is executed, the next micro-instruction is being read from the microprogram control store. The speed of the computer is improved because the data operation occurs simultaneously with the next access in the microprogram control store.

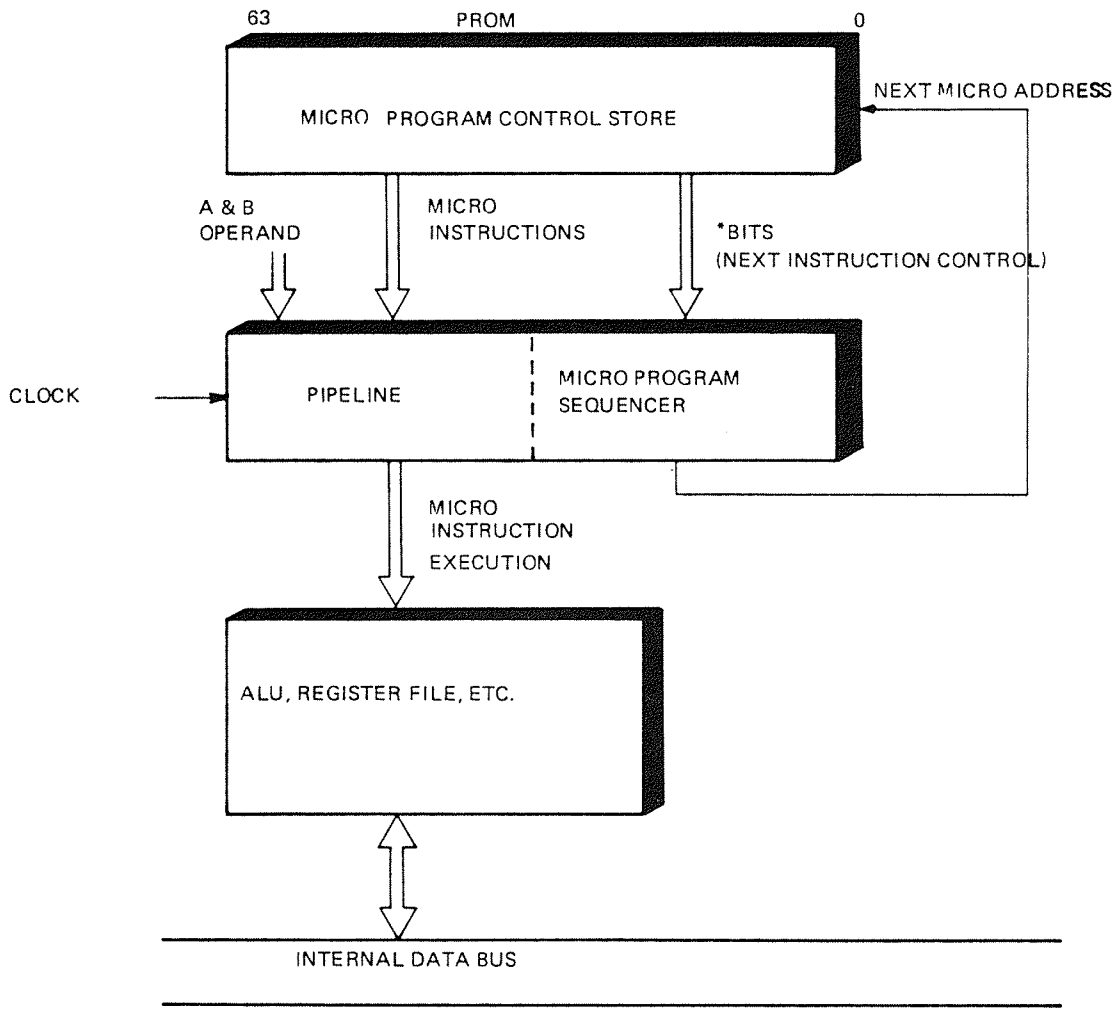


Figure II.7.1: Pipeline Block Diagram

Figure II.7.2 shows a timing diagram for the events in the pipeline blocks.

The diagram with "PROM VALID" for the previous microinstruction. This means that the output of the microprogram control store is valid (the access time is finished) and, with the low-to-high transition of the clock, the pipeline and the micro address for the microprogram control store will be clocked (pipeline valid, micro address valid).

Pipeline valid allows the data manipulation bits to go to the data hardware for execution of the previous microinstruction. This operation ends up with a result on the internal data bus, shown as "IDB valid". While this happens, a new access in the microprogram control store has already been made, which ends up with "PROM VALID". Now the next microinstruction has arrived at the input of the pipeline register and the microprogram sequencer, and will be executed on the rising edge of the next clock pulse. The sequence of the executing micro-instructions will continue in this way.

A clock cycle is the time interval between two successive pipeline register load signals. This is the time available for data manipulating elements to perform their functions.

Without the pipeline register, the blocks in the figure would be placed sequencially one after another. This would make the CPU significantly slower, and the access time of the PROM would be important in determining the speed. Presently, the PROM can be rather slow, because most time-consuming operations are controlled from the pipeline.

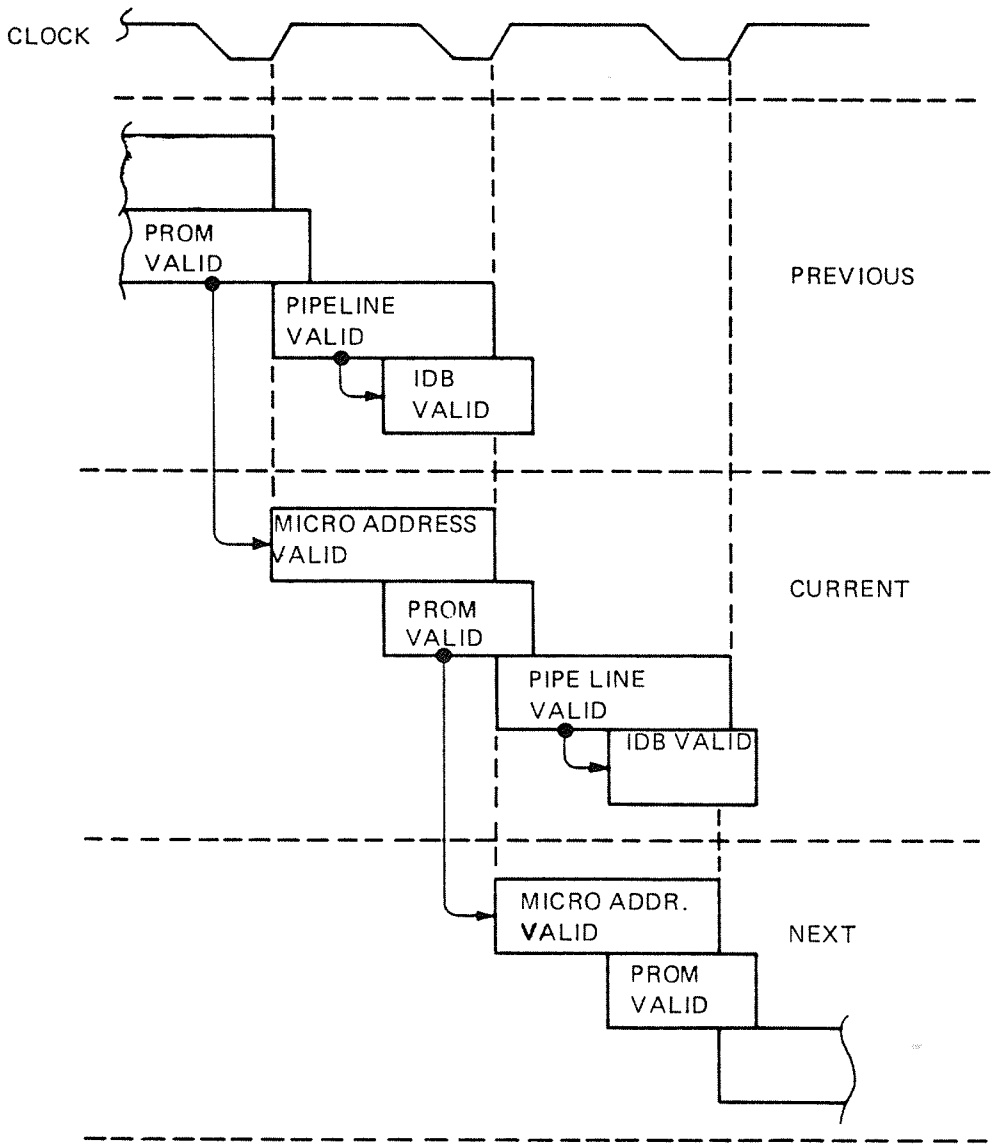


Figure II.7.2: Pipeline Blocks Timing Diagram

II.8 THE ARITHMETIC LOGIC UNIT

II.8.1 *GENERAL*

The arithmetic logic unit (ALU) is that part of the processor that computes. Under control of the microprogram, the ALU performs a number of different arithmetic, logic and manipulative operations on data in the working registers or from the internal data bus.

Figure II.8.1 shows the position of the ALU in the system. The control lines from the pipeline register go in as instructions, controlling the ALU operation, and as shift linkage control, controlling the shift operations of the ALU with the right in/left out and right out/left in lines.

A operand select and B operand select selects two operands in the working register block inside the ALU, to be operated on. An operand can also be taken from the IDB.

The result of the arithmetic logic operation may be stored in one of the working registers inside the ALU or enabled onto the IDB. Any flags, such as overflow, carry, etc. are reported to the status register, together with flags from the shift linkage circuitry during shift operations.

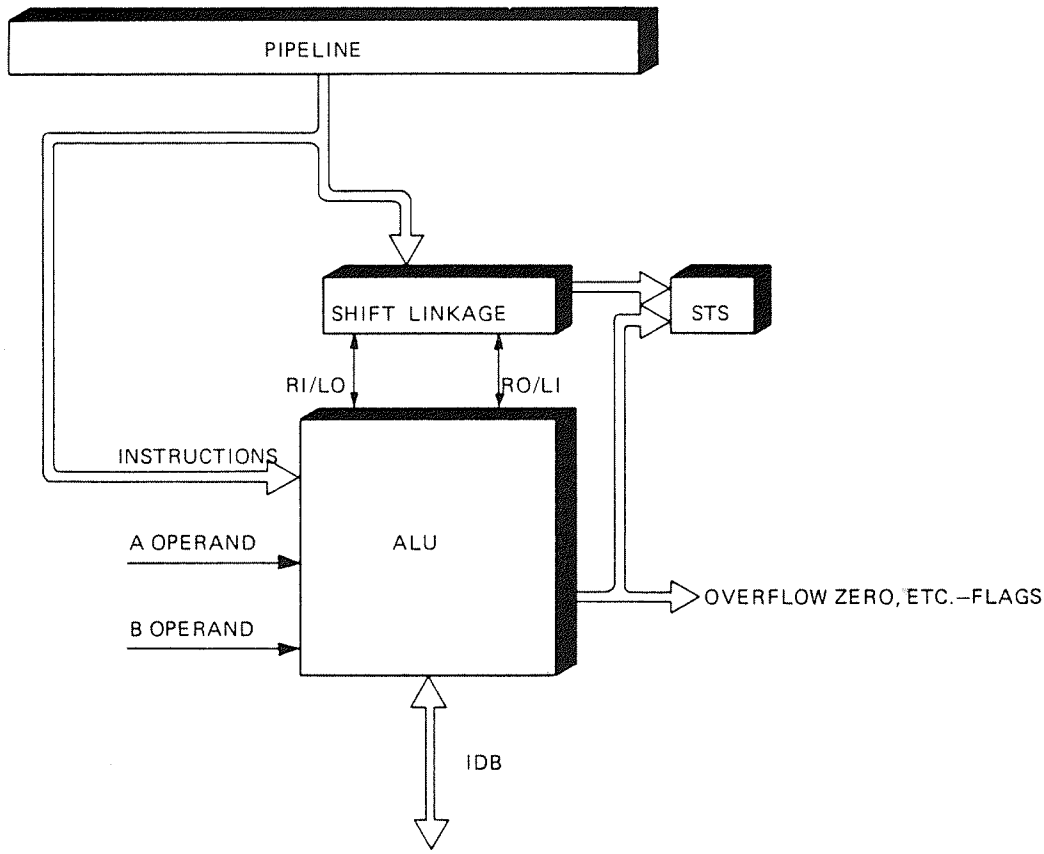


Figure II.8.1: The Arithmetic Logic Unit

ND-06.015.01

II.8.2 *THE ALU BIT SLICE*

Figure II.8.2 shows the register file and main arithmetic.

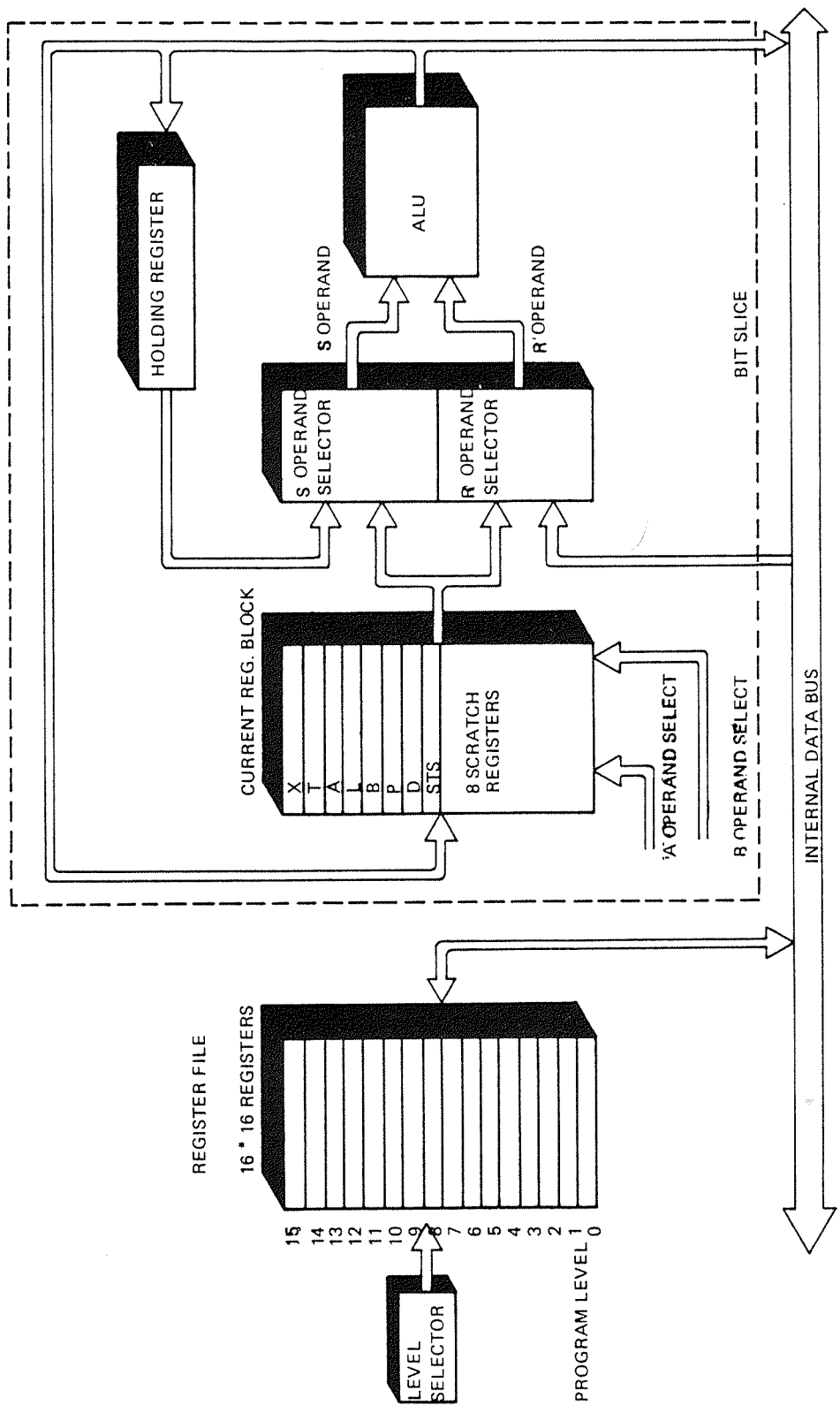


Figure II.8.2: Register File and Main Arithmetic

ND-06.015.01

The main part of the arithmetic unit consists of the parts called the bit slices.

A bit slice is an integrated circuit (chip) containing a 4 bit subsection of the 16 bit wide ALU and register section. These complex circuits must be programmed to execute the system functions. In a microprogrammed processor system such as the NORD-100, bit slices are driven by a set of control lines (bits) from a micro-instruction. This gives the possibility of changing or modifying the microprogram while keeping the existing hardware.

The 4 ALU bit slices in the NORD-100 consist of one 16 bit register block with the 8 working registers on the current level plus 8 scratch registers. The A source select and the B source select, selects one 16 bit register each from the current register block. The selected values to the R operand selector and the S operand selector, are passed. Other inputs to these selectors are data directly from the internal data bus and from a holding register. Which of these inputs that are to be selected as the R and S operands is controlled from the microprogram directly.

The selected R and S operands enter the ALU. The ALU can perform several functions controlled from the microprogram. The result of the operation in the ALU can be placed in the holding register, enabled onto the IDB or written back to one of the registers in the current register block, depending on control lines from the microprogram.

When changing from one level to another, the working registers (X, T, A, L, B, P, D, STS) in the current register set will be written into the current level registers in the register file. The working registers on the new level will be copied from the register file into the registers of the bit slices. The eight scratch registers contain temporary information, managed by the microprogram, like addresses during memory reference instructions, temporary results during floating point operations, etc. These scratch registers will not be saved under a level change.

The holding register can be used to keep results from ALU operations. This register can be shifted right or left linked to any of the working registers for double shifts, etc.

The ALU is controlled by the microprogram, through the lines selecting the A and B operands, the kind of operation to be performed in the ALU and where to place the result. These lines are called instructions in Figure II.8.1. In one clock cycle, two operands can be supplied to and manipulated by the ALU, with one result placed either in the holding register or the current register block, or sent to the IDB. The ALU also provides a set of condition codes as a result of the arithmetic/logic operation. The condition codes (overflow, etc.), together with other computer status information, are stored in the status register.

Example 1:

The LDA instruction:

The operand to be loaded into the A register is enabled from the DBR onto the IDB. The microprogram will select the IDB input as R operand and pass it unmodified through the ALU. Destination will be selected to be the current register block, the register number will then be given by the B source input, in this case 5. The operand will therefore be loaded into the A register. A source select and B source select are used when reading from the register file but only the B source select is used when writing to the current register set.

Example 2:

The RADD SA DB instruction:

A source select reads out the A register contents from the current register block and the B source select reads the B register contents. The microprogram selects the A register value as R operand and the B register value as S operand. ALU function is selected to be R + S, and the destination for the result of the operation to be the current register block. The register in the current register set to be written into is given by the B source select, which is equal to the B register.

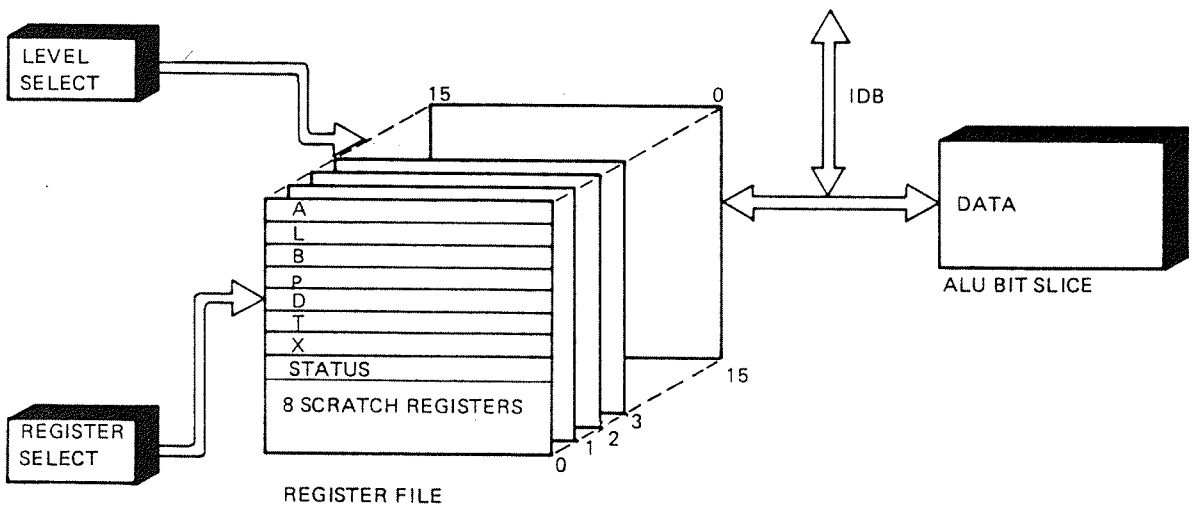


Figure II.8.3: ALU and Register File Connection

Figure II.8.3 shows how the communication between the register file and the ALU is performed over the internal data bus. Any register on any level, specified by the level select and the register select, may be enabled onto the IDB. It may then be selected as R operand and operated on in the ALU bit slice. Saving and unsaving of the working registers in the ALU are performed this way during a level change.

II.9 THE INTERRUPT SYSTEM

II.9.1 *GENERAL*

One CPU can handle many simultaneous processes but only one process can be actively involved at a time. These processes are almost without exception asynchronous such as input/output device service requests, external timer signals, program errors and power failure.

To have automatic response to the different conditions outside the CPU or in the processor itself, it is important to have an efficient synchronization system handling these asynchronous events, an "asynchronous event handler".

II.9.1.1 *Polling*

The simplest approach to asynchronous event handling is the *poll* approach.

A status indicator is associated with each possible asynchronous event. The processor tests each indicator in sequence and, in effect, "asks" if service is required. This program-driven method is inefficient for a number of reasons. Much time is consumed polling when no service is required; programs must have frequent test points to poll indicators, and since indicators are polled in sequence, considerable time may elapse before the processor responds to an event.

For a system with many I/O devices with high transfer rate, the polling system is CPU time consuming, i.e., a great part of the CPU time is spent in the polling routines.

II.9.1.2 *Interrupts*

The interrupt method is a much more efficient way of servicing asynchronous requests. An asynchronous event requiring service generates an interrupt request signal to the processor. When the processor receives the interrupt request, it may suspend the program it is currently executing, execute an interrupt service routine which services the asynchronous request, then resume the execution of the suspended program. In this system, the execution of the service routine is initiated by an interrupt request; thus, the system is interrupt driven and service routines are executed only when service is requested. Although hardware cost may be higher in this type of system, it is more efficient since system throughput is higher and response time is faster.

There are several kinds of interrupt systems:

— Single line interrupt system

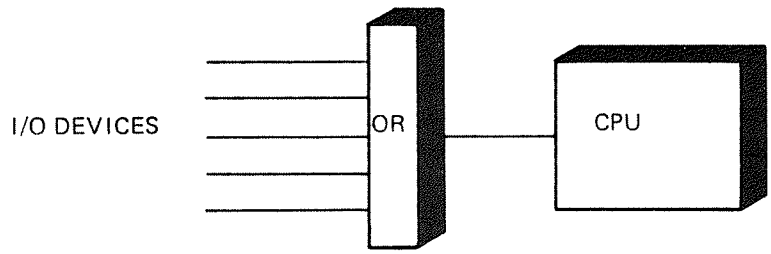


Figure II.9.1: Single Line Interrupt System

In the single line interrupt system all I/O devices are ORed together through a single interrupt line.

Once the interrupt is received, all of the I/O devices are polled to determine which one caused the interrupt.

Advantage: Simple interrupt system

Disadvantage: Slow interrupt identification, system overhead

— Multilevel interrupt system

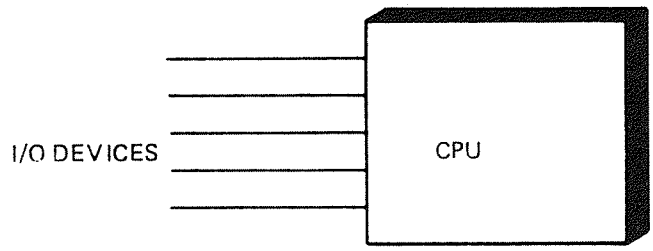


Figure II.9.2: Multilevel Interrupt System

In the multilevel interrupt system, there are several interrupt lines. Each I/O device has its own line, thus no polling is required by the CPU. Different priorities are normally assigned to the levels.

Advantage: Fast interrupt identification, no system overhead

Disadvantage: Non-flexible with respect to expansion

— Vectored interrupt system

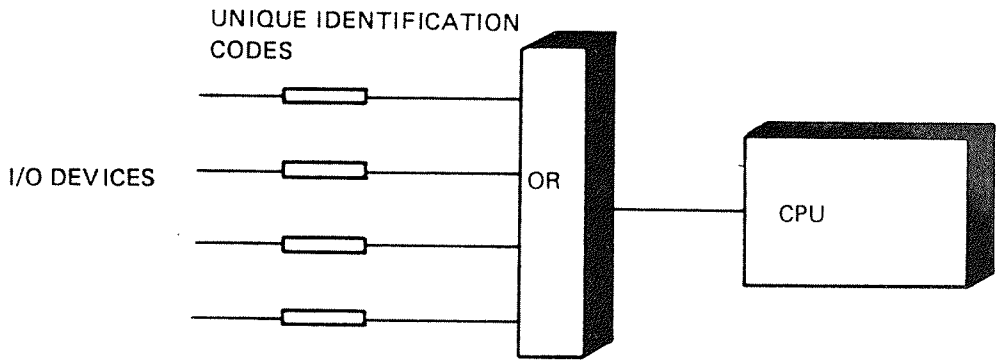


Figure II.9.3: Vectored Interrupt System

In this system there is only one interrupt line. However, associated with the interrupt a code will be issued from the interrupting device. This code will identify the device.

Advantage: Flexible system, may easily be expanded

Disadvantage: Some system overhead

II.9.2 *NORD-100 INTERRUPT SYSTEM*

II.9.2.1 *General*

The NORD-100 interrupt system is designed to simplify programming and to allow high efficiency interrupt handling. To provide this, the NORD-100 interrupt system is a combination of the multilevel and the vectored interrupt system.

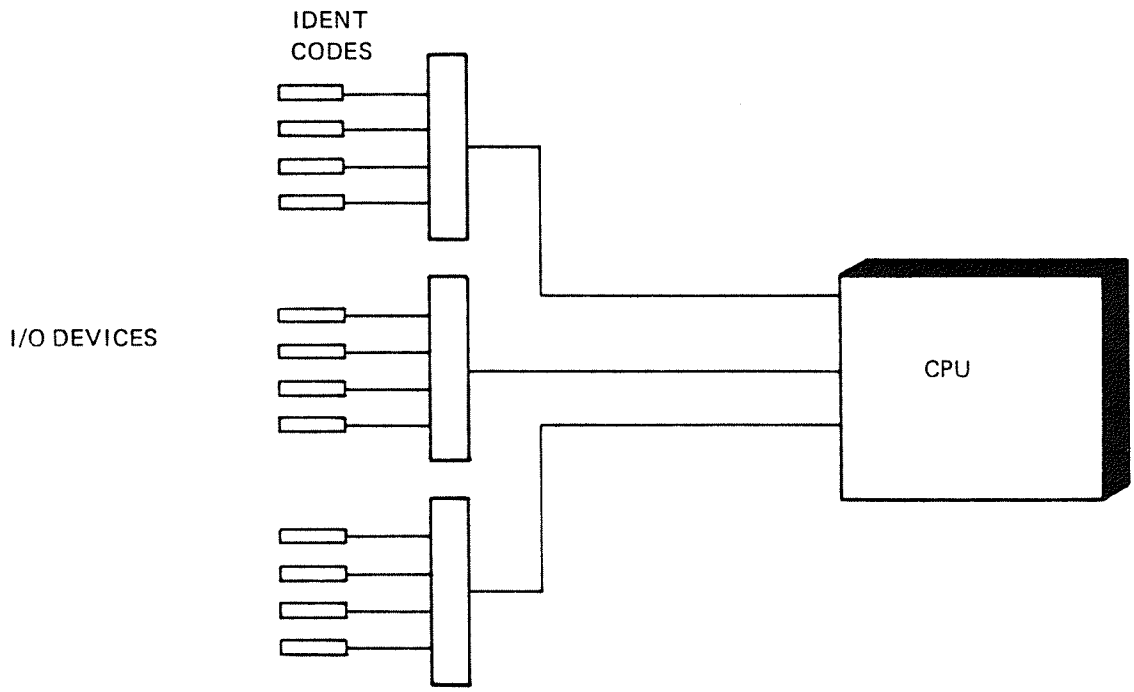


Figure II.9.4: NORD-100 Interrupt System

This will form a fast and flexible interrupt system with the following features:

— Multiple Interrupt Request Handling

  Since interrupt requests are generated from a number of different sources, the interrupt system's ability to handle interrupt requests from several sources is important.

— Interrupt Request Priorities

  Since the processor can service only one interrupt at a time, it is important that the interrupt system has the ability to assign priorities to the requests and determine which has the highest priority.

— Interrupt Service Routine "Nesting"

This feature allows an interrupt service routine for a given priority request to be interrupted in turn, but only by a higher priority interrupt request. The service routine for the higher priority request is executed, after which the execution of the interrupted service routine is resumed.

— Dynamic Interrupt Enabling/Disabling

The ability to enable/disable dynamically during microprogram or a macroprogram (ION/IOF) control can be used to prevent interruption of certain processes.

— Dynamic Interrupt Request Masking

The ability to selectively inhibit or "mask" individual interrupt requests under microprogram control is useful.

— Interrupt Request Vectoring

Often a particular interrupt request requires the execution of a unique interrupt service routine. For this reason, the generation of a unique binary coded vector for each interrupt request is very helpful. This vector can be used as a pointer to the start of a unique service routine.

— Fast Interrupt System Response Time

Quick interrupt system response provides more efficient system operation. Fast response reduces real-time overhead and increases overall system throughput.

There are 16 program levels in NORD-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 a status register with the status indicators O, Q, Z, C, M, K, PTM and TG. There are also 9 registers that are only accessible from the 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 when running the SINTRAN III operating system.

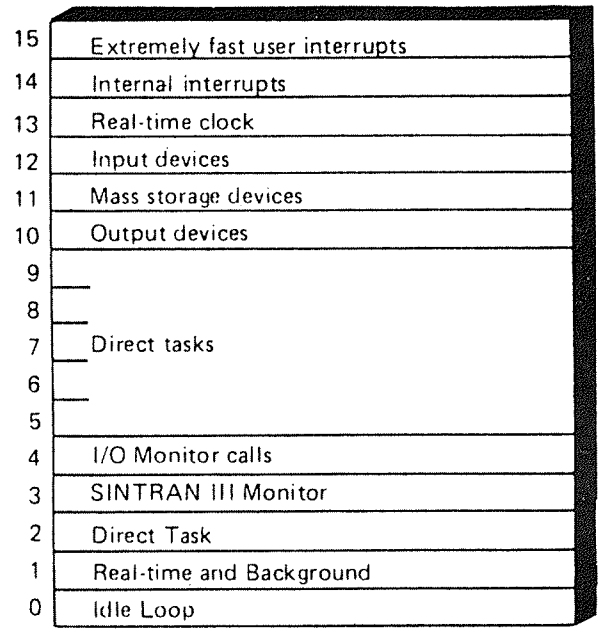


Figure II.9.5: Level Assignments

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. The result of the IDENT instruction is a unique identification code in the A register upon completion of the IDENT instruction.

Program level 15 may only have one I/O interrupt source. It is not used by standard NORD equipment or software, but is available for users who need an immediate access to the CPU.

Program level 14 is used by the internal interrupt system, which monitors error conditions or traps in the CPU.

The "real-time clock", the multiport memory error log and HDLC input are connected to level 13.

Character input devices like teletypes, tape readers, etc. are connected to level 12, and also HDLC output.

To level 11 are connected mass storage devices like disk, floppy disk, mag. tape, etc.

Level 10 is assigned to character output devices like line printers, paper tape punches, displays, etc.

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.

Figure II.9.6 shows the interrupt system part of the CPU.

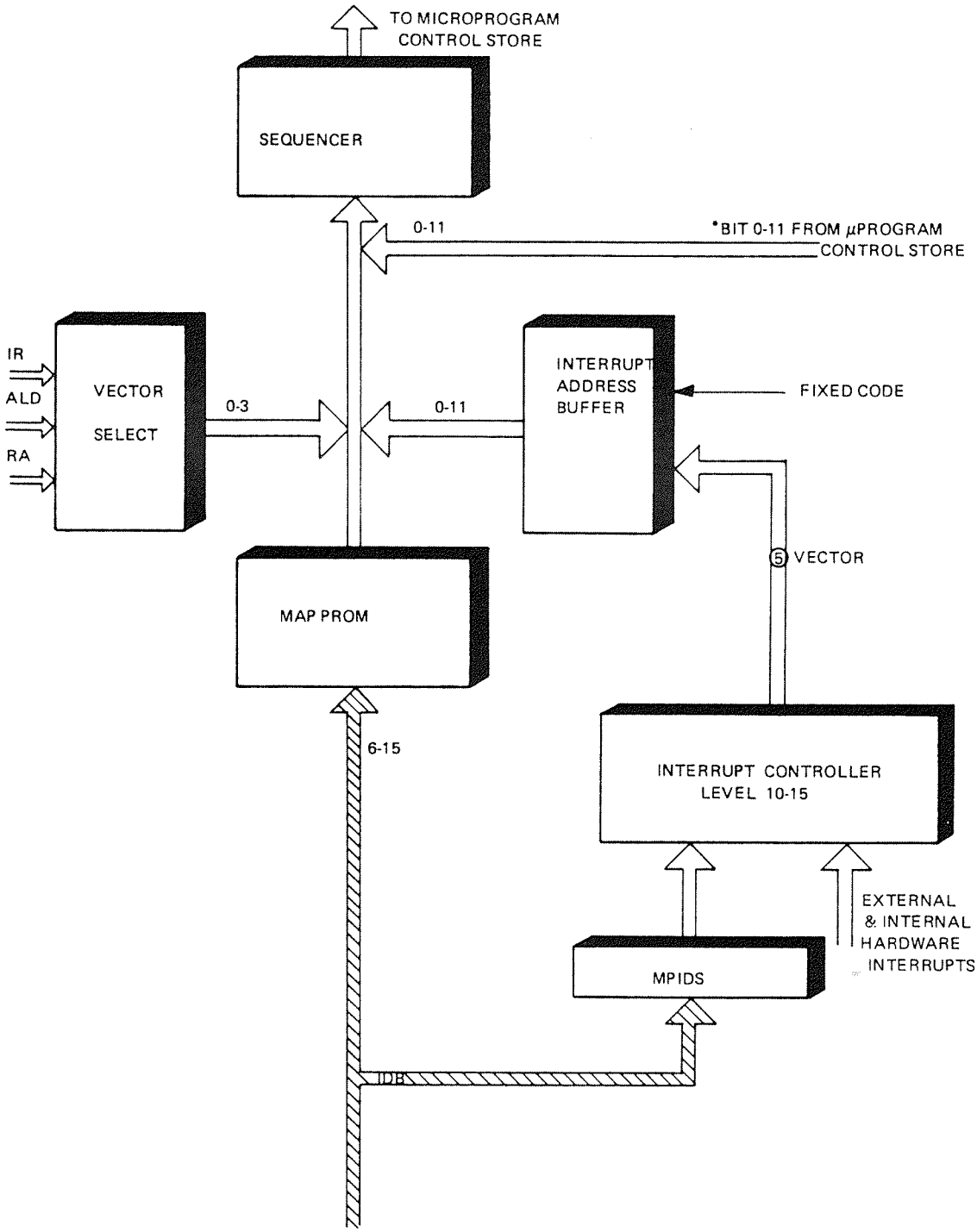


Figure II.9.6: The Interrupt System Part of the CPU

The interrupt controller takes care of all interrupts on levels 10-15, including all internal interrupts. Interrupts on levels 0-9 are implemented in firmware, which means that they are managed by the microprogram. Both the external and the internal interrupts are hardware signals, which occur when a device wishes to indicate to the program or the CPU that an interrupt condition has occurred. For the external interrupt system this may be things such as disk transfer completed, device ready for transfer, etc. The internal interrupts system reports things like memory out of range, power failure, etc.

Based upon these interrupts, the interrupt controller gives out a 5 bit vector specifying the interrupt. These bits, together with a fixed code goes into the interrupt address buffer. When this is enabled, it will give a branch address to the microprogram control store, via the microprogram sequencer, to start the interrupt microprogram.

MPIDS is a buffer, which the microprogram uses to generate interrupts on level 10-15 including all internal interrupts.

II.9.2.2 *Functional Operation*

Figure II.9.7 shows the functional operation for the complete priority interrupt system.

There is one bit for each level (10-15) 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 is implemented in firmware which means that the microprogram takes care of the detection of interrupts on these levels by setting the proper bits in this register.

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, the detect register and the mask register are ANDed together and the priority encoder gives a level value corresponding to the highest bit set in both 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 interrupt system allows the processor to continually compare its own priority level with the level of any interrupting devices and to acknowledge the device with the highest level above the current priority level. Servicing an interrupt for a device can be interrupted for servicing a higher priority device. Service to the lower priority device is resumed automatically upon completion of the higher level servicing.

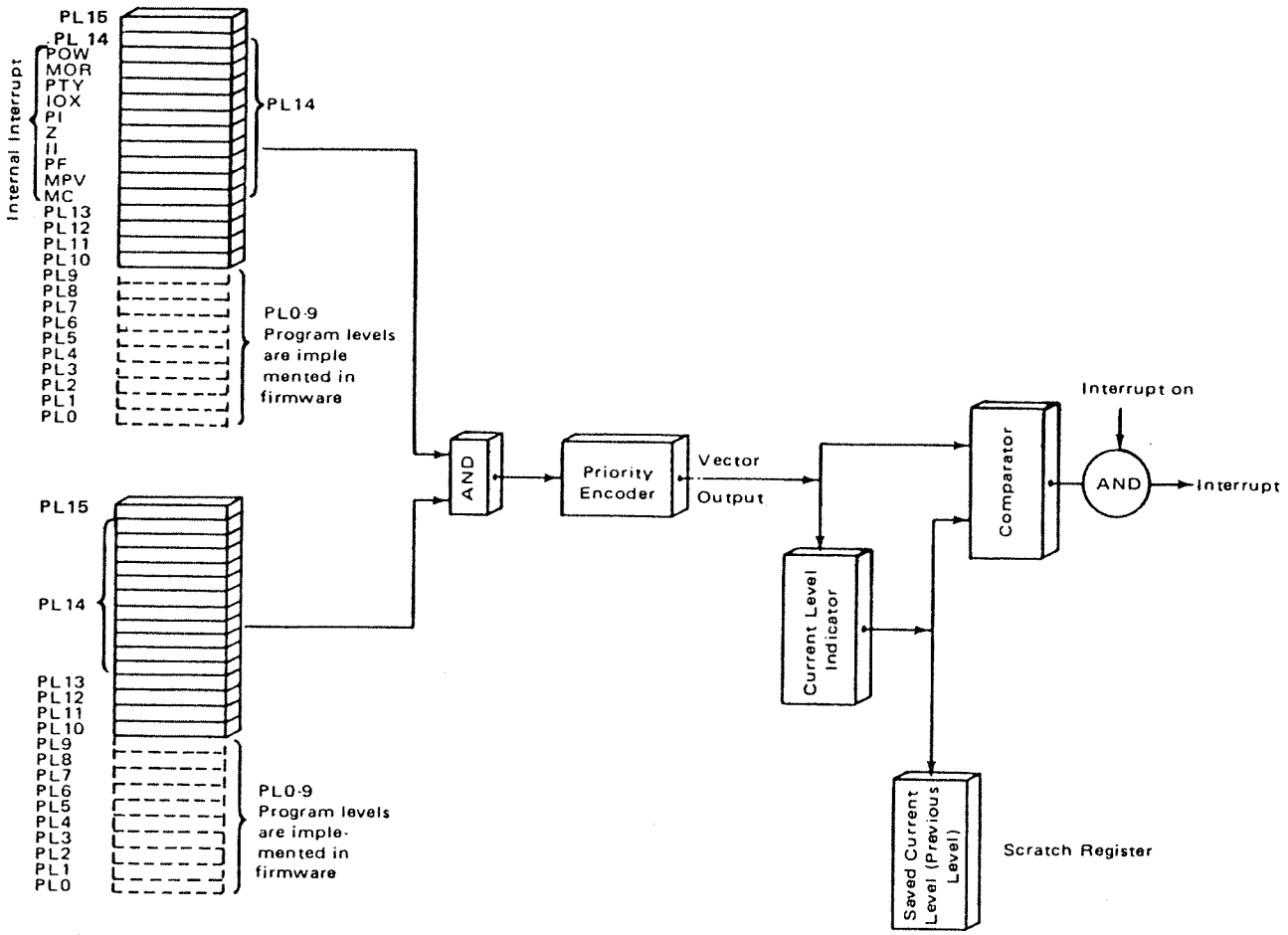


Figure II.9.7: Priority Interrupt System

II.9.2.3 *The External Interrupt System*

Figure II.9.8 gives a block diagram presentation of the external interrupt system.

The program level to run is controlled from the two 16 bits 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 PL (0-15). The 4 bit PIL register controls which register block to use.

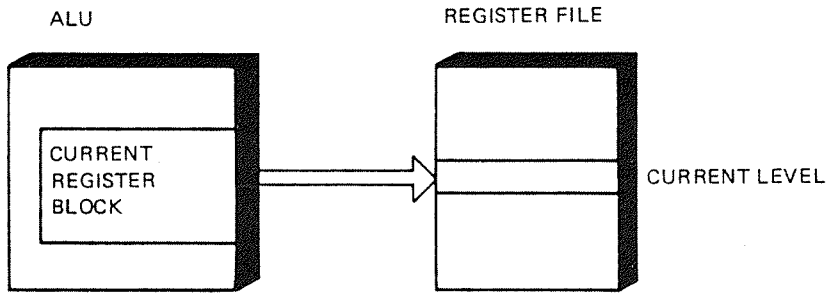
The PIL number is constantly compared to PK. PK always contains the number of the highest program level which has its corresponding bits set in both PIE and PID. Whenever PK is unlike PIL, an automatic change of context block will take place through a microprogram sequence.

The CPU will not ask for the next machine instruction but enter a microprogram that will change the program level to the one in PK.

The level change can be illustrated as follows:

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

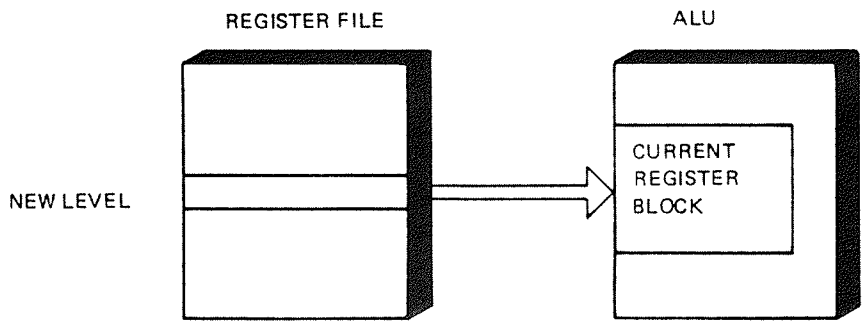
2. The working register block on the current level in the ALU is saved in the register file.



3. The PIL (program level) on the old level 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 level change takes place at this time.

5. The working register block on the new level in the register file is moved to the current register block in the ALU.



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 register 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.

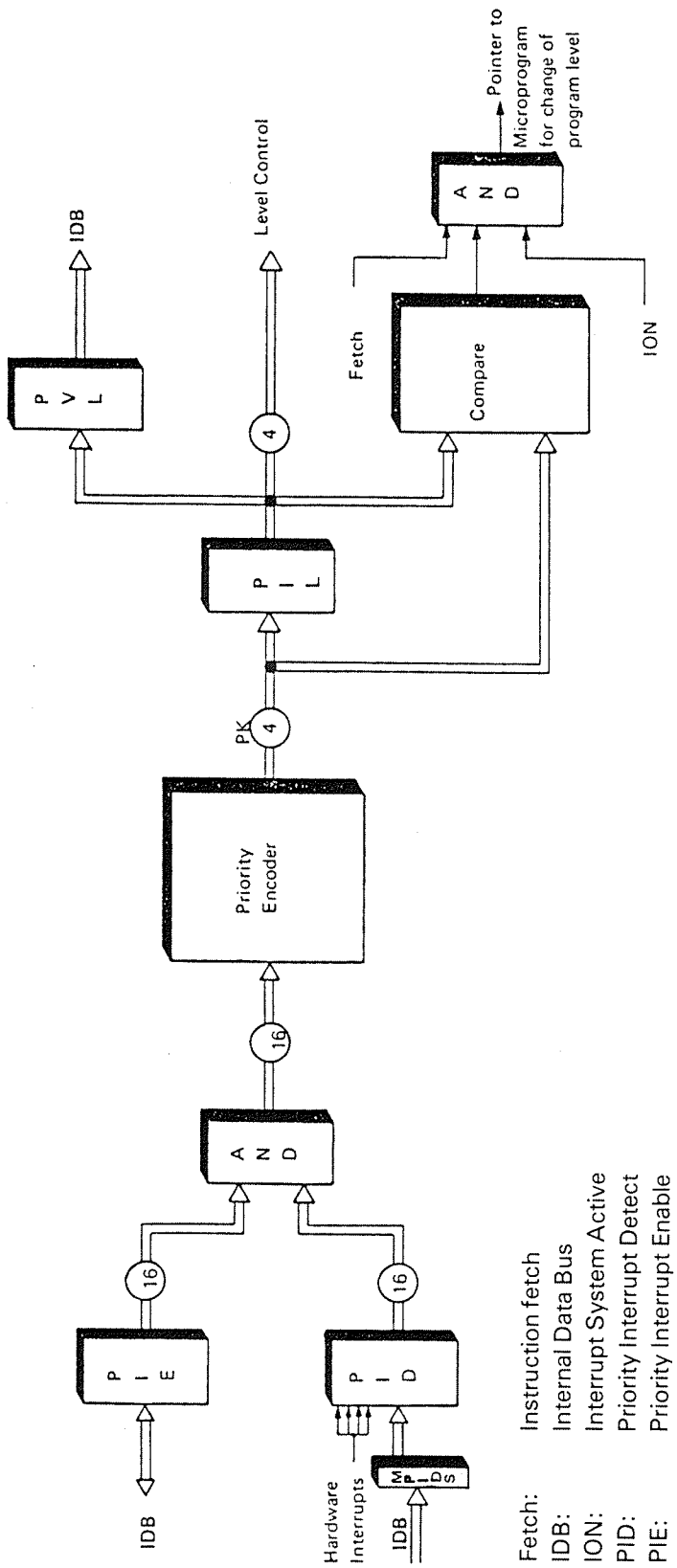


Figure II.9.8: External Interrupt System

II.9.2.3.1 External Interrupt Identification

Since a vectored interrupt system is used, more than one device can use the same interrupt line. The vector or ident code is found by using an IDENT instruction to identify the interrupt. The instruction has the following format:

IDENT<program level>

In NORD-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

When an IDENT instruction is executed, a hardware search on the indicated level is performed. The first interrupting device found will respond with its 9 bit identification code and reset its interrupt condition. The identification code will be received in the A register. The CPU will use this code (vector) to find the driver for the interrupting device. 9 bits gives 512 different vectors on each of the four actual levels, giving up to 2048 different vectors altogether.

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           % clear A register
            IDENT   PL13        % Identify device on level 13
            RADD    SA  DP      % Computed GO TO - add A reg. to
                                % P reg (PC)
            JMP     ERR13       % Code 0, error
            JMP     DRIV1       % Code 1
            JMP     DRIV2       % Code 2
            —
            —
            JMP     DRIVN       % Code N
```

II.9.2.4 *The Internal Interrupt System*

The internal interrupt system is also a vectored system.

The functional operation of the internal interrupt system is basically the same as the external one. Refer to Figure II.9.9.

As previously mentioned, the internal interrupt system is connected to level 14. It is controlled from the two registers:

IIE: Internal Interrupt Enable
IID: Internal Interrupt Detect

IIE is controlled by program only, i.e., the various internal interrupts are enabled/disabled by a program setting/clearing the corresponding bits in the IIE register. The internal hardware status interrupts are assigned to the IIE register in the following way:

15 … 10	9	8	7	6	5	4	3	2	1	0
NOT ASSIGNED	POW	MOR	PTY	IOX	PI	Z	II	PF	MPV	MC	NA

An internal hardware signal will set one of the bits in the IID register. IIE and IID are ANDed together and go into the priority encoder which gives a 4 bit code, the internal interrupt code (IIC). This code has a value between $0\text{-}12_8$, which will identify the internal interrupt condition which forced the CPU to level 14. The operating system will then read the IIC register to find the reason for the interrupt. Bit no. 14 in the PID register is also set to one. When the internal interrupt code is ready the IID register bit is reset.

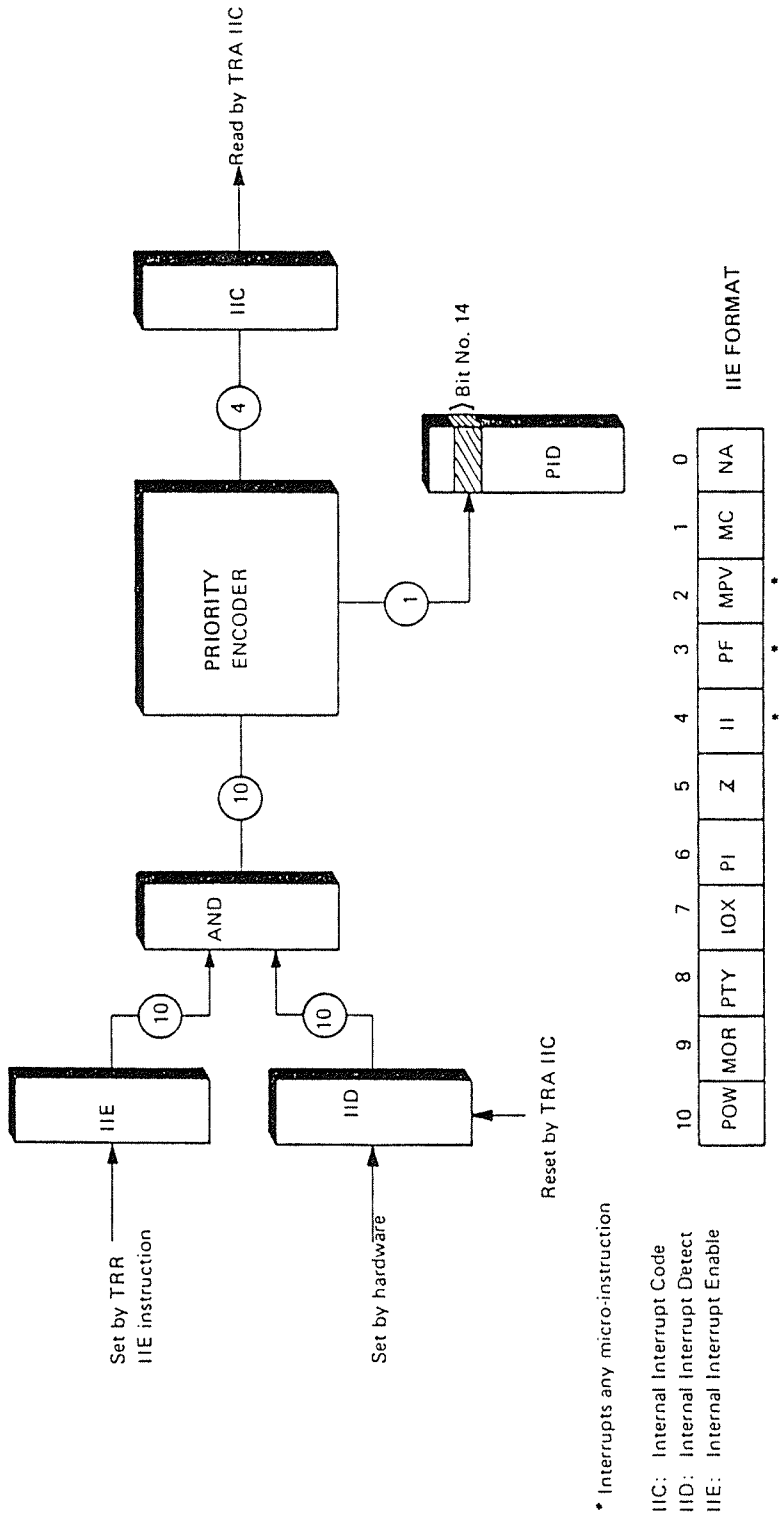


Figure II.9.9 Internal Interrupt System, Block Diagram

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

	Bit No.: (Decimal)	IIC Code: (Octal)	Cause:
NA	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

MPV, PF and II will interrupt the microprogram, i.e., within a machine instruction. PI, IOX, MOR, MC, Z, PTY and POW will not give an internal interrupt until the current machine instruction has been completed.

If PF or PV occur during a fetch (prefetch) cycle, the PC (program counter) is not incremented. In all other cases, PC points to the next machine instruction.

There is not priority assigned to the different internal interrupts as only one condition may arise at any time (except if power fail occurs, in which case POW has the highest priority).

The PIE bit number 14 has to be set in order to enable the internal interrupts and the appropriate bit mask must be set up in IIE — Internal Interrupt Enable.

The interrupt system must be turned on by the ION instruction in order to receive an external or internal interrupt.

II.9.2.4.1 Internal Hardware Status Interrupts

Monitor Call Interrupt

One of the internal interrupt sources is the monitor call instruction 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 extension (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.

The paging system has to be turned on (PON) to receive this interrupt.

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.

The paging system has to be turned on (PON) to receive this interrupt.

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 conditions:

- FDV with 0.0
- EXR of an EXR instruction
- DNZ overflow
- RDIV overflow
- programmed setting of Z (BSET, MST or TRR)

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

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.

The paging system has to be turned on (PON) to receive this interrupt.

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 (TRA PEA).

Further information may be read from the PES register.

Memory Out of Range Interrupt

This interrupt occurs when the program addresses non-existing 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.

Reset of IIC

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.)

II.9.2.4.2. Internal Interrupt Identification

An internal interrupt will force the CPU to level 14_{10}. The IIC (Internal Interrupt Code) register will hold a code (vector) indicating the source for the interrupt. The register will be locked to prevent overwriting.

After executing a

```
TRA IIC
```

the IIC register is cleared and the A register holds the IIC code. A branch to the proper internal interrupt routine can then be made.

Example:

The IIC code may be analyzed by the following routine.

```
LEV14,      TRA     IIC     % Place IIC code in A reg. and reset
                            % error lock
            RADD    SA  DP  % computed go to - add A reg. to
                            % P. reg. (PC)
            JMP     ERROR   % 0, not assigned
            JMP     MONCL   % 1, monitor call
            JMP     PROTN   % 2, protection violation
            JMP     PAGEF   % 3, page fault
            —
            —
            —
            JMP     POW     % 10, power failure
            —
            —
            —
MONCL,      Execute a
            monitor call
            —
            —
            —
EXIT        WAIT
            JMP     LEV14
```

PC will point to the instruction after WAIT when LEV14 is reentered.

II.9.2.5 *Programming Control of the Interrupt System*

II.9.2.5.1 Control of the Interrupt System

When power is turned on, the power up sequence will reset PIE and the register block 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

Format: ION

The ION instruction turns on the interrupt system. At the time the ION is executed, the computer will resume operation at the program level with highest priority. If a condition for change of program levels exists, the ION instruction will be the last instruction executed at the old program level, and the 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

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 PIL register will remain unchanged. The interrupt indicator on the operator's display is reset by the IOF instructions. The IOF instruction is privileged.

II.9.2.5.2 Programming the Interrupt Registers

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.

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, called programmed interrupts.

Examples of Programmed Interrupts:

1. Changing to a higher level

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

```
        SAA 0
        BSET ONE 110 DA     % Set bit 9 to 1 (11₈ = 9₁₀)
        MST PID             % Set PID bit 9
NEXT,   ...
```

2. Changing to a lower level

   Assume that the CPU is currently running on level 10_{10} and one wishes to continue on level 5.

```
        SAA 0               % clear A reg.
        BSET ONE 50 DA      % set bit 5 to 1
        MST PID             % set PID bit 5 to one
        WAIT                % give up priority
NEXT,   ...
        ...
```

Enabling of the desired internal interrupt sources by proper mask setting of the IIE register is done by the TRR IIE.

II.9.2.5.3 Leaving the Interrupting Level

When completing an interrupt routine on a higher level, the CPU should continue on the highest level holding an active interrupt (indicated by the highest bit in the PID register).

Leaving the level, also referred to as giving up priority, is performed by executing a WAIT instruction.

The WAIT will cause an exit from the program level now operating, the corresponding bit in PID is reset, and the program level with the highest priority will be entered, which normally will then have a lower priority than the program level which executes the wait instruction. Therefore, the WAIT instruction means "give up priority".

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

Note: The saved program counter will point to the instruction *after* the WAIT instruction.

II.9.2.5.4 Use of the PVL Register

When an internal interrupt occurs, the program counter on the offending level has been incremented and points to the instruction after the one that caused the interrupt.

In some cases after being forced to level 14_{10}, the CPU (i.e., operating system) wants to know which level was the last one and the value of the program counter on that interrupting level. This may be useful for a number of purposes. This is done by help of the TRA PVL instruction, which has a rather unusual format. This instruction will read the contents of the PVL register (4 bits) into the A register in bits 3-6. At the same time, the micro program will set bits 7-15 in the A register to a bit pattern which gives the operation code for the IRR instruction (inter register read). Bits 0-2 in the A register are set to DP (destination P register). Now the A register will hold the instruction code for inter register read, with level specified as the interrupting level and the register to read specified as the P register:

IRR <previous level * 10_8> DP

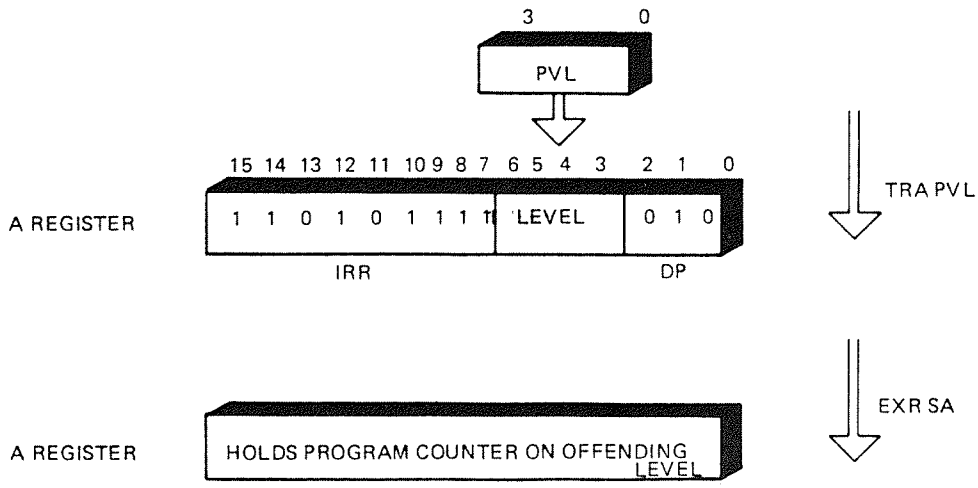


Figure II.9.10: TRA PVL Instruction Execution

By then executing an EXR SA (execute register) the content of the A register will be executed as an instruction. After this instruction is executed, the program counter on the level which caused the interrupt will be found in the A register.

Note that there are some cases where the program counter has not been incremented, for example if a memory protect violation interrupt occurs. If this interrupt occurs during the fetch of an instruction, the program counter is not incremented, but if it occurs during the data cycle of an instruction, the program counter is incremented (refer to the memory management system).

II.9.2.6 *Initializing of the Interrupt System*

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

1. Enabling 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 saved program counters, on the level 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 must be 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 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   saved PC in use
TRA             IIC             % Unlock IIC
TRA             PEA             % Unlock PEA and PES
ION                             % Turn on interrupt system
JMP             START           % Go to main program
```

II.10 THE ADDRESS ARITHMETIC

II.10.1 *GENERAL*

In order to communicate with memory, an address must be formed. This is the task of the address arithmetic. The address arithmetic is implemented in firmware, which gives the possibility of changing the structure by rewriting the micro-program.

Three types of information are stored in memory:

1. Data (operands or results)
2. Instructions (program elements)
3. Indirect addresses

Memory, however, regards all as information. The CPU must then, to tell the difference, place itself in one of four modes when communicating with memory. Those modes are:

1. Fetch — fetch next instruction
2. ROP — read operand
3. RADDR — read indirect address
4. STO — store operand

This can be illustrated in the following way:

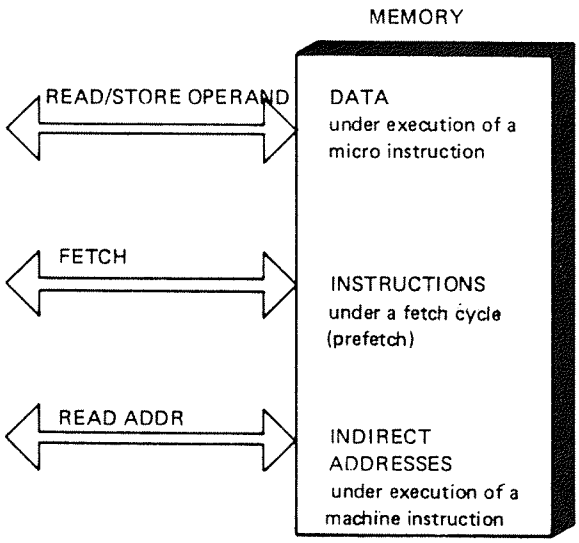


Figure II.10.1: Different Memory Accesses

II.10.2 *ADDRESSING STRUCTURE*

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

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.)

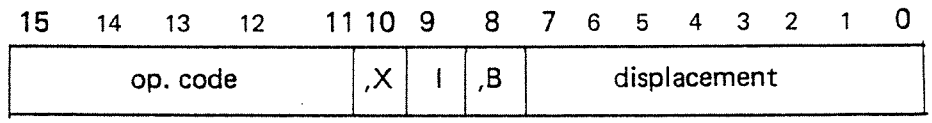


Figure II.10.2: Format of the Memory Reference Instructions

NORD-100 uses a relative addressing system, which means that the address is specified relative to the contents of the program counter or relative to the contents of the B and/or X registers.

The three addressing mode bits called ",X", "I" 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 takes 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 modification (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

The different addressing modes will now be examined.

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

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:

STA * +2

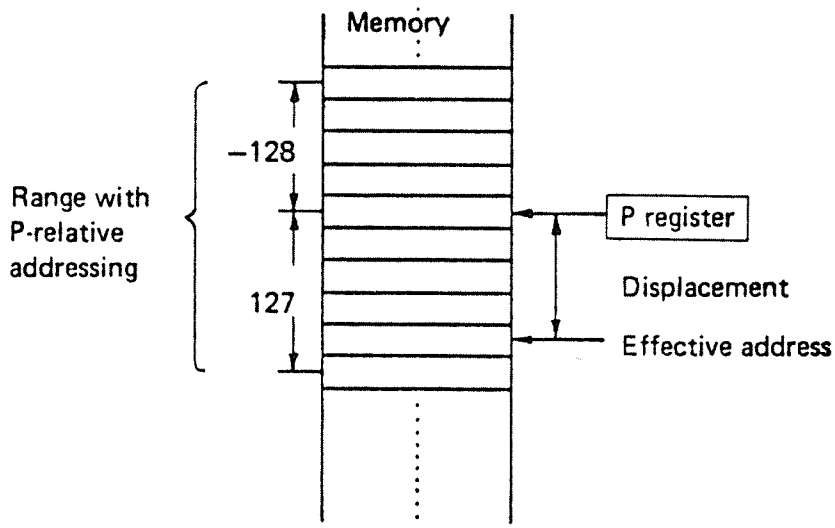


Figure II.10.3: Schematic Illustration of P Relative Addressing

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.

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 memory location which is accessed instead.

Example:

STA I * +2

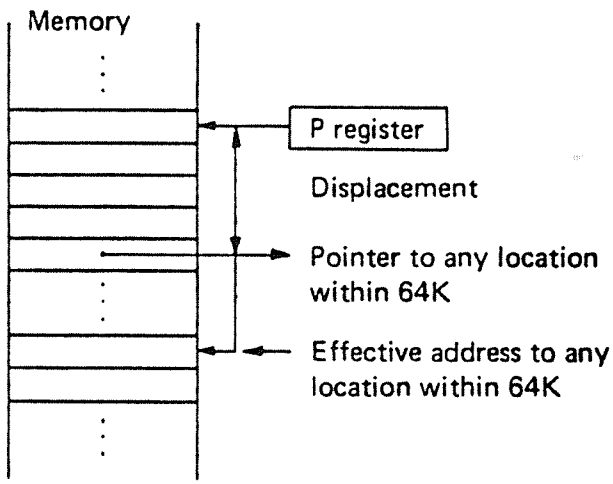


Figure II.10.4: Schematic Illustration of Indirect P Relative Addressing

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

This mode is very useful when two subprograms want to address a common data area directly for internal communication. This addressing mode is related to P relative addressing, but the displacement is added to the current value of the B register instead and the resulting sum is used to specify the memory location accessed.

Example:

STA * +2,B

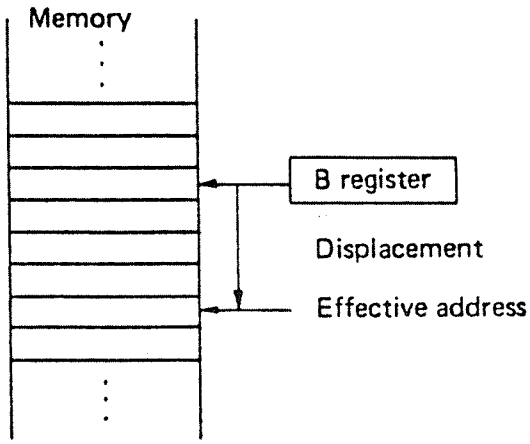


Figure II.10.5: Schematic Illustration of B Relative Addressing

Indirect B relative addressing (,X = 0 I = 1 ,B = 1)

This mode has the same relationship to B relative addressing that 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:

STA I * +2,B

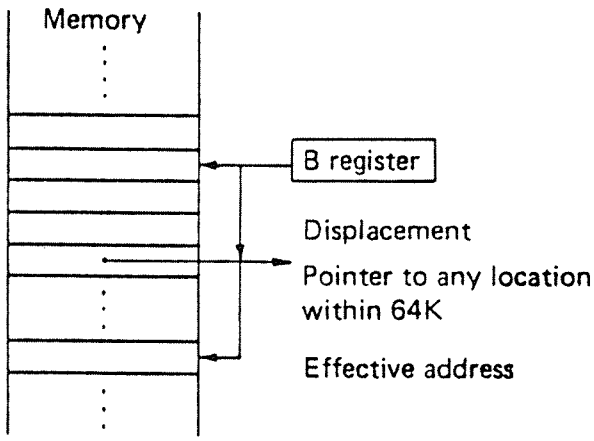


Figure II.10.6: Schematic Illustration of Indirect B Relative Addressing

X relative (or indexed) addressing (,X = 1 I = 0 ,B = 0)

Here the displacement is added to the X register 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:

STA * +2,X

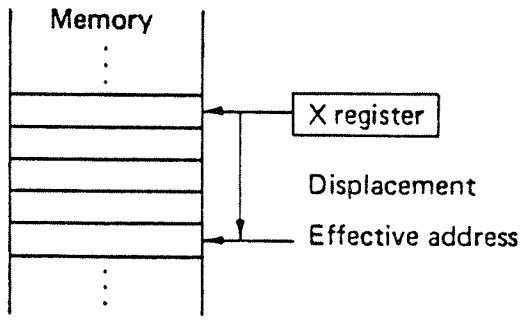


Figure II.10.7: Schematic Illustration of X Relative Addressing

B relative indexed addressing (,X = 1 I = 0 ,B = 1)

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.

Example:

STA * +2, X, B

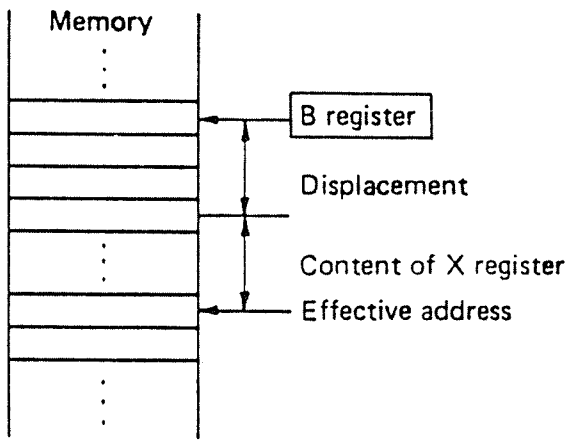


Figure II.10.8: Schematic Illustration of B Relative Indexed Addressing

Indirect P relative indexed addressing (,X = 1 I = 1, B = 0)

The contents of the P register are added to the displacement and produce a sum which is an address. The contents of the location of this address are added to the contents of the X register, which gives the effective address.

This mode allows successive elements of an array arbitrarily placed in memory to be accessed in a convenient manner.

Example:

STA I * +2,X

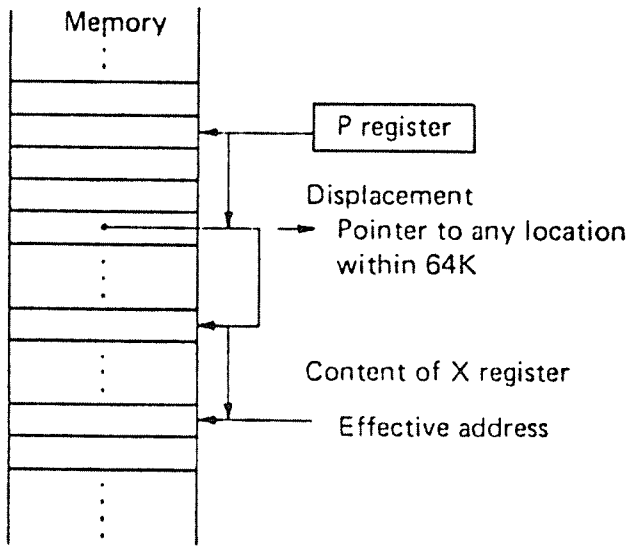


Figure II.10.9: Schematic Illustration of Indirect P Relative Indexed Addressing

Indirect B relative indexed addressing (,X = I 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.

Example:

```
STA  I  * +2,X,B
```

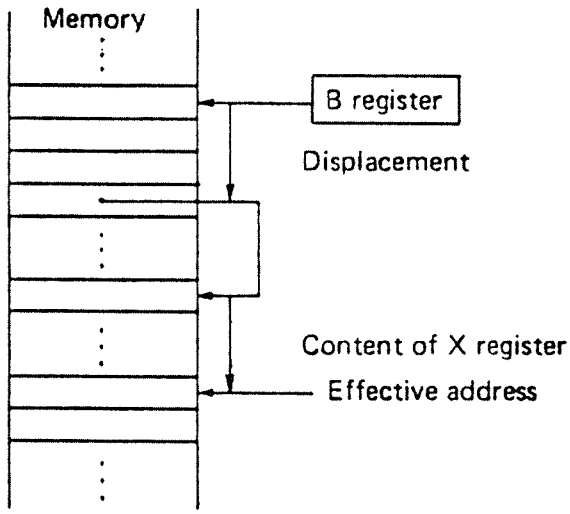


Figure II.10.10: Schematic Illustration of Indirect B Relative Indexed Addressing

Further information about these addressing modes is given in the NORD-100 Reference Manual.

II.10.3 *PRINCIPLES OF ADDRESS ARITHMETIC*

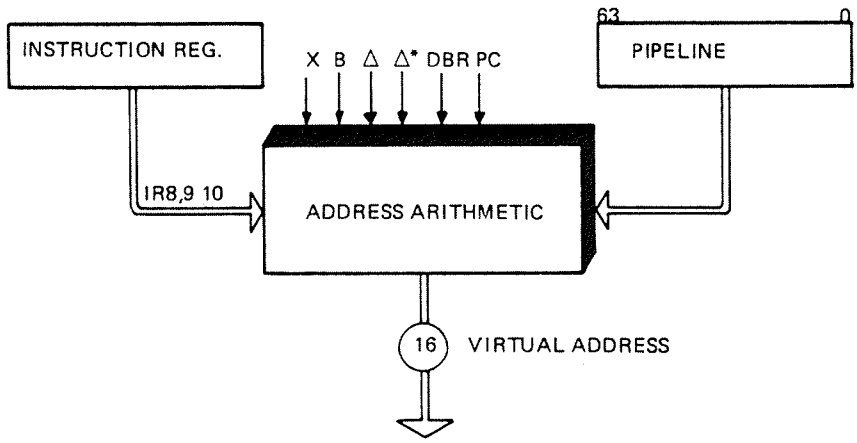


Figure II.10.11: Address Arithmetic — Input and Control

Data inputs to the address arithmetic are the following:

- the X register
- the B register
- the program counter (PC)
- DBR (data bus read register) used for indirect addressing
- the displacement (Δ) of the instruction, taken from GPR
- the displacement sign extended (Δ*) also taken from GPR

Sign extension is performed by setting bit 8 to 15 of the displacement equal to bit 7. The sign bit from the instruction will then be extended to bit 15.

Three bits from the instruction register (IR) give the addressing mode information. The microprogram controls the whole operation of the address arithmetic by giving the manipulation signals to the data hardware via the pipeline register.

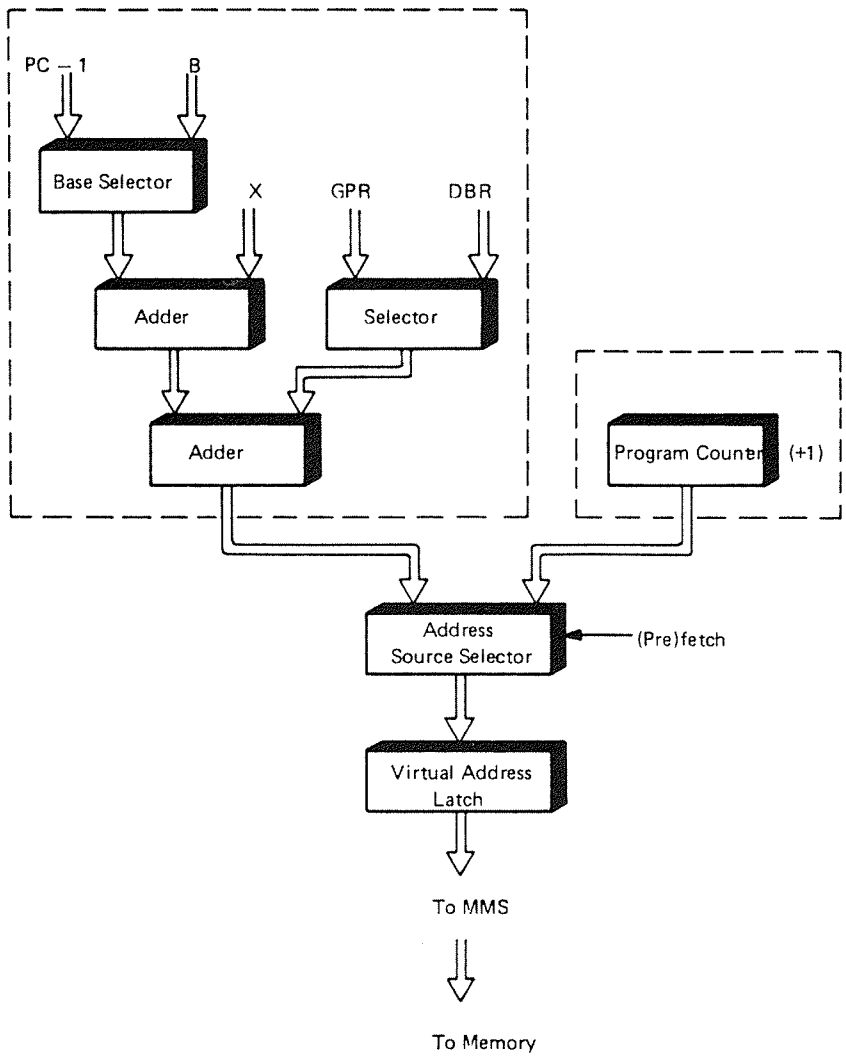


Figure II.10.12: Address Arithmetic — Functional Operation

Figure II.10.12 shows the functional operation of the address arithmetic. The program counter (PC), located in the ALU will always be selected and read by the control lines from the pipeline register when a new instruction is fetched. In the same operation the PC will be incremented with one. The new value of the program counter will go out to memory either as a virtual address via the memory management system or directly to prefetch the next instruction. In all memory references, the address calculation is performed by the micro program.

Let us assume that the last microinstruction in a given machine instruction is being processed and a fetch is issued:

Example 1:

The terminated instruction modified the PC and the prefetched instruction has no value. An ordinary fetch is issued by selecting the new program counter in the address selector, as an address to the next machine instruction. The PC will, at the same time, be incremented and a prefetch is issued.

Example 2:

The terminated instruction did not modify the PC and the next instruction is found prefetched in GPR. Let us now assume that this is a

LDA Δ instruction (for example, LDA * +2)

When this instruction is started by being mapped to the LDA routine, the PC, which is now incremented, will be selected to do a prefetch cycle.

The microprogram will then calculate the effective address by adding PC — 1 (because PC is now incremented by 1) to the displacement, which is found in GPR.

Example 3:

The same as in Example 2, but assume that the instruction is

LDA I Δ , B, X (for example, LDA I +2, B, X)

We notice that indirect addressing is used, and will force the address arithmetic to make two memory references. Pre- and post-indexing will also be performed.

As mentioned in Example 2, when starting to execute this instruction, a prefetch cycle will also take place.

First Memory Cycle:

To find the indirect address, the B register is added to the displacement (pre-indexing) found in GPR, and go out as a virtual address. The indirect address will be loaded into DBR from memory.

Second Memory Cycle:

In order to find the address of the operand to be loaded into the A register, the indirect address (DBR) is added to the content of the X register (post-indexing).

SECTION III

THE MEMORY MANAGEMENT SYSTEM

III.1 GENERAL

The hardware memory management module is necessary for running the SINTRAN III/VS (Virtual Storage) operating system which 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
- paging mechanism

This means:

Virtual address space

For each programmer, a virtual storage of 64 K is available regardless of the size of the physical storage. The physical storage may be greater or smaller than this. The programmer does not have to worry about whether there is enough physical address space in storage when the program is to be run, or whether other programs are using that part of storage or not.

In order to implement virtual storage, an intellegent addressing translation mechanism must be employed. This mechanism is under control of the operating system. A program is always written for virtual storage and the addresses used will be virtual addresses. The virtual addresses will be translated into physical addresses.

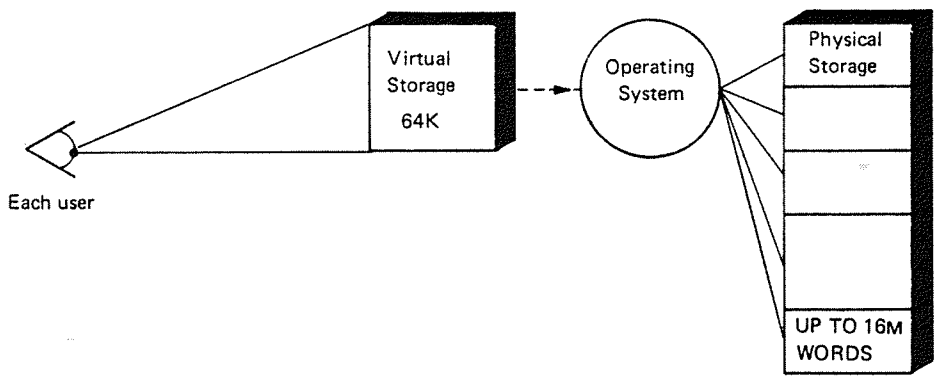


Figure III.1.1: Virtual Address Space

In NORD-100, up to 16 M words of physical memory may be used; a 16 bits virtual address will therefore be translated into a 24 bits physical address.

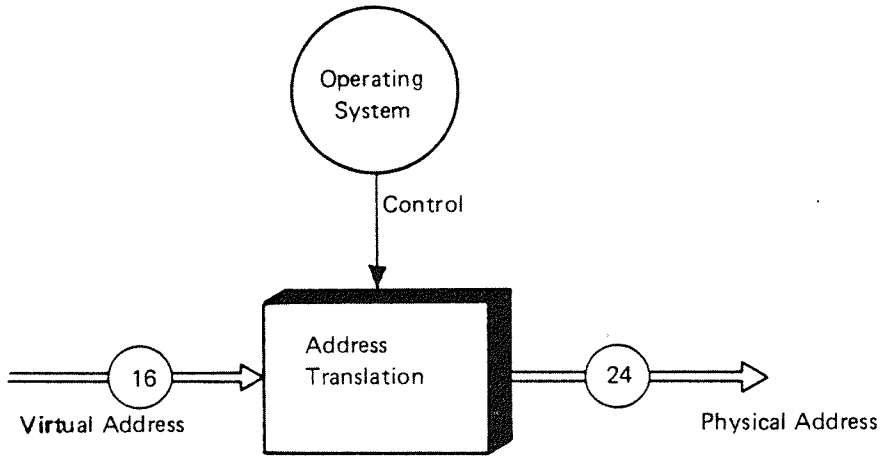


Figure III. 1.2: Virtual to Physical Address Translation

Dynamic allocation

Regardless of the virtual address space being used, the address translation mechanism will put the program in the most suitable physical address space at the time. For best storage utilization, the program may be scattered in physical storage.

Dynamic relocation

Since the address translation mechanism is dynamic, the program may be moved to any place in the physical storage.

Memory protection

Memory protection is not attached to predefined memory areas, but to the program parts and will follow those parts as they move around.

No external fragmentation

Due to the paging mechanism, no unused areas between program will occur. Programs are broken up in physical storage and loaded where vacant pages are found.

More parallelism

Also due to the paging system, only at the moment actual parts of a given program reside in primary storage. This gives room for more programs to be executed in parallel (multi-processing).

Some drawbacks must also be accepted:

— Increased memory access time

  Some time is required for the address translation. This time will be added to each memory cycle.

— Increased execution time

  Data transport to and from mass storage during paging is slow compared to the speed of the processor. A longer execution time can therefore be the result for a given program. To reduce the amount of paging, well structured programs are preferable.

— System overhead

  Control associated with the dynamic address translation is managed by the operating system. Prior to a mass storage transport initialization overhead must be accepted.

III.2 *IMPLEMENTATION*

III.2.1 *THE PAGING AND PROTECTION SYSTEM*

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

— the paging system
— the memory protection system

The paging system can work in two modes, the normal and the extended mode. The normal mode, which is compatible with the NORD-10 paging system, maps a 16 bit virtual address into a 19 bit physical address, extending the physical address space from 64 K to 512 K words (128 K to 1 M bytes).

The extended mode, which covers an address range of 16 M words, maps the 16 bit virtual address into 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. Data and instruction pages may be allocated anywhere in memory without restriction.

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 user 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.

Four page tables, each consisting of a protect table and a mapping table, hold the paging and protect information assigned to an active program. These tables are located in high speed registers directly connected to the internal data bus (IDB) in the CPU, reducing page overhead to practically zero.

III.2.2 *CPU CONNECTION*

The CPU module is normally located in position number 1 in the card crate. When the memory management module is present, it is placed in position number 2. The communication between the two modules is performed over a special backwiring connected to the B plug. Figure III.2.1 shows the CPU memory management module connection.

The memory management module contains, in addition to the memory management system, the cache memory and the display processor, controlling the optional display. These are connected to the internal memory management bus (MMB). MMB is an extension of IDB in the CPU for exchange of data and addresses.

The memory management module is also connected to the NORD-100 bus, for physical address generation and for receiving data to the cache memory.

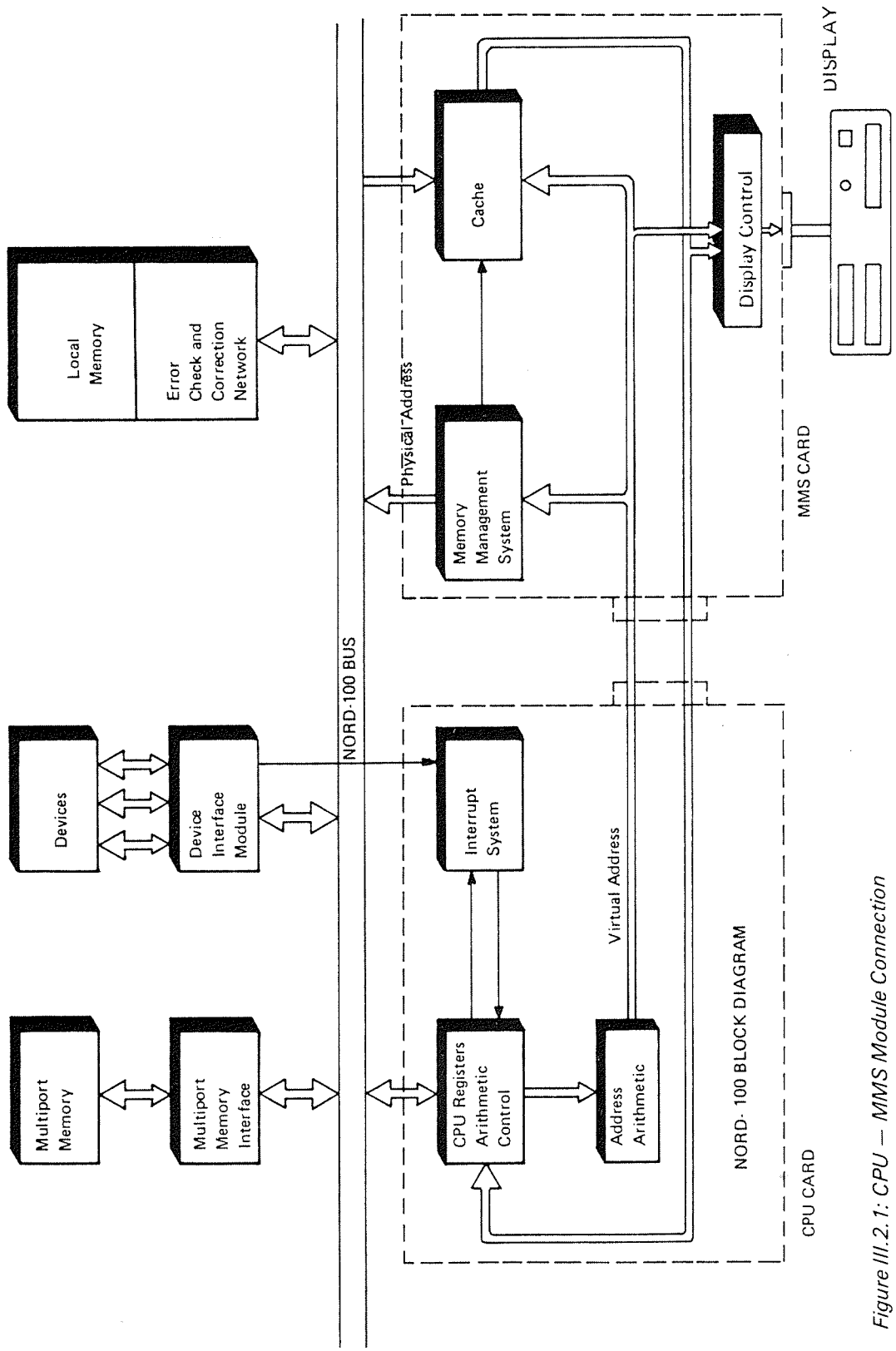
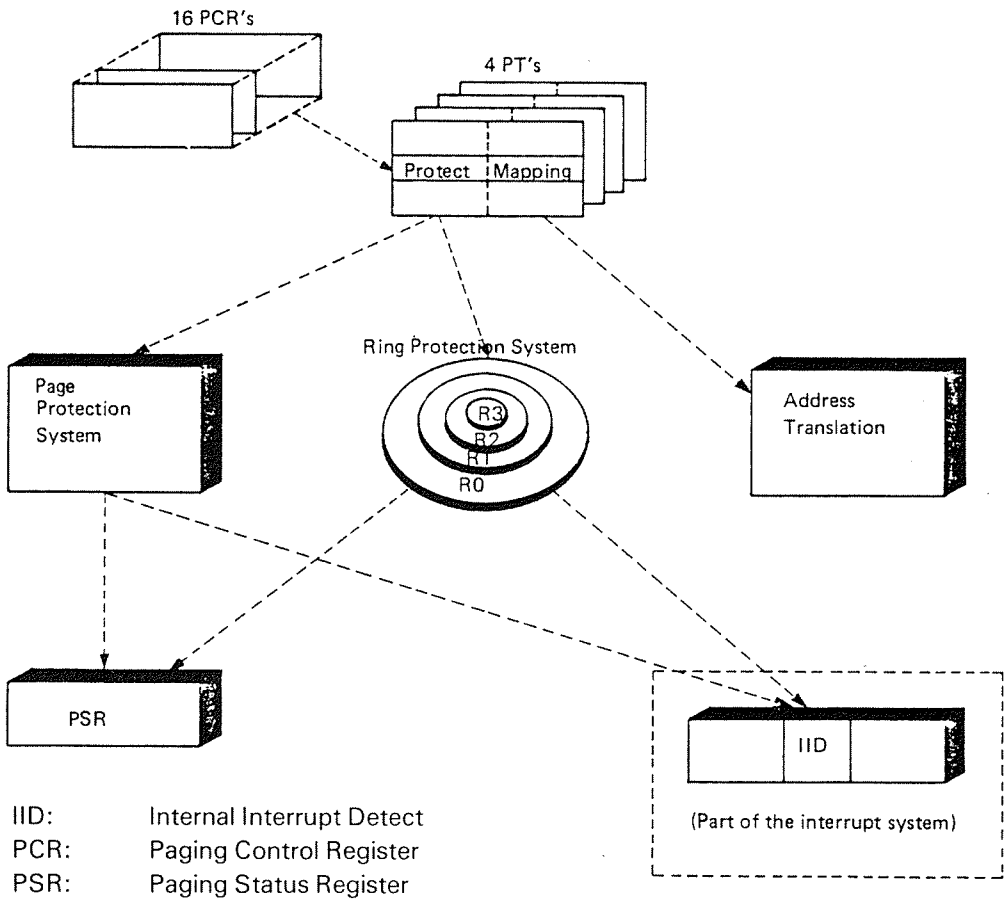


Figure III.2.1: CPU – MMS Module Connection

III.2.3 *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



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

Figure III.2.2: Memory Management Building Blocks

III.3 ADDRESS TRANSLATION

III.3.1 *VIRTUAL TO PHYSICAL ADDRESS MAPPING*

Refer to Figure III.3.1.

The paging system maps the 16 bit virtual address from the address arithmetic into a physical address. The number of bits in the physical address depends on if the normal (19 bits) or extended (24 bits) addressing mode is used. In the following explanations, the extended mode is used. The normal mode is found in brackets ( ).

The paging system divides the memory into memory blocks or pages of 1024 words or 1 K words. The pointer to these pages are found in the page tables. In order to map 64 K words of virtual address space, a 64 entry page table (PT) is required. The 16 bit virtual address from the address arithmetic is sent from CPU to the memory management system.

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

The program level (PIL) from the status register determines which of the 16 paging control registers (PCR) to use. The PCR contains, among other things, the page table select (PT) and alternate page table select (APT). Which of these in the selected PCR is to be used, is determined from the page table modus given status bit 0. The PCR determines which of the 4 page tables to select, and VPN addresse an entry in the selected PT.

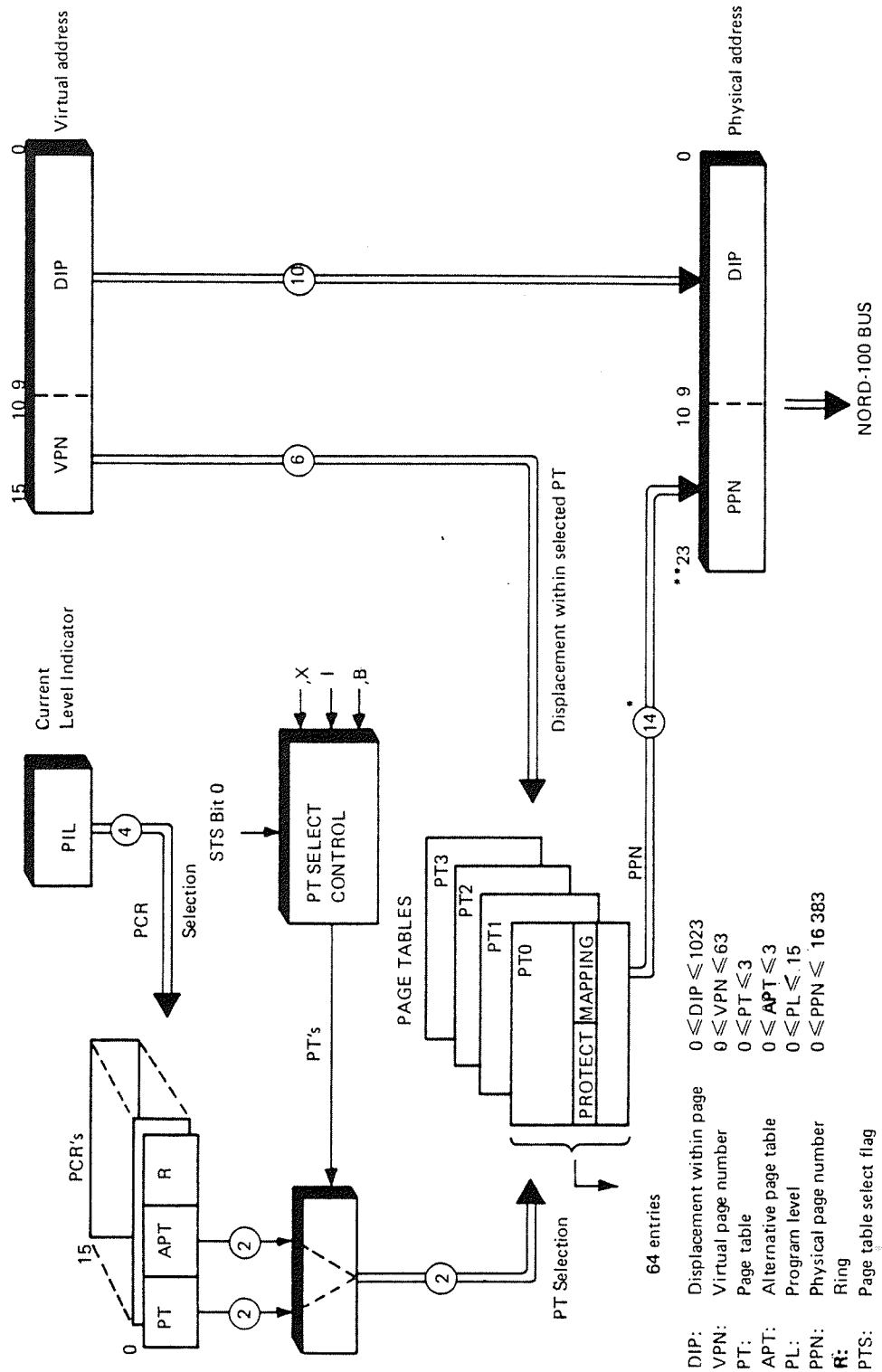


Figure III.3.1: Virtual to physical address mapping

When a memory request is performed, the content of the PT is looked up. Seven bits of the protect table in the PT entry are used for the protection system and are discussed later. Fourteen bits (or 9) from the mapping table, called physical page number (PPN) are transferred to the NORD-100 bus. PPN can attain values from 0-16383 (511), hence it is possible to access 16 M (512 K) words physical memory.

Prior to program start, the operating system will set a proper value in the protect and the mapping table in the PT. The address translation is therefore under control of the operating system.

III.3.2 *PAGE TABLE SELECTION*

NORD-100 has 4 page tables. Which one is 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. 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.

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

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.

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, giving one process access to 128 K of virtual memory instead of 64K. Normally, however, PT and APT will both point to the same page table so that accesses via PT and those via APT will give the same result.

III.3.3 *PAGE TABLE ASSIGNMENT*

Figure III.3.2 shows page table assignment under SINTRAN III.

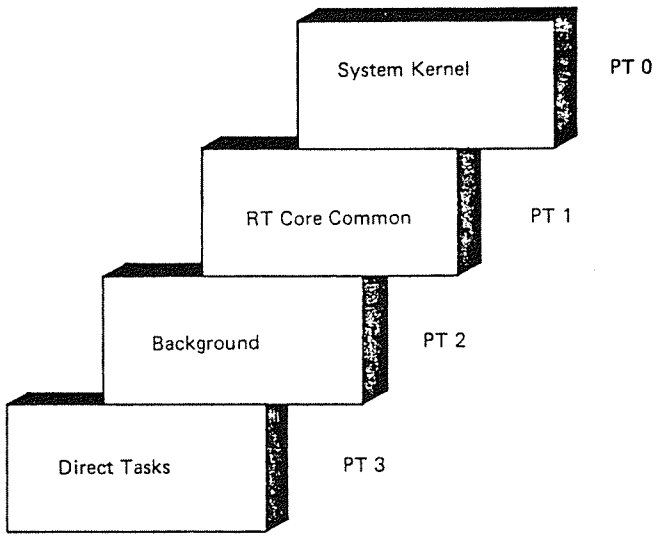


Figure III.3.2: Page Table Assignments

Page table 0:

Contains physical page numbers which point to the core resident part of the operating system (SINTRAN III), and system segments used by the operating system.

Page table 1:

Pointers to the physical pages for core common and real-time (RT) programs are located here.

Page table 2:

Physical page numbers pointing to background users and batch jobs for editing, compiling and loading of programs under timesharing are located here.

Page table 3:

Contains physical page numbers which point to pages in memory used for special direct task applications.

The page table entries for core resident and core common will be initialized at system start and will never be changed. The rest will contain dynamic entries for the currently running programs using the different pages. The unused table entries contain zero.

III.4 MEMORY PROTECTION SYSTEM

III.4.1 *GENERAL*

The memory management system employs two memory protection systems: a page 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 1 K 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 violation).

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.

III.4.2 *LAYOUT OF PAGE TABLES*

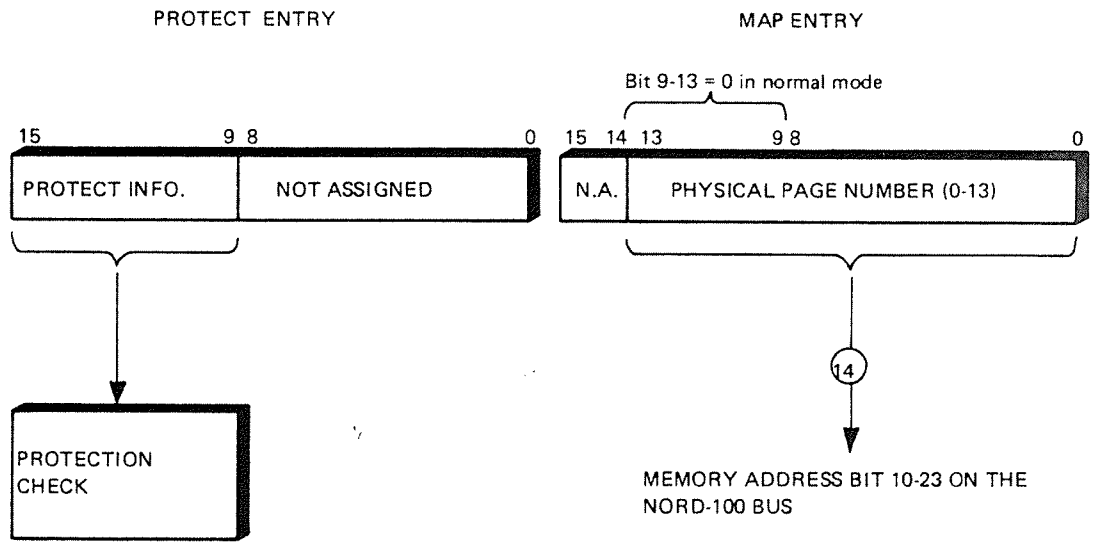


Figure III.4.1: Layout of an Entry in the Page Table

In the following, it is of vital importance to separate what the page tables look like as seen from program (as shadow memory) from how it works physically during the paging process. Here, the latter is described which is the same in normal and extended mode. Chapter III.6 describes the page tables operated as shadow memory. That handling is very different in normal and extended mode.

The paging process begins with lookup of an entry selected by the 6 bit VPN (see Figure III.3.1). This entry is 32 bits long for both modes and consists of two parts, *protect* and *map* as shown in Figure III.4.1.

Thus, each 1K page has an associated entry in the page table which describes precisely what to do when the program uses that page.

The *map* part is straight-forward. It simply tells where in memory the page is placed, i.e., the physical page address. Since the address *within* a 1 K page is unaltered (DIP in Figure III.3.1), 24 — 10 = 14 additional bits are necessary to uniquely address 16 M words (extended mode). The normal mode of 512 K words is obtained by simply forcing the 5 most significant bits to zero (see Figure III.4.1).

III.4.3 *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 protect table 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 protect table have the following meanings.

Bit 15: WPM — Write Permitted

WPM = 0. It is impossible to write into locations in the 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 possibly 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.

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 pages which have RPM = 1.

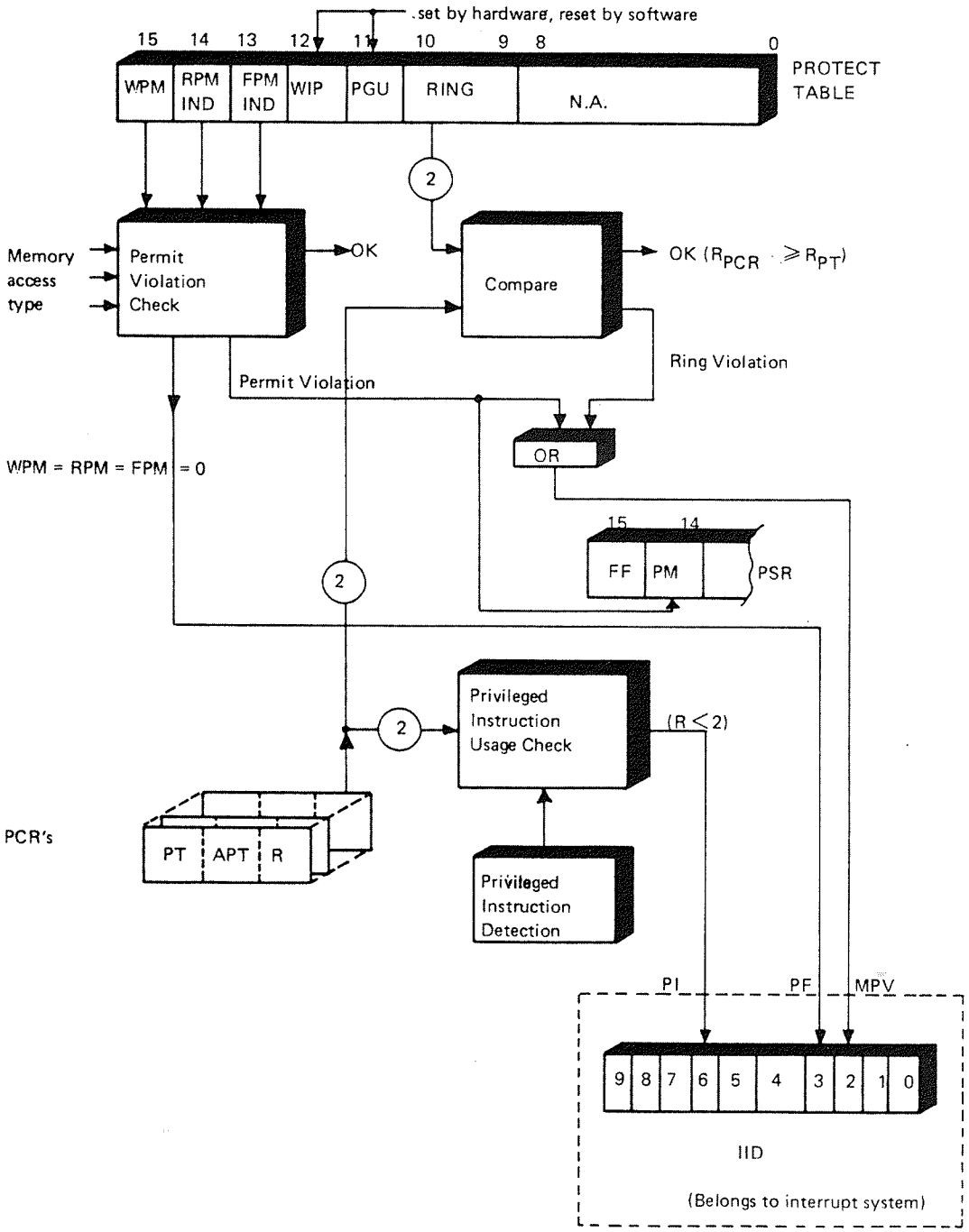


Figure III.4.2: Memory Protection

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 code, IIC, equal to page fault.

Refer to Figure III.4.2 which shows the memory protection system.

Abbreviations in Figure III.4.2:

FF:	fetch fault
FPM:	fetch permitted
IID:	internal interrupt detect register
IND:	indirect address readout permitted
MPV:	memory protect violation
PCR:	paging control regisrer
PF:	page fault
PGU:	page used
PI:	privileged instruction interrupt
PM:	permit violation
PPN:	physical page number
PGS:	paging status register
PT:	page table
R:	ring number
RPM:	read permitted
WIP:	written in page
WPM:	write permitted

III.4.4 *RING PROTECTION SYSTEM*

III.4.4.1 *General*

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

There has been some confusion about the ring concept due to ambiguities in earlier manuals. To clarify the ideas, a somewhat metaphorical description is given. In this chapter the definitions below are convenient but *they may not apply elsewhere.*

III.4.4.2 *User*

Each of the 16 interrupt levels are dedicated to a *process* or *user*. "User" is here meant in a broad sense and is not restricted to "background users" or "timesharing users" which are often abbreviated "user". Hence, *user* is the "manager" of programs.

III.4.4.3 *User Ring*

The *user* may belong to one of four *rings*depending on whether he is noble or public (see below). The *user ring* is defined by the Paging Control Register (PCR). Each level and hence the *user* of that level is assigned a specific ring.

III.4.4.4 *Program*

A program is the tool a user has to perform a specific task and it consists of instruction code and pure data. In this context a *program* is executed by a *user*. A *program* itself cannot execute code because it *is* code.

III.4.4.5 *Program Ring*

A program may be good or bad depending on how much harm the user may cause by executing them. To protect delicate yet powerful programs from potential riot users, fences are built around them. The field inside the concentric fences are called program rings. Each page of program has its program ring defined in the associated page table entry (protect bits 9-10).

III.4.4.6 *Ring Usage*

Of the four rings 0-3, number 3 is the most distinguished and never available to the general public. It is reserved to the Most Noble User, the kernel of the operating system.

As in human societies, the nobler the user is the more privileges he acquires. The privileges of the four user rings are defined by the paging control register bit 0-1:

PCR bits 1 0

0 0 Ring 0:

Users of this ring may not execute privilged instructions. They may only access locations in program ring 0. Locations outside ring 0 are completely inaccessible.

0 1 Ring 1:

Users of this ring may not execute privileged instructions. They may access locations in program ring 1 and ring 0.

1 0 Ring 2:

All instructions are permitted on this ring. The user may access locations in program rings 2, 1 and 0.

1 1 Ring 3:

All instructions are permitted and the whole address space, including the page tables, is accessible if not protected by the RPM, WPM and FPM bits.

The program rings do not imply any privileges but describe the quality of the program.

Ring 3 users with all privileges must therefore dispose program of top quality since even a small slip may destroy the system. A ring 0 user can only create havoc on his own accessible programs, other users are unperturbed by his thrashings.

An illegal ring access or illegal use of privileged instructions will cause an internal interrupt on level 14 and the forbidden action will be avoided. It should be realized that the most perfect program may cause disaster in the hands of an irresponsible user. But equally dangerous are lousy programs executed by superior users. If, therefore, a ring 3 user tries to execute instructions on program rings 0, 1 or 2 he is allowed to do so. However, he is immediately expelled from high society and degraded to the ring of the accessed program. Such accesses are detected by hardware which automatically changes the user ring on current level in the PCR register.

Observe that this degrading only happens by executing of lower ring *instruction codes* but not by access of pure data.

III.4.4.7 *Ring Assignment*

The recommended way of assigning user rings is:

Ring 0: Timesharing users
Ring 1: Compilers, assemblers, data base systems
Ring 2: Operating system, file system, I/O system
Ring 3: Kernel of operating system

This may be visualized in the following manner.

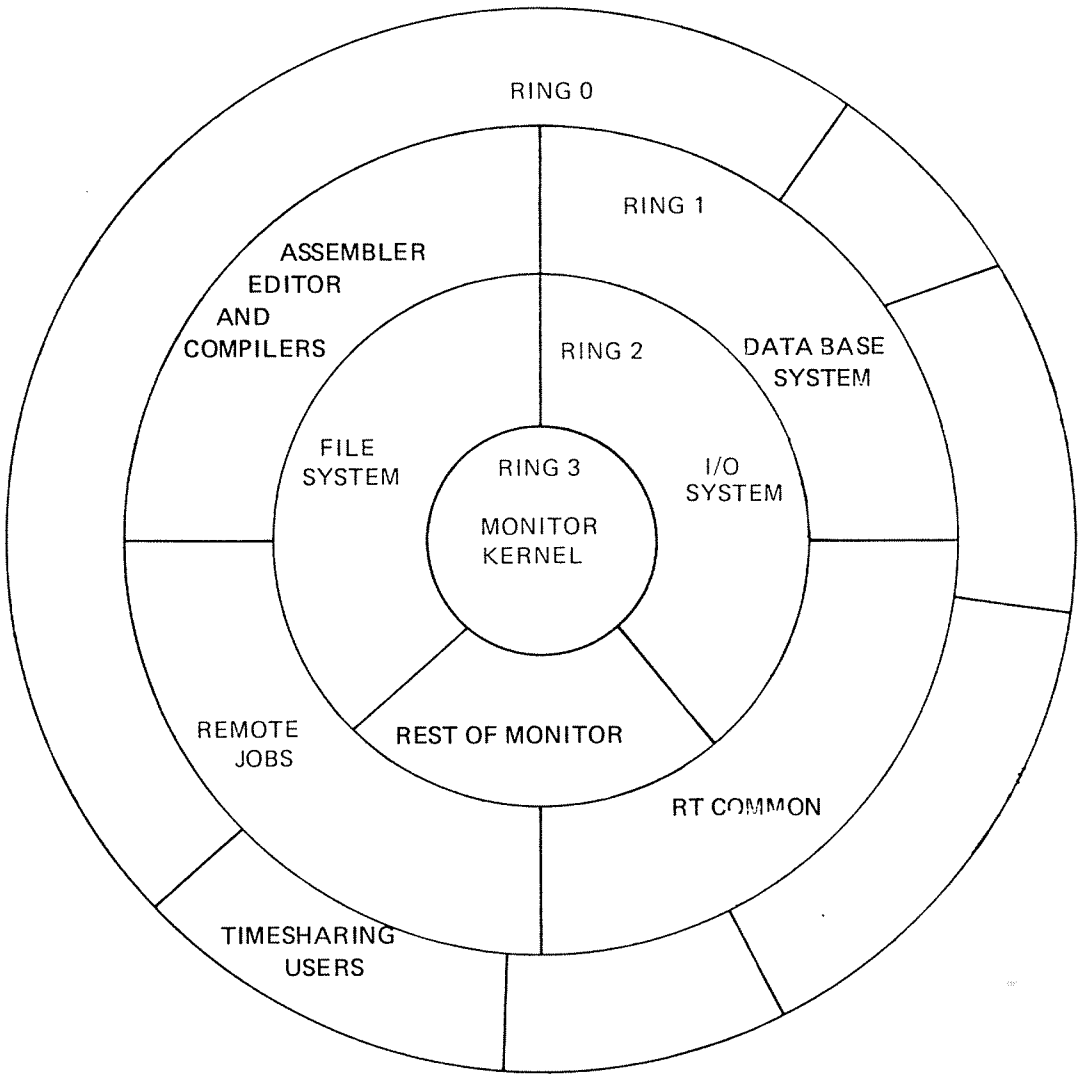


Figure III.4.3: Ring Assignment

III.4.5 *PRIVILEGED INSTRUCTIONS*

In a multitask system, a background user is not permitted to use all instructions in the instruction set. Some instructions may only be used by the operating system, and these 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 rings 2 and 3, i.e., only by the operating system. If users on rings 0 and 1 try to execute any privileged instructions, a privileged instruction interrupt will be generated and the instruction will not be executed.

III.4.6 *PAGE USED AND WRITTEN IN PAGE*

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 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 brough 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 by the operating system to maintain a record of the access frequency of a page. This is used in decisions making the replacement algorithm, i.e., to determine which page should be swapped.

III.5 MEMORY MANAGEMENT CONTROL AND STATUS

III.5.1 *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.

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".

III.5.2 *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 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.

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.

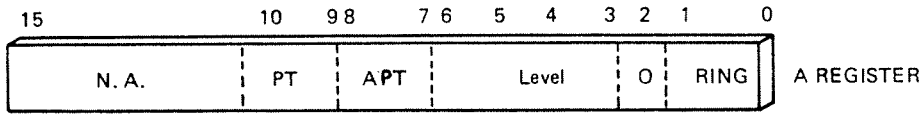
Bit number 13 in the status register is reset, indicating normal address mode.

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

III.5.3 *PAGING CONTROL REGISTER*

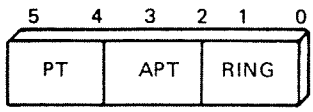
There is one PCR (paging control register) for each level. The setting of the PCR's is done by the operating system prior to the program execution. One PCR may be written into at a time by the instruction TRR PCR.

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



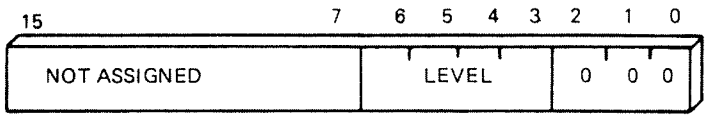
Bits 11-15:	Not assigned
Bits 9-10:	Page table number (0-3)
Bits 7-8:	Alternative page table number (0-3)
Bits 3-6:	Program level (PCR number) (0-15)
Bit 2:	Equals zero
Bits 0-1:	Ring number (0-3)

After the TRR PCR instruction the PCR will be organized as sixteen 6 bit wide registers on the memory management module:

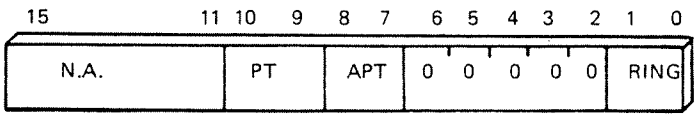


PCR FORMAT

For maintenance purposes, it may be desireable to read back the content of the 16 PCR's. This is accomplished by the TRA PCR instruction. Before execution the content of the A register must be:



After execution the content of the A register will be:

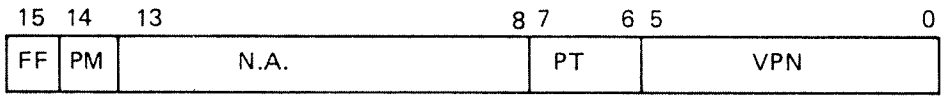


III.5.4 *PAGING STATUS REGISTER*

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 register, TRR PSR unlocks it again.

The bits in PSR have the following meaning:



PSR Format

Bit 15:

Memory management interrupt occurred during an instruction fetch.

Bit 14:

1 = permit violation (read, write, fetch protect system)
0 = ring protection violation interrupt
Permit violation has priority if both conditions occur.

Bits 6-7:

Page table number

Bits 0-5:

Virtual page number

Note that bits 0-7 are the address of the page table entry that failed. In normal mode they are the 8 least significant bits of the shadow address (refer to Chapter III.6).

In extended mode the address (bits 0-7) must be multiplied by 2 to give the 9 least significant bits of shadow address.

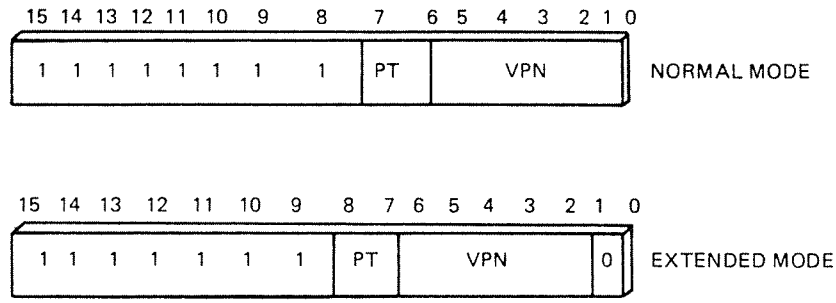


Figure III.5.1: Correspondence between Paging Status Register Bits 0-7 and Shadow Addressing

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.

III.6 CONTROL OF PAGE TABLES

III.6.1 *GENERAL*

The operating system manages the information to be put in the page tables. Some parts are fixed from system start time and others are dynamically changed and updated. When a new background user is started the operating system examines an administration table to see which pages are free. The map part of the page table is filled with the number of the free pages (PPN). If necessary pages are taken from other processes. According to the program properties the protect part is also filled.

As mentioned in Section III.4.2, one must separate the appearance of page tables as seen from program (shadow memory) and how it works during the paging process. During paging it is essentially the same whether in normal or extended mode. There are 4 subtables, PT 0-3, each consisting of 64 entries (see Figure III.6.1).

III.6.2 *SHADOW MEMORY*

In normal mode the content of each entry (16 assigned bits) can be transferred as one word. In extended mode each entry needs 21 bits and must there be transferred in two words.

To ease reading and writing of the page tables they are treated as normal memory. 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	177700 - 177777	177600 - 177777

This area is called shadow memory because it lies in the shadow of 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 III.6.1 shows the shadow memory addressing.

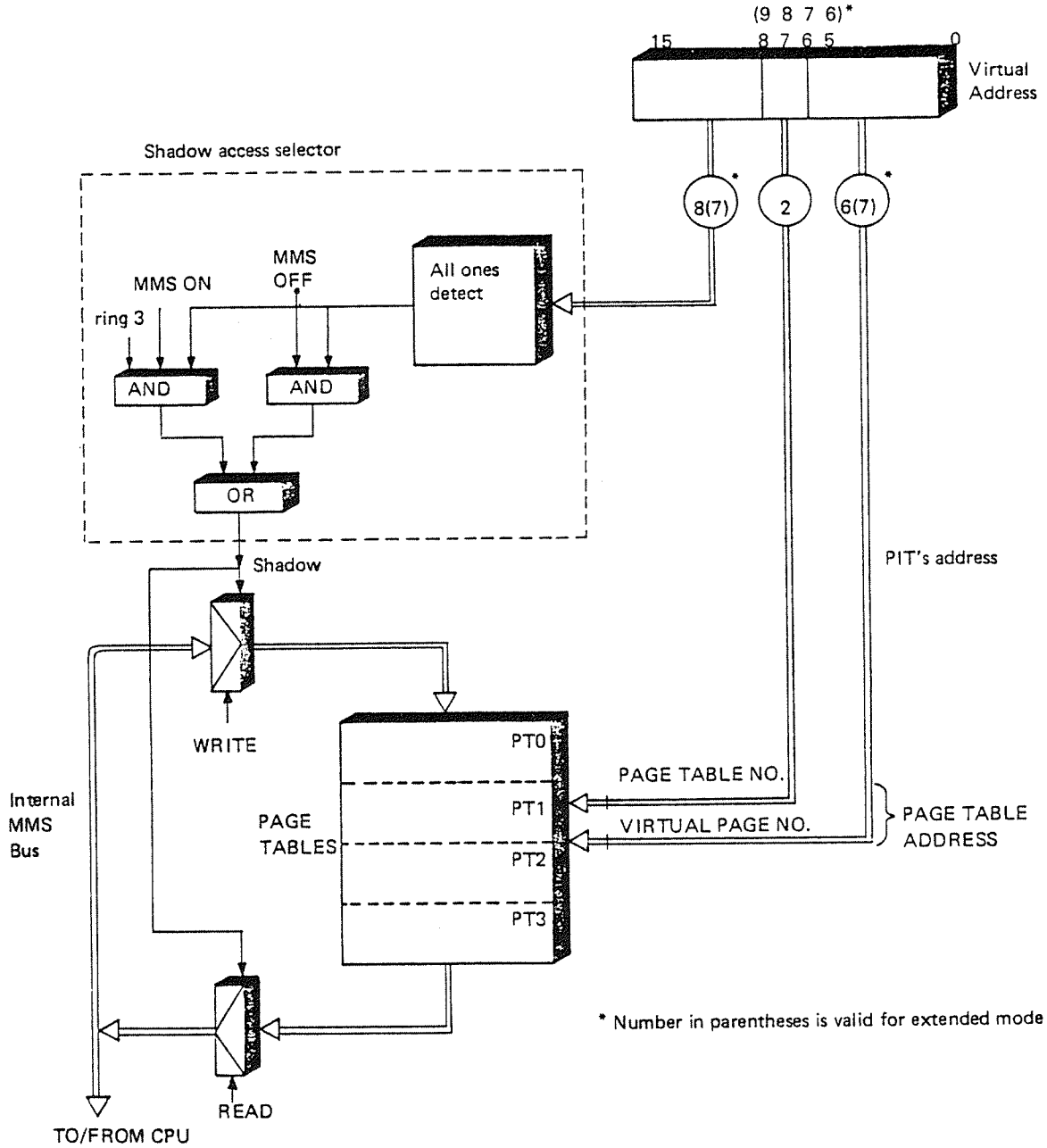


Figure III.6.1: Shadow Memory Access

III.6.3 *READING AND WRITING IN PAGE TABLES*

It should be kept in mind that whether shadow covers 256 or 512 addresses, the physical high speed memory is the same. Normal and extended modes determine two different ways of filling up the same tables as seen from program. This process is shown in Figures III.6.2 and III.6.3. For simplicity only page table 3 is shown but all tables are handled in the same way.

The reason for the unused bits in the page tables is that it shall be possible to read and write any content in the tables without interpreting it as paging information. When paging is off the page tables may be used as ½ K very fast random access memory.

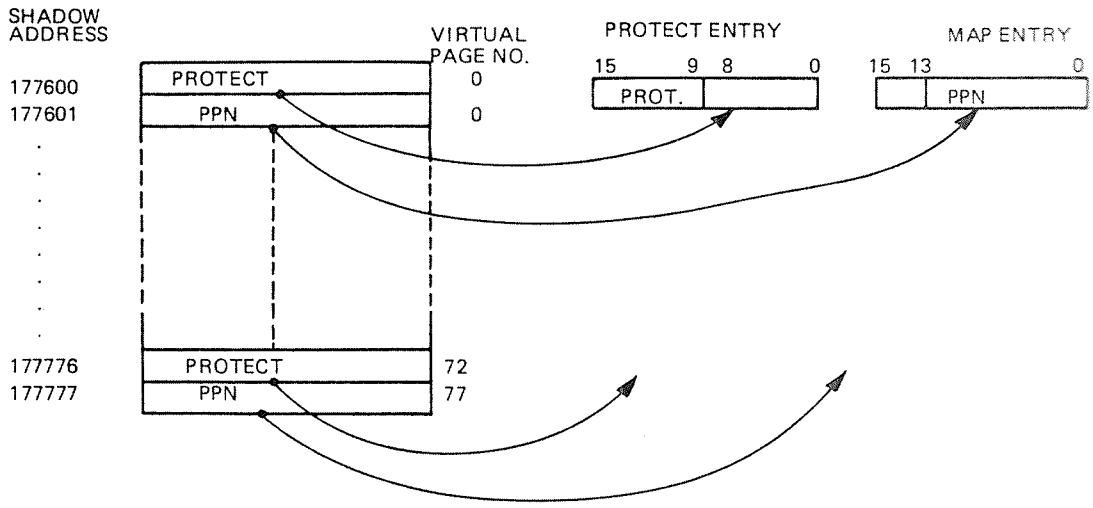


Figure III.6.2: Reading and Writing Page Table 3 Entries as seen from Program in Extended Mode

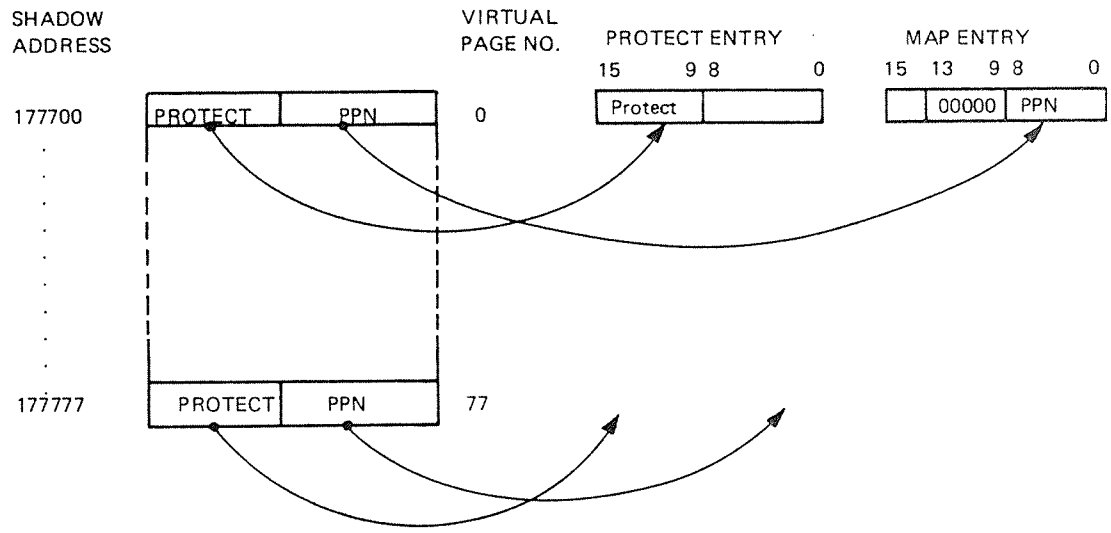


Figure III.6.3: Reading and Writing Page Table 3 Entries as seen from Program in Normal Mode

III.7 TIMING

As soon as the virtual address is calculated in the CPU, it is present on the memory management module, because of its close connection to the CPU's internal data bus. Page table accesses are performed in parallel with cache memory lookup (also located on the memory management module) and, consequently, there is no timing overhead associated when the content is in cache. However, if there is no hit in cache, the paging system will introduce 50 ns overhead.

III.8 EXAMPLES

Example 1:

Assume that a user has a 3 K program. The start address is 40000_8, i.e., the address space is 40000_8 — 45777_8 since 3K = 3072_{10} = 6000_8. Refer to Figure III.8.1.

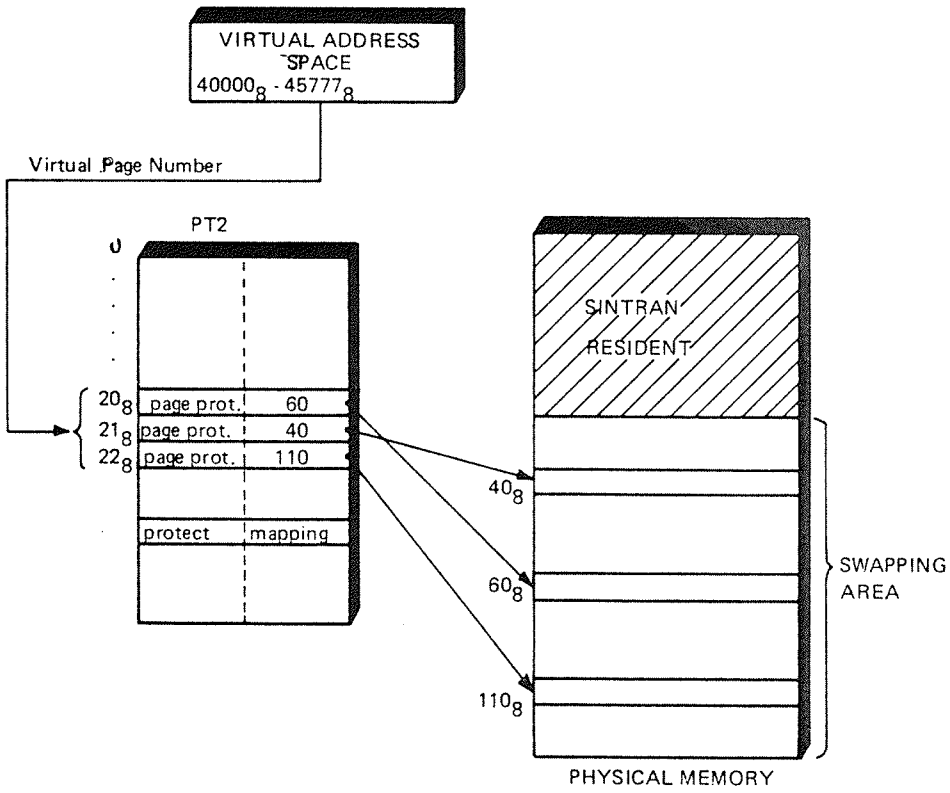


Figure III.8.1: Virtual to Physical Address Conversion

From Figure III.8.1, it can be seen that the logical or virtual address space of 40000_8 — 45777_8 always will use the virtual or logical page numbers 20_8 — 22_8. However, where in physical memory the 3 pages will be located, is controlled by the physical page number.

Example 2:

A user has a 3 K program and the address space for the program is 40000_8 — 45777_8. Belonging to the program is also 3 K of data in the address space 46000_8 — 53777_8. Assume that the program is to be started on level 1, which means that PCR 1 is selected. Figure III.8.2 shows the page table selection if the following conditions are fulfilled:

— Status register bit 0 (PTM — page table modus) is one, to enable use of the alternative page table select.

— PCR 1 contains 2 in the PT field and 3 in the APT field.

— All instructions are fetched using P relative addressing (always true, except for indirect jump).

— All data is accessed using B or X relative addressing (controlled by the programmer or eventually a compiler). The data accesses will then use the APT (refer to the table in the address translation section — III.3.2).

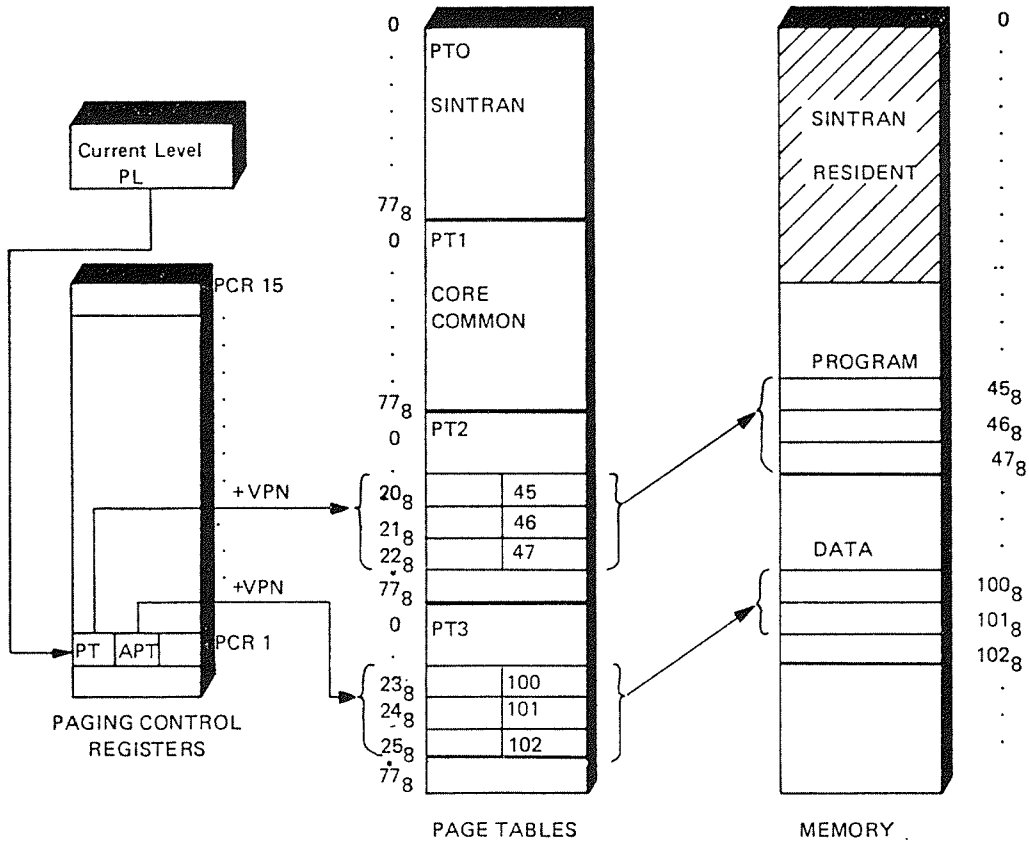


Figure III.8.2: PCR and PT Usage

Example 3:

In the previous example, it was pointed out that there was a possibility of destroying the operating system. By introducing the ring protect system we will show how the operating system is protected.

Figure III.8.3 shows the same situation as in Figure III.8.2 with the addition of PCR for program level 1, the level the program will be executing on.

During execution, user 2 makes an error and tries to write outside his predefined address space into the operating system virtual page number 14. Virtual page number 14 would permit a write into the page because WPM is set to one. This means that according to the read, write and fetch protect system, the access is legal. However, the protect table's ring bits for virtual page number 14 are compared to user 2's ring bits in the PCR 1. The ring bits for virtual page number 14 are equal to 2 and the ring bits for user 2 in PCR are 1. Therefore, according to the ring protection system, this is an illegal access. An internal interrupt, memory protect violation, will be generated. In this way, the operating system can be protected from users (i.e., all program systems which run on a lower ring than the operating system).

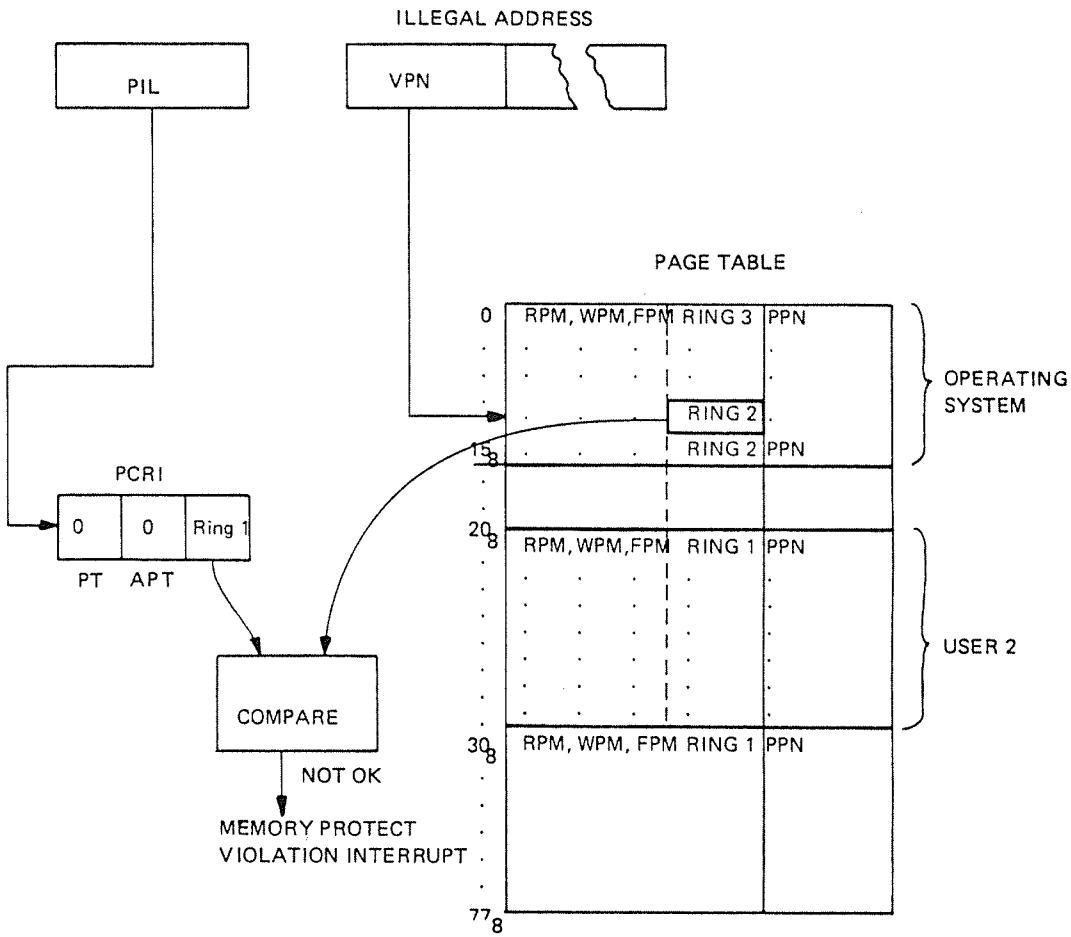


Figure III.8.3: Page Table Usage, Example

SECTION IV

NORD-100 BUS SYSTEM

IV.1 GENERAL

All system components and peripherals are connected to the high-speed NORD-100 bus. The NORD-100 bus provides the communication path through its bidirectional lines for addresses, data and control lines for devices in the bus except for communication directly between the CPU and the MMS and cache. Figure IV.1.1 shows the interconnections between the modules. The bus is general purpose, thus all system elements communicate with each other in identical fashion over this bus, allowing all modules of the NORD-100 system simply to be "plugged" into the system. 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

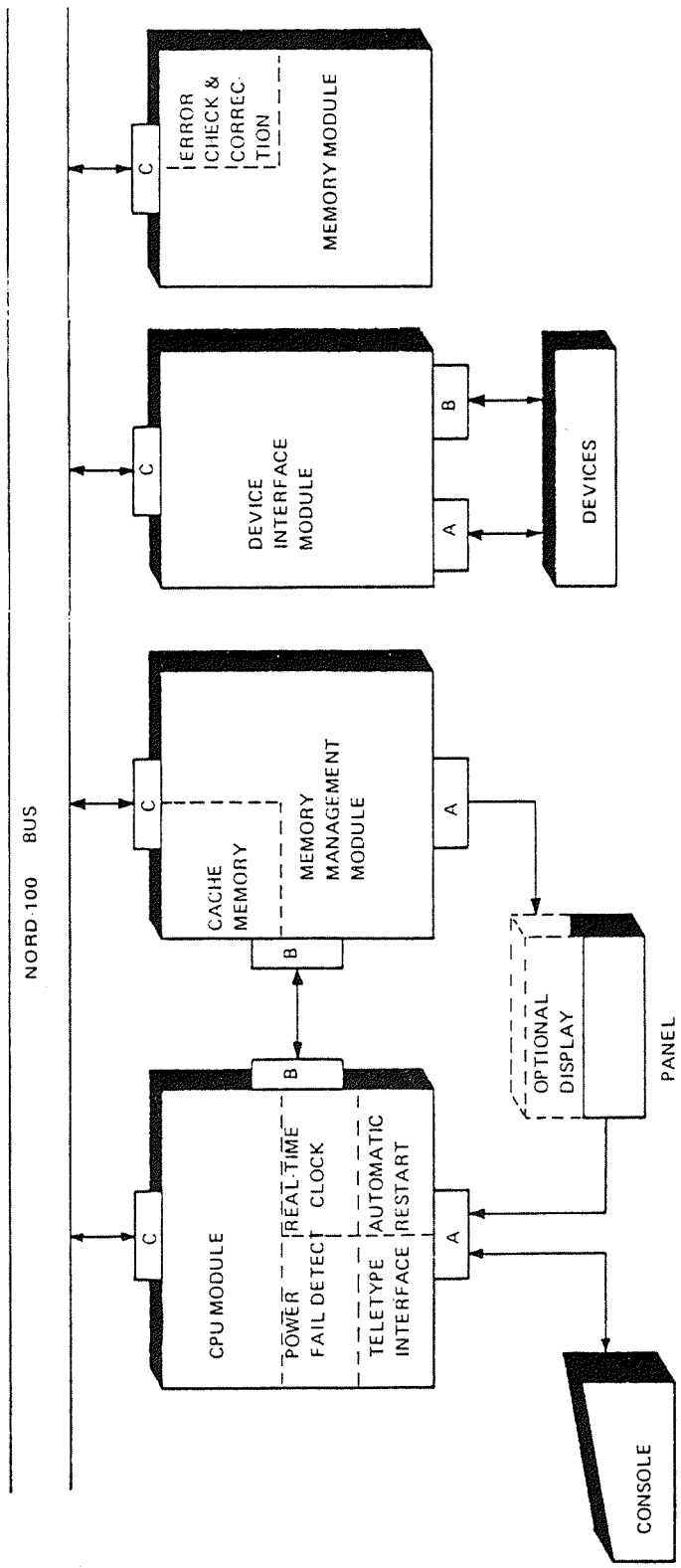


Figure IV.1.1: Interconnection between Modules

IV.2 BUS CONTROL

The NORD-100 bus is completely controlled by the Bus Control, which is an integrated part of the CPU. The control functions carried out by the bus control may be divided into two parts:

- allocating of the NORD-100 bus to one of the possible requesting bus users
- supervising that the NORD-100 bus is released by the granted user within a certain time limit

Before the NORD-100 bus may be used, it has to be allocated. That is, when either

- the CPU,
- a DMA controller or
- a memory refresh cycle

needs the NORD-100 bus, a bus allocation request is sent to the bus control.

The bus control controls the allocation of the NORD-100 bus for exchange of data, by setting up connections of the following types:

- the CPU to the memory system
- the CPU to the input/output system
- the DMA controllers to the memory system (by cycle stealing)

When several sources request the NORD-100 bus at the same time, the following priority is given by the bus control:

1. Refresh
2. DMA
3. CPU

Sources that gain bus allocation will not be interrupted by requests from sources on higher priority in the bus control during the bus cycle.

IV.3 PHYSICAL ARRANGEMENT OF THE NORD-100 BUS

Physically, the NORD-100 bus is available as a printed backplane. The backplane contains 12 positions for module connection. Including power and ground lines, a total of 96 lines are available.

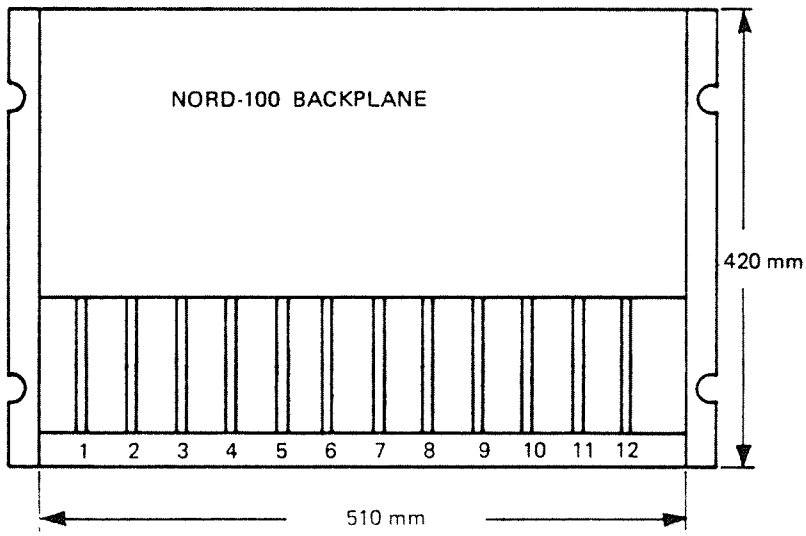


Figure IV.3.1: NORD-100 Crate Layout (Top View)

The NORD-100 bus may be divided into two logical parts:

- 24 bit wide parallel multiplexed address/data bus, supporting a physical address space of 16 M words
- control lines

All positons in the backplane contain the same information, i.e., all modules connected to the bus are presented the same information simultaneously and are continuously "listening" to the bus activity. This allows flexible configuration and reconfiguration of hardware.

The control lines are used to define the valid information on the bus (addresses or data) and to connect one source to one destination, during transfer of data between system elements.

A few extra control lines have been included for further extensions to multiple control units (bus switch).

IV.4 ORGANIZATION OF NORD-100 MODULES

One NORD-100 card crate can contain up to 12 modules.

All NORD-100 modules are made to a common standard. Every module has at least one connector used for connection to the NORD-100 bus. In addition, a NORD-100 module may have one or two extra connectors carrying a total of 128 lines (64 lines in each connector).

Figure IV.4.1 shows a standard NORD-100 I/O module and the connector usage.

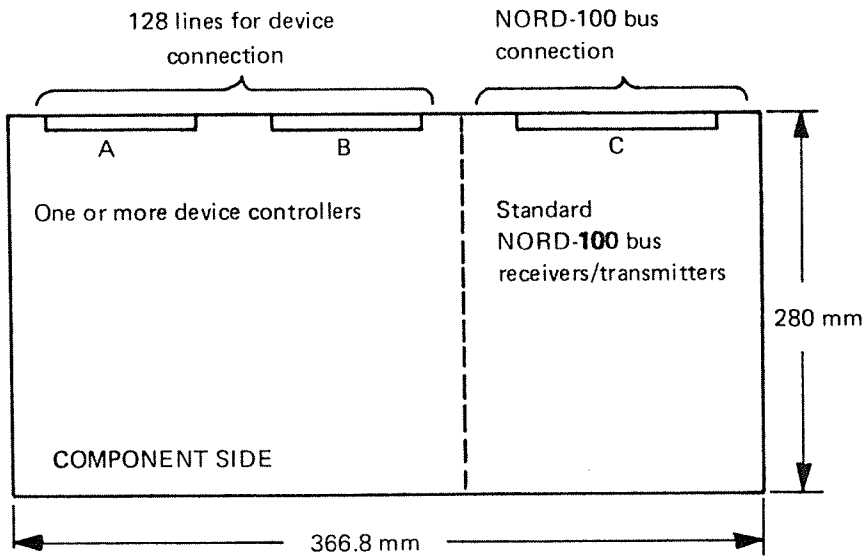


Figure IV.4.1: NORD-100 Module and Connectors

The plugs are assigned an identification letter as illustrated. The plug used for connection to the NORD-100 bus is assigned the letter C. This plug contains 96 pins, each of them defined in accordance to the NORD-100 backplane (bus) standard.

The plugs A and B each carry 64 lines with nondefined use. In the design of I/O device controllers, there plugs are used for connection to the external devices.

Figure IV.4.2 below shows the card crate and the usual placement of modules in a medium sized NORD-100 system.

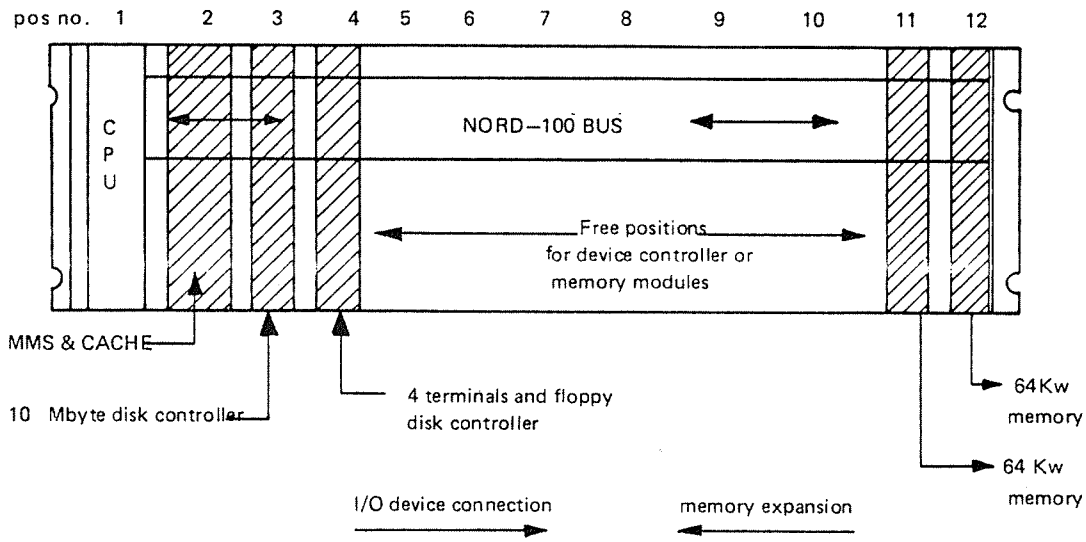


Figure IV.4.2: The Card Crate and Usual Placement of Modules

If the memory management system and the and cache module are 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 the cache module are *not* present, all I/O modules should be moved one position to the left.

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

If 12 positions (cards) are not enough, a new card crate could be added, thus expanding the NORD-100 bus with 12 new positions, with the help of a bus extender card.

IV.5 BUS TIMING CONSIDERATIONS

On a multiplexed bus, addresses and data share the same signal lines. Hence, every cycle consists of two phases, one where an address is present and one where the data is present. One could then be led to the conclusion that a cycle takes twice as long as for a bus with separate address and data lines.

This is fortunately not the case for two reasons:

1. The microprogram prepares or computes the address and data sequentially. Therefore, it is most efficient to also put them on the bus sequentially. (This would not be true if there were a separate address arithmetic unit.)

2. During a write to memory, the address is needed before the data. Since the data then comes immediately after the address, no time is lost waiting for it. In fact, as Figure IV.5.1 shows, during write cycles the memory availability for the CPU is better with the multiplexed bus! This is because the address can be sent out before the data is prepared. For read cycles, the timing is the same for both types of bus.

In IOX write cycles, there will normally be a *small* penalty in bus utilization but this is completely insignificant.

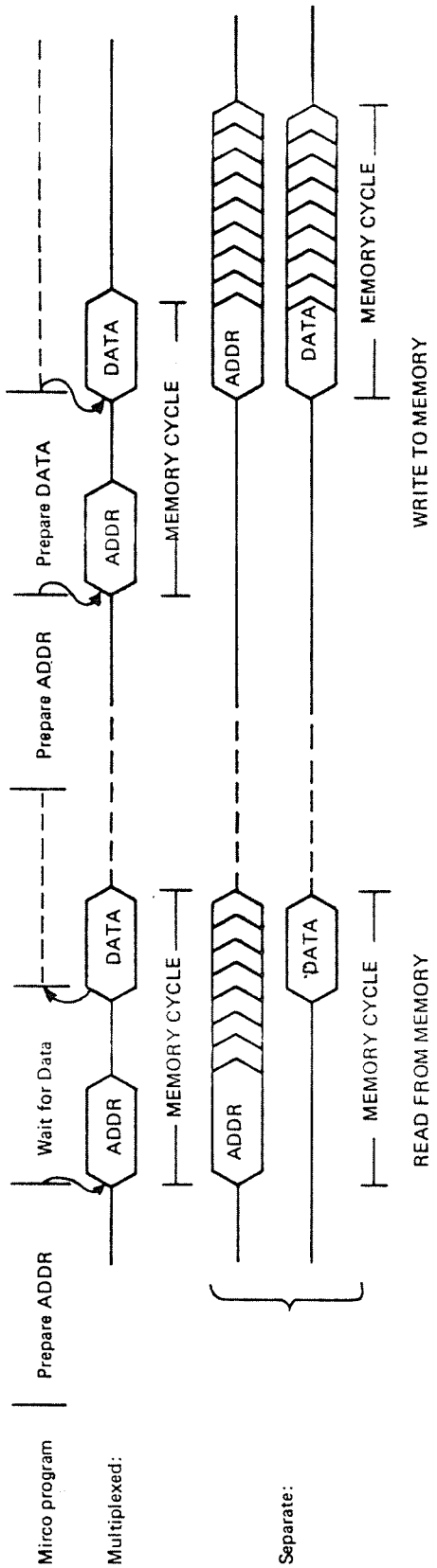


Figure IV.5.1: Comparison of Busses with Multiplexed and Separate Address and Data

A common bus offers also other advantages:

- few physical lines
- few drivers/receivers
- same type of drivers/receivers
- all devices listening (to the same bus) simultaneously giving fast response

In addition, precise balance and termination give very fast address/data set up time (typically 20 ns).

The bus is also fast enough to handle both DMA (direct memory access) activity and CPU activity at the same time without slowing down the CPU.

A CPU memory reference will occupy the bus for typically 450 ns, and a DMA transfer typically 550 ns.

SECTION V

THE NORD-100 STORAGE SYSTEM

V.1 GENERAL

Storage is one of the major building blocks in a computer system.

It is used to hold programs (instructions), data, addresses and results. Memory will regard all of these as information, i.e., it does not descriminate between the different types of information.

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

- low access time
- low storage cost
- large capacity

These requirements are usually conflicting.

V.2 THE MEMORY HIERARCHY

In an effort to satisfy these requirements, a memory system as illustrated below is employed.

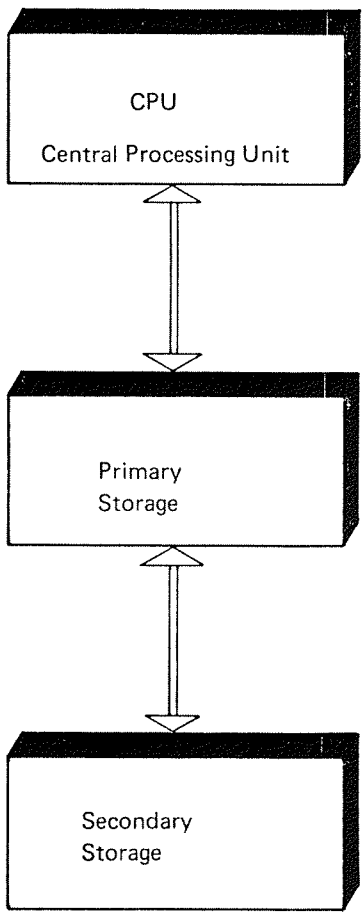


Figure V.2.1: Two Level Storage System.

In the model above, we have a two level storage system with

— a fast primary storage and
— a cheap secondary storage.

Since just a small fraction of the storage capacity is held by the primary storage, the cost per bit stored will mainly be dominated by the storage cost of the secondary storage. Storage cost in this system is accordingly relatively low.

Before program start, the program to be executed, or part of it, is transported from secondary to primary storage. While executing, CPU references will be made in the primary storage with a relatively short access time. The speed of the illustrated storage system is therefore close to the speed of the primary storage system.

V.3 NORD-100 MEMORY SYSTEM

In the NORD-100 memory system, a compromise between the conflicting requirements is achieved through the implementation of a multilevel hierarchical memory system. Figure V.3.1 shows the major building blocks in this system.

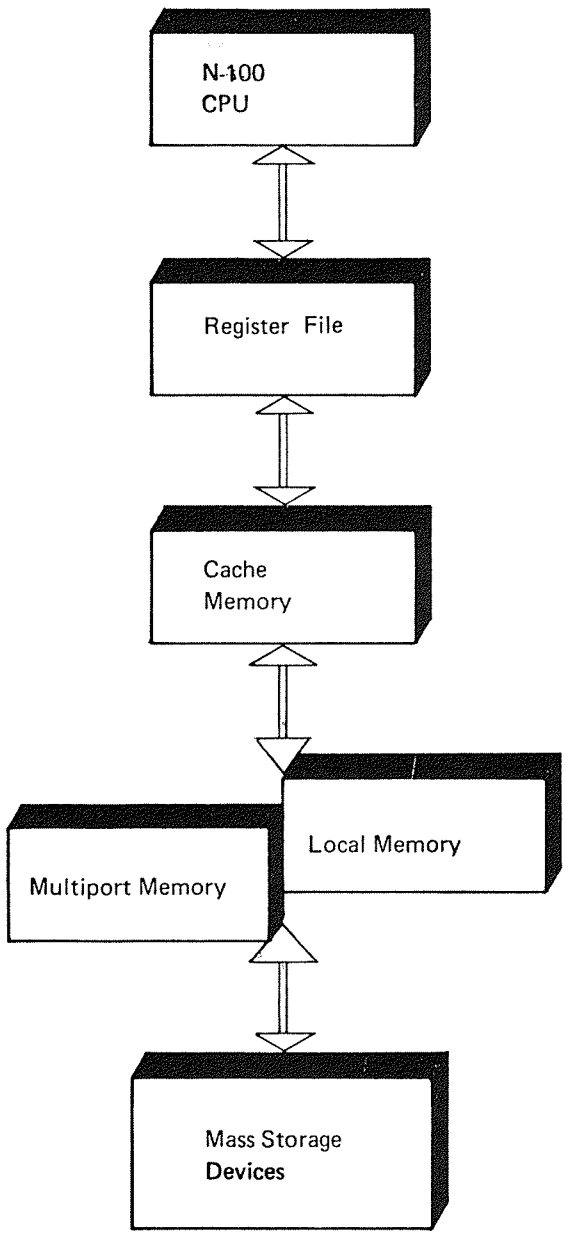


Figure V.3.1: Multilevel Storage System

The main idea 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 first level in the storage system is the register file, holding 128 programmable registers. Since the register file is implemented on the CPU board with direct communication with the IDB, a very short access time is achieved.

–

Cache memory is located on the memory management module and is directly connected to the IDB over the B connector. It is a selective high speed bipolar memory of 1 K words, dynamically updated to hold the most recent data and instructions to be processed. The cache memory will reduce average memory access time significantly and will not introduce any overhead in the system because it works in parallel with the micro program.

– Local (main) memory is located in the same card crate as the CPU and may have any size from 32 K words (64 K bytes) to 16 M words (32 M bytes) in steps of 32 K words. each module may contain a maximum 64 K words (128 K bytes). Each word in local memory is stored with a 6 bit error correction code which makes it possible to:

  – correct and report single bit errors
  – detect and report all double bit errors and most multiple bit errors

– On the same level as local memory, a "Big multiport" memory (refer to the NORD-10/S Reference manual) may be installed. Multiport memory is accessed through a multiport memory transceiver in the NORD-100 bus, connected to one port in a separate card crate. The multiport memory has four ports, allowing four sources to access the same physical memory area. Maximum address space for the multiport memory system is 2 M words (4 M bytes).

– The next level of the NORD-100 storage system consists of mass storage devices. Some of the information to be processed is seldom required and does not have to be "in memory" at all times. This information can be stored on magnetic tape, disk packs or floppy disks in a data library and available if mounted on a disk unit.

  Large amounts of information can be stored here at a low storage price. Prior to usage of information stored on mass storage devices, the information must be transferred to a higher memory level (local or multiport memory). Software overhead and rather long access time must be accepted in connection with such a data transport. Information to and from mass storage devices goes through the input/output system and usually over a direct memory access channel (DMA).

All memory modules in the NORD-100 system have asynchronous timing relative to the CPU. That is, several handshaking signals must be exchanged between the CPU and the memory system during the transfer. Memory modules with different speeds may also be mixed.

V.4 MOS MEMORY OPERATING PRINCIPLES

V.4.1 *GENERAL*

The read/write memory used in NORD-100 is random access MOS memory (RAM). Principally, two classed of RAM exist:

- static
- dynamic

- Static RAMs store each bit of information in a flip-flop, and this information is retained as long as power is supplied to the circuit.

  The static RAM is very fast memory. Therefore, static RAMs are used in the NORD-100 cache memory to have a very fast access time to this part of the memory system.

- Dynamic RAMs are devices in which the information is stored in the form of electric charges on the gate-to-substrate capacitance of a MOS transistor. This charge dissipates in a few milliseconds, and the element must be refreshed, i.e., capacitance recharged periodically. In dynamic RAMs fewer elements are involved in storing one bit of information, so that more bits can be packed into a given physical area. The ratio is 4 to 1 compared to the static RAM. Due to the higher density, they turn out to be more suitable for memory sizes over a given limit. They also consume less power than static RAMs in the quiescent state.

  The drawback, however, is the necesary refresh cycle that requires additional internal and external circuitry.

  Dynamic RAMs are used in the local (main) memory in the NORD-100 and only this type will be discussed in the following.

V.4.2 *A MEMORY CELL*

A typical three transistor dynamic RAM cell is depicted in Figure V.4.1.

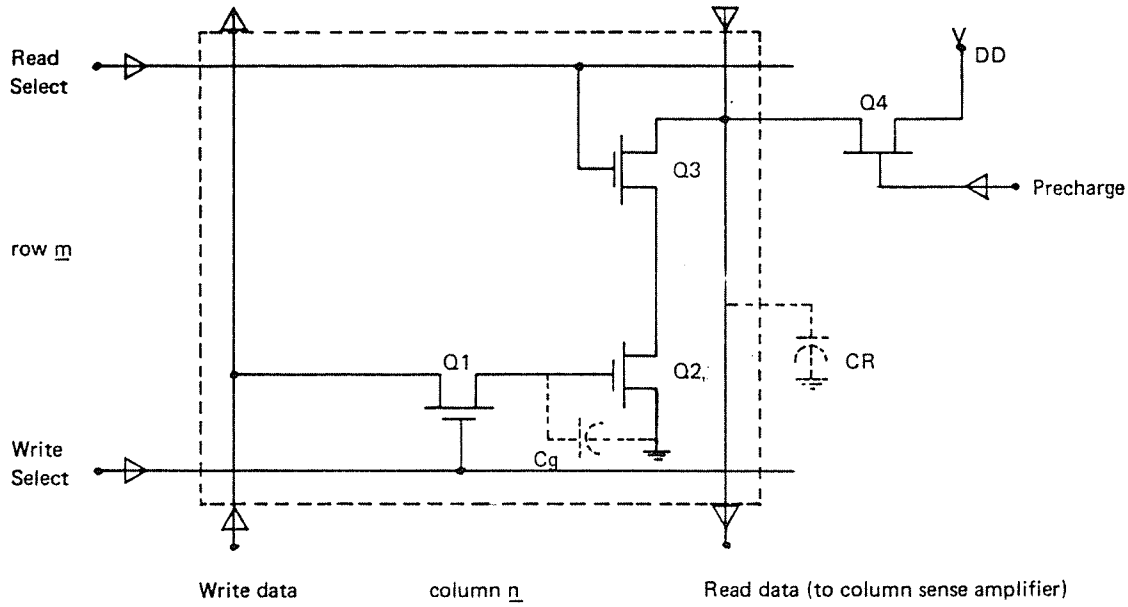


Figure V.4.1: A Memory Cell

In addition to the three transistors (Q1 - Q3) which are required to implement a storage cell, a fourth transistor (Q4) is needed to precharge the output capacitor (CR). CR is implemented by the parasitic capacitance of the "Read Data" column line. The basic memory cell consists of a capacitor, Cg, formed by the gate-to-substrate capacitance of Q2.

If a logical "one" is stored in the cell, Cg is charged and Q2 will be held in its conducting state. If a logical "zero" is stored, Cg is discharged and Q2 will remain in its off state.

Write Operation

A write operation is accomplished by applying a high ("1") or low ("0") level on the "write data" line and pulsing the "write select" line. Q1 will be turned on, charging Cg to the level of the "write data" line. A logical "1 or "0" has been stored.

Read Operation

A read operation is accomplished in two steps:

1. A precharge is initiated by pulsing the precharge line. Q4 will conduct and charge up CR (represented by parasitic caqpacitance of the column line read data line).

2. The "Read Select" line is then enabled turning Q3 on. If a "1" is stored (Q2 conducting), CR will be discharged. This is sensed by the "sense amplifier" associated with the "Read Data" line.

   If a "0" is stored (Q2 not conducting), CR will now be discharged, interpreted by the sense amplifier as "0".

Refresh Operation

Even though MOS transistors with high input impedence are used, Cg will discharge rather quickly due to the small capacitance. To oppose the discharging effect, the cell will be periodically recharged. This is accomplished by doing a combined read/write operation internally. This is controlled by the "refresh control logic".

V.4.3 *16 K * 1 BIT MEMORY CHIP*

A memory chip is built up around a matrix of elements as previously described. For a 16 K chip the matrix consists of 128 rows and 128 columns (giving 16,384 cells).

One pair of "Read Select" and "Write Select" makes up one row, while one pair of "Write Data" and "Read Data" makes up a column. Refer to Figure V.4.2.

Due to pin limitation, the 14 bit address required to decode one of 16,384 cells is multiplexed into the input address buffer.

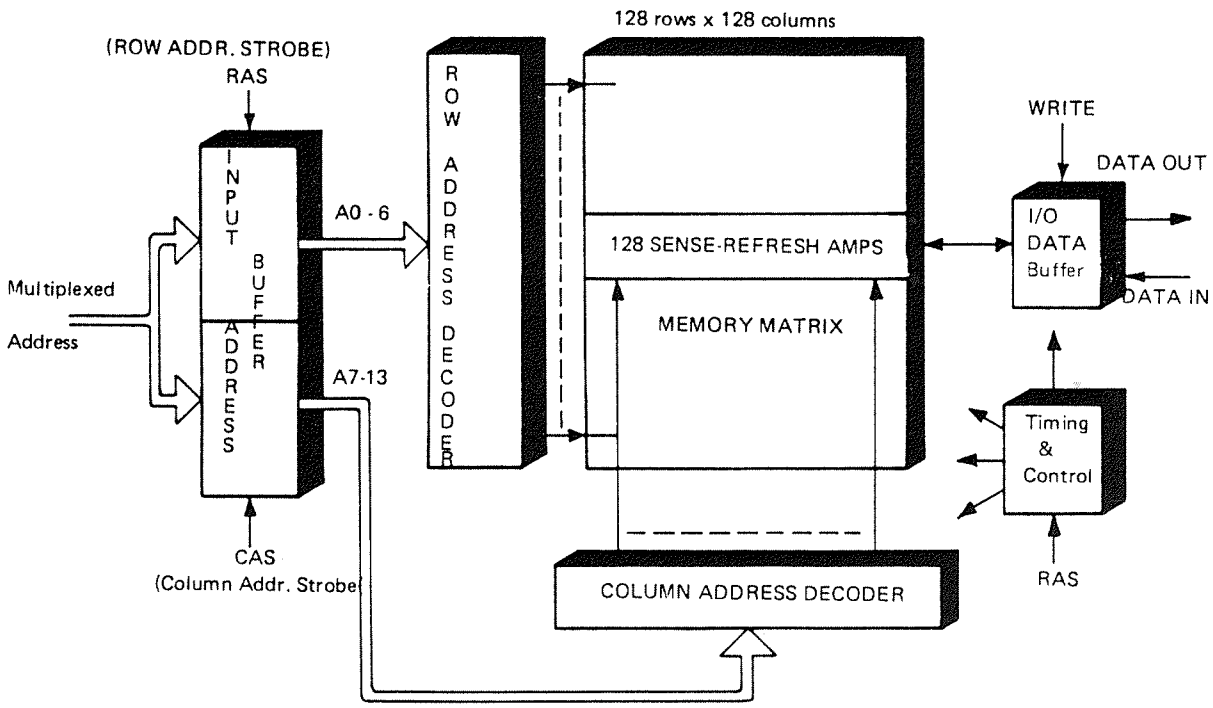


Figure V.4.2: A Memory Chip — Functional Blocks

V.4.3.1 *Functional Operation*

Row address strobe (RAS) will initiate a memory cycle, which will start a series of internal clocks for the following sequence. The row address is then decoded for selection of the proper row in the memory cell matrix. All transistors in that row become conductive, transferring charges from their respective sense lines, destructively reading data.

Each column has its own sense amplifier whose function is to detect this charge from the sense lines and to amplify the signals caused by this charge.

Column address strobe (CAS) will strobe the 7 bit column address into the column address decoder. This address will point to one of the 128 sense amplifiers, which will then be read or written depending on the type of cycle.

The amplified signals from the sense amplifiers are feed back to their respective cells, refreshing the voltage levels in the cells.

V.4.3.2 *Read Cycle*

Refer to the timing diagram, Figure V.4.3.

1. A seven bit row address is strobed into the input address buffer by a RAS signal.

2. The row address is decoded, selecting the proper row in the memory cell matrix.

3. All 128 columns in the selected row will be read to the sense amplifiers for that row, latched and refreshed by restoring.

4. CAS (column address strobe) will strobe the seven bit column address into the input address.

5. The column address is decoded, selecting the proper column, which will be coupled to the I/O stage.

6. Data is transferred form the selected sense line to the I/O data buffer.

One bit of data has now been read.

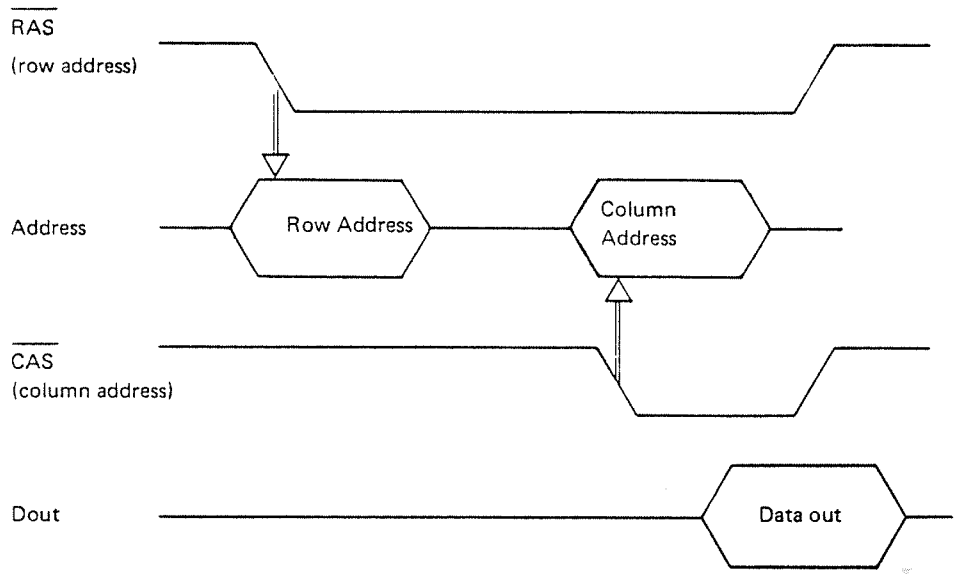


Figure V.4.3: Read Cycle

V.4.3.3 *Write Cycle*

Refer to the timing diagram, Figure V.4.4.

1. Same sequence as read cycle, steps 1 to 5, except that the write signal is activated.

2. Data from the I/O data buffer will be forced directly into both the sense amplifiers and the selected cell by CAS.

One bit of data has now been stored.

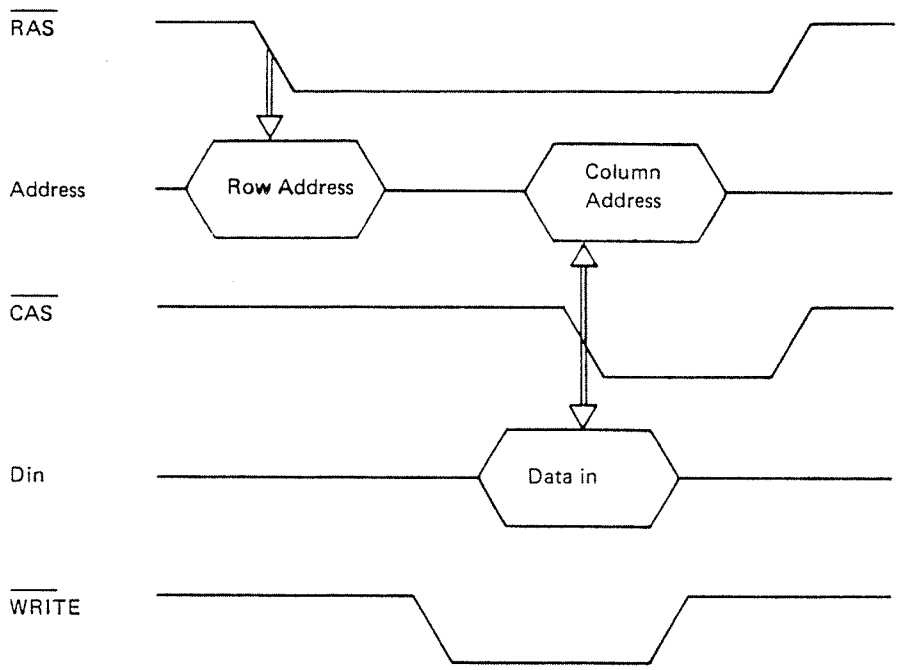


Figure V.4.4: Write Cycle

V.4.3.4 *Refresh Cycle*

Refer to the timing diagram, Figure V.4.5.

Same sequence as read cycle steps 1 to 3. All 128 cell locations for the selected row are latched into the sense amplifiers, which in turn restore the same data back into the cells. Refresh of the dynamic cell matrix is accomplished by performing a memory cycle at each of the 128 row addresses with a 2 millisecond time interval.

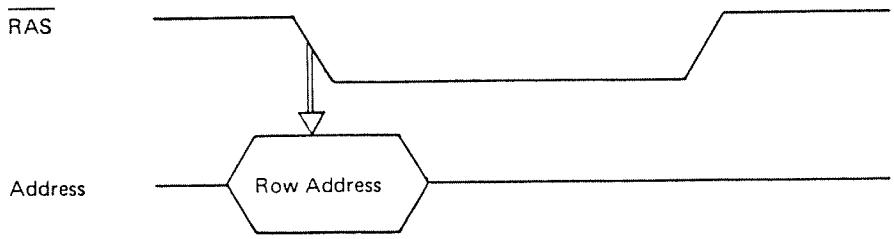


Figure V.4.5: Refresh Cycle

V.5 CACHE MEMORY

V.5.1 *GENERAL*

The part of the memory hierarchy which is located closest to the CPU is the cache memory. Physically, it is located on the memory management module. The memory management module is directly connected to the internal data bus (IDB) in the CPU, which is a condition for fastest access time in cache. Cache memory is placed between the CPU and local memory. The purpose of the cache memory is to hold the most recent data and instructions to be processed, reducing average memory access time significantly.

Cache is a high speed bipolar memory of 150 ns cycle time. The access time for cache memory is virtually zero because it works in parallel with the microprogram.

The cache memory is optional.

V.5.2 *CACHE MEMORY ARCHITECTURE*

V.5.2.1 *Type*

The cache memory should hold the most recently used data and instructions. An associative cache memory is chosen because of its simplicity, low cost and reliability. An associative memory is divided into two parts, data and directory.

V.5.2.2 *Size*

The decrease in memory access time is achieved by keeping copies of the most recently, and thus most frequently, referenced memory words in cache memory.

The size is therefore mainly dependent upon program loop sizes. In other words, the hit ratio one wishes to achieve will determine the size. Hit ratio is defined as the ratio between the number of read references from cache and the total number of read references over a given time.

Figure V.5.1 shows the cache size versus hit ratio for a given program.

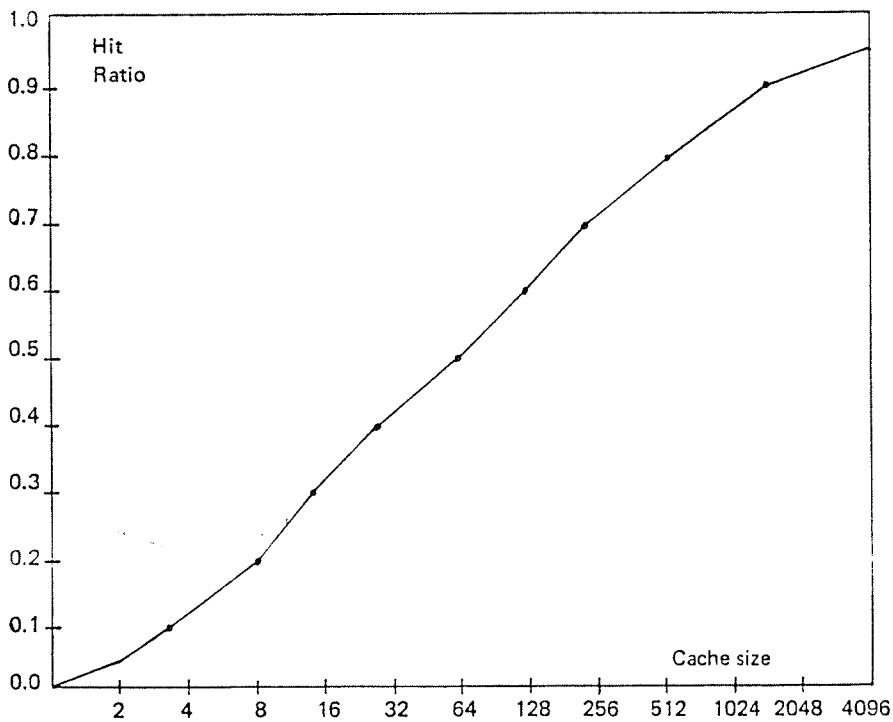


Figure V.5.1: Hit Ratio versus Cache Size

It should be noted that the curve will change, depending on how well a given program is structured and on the program type. However, as is shown in the section on the cache memory organization, it is most practical to use a cache memory of the same size as the page size (i.e., 1 K words).

V.5.2.3 *Placement/Replacement Algorithm*

The algorithm used in the cache memory is called "Write Through" (WT). The algorithm 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 the CPU and also into the cache memory (for probable later use).

The advantages of this algorithm are:

- easy to implement
- requires little hardware
- more reliable in case of power break (main memory is always kept updated)

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.

V.5.2.4 *Program Start*

When a new program starts, no information belonging to this program is kept in cache. The hit ratio is therefore low. As the instructions and data are accessed once more (loop, etc.) the hit ratio increases. The approximate effect of cache memory is also illustrated in Figure V.5.2 where the absissa illustrates the number of memory references.

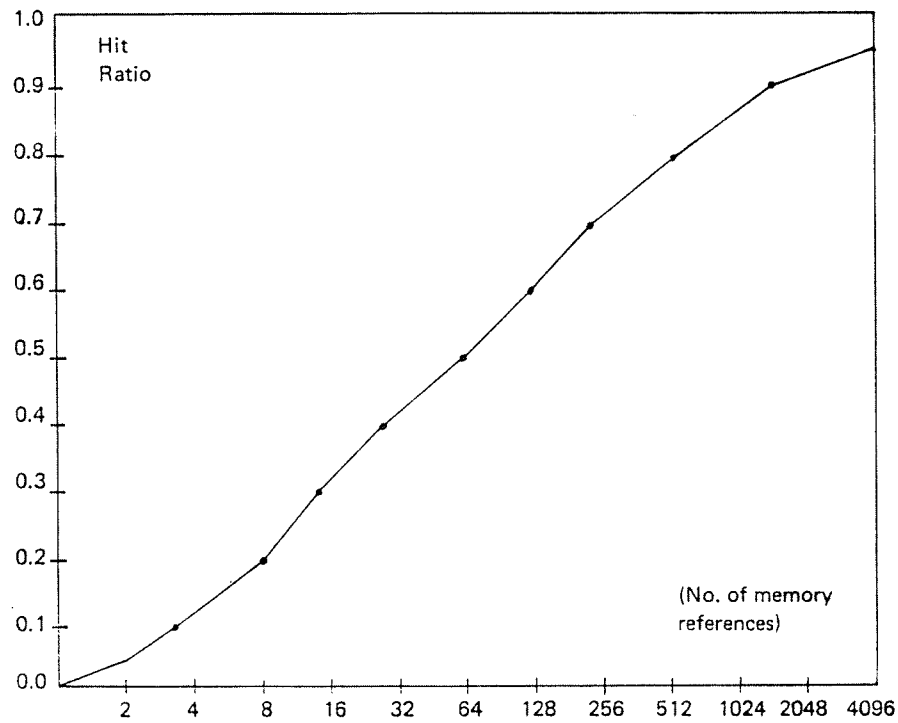


Figure V.5.2: Hit Ratio versus Number of Memory References

V.5.3 *CACHE MEMORY ORGANIZATION*

The cache memory is homogenous, i.e., the cache memory does not discriminate between data words, instructions or indirect address stored in main memory.

The cache memory is organized as a 1 K by 31 bit lookup table, handling both the normal and the extended mode. See Figure V.5.3. 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 (DIP).

In order to link each cache word with the physical page, a directory is used. The directory is 14 bits (9 for normal mode) telling which pages (PPN) the word belongs to.

During write, the directory is updated with the physical page number (PPN) accessed simultaneously with the writing of data. The used bit is set, showing that this location contains valid information.

During read, the directory is compared with the current accessed PPN. If they are equal and the data word is valid (the used bit is set), cache data is sent to the CPU. If they are not equal, the cache word belongs to another page than the one accessed and a request to main memory is made.

The address used to look up in the cache is address bits 0-9, "displacement within page" (DIP). Data from different pages with the same displacement will therefore compete for the same cache location. For example, the content of address 17, 2017 and 14017 will all be put in cache address 17. The content of directory location 17 will be 0, 1 and 6 respectively.

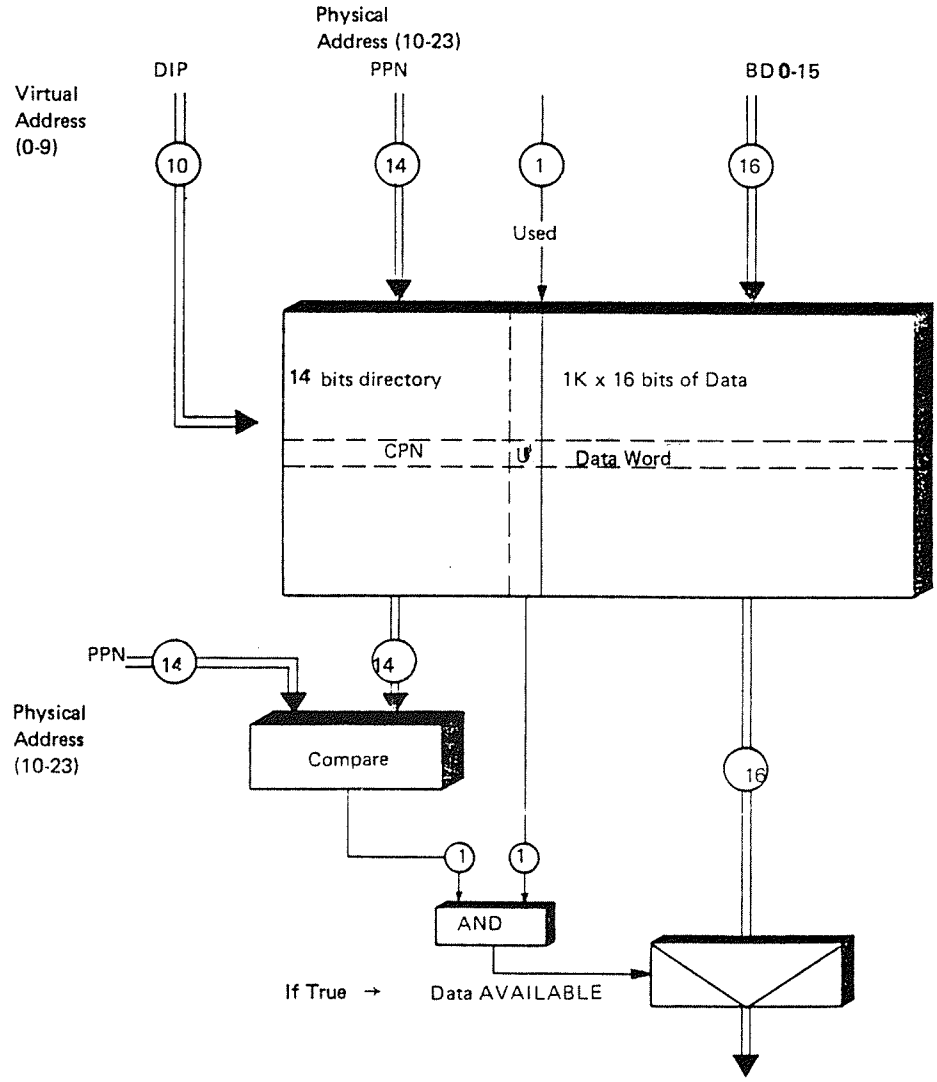


Figure V.5.3: Cache Operation Principles

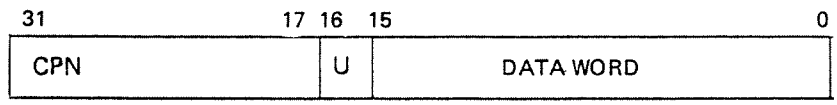


Figure V.5.4: Format of One Cache Word

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.

V.5.4 *CACHE MEMORY ACCESS*

V.5.4.1 *Cache Addressing*

The cache is addressed by DIP, which means that all physical pages with the same DIP will share one location in cache. This cache location will belong to the PPN most recently accessed with this particular DIP. The CPN indicates which PPN the associated data word belongs to. Refer to Figure V.5.5.

When the address arithmetic in the CPU has calculated the virtual address (16 bits) it is sent to the memory management modules internal data bus.

The cache look up is done in parallel with the page table look up. The PPN is then available at the same time as the CPN and the two values are compared. If they match, the data is found in cache. The memory request which includes controls signals and address to the NORD-100 bus, is then inhibited.

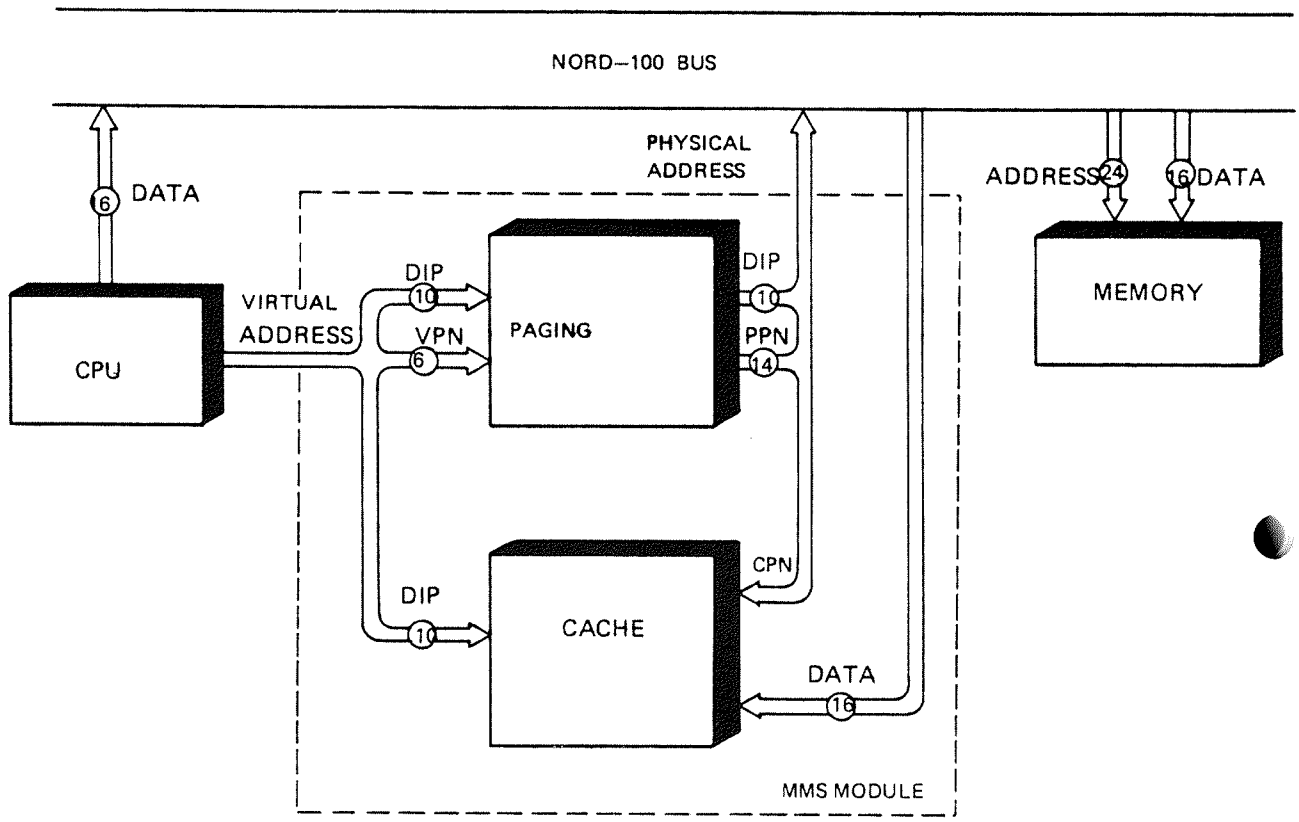


Figure V.5.5: Cache Memory, Write Access

V.5.4.2 *Write Access*

Refer to Figure V.5.5.

A memory request always starts with an address cycle. The address is calculated and the corresponding control signals are generated in the bus control on the CPU module. When the memory management system is on (PON) the physical address is calculated there. The word is then written in main memory. In parallel with the main memory access, a copy is written into the cache memory along with its corresponding PPN, and the user bit (U) is set. In this manner, the main memory will always contin relevant and correct information. This is of special importance in case of power failure.

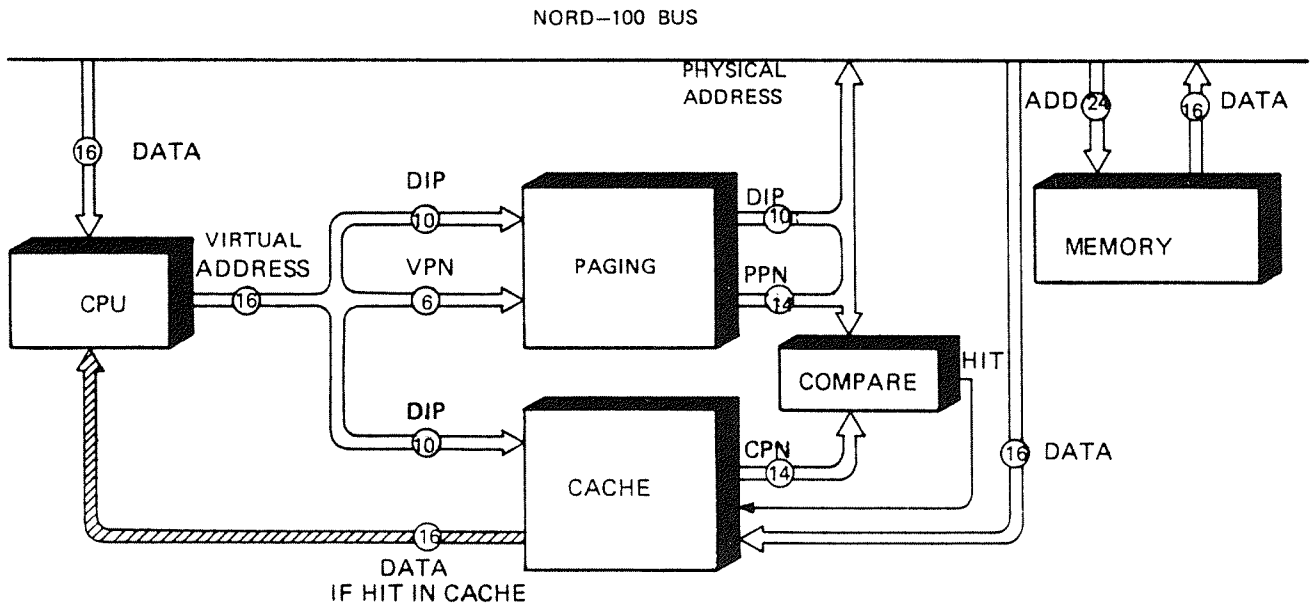


Figure V.5.6: Cache Memory, Read Access

V.5.4.3 *Read Access*

Refer to Figure V.5.6.

During a read operation, information may or may not be found in the cache memory.

The virtual address, calculated in the start of the memory cycle, is present to the memory management system and the cache memory at the same time. Displacement in page (DIP), bit 0-9 of the virtual address, is the address to the cache memory. The access time in the page tables is the same as the access time in the cache memory (50 ns), and consequently the PPN and CPN are available at the same time. If PPN = CPN and the used bit is true (HIT), information is available from the cache memory. It is sent directly to the IDB on the CPU. The main memory request is then inhibited.

If PPN ≠ CPN or the used bit is false, the information is not present in the cache memory, and a normal memory cycle is initiated. When data to the CPU from main memory is ready on the NORD-100 bus it is also written into the data part of cache memory, the PPN will be written into the directory part (CPN = PPN), and the used bit is set equal to one. When writing into cache the old information is simply overwritten.

V.5.5 *CACHE CONTROL AND STATUS*

Cache memory contains:

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

V.5.5.1 *Cache Control*

The operating system may perform two actions to control the cache memory:

- clear cache
- setting of the cache inhibit limit registers

Clearing Cache

NORD-100 cache concept requires that all changes in main memory should be updated in cache. This is done automtically 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. Because of this, the operating system will exeucte the instruction

```
TRR  10          % Clear cache
```

when a DMA transfer is completed. This will clear all used bits in the cache memory. After the TRR 10, the cache memory will be disabled for 60μs while the used bits are being cleared. This is also done in the initialization procedure during start of SINTRAN III.

Setting of Cache Inhibit Limit Registers

The cache memory system contains two 14 bit registers, lower cache inhibit limit register (LCIL) and upper cache inhibit limit register (UCIL). By proper setting of the cache inhibit register, a selected area may be excepted from cache, i.e., data in this area will not be copied into the cache when accessed. The inhibited area includes all pages with

lower limit $\leqslant$ PPN $\leqslant$ upper limit

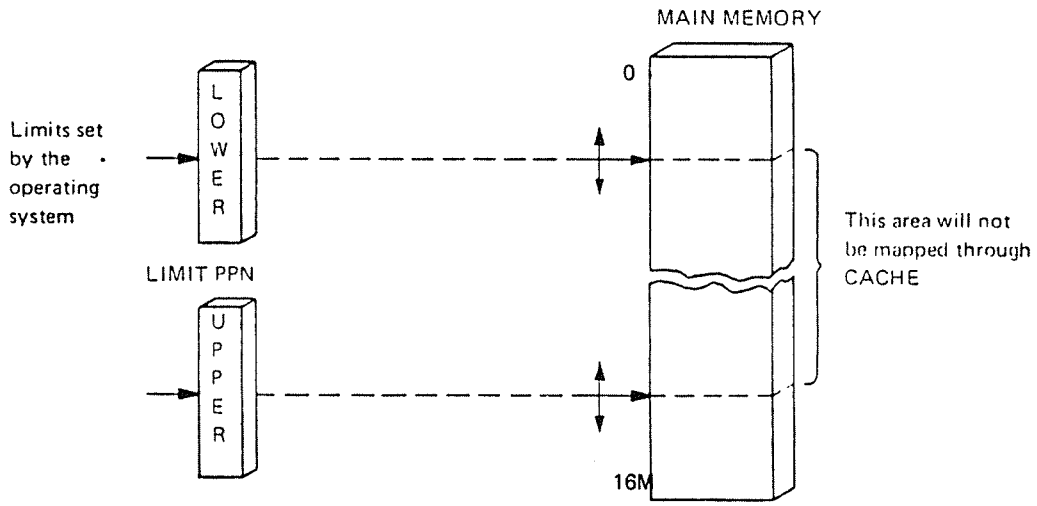


Figure V.5.6: Cache Inhibit Limits

The limit setting is used 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 no.
TRR 11               % set lower limit
```

and

```
LDA<upper limit>     % upper limit page no.
TRR 12               % set upper limit
```

The inhibit feature is intended for use on memory areas that are operated upon by DMA transfers and/or other processors, to ensure that the CPU does not operate upon unvalid data that might reside in cache.

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 accompanied by clear cache. Note that the whole address range is inhibited after master clear.

V.5.5.2 *Cache Status*

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

TRA 10 % 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 instruction 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 60μs 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 1 X X if the cache option is not installed.

V.6 LOCAL MEMORY

V.6.1 *GENERAL*

The next level in the storage hierarchy is the local memory.

Local memory facts:

- Memory size from 32 K words in steps of 32 K words up to 512 K words (normal address mode), 16 M words (extended address mode).
- Maximum 64 K words per memory module
- Direct connection to NORD-100 bus for low access time
- Error correction of single bit failures and detection of double bit failures

Local memory may consist of several modules plugged directly into the NORD-100 bus. All communication with memory is performed over this bus. The NORD-100 bus connects the memory to CPU as well as interfaces for direct memory access (DMA) communication.

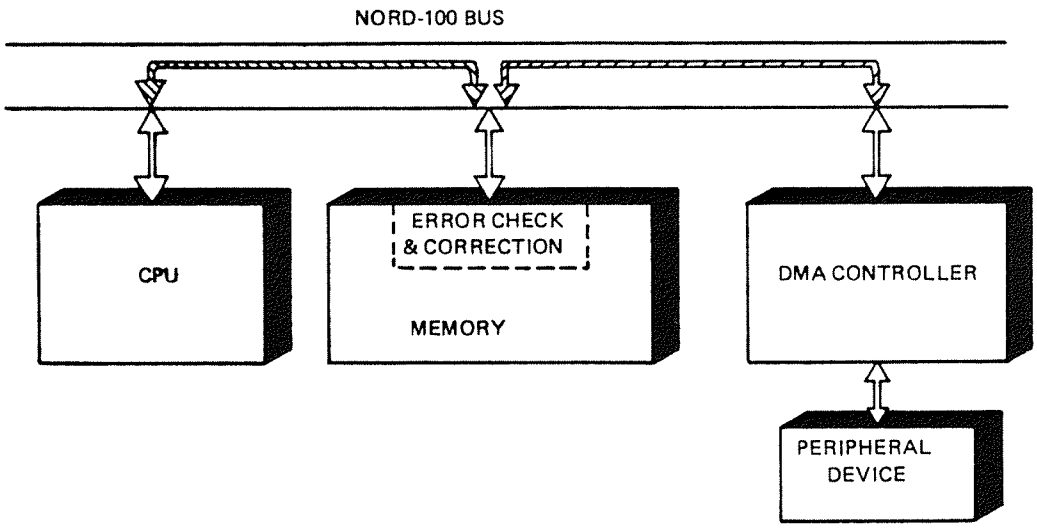


Figure V.6.1: Memory Interconnection

The error checking and correction network is located on each memory module.

V.6.2 *MEMORY MODULE PLACEMENT*

Memory modules should be placed from the rear position (position 12) in the NORD-100 bus and expanded towards the CPU module.

Module address range may be defined in two different ways:

— prewired positon code in each bus slot
— thumbwheel setting of a module address area

V.6.2.1 *The Position Code*

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

V.6.2.2 *The Thumbwheel Setting*

It is possible to mix module sizes from 16 K words and upwards in the same memory system. In this case, the position code cannot be used. The thumbwheel setting allows an address resolution of 16 K per position and should be used in cases where module sizes are mixed.

The thumbwheels are physically located on the top of the module, and define the lower address limit for the module in 16 K units. The module itself knows its size, which is added to the lower limit, and is presented on a display showing lower limit for the next module. The lower address thumbwheel must be set manually for each module.

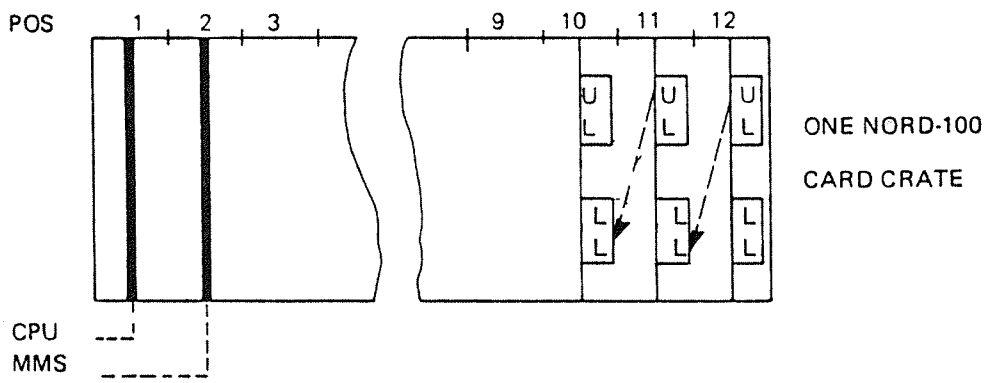


Figure V.6.2: The Memory Module Placement in the Card Crate

LL: Lower limit is set by two hexadecimal thumbwheels (value from 0 - 15) or given by module placement (the position code). Lower limit defines the lowest address to access the module, in 16 K units. Only setting 0 - 8 are legal.

UL: Upper limit is displayed in 16 K units as two octal digits and defines the highest address to access the memory module. The upper limit is geneated internally on the memory module as a sum of the lower limit and the size of the memory module.

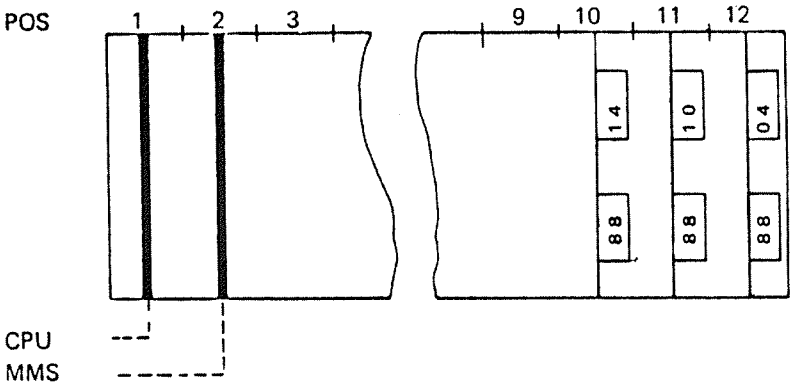
As indicated in the above figure, the upper limit on a memory module covering one part of the address range, should be equal to the lower limit on the next memory module covering the succeeding addresses.

Each position in the slot has a position code. By setting the lower limit to 88, the position code is used by the memory module to determine the upper limit. Only modules of 64 K must then be used. See case 1.

CASE 1:

Lower limit defined by the position code.

Thumbwheel should be 

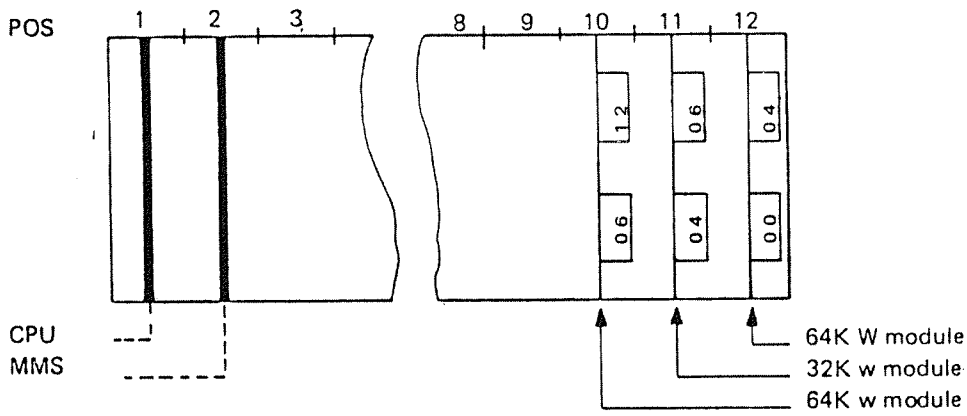


Position 12: Address range 0 - 64 K words
Position 11: Address range 64 - 128 K words
Position 10: Address range 128 - 192 K words
and so on

Requirement: All memory modules must be 64 K words.

CASE 2:

Lower limit defined by thumbwheel.



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

V.6.3 *ADDRESSING*

In order to store or read information, control signals must be established together with an address to tell *where* to read or write.

The 16 bit virtual address is generated by the address arithmetic located in the CPU. The physical address (24 bit) is sent out either from the CPU or from the memory management system if present, together with control signals. The control signals are described in the memory access chapter.

All memory modules connected to the NORD-100 bus "listen" to the address on the bus. The module containing the given address will answer. The address is decoded in the following way.

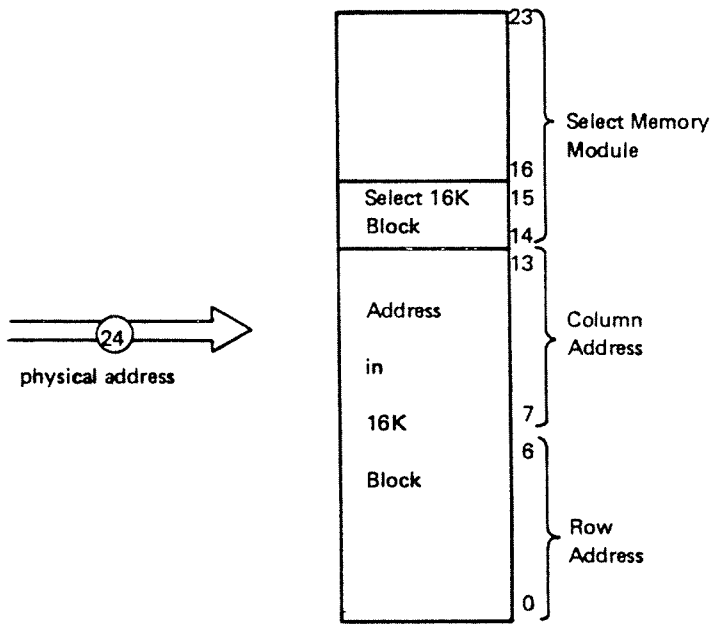


Figure V.6.3: Address Decoding

Bits 14 - 23 of the physical address select the memory module. Each of the memory modules consist of four blocks of 16 K words each, and bits 14 and 15 select one of these four blocks. The remaining 14 bits is the address in the selected 16 K block. Bits 0 - 6 are used as row address, and bits 7 - 13 are used as column address (refer to Section V.4.3).

The physical address range for the different modules is determined by the thumbwheel setting and the size of each module, or the postion in the card crate.

Which of the 16 K blocks is accessed (0 - 16 K, 16 - 32 K, 32 — 48 K, 48 - 64 K), is shown by 4 yellow lamps on the edge of the memory modules. An access on block 0 (0 - 16 K) is shown by light in the yellow lamp labelled 0 - 16, block 1 lights the yellow lamp labelled 16 - 32, etc.

V.6.4 *DATA*

The NORD-100 bus is 24 bits wide, but data to/form memory is only 16 bits. On the memory module, a 6 bits error correction code is attached to the 16 data bits, for increased storage reliability and 22 bit words are stored in memory. Refer to Figure V.6.4.

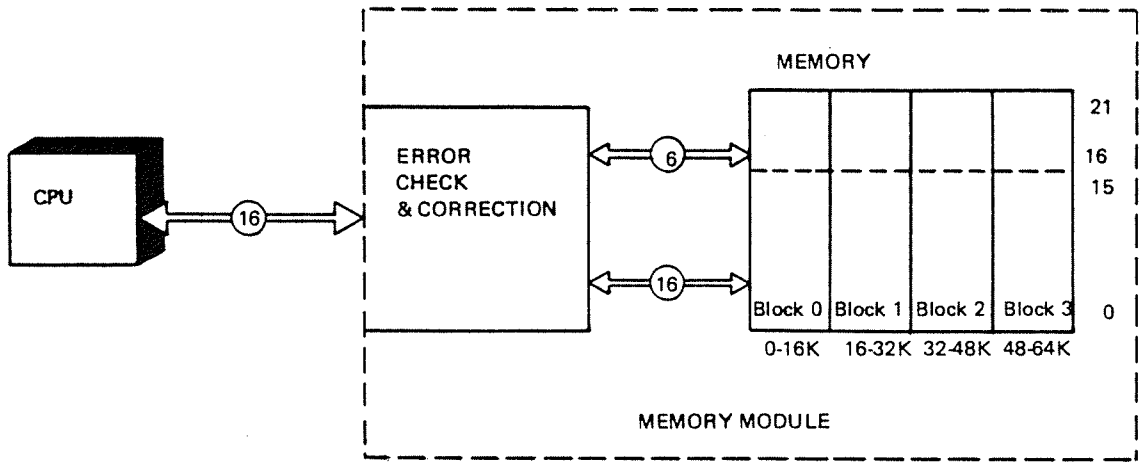


Figure V.6.4: Error Correction Memory

The error correction network will be described in detail later.

V.6.5 *MEMORY ACCESS*

Local memory operates asynchronously with the CPU and each memory module has its own timing.

Local memory is available to three sources with the following priority:

1. Refresh — Refresh circuitry
2. DMA — Direct Memory Access controller
3. CPU — Central Processor

These sources again operate asynchronously with respect to each other.

A priority and allocation circuitry is therefore required. This is found in bus control logic located on the CPU module (refer to Chapter IV). The different control signals which must be established for proper operation are generated in the bus control.

Figure V.6.5 illustrates the control signals on the NORD-100 bus involved in a data transfer between memory and CPU.

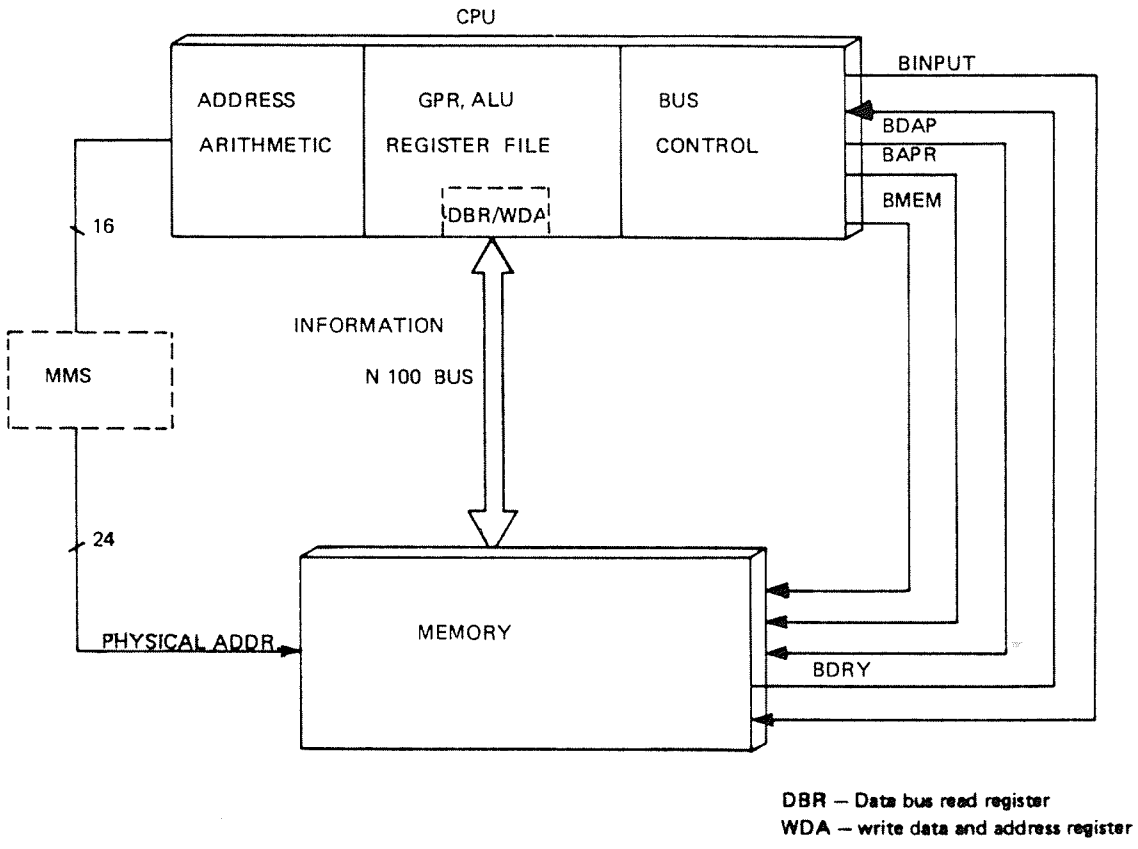


Figure V.6.5: CPU — Memory Operation

Memory Read

A memory reference starts with a "memory request" to the bus control, which generates the "BMEM" signal to memory. This signal tells that a memory cycle is in progress. Below is described a CPU access. However, a DMA access is quite similar. After BMEM BINPUT is made false to indicate read from memory, the physical address is sent out on the NORD-100 bus from the CPU, calculated by the address arithmetic, together with the bus address present (BAPR) signal. If a memory management module is present, the virtual address from the CPU will be converted to a 24 bit physical address and then sent to the NORD-100 bus, accompanied by the BAPR signal.

BAPR tells the memory modules that the address is present on the bus. The address bits 14 - 23 is checked against the limits on all modules and one is selected. On the selected one the address will be further decoded to block select and to row and column address.

After a short delay, data is ready to be sent from memory, and will be enabled onto the NORD-100 bus. The CPU allows the memory modules to enable data to the bus by the bus data present (BDAP) signal.

When data is put on the NORD-100 bus, the memory gives the bus data ready (BDRY) signal to the CPU, telling that data is ready on the bus. This signal will storbe the data into the data bus read register (DBR) in the CPU. The data will be enabled onto the IDB, where it is available for all parts connected, like the register file, GPR, ALU, etc.

The time from BADR to BDRY is 320 ns and is defined as memory access time. If an error is detected, 40 ns are added to allow correction.

Memory Write

The write transfer is similar to read and the same control signals apply. The bus is requested and "BMEM" is sent out together with the "BINPUT" signal true, telling that this is an input to memory. The address is sent out on the bus, with "BAPR" as strobe signal. Data to transferred is enabled from the write data/address (WDA) register onto the bus. The bus control issues the "BDAP" signal, telling that data is on the NORD-100 bus. Memory latches the data present on bus into the module with this signal. "BDRY" is sent from memory to tell the CPU that data is accepted by the memory module.

The time from BAPR to BDRY, the memory access time, is 200 ns for a write operation.

V.6.6 *REFRESH*

Refer to Figure V.6.6.

As previously mentioned, dynamic MOS memory must be refreshed periodically. An oscillator located on the bus control in the CPU gives a request every 13 μsec. for a refresh cycle. Refresh cycles have highest priority in the bus control and a refresh signal "BREF" are sent on the NORD-100 bus to all memory modules.

On the memory modules, a seven bit row address counter is activated by the "BREF" signal. The counter is incremented each time a "refresh request" occurs, and given to the memory chips as row address. All columns of that row will be refreshed and written back in one operation (refer to Chapter V.4). Since a 16 K chip has 128 rows (2^7), a seven bit row address counter is sufficient to address all rows.

A "BDRY" signal from the memory terminates the refresh cycle.

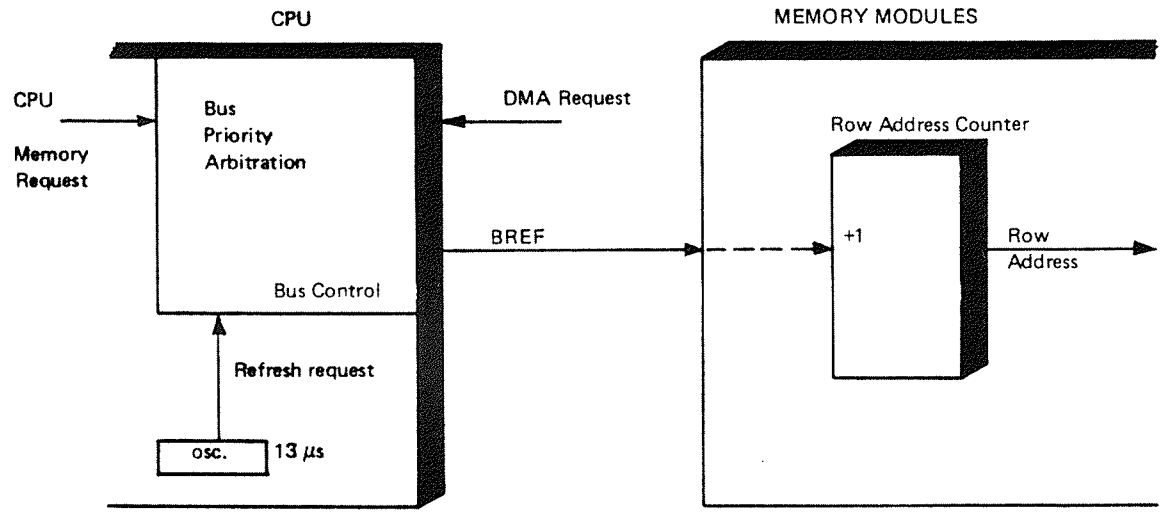


Figure V.6.6: Refresh

V.7 MEMORY ERROR CHECK AND CORRECTION

V.7.1 *GENERAL*

In today's systems, the trend is that the memory is swelling, and higher data reliability becomes more urgent. The increased demand for more reliable data storage systems has led to development of more sophisticated error detection and correction schemes, so called error correcting coding (ECC) systems.

Statistically independent single bit errors represent about 99% of all memory errors. A greath part of those errors are random intermittent errors caused by noise injection, and over a long period of time the same identical error will not occur (i.e., "soft errors").

By using 6 bit error correction codes on 16 bit data, as in NORD-100, the mean time between failure (MTBF) for the memory boards may be increased by several orders of magnitude.

The ECC used in NORD-100's memory system is related to a modified Hamming code, working on 16 bits in parallel.

The big advantages of error correction are:

- More reliable data
- Great improvement in MTBF
- Solid chip failure can be corrected at a convenient time (the memory board is replaced when serviced)

This can be implemented with the cost of:

- More memory chips
- Introducing overhead to the memory access
- More hardware in the check/correct circuits

However, great effort have been made in reducing check time to a minimum. The overhead is thus about the same as the more traditional parity circuits.

The 6 correction bits are parities formed by the data bits in a unique way. Six parity bits may be combined in 64 ($=2^6$) ways. To be useful, the parity bits must be formed in such a manner that *only one* combination means NO ERROR = GOOD and the remaining 63 mean some kind of error.

In NORD-100 local memory, the following situations are handled:

- accept good data (no errors)
- detect, correct and report single bit errors
- detect double bit errors and interrupt the CPU for uncorrectable memory failure

- detect and interrupt the CPU for memory failures on multiple bits (on the average 65% of all multiple errors are detected)

V.7.2. *FUNCTIONAL DESCRIPTION*

Since so many people seem to be interested in the subject of error correction it will be covered in some detail.

V.7.2.1 *Parity Checking*

As error correction is a sophisticated form of parity checking, that concept is first described.

The parity of a group of signals — say 0 - 3 — is the exclusive OR of all signals.

$P = 0 \oplus 1 \oplus 2 \oplus 3$

Hence, P tells if there is an even or an odd number of ONEs in the data pattern.

It is also obvious that if one bit changes the value of P changes.

It is quite immaterial what the original value of the bit was as we are only concerned about *changes.*

Example:

$\overline{0} \oplus 1 \oplus 2 \oplus 3 = \overline{0 \oplus 1 \oplus 2 \oplus 3} = \overline{P}$

However, change of *two* bits leave P unchanged:

$\overline{0} \oplus \overline{1} \oplus 2 \oplus 3 = 0 \oplus 1 \oplus 2 \oplus 3 = P$

This follows readily from the table of exclusive OR.

R	W	R ⊕ W
0	0	0
0	1	1
1	0	1
1	1	0

Since we want to detect change in data bits during stopover in memory, parity checking is suited for that purpose.

When the data pattern is written into memory the parity Pw is formed in a so-called "parity tree" and stored in an extra memory cell together with the data. The value at write time is:

EQ7.1 $P_W = 0_W \oplus 1_W \oplus 2_W \oplus 3_W$

When the data is read back the following parity is formed:

EQ7.2 $S = 0_R \oplus 1_R \oplus 2_R \oplus 3_R \oplus P_R$

where R indicates the read value of previously written data and parity bit. Now insert the identity:

$P_R \equiv P_R \oplus P_W \oplus P_W$

(because $P_W \oplus P_W = 0$ and $P_R \oplus 0 = P_R$)

EQ7.2 then reads:

EQ7.3 $S = 0_R \oplus 1_R \oplus 2_R \oplus 3_R \oplus P_W \oplus (P_R \oplus P_W)$

Insert the value of P_W from EQ7.1 and obtain:

EQ7.4 $S = (0_R \oplus 0_W) \oplus (1_R \oplus 1_W) \oplus (2_R \oplus 2_W) \oplus (3_R \oplus 3_W) \oplus (P_R \oplus P_W)$

This is exactly the form which enables us to see if there has been a change between read and write.

If none of the bits have chnaged $S = 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 = 0$. If one bit has been changed $S = 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 = 1$ and so on.

Summing up:

S = 0 means 0, 2 or 4 errors
S = 1 means 1, 3 or 5 errors

We are thus able to decide something about the *number* of errors but not to identify a specific bit error. The reason is that all bits contribute in the same manner.

As we shall see, the crux of identifying specific bits is to form several parities and let each bit contribute in a unique manner.

V.7.2.2 *Error Correction*

Each memory module has its own ECC network which is shown simplified in Figure V.7.1.

For the sake of clarity, parity generation and checking are shown separated although they actually have common circuitry.

The following 6 parities serve as correction bits and are generated at each write access:

T16 = 0 ⊕ 1 ⊕ 2 ⊕ 3 ⊕ 4 ⊕ 6 ⊕ 8 ⊕ 10
T17 = 0 ⊕ 2 ⊕ 4 ⊕ 7 ⊕ 9 ⊕ 10 ⊕ 12 ⊕ 14
T18 = 1 ⊕ 2 ⊕ 5 ⊕ 6 ⊕ 7 ⊕ 11 ⊕ 12 ⊕ 15
T19 = 3 ⊕ 4 ⊕ 5 ⊕ 6 ⊕ 7 ⊕ 13 ⊕ 14 ⊕ 15
T20 = 8 ⊕ 9 ⊕ 10 ⊕ 11 ⊕ 12 ⊕ 13 ⊕ 14 ⊕ 15
T21 = 0 ⊕ 1 ⊕ 3 ⊕ 5 ⊕ 8 ⊕ 9 ⊕ 11 ⊕ 13

They are stored together with the 16 data bits (T0-T15) in memory.

When data is read back, the following parities, call syndrome bits* are formed.

S0 = 0 ⊕ 1 ⊕ 2 ⊕ 3 ⊕ 4 ⊕ 6 ⊕ 8 ⊕ 10 ⊕ P16
S1 = 0 ⊕ 2 ⊕ 4 ⊕ 7 ⊕ 9 ⊕ 10 ⊕ 12 ⊕ 14 ⊕ P17
S2 = 1 ⊕ 2 ⊕ 5 ⊕ 6 ⊕ 7 ⊕ 11 ⊕ 12 ⊕ 15 ⊕ P18
S3 = 3 ⊕ 4 ⊕ 5 ⊕ 6 ⊕ 7 ⊕ 13 ⊕ 14 ⊕ 15 ⊕ P19
S4 = 8 ⊕ 9 ⊕ 10 ⊕ 11 ⊕ 12 ⊕ 13 ⊕ 14 ⊕ 15 ⊕ P20
S5 = 0 ⊕ 1 ⊕ 3 ⊕ 5 ⊕ 8 ⊕ 9 ⊕ 11 ⊕ 13 ⊕ P21

PTOT = S0 ⊕ S1 ⊕ S2 ⊕ S3 ⊕ S4 ⊕ S5.

P16 - P20 are read values of the written parities T16 - T20.

S5 is not used in practice, it is only an intermediate value used for construction of PTOT. PTOT is the parameter we are interested in because when the values of S0 - S5 are inserted one obtains:

EQ7.5 PTOT = 0⊕1⊕2⊕3⊕..... 19⊕20⊕21

That is, it is the total parity of all 22 data and correction bits. (Verify for yourself!)

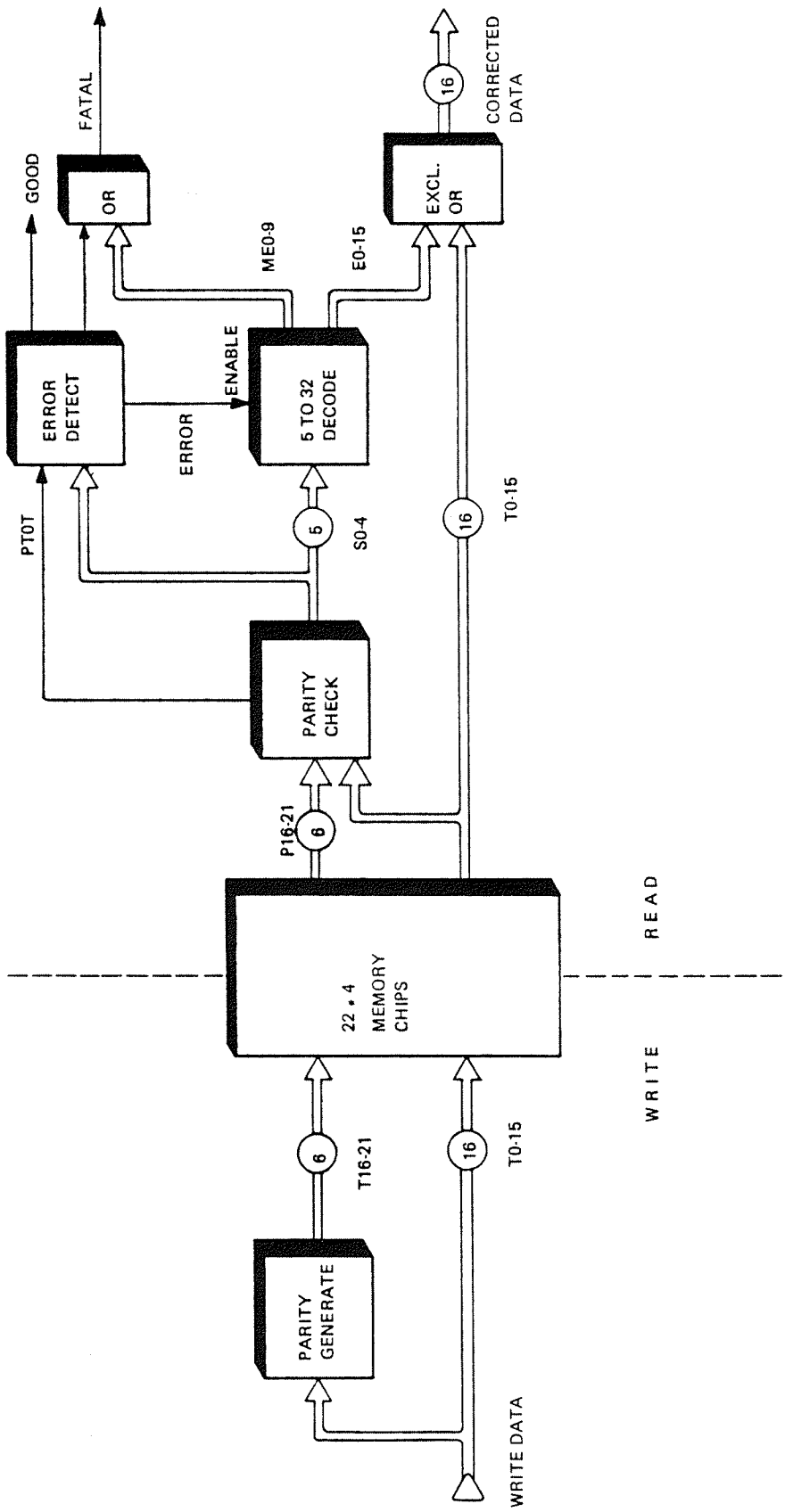
As outlined in the previous section:

PTOT = 0 means 0, 2, 4 .... 22 errors
PTOT = 1 means 1, 3, 5 .... 21 errors

PTOT is therefore exactly what we need to separate between single and double errors.

* The greek word syndrome means a set of symptoms. Individually, they reveal nothing but together they point to one specific disease.

However, first we will look upon identification of single and double errors by means of the 5 syndrome bits S0 - S4.



ND-06.015.01

Figure V.7.1: Error Correction Scheme During Write and Read

If no errors have been introduced during the stay in memory, all parities are unchanged and hence all 5 syndrome bits must be zero. (Refer to equation 7.4 in the previous section.) This combination of all zeros is called GOOD. If GOOD does not show up there must be a single bit or a multiple bit error.

How can these five bits really point out the faulty bit? The reason is that two different data bits never contribute in the same manner in all parities (e.g. bit 2 is a part of S0, S1 and S2). If bit 2 has been changed (inverted) since generation, S0, S1 and S2 will also be inverted. Hence, the error code 00111 is produced instead of the GOOD code 00000 which means no bit change. Reference to Table V.7.1 will show that in fact 00111 is the error code for bit 2. Since error in bit 1 changes S0 and S2, 00101 is the error code for bit 1, etc.

	Error Code	Syndrome Bits Fatal	S4	S3	S2	S1	S0	No Error	Single code Error	Single data Error	
	0	0	0	0	0	0	0	Good			
	1	0	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		
Multiple Errors	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		

Table V.7.1: Error Codes as Reported in the PES Register

It is important to note that only *error in one bit at the time* is assumed.

Since 21 bits can produce more than 21 single error codes + GOOD, the remaining 10 codes must be caused by errors in some combination of two or more bits. These errors cannot be corrected and are therefore fatal to the system.

Even if double bit errors cannot be corrected they may be detected. Since they are fatal it is important to have them reported.

As mentioned, PTOT is used for that purpose. The 32 syndrome codes are divided into 3 groups: GOOD, SING and MULT. The following combinations between them are possible:

PTOT	SYNDROME	ERRORS DETECTED
0	GOOD	0 errors / 3% of 4, 6, 8 ... bit errors
0	SING	100% of 2 bit errors/
0	MULT	97% of 4, 6, 8 ... bit errors
1	GOOD	100% of 1 bit error/
1	SING	65% of 3, 5, 7 ... bit errors
1	MULT	35% of 3, 5, 7 ... bit errors

Unfortunately, multiple errors ⩾3 produces the same code as some good and single error patterns. However, that should be expected because multiple errors must produce *some* code and there are not more than 64 available.

Table V.7.2 shows how many of each code is produced for all possible 1, 2, 3, 4, 5 and 6 bit errors. The distribution between codes is fairly equal. Fortunately, GOOD is only produced by a few 4 and higher order errors.

Even if 65% of triple errors are not detected (they are misinterpreted to be single errors) they are extremely unlikely to occur. In a normal sized memory 100K — 1M words the mean time between triple failure will be of order 100 - 1000 years.

Now turn to Figure V.7.1 again. The section ERROR DETECT check syndrome bits and PTOT according to the mentioned conditions. When errors are revealed a decoder is enabled that decide which bit is in error. In case of a single bit error the faulty bit is inverted by an EXCLUSIVE-OR gate. Since a bit can only be true or false, an erroneous bit must be correct when it is inverted.

PTOT	SYNDROME 4 3 2 1 0	NUMBER OF ERRORS 0	1	2	3	4	5	6	INTERPRETATION
0	0 0 0 0 0	1	0	0	0	252	0	2288	GOOD (NO ERRORS)
0	0 0 0 0 1	0	0	7	0	230	0	2332	EVEN MULT ERROR
0	0 0 0 1 0	0	0	9	0	216	0	2372	EVEN MULT ERROR
0	0 0 0 1 1	0	0	7	0	230	0	2332	EVEN MULT ERROR
0	0 0 1 0 0	0	0	8	0	223	0	2352	EVEN MULT ERROR
0	0 0 1 0 1	0	0	8	0	223	0	2352	EVEN MULT ERROR
0	0 0 1 1 0	0	0	8	0	223	0	2352	EVEN MULT ERROR
0	0 0 1 1 1	0	0	7	0	232	0	2316	EVEN MULT ERROR
0	0 1 0 0 0	0	0	8	0	223	0	2352	EVEN MULT ERROR
0	0 1 0 0 1	0	0	8	0	223	0	2352	EVEN MULT ERROR
0	0 1 0 1 0	0	0	8	0	223	0	2352	EVEN MULT ERROR
0	0 1 0 1 1	0	0	7	0	232	0	2316	EVEN MULT ERROR
0	0 1 1 0 0	0	0	9	0	216	0	2372	EVEN MULT ERROR
0	0 1 1 0 1	0	0	6	0	239	0	2296	EVEN MULT ERROR
0	0 1 1 1 0	0	0	8	0	225	0	2336	EVEN MULT ERROR
0	0 1 1 1 1	0	0	6	0	239	0	2296	EVEN MULT ERROR
0	1 0 0 0 0	0	0	7	0	230	0	2332	EVEN MULT ERROR
0	1 0 0 0 1	0	0	9	0	216	0	2372	EVEN MULT ERROR
0	1 0 0 1 0	0	0	7	0	230	0	2332	EVEN MULT ERROR
0	1 0 0 1 1	0	0	8	0	225	0	2336	EVEN MULT ERROR
0	1 0 1 0 0	0	0	8	0	223	0	2352	EVEN MULT ERROR
0	1 0 1 0 1	0	0	7	0	232	0	2316	ALL BITS EQL ZERO
0	1 0 1 1 0	0	0	7	0	232	0	2316	EVEN MULT ERROR
0	1 0 1 1 1	0	0	7	0	232	0	2316	EVEN MULT ERROR
0	1 1 0 0 0	0	0	8	0	223	0	2352	EVEN MULT ERROR
0	1 1 0 0 1	0	0	7	0	232	0	2316	EVEN MULT ERROR
0	1 1 0 1 0	0	0	7	0	232	0	2316	EVEN MULT ERROR
0	1 1 0 1 1	0	0	7	0	232	0	2316	EVEN MULT ERROR
0	1 1 1 0 0	0	0	6	0	239	0	2296	EVEN MULT ERROR
0	1 1 1 0 1	0	0	8	0	225	0	2336	EVEN MULT ERROR
0	1 1 1 1 0	0	0	6	0	239	0	2296	EVEN MULT ERROR
0	1 1 1 1 1	0	0	8	0	224	0	2345	EVEN MULT ERROR
1	0 0 0 0 0	0	1	0	46	0	828	0	BIT21 SING ERROR
1	0 0 0 0 1	0	1	0	46	0	828	0	BIT16 SING ERROR
1	0 0 0 1 0	0	1	0	46	0	828	0	BIT17 SING ERROR
1	0 0 0 1 1	0	1	0	45	0	837	0	BIT 0 SING ERROR
1	0 0 1 0 0	0	1	0	44	0	844	0	BIT18 SING ERROR
1	0 0 1 0 1	0	1	0	47	0	821	0	BIT 1 SING ERROR
1	0 0 1 1 0	0	0	0	53	0	808	0	ODD MULT ERROR
1	0 0 1 1 1	0	1	0	47	0	821	0	BIT 2 SING ERROR
1	0 1 0 0 0	0	1	0	44	0	844	0	BIT19 SING ERROR
1	0 1 0 0 1	0	1	0	47	0	821	0	BIT 3 SING ERROR
1	0 1 0 1 0	0	0	0	53	0	808	0	ALL BITS EQL ONE
1	0 1 0 1 1	0	1	0	47	0	821	0	BIT 4 SING ERROR
1	0 1 1 0 0	0	1	0	45	0	837	0	BIT 5 SING ERROR
1	0 1 1 0 1	0	1	0	45	0	837	0	BIT 6 SING ERROR
1	0 1 1 1 0	0	1	0	45	0	837	0	BIT 7 SING ERROR
1	0 1 1 1 1	0	0	0	54	0	801	0	ODD MULT ERROR
1	1 0 0 0 0	0	1	0	46	0	828	0	BIT20 SING ERROR
1	1 0 0 0 1	0	1	0	45	0	837	0	BIT 8 SING ERROR
1	1 0 0 1 0	0	1	0	45	0	837	0	BIT 9 SING ERROR
1	1 0 0 1 1	0	1	0	45	0	837	0	BIT10 SING ERROR
1	1 0 1 0 0	0	1	0	47	0	821	0	BIT11 SING ERROR
1	1 0 1 0 1	0	0	0	53	0	808	0	ODD MULT ERROR
1	1 0 1 1 0	0	1	0	47	0	821	0	BIT12 SING ERROR
1	1 0 1 1 1	0	0	0	52	0	817	0	ODD MULT ERROR
1	1 1 0 0 0	0	1	0	47	0	821	0	BIT13 SING ERROR
1	1 1 0 0 1	0	0	0	53	0	808	0	ODD MULT ERROR
1	1 1 0 1 0	0	1	0	47	0	821	0	BIT14 SING ERROR
1	1 1 0 1 1	0	0	0	52	0	817	0	ODD MULT ERROR
1	1 1 1 0 0	0	1	0	45	0	837	0	BIT15 SING ERROR
1	1 1 1 0 1	0	0	0	54	0	801	0	ODD MULT ERROR
1	1 1 1 1 0	0	0	0	54	0	801	0	ODD MULT ERROR
1	1 1 1 1 1	0	0	0	54	0	801	0	ODD MULT ERROR
TOTAL OCCURENCE:		1	22	231	1540	7315	26334	74613	

Table V.7.2. Distribution of Syndrome Code Occurrences for 0-6 Erroneous Bits

V.7.3 *IMPLEMENTATION*

The practical implementation of ECC is as outlined in Figure V.7.2 with a few modifications.

Parity generation and checking use the same parity trees because it would be extravagent to have double sets.

(The single names on the detailed diagram will therefore differ from Figure V.7.1.)

The reported error codes (to PES register) are not identical to the syndrome/PTOT codes. All multiple errors are sorted out by hardware and identified by setting of a FATAL bit. The 10 multiple errors which would have the codes 6, 12, 17, 25, 27, 31, 33, 35, 36 and 37 have the new numbers 46, 52, 57, 65, 67, 71, 73, 75, 76, and 77 respectively. Error in parity bit 21 is likewise catched by hardware and given the nonfatal code 37.

Some people find it confusing that some of the parities are stored in memory in EVEN = true and some with ODD = true. This does not affect the ECC principles at all and could be overlooked. It is done for one single purpose, namely the ability to detect that all 22 bits are = 0 or = 1. This is a situation that is only likely after power fail on memory.

It is easily verified that when all memory bits are zero the syndrome code will be 10101. When all are one the code will be 01010.

The memory modules have a switch for manual disable of correction, and a red light warns when it is on. The switch is intended for maintenance purposes and failure may result if operated during an access.

When the ECC system detects an error of any kind another red indicator is lit. It will stay lit until cleared by program or the disable switch is operated.

V.8 MEMORY CONTROL AND STATUS

V.8.1 *ERROR CORRECTION CONTROL REGISTER (ECCR)*

This register controls the error correction network.

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

TRR 15

The format is as follows:

15	4	3	2	1	0
NOT ASSIGNED	6 TST	DIS	ANY	15 TST	0 TST

Description:

Bits 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.

Bit 1: Set to "1" simulates memory error in bit 15.

Bit 2: Interrupt condition control bit.

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

Bit 3: Disable.

When this bit is set, error correction and parity error interrupt are disabled. However, generation of correction codes during WRITE are intact.

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

The conditions causing partiy error are:

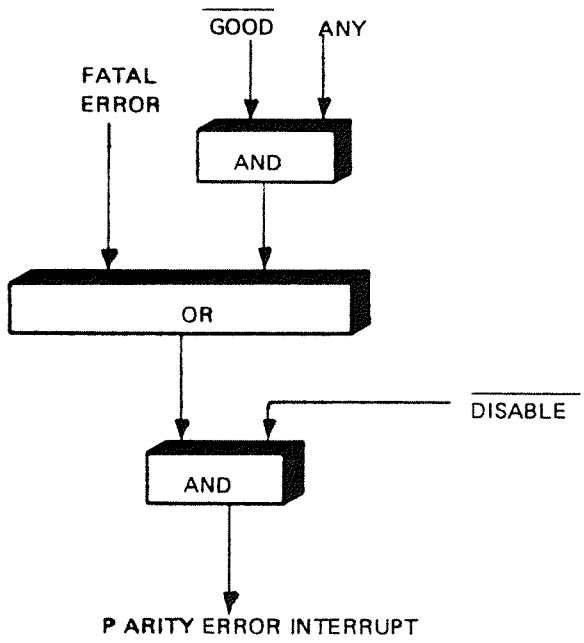


Figure V.8.1: Parity Error Interrupt

Operation of the test bits produce an erroneous correction code (6 bits) but the 16 data bits are not affected. The test bits force the respective data bits to 1 in the correction code generator. Hence, a data bit which is already "1" will not have an erroneous code generated. To test the correction network, it is therefore required that bits 0, 6 and 15 in the test word are all zero. Furthermore, correction must be disabled (control bit 3 = 1) while the test word is written into memory. Depending on the test bits (control bit 0, 1 and 4) various erroneous correction codes may be imposed on the otherwise correct test word.

When the test word is read back the control word must be 00100.

Depending on the errors imposed on the test word, the following error codes will be produced:

ECC bit:	TST15 1	TST6 4	TST0 0	Error Code PES bit 13-8	Interpretation from table V.7.1
0	0	0	000000	Good
0	0	1	000011	E0
0	1	0	001101	E6
0	1	1	101110	Multiple error
1	0	0	011100	E15
1	0	1	111111	Multiple error
1	1	0	110001	Multiple error
1	1	1	010010	E9

The single bit errors will produce an error code in PES, telling that a single bit error has been corrected. The multiple errors have "FATAL" set, and will generate an interrupt.

All test bits set (triple error) will generate an error code, telling that bit number 9 is failing, which is not true. This is one of the errors which are not detected.

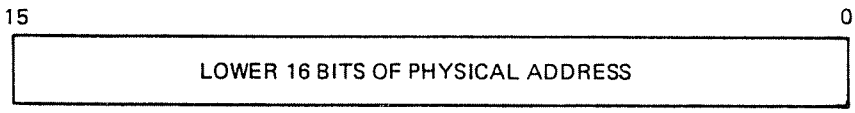
V.8.2 *MEMORY STATUS REGISTERS*

Two internal registers will give additional information when a memory error has occurred (parity error or memory out of range). The two registers are:

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

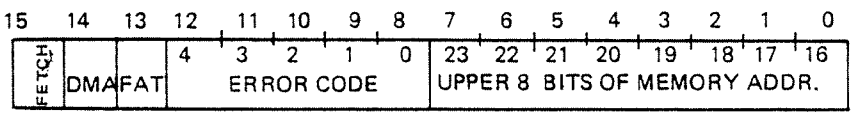
Both registers are read to the A register by the TRA instruction.

Format of PEA (A register after TRA 15):



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

Format PES (A register after TRA 13):



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 for decoding of the error code (Table V.7.1).

Bit 13: Fatal.

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

Bit 14: DMA; error occcurred during DMA reference.

Bit 15: Fetch — error occurred during instruction fetch, an EXAM or a 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 does not contain correct information unless an internal interrupt with code 10 or 11 (parity error and memory out of range) is detected.

V.9 ERROR LOGGING

Using the error correction feature, error correction is automatically done on single bit errors and the CPU is not alerted.

In a running system, the permanent memory failures are of interest while the intermittant ones generally are not.

A periodic RT program, belonging to the Operating System, will operate on the ANY bit (error correction control bit 2) once each hour. Single error information may then be put on the error file. This file is useful from a maintenance point of view.

V.10 MULTIPORT MEMORY

V.10.1 *GENERAL*

NORD-100 can be equipped with a multiport memory transceiver in the NORD-100 bus to access the multiport memory (MPM) system. The MPM system, delivered with NORD-100, is called "Big Multiport Memory" in NORD-10/S systems. It is described in detail in the manual "Big Multiport Memory System".

MPM is a high speed, flexible and modular memory system used for two major applications:

- multiprocessor communication through a shared memory system
- shared memory between CPU and high speed DMA devices

The MPM system allows up to four sources to access the same physical memory area (memory bank).

Each source is connected to one of four MPM ports through a multiport channel. All devices meeting the multiport channel specification are allowed access to this memory system. Each memory channel has an address range of 2 M words (21 bits) and may be connected to 1 to 10 ports.

The MPM system is physically located in a separate card crate. One card crate can hold 384 K words, which may be expanded up to 8 crates.

MPM is equipped with an ECC system, which detects and corrects single bit errors and detects multiple errors.

A special service channel is provided for a number of operational and maintenance purposes.

V.10.2 *MULTIPROCESSOR SYSTEM*

Figure V.10.1 shows an example of multiport memory usage. In this example two NORD-100 processors are connected together, each having local memory and a common multiport memory, consisting of 2 banks. The two sources are connected to the multiport by a multiport channel. Since each bank has four parts, two are still free and other channels may be connected.

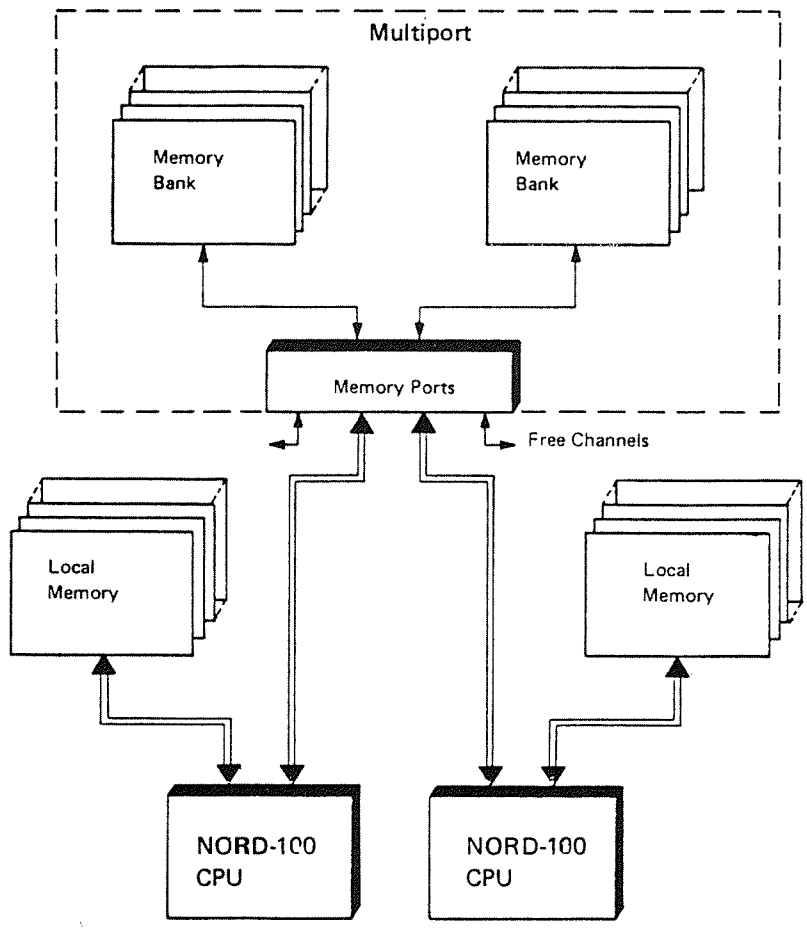


Figure V.10.1: NORD-100 Two Processor System

V.10.3 *SYSTEM PARTS*

V.10.3.1 *Crate*

Refer to Figure V.10.2 which shows the card crate.

Although the bank is logically the basic building block, the card crate is physically the basic unit, containing one or two banks, an X bank and a Y bank. This is done purely for space and wiring reasons. There is no logic in common between the two banks. Yet the service channel has common logic since it will always be connected in a predetermined manner.

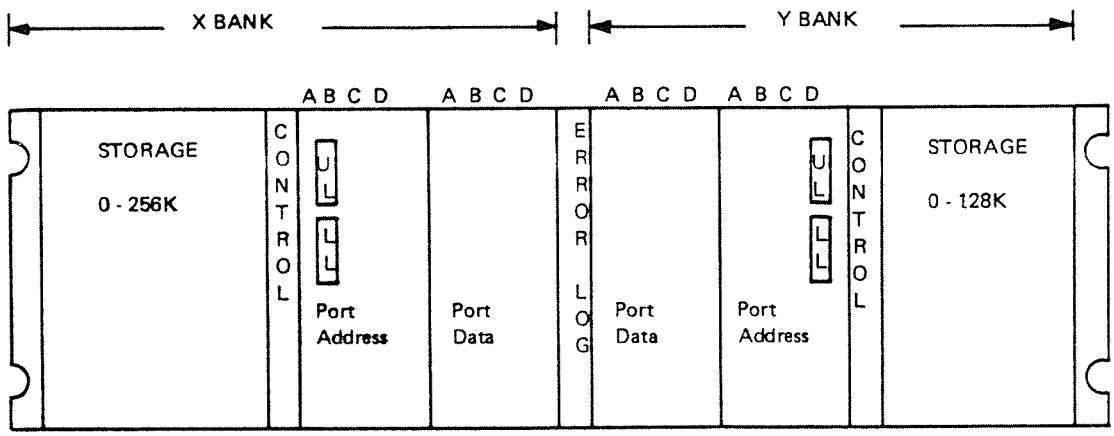


Figure V.10.2: Multiport Memory

V.10.3.2 *Source*

A source can be any kind of electronic unit which is capable of communication with MPM over a channel. Examples of sources can be: NORD-100, NORD-10/S, NORD-50 or DMA interfaces. The sources must conform to the channel specifications.

V.10.3.3 *Channel*

A channel is defined as a set of signals necessary to establish communication between one source and several banks. Physically, these signals are carried on two flat calbes interconnecting the source with the appropriate ports.

All channels meet only one type of port. Hence, all sources must conform to the same*specifications* when requesting memory.

It is possible to connect several banks to one channel.

Every port has one input and one output connector for each cable, which allows *daisy chaining* of up to 10 ports. At the end of the chain a *termination* plug is mounted.

V.10.3.4 *Bank*

A bank is a memory unit with a separate and independent timing circuitry. A bank consists of a storage section, a control section and several (1 - 4) ports through which it communicates with a source. In addition, each bank has an error log which records all errors discovered in the bank.

The storage section consists of 1 to 8 (in X bank) and 1 to 4 (in Y bank) memory modules, each of 32 K x 21 bits.

V.10.3.5 *Port*

A port is the channel's entrance to a bank. Each port consits of receivers for addresses, and transmitters and receiver for data and control lines.

Up to 10 ports can be linked together (daisy chained) by a channel, i.e., a source can access up to 10 banks of storage.

A port consits of a data and an address module. A port carries out two major functions:

1. Selecting a bank for the source and converting the channel address to a local bank address.

2. Generate a 5 bit control code to be attached to data during write and checking the data against the control code during read.

Each port defines the address range for a bank of the source(s) connected. Normally, the address range is the same seen from the different ports, but for special purposes they may be different. One such example is the NORD-100/NORD-10 communication using MPM as a common memory.

V.10.3.6 *Control*

It is essential that every bank has a private controller which allows it to operate independently of other banks.

The controller solves two major tasks.

1. To serve as a "switchboard" between the ports on one side and the storage on the other.

2. To refresh the MOS memory elements at regular intervals.

Since up to 4 ports (A-D) may access the same storage, a priority network allocates the storage to one port for one storage cycle at a time.

The ports and refresh are assigned a fixed priority which is as follows beginning at the highest priority.

New request from the same port as previous one, REFRESH, PORT A, PORT B, PORT C, PORT D.

When two ports make requests simultaneously the highest priority port will be served first.

It is not possible for *one* high priority port to lock out lower priority ones because the storage section is faster than a port. However, *two* high priority ports are able to lock the bank for lower priority ports.

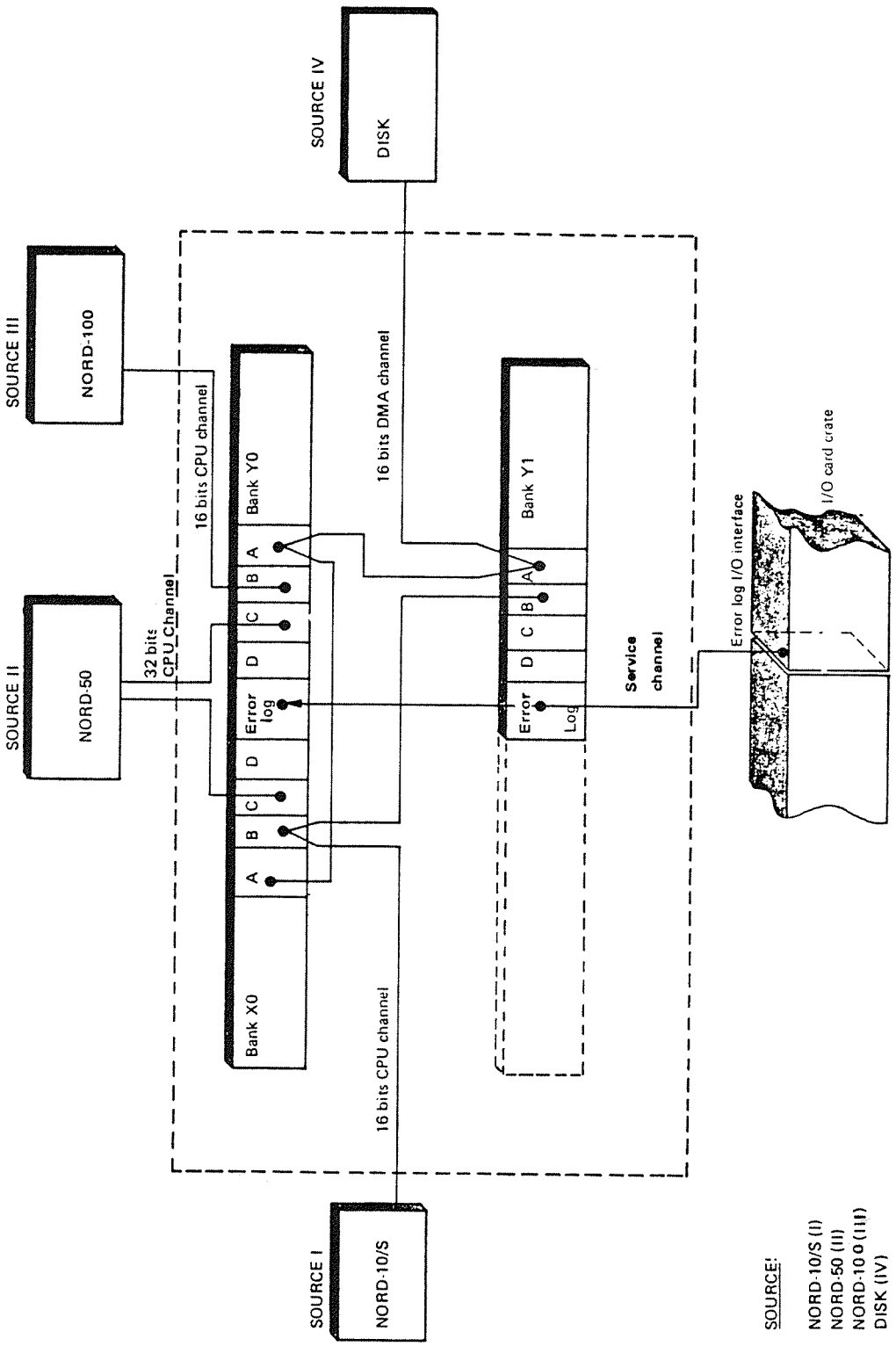


Figure V.10.3: Multiport Configuration – Example

V.10.4 *ADDRESSING*

V.10.4.1 *Address Conversion*

The total address range of a channel is 2048 K (21 address bits). Since each bank has a range of 256 K, 8 banks are necessary to cover the complete range. Hence, the 3 most significant address bits could be decoded for bank selection.

However, since the minimum size of a bank is 32 K (1 storage module) it is desirable that one has a resolution of 32 K for selection. Hence, the 6 most significant bits are needed for decoding of 32 K banks.

Straight decoding requires banks of equal size. To permit banks of different sizes a more flexible methods is required.

As for local memory described earlier, each port has therefore a set of limit switches which define the address range the port will respond to.

One pair determines the lower limit, LL, and one pair determines the upper limit, UL. Each pair holds a two digit octal number. Refer to the chart in the table below for the corresponding address range.

When a request appears on the channel, all ports test the address against their limit switches. The request has access if:

$LL \leqslant \text{Channel Address} < UL$

The request will also be sent to local memory but no response will be given since local memory should not hold any physical memory in the address range of the multiport memory.

Inside all banks the address range is 0 - 256 K, so the channel address must normally be transformed to an internal bank address which is offset by the lower limit (LL).

This transformation is performed at the port, and it is a simple subtraction:

$$\text{Bank Address} = \text{Channel Address} - \text{Lower Limit}$$

Accordingly, channel address equal to lower limit is directed into bank address 0.

Observe that two channels *may* access the same memory cell although their channel addresses are *different*. Address skew between channels should be avoided as much as possible since it makes the system very opaque both from a hardware and software point of view.

Upper (UL) and Lower (LL) Limit Switch Setting to Address Limit Correspondance:

Switch Setting:	Address Limit:
00	0 K
01	32 K
02	64 K
03	96 K
04	128 K
05	160 K
06	192 K
07	224 K
10	256 K
20	512 K
:	:
30	768 K
:	:
40	1024 K
:	:
50	1280 K
:	:
60	1536 K
:	:
70	1792 K
:	:
77	8016 K

Table V.10.1: The Switch Settings

In Figure V.10.4 is shown how two sources (e.g., processors) may communicate via MPM.

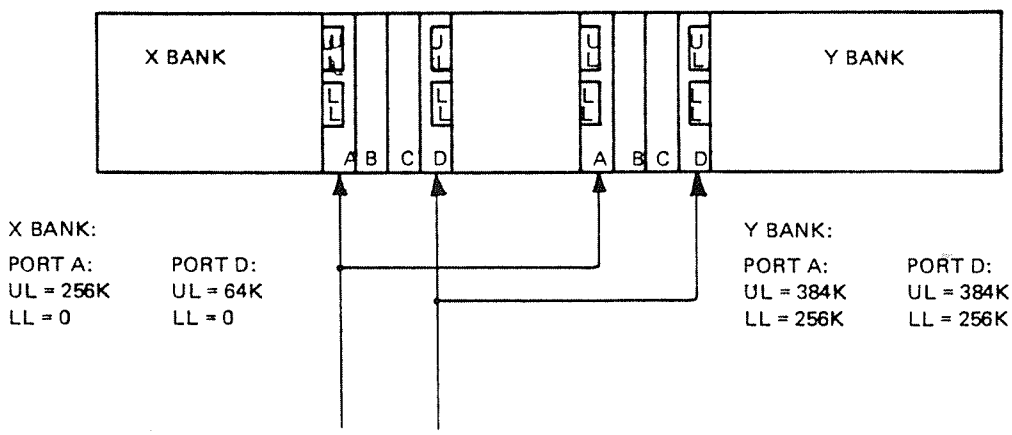


Figure V.10.4: Switch Settings for Common Memory Area

With switch settings as shown in Figure V.10.4, source 1 and 2 will have a common memory area, and source 1 will also have a "private" memory area.

V.10.5 *INTERLEAVE*

To increase memory bank width an interleave technique can be used. However, it is only meaningful in a pipeline system where a new request is issued before the previous is serviced.

In a less critical sense "interleave" is used to designate shifting of the address bits.

If the address is rotationally shifted one place to the right, bit 0 will receive position 20, bit 20 position 19 and so on. The effect is that all even addresses will seen to lie between 0 and 1024 K while all odd addresses will appear to lie between 1024 K and 2048 K.

Two consecutive addresses (as seen from the program) will therefore lie in different banks. The bank width benefit expectation from this shifting is close to zero. However, interleaving is necessary when 16 bit and 32 bit channels are to communicate.

Two 16 bit words, at consecutive addresses, may be read as *one bit* word when the two 16 bit words *reside in different banks*.

The address shifting is performed by modifying the address cable. For more details one should refer to the manual "Big Multiport Memory System".

The concept may be illustrated in the following example.

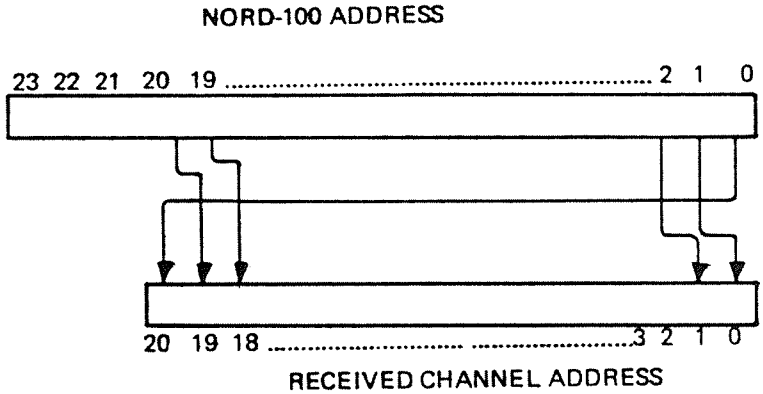


Figure V.10.5: Shifting of Address Bits

Address bit number 20 is used to select the bank.

Assuming the NORD-100 has stored 4 words in memory in consequitive locations (as seen from NORD-100), the lowest bit on the issued address will toggle and alternating banks will be selected. If the issued addresses are 0, 1, 2 and 3, the interleaved addresses would be:

Interleave Address:	Word No.:
0000000	1
4000000	2
0000001	3
4000001	4

etc.

The following illustration will show where the four words will be stored.

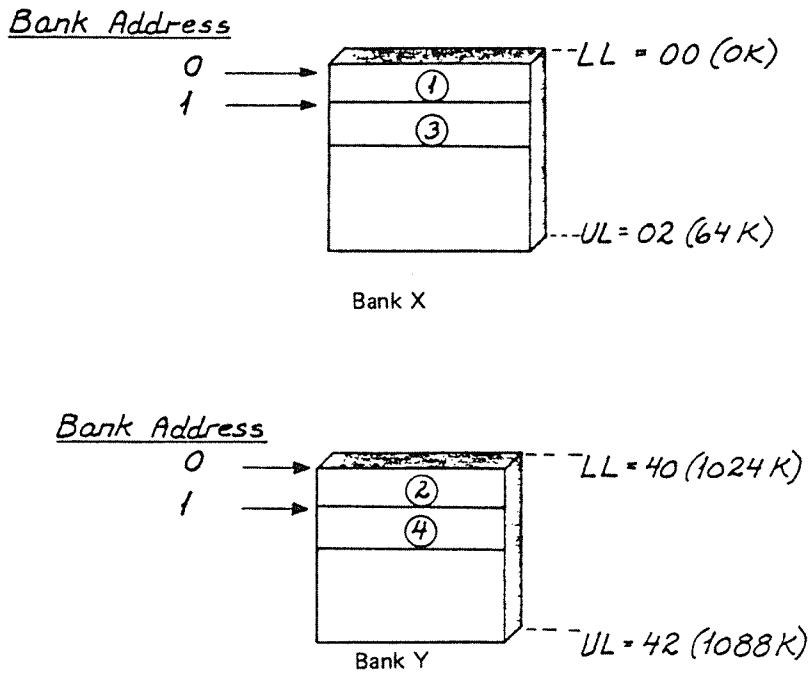


Figure V.10.6: Banks as Seen from NORD-100

V.10.6 *NORD-100 — NORD-50 COMMUNICATION*

Using multiport memory as a common storage, a NORD-100 — NORD-50 Memory Communication can be established.

However, the NORD-50 is a 32 bit computer and likes to see a memory system with its own word length. This can be done by giving two banks the same address space as seen from the NORD-50 channel, while the two banks are interleaved as seen from the NORD-100.

If we regard the example from Section V.10.5, the NORD-100 has stored four words in memory, two in bank X and two in bank Y. The first and second words are stored in the same local address in the two banks, and the third and fourth words are stored in the next locatin in the two banks.

When the NORD-50 reads those 4 NORD-100 words, it regards the same two banks as one bank with double word length. The 4 bit words will then be read as two 32 bit words.

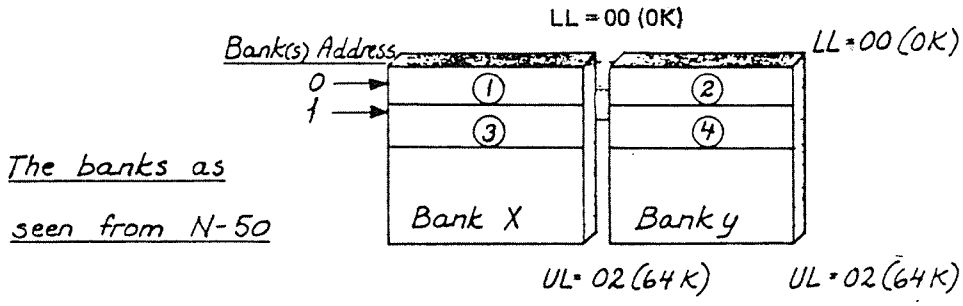


Figure V.10.7: Banks as Seen from NORD-50

Figure V.10.8 shows the physical connection between NORD-100 and NORD-50.

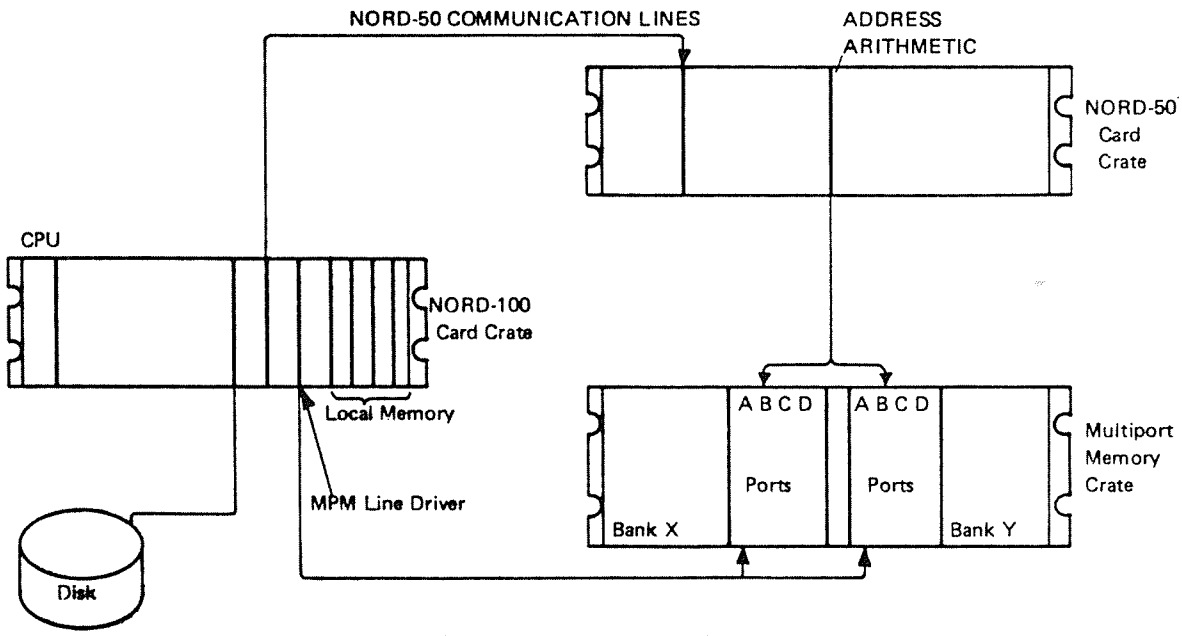


Figure V.10.8: Physical Connection between NORD-100 and NORD-50.

V.10.7 *Service Channel*

The service channel connects error log modules in each crate to the error log I/O interface module located in the NORD-100 bus. The error log system is regarded as one I/O device as seen from the NORD-100.

An error log module is logically divided into two parts, one serving the X bank and the other the Y bank. Each part contains an error log memory. All errors (single or multiple) occurring in the bank and detected by the ports, will be recorded in this special memory. The error log memory is organized as a 512 x 1 bit memory where the address is a pointer to the failing memory integerated circuit.

A thumbwheel switch located on this module defines the crate number (0 - 7).

MPM uses an error detection and correction network (ECC) which is very similar to the one used in local memory (for further explanations refer to Section V.7).

In multiport memory 5, error correction bits are stored in addition to the normal 16 data bits. Each port has its *private* error correction network in order to load the common memory resources as little as possible. Refer to Figure V.10.9.

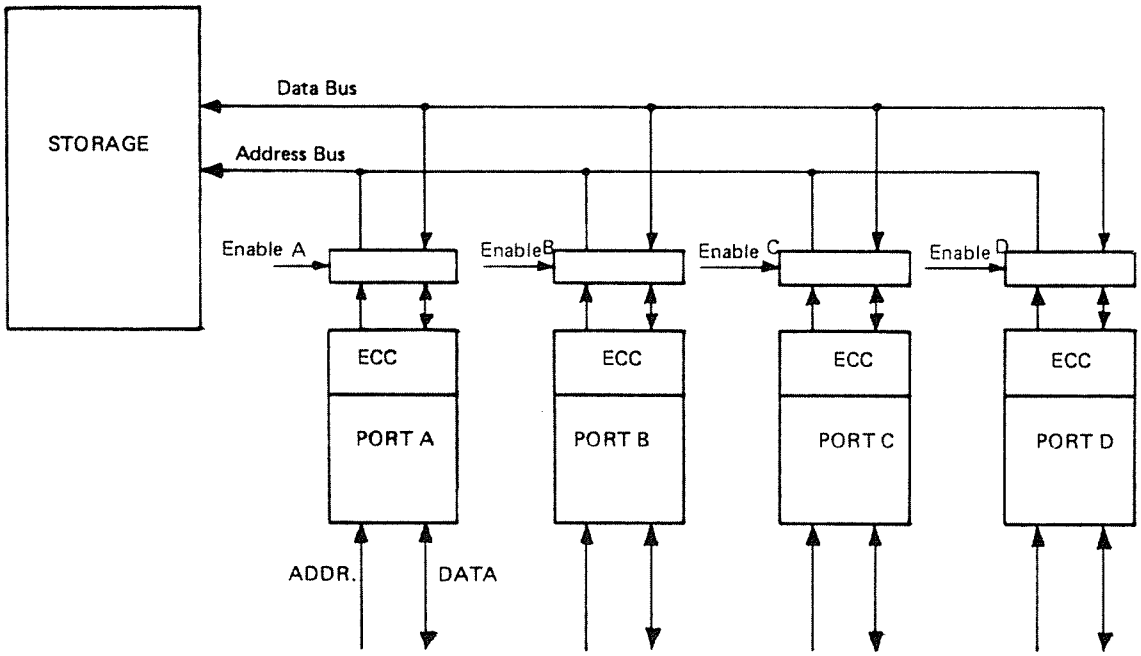


Figure V.10.9: Error Detection and Correction in a Bank

This network detect and correct single bit errors and report multiple errors. Because the multiport channel specifications appeared before the error correcting technique only parity errors may be detected. When the ECC network detects multiple (fatal) errors it forces a parity error which is detected by a normal parity check in the source. Single bit errors are (of course) not reported to the source. However, all errors are written in the error log which is read through the service channel.

SECTION VI

THE INPUT/OUTPUT SYSTEM

VI.1 GENERAL

The purpose of the input/output system (I/O system) is to perform the physical communication between connected peripheral equipment and the NORD-100 computer system.

The user normally does not interact directly with the I/O system, only indirectly via the operating system.

However, privileged users may access the I/O system directly and users with special real-time requirements (running direct tasks) may bypass the I/O system for direct access to specific devices.

The I/O system provides a two-way communication between the CPU and its peripherals, and is designed to be a flexible system providing communication between slow, character oriented devices as well as high speed, block oriented devices. 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.

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

- CPU controlled, Program Input/Output (PIO)
- with Direct Memory Access (DMA)

VI.2 PROGRAMMED INPUT/OUTPUT — PIO

External devices may be classified as:

1. Slow character/word oriented devices (for example, terminals)

2. High speed block oriented mass storage devices (for example, disks, mag. tape)

Data exchanged between these classes of peripherals and NORD-100 fall into one of these two categories. The first is completely controlled by program and is called Programmed Input/Output (PIO).

A PIO interface is always designed to handle slow byte/word oriented devices (tape reader, line printer, etc.) and is completely controlled by the CPU. In programmed data transfers, each word or byte to be exchanged has to be programmed between the selected I/O device controller (interface) and the A register in the CPU, as illustrated in Figure VI.2.1.

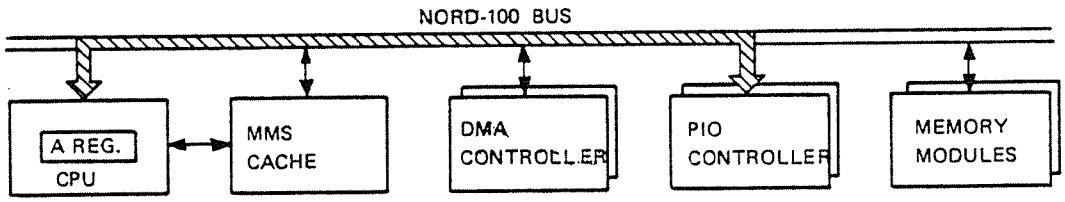


Figure VI.2.1: PIO Data Exchange

VI.3 DIRECT MEMORY ACCESS — DMA

The second class is for high speed block oriented peripherals, which exchanges data direct with the NORD-100 memory system. This is called direct memory access (DMA).

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 main memory via the NORD-100 bus. This connection is called a DMA channel.

The IOX interface which controls the transfer, is activated by a driver program in SINTRAN III. The activation includes telling the I/O interface where to place/find data in memory, where to find/place data on the external device and the number of words to exchange.

After activation, a DMA transfer runs completely independent 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 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.

A HAWK disk, for example, will steal one cycle of 550 ns per each 6.4 μs 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.

The NORD-100 bus is allocated for a DMA transfer when a word is ready to be exchanged. The connection between a DMA controller and the memory system is referred to as a "DMA channel". More than one DMA controller (interface) may be activated at the same time, thus sharing the DMA channels total band width (1.8 M words/second).

Typical DMA devices are:

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

A DMA transfer is illustrated in Figure VI.3.1.

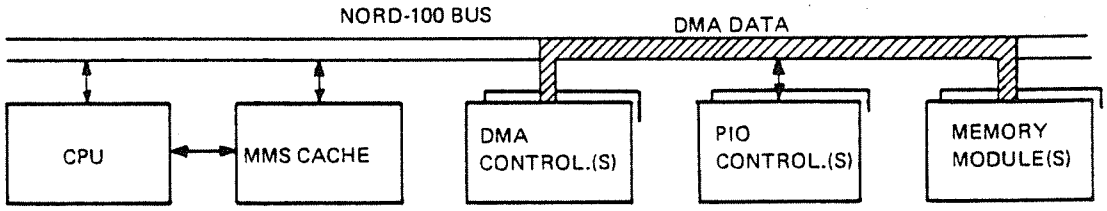


Figure VI.3.1: DMA Data Exchange

The NORD-100 bus is allocated to the transfer of data, and the CPU is busy for every byte/word which is exchanged.

VI.4 NORD-100 BUS IN THE I/O SYSTEM

VI.4.1 *GENERAL*

The NORD-100 bus provides the communication between functional blocks in NORD-100 computer system. On the NORD-100 modules, the parts which communicate with the bus are made to a common standard, i.e., all communication is carried out in an identical fashion. This convention also includes I/O device controllers.

VI.4.2 *ORGANIZATION OF AN I/O DEVICE CONTROLLER MODULE*

The device controllers are normally divided into two parts, the NORD-100 dependent part and the device dependent part.

Figure VI.4.1 shows a standard NORD-100 I/O module.

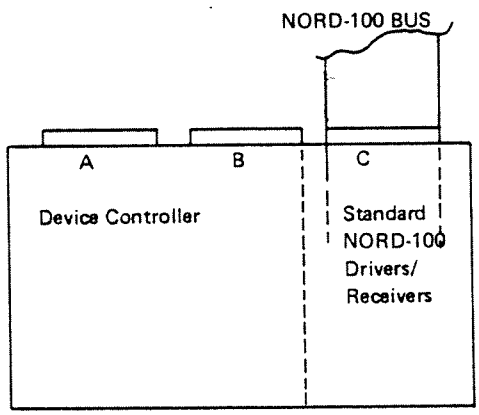


Figure VI.4.1: Standard NORD-100 I/O Module

The NORD-100 dependent part includes bus handshake control logic. This part is standarized for:

- all PIO device controllers
- all medium speed DMA controllers (10Mb disk, mag. tape)

The device dependent part may handle up to four different PIO devices or one DMA controller. A DMA controller may handle up to four units.

Example — one module:

- four terminals and floppy disk
- 8 terminals
- one 10Mb disk controller (up to 4 units)
- one mag. tape controller (up to 4 units)

VI.4.3 *ALLOCATION OF THE NORD-100 BUS*

One of the functions of the Bus Control, is to allocate the NORD-100 bus to one of the possible requesting bus users (refer to Section IV). That is, to:

- the CPU
- a DMA controller
- a memory refresh cycle

These sources request the NORD-100 bus asynchronously, and therefore a priority arbiter network is implemented in the Bus Control.

In order for the Bus Control to know who has initiated a request, each bus user is assigned a unique bus request signals.

CPU Bus Request

The CPU may allocate the NORD-100 bus for one of six reasons:

- instruction fetch* } memory access
- operand read* } memory access
- indirect address read* } memory access
- operand store } memory access
- programmed access to the I/O system — I/O system access
- programmed access to external system control regsiters**

* Instruction, operand, indirect address not found in cache memory (if cache is present)

** Control registers not located on the CPU or MMS module (for example, Error Correction Control Registers (TRR ECCR) on memory modules).

DMA Bus Request

A DMA controller tries to allocate the NORD-100 bus to establish the DMA channel to memory each time a word is ready to be exchanged. The frequency of the DMA requests depend on the speed of the peripheral using the DMA channel and the number of active DMA controllers sharing the DMA channel.

Memory Refresh Bus Request

The memory refresh cycle is initiated every 13 μs by an oscillator on the CPU module.

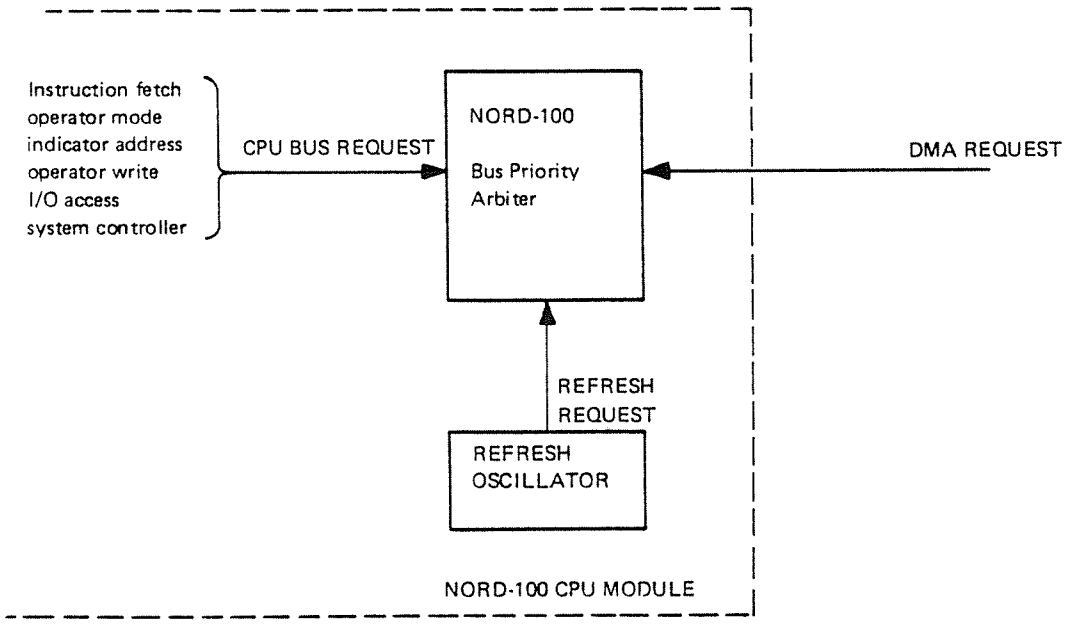


Figure VI.4.2: Requesting Sources to the Bus Control Unit

The NORD-100 bus is allocated and released on cycle basis, i.e., for every byte/word to be exchanged. Due to the multiplexed bus (refer to Section IV) the cycle may be divided into an address cycle and a data cycle.

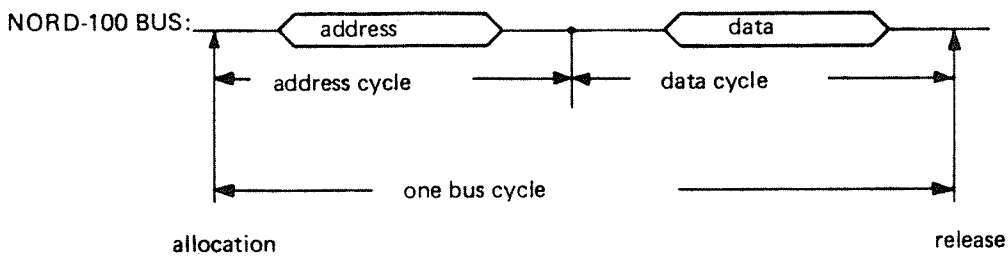


Figure VI.4.3: NORD-100 Bus Cycle

The time from allocation to release of the NORD-100 bus shall not exceed 10 μs. This is monitored by the Bus Control.

The accessed device, i.e., the I/O system or memory system, releases the bus when ready (bus data ready — BDRY).

If a bus is not released it causes system hang-up. To prevent such a situation, the Bus Control will abort a bus cycle which exceeds 10 μs. The faulty cycle is reported to be CPU as internal interrupt on level 14 (MOR, IOXERR).

VI.5 PROGRAMMING OF I/O DEVICE CONTROLLERS

VI.5.1 *GENERAL*

To start an I/O transfer, the PIO interface or the DMA controller has to be activated. This is done by the device driver program when a transfer is requested, either from a user program or from an I/O device controller through a hardware interrupt.

The control of an I/O device interface includes programmed access to physical hardware registers on the interface. This is performed by the two instructions, IOX and IOXT, which is implemented for this purpose.

VI.5.2 *THE INPUT/OUTPUT INSTRUCTIONS IOX AND IOXT*

In the NORD-100 instruction set there are two instructions used for information exchange between the hardware device controllers and the CPU, the IOX and the IOXT instructions. These are privileged instructions, that is, only privileged users may use these instructions.

If the operating system is not running and paging is off, IOX and IOXT is available as other nonprivileged instructions.

In the NORD-100 instruction set, IOX and IOXT are the only instructions that can be used in information exchange between CPU and I/O device controllers.

Their purpose is to carry information between the CPU's A register and a specified I/O device register.

The actual function of the IOX/IOXT instructions *depend on the selected I/O device register.*

All I/O device controllers are assigned a group of registers, each of them having a special meaning on the interfaces. Therefore, the programmer, by specifying an appropriate register on a device, assigns the exchanged information a special function.

Thus, all types of information may be exchanged:

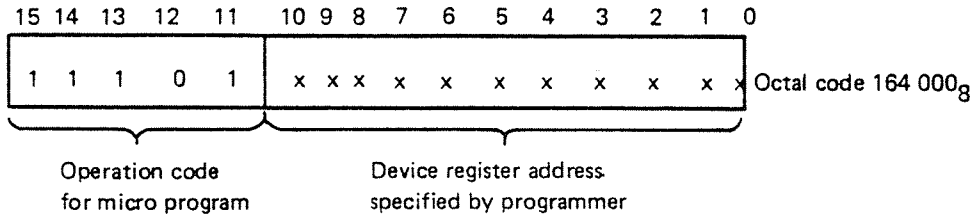
- data (byte or word) to or from PIO interface data registers (not DMA interfaces)
- transfer control information to PIO and DMA interfaces control registers
- transfer status information from PIO and DMA interface status registers.

All I/O device registers are assigned an address referred to as "device register address". That is, both the IOX and the IOXT instructions access an I/O device register by its address.

IOX Instruction Format

In the IOX instruction, the address of the I/O device register to access is specified in the 11 lower bits of the instruction itself.

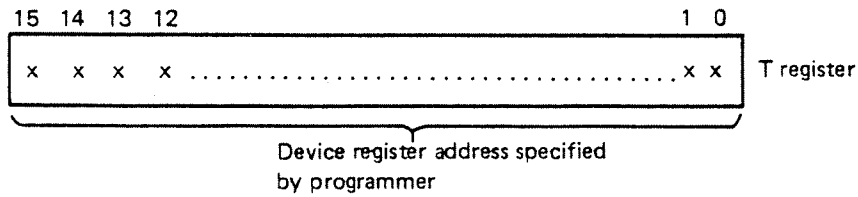
IOX <device register address>

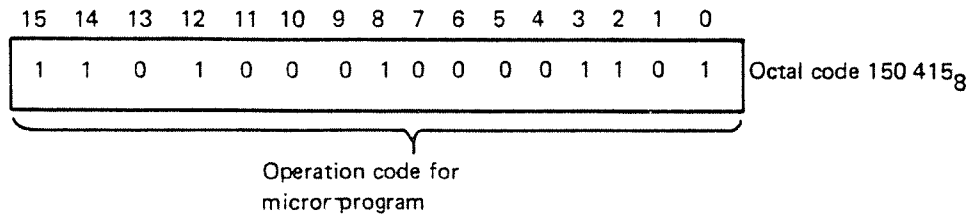


IOXT Instruction Format

In the IOXT instruction, the device register address should be loaded into the T register prior to executing IOXT.

LDT <device register address>
IOXT





VI.5.3 *THE TRANSFER DIRECTION*

The IOX and IOXT instructions handle both input and output transfers. An input transfer in this context means input to the CPU A register from a specified I/O device register. An output transfer means output from the CPU A register to a specified I/O device register.

The actual transfer direction of the IOX and IOXT instructions is decoded from the device register address based on the following convention:

— The transfer direction is *input* if the device register address is *even*. That is, bit 0 of the address is "0".

— The transfer direction is *output* if the device register is *odd.* That is, bit 0 of the address is "1".

The programmer is not affected by this convention. All I/O device registers which should be loaded from the CPU A register (output transfer) are assigned an odd device register address.

VI.5.3.1 *THE IOX INSTRUCTION TRANSFER DIRECTION*

The device register address of the IOX instruction is decoded to:

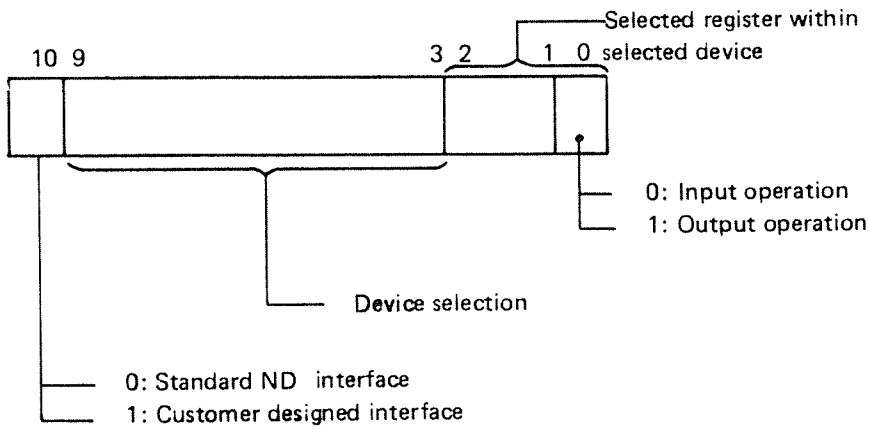


Figure VI.5.1: IOX Device Register Address Decoding Details

Input Transfer

An I/O device register specified with an even device register address (bit 0 = "0") is loaded into the A register.

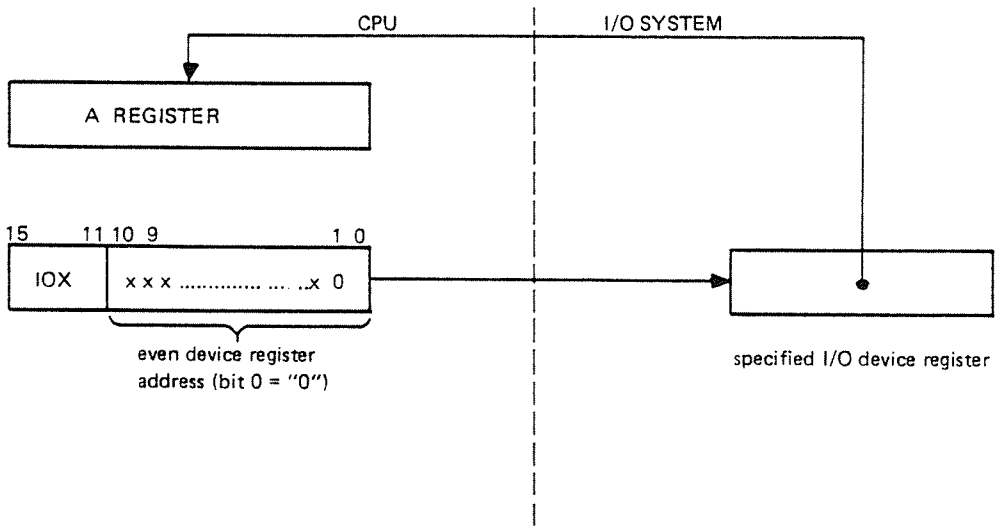


Figure VI.5.2: IOX Input Transfer

Output Transfer

The CPU A register is written to an I/O device register specified by an odd device register address (bit 0 = "1").

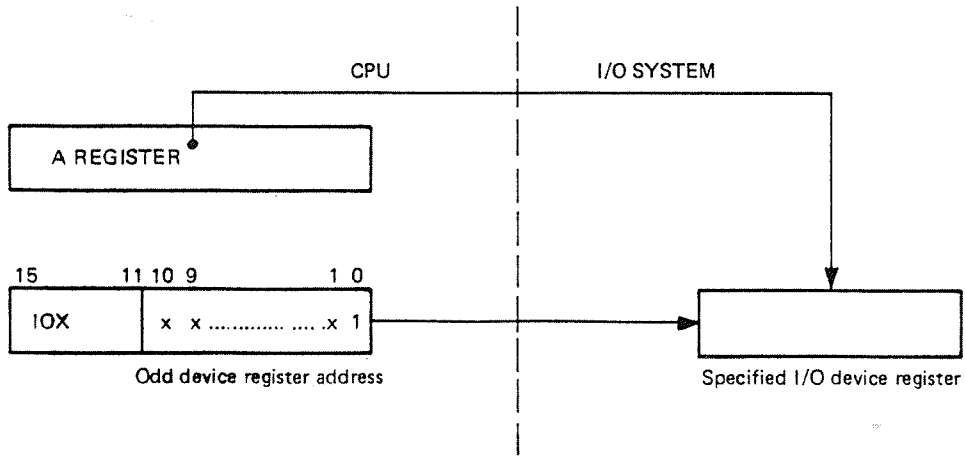


Figure VI.5.3: IOX Output Transfer

VI.5.3.2 *THE IOXT INSTRUCTION TRANSFER DIRECTION*

In the IOXT instruction the T register holds the 16 bits device register address.

Input Transfer:

An I/O device register specified with an *even* device register address is loaded into the A register.

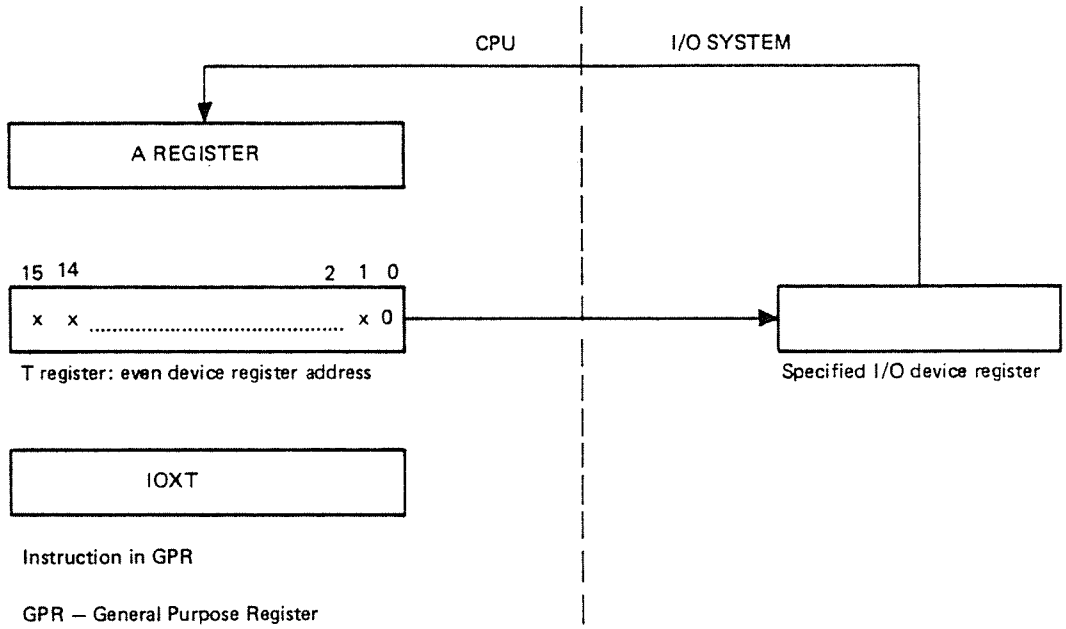


Figure VI.5.4: IOX Input Transfer

Output Transfer:

The CPU A register is written to an I/O device register specified by an *odd* device register address.

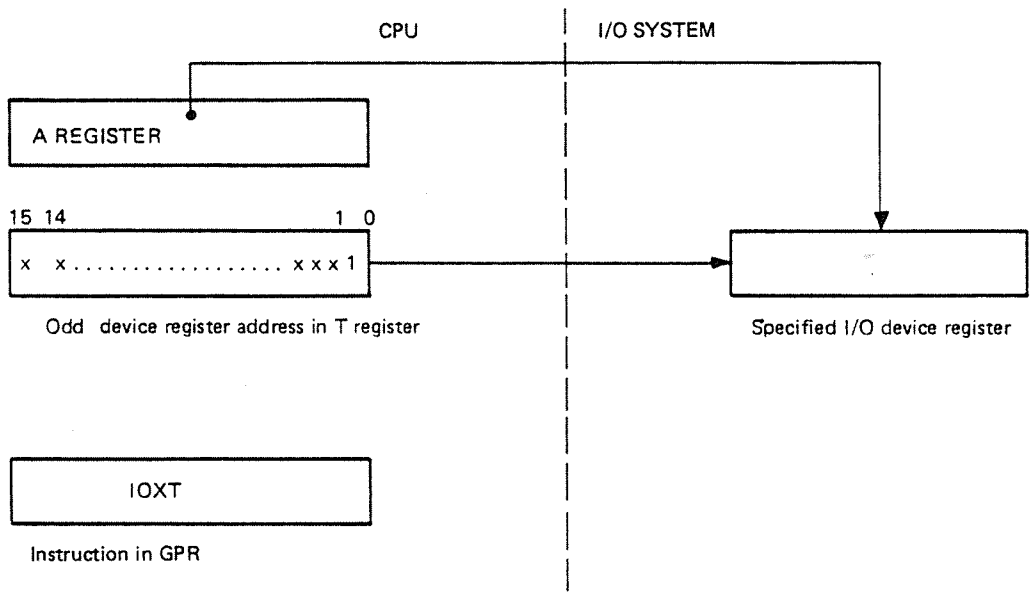


Figure VI.5.5: IOXT Input Transfer

VI.5.4 *CALCULATION OF THE DEVICE REGISTER ADDRESS*

VI.5.4.1 *The IOX Instruction Address Range*

By using the IOX instruction a total of 2K registers may be specified, i.e., addresses from 0-3777_8. However, the collection of all device registers implemented on interfaces designed at Norsk Data only cover the address area 0 - 1777_8, i.e., 1K register. The address range covered by the IOX instruction is organized as illustrated below.

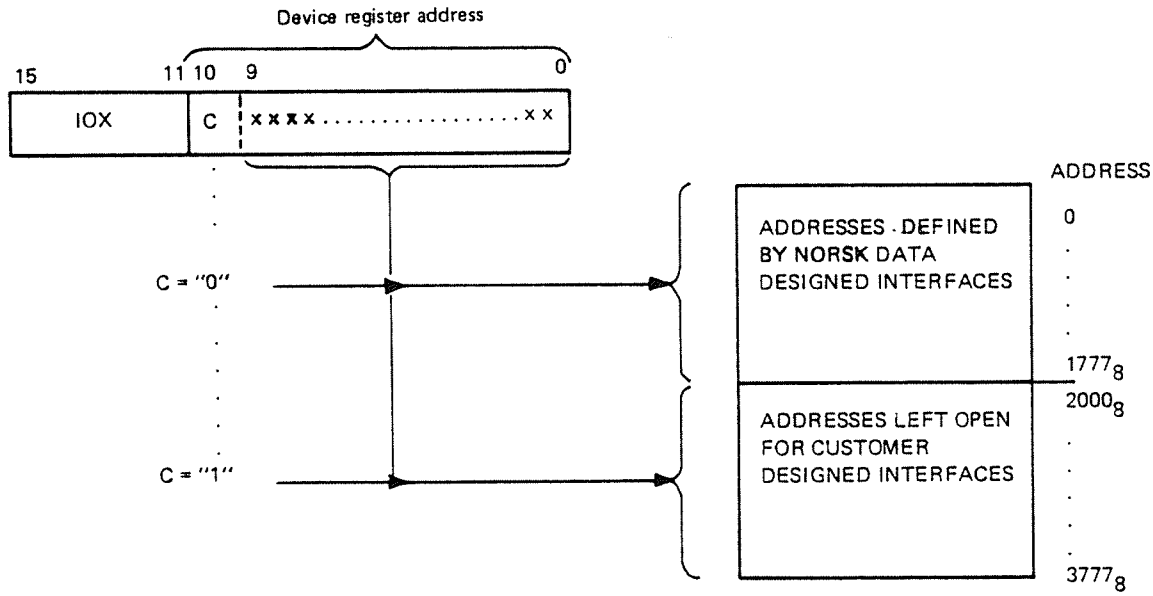


Figure VI.5.6: IOX Address Range

As indicated, bit 10 in the IOX device register address, equal to "0", defines an address defined by Norsk Data's equipment. If bit 10 is "1", the specified address has to be defined by some customer designed equipment.

VI.5.4.2 *THE IOXT INSTRUCTION ADDRESS RANGE*

The IOXT instruction, which uses the 16 bit T register to hold the device register address, may theoretically address up to 64K registers (addresses from 0-177777_8).

The T register should be initiated with a relevant device register address prior to the execution of IOXT.

The address range covered by the IOXT instruction is organized as shown below.

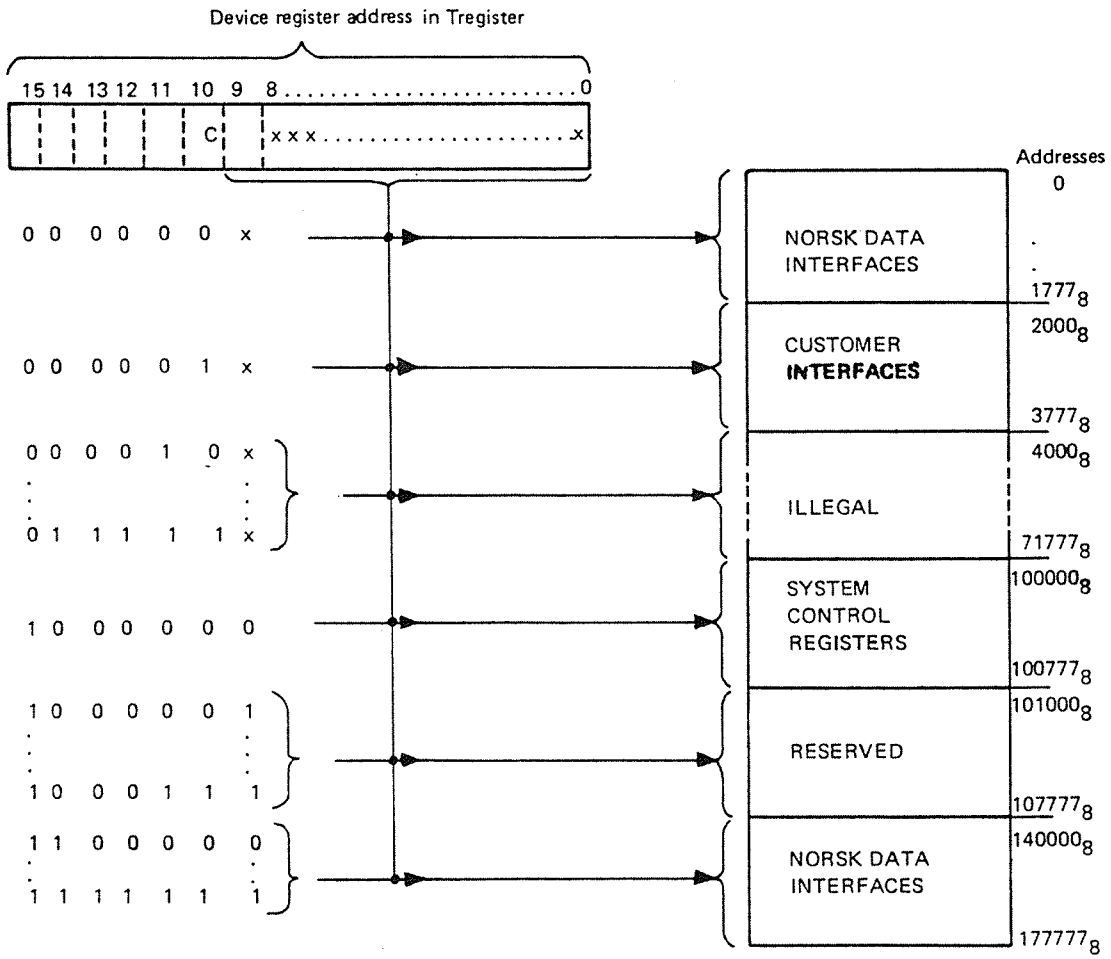


Figure VI.5.7: IOXT Address Range

As indicated, if the bits 11-15 are all zero, the device register address of the IOXT instruction overlap in the IOX instruction's address range.

If one of the bits 11-14 is "1" and bit 15 is "0", the address is illegal.

If bit 15 is set ("1"), registers not accessible by the IOX instruction may be specified.

Addresses from 100000_8 - 100777_8 are used to specify system control registers which have to be accessed via the NORD-100 bus. An example is the Error Correction Control Register (ECCR), physically located on the memory modules.

ECCR is loaded by the TRR instruction. However, since ECCR is accessed via the NORD-100 bus, the micro program converts the TRR instruction to an IOXT instruction (IOXT 100115_8).

The registers in this address area are not relevant to the I/O system and are therefore not discussed any further in this manual.

Addresses from 101000_8 - 107777_8 are reserved by Norsk Data for future needs.

Addresses from 140000_8 - 177777_8 are reserved by Norsk Data for future extension of the I/O device register address range.

Since all present I/O device controllers designed at Norsk Data may be specified in the address area 0 - 1777_8, and may be specified by both the IOX and the IOXT instruction, the IOXT instruction is used in the programming examples and when referred to.

VI.5.4.3 *SPECIFICATION OF AN I/O DEVICE REGISTER ADDRESS*

To each I/O device controller is assigned a group of registers with consecutive device register addresses. The total number of regsiters assigned one I/O interface may be from 4 to 16 depending on the control functions needed on a device.

Therefore, the device register address may be divided up into two parts:

— device selection address (hardware device number)

— register selection address to select a register within the selected device (register number — in selected device)

The IOX/IOXT device register address is then calculated by the formula:

<device register address> = device number + register number

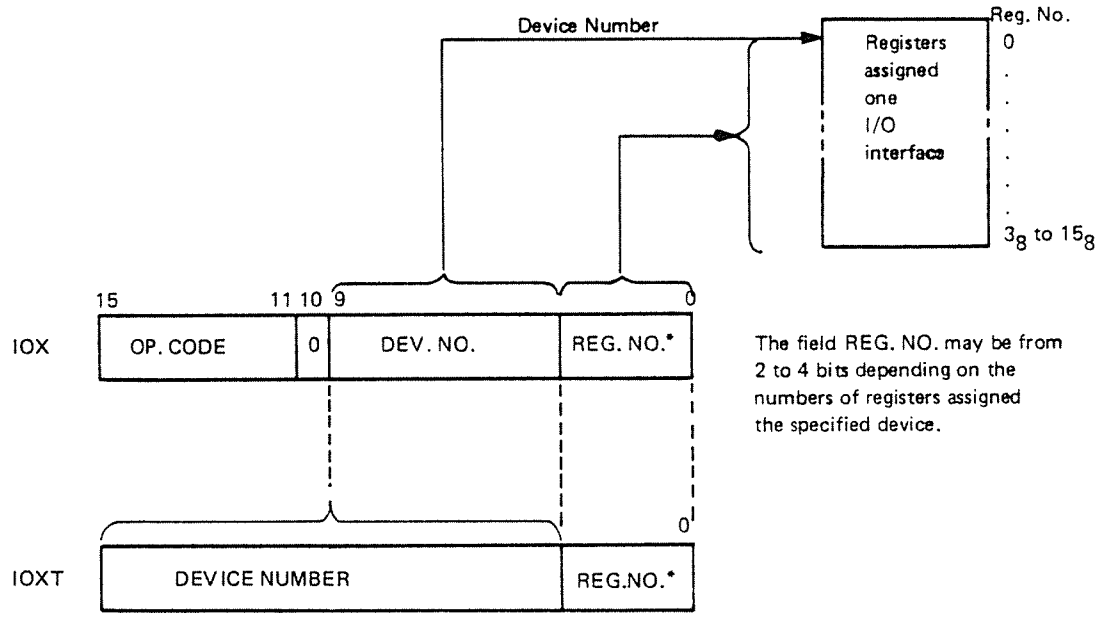


Figure VI.5.8: Specification of Device Register Address

For Norsk Data produced device controllers, both the device number and the register number have been standarized, given in Appendix C.

The numbers assigned the various registers on an I/O interface are given in the specifications following each I/O interface. See Appendix B (Programming Specifications for some I/O Devices) for more details.

Example:

The device number for terminal number 1 is found in Appendix C, and is equal to 300_8 - 307_8, which is this device's register address range. Eight registers are assigned terminal 1 (these are described in more detail in Appendix B).

There is always a correspondence between a peripheral, the I/O interface controlling the peripheral and a device number.

The device number corresponding to an I/O interface is selectable by a thumbwheel on the module (refer to Appendix D). This is done to allow equal hardware modules to cover all possible device numbers assigned one class of peripherals.

Note that there is not relationship between a module device number and its slot number in the NORD-100 bus.

VI.5.4.4 *The Device Registers on I/O Interfaces*

Each register implemented on an I/O interface is assigned a unique number in the interface, a register number (reg. no.). When both the device number and a register number are specified, the information exchanged by IOX/IOXT has a defined meaning, given by the function of the selected device register which is related to a special function on the interface.

Each I/O device controller is described in a specification referred to as "the programming specification". In the programming specification of an interface, the functions and the associated numbers of each register is described (programming specifications for some I/O interfaces is given in Appendix B).

An I/O interface is said to have two channels if it can handle both input and output transfers simultaneously. A one channel interface may handle either input or output transfers.

Examples:

One channel devices:

- line printer interface (output only)
- card reader interface (input only)

Two channel devices:

- terminal interface (output to terminal and input from keyboard)
- communication controllers (output to line and input from line)

Bidirectional (one channel) Device Controllers:

- disk interfaces (may handle both input and output but not simultaneously)
- mag. tape interface (same as for disk)

On a PIO interface designed by Norsk Data, each channel is assigned at least three registers, which are accessible by IOX/IOXT.

To a PIO interface, it is assigned at least three registers to each channel:

- a control register
- a status register
- a data register

The control register in a "write only" register (IOX/IOXT output). Commands (start/stop transfer, mode of operation) from a device driver program to an I/O interface channel is given through this register.

The status register is a "read only" register (IOX/IOXT input). By reading the register, the status of an I/O interface channel (ready for transfer, busy, errors, etc.) may be investigated.

The data registers are "write only" if they belong to an output channel or "read only" if they beong to an input channel.

Example of device register address calculation for PIO devices (refer to Appendix B).

Example 1:

Paper tape punch interface.

The paper tape punch interface has only one channel, the output channel. The regsiters assigned this interface follow Norsk Data's standard, and are defined in the paper tape punch programming specifications.

Register:	Register Number:
Output channel control register	3
Output channel status register	2
Output channel data register	1
Read data under test	0

The device register address of the status register on paper tape punch number 2, is calculated from:

device register address = device no. + register no.

Device number (dev. no.) selects paper tape punch number 2. In Appendix C this is found to be 414_8. Using the formula, the device register address will be:

dev. reg. addr. = $414_8 + 2 = 416_8$

In the same way, the device register address for the control register and the data register is found.

Assume that paper tape punch number 2 is read to accept data. The following program will write the content of location DATA to the paper tape punch data register:

```
        LDA     DATA    % Load A register
        IOX     415     % Write content of A reg. (DATA) to
                        % paper tape punch no. 2 data reg.
DATA,
```

VI.6 CONTROL AND STATUS REGISTERS ON ND DESIGNED PIO AND DMA INTERFACES

VI.6.1 *GENERAL*

All information exchange, i.e., control, status and data, between PIO devices and NORD-100 CPU is programmed via the A register by means of IOX/IOXT instructions.

On DMA controllers, control and status are programmed by IOX/IOXT instructions, while data is exchanged directly to memory.

The format of the control and status registers are device dependent. On Norsk Data designed interfaces, both function and format of these registers have been standarized. The format of the data register follows the format accepted by the external device.

VI.6.2 *FORMAT AND FUNCTION OF THE CONTROL REGISTER*

Commands to a device are given through the control register, which are loaded by:

```
LDA<command>          % initiate A register
IOX<dev. reg. addr.>  % dev. reg. addr. of I/O interface
                      % control reg.
```

The format of the control register has been standarized for all Norsk Data produced interfaces and is equal both for the input and output channel of a device.

Format of the control register:

Bit 0:	Enable interrupt on device ready for transfer
Bit 1:	Enable interrupt on errors
Bit 2:	Activate device
Bit 3:	Test mode
Bit 4:	Device clear
Bit 5:	Not assigned
Bit 6:	Not assigned
Bit 7:	Not assigned
Bit 8:	Not assigned
Bit 9:	Unit } Multiple unit control interfaces
Bit 10:	Unit } Multiple unit control interfaces
Bit 11:	Device operation
Bit 12:	Device operation
Bit 13:	Device operation
Bit 14:	Device operation
Bit 15:	Device operation

Bit 0 — Enable Interrupt on Device Ready for Transfer

Bit 0 set to "1" in an I/O interface channel enables for interrupt on device ready for transfer, i.e., if status bit 3 = "1".

Bit 1 — Enable Interrupt of Errors

Bit 1 set to "1" in an I/O interface channel enables for interrupt on errors in the channel.

Bit 2 — Activate Device

The control register bit 2 set to "1" on an interface will:

- in the input channel, enable reception of incoming data from external devices
- in the output channel, start output of the content in output channel data register
- start a DMA transfer (DMA controllers)

The activate bit is normally static, i.e., it will remain on until Master Clear, Device Clear (control regsiter bit 4) or a loading of the control register with bit 2 = "0" is done. Other reasons for resetting are given in the programming specifications.

Bit 3 — Test Mode

This bit set to "1" will loop output data back as input data in "off line" testing of an interface.

Bit 4 — Device Clear

The control regiser loaded with bit 4 = "1", generates a reset pulse on the accessed I/O interfaces. The reset pulse will clear all bits set in both the control and the status registers.

Bits 5-15

These bits are specified in the interface's programming specifications.

VI.6.3 *FORMAT AND FUNCTION OF THE STATUS REGISTER*

Feedback from a device to the CPU are given by the status register. The information available here is set by the interface itself, and when read to the CPU A register, the state of an interface channel is to be investigated. The status register is read by:

IOX<dev. reg. addr.>	% dev. reg. addr. of status reg. to % access

The format of the status register has been standarized for all Norsk Data produced interfaces and is equal both for the input and output channel of a device.

Format of the Status Register:

Status Word

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

Bit 0 - 2

The status register bits 0-2 are direct feedback of the corresponding bits in the control register of the same I/O interface channel (see control register format).

Bit 3 — Device Ready for Transfer

Status regsiter bit 3 set or not set tells whether an I/O interface channel is ready to operate or not. What is meant by "ready for transfer" (bit 3 "1") differs from the input to the output channel.

PIO device input channel

Bit 3 equal to 1:

Input data register contains valid information ready to be read.

Bit 3 equal to 0:

Input data register does not contain valid information.

PIO device output channel

Bit 3 equal to 1:

Ouptut data register is empty and may accept the next output data character.

Bit 3 equal to 0:

Output data register is *not* empty, i.e., the data register should *not* be loaded.

DMA controllers:

Bit 3 = "1"

DMA transfer completed.

Bit 3 = "0".

DMA transfer is active.

Bit 4 — Inclusive Or of Errors

Bit 4 = 1: An error has occurred in the I/O interface channel. More information about the error is given in status register bit 5 - 15.

Bit 5 - 15 — Nondefined

Described in the programming specifications for the actual device.

VI.7 LOADING SEQUENCE FOR DEVICE REGISTERS ON I/O INTERFACES

VI.7.1 *THE LOADING SEQUENCE*

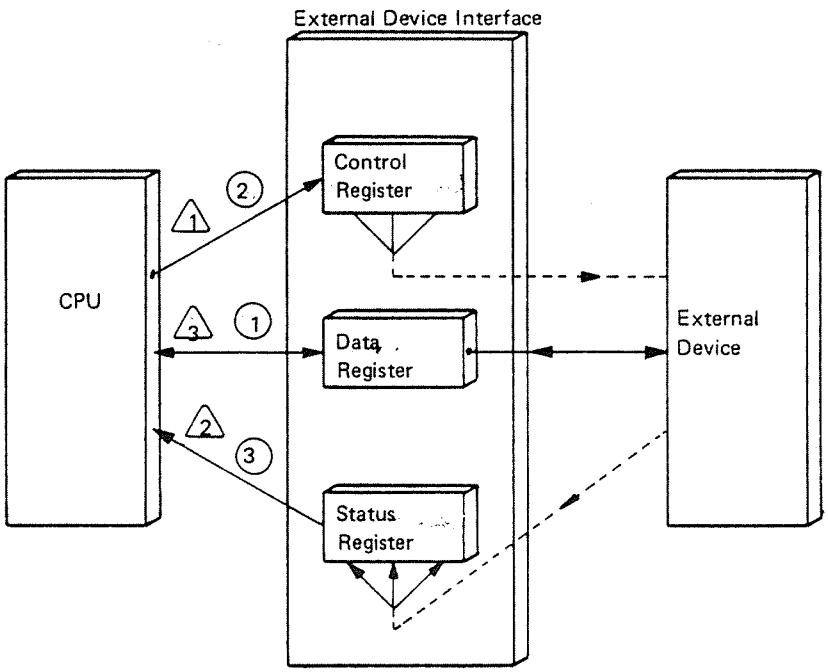


Figure VI.7.1: Loading Sequence for Device Registers

The control register can only be loaded from the CPU and the status register can only be read to the CPU. The data register has a two-way communication with the CPU.

If the external device is a two channel device (input and output), another set of registers is required. (In the illustration, the same set is used.)

Input:

1. The control register must be loaded. The control register will enable the input transfer.

2. The status will change when the external device transfers data to the data regiser. By reading and analyzing the status register the CPU will discover this.

3. The input of the data register will take place.

Output:

(1) Output of the data register is performed.

(2) The control register is loaded to tell the external device to take the data register from the interface.

(3) When the data transfer has taken place, the status will change, and by reading and analyzing the status reguster, the CPU will discover that a new transfer can be initiated.

VI.7.2 *PROGRAMMING EXAMPLE OF A NONINTERRUPTING I/O ROUTINE*

A PIO device may be driven either by an interrupt controlled driver routine or by a driver which continuously senses status for the correct responses after initiation of transfer.

The following programming example shows how a noninterrupt controlled driver reads a character from teletype 1 with echo.

```
START,   SAA 4              % A reg. bit 2 = 1, activate device
         IOX 303            % Load control word reg.
                            % Bit 2 = 1 (activate read)
         IOX 302            % Read status reg.
         BSKP ONE 30 DA     % Is bit 3 = 1 (device ready for
                            % transfer)?
                            % If yes, skip one location.
         JMP * –2
         IOX 300            % Read data reg., char. in A reg.
         COPY SA DX         % Save character
         IOX 306            % Read status reg. to check if ready for
                            % output
         BSKP ONE 30  DA    % If status bit 3 = 1 (ready for transfer)
                            % skip one location
         JMP * –2
         COPY SX DA         % Unsave character
         IOX 305            % Load data write reg., echo
         SAA 4              % A reg. bit 2 = 1, activate device
         IOX 307            % Load control word bit 2 (transfer
                            % character to TTY)
         IOX 306            % Read status reg. for output device
         BSKP ONE 30 DA     % Char. transfer completed?
         JMP * –2
         JMP START          % Repeat
```

VI.8 NORD-100 BUS SIGNALS DURING IOX INSTRUCTIONS

VI.8.1 *IOX INPUT*

Figure VI.8.1 shows the different control signals and information which is present on the NORD-100 bus during the execution of an IOX input instruction.

When start execution of the IOX instruction, the 11 bit device register address is sent out on the NORD-100 bus, together with the control signal "BAPR" — bus address present — from the bus control in the CPU. This signal tells all devices that a device address is present on the NORD-100 bus. This bus address is compared with the device register addresses on the different modules, to find the right one. Then the "BIOXE" signal is issued from the bus control. This is a strobe signal to enable the data transfer to or from an I/O device. The actual device interface will not send a "BINPUT" signal, telling that this is an input request from a device. The bus control will, as an answer to "BINPUT", send out the "BINACK" — input acknowledge. This signals that the interface requesting an input operation may enable data.

The 16 bits data word from the data register on the interface will now be enabled onto the NORD-100 bus. The interface will then issue the "BDRY" — bus data ready — signal, telling that data is ready on the NORD-100 bus. Data on the bus will be enabled into the A register, and the bus cycle will be terminated.

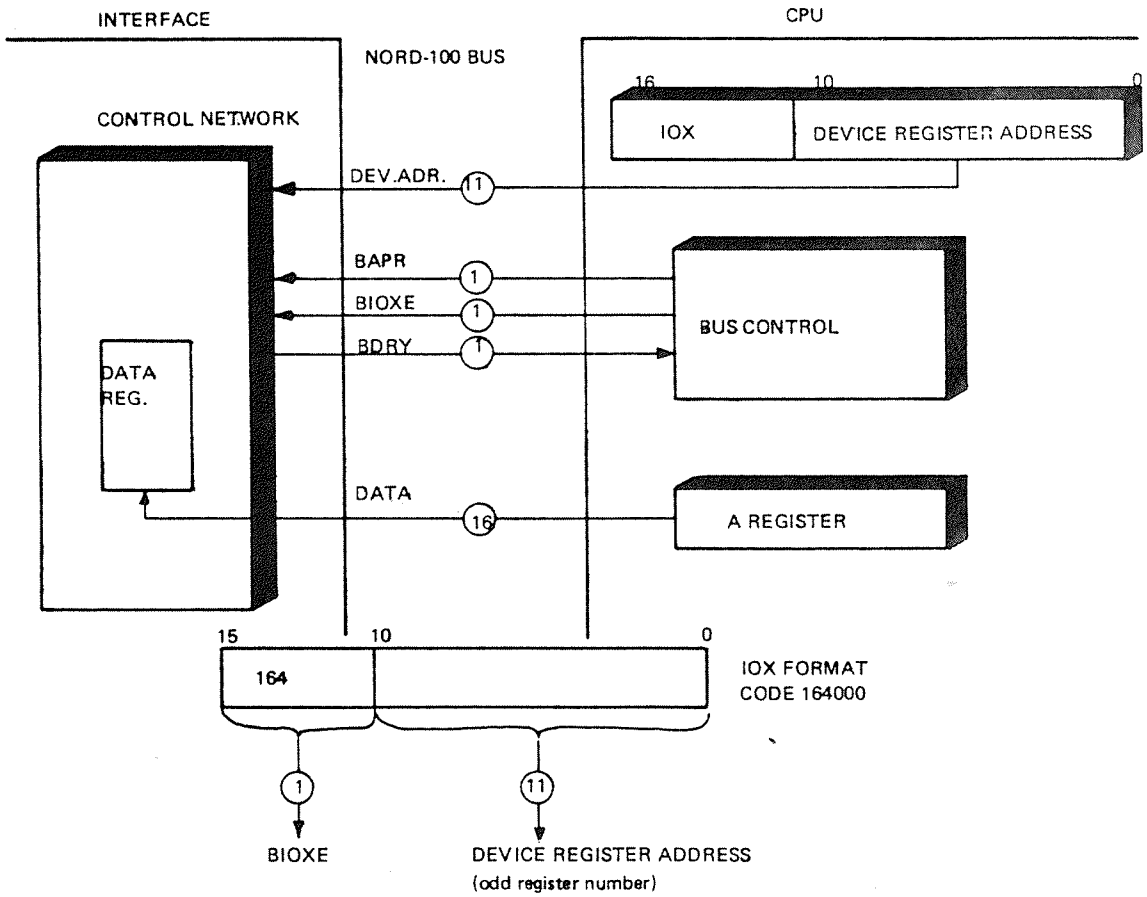


Figure VI.8.1: The Control Signals and Information Flow During an IOX Input Operation

VI.8.2 *IOX OUTPUT*

Figure VI.8.2 shows the different control signals and information which is present on the NORD-100 bus during the execution of an IOX output instruction. The 11 bit device register address is ent out on the NORD-100 bus under execution start. The "BAPR" – bus address present control signal tells all interfaces connected, that a device address is present on the bus. The device register address on the bus is compared to the device register addresses on the different modules, to find the right one.

Then the "BIOXE" signal is issued from the bus control. This is a strobe signal to enable the data transfer to or from an I/O device. The 16 bits of data from the A register will now be enabled onto the NORD-100 bus.

When data is accepted on the device interface, the "BDRY" – bus data ready control signal is issued from the interface. This signal will terminate the bus cycle.

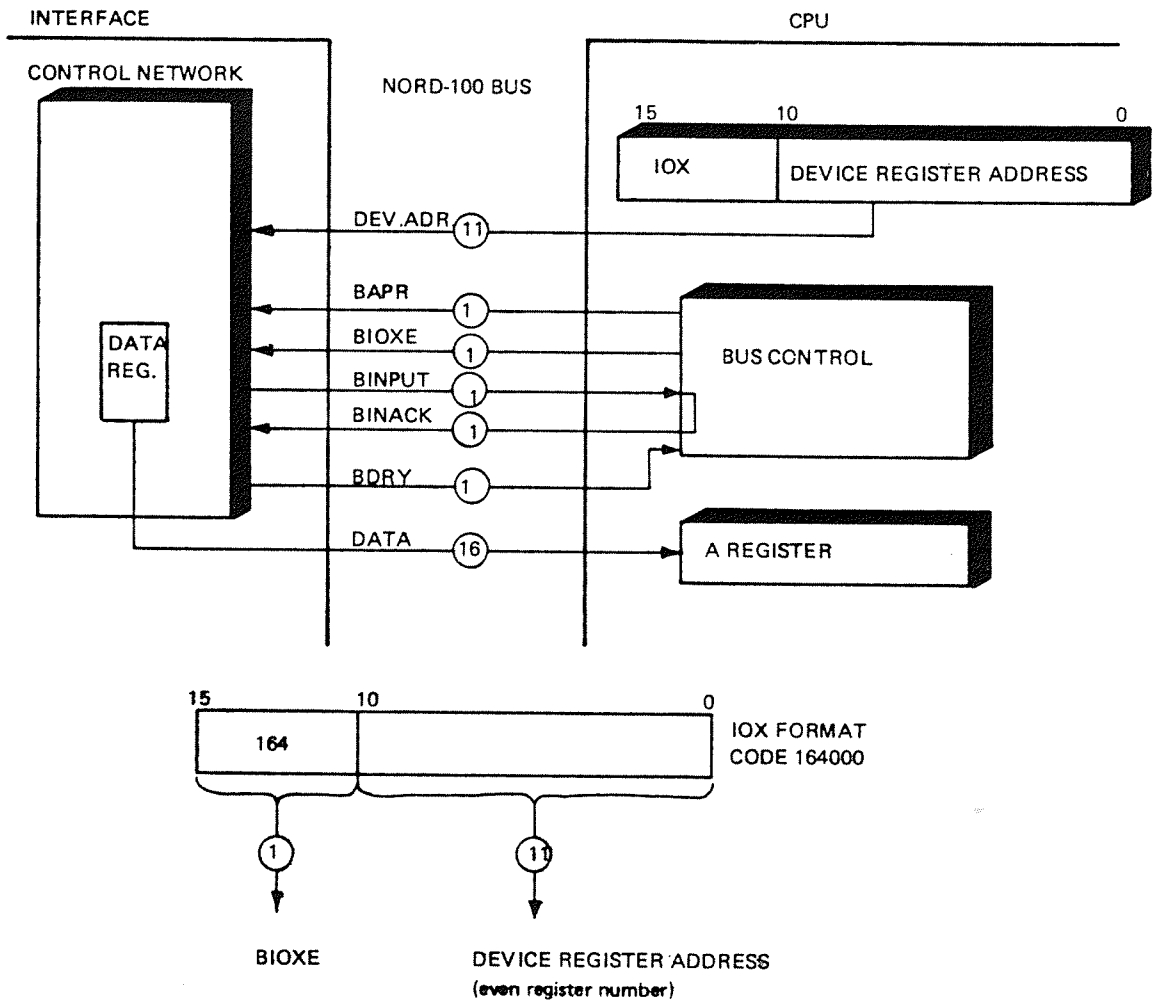


Figure VI.8.2: Control Signals and Information Flow during an IOX Output Operation

If "BDRY" is not received in the bus control within 10 μs after starting the instruction, an interrupt will occur. This is called timeout. The cycle is terminated, and an internal interrupt, IOX ERROR, is sent to the interrupt system. This is true both for IOX input and output.

The different control signals and information will be the same for the IOXT instruction, except for the device register address. This is taken from the T register as 16 bits.

VI.9 THE I/O SYSTEM'S CONNECTION TO THE INTERRUPT SYSTEM

VI.9.1 *GENERAL*

Under a running system (SINTRAN III), all I/O devices connected to the NORD-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

Which of the two status changes that actually caused an interrupt is found by reading the status register of the interrupting channel on the interrupting device.

VI.9.2 *THE I/O DEVICE CONTROLLER LEVELS*

Interrupt level 10-13 and 15 may be activated by hardware through physical lines available in the NORD-100 bus. These lines go directly to the priority interrupt controller (PID register) in the CPU. For Norsk Data produced equipment, the use of these lines have been standarized:

— 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

VI.9.3 *DEVICE INTERRUPT IDENTIFICATION*

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

When an IDENT <PL> instruction is executed a hardware search is performed on the level indicated by the lower 6 bits of the instruction. The first device found on the indicating level having an interrupt condition, will respond with a 9 bits identification code, to the A register.

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

There is correspondence between an external device, the device number and the ident code. For Norsk Data produced interfaces this has been standarized, and is given in Appendix C. The IDENT <PL> instruction will only search for interrupts on the levels specified in PL (10-13).

The Interrupt Sequence:

1. An interrupt condition occurs in a device. The local interrupt FF sets driving the interrupt line (10, 11, 12 or 13).

2. Provided the CPU is operating on a lower level, the CPU is forced to the interrupting level.

3. An IDENT instruction is issued. The IDENT CODE is received in the A register. The IDENT instruction resets the interrupt condition on the interface.

4. Using the vector as a branch address, the CPU will enter the device driver.

5. To analyze the reason for the interrupt, the status register is read.

6. The actual routine is executed ending with a WAIT instruction giving up the priority.

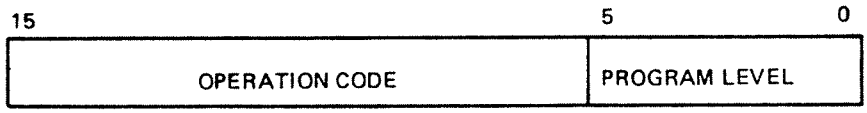
7. The CPU will resume the main program.

On the interfaces, the ident code (as the device number) is selectable by a thumbwheel to allow equal hardware modules to cover all ident codes related to one kind of peripheral.

VI.9.4 *THE IDENT INSTRUCTION*

VI.9.4.1 *The Ident Instruction Format*

IDENT <PL>



The ident instruction is a privileged machine instruction used in device interrupt identification, and when executed, searches for interfaces with interrupt condition set, and returns the interfaces' ident code to the A register.

To maintain the interrupt priority the ident instruction searches only for interrupts on a specified level. The level to search on is specified in the ident instruction format.

The instruction IDENT PL12 will only search for interfaces driving interrupt line level 12 (BINT12). A possible existing interrupt on level 10 or 11 is ignored and handled later by IDENT PL10 and IDENT PL11 respectively.

VI.9.4.2 *NORD-100 BUS SIGNALS DURING IDENT INSTRUCTIONS*

Figure VI.9.1 shows the control signals and information flow during an ident instruction.

The interrupt lines on levels 10 to 13 (BINTxx) will set the proper bit in the priority interrupt detect register (PID register). The level will be changed to the interrupted one, and an ident instruction will be issued with current level as <PL>.

A six bits program level code (refer to Appendix A for more about PL) describing the level, is enabled onto the NORD-100 bus. The "BAPR" — bus address present — signal is issued to tell all connected devices that a level code is present on the bus.

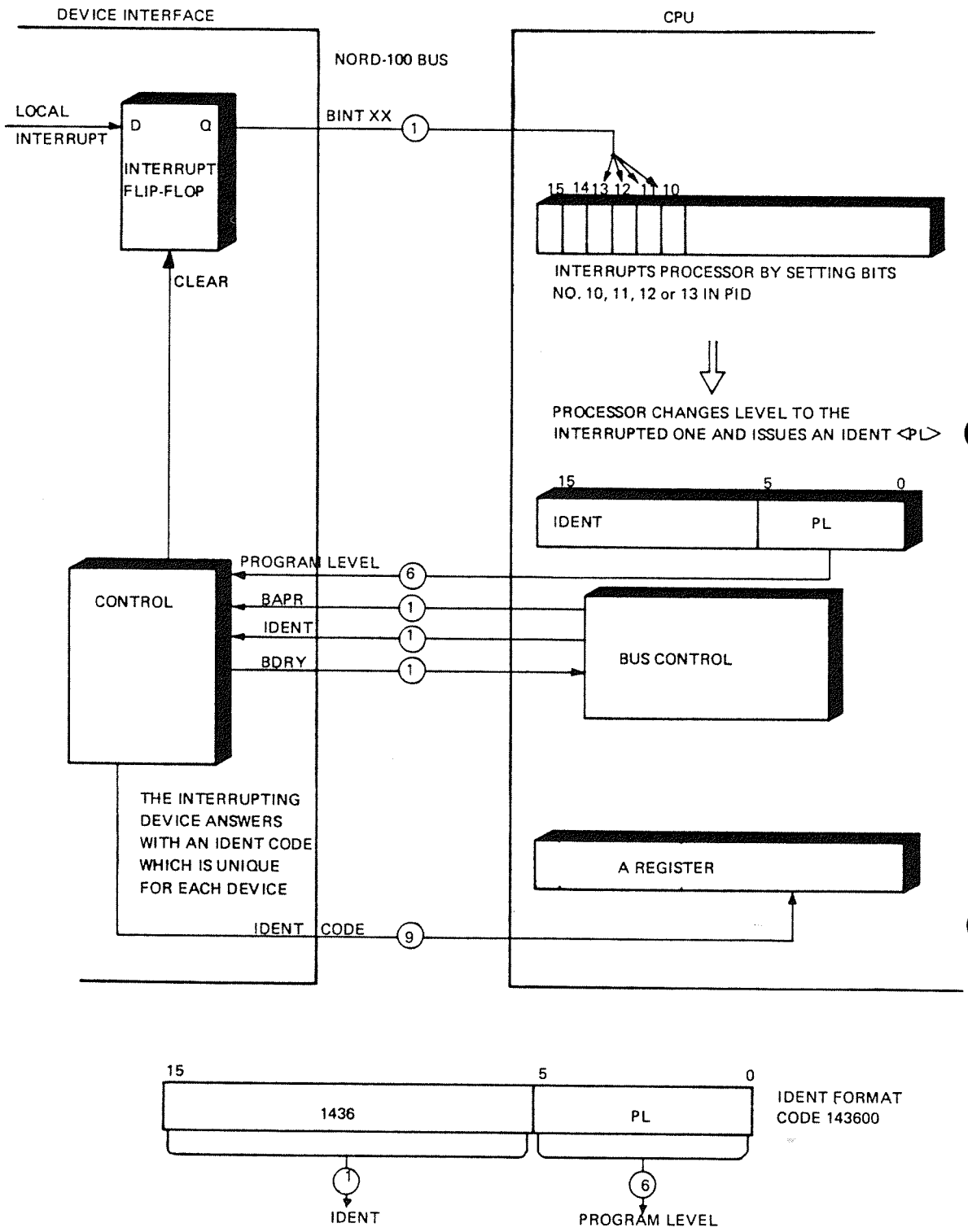


Figure VI.9.1: Control Signals and Information Flow During an IDENT Instruction

Then the "IDENT" signal is generated. This search signal is daisy chained via the module nearest to the CPU, over to the next, and so on. The "IDENT" signal is stopped by the interrupting device interface.

The interrupt condition is cleared on the interface by clearing the interrupt flip-flop holding the interrupt line.

Then the ident code is sent to the A register in the CPU, together with the control signal "BDRY" — bus data ready. This signal tells the CPU that data is ready on the NORD-100 bus, and the ident code is strobed into the A register. Then the cycle is terminated.

There should never be empty positions in the NORD-100 bus between the CPU and any I/O device controller. An empty position will stop the search signal and never release interrupts on modules in higher slot position numbers than the empty one.

Between interfaces generating interrupt on the same level, the interface nearest the CPU has highest priority within the level.

VI.9.5 *PROGRAMMING INPUT/OUTPUT USING INTERRUPT*

Programming input/output by using waiting loops, as shown in Section VI.7.2 is very ineffective. Most of the computer time will be spent in the input/output loops. This will be avoided by utilizing the interrupt system. An interrupt will occur every time the device is ready for transfer.

Example:

Interrupt controlled driver for reading and printing a character on teletype. Device register number for terminal number 1 is given in brackets ( ).

Initialize the interrupt system on levels 0, 1, 10 and 12.

```
LEV0,       SAA2
            MST PID          % Generate interrupt to level 1
            JMP * 0
LEV1,       SAA7             % Set bit no. 0, 1 and 2 in A reg.
            IOX CONTW        % Set control reg. for input (303)
            WAIT             % Give up priority
            SAA7             % Set bit no. 0, 1 and 2 in A reg.
            IOX CONTW        % Set control reg. for output (307)
            WAIT             % Give up priority
            JMP LEV1
LEV10,      IDENT PL10       % Identify interrupt
            CHECK IDENT CODE
            LDA BUFF         % Get saved data
            IOX WDATA        % Load data output reg. (305)
            IOX STATUS       % Read output status reg. (306)
            BSKP ZERO 40 DA  % Check the error bit (no. 4)
            JMP ERROR
            SAA 2            % Set bit no. 1 in A reg.
            MST PID          % Generate interrupt to level 1
            WAIT             % Give up priority
            JMP LEV10
LEV12,      IDENT PL12       % Identify interrupt
            CHECK IDENT CODE
            IOX STATUS       % Read input status register (302)
            BSKP ZERO 40 DA  % Check the error bit (no. 4)
            JMP ERROR
            IOX RDATA        % Read data (300)
            STA BUF          % Save data
            SAA 2            % Set bit no. 1 in A reg.
            MST PID          % Generate interrupt to level 1
            WAIT             % Give up priority
            JMP LEV12
BUFF, 0
ERROR, JMP * 0
```

Comments:

LEV1:

The control register for input enable for interrupt on device ready for transfer and interrupt on errors, and activate device. Wait will give up priority. If a key on the terminal is pushed, an interrupt on level 12 is generated.

LEV12:

The error bit in the status register is checked. If no errors, data is read and saved in the location with label BUFF. An interrupt to level one is generated.

LEV1 (second time):

The control register for output is loaded. Bit number 0 will enable for interrupt on device ready for transfer. A level 10 interrupt will then be generated from the device.

LEV10:

Data is loaded from BUFF into the A register. The data register for output is loaded. The quality of the data transfer is checked by reading the status register for output. An interrupt to level one is generated.

VI.10 DIRECT MEMORY ACCESS

VI.10.1 *GENERAL*

Direct memory access (DMA) is used in data exchange between a high speed block oriented peripheral and the NORD-100 memory system. A DMA controller is connected directly to main memory via the NORD-100 bus, called a DMA channel.

In NORD-100 all DMA controllers have at least 16 words buffering 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.

In a DMA output operation, the DMA controller gives address to memory and data is sent back to the controller from memory.

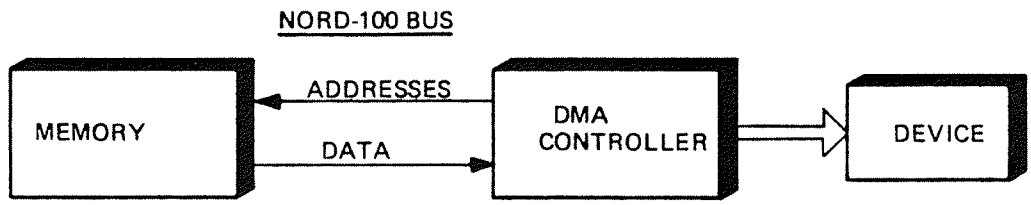


Figure VI.10.1: DMA Output

In a DMA input operation, the DMA controller gives both addresses and data to memory.

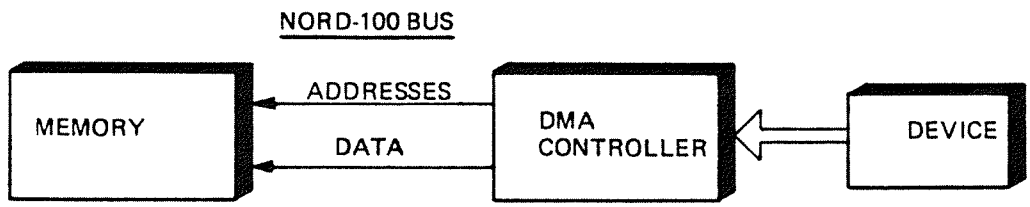


Figure VI.10.2: DMA Input

A DMA transfer may be divided into 3 steps:

- Initialization
- Transfer
- Termination and status check

VI.10.2 *INITIALIZATION*

A DMA controller has to be initialized before a transfer can be started. The initialization is done by a device driver program activated by the operation system when a 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. The transfer parameters are written into physical device registers located on the controllers.

Typical transfer parameters and registers 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 (DMA input) from or write (DMA input) to on the physical device (for example, disk).

— Word count register holds the number of words to be transferred.

— Control register gives device function (read, write, etc.) and start (bit 2 — activate device).

In addition, a status register, the MAR and the BAR, may be read for status check and test purposes.

The format of the registers and their associated register numbers are given in the hardware programming specifications.

VI.10.3 *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.

Two of the registers on the DMA controller are updated dynamically as each word is exchanged between the DMA controller and memory. The word count register is decremented, and the memory address register is incremented.

VI.10.4 *TERMINATION AND STATUS CHECK*

The DMA transfer is completed when the word counter is zero. On the DMA controller, status register bit 3 (ready for transfer) is turned on. If the interrupt system is on (ION) and interrupt is enabled on the controller, this causes interrupt on level 11. If the interrupt system is not on, a completed transfer is found by polling (continuous reading) of status register bit 3. The device driver is again activated to read the device status which gives informatoin on the status of the transfer.

VI.10.5 *NORD-100 BUS SIGNALS DURING A DMA TRANSFER*

VI.10.5.1 *DMA INPUT*

Figure VI.10.3 shows the different control signals and information which is present on the NORD-100 bus during a DMA input transfer. The DMA controller will start with a "BREQ" – bus request – signal to the bus control in the CPU, requesting for a DMA cycle. The bus control will issue a "BMEM" bus memory cycle, to memory to signal that a bus cycle accesses memory.

An "INGRANT" signal is also issued. This is a response to "BREQ", indicating that the bus is available for a DMA cycle. An interface which issued "BREQ" may use the bus for a single memory write cycle. The interface will stop the "INGRANT" signal. Otherwise, the "INGRANT" is passed onto "OUTGRANT" which is connected to "INGRANT" of the next lower priority card position, which possibly had issued the "BREQ" signal.

The "BINPUT" signal is issued from the requesting controller, telling memory that this is an input operation. Then the address from the memory address register is enabled onto the bus, and "BAPR" – bus address present – is issued to tell memory that the 24 bits address is present on the NORD-100 bus. This is used for enabling of addresses to the memory modules.

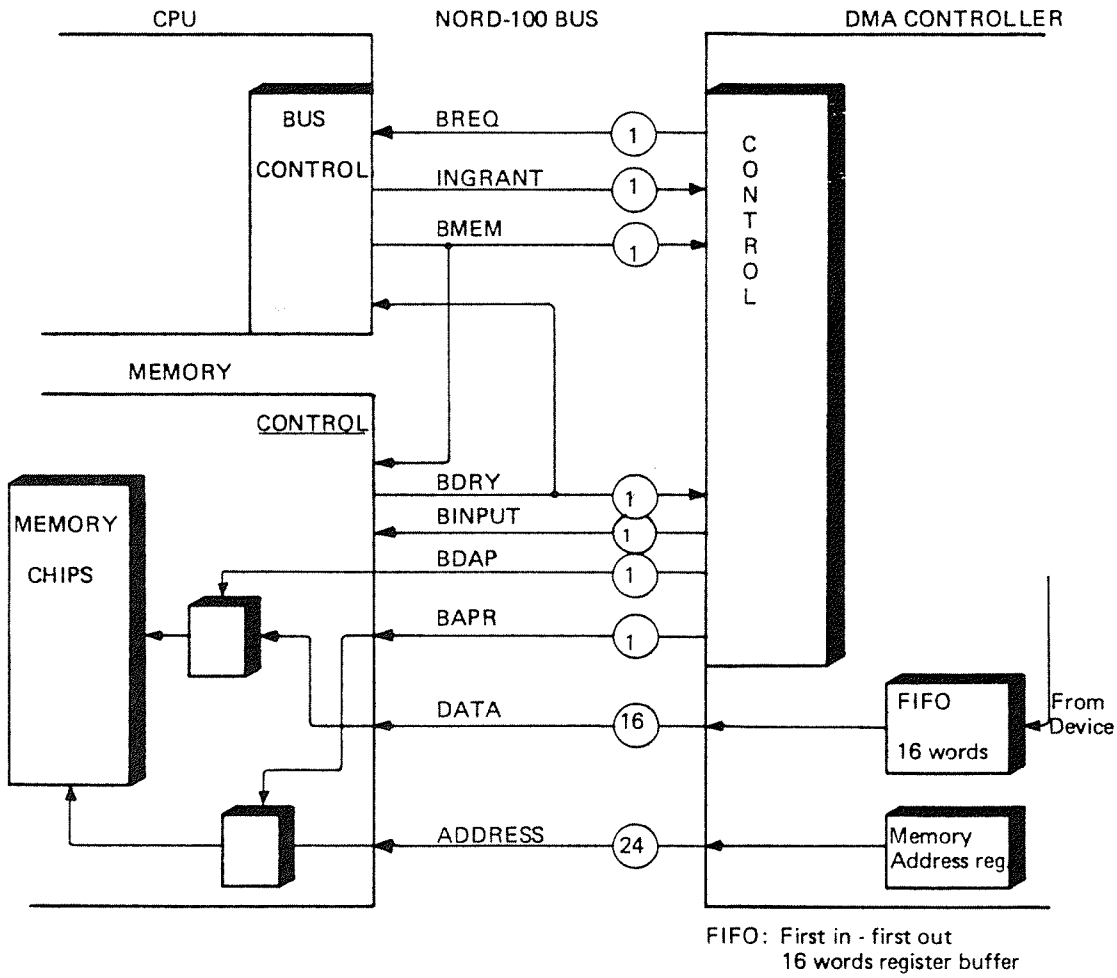


Figure VI.10.3: NORD-100 Bus Signals During a DMA Input Transfer

Then the 16 bits of data will be present on the NORD-100 bus, together with the "BDAP" — bus data address present — signal. This is used to latch the present data on the memory module. The data will now be written into the memory chips to the already decoded address.

"BDRY" — bus data ready is issued from memory for termination of the cycle.

VI.10.5.2 *DMA OUTPUT*

The actual DMA controller will send a "BREQ" signal to the bus control in the CPU, requesting for a DMA cycle.

"BMEM" will be issued to signal that a bus cycle accesses memory.

"INGRANT" will be issued from the bus control as a response to "BREQ", indicating that the bus is available for a DMA cycle. This signal will go to the first module in the crate, which will send it out as "OUTGRANT" if this module has not generated "BREQ". "OUTGRANT" is connected to "INGRANT" for the next lower priority card position. The interface issuing "BREQ" will stop "INGRANT", telling this interface that it may use the bus for a single memory read cycle.

The DMA controller will send out a 24 bits address from the memory address register to the NORD-100 bus. A "BAPR" — bus address present — signal is issued to tell the memory modules that the address is present on the bus. This address will be enabled to the memory modules, finding the memory location.

"BDAP" — bus data present — is issued from the DMA controller, telling memory that the accessed data may be enabled to the NORD-100 bus. Sixteen bits of data are then enabled to the NORD-100 bus, and into the FIFO (first in-first out) register buffer on the DMA controller.

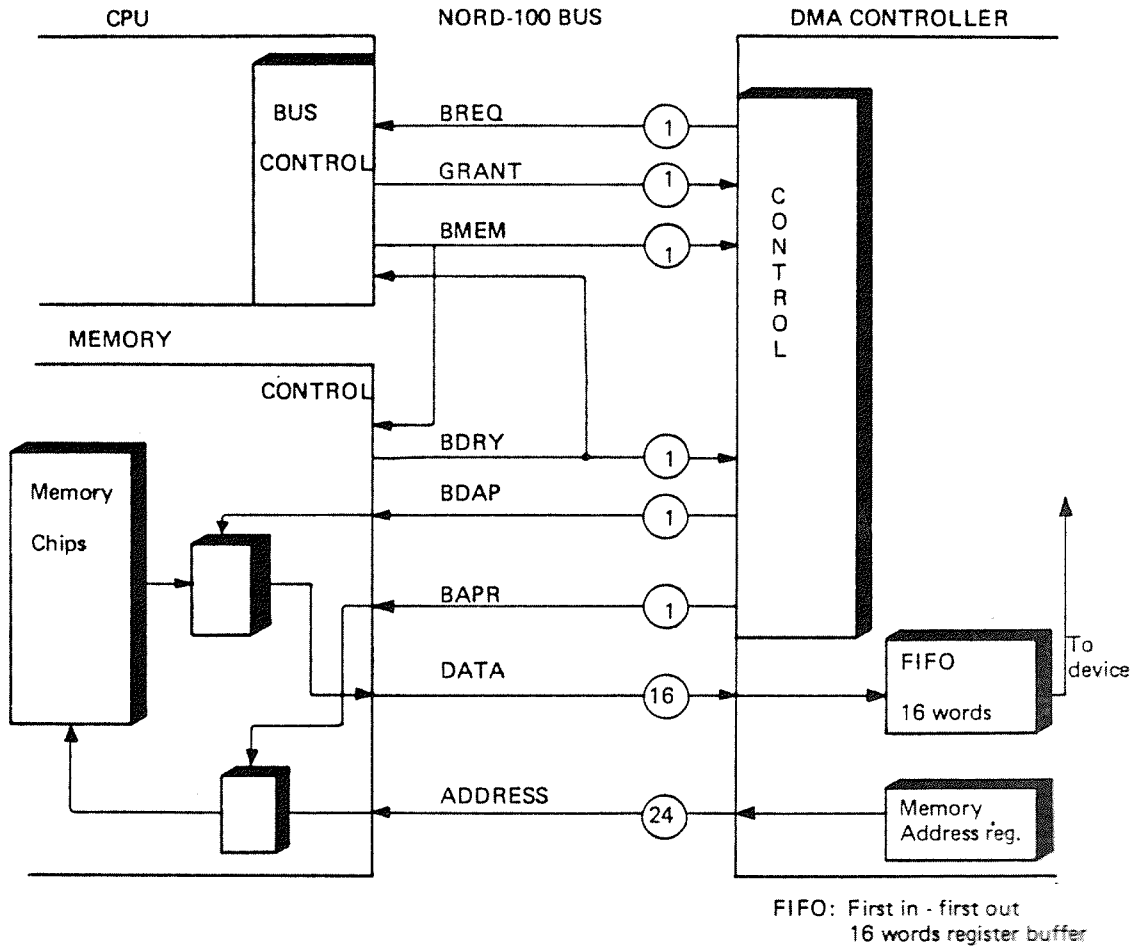


Figure VI.10.4: NORD-100 Bus Signals During a DMA Output Transfer

"BDRY" — bus data ready, is then issued for termination of the cycle.

VI.10.6 *PROGRAMMING OF A DIRECT MEMORY ACCESS CHANNEL – DMA*

DMA device controllers are initialized by PIO. When the controller has been properly initialized, the transfer is started by loading the control word register with the activate code.

Example:

The following is an example of how a disk transfer may be programmed:

```
READ STATUS                 % Read disk status register
CHECK STATUS                % and check if ready and on cylinder
LDA UMEMADR
IOX MEMADDR                 % Load most significant part (8 bits) of
                            % memory address register (MAR)
LDA LMEMADR
IOX MEMADR                  % Load least significant part (16 bits)
                            % of memory address register (MAR)
LDA WORDCNT
IOX WORDCNT                 % Load word count register
LDA DISKADR
IOX BLCADR                  % Load block address reg. (BAR)
LDA CONTRW
IOX CONTRW                  % Load control word register
    .                       % Enable for interrupt upon transfer
    .                       % completion, select unit, specify read
    .                       % or write and active device
INTERRUPT ON
COMPLETION OR
ERROR:
IDENTIFY INTERRUPT BY READING STATUS
```

Check status register for normal completion. Read memory address register. Memory address register before transfer added to the word count register, should be equal to the memory address register after transfer completion.

For further information and description of a DMA interface, refer to Appendix B.

VI.11 PROGRAMMING SPECIFICATIONS FOR I/O DEVICES ON THE CPU BOARD

The real-time clock (device numbers 10-13) is always located on the CPU board. The terminal with device numbers 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.

VI.11.1 *THE CURRENT LOOP INTERFACE*

The current loop interface, located on the CPU board, has device numbers 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 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 tive.

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 give 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.

VI.11.2 *THE REAL-TIME CLOCK*

The real-time clock on the CPU board has device numbers 10-13.

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 instruciton is executed repeatedly, the counter will never be incremented, and no clock pulses will occur. This may affect the execution of operator's communication 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.

VI.12 NORD-10/S MODULES USED IN NORD-100

Even if the NORD-10/S modules have a different size, it is possible to use them in the NORD-100 computer system.

On a special NORD-100 module, two standard NORD-10/S PIO (programmed input/output) modules may be plugged in, and adapted to the NORD-100 bus system. By help of this module, the wide range of PIO modules from NORD-10/S may be used in the NORD-100.

SECTION VII

OPERATOR'S INTERACTION

VII.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 VII.2.1), when the CPU goes through the power up sequence, or when the bus line called MCL is activated by an interface.

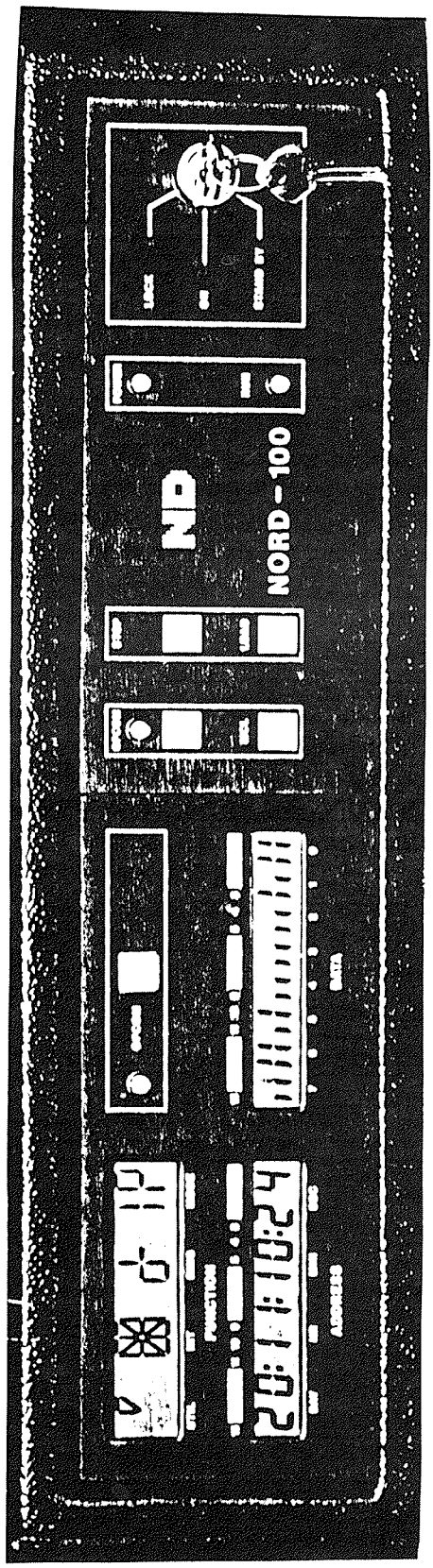
The master clear routine turns off the green indicator lamp, then the PIE register is cleared. The paging and interrupt systems are turned off. The paging system is set in REX mode. Subsequent memory examine functions with MOPC are set to 24 bit physical examine mode. The CPU self test microprogram is 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.


ND
NORD – 100

VII.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 macines delivered from January 1980.

VII.1.2 *STATUS INDICATORS*

POWER ON

Indicates that +5V is present in the rack.

RUN

Indicates that the CPU is running.

OPCOM

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.

VII.2 MICROPROGRAMMED OPERATOR'S COMMUNICATION

VII.2.1 *GENERAL CONSIDERATIONS*

The NORD-100 has a microprogram in the read only memory for communication between the operator and the machine. This program is called MOPC (Micro-programmed Operator's Communication) and is used for operational control of the NORD-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 NORD-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).

VII.2.2 *CONTROL FUNCTIONS (Does not affect display)*

VII.2.2.1 *System Control*

VII.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 - IV.1.

VII.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.

VII.2.2.1.3 ALD LOAD

In the following table the different columns signify:

ALD — Position of the ALD thumbwheel switch on the CPU board (refer to Appendix A).

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 $ 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 load from 500	Mass load from 500
12	21540	Start in address 20	Mass load from 1540	Mass 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 from 500	Mass storage from 500
4	121540	Mass storage from 1540	Mass storage from 1540	Mass storage 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

VII.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".

VII.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).

VII.2.2.2 *Program Execution*

VII.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.

VII.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).

VII.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.

VII.2.2.2.4 Instruction Breakpoint

xxxxxx.

This command starts execution in the same semi-RUN mode as described in Section IV.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.

VII.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.

VII.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 OPR register (see Section VII.2.3.3.5). Returned data is printed after the slash and not stored anywhere. No working registers are affected.

VII.2.2.3 *Miscellaneous Functions*

VII.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

VII.2.2.3.2 Delete Entry

When @ or ␣ (space) is typed, all characters written before this character are ignored.

VII.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.

VII.2.3 *MONITOR FUNCTIONS (Also shown on Display)*

VII.2.3.1 *Memory Functions*

VII.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.

VII.2.3.1.2 Virtual Examine Mode

nE↙

This command will change the examine 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.

VII.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 VII.2.3.2.1). 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
```

VII.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.
```

VII.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).

VII.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.

VII.2.3.2 *Register Functions*

VII.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.

VII.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

VII.2.3.2.3 Register Dump

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 register block should be printed, xx must be equal to yy.

Note the case: <RD↙ dump register block on level 0.

VII.2.3.2.4 User Register

U/

The last value written by TRR LMP, is selected as display source.

VII.2.3.2.5 Operator Panel "Switches"

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.

VII.2.3.3 *Internal Register Functions*

VII.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
```

VII.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	CCLR	Cache Clear
11	LCILR	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
```

VII.2.3.3.3 Internal Register Dump

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.

VII.2.3.3.4 Scratch Register Dump

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.

VII.2.4 *DISPLAY FUNCTIONS (Affects only display)*

VII.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 unary display.

- zz = $0\text{-}24_8$ — Position of lowest bit position to be represented in unary 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. (If the display shows an address, this is equivalent to push the DECODE ADDRESS button on NORD-10/S.)

VII.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.

VII.2.4.3 *Display Activity*

ACT/

With this display mode active levels, clock and indicator functions are displayed.

VII.2.5 *BOOTSTRAP LOADERS*

The NORD-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 NORD-100.

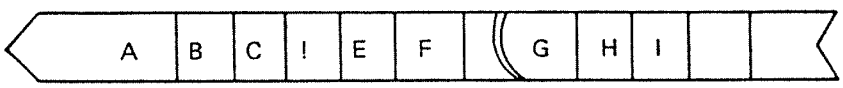
VII.2.5.1 *Binary Format Load*

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.

VII.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".

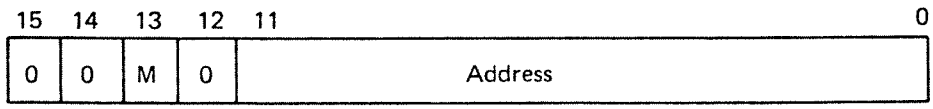
When loading from mass storage, 1K words will be read from mass storage address 0 into main memory starting in address 0. After a successful load, the CPU is started in main memory address 0.

The mass storage device must conform with either drum or disk programming specifications.

VII.2.5.3 *Automatic Load Descriptor*

The NORD-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.

VII.3 THE DISPLAY

VII.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 has the same function as the other "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.

More information about controlling the display processor is found in Section IX (Panel Processor Programming Specification) and in Appendix E (Microprogram Panel Processor Communication).

VII.3.2 *THE DIFFERENT DISPLAY FUNCTIONS*

Figure VII.3.1 shows the normal activity on the display when the machine is running.

The DATA field displays information in binary or octal format (see Section VII.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 NORD-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 (refer to the section on Power Supply), 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 VII.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

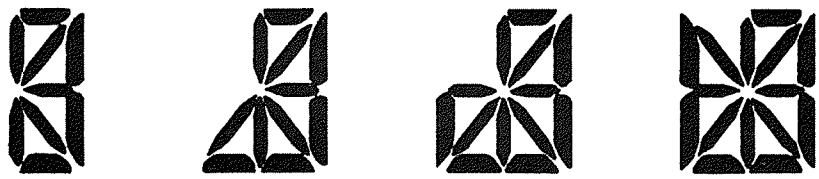
UTIL, utility of the machine, is shown here. That is, how much time the machine spends on level 0 (idle). The more utility, the less the time spent on level 0 and more segments on the display is lit up.

Example:

No Activity.

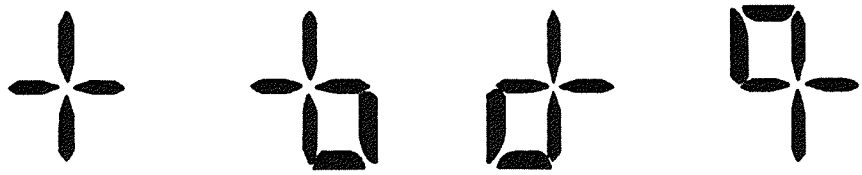
HIT, tells the hit rate in cache memory. The more hit in cache, the more segments are lit up on the display.

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.

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.

SECTION VIII

NORD-100 POWER SUPPLY

VIII.1 NORD-100 POWER UNIT

VIII.1.1 *GENERAL*

The NORD-100 power unit is located in the card crate as illustrated below:

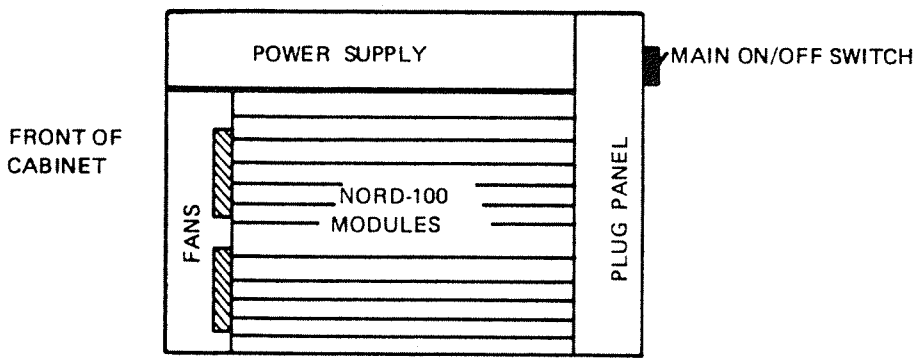


Figure VIII.1.1: The NORD-100 Card Crate Including Power Supply Unit (top view)

The NORD-100 power supply comprises three voltage regulators with transient voltage indicators as well as a rechargeable battery with a charging circuit.

VIII.1.2 *5V/50A MAIN POWER SUPPLY*

The one voltage regulator with 5V output and a load current of maximum 50A is a direct off-line switching type regulator, used for CPU, I/O and memory control logic.

VIII.1.3 *12V/2A AND 5V/7A STAND-BY POWER SUPPLY*

The two other regulators supplying 5V at 7A maximum load current and 12V at 2A maximum load current respectively, are also switch type regulators. These two latter regulators are, however, supplied by a voltage derived by means of an ordinary 50Hz mains transformer and rectifier.

In case of short mains interruptions, the latter voltage regulators are supplied by the rechargeable battery. This, so-called stand-by power, is used to feed refresh pulses to MOS memory under a power break. The control logic in the CPU and memory, covering the refresh logic, are connected to these regulators, in addition to the memory chips.

VIII.1.4 *STAND-BY POWER HOLDING TIME*

In case of power failure the voltages from the stand-by power will be present for minimum 18 minutes with full load on both outputs (5V and 12V) and fully charged battery. The voltage supplied from the battery may drive up to 8 modules of 64 K words each.

VIII.1.5 *TRANSIENT VOLTAGE INDICATORS*

On the one side of the crate, there are three red light emitting diodes (LEDs) which are normally not lit (refer to Figure VIII.1.2). These diodes are activated if appropriate voltage exceeds nominal voltage by 10% or drops 10% below nominal.

All three voltage regulators are equipped with marginal setting switches, located beside the three LEDs. When activated, these switches cause the output voltage to shift by 5%, and the diodes to be lit up.

Upper output voltage:	5.25±0.025V	12.6V ± 0.06V
Lower output voltage:	4.75± 0.025V	11.4V ± 0.06V

This is used for maintenance only.

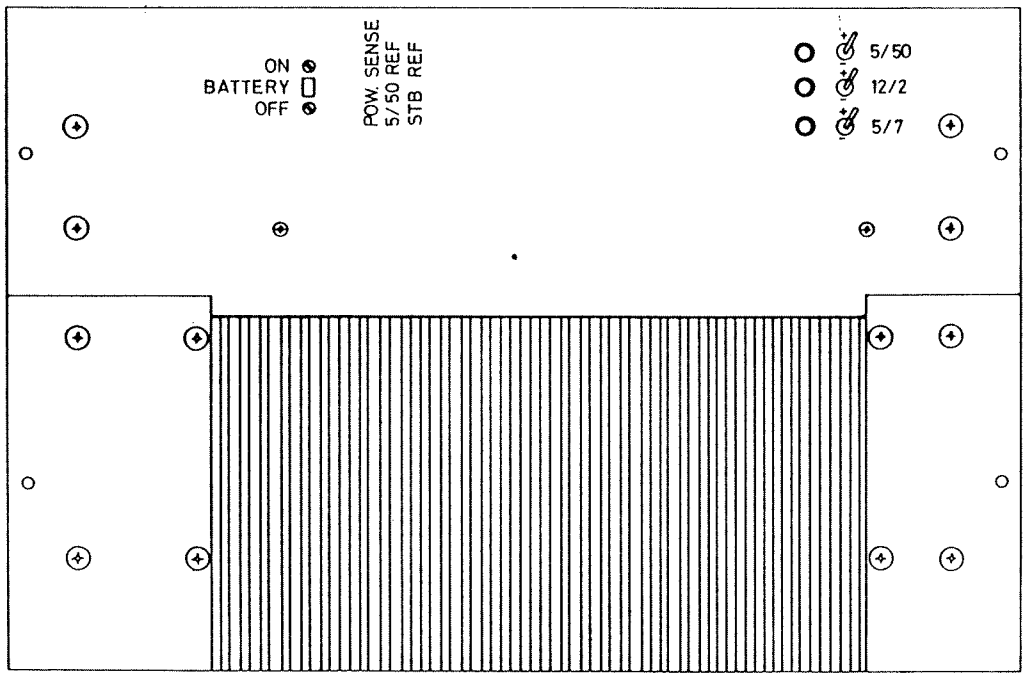


Figure VIII.1.2: The Power Supply Side of the NORD-100 Card Crate

VIII.1.6 *ADJUSTMENTS*

Refer to Figures VIII.1.2 and VIII.1.3.

POWER SENSE:	Power failure threshold adjustment. Factory adjusted.
5/50 REF:	Adjustment of the reference voltage (5.50V) for the stand-by regulator.
12/2 ADJ:	Adjustment of the 12 volt stand-by power supply.
5/7 ADJ:	Adjustment of the 5 volt stand-by power supply.
5/50 ADJ:	Adjustment of the main 5 volt power supply.
BATTERY:	Normally in ON position. Connects/disconnects the charge circuits for voltage back-up batteries.

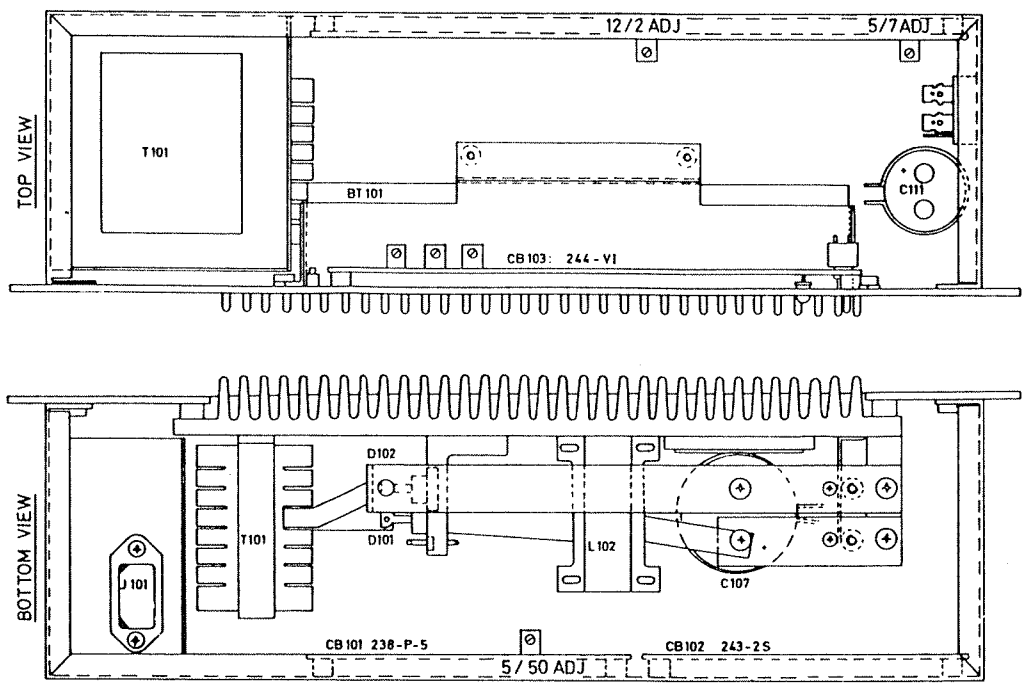


Figure VIII.1.3: Top and Bottom View of the Power Supply

VIII.2 POWER FAIL AND AUTOMATIC RESTART

VIII.2.1 *GENERAL*

The power fail unit is physically located in the power supply. The purpose of the power fail unit is to detect the presence of the input voltage (220V AC) and give an early warning to the CPU in case of power failure. This early warning is given through the internal interrupt system by a power fail interrupt.

When notified that a power fail is in progress, the operating system will make the necessary steps towards a well defined stop point with its registers saved in memory. When the main power is restored, sensed by the Power Fail Unit, the operating system will go through a restart procedure enabling the executing programs to resume.

VIII.2.2 *POWER FAIL*

In the power fail unit the input voltage (220V AC) is full-wave rectified and applied to an RC network with a time constant of approximately 8 ms, located on the CPU module.

Each 10 ms (half period) the capasitor is charged to the peak voltage, and then allowed to discharged towards a sense level given by a schmitt-trigger, which is set up to approximately half the peak value.

At normal operation, the capasitor is recharged again before reaching the sense level. The power fail unit will then output a steady POWOK (power ok) signal. If the input voltage drops below a certain level (approximately 195V AC) or is missing completely, the capasitor will discharge below the sense level, driving the power sense line to high and giving a false POWOK signal (refer to Figure VIII.2.1).

A power fail interrupt is generated immediately and after the CPU has saved all its registers, the CPU will go into a defined stop condition. Then the MINH (memory inhibit) signal will be enabled. The MINH signal is used to prevent uncontrolled write operations to multiport memory and disks.

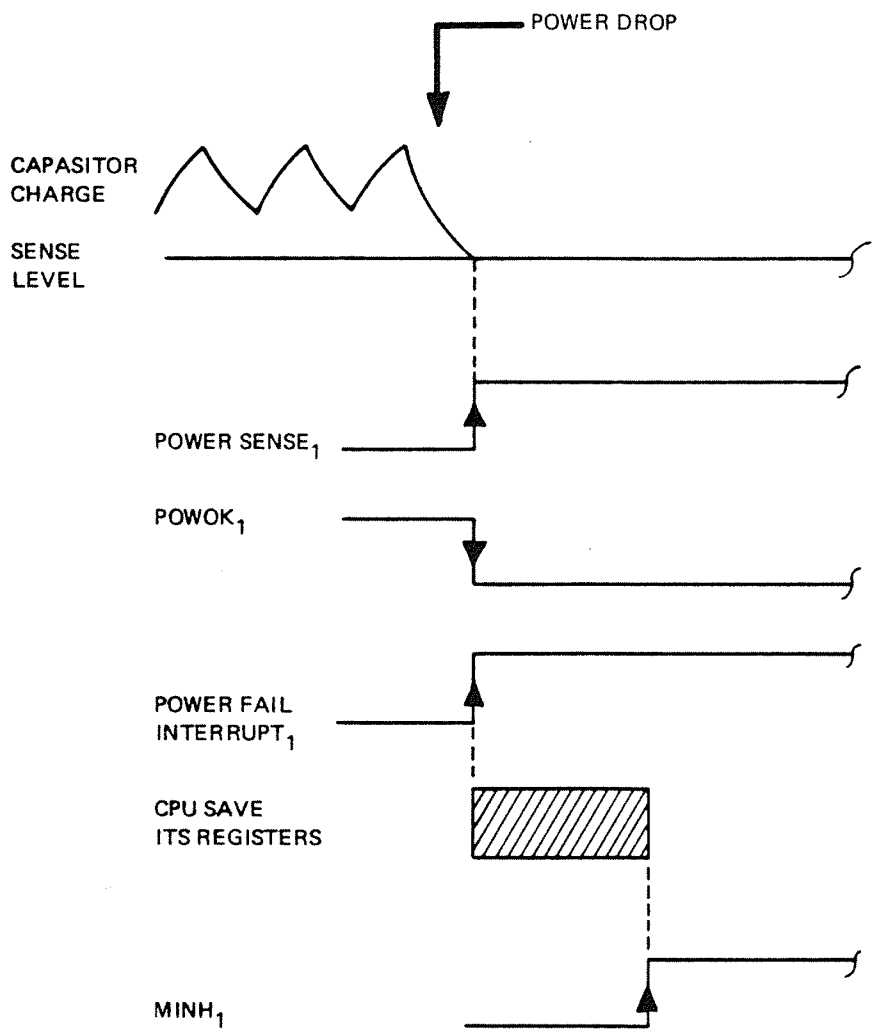


Figure VIII.2.1: Power Fail Sequence

VIII.2.3 *AUTOMATIC RESTART*

When power again is restored, the capacitor is recharged above the sense level and the power sense signal will go false. After a time delay of approximately 0.6 seconds, the POWOK is activated (refer to Figure VIII.2.2). The power fail interrupt line is reset together with the MINH signal.

The CPU will now enter the microprogram routine specified by the automatic load descriptor (ALD) if the panel is locked (refer to Chapter VI). An automatic restart or an automatic load may then be performed, dependent of th ALD thumbwheel setting which gives the adequate information about the operation. If the panel is not locked, the machine will enter the stop mode.

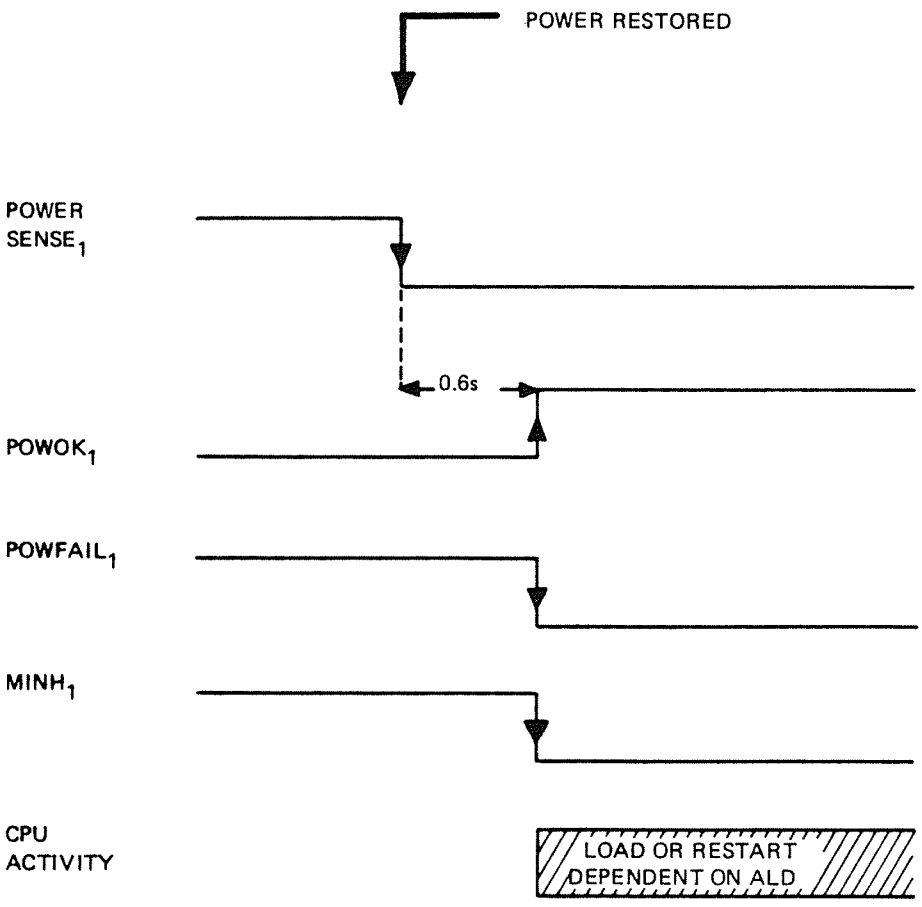


Figure VIII.2.2: Power Up Sequence

SECTION IX — MISCELLANEOUS

IX.1 PRIVILEGED INSTRUCTIONS

IX.1.1 *GENERAL*

The instructions termed privileged instructions are available only to:

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

IX.1.2 *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.

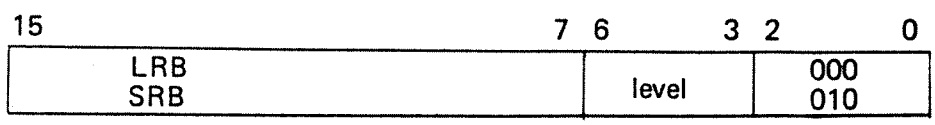
A register block always consists of the following registers in this sequence:

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: (EP), + 1 + 2 + 3 + 4 + 5 + 6 + 7
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.

IX.1.3 *INTER-LEVEL REGISTER INSTRUCTIONS*

In the NORD-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 or written, 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.

IX.1.4 *ACCUMULATOR TRANSFER INSTRUCTIONS*

The internal registers in NORD-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

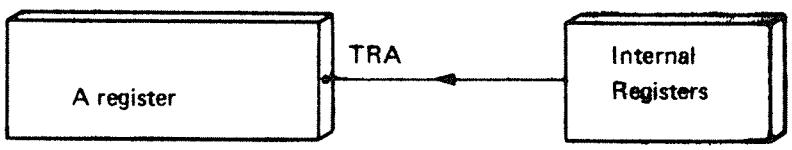
The internal registers controlled by these instructions are described in Section IX.2.

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

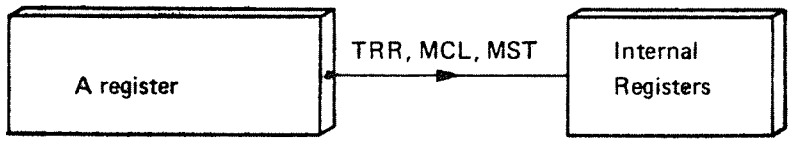
Format: TRR <register name>

The contents of the A register are copied into 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 opreate 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			
CCLR	10		X		
LCILR	11		X		
ACTL	11	X			
ALD	12	X			
UCILR	12		X		
PES	13	X			
PCR	14	X			
PEA	15	X			

IX.1.5 *SYSTEM CONTROL INSTRUCTIONS*

The following 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
OPCOM	Start MOPC

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

IX.1.5.1 *Interrupt Control Instructions*

The NORD-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 interupts
14	Internal hardware status interrupts
13 - 10	Vectored interrupts, maximum 2048 vectored interrupts
9 - 0	Software controlled 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. Refer also to the interrupt system in Chapter II.

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. These instructions are described in the accumulator transfer instruction in this section.

When power is turned on, the power-up sequence will reset PIE and the register set on program level zero will be used. Two instructions are used to control the on-off function of the interrupt system.

ION — Interrupt system on — Code: 150 402

Format: ION

The ION instruction turns on the interrupt system. At the time the ION is executed, the computer will resume operation at the program level with highest priority. If a condition for change of program levels exists, the 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 NORD-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 NORD-100 the identification codes are standarized for input/output devices delivered from Norsk Data.

IX.1.5.2 *Memory Management Control Instructions*

A full description of memory management is given in Section III. 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. PON is privileged

POF — Memory management off — Code: 150 404

Format: POF

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: 150 405

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.

IX.1.5.3 *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 OPCOM lamp on the operator's panel is lighted up. To start the program in the instruction after the WAIT, type ! (exclamation mark) on the console.

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 privileged.

IX.1.5.4 *Monitor Call Instruction*

MON Monitor Call Code: 153 000

Format: MON <number>

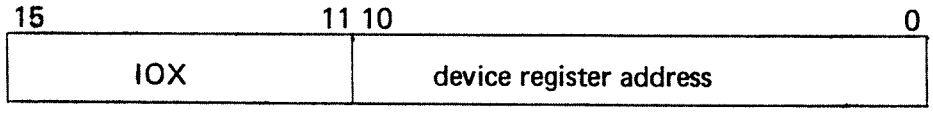
The instruction is used for monitor calls, and causes an internal interrupt to program level 14. The parameter <number> following MON must be specified between -200_8 and 177_8. This provides for 256 different monitor calls. This parameter, sign extended, is also loaded into the T register on program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

IX.1.6 *INPUT/OUTPUT CONTROL INSTRUCTIONS*

IX.1.6.1 *IOX — Input/Output Execute*

IOX Input/Output Execute Code: 164 000

Format: IOX <device register address>



All transfers between the NORD-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.

IX.1.6.2 *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 addressed.

IOXT is privileged.

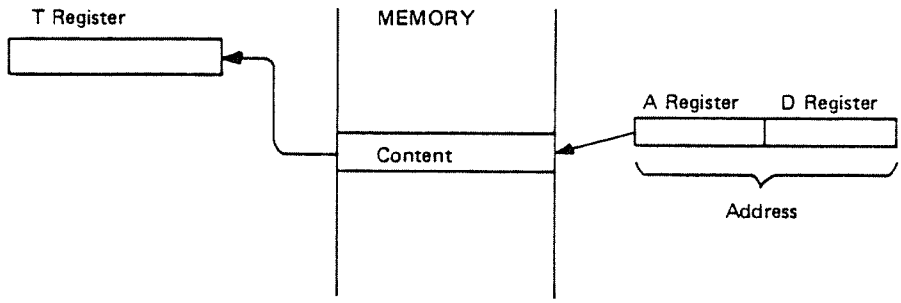
IX.1.7 *EXAMINE AND DEPOSIT*

IX.1.7.1 *Examine*

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.

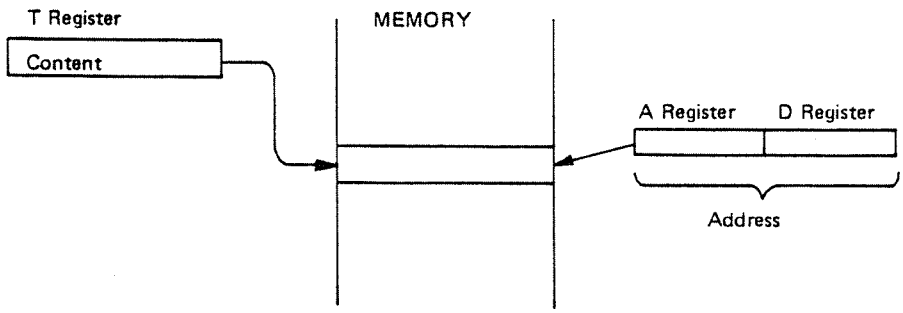


X.1.7.2 *Deposit*

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.

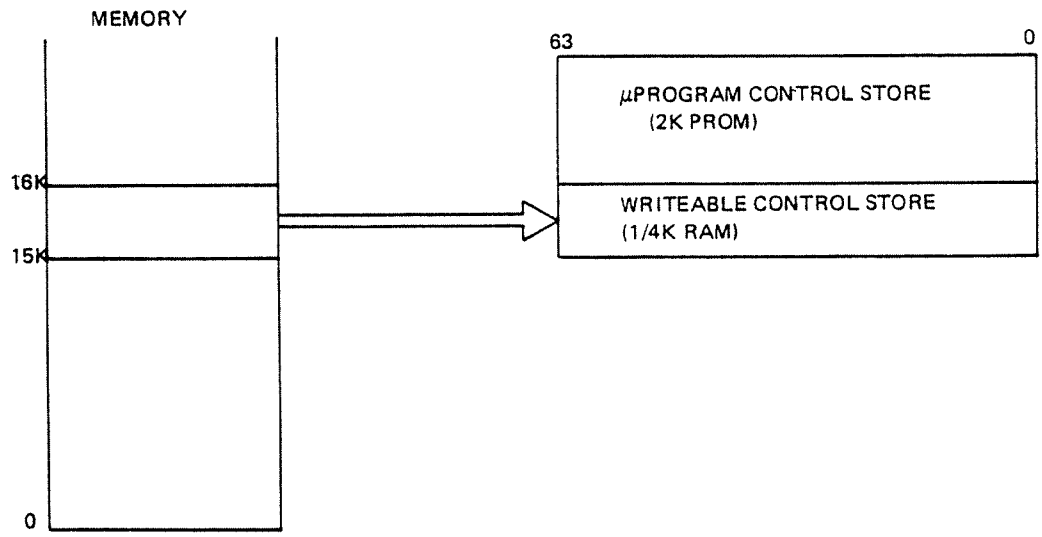


IX.1.8 *LOAD WRITEABLE CONTROL STORE*

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 or memory locations with address from 15 - 16 K in main memory. Microprogram addresses from 4000_8 to 4377 will then be accessible. When the instruction is finished, all microprogram addresses are legal. The illegal instruction interrupt — ROM out of range — 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.

IX.2 INTERNAL REGISTERS

The following internal registers are implemented for internal control and status of the CPU, memory management system and memory. These registers are only accessed by privileged instructions, and could not be accessed by an ordinary customer's program. Internal registers can be accessed when the computer is in STOP or OPCOM mode. Refer also to Section VII.

The following list will show all the internal registers in NORD-100. They are completely described under their respective chapters. An overview with bit assignment is given in Appendix G.

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 micro-program. 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

IX.3 WRITEABLE CONTROL STORE

IX.3.1 *GENERAL*

Refer to Figure IX.3.1.

The illustration shows an instruction readout. The 12 bits ROM address goes into the microprogram control store which is 2K words by 64 bits ROM. The microprogram control store may be extended with a 256 words by 64 bits RAM, so-called Writeable Control Store (WCS). This writeable control store enables the user to write his own microprogram, extending the NORD-100 instruction set for special applications. This microprogram can define special functions or instructions which enhance the performance when the computer is used for a special application.

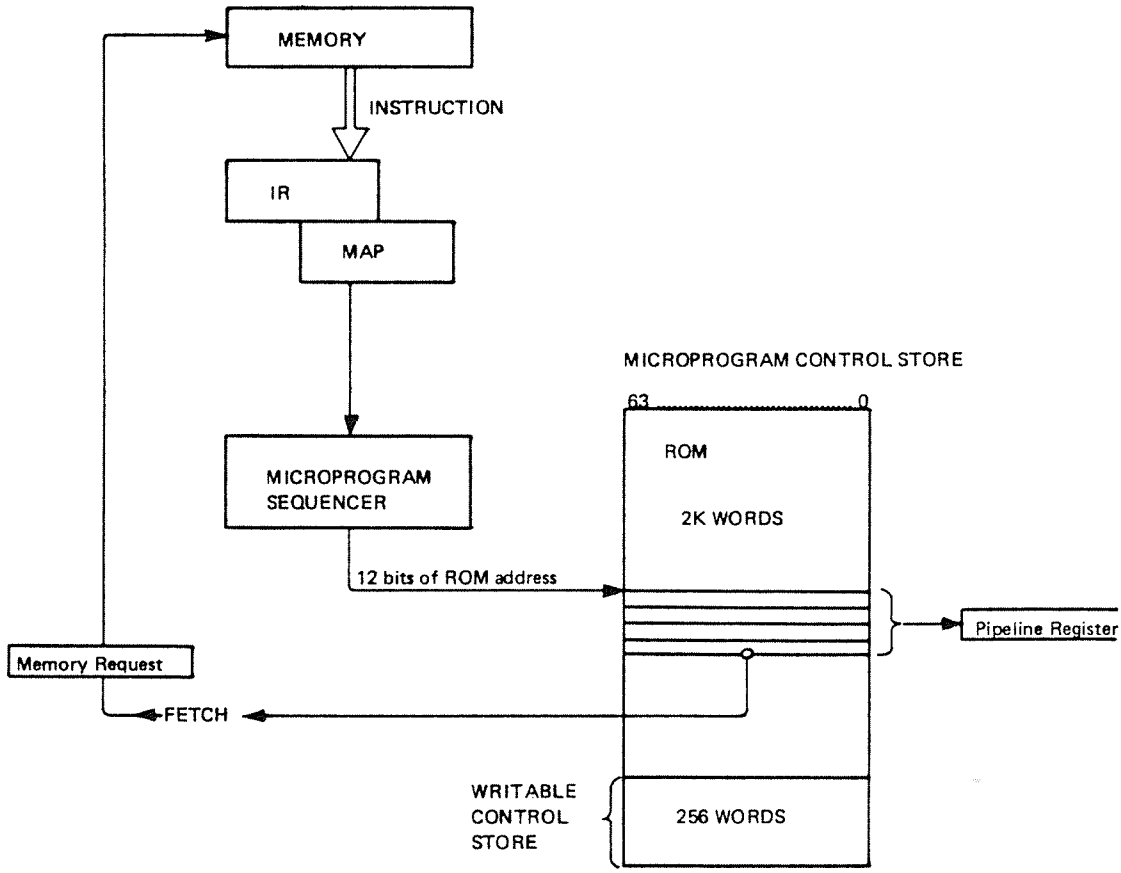


Figure IX.3.1: Instruction Readout

IX.3.2 *WRITING MICRO CODE FOR WRITEABLE CONTROL STORE*

In order to write micro code for the writeable control store, a specification of the new machine instruction to be implemented must be done. The next step is design and coding of the necessary micro instruction. This requires a good knowledge of the CPU architecture and the structure of the microprogram.

The symbolic microprogram will most easily be prepared using an editor with SINTRAN III. It should be saved on an ordinary file. Then the assemblage of the symbolic microprogram into binary format is done with the NORD-100 assembler. The binary representation of the WCS contents must be placed in main memory in physical addresses 36000 to 37777.

IX.3.3 *LOAD THE WRITEABLE CONTROL STORE*

To load the writeable control store, the privileged instructions LWCS (load writeable control store) must be executed. The 256 word by 64 bit RAM (random access memory) writeable control store will be loaded with the contents from memory locations with address from 15-16K in main memory as shown in Figure IX.3.2.

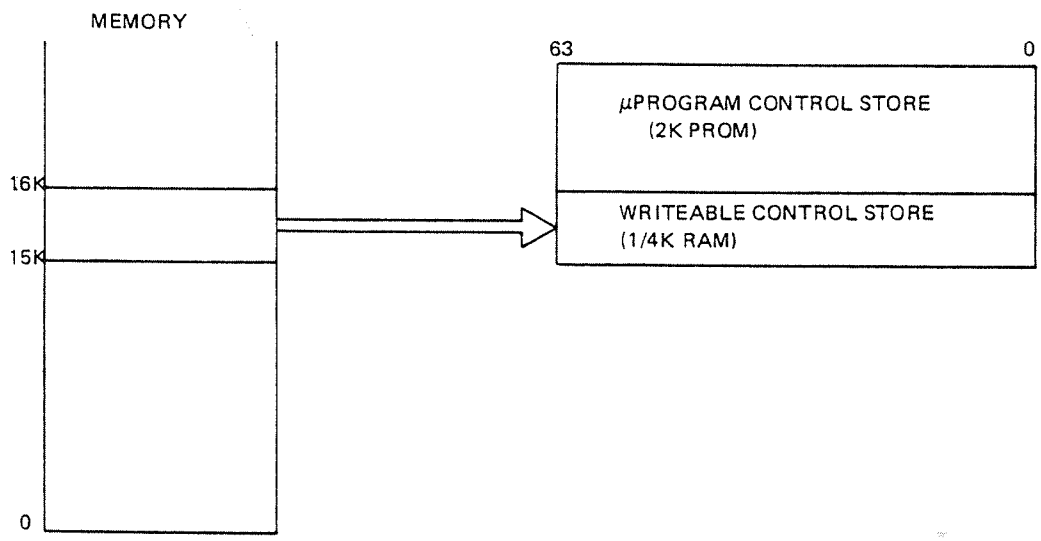


Figure IX.3.2: Loading of the Writeable Control Store

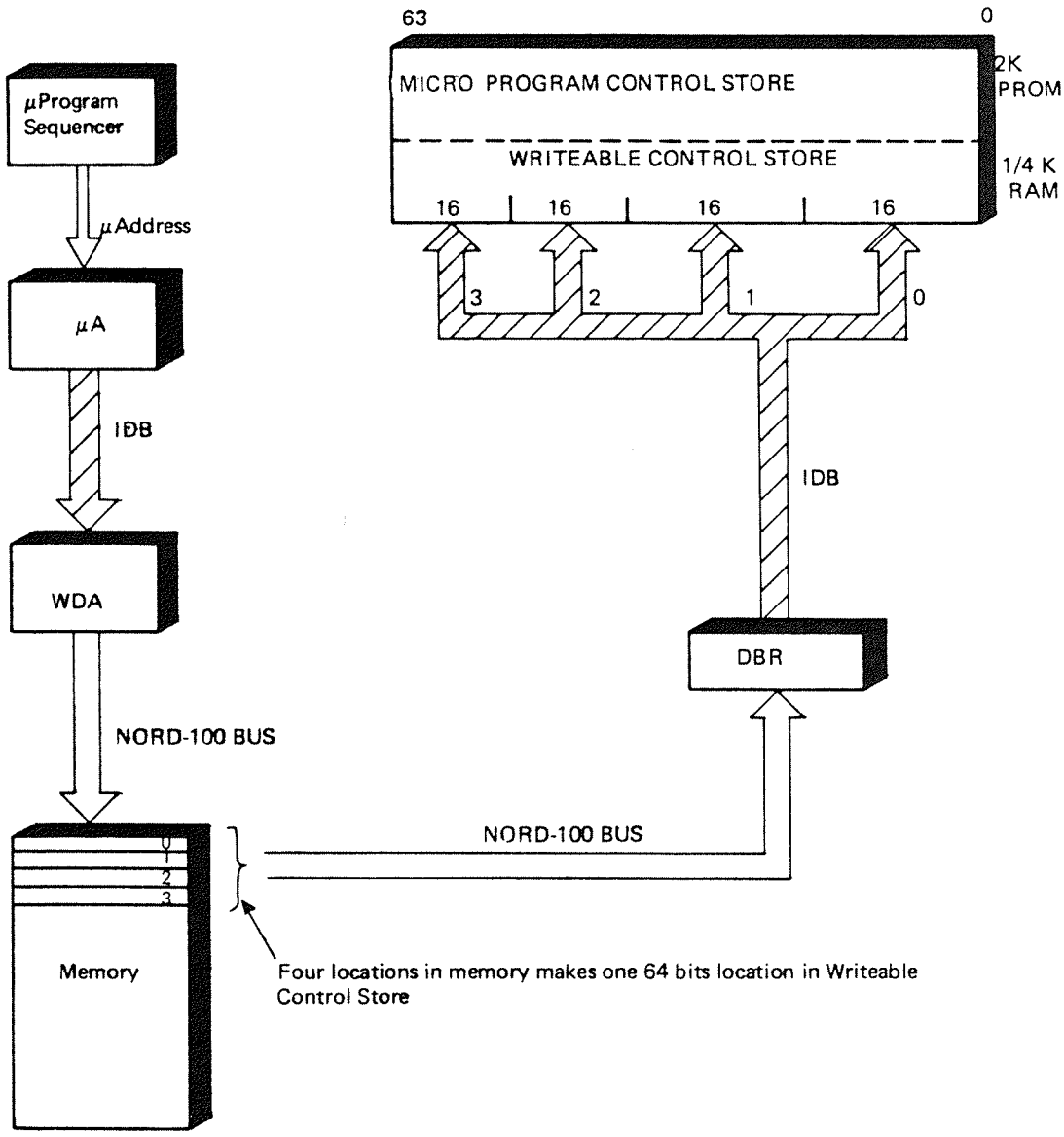


Figure IX.3.3: The Load Procedure of WCS

The LWCS instruction forces the microprogram to generate 1K of read requests to memory. The addresses are generated by the microprogram sequencer, enables them onto the internal data bus (IDB) by means of the micro address register (μ A). Refer to Figure IX.3.2.

The write data and address register (WDA) will enable the addresses onto the NORD-100 bus. The first addressed location in main memory, will go into the writeable control store through the data bus read register (DBR). In WCS these 16 bit words will be placed in bit 0-15 of the WCS word.

The next location in main memory will be placed as bit 16-31 of the WCS word. The third word will be placed as bit 32-47, and the fourth word as bit 48-63 of the WCS word. In this way one 64 bit location in WCS is made by four following locations in main memory.

Figure IX.3.3 shows the address in main memory where the words to build up a WCS word are located, and the addresses used in WCS. All numbers given are octal.

When the LWCS instruction is finished (about 635 μs), microprogram addresses from 4000_8 to 4377_8 will then be accessible, all microprogram addresses are legal.

Thereafter, customer specified instructions may be executed. If they are executed before LWCS is performed, a ROM out-of-range internal interrupt will be generated.

After power up or after pushing the master clear button of the computer, WCS will be unaccessible and LWCS must be executed in order to allow customer specified instructions to be executed.

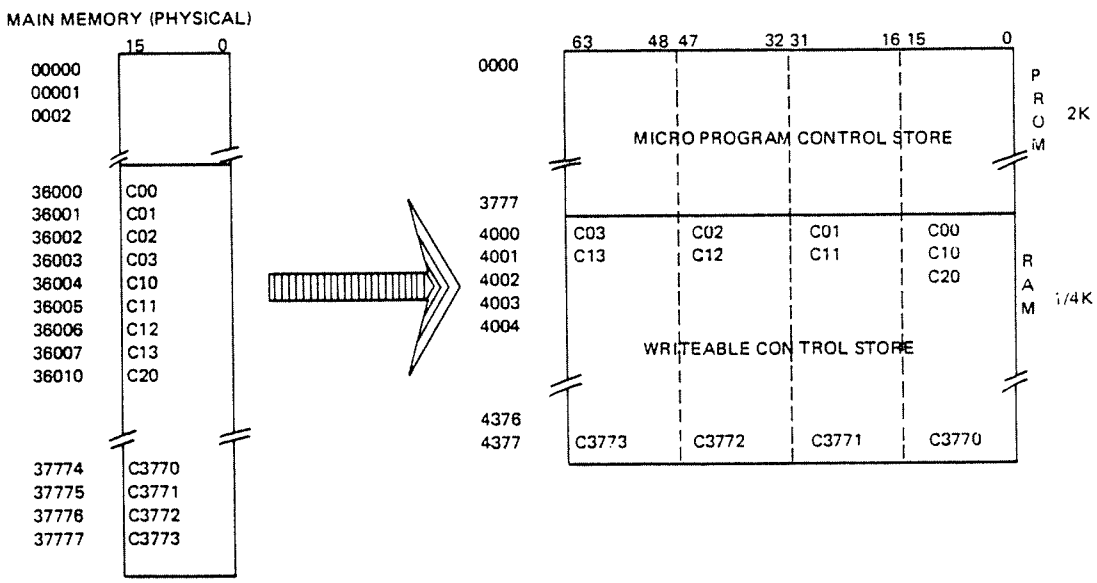


Figure IX.3.4: The Addresses used in Main Memory and Writable Control Store under a LWCS Instruction

IX.3.4 *CUSTOMER SPECIFIED INSTRUCTIONS*

The remaining free codes may be used to extend the NORD-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	1427XX
1431XX			

These 13 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 pont 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 μprogram	4011_8
1423XX	CUST 12	Entry point in μ program	4012_8
1425XX	CUST 13	Entry point in μ program	4013_8
1427XX	CUST 14	Entry point in μ program	4014_8
1431XX	CUST 15	Entry point in μ program	4015_8

All micro instruction codes are available for new customer specified instructions.

More information about using the WCS is found in the "NORD-100 Microprogramming Description" manual.

IX.4 PANEL PROCESSOR PROGRAMMING SPECIFICATION

A program can communicate with the Panel Processor (PAP) by means of the following internal registers:

PANS = Panel Status
PANC = Panel Control

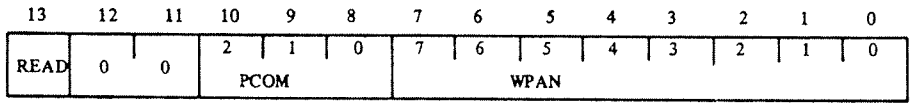
The following privileged instructions apply:

TRA 0 = Transfer PANS to A-register
TRR 0 = Transfer A-register to PANC

IX.4.1 *PANEL CONTROL REGISTER*

The Panel Control register (PANC) is used to send commands to the Panel Processor (PAP) together with possible output data.

The format is as follows:



Bit 13 = READ. The command requests data from PAP which in turn will appear in PANS register.
Bit 11-12 = 0. Must be zero for macroprogram.
Bit 8-10 = PCOM. Panel Command code.
Bit 0-7 = WPAN. Write-data to PAP. Meaning depends on READ and PCOM.

When the command is processed by PAP "Command Ready" appears in Bit 12 of PANS together with possible returned data.

NOTE!

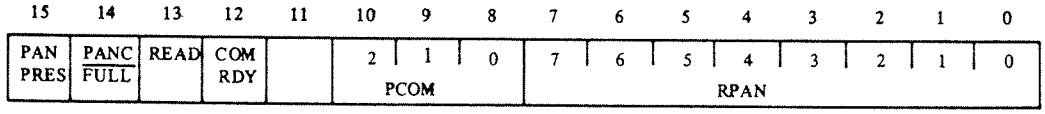
PANC-register is a FIFO that is shared by microprogram. Malfunction may appear if the buffer is over-filled. Therefore, "Command Ready" should be checked when a sequence of commands is sent.

PAP has no interrupt link to macroprogram!

IX.4.2 *PANEL STATUS REGISTER*

This register is used for handshake with macroprogram together with possible input data.

The format is as follows:



Bit 15 = PAN PRES. Panel option is installed.

Bit 14 = PANC-FULL. Panel control register. FIFO is not full. If Bit 14 = 0 for more than 2 ms the PAP is not working.

Bit 13 = READ. The last processed command was requesting data from panel. Read-data is contained in Bits 0-7.

Bit 12 = COM RDY. Command Ready. The command in PCOM has been processed and if READ = 1, the desired data is contained in Bits 0-7.
This bit is cleared by TRA PANS.

Bit 11 = Undefined.

Bit 8-10 = PCOM. The last processed command. See Section 1.3.

Bit 0-7 = RPAN. Read-data from PAP requested by certain panel commands. If no Read-data is requested, RPAN is a copy of WPAN. This may be used for test purposes.

IX.4.3 *PANEL COMMANDS*

Following are the seven possible WRITE and READ commands:

000	Illegal
400	Future extension
1000	Message Append (Write only)
1400	Message Control (Write only)
2000	Update Low Seconds (Write and Read)
2400	Update High Seconds (Write and Read)
3000	Update Low Days (Write and Read)
3400	Update High Days (Write and Read)

IX.4.3.1 *Message on Function Display*

From macroprogram it is possible to send a message to the Function display with the ASCII-codes found in Appendix J. The message is filled in a text buffer of maximum 40 characters. To display more than 4 characters it can be rotated. Hence, the text-string is moving over to the display at readable speed.

MESSAGE CONTROL (PCOM = 3) interprets WPAN in the following manner:

1400	=	STOP rotation
1401	=	ABORT. Return display to owner, OPCOM
1402	=	INIT. Clear text buffer and Function display. Prepare for text to be appended
1404	=	ROT. Rotate text buffer content on Function display
1406	=	INIT and ROT

Remember!

The display is only on loan from OPCOM. All messages should therefore be terminated by ABORT.

If the programmer fails to do so, the operator can type the following codes on the console:

100000 F ↙	=	Abort Message
20000 F ↙	=	Inhibit display of Message until F ↙ is typed

MESSAGE APPEND (PCOM = 2) interprets WPAN as an ASCII character. It is placed at the end of a text buffer with a maximum of 40 characters. If ROT is not specified, the Function display will always show the last 4 characters in the buffer. If ROT is specified, the text buffer is rotated at readable speed. However, text can be appended at the end of the buffer simultaneously. When the text buffer is full, additional characters are ignored until the INIT-command is sent.

Dislayable ASCII-codes are shown in Appendix J.

IX.4.3.2 *Update Calendar*

The PAP has an internal Hardware Calendar circuit that is powered by Standby power. It is incremented by the CPU's RT-Clock oscillator and is therefore tracking the RT-Clock precisely.

The calendar can be adjusted from program, normally the operating system. Once adjusted, it will reveal correct date and time as long as the battery is intact. The operating system may therefore read correct time from PAP when itself has lost track of it (at System Start or Power Fail).

The Calendar consists of a 32 bit counter which is incremented every second.

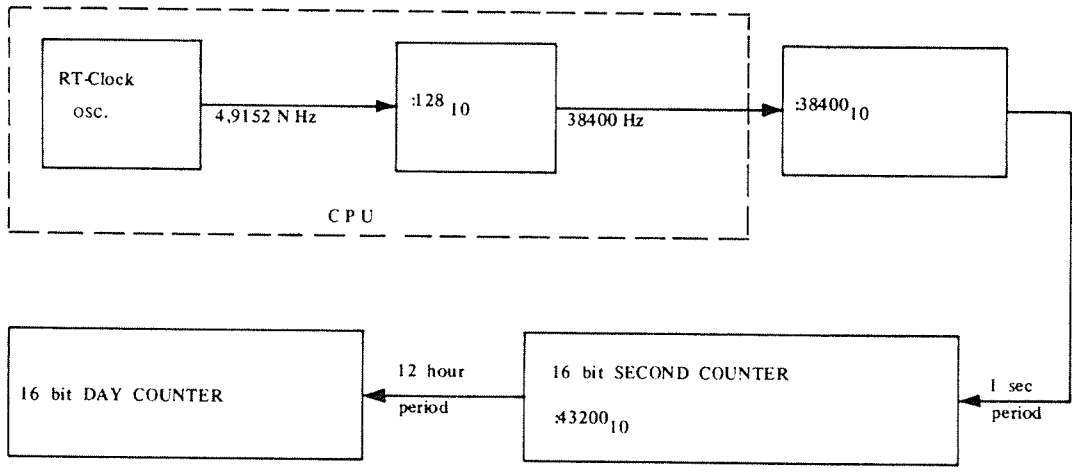


Figure IX.4.1: Calendar Counters

The second counter has shortened count length producing carry every 12th hour. The Second and Day counters are both write and readable with the commands specified in Section IX.4.3. The content of each register is transferred in two (octal coded) bytes. Of the 8 Update commands two have a special meaning.

When adjusting the calendar the new date is loaded into the counters by the *Write/Update High Days* command. When reading the calendar the date is sampled by the *Read/Update Low Seconds* command.

Zero time is set to:

00.00, January 1, 1979

and overflow of the day counter will appear at:

00.00, August 28, 2068

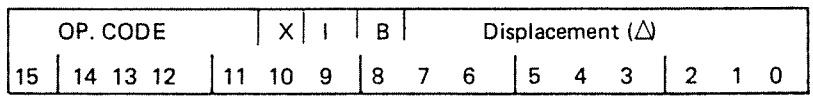
The *displayed* time accounts for leap years.

The time is only displayed when ACT/ is typed on the console. To show year and month instead of clock, etc., type 10000F ↙. Recover with F ↙.

IX.5 NORD-100 INSTRUCTION CODES

Instruction formats and explanations found in the NORD-100 Reference Manual.

MEMORY REFERENCE INSTRUCTIONS



Effective Address:

	000000	Address relative to P;	EL = P +
,X	002000	Address relative to X;	EL = X +
I	001000	Indirect address;	EL = (P +)
,B	000400	Address relative to B;	EL = B +

= displacement

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:

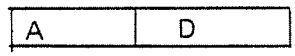
LDA	044000	Load A;	A: = (EL)
LDT	050000	Load T;	T: = (EL)
LDX	054000	Load X;	X: = (EL)

Arithmetical and Logical Instructions:

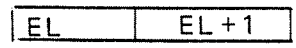
ADD	060000	Add to A (C, O and Q may also be affected);	A: = A + (EL)
SUB	064000	Subtract from A (C, O and Q may also be affected);	A: = A – (EL)
AND	070000	Logical AND to A;	A: = A (EL)
ORA	074000	Logical inclusive OR to A;	A: = A (EL)
MPY	120000	Multiply integer (O and Q may also be affected);	A: = A*(EL)

Double Word Instructions:

DA



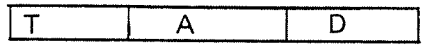
DW



STD	020000	Store double word;	(DW): = AD
LDD	024000	Load double word;	AD: = (DW)

Floating Instructions:

TAD



FW

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

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) = $(sr)_0$

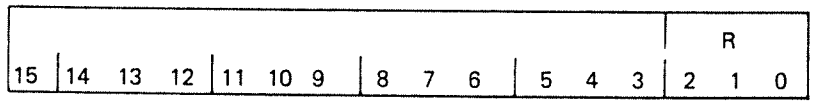
Combined Instructions:

EXIT	146142	COPY SL DP,	Return from subroutine
RCLR	146100	COPY,	Register clear
RINC	146400	RADD AD1,	Register increment
RDCR	146200	RADD CM1,	Register decrement

Extended Arithmetic Operations:

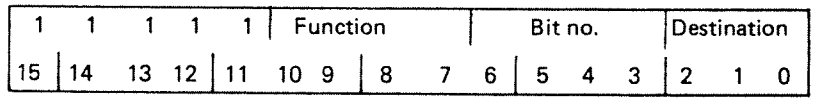
RMPY	141200	Multiply source with destination. Result in double accumulator	AD: = (sr)*(dr)
RDIV	141600	Divide double accumulator with source register. Quotient in A, remainder in D (AD = A*(sr) + D)	A: = AD/(sr)

EXECUTE INSTRUCTION



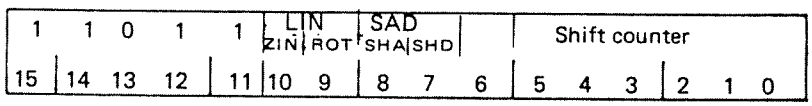
EXR	140600	Execute instruction found in specified register.

BIT INSTRUCTIONS



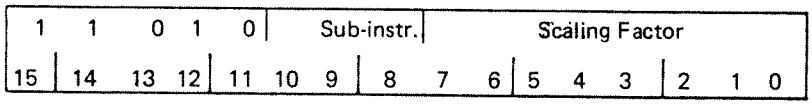
BSKP	175000	Skip next location if specified condition is true;	P: = P + 1
BSET	174000	Set specified bit equal to specified condition;	
BSTA	176200	Store and clear K;	(B): = K; K: = 0
BSTC	176000	Store complement and set K;	(B): = K_0; K: = 1
BLDA	176600	Load K;	K: = (B)
BLDC	176400	Load bit complement to K;	K: = $(B)_0$
BANC	177000	Logical AND with bit compl;	K: = K $(B)_0$
BORC	177400	Logical OR with bit compl.;	K: = KV$(B)_0$
BAND	177200	Logical AND to K;	K: = K (B)
BORA	177600	Logical OR to K;	K: = KV(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
	000000	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



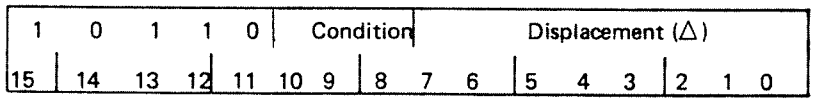
NLZ	151400	Convert the number in A to a floating number in FA.
DNZ	152000	Convert the floating number in FA to a fixed point number in A.
NLZ+20	151420	Integer to floating conversion.
DNZ–20	152360	Floating to integer conversion.

SEQUENCING INSTRUCTIONS

Unconditional Jump:

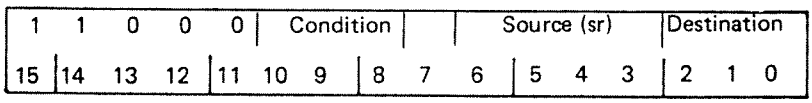
JMP	124000	Jump;	P = EL
JPL	134000	Jump to subroutine;	L = P; P = EL

Conditional Jump:



JAP	130000	Jump if A is positive; P = + 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;	A≠0
JXN	133400	Jump if X is negative;	X<0
JPC	132000	Increment X and jump if positive; X = X + 1; P = P + if	X≥0
JNC	132400	Increment X and jump if negative; X = X + 1; P = P + if	X<0
JXZ	133000	Jump if X is zero;	X = 0

Skip Instructions:

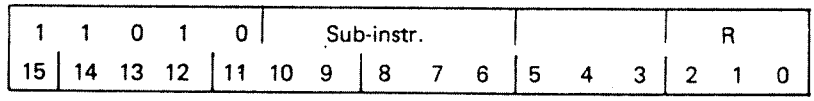


SKP	140000	Skip next location if specified condition is true;	P = P + 1

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

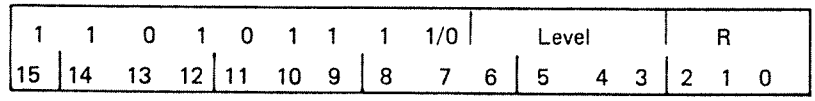
TRANSFER INSTRUCTIONS



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

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

1	1	0	1	0	1	0	1 1	0 1	Level				0 0	0 1	0 0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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

APPENDIX A

NORD-100 MNEMONICS AND THEIR OCTAL VALUES

AAA	: 172400	FMU	: 110000
AAB	: 172000	FSB	: 104000
AAT	: 173000	GEQ	: 000400
AAX	:173400	GRE	: 001000
ADC	: 001000	I	: 001000
ADD	: 060000	IDENT	: 143600
AD1	: 000400	IF	: 000000
ALD	: 000012	IIC	: 000005
AND	: 070000	IIE	: 000005
,B	: 000400	IOF	: 150401
BAC	: 000600	ION	: 150402
BANC	: 177000	IOX	: 164000
BAND	: 177200	IOXT	: 150415
BCM	: 000400	IRR	: 153600
BLDA	: 176600	IRW	: 153400
BLDC	: 176400	JAF	: 131400
BORA	: 177600	JAN	: 130400
BORC	: 177400	JAP	: 130000
BSET	: 174000	JAZ	: 131000
BSKP	: 175000	JMP	: 124000
BSTA	: 176200	JNC	: 132400
BSTC	: 176000	JPC	: 132000
CCLR	: 000010	JPL	: 134000
CILR	: 000012	JXN	: 133400
CLD	: 000100	JXZ	: 133000
CM1	: 000200	LBYT	: 142200
CM2	: 000600	LCIL	: 000011
COPY	: 146100	LDA	: 044000
CSR	: 000010	LDD	: 024000
DA	: 000005	LDF	: 034000
DB	: 000003	LDT	: 050000
DD	: 000001	LDX	: 054000
DEPO	: 150417	LIN	: 003000
DL	: 000004	LMP	: 000002
DNZ	: 152000	LRB	: 152600
DP	: 000002	LSS	: 002400
DT	: 000006	LST	: 003000
DX	: 000007	LWCS	: 143500
ECCR	: 000015	MCL	: 150200
EQL	: 000000	MGRE	: 001400
EXAM	: 150416	MIN	: 040000
EXIT	: 146142	MIX3	: 143200
EXR	: 140600	MLST	: 003400
FAD	: 100000	MON	: 153000
FDV	: 114000	MPY	: 120000

MST	: 150300	SB	: 000030
NLZ	: 151400	SBYT	: 142600
ONE	: 000200	SD	: 000010
OPCOM	: 150400	SEX	: 150406
OPR	: 000002	SHA	: 154400
ORA	: 074000	SHD	: 154200
PCR	: 000003	SHR	: 000000
PEA	: 000015	SHT	: 154000
PES	: 000013	SKP	: 140000
PGS	: 000003	SL	: 000040
PID	: 000006	SP	: 000020
PIE	: 000007	SRB	: 152402
PIOF	: 150405	SSC	: 000060
PION	: 150412	SSK	: 000020
PL10	: 000004	SSM	: 000070
PL11	: 000011	SSO	: 000050
PL12	: 000022	SSQ	: 000040
PL13	: 000043	SSTG	: 000010
POF	: 150404	SSZ	: 000030
PON	: 150410	ST	: 000060
PVL	: 000004	STA	: 004000
RADD	: 146000	STD	: 020000
RAND	: 144400	STF	: 030000
RCLR	: 146100	STS	: 000001
RDCR	: 146200	STT	: 010000
RDIV	: 141600	STX	: 014000
REX	: 150407	STZ	: 000000
REXO	: 145000	SUB	: 064000
RINC	: 146400	SWAP	: 144000
RMPY	: 141200	SX	: 000070
RORA	: 145400	TRA	: 150000
ROT	: 001000	TRR	: 150100
RSUB	: 146600	UCIL	: 000012
SA	: 000050	UEQ	: 002000
SAA	: 170400	WAIT	: 151000
SAB	: 170000	,X	: 002000
SAD	: 154600	ZIN	: 002000
SAT	: 171000	ZRO	: 000000
SAX	: 171400		

APPENDIX B

PROGRAMMING SPECIFICATION FOR SOME I/O DEVICES

B.1 *NORD-100 TELETYPE PROGRAMMING SPECIFICATIONS*

Teletype Addresses:

The codes below are relevant for the first teletype (Teletype no. 0). The codes for the first 8 teletypes are found by adding 10_8.N to the codes given. N = teletype number (0, 1, 2, ..., 7). For the next 8 teletypes the codes are found by adding $(1000 + 10(N - 10))_8$.N = teletype number $(10, 11, 12, ..., 17)_8$.

Input Channel (Interrupt Level 12):

Read Data Register

IOX 300

The number of data bits into the A register are specified by bits 11 and 12 in the input channel control register. The received character is right justified (from bit 0 and upwards).

Read Status Register

IOX 302

Write Control Register

IOX 303

Output Channel (Interrupt Level 10):

Write Data Register

IOX 305

The number of bits specified by bits 11 and 12 in the *input* channel control register is written to the output data register, starting with bit 0 and counting upwards.

Read Status Register

IOX 306

Write Control Register

IOX 307

Data Rate Selection:

IOX 301

There are two possibilities to control the data rate for input and output serial data.

The data rate can be selected by:

a) Switch Setting on card. All four interfaces are operated at the data rate selected by the switches each time Master Clear is pressed. The switch setting is common for input and output.

b) IOX Instruction. If data rate is selected by software for one of the interfaces the programmed data rate is selected for all four interfaces.

Input and output are inpdendent and are selected by the same IOX instruction (group device number + 1). The content of the A register before the IOX instruction is executed determines the baud rate. The 4 least significant bits (0-3) are used for the input channel, and the next 4 bits (4-7) are used for the output channel. The table below gives the bit pattern and corresponding baud rate.

BAUD RATE	OUTPUT 7	6	5	4	INPUT 3	2	1	0
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1
50	0	0	1	0	0	0	1	0
75	0	0	1	1	0	0	1	1
110	1	1	1	1	1	1	1	1
134.5	0	1	0	0	0	1	0	0
150	1	1	1	0	1	1	1	0
200	0	1	0	1	0	1	0	1
300	1	1	0	1	1	1	0	1
600	0	1	1	0	0	1	1	0
1200	1	0	1	1	1	0	1	1
1800	1	0	1	0	1	0	1	0
2400	0	1	1	1	0	1	1	1
2400	1	1	0	0	1	1	0	0
4800	1	0	0	1	1	0	0	1
9600	1	0	0	0	1	0	0	0

IDENT Code:

The IDENT code for the input channel and the output channel will be the same, with the input channel responding to level 12 and the output channel to level 10.

The IDENT codes are:

Teletype 0: 1
Teletype 1: 5
Teletype 2: 6
Teletype 3: 7

Input Channel:

Status Register

Bit 0: Ready for transfer interrupt enabled
Bit 1: Error interrupt enabled
Bit 2: Device active
Bit 3: Device ready for transfer
Bit 4: Inclusive OR of errors
Bit 5: Framing error
Bit 6: Parity Error
Bit 7: Overrun
Bits 8-14: Not used
Bit 15: Motor stopped

Note: The meaning of the error indicator bits are as follows:

Bit 5: Framing error means that the stop bit is missing.

Bit 6: Parity error means that a parity error has occurred while working in parity generation/checking mode.

Bit 7: Overrun means that at least one character is overwritten.

The meaning of bit 15, motor stopped, is that the automatic start/stop function has given a stop signal to the teletype motor.

Control Register:

Bit 0: Enable interrupt on device ready for transfer
Bit 1: Enable interrupt on errors
Bit 2: Active device
Bit 3: Test mode
Bit 4: Device clear
Bits 5-10: Not used
Bits 11-12: Character length
Bit 13: Number of stop bits
Bit 14: Parity generation/checking
Bit 15: Motor autostop

Notes:

Bit 2: After a Master Clear or a Device Clear (bit 4), the device has to be activated by writing a 1 into bit 2 of the control register. This will enable the received data into the data register. The device will then be active until bit 2 is cleared by writing 0 into it.

Bit 3: Test mode will loop transfer data back to received data, and nothing is transmitted to the peripheral device.

Bits 11-12: The content of these bits give the following character lengths, both for the input channel and the output channel.:

Bit:

12	11	
0	0	8 bits
0	1	7 bits
1	0	6 bits
1	1	5 bits

If bit 14 is a 1, a parity bit is *added* to the number given in this table.

Bit 13: The number of stop bits will be two if the control bit is 0, and one if the control bit is 1.

Bit 14: If this control bit is 0, no parity bit will be added to the character on the output channel, and the received character will not be checked for parity. A 1 in this control bit will add an even parity bit to the character on the output channel, and give an error indication if the received character has an odd parity.

Bit 15: If this control bit is 0, the monitor of the teletype is supposed to be running all the time. If this bit is a 1, the motor is given a stop signal, about 37 seconds after the last character was transmitted, and a start signal about 0.6 seconds after a new character is transmitted after a stop signal.

Output Channel:

Status Register

Bit 0: Ready for transfer interrupt enabled
Bit 1: Error interrupt enabled
Bit 2: Device active
Bit 3: Device ready for transfer
Bit 4: Inclusive OR of errors
Bit 5: Framing error
Bits 6-14: Not used
Bit 15: Motor stopped

Notes:

Bit 2: This status bit will be a 1 as long as the device is busy transmitting characters.

Bit 3: This bit indicates that the output data buffer is ready to receive a new character. This will be a 1 if bit 2 is 0, but may as well be a 1 when bit 2 is 1 due to the double buffer.

Bits 4-5: As there is only one error indicator for the output channel, these two bits have the same meaning. A 1 indicates that the current loop is broken, caused by either a broken line or no peripheral device connected.

Bit 15: The same bit as in Input Channel Status.

Control Register

Bit 0: Enable interrupt on device ready for transfer
Bit 1: Enable interrupt on errors
Bit 2: Activate device
Bit 3-15: Not used

Notes:

Bit 2: Activate device should be used in the same manner as for other output devices, although the output channel is activated by writing data to the output data register.

B.2 *SPECIFICATION OF TAPE READER INTERFACE*

Standard device number 0400 $(0400\text{-}0403)_8$
Number of device number 4
Standard interrupt level 12_{10}
Standard ident number 2

Control Word IOX Device Number + 3:

Bit 0:	Enable interrupt on ready for transfer
Bit 1:	Not used
Bit 3:	Test
Bit 4:	Device clear
Bits 5-15:	Not used

Status Word IOX Device Number + 2:

Bit 0:	Interrupt enables on ready for transfer
Bit 2:	Read active
Bit 3:	Reader ready for transfer (character read)
IOX device no. + 1:	Not used
IOX device no.	Read character. The same character may be read several times if desired.

Test:

When bit 3 in the control word is set to 1, the interface may be tested without reader.

If bit 2 is also set to 1, the interface will give "ready for transfer" after a while.

If "ready for transfer" is constantly 1, the data register will increment for each time IOX device number + 3 (control word write) is used, and then it is possible to test the data patch.

B.3 *SPECIFICATION OF THE PAPER TAPE PUNCH INTERFACE*

Standard device number 0410 $(0410\text{-}0413)_8$
Number of device number 4
Standard interrupt level 10_{10}
Standard ident number 2

Write Control Word IOX Device Number + 3:

Bit 0:	Enable interrupt
Bit 1:	Not used
Bit 2:	Activate device (punch character now in buffer)
Bit 3:	Test
Bit 4:	Device clear
Bits 5-15:	Not used

Read Status Word IOX Device Number + 2:

Bit 0:	Interrupt enabled
Bit 2:	Device active 0
Bit 3:	Device ready

Write data word IOX device number + 1.
Write the 8 bits to be punched in a buffer register.

Read data IOX device number.
Only used under test.

It is not wise to write a character into the buffer if the punch is not ready.

Test:

The interface may be tested without punch.

When bit 3 i nthe control word is one, the buffer register may be read back by IOX device number. If the interface is activated, it will become "ready for transfer" after a while.

B.4 *NORD-100 FLOPPY DISK PROGRAMMING SPECIFICATION*

B.4.1 *Device Register Address*

Codes given below are only relevant for disk system I.

Read Data Buffer

IOX 1560 (IOX RDAT)

Write Data Buffer

IOX 1561 (IOX WDAT)

Read Status Register No. 1

IOX 1562 (IOX RSR1)

Write Control Word

IOX 1563 (IOX WCWD)

Read Status Regiser No. 2

IOX 1564 (IOX RSR2)

Write Drive Address/Write Difference

IOX 1565 (IOX WDAD)

Read Test

IOX 1566 (IOX WDAD)

Write Sector/Write Test Byte

IOX 1567 (IOX WSCT)

For disk system II add 10_8 to the codes specified above. Each disk system can handle up to 3 drives. Ident code for disk system I is 21_8. Ident code for disk system II is 22_8. Interrupt level is 11_{10}.

B.4.2 *Instruction Formats and Descriptions*

B.4.2.1 Read Data Buffer (IOX RDAT)

Reads one 16 bit word from the interface buffer.

Buffer address is automatically incremented after execution of the instruction.

B.4.2.2 Write Data Buffer (IOX WDAT)

Writes one 16 bit word to the interface buffer.

Buffer address is automatically incremented after execution of the instruction.

B.4.2.3 Read Status Register Number 1 (IOX RSR1)

Bit 0: Not used.
Bit 1: Interrupt enabled
Bit 2: Device busy
Bit 3: Device ready for transfer
Bit 4: Inclusive or of bits set in status regsiter number 2.

Note: When bit 4 is set, an error has occurred and status register 2 *must* be read before proceeding.

Bit 5: Deleted Record.

This bit is set after read data command if the sector contained a deleted data address mark.

Bit 6: Read/Write Complete

A read or write operation is completed.

Bit 7: Seek Complete

The status bit is set after seek or recalibration command when the disk has finished moving the R/W head.

Bit 8: Time Out.

Approximately 1.5 seconds.

Bits 9-11: Only used when formatting.

Bit 9 is active when buffer address bits 1 and 6 are active.
Bit 10 is active when buffer address bits 1 and 7 are active.
Bit 11 is active when buffer address bits 1 and 8 are active.

Note: Bits 4-7 are only significant after interrupt or when device busy is reset.

Bits 12-15: Not used.

B.4.2.4 Write Control Word (IOX WCWD)

Bit 0: Not used
Bit 1: Enable interrupt
Bit 2: Not used
Bit 3: Test Mode (for description see IOX RTST and IOX WSCT)
Bit 4: Device Clear (NB: selected drive is deselected)
Bit 5: Clear interface buffer address
Bit 6: Enable time-out
Bit 7: Not used

The following bits are commands to the floppy disk and these are the only control bits that generate device busy and give interrupt. (NB: with the exception of bit 15, control reset.)

Bit 8: Format track
Bit 9: Write data
Bit 10: Write deleted data
Bit 11: Read ID
Bit 12: Read data
Bit 13: Seek
Bit 14: Recalibrate
Bit 15: Control reset

B.4.2.5 Read Status Register Number 2 (IOX RSR2)

Bits 0-7: Not used
Bit 8: Drive not ready

This bit is set if the addressed drive is not powered up, its door is open, the diskette is not properly installed or drive address is invalid.

Bit 9: Write protect

This bit is set if a write operation is attempted on a write protected diskette.

Bit 10: Not used

Bit 11: Sector missing + No AM

This bit is set if the desired sector for a Read Data Write Data or Write Deleted Data cannot be located on the diskette.

In addition, this bit may indicate a non-locatable data field address mark, or a non-locatable ID field address mark.

Bit 12: CRC error

Bit 13: Not used

Bit 14: Data overrun

A data byte was lost in the communication between the NORD-10/S interface and the floppy disk system.

Bit 15: Not used.

B.4.2.6 Write Drive Address/Write Difference (IOX WDAD)

This is 2 instructions, depending on bit 0 in the A register.

A) Bit 0 = 1: Write drive address

This instruction selects drive and format.

Bits 1-70 Not used

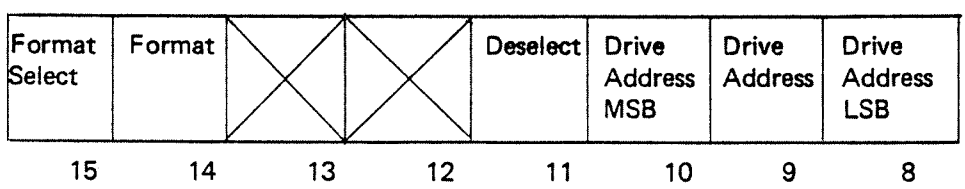
Bits 8-10: Drive address (unit number) 0, 1 or 2

Bit 11: Deselect drives

Bits 12-13: Not used

Bits 14-15: Format select

Bit 15:	Bit 14:	Format (all numbers decimal)
0	x	IBM 3740 128 bytes/sector 26 sectors/track
1	0	IBM 3600 256 bytes/sector 15 sectostrack
1	1	IBM System 32-II 512 bytes/sector 8 sectors/track



B) Bit 0 = 0: Write difference

This is the difference between current track and desired track. It is used as an argument for the seek command.

Bits 1-7: Not used
Bits 8-14: Difference between current and desired track
Bit 15: Direction select
Bit 15 = 0: Access "out" to a lower track address
Bit 15 = 1: Access "in" to a higher track address

In/Out	Diff. MSB	Diff.	Diff.	Diff.	Diff.	Diff.	Diff. LSB
15	14	13	12	11	10	9	8

B.4.2.7 Read Test Data (IOX RTST)

This instruction is used for simulation of a data transfer between the floppy disk system and NORD-10/S interface.

It does *not* transfer data from the NORD-10/S interface to the A register, but puts one 8 bit byte into the interface buffer, each time the instruction is executed. The bytes are packed to 16 bit words in the buffer and may later be read by IOX RDAT instructions.

The byte may be chosen by using the IOX WSCT instruction (see description of IOX WSCT).

IOX RTST is used for test purposes only and does not generate interrupt and busy signals.

The instruction is only active when the interface is set in test mode by the following instructions:

```
SAA 10
IOX WCWD
```

To reset test mode, the following instructions must be used:

```
SAA 0
IOX WCWD          % reset test mode bit
SAA 20
IOX WCWD          % device clear
```

B.4.2.8 Write Sector/Write Test Byte (IOX WSCT)

When the interface is set in test mode, this instruction loads the test byte which is transferred by the IOX RTST command. If not in test mode, this instruction loads the sector number to be used in a subsequent, read/write command.

A) Not in test mode:
Bits 0-7: Not used
Bits 8-14: Sector to be used in a subsequent read write command.
Sector range (octal) for different formats:

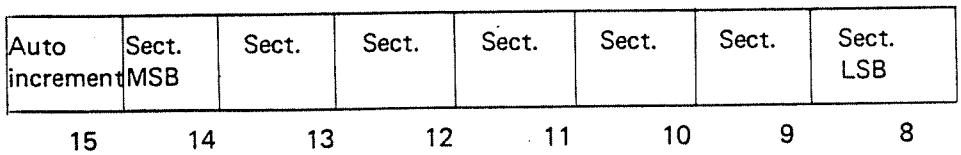
1-32 for IBM 3740
1-17 for IBM 3600
1-10 for IBM System 32-II

NB: 0 must not be used.

Bit 15: Sector auto increment.

If this bit is true, the sector register is automatically incremened after each read/write command.

Note: This auto increment is not valid past the last sector of a track.



Read Status Word IOX Device + 2:

Bit 0: Interrupt enabled on ready for transfer
Bit 1: Interrupt enabled on error
Bit 2: Card reader active. Signal from card reader.
Bit 3: Ready for transfer): a column may be read. This bit is turned off by "read data" IOX DEV.
Bit 4: bits 5-9 set, error
Bit 5: Hopper check error set from card reader
Bit 6: Light or dark error from card reader
Bit 7: Motion check error from card reader
Bit 8: Overrun. One column was lost because it was not read before the next column was strobed into buffer. Cleared by Master Clear.
Bit 9: End of card
Bits 10-15: Not used

Be aware that the card reader sends hopper check while reading the last card.

Write Data IOX Device + 1:

Only used for testing the interface without card reader.

Read Data IOX Device:

Read last column.

Test:

The interface may be tested without card reader. When bit 3 in the control word is set to one, the interface is in "test mode".

An IOX device + 1 will simulate a column read by the interface, set ready for transfer and increment the data register.

An "end of card" is simulated by IOX device + 1 and bit 5 is set to one.

Programming Example:

(Without interrupt)

```
CRDEV = 420
RD = 0
RS = 2
RC = 3

)FILL
%
%                   Error bit set
%
CRERT,              BSKP 110 DA ZRO; JMP CREOC
%                   Error status in A regsiter
                    Exit
%
%
CRER2,
%                   Error to many or few columns read
                    Exit
CREOC,              JXZ * + 2; JMP CRER2
                    LDX (CBUF; EXIT AD1
)FILL
CBUF,               0
CPUF + 120/0
```

B.5 *NORD-100 DISK PROGRAMMING SPECIFICATIONS*

Disk Device Register Address:

The codes below are relevant for disk system I. Each disk system may consist of 4 disk units. For disk system II add 10_8 to the specified codes.

Read Memory Address

IOX 500

Load Memory Address

IOX 501

Read Sector Counter

IOX 502

Load Block Address

IOX 503

Read Status Register

IOX 504

Load Control Word

IOX 505

Seek Instruction

IOX 506

Load Word Counter Register

IOX 507

The minimum number of words to be transferred is one sector, i.e., 200_8 words, the maximum number of words is one track, i.e., 25 sectors.

Read Block Address

This instruction is implemented for maintenance purposes only. By first loading a control word with bit 3 (test mode), the instruction

IOX 506

will return the previously loaded block address to the A register.

Control Word

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

Unit Select Code:

Bit 10	9	
Bit 0	0	Unit 0
Bit 0	1	Unit 1
Bit 1	0	Unit 2
Bit 1	1	Unit 3

Device Operation Code:

Bit 12	11	
Bit 0	0	Read transfer
Bit 0	1	Write transfer
Bit 1	0	Read parity
Bit 1	0	Compare

To format a disk, the key for formatting has to be turned on, Write Transfer, and Write Format must be specified.

Status Word

Bit 0:	Ready for transfer, interrupt enabled
Bit 1:	Error interrupt enabled
Bit 2:	Device active
Bit 3:	Device ready for transfer
Bit 4:	Inclusive OR of errors (status bits 5-11)
Bit 5:	Write protect violate
Bit 6:	Time out
Bit 7:	Hardware error
Bit 8:	Address mismatch
Bit 9:	Parity error
Bit 10:	Compare error
Bit 11:	Missing clock error
Bit 12:	Transfer complete
Bit 13:	Transfer on
Bit 14:	On cylinder
Bit 15:	Bit 15 loaded by previous control word

Interrupt

The disk interrupt level is 11 and the ident number for the first disk system is 1.

APPENDIX C

STANDARD NORD-100 DEVICE NUMBERS* AND IDENT CODES

* Corresponding logical device numbers used in SINTRAN III are found in the SINTRAN III User's Guide (ND-60.050).

Device No.	*Interrupt Level*	*Ident Code (octal)*	*Device*	*SINTRAN III*
4- 7	13	4	Memory Parity N-12	
10- 13	13	1	Real Time Clock 1	
14- 17	13	2	Real Time Clock 2	
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	4	Synchr. Modem 1	
110-117	10-12	14	Synchr. Modem 2	
120-127	10-12	20	Synchr. Modem 3	
130-137	10-12	24	Synchr. Modem 4	
140-147	10-12	30	Synchr. Modem 5	
150-157	10-12	34	Synchr. Modem 6	
200-207	10-12	60	Asynchr. Modem 1	Terminal 17
210-217	10-12	61	Asynchr. Modem 2	Terminal 18
220-227	10-12	62	Asynchr. Modem 3	Terminal 19
230-237	10-12	63	Asynchr. Modem 4	Terminal 20
240-247	10-12	64	Asynchr. Modem 5	Terminal 21
250-257	10-12	65	Asynchr. Modem 6	Terminal 22
260-267	10-12	66	Asynchr. Modem 7	Terminal 23
270-277	10-12	67	Asynchr. Modem 8	Terminal 24
300-307	10-12	1	Teletype 1	Terminal 1
310-317	10-12	5	Teletype 2	Terminal 2
320-327	10-12	6	Teletype 3	Terminal 3
330-337	10-12	7	Teletype 4	Terminal 4
340-347	10-12	44	Teletype 5	Terminal 5
350-357	10-12	45	Teletype 6	Terminal 6
360-367	10-12	46	Teletype 7	Terminal 7
370-377	10-12	47	Teletype 8	Terminal 8
400-403	12	2	Paper Tape Reader 1	
404-407	12	22	Paper Tape Reader 2	
410-413	10	2	Paper Tape Punch 1	
414-417	10	22	Paper Tape Punch 2	
420-423	12	3	Card Reader 1	
424-427	12	23	Card Reader 2	
430-433	10	3	Line Printer 1	

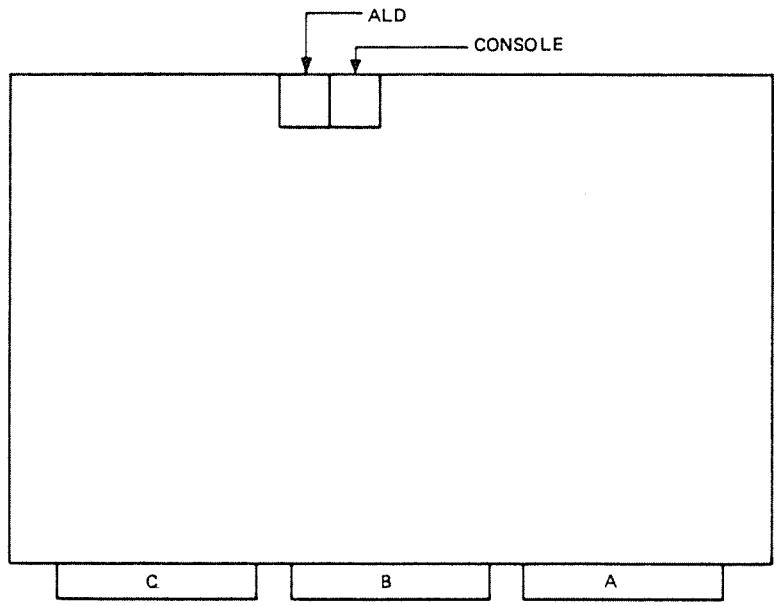
Device No.	Interrupt Level	Ident Code (octal)	Device	SINTRAN III
434-437	10	23	Line Printer 2	
440-443	10	11	Calcomp Plotter 1	
444-447	10	12	Card Punch 1	
454-457	10	13	Card Punch 2	
500- 507	11	1	Disk System 1	
510- 517	11	5	Disk System 2	
520- 527	11	3	Magtape Controller 1	
530- 537	11	7	Magtape Controller 2	
540- 547	11	2	Drum 1	
550- 557	11	6	Drum 2	
560- 577	12-13	156	HASP DMA	
600- 607	11	4	Versatec 1	
610- 617	11	11	Core-to-Core 1	
620- 637	11	10	CDC I/O Link	
640- 647	10-12	124		Terminal 33
650- 657	10-12	125		Terminal 34
660- 667	10-12	126		Terminal 35
670- 677	10-12	127		Terminal 36
700- 707	12	11	CATSY 1	
710- 717	12	21	CATSY 2	
720- 727			A/D Converter	
730- 737	10	10	D/A Converter	
750- 753	13	5	BIG MPM LOG module	
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	
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	
1060-1077			NORD-50/2 Reg.	
1100-1107	10-12	130		Terminal 37
1110-1117	10-12	131		Terminal 38
1120-1127	10-12	132		Terminal 39
1130-1137	10-12	133		Terminal 40
1140-1147	10-12	134		Terminal 41
1150-1157	10-12	135		Terminal 42
1160-1167	10-12	136		Terminal 43

Device No.	*Interrupt Level*	*Ident Code (octal)*	*Device*	*SINTRAN III*
1170-1177	10-12	137		Terminal 44
1200-1207	10-12	70	Asynchr. Modem 9	Terminal 25
1210-1217	10-12	71	Asynchr. Modem 10	Terminal 26
1220-1227	10-12	72	Asynchr. Modem 11	Terminal 27
1230-1237	10-12	73	Asynchr. Modem 12	Terminal 28
1240-1247	10-12	74	Async 13/Photo 1	Terminal 29
1250-1257	10-12	75	Async 14/Photo 2	Terminal 30
1260-1267	10-12	76	Async 15/Photo 3	Terminal 31
1270-1277	10-12	77	Async 16/Photo 4	Terminal 32
1300-1307	10-12	50	Teletype 9	Terminal 9
1310-1317	10-12	51	Teletype 10	Terminal 10
1320-1327	10-12	52	Teletype 11	Terminal 11
1330-1337	10-12	53	Teletype 12	Terminal 12
1340-1347	10-12	54	Teletype 13	Terminal 13
1350-1357	10-12	55	Teletype 14	Terminal 14
1360-1367	10-12	56	Teletype 15	Terminal 15
1370-1377	10-12	57	Teletype 16	Terminal 16
1400-1407	10-12	140		Terminal 45
1410-1417	10-12	141		Terminal 46
1420-1427	10-12	142		Terminal 47
1430-1437	10-12	143		Terminal 48
1500-1507	10-12	144		Terminal 49
1510-1517	10-12	145		Terminal 50
1520-1527	10-12	146		Terminal 51
1530-1537	10-12	147		Terminal 52
1540-1547	11	17	Big Disk System 1	
1550-1557	11	20	Big Disk System 2	
1560-1567	11	21	Floppy Disk 1	
1570-1577	11	22	Floppy Disk 2	
1600-1603			HDLC Auto Load	
1640-1657	12-13	150		HDLC 1
1660-1677	12-13	151		HDLC 3
1700-1780	12-13	152		HDLC 3
1700-1707	10-12	154		Terminal 57
1720-1727	10-12	156		Terminal 59
1730-1737	10-12	157		Terminal 60
1740-1747	10-12	160		Terminal 61
1750-1757	10-12	161		Terminal 62
1760-1767	10-12	162		Terminal 63
1770-1777	10-12	163		Terminal 64

APPENDIX D

SWITCH SETTINGS FOR THE DIFFERENT NORD-100 MODULES

D.1 *SWITCHES ON THE CPU MODULE (3002)*



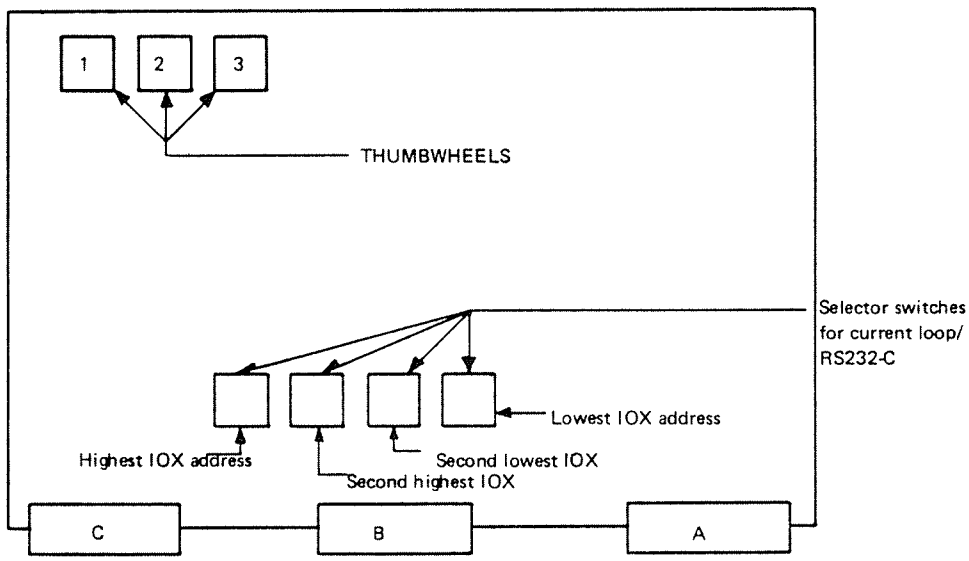
D.1.1 *ALD — Automatic Load Descriptor*

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 load from 500	Mass load from 500
12	21540	Start in address 20	Mass load from 1540	Mass 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 load from 500	Mass load from 500
4	121540	Mass load from 1540	Mass load from 1540	Mass 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

D.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	—

D.2 *SWITCHES ON FLOPPY AND 4 TERMINALS MODULE (3010)*



1 = Floppy disk system
2 = terminal group
3 = initial band rate for terminals

D.2.1 *1 Floppy Disk System*

0 = floppy system no. 1 (10X 1560 - 1567, IDENT = 21)
1 = floppy system no. 2 (10X 1570 - 1577, IDENT = 22)

2-15 are unused, will answer on 1OX 0-7.

D.2.2 *2 Terminal Group:*

Each group consists of 4 terminals with consecutive I0X addresses and ident codes.

0 = terminals 1-4	(IOX 300 - 340,	IDENT 120-123)
1 = terminals 5-8	(IOX 340 - 370,	IDENT 44-47)
2 = terminals 9-12	(IOX 1300 - 1330,	IDENT 50-53)
3 = terminals 13-16	(IOX 1340 - 1370,	IDENT 54-57)
4 = terminals 33-36	(IOX 640 - 670,	IDENT 124-127)
5 = terminals 37-40	(IOX 1100 - 1130,	IDENT 130-133)
6 = terminals 41-44	(IOX 1140 - 1170,	IDENT 134-137)
7 = terminals 45-48	(IOX 1400 - 1430,	IDENT 140-143)
8 = terminals 49-52	(IOX 1500 - 1530,	IDENT 144-147)
9 = terminals 53-46	(IOX 1640 - 1670,	IDENT 150-153)
10 = terminals 57-60	(IOX 1700 - 1730,	IDENT 154-157)
11 = terminals 61-63	(IOX 1704 - 1770,	IDENT 160-163)
12 = async. modem 1-4	(IOX 200 - 230,	IDENT 60-63)
13 = async. modem 5-8	(IOX 240 - 270,	IDENT 64-67)
14 = async. modem 9-12	(IOX 1200 - 1230,	IDENT 70-73)
15 = async. modem 13-16	(IOX 1240 - 1270,	IDENT 74-77)

D.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

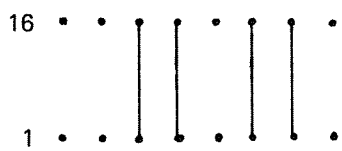
Selector switches for current loop/RS232-C.
Switch set to 0 selects current loop.
Switch set to 1 selects RS232-C.
Switch settings under Terminal Group and Initial baud rate for terminals are also valid for the 8 terminal modules (3013).

This description is correct for the 2 position switch.

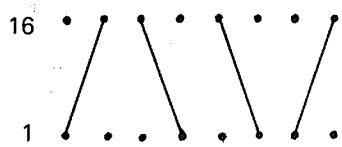
If a hexadecimal switch is used:

0 = current loop
F = RS232-C

If component houses are used:

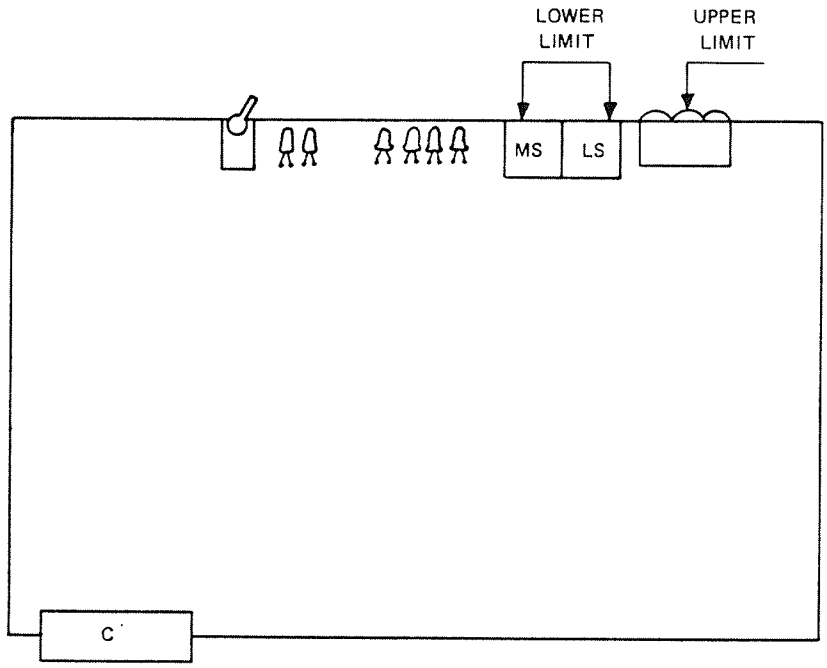


Selects current loop



Selects RS-232-C

D.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.

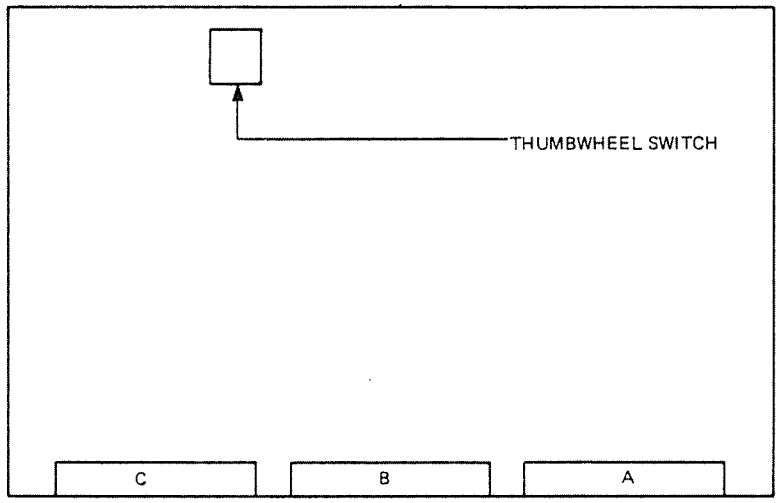
Lower limit	Size	Upper limit	Address range
0 0	16K	0 1	0 - 16K
0 0	32K	0 2	0 - 32K
0 0	64K	0 4	0 - 64K
0 3	32K	0 5	48 - 80K
0 3	64K	0 7	48 - 112K
3 4	64K	4 0	448 - 512K

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.

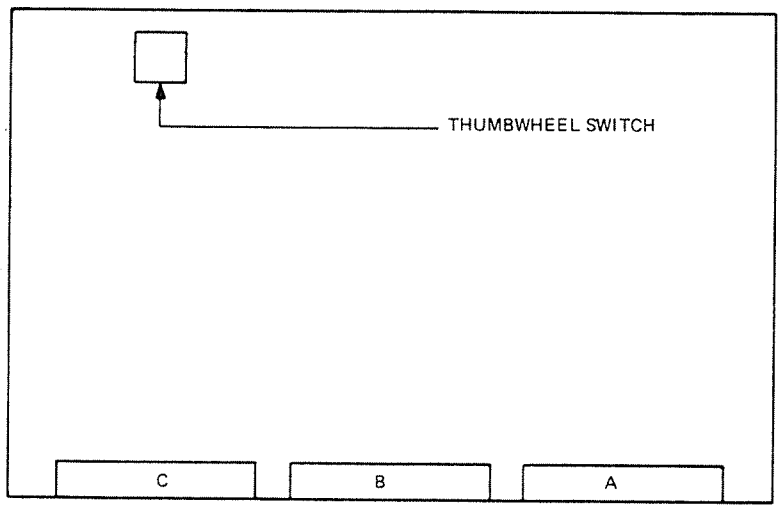
D.4 *SWITCHES ON THE 10MB DISK MODULE (3004)*



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.

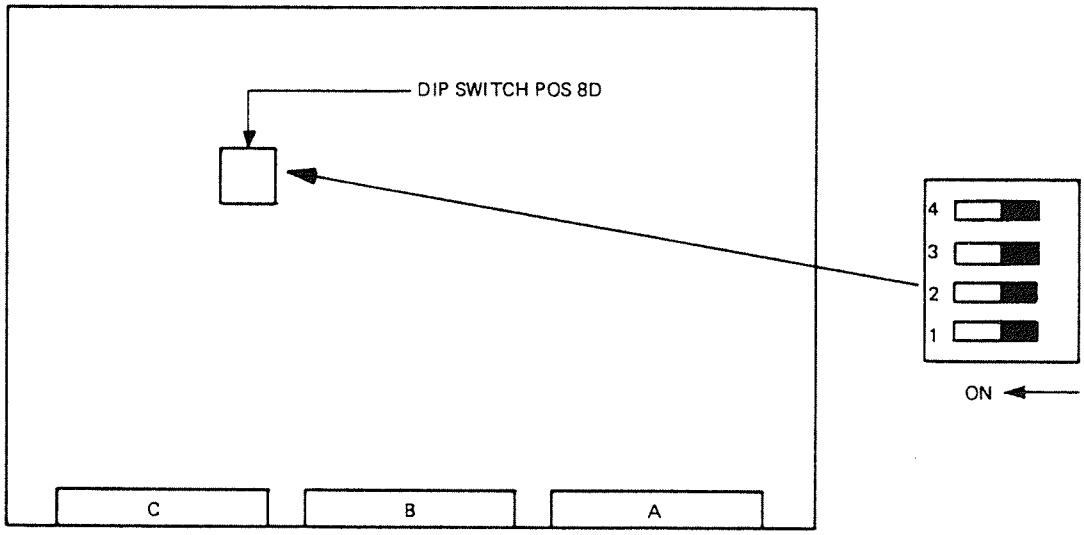
D.5 *SWITCH SETTING ON THE PERTEC MAGNETIC TAPE MODULE (3006)*



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.

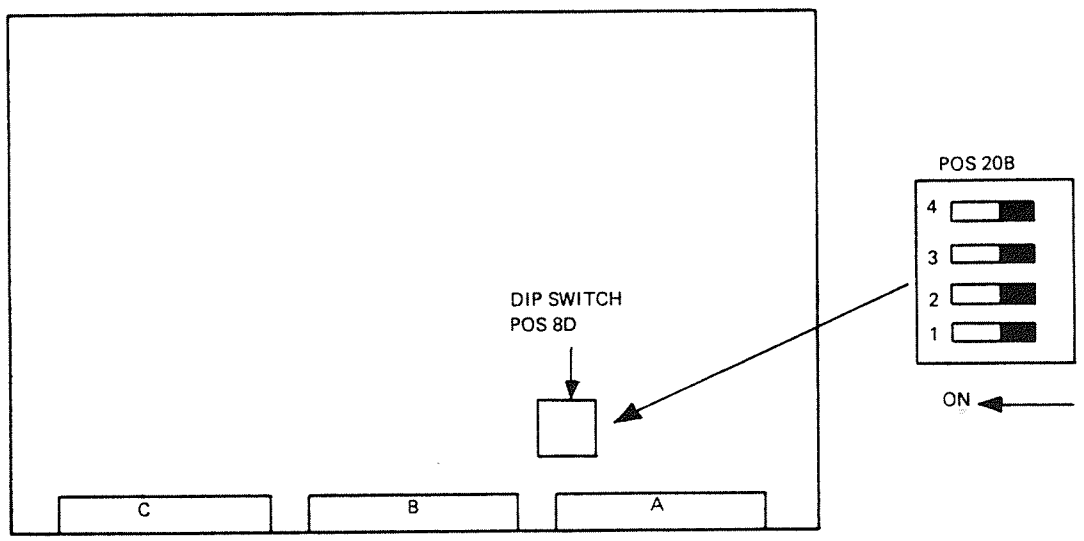
D.6 *SWITCH SETTING ON NORD-100 BUS ADAPTER (3008)*



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.

D.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 E

MICROPROGRAM-PANEL PROCESSOR COMMUNICATION

E.1 *INFORMATION TO PANEL PROCESSOR*

The μ-program can load the panel control register as an internal register with number 40_8. This register is 8 bit wide and information is sent in packets consisting of 2-6 bytes. The first byte is a command code telling what the succeeding bytes means. There are 8 possible packets of display information that the panel will accept from μ-program.

The DATA and ADR displays need up to 3 bytes of information:

bits 0-7, 8-15 and 16-23

The FUNCTION display needs up to 4 bytes of ASCII-codes numbered 0-3 from right to left. Some packets are INFO to the processor that is not displayed directly.

The 8 packets are as follows:

MEMORY EXAMINE:

1. 10
2. ADR 16-23
3. ADR 0-7
4. ADR 8-15
5. DATA 0-7
6. DATA 8-15

REGISTER EXAMINE:

1. 11
2. DATA 0-7
3. DATA 8-15
4. ADR 0-7
5. ADR 8-15

DISPLAY ACTIVITY:

1. 12
2. DATA 0-7
3. DATA 8-15

DISPLAY MEMORY BUS

1. 13
2. DATA 16-23
3. DATA 0-7
4. DATA 8-15

REGISTER LABEL:

1. 16
2. FUNC 2
3. FUNC 3
4. FUNC 0
5. FUNC 1

DISPLAYED FORMAT:

1. 17
2. INFO 0-7
3. INFO 8-15

EXAMINE TYPE:

1. 15
2. INFO 0-2

MEMORY BUS TYPE:

1. 14
2. INFO 0-4

E.2 *PANEL INTERRUPT*

When the Panel Processor has completed the latest command and updated the displays, it issues an interrupt, PANREQ, to microprogram. This interrupt triggers the MOPC routine which reads panel status, internal register 40. This reading resets PANREQ. Then MOPC takes a sample of the information previously requested by the operator and sends it to the Panel Processor.

In case of Panel Processor malfunction, Operators Communication may halt due to missing panel interrupts. If such a fault is suspected, IC type 8156 in position 25D on Memory Management can be removed from the socket. This will make the microprogram believe that there is no panel present and then it functions as explained in Section 1.3.

E.3 *PANEL STATUS*

The μ-program can read panel status as an internal register with 40_8. It is 16 bits wide, but it is only interested in:

Bit 15 = Panel Present

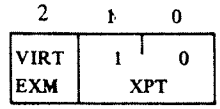
At each clock interrupt (normally every 20ms) the microprogram readsd panel-status to see if a panel processor is present. If true, MOPC is not triggered because that shall be postponed until a panel-interrupt appears. If a Panel Processor is *not* present there will never come any panel-interrupt. Therefore, the MOPC routine is enterted each 20ms in absence of Panel Processor.

APPENDIX F

MICROPROGRAMMABLE REGISTERS ON MEMORY MANAGEMENT

F.1 *WRITE ONLY REGISTERS*

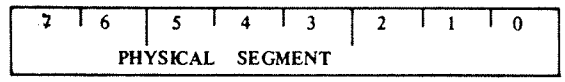
IR NO. 21 Examine Mode



Bit 2 = 1 Enable virtual examine via page table no. XPT
= 0 Physical examine with address bits 16-23 from segment register.

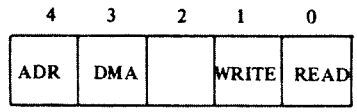
Bit 0-1 = XPT, Page table number used at virtual examine.

IR NO. 22 Segment Register



Bit 0-7 = PSEG. Number of 64k segment at physical examine.
PSEG. 0-7 are used directly as memory address 16-23.

IR NO. 23 Select Memory Bus Register



Selects which Memory Bus information that should be put into the Bus Monitor register.

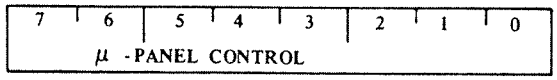
Bit 4 = 1 Strobe Memory Address
= 0 Strobe Memory Data

Bit 3 = 1 Strobe on DMA-cycles
= 0 Strobe on CPU-cycles

Bit 1 = 1 Strobe Write accesses

Bit 0 = 1 Strobe Read accesses

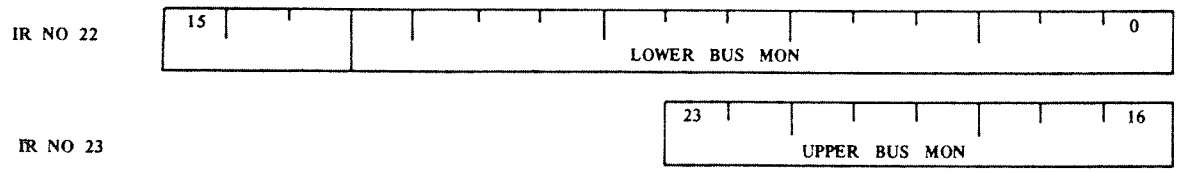
IR NO. 40 Panel Control Register



Bit 0-7 = μPANC. Used by panel processor for display information and control. See Appendix E.

F.2 *READ ONLY REGISTERS*

IR NO. 22 and 23 Bus Monitor Register



Bit 0-23 = Content of memory bus sampled according to "Select Memory Bus" register.

IR NO. 40 Panel Status Register

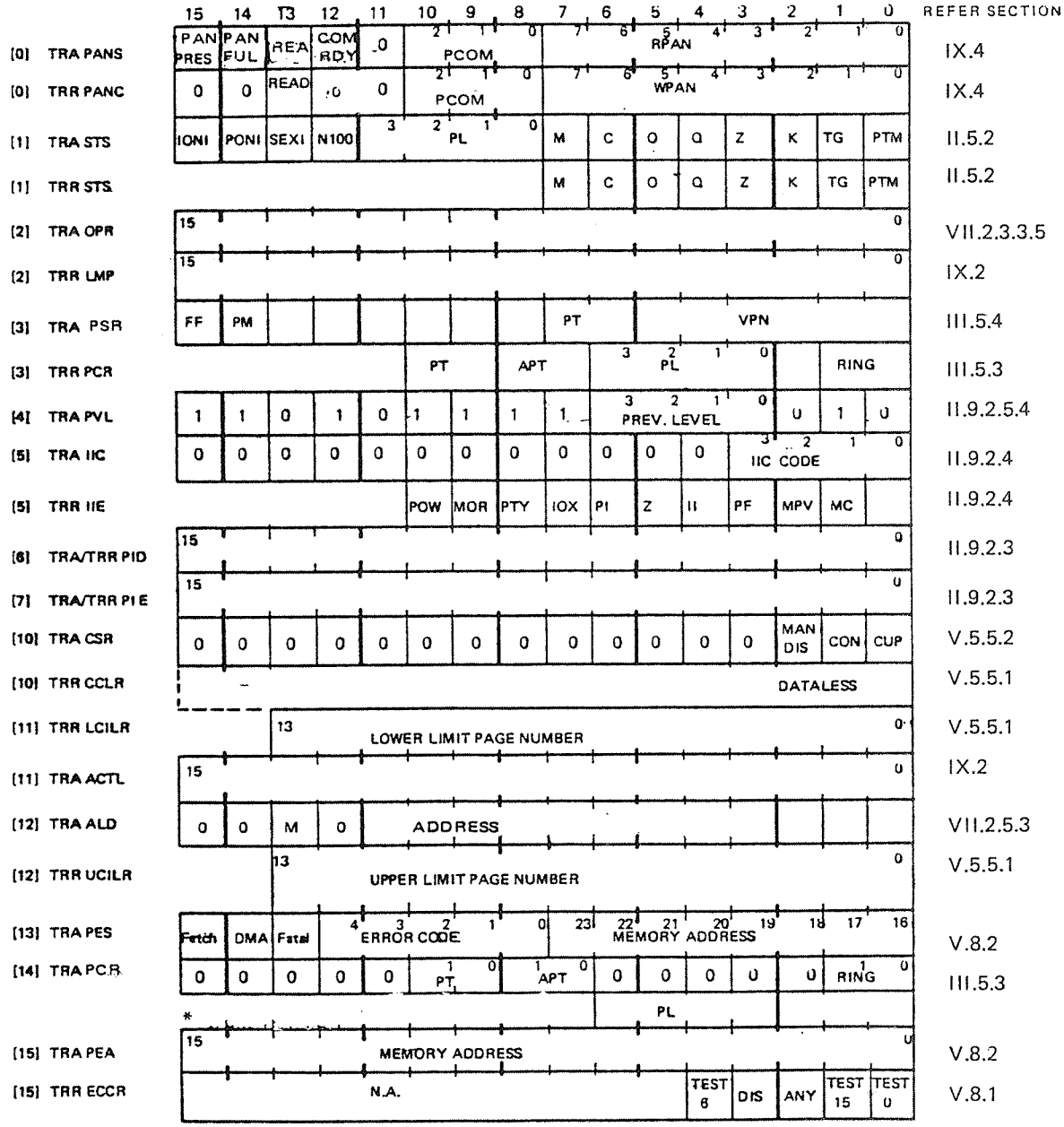
Bit 15 = 1 Panel option is installed. MOPC is triggered by panel interrupt.

= 0 Panel not present. MOPC is triggered by 20ms clock interrupt.

Reading of panel status also clears panel-request.

APPENDIX G

INTERNAL REGISTERS AND THEIR BIT ASSIGNMENT



* Require level information in A register before TRA PCR

APPENDIX H

OPERATOR'S COMMUNICATION INSTRUCTION SURVEY

H.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

H.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)

H.3 *MONITOR 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 I

NORD-100 TECHNICAL SPECIFICATIONS

I.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

I.2 *PHYSICAL*

NORD-100 CPU Crate, Rack Mountable:

Dimensions

Height:	400 mm
Width:	482 mm
Depth:	505 mm

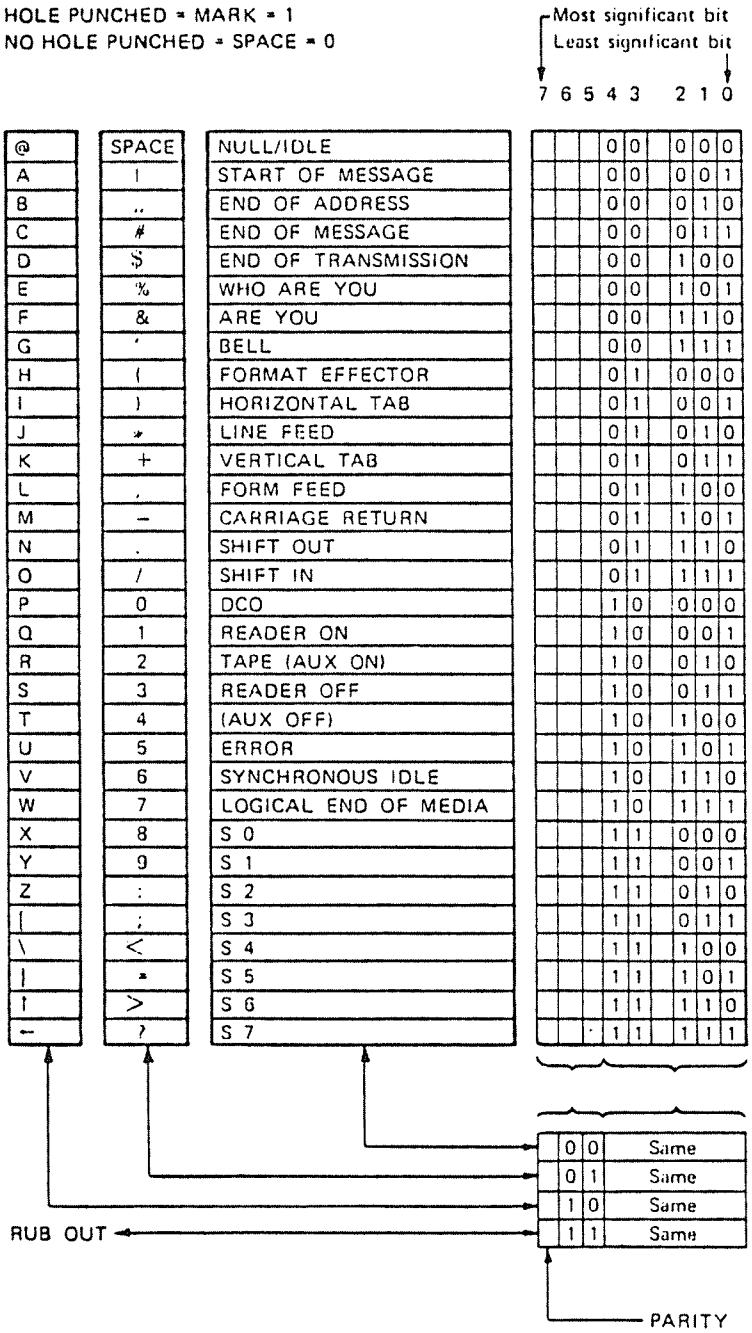
Can be mounted in 19 inch cabinets of various heights, depending on configuration.

Power:	230V, range 198 - 264V (115V, range 90-132V) 45-440 Hz Max. 2 Amp. 230V

APPENDIX J

MODEL 33 ASR/KSR TELETYPE CODE (ASCII) IN BINARY FORM

HOLE PUNCHED = MARK = 1
NO HOLE PUNCHED = SPACE = 0



			7	6	5	4	3	2	1	0
@	SPACE	NULL/IDLE				0	0	0	0	0
A	!	START OF MESSAGE				0	0	0	0	1
B	"	END OF ADDRESS				0	0	0	1	0
C	#	END OF MESSAGE				0	0	0	1	1
D	$	END OF TRANSMISSION				0	0	1	0	0
E	%	WHO ARE YOU				0	0	1	0	1
F	&	ARE YOU				0	0	1	1	0
G	'	BELL				0	0	1	1	1
H	(	FORMAT EFFECTOR				0	1	0	0	0
I	)	HORIZONTAL TAB				0	1	0	0	1
J	*	LINE FEED				0	1	0	1	0
K	+	VERTICAL TAB				0	1	0	1	1
L	,	FORM FEED				0	1	1	0	0
M	–	CARRIAGE RETURN				0	1	1	0	1
N	.	SHIFT OUT				0	1	1	1	0
O	/	SHIFT IN				0	1	1	1	1
P	0	DCO				1	0	0	0	0
Q	1	READER ON				1	0	0	0	1
R	2	TAPE (AUX ON)				1	0	0	1	0
S	3	READER OFF				1	0	0	1	1
T	4	(AUX OFF)				1	0	1	0	0
U	5	ERROR				1	0	1	0	1
V	6	SYNCHRONOUS IDLE				1	0	1	1	0
W	7	LOGICAL END OF MEDIA				1	0	1	1	1
X	8	S 0				1	1	0	0	0
Y	9	S 1				1	1	0	0	1
Z	:	S 2				1	1	0	1	0
[	;	S 3				1	1	0	1	1
\	<	S 4				1	1	1	0	0
]	=	S 5				1	1	1	0	1
↑	>	S 6				1	1	1	1	0
←	?	S 7	P			1	1	1	1	1

0	0	Same
0	1	Same
1	0	Same
1	1	Same

INDEX OF ABBREVIATIONS

MOP	MEMORY OUT OF RANGE	II.9.2.4
MOS	METAL OXIDE SEMICONDUCTOR	V.4
MPM	MULTIPORT MEMORY	V.10
MPV	MEMORY PROTECT VIOLATION	III.4.3
MSB	MOST SIGNIFICANT BIT	II.6.4
MSI	MEDIUM SCALE INTEGRATION	II.3.2
MTBF	MEAN TIME BETWEEN FAILURE	V.7.1
NA	NOT ASSIGNED	II.9.2.4
O	STATIC OVERFLOW INDICATOR	II.5.2
OPR	OPERATOR'S REGISTER	VII.2.3.3.5
P	P-REGISTER (PROGRAM COUNTER)	II.5.1
PA	PHYSICAL ADDRESS	III.3
PANC	PANEL CONTROL REGISTER	IX.4
PANS	PANEL STATUS REGISTER	IX.4
PC	PROGRAM COUNTER	II.4.2
PCR	PAGING CONTROL REGISTER	III.5.3
PEA	PARITY ERROR ADDRESS REGISTER	V.8.2
PES	PARITY ERROR STATUS REGISTER	V.8.2
PF	PAGE FAULT	III.4.3
PGU	PAGE USED	III.4.3
PI	PRIVILEGED INSTRUCTION	II.9.2.4
PID	PRIORITY INTERRUPT DETECT	II.9.2.3
PIE	PRIORITY INTERRUPT ENABLE	II.9.2.3
PIL	CURRENT LEVEL INDICATOR	II.9.2.3
PIO	PROGRAMMED INPUT/OUTPUT	VI.2
PK	PRIORITY CODE	II.9.2.3
PONI	PAGING ON INDICATOR	II.5.2
POW	POWER FAIL INTERRUPT	II.9.2.4
PPN	PHYSICAL PAGE NUMBER	III.3.1
PROM	PROGRAMMED READ ONLY MEMORY	II.3.2
PSR	PAGING STATUS REGISTER	III.5.4
PT	PAGE TABLE	III.2.3
PTM	PAGE TABLE MODUS	II.5.2
PTY	PARITY ERROR INTERRUPT	II.9.2.4
PVL	PREVIOUS LEVEL INDICATOR	II.9.2.3
Q	DYNAMIC OVERFLOW INDICATOR	II.5.2
RAM	RANDOM ACCESS MEMORY	V.4.1
RAS	ROW ADDRESS STROBE	V.4.3
ROM	READ ONLY MEMORY	II.1
RPM	READ PERMITTED	III.4.3
RT	REAL TIME PROGRAM	III.3.3
SEXI	EXTENDED ADDRESS MODE INDICATOR	II.5.2
STB	STAND BY POWER	VIII.1.3
STS	STATUS REGISTER	II.5.2
T	T-REGISTER (TEMPORARY)	II.5.1
TG	FLOATING ROUNDING INDICATOR	II.5.2
U	USED (VALID DATA)IN CACHE	V.5.3
UCIL	UPPER CACHE INHIBIT LIMIT REGISTER	V.5.5.1
UL	UPPER LIMIT	V.6.2.2
V	VOLT	VII.1.1
VA	VIRTUAL ADDRESS	III.1
WCS	WRITABLE CONTROL STORE	IX.3
WDA	WRITE DATA AND ADDRESS REGISTER	V.6.5
WIP	WRITTEN IN PAGE	III.4.3
WPM	WRITE PERMITTED	III.4.3
WT	WRITE THROUGH	V.5.2.3
X	X-REGISTER (POST INDEX REGISTER)	II.5.1
Z	ERROR INDICATOR	II.9.2.4

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

COMMENT AND EVALUATION SHEET

NORD-100 FUNCTIONAL DESCRIPTION ND-06.015.01

August 1980

In order for this manual to develop to the point where it best suits your needs, we must have your comments, corrections, suggestions for additions, etc. Please write down your comments on this preaddressed form and mail it. Please be specific wherever possible.

FROM .....

.....

.....

– we make bits for the future

NORSK DATA A.S BOX 4 LINDEBERG GÅRD OSLO 10 NORWAY PHONE: 39 16 01 TELEX: 18661