ND-100
Functional Description

ND-06.015.02

NOTICE

PRINTING RECORD

Printing	Notes
08/80	Original Printing – Version 01
04/85	Version 02

ND -100 Functional Description
Publ No. ND-60.015.02

Norsk Data

Norsk Data A.S
Graphic Center
P.O.Box 25, Bogerud
0621 Oslo 6, 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 and comments should be sent to:

Documentation Department
Norsk Data A.S
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

Requests for documentation should be sent to the local ND office or (in Norway) to:

Graphic Center
Norsk Data A.S
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

PREFACE

The Reader

The ND-100 Functional Description describes the ND-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 ND-100 computer system. Since the ND-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 ND-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 each section may be read independently. After an introduction in Section 1, the manual starts in Section 2 with a description of the central processor. In Section 3 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, the ND-100 bus, is described in Section 4. The storage system is discussed in Section 5. The communication with peripherals, the input/output system, is described in Section 6. The operator's interaction is taken care of in Section 7, while a short description of the power supply is given in Section 8. Section 9 gives more detailed and specified information of certain parts. Appendices are mostly references for special information while working with the ND-100.

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

ND-100 Reference Manual (ND-06.014)
ND-100 Input/Output System (ND-06.016)
ND-100 Microprogramming Description (ND-06.018)

TABLE OF CONTENTS

+ + +

Section: *Page:*

Section: *Page:*

Section: *Page:*

Section: *Page:*

Section: *Page:*

Section: *Page:*

Section: *Page:*

Section: *Page:*

Section: *Page:*

Section: *Page:*

Section: *Page:*

Section: *Page:*

Section: *Page:*

1 ND-100 ARCHITECTURE

1.1 INTRODUCTION

ND-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 64 K words (128 K bytes) without the memory management system, and 16 M words (32 M bytes) with the memory management system. It is completely software compatible with NORD-10/S and runs the same operating system, SINTRAN III.

1.2 THE INSTRUCTION SET

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

— bits
  bytes
  single words
  double words
  triple words
  register 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 ND-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 storage of the register blocks and interprogram level read/write operations.

The ND-100 is microprogrammed. All instructions are executed by firmware residing in a 2 K by 64 bits programmable read only memory (PROM) called the microprogram control store. By expanding the microprogram PROM to 4 K by 64 bits, a number of instructions are introduced. These instructions comprise what is known as the CX-option. In the CX-option you find:

— decimal instructions
— stack handling instructions
— SINTRAN III segment-change instructions
— MOVEW (move block of words) instruction
— TSET (test and set) instruction
— RDUS (read do not use cache) instruction

To allow dynamic microprogramming, a 256 word by 64 bits writable control store is available as an option. This makes it possible for software to extend the ND-100 instruction set for special applications.

The ND-100 instructions are described in the manual ND-100 Reference Manual, ND-06.014.

1.3 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 ND-100 can be changed by rewriting the microprogram.

1.4 THE BUS STRUCTURE

The main highway for addresses and data in the system is the ND-100 BUS, a multiplexed address and data bus. All communication between ND-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 ND-100 bus, the CPU has the same easy access to peripherals as it has to memory.

1.5 FUNCTIONAL MODULES

1.5.1 General

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

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

Figure 1.5.1 shows the ND-100 module connection.

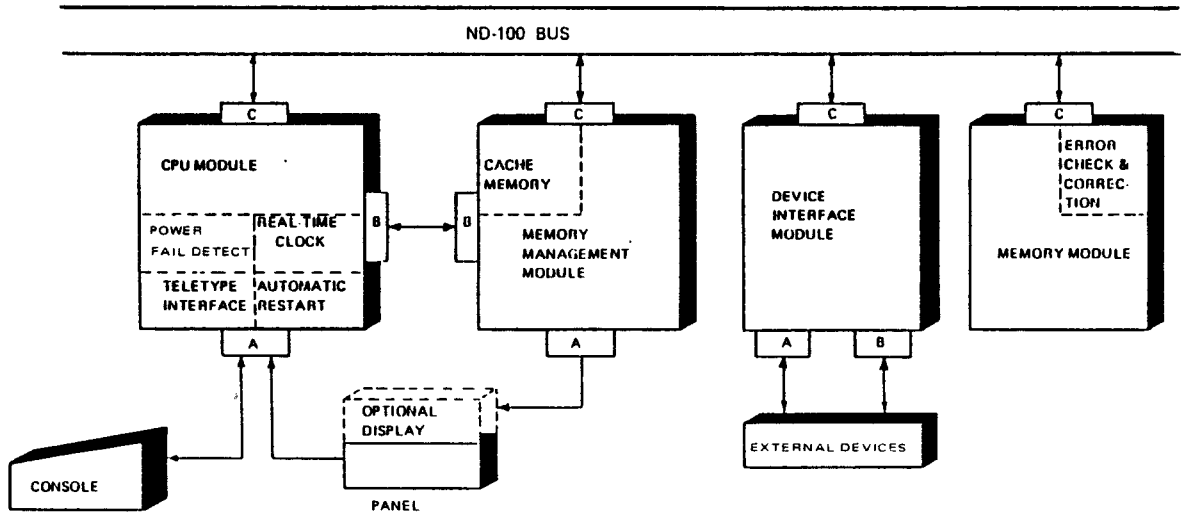


Figure 1.5.1: ND-100 Module Connection

1.5.2 The Central Processing Unit (CPU) Module

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

- a real-time clock
- a terminal interface with switch selectable speeds, 50 - 9600 bps (bits per second)
- power fail and automatic restart

THE PROCESSOR:

ND-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 ND-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 ND-100 CPU will fetch the next instruction at the same time as the current instruction is executed.

THE INTERRUPT SYSTEM:

The ND-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.

1.5.3 The Memory Management System

The hardware memory management module (necessary to run the SINTRAN III/VS, Virtual Storage, operating system) includes:

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

— dynamic allocation/relocation of programs in memory

— memory protection

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

— The Paging System
— The Memory Protection System

THE PAGING SYSTEM:

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 64 K word virtual memory) into a 19-bit physical address. In this mode the physical address space can be extended up to 512 K words (1 M byte).

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

The implementation of paging is based on dividing physical memory into 1 K 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.

1.5.4 The Memory System

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

— 1 K word (2 K byte) CACHE memory
— Up to 16 M word (32 M byte) main memory
— Memory channel to the multiport memory system

MAIN MEMORY:

Main memory can have any size from:

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

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

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

One memory module 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 is physically located on the memory management module.

MULTIPORT MEMORY:

In order for the ND-100 to access the 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.

1.5.5 The Input/Output System

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

Depending on the speed, a device could be connected to the ND-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, ie., 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 1.8 M words per second.

1.6 THE ND CABINETS

The ND-100 computer is delivered in two cabinet sizes, small and large. The small cabinet houses a 12-position card crate, while the large cabinet's card crate can hold 21 cards.

1.6.1 The Small Cabinet

Figure 1.6.1 shows the small ND-100 cabinet, which is 54 cm wide, 96 cm high and 91 cm deep.

The operator's panel is vertically divided in two, the push button panel and the display panel. The push button panel is located to the right and contains four push buttons and a key for locking the panel. The display panel (option) is located to the left. The B panel 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. Chache hit ration and degree of utilization of the CPU are also montiored by the display microprocessor.

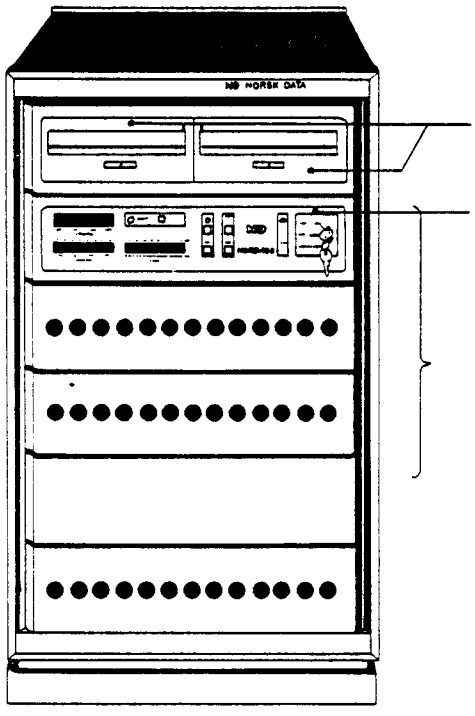


Figure 1.6.1:
ND-100 Small Cabinet, 12 Positions.

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

1.6.2 The Large Cabinet

Figure 1.6.2 shows the large ND-100 cabinet, which is 60 cm wide, 169 cm high and 91 cm deep. The operator's panel is the same as on the small cabinet described in section 1.6.1. It is possible to install two card crates in the large cabinet.

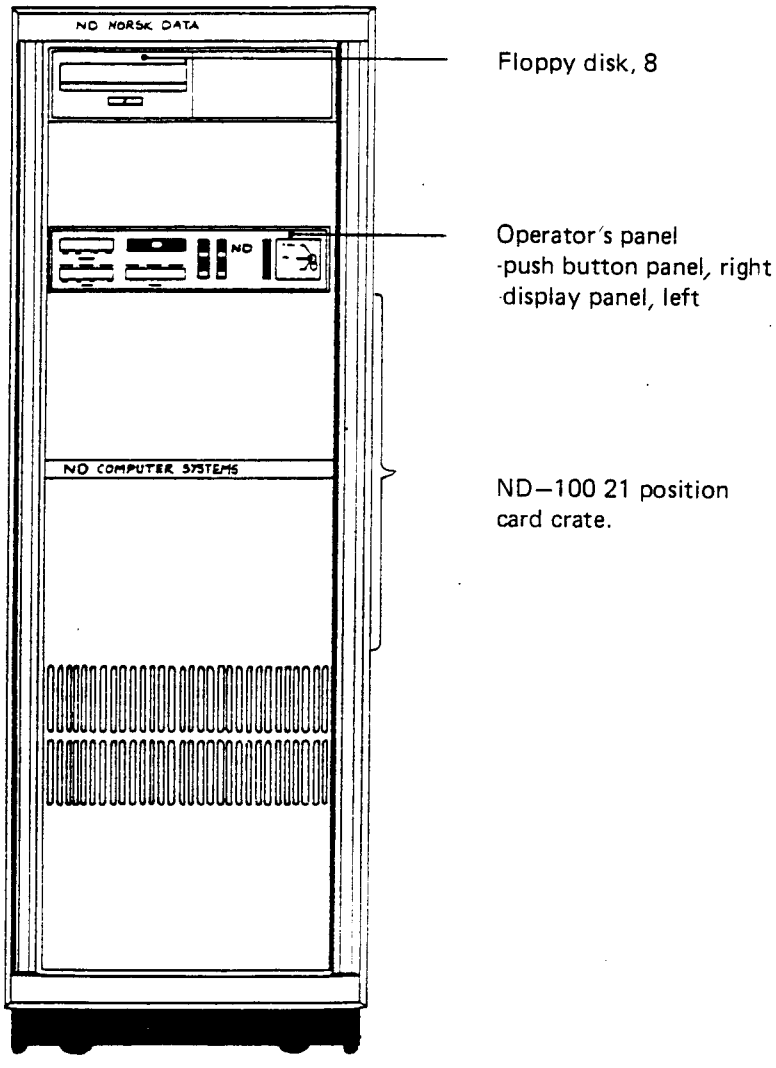


Figure 1.6.2: ND-100 Large Cabinet, 21 Positions.

2 CENTRAL PROCESSOR

2.1 GENERAL

ND-100 is based upon a microprogrammed CPU architecture. A microprogrammed architecture means that:

- 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 ND-100 bus.

2.2 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 2.2.1 shows how the IDB communicates with the central parts in the CPU, the memory management and cache, and with the sources connected to the ND-100 bus.

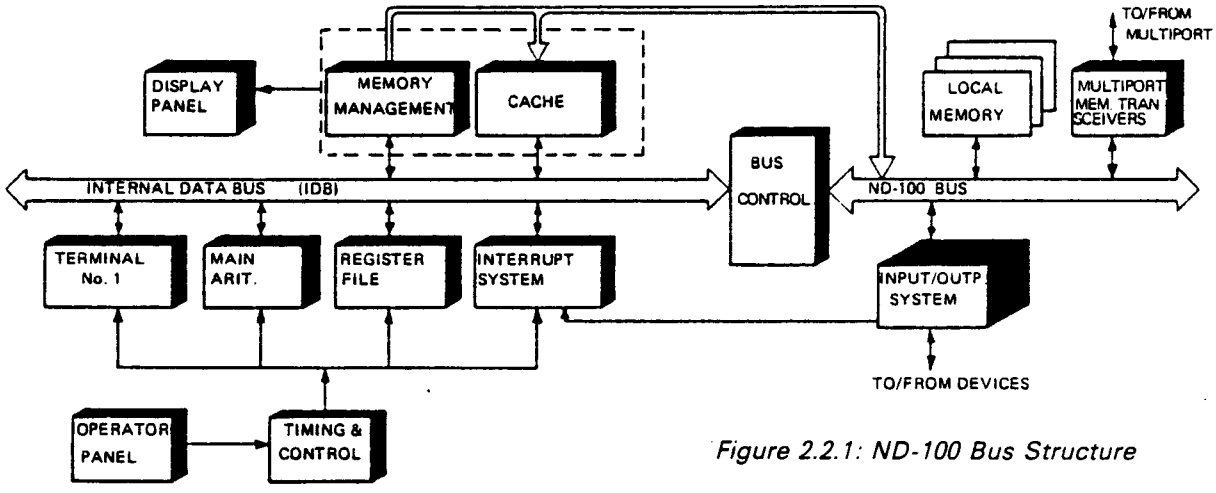


Figure 2.2.1: ND-100 Bus Structure

The main parts communicating with the IDB and with the ND-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, ie., the part in the processor that computes. It also contains the current register block.

The 16-bit virtual addresses are also calculated here, under the 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 ND-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 ND-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 micro processor on the memory management module.

— Bus Control

The ND-100 bus is controlled by the CPU board. All requests for the ND-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 ND-100 bus. Direct memory access devices, which operate directly on memory, communicate over the ND-100 bus.

— Local Memory

This contains memory modules with up to 1 M words on each module. Local memory is directly connected to the ND-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 ND-100 bus and the multiport memory system.

2.3 FUNDAMENTAL BUILDING BLOCKS

2.3.1 General Considerations

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

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

The microprogram PROM may be expanded to 4 K by 64 bits. This microprogram contains a number of extra instructions called the CX-option.

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

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.

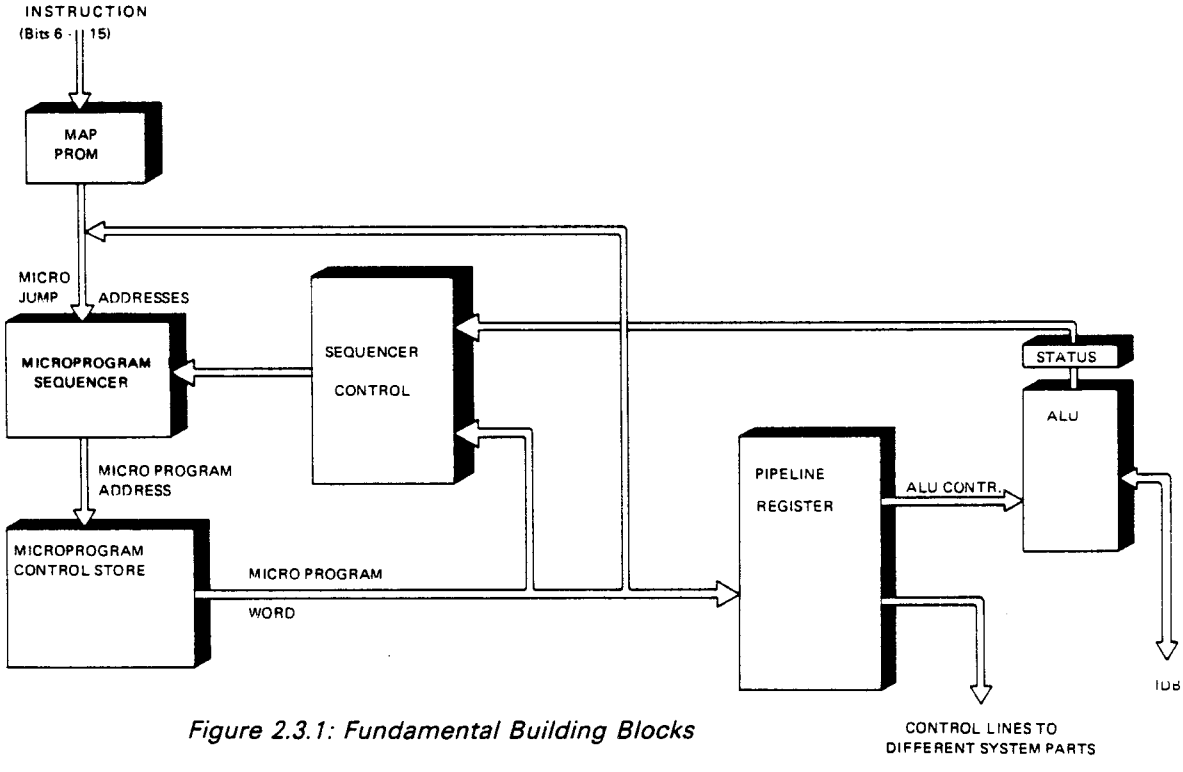


Figure 2.3.1: Fundamental Building Blocks

2.3.2 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.

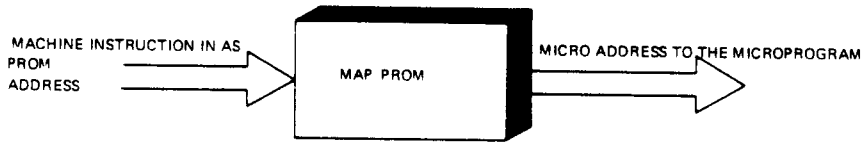


Figure 2.3.2: The Mapping PROM

Several microinstructions may be needed to execute one machine instruction. The microprogram sequencer controls the sequence of these microinstructions. 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 microinstruction. 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, which has a built-in stack for that purpose.

Branch addresses in the microprogram are fed directly back to the microprogram sequencer, to be used as the next control store address. All the microinstruction 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 microinstruction which are fed back to the sequencer to determine the next address is clocked into the microprogram sequencer at the same time as 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.

2.4 INSTRUCTION FETCH AND EXECUTION

2.4.1 Instruction Readout

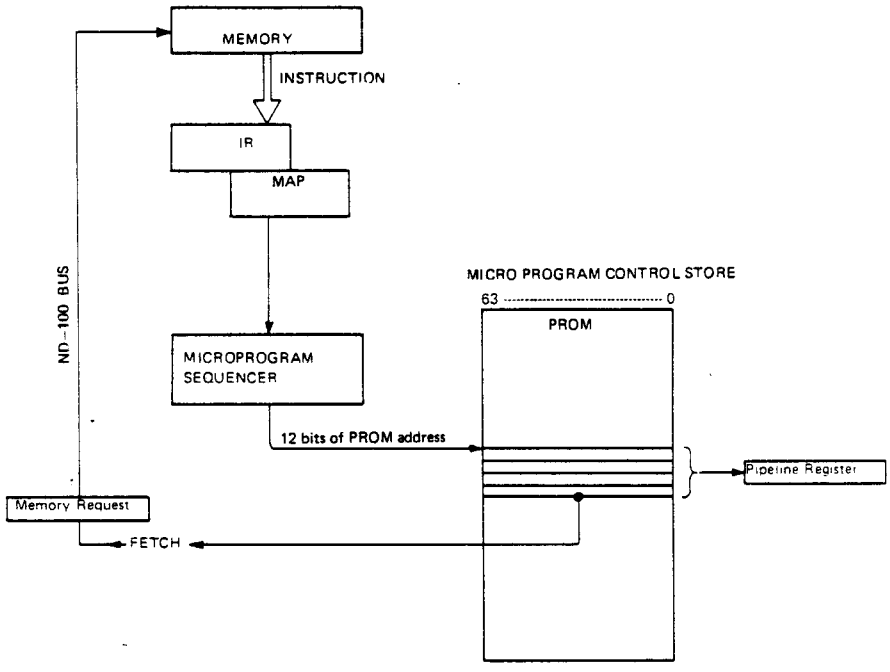


Figure 2.4.1: Instruction Readout

The machine instructions to be executed reside in memory. When the microinstruction sequence of one machine instruction is finished, a fetch for a new machine instruction is issued. The program counter is enabled onto the ND-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 2 K by 12 bits 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 microprogram 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 addresses and their sequence. The map contains two almost equal entry point maps, one containing all ND-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 normally take either 190 ns or 150 ns, depending on whether it is a slow or a fast version of the ND-100. This time it is referred to as a microcycle, or the internal CPU cycle time. When a microcycle 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.

2.4.2 Prefetch

ND-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 ND-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 is to be executed, it will most often be available.

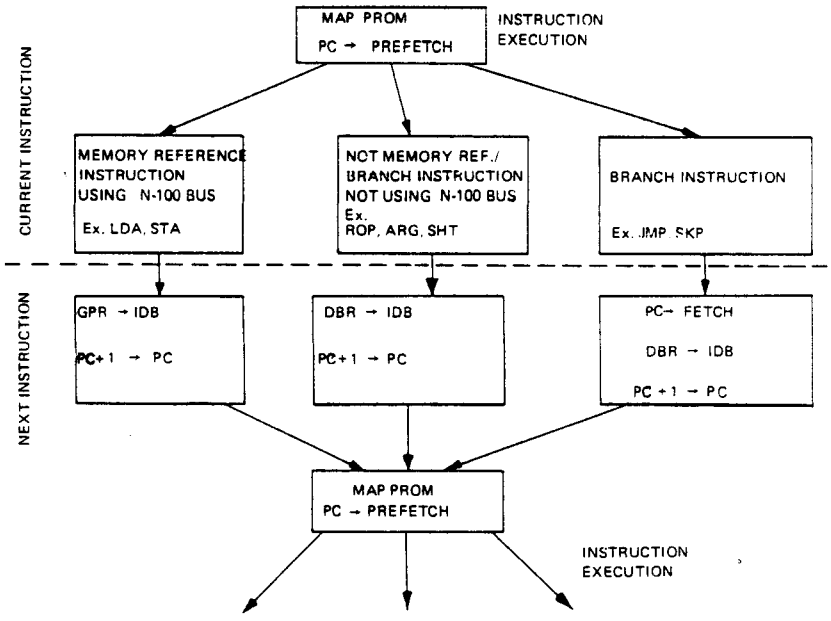


Figure 2.4.2: Instruction Prefetch

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 instructions 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 ND-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 upon the instruction operands.

Example:

JMP, SKP, BSKP

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

3. Other Instructions

These are instruction which do not make memory references, ie., do not use the ND-100 bus and do not 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 ND-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 2.4.2.

When the current instruction is finished, the next instruction is 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 is repeated.

If the current instruction is one in group three, the prefetched instruction is loaded (automatically by hardware) into the DBR (Data Bus Read register). Because these instructions do not use the ND-100 bus, like the memory reference instructions, the instruction does not 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 is enabled from DBR to the internal data bus for start of execution. At the same time, the program counter is incremented by one, and when the execution starts, this program counter is used for a prefetch request.

So, in the case of branch instructions the prefetched instruction is skipped and a new instruction is found. The maximum benefit of prefetch is therefore 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.

2.4.3 Instruction Execution

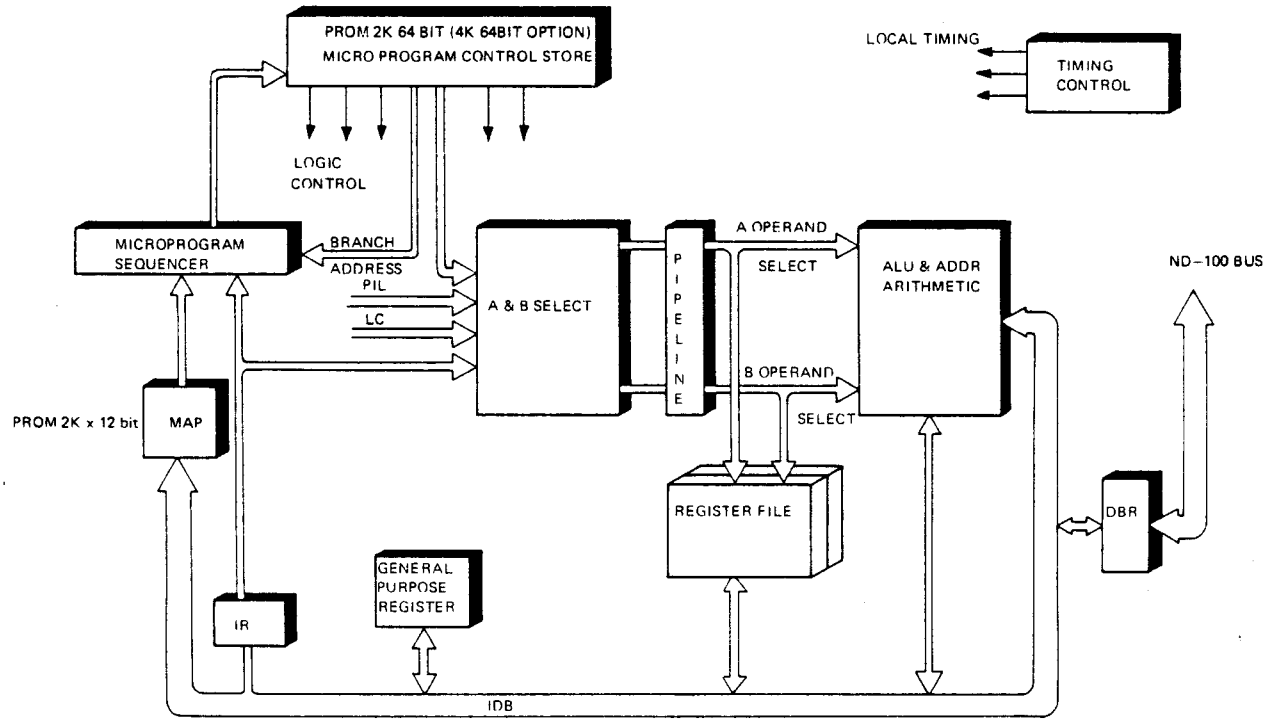


Figure 2.4.3: The Instruction Part

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 microinstructions 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 the microprogram control store is called a microinstruction and the time required to execute a microinstruction is called a microcycle.

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, the address mode 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 microprogram. 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 takes place over the IDB.

The ALU is controlled from the microprogram and, information concerning which operands to use in ALU operations is 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 take 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 take 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 ND-100 Microprogram Description manual).

2.5 THE REGISTER FILE

Refer to figure 2.5.1.

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

2.5.1 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.

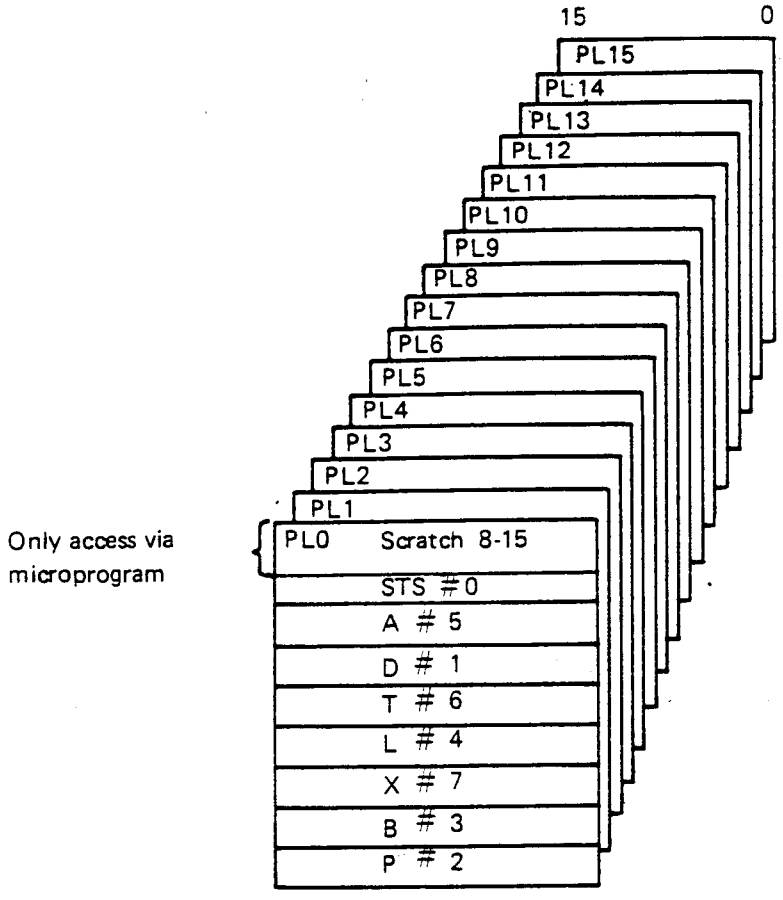


Figure 2.5.1: The Register File

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. All registers or levels can be read or written by specifying register and level information.

2.5.2 Status Indicators

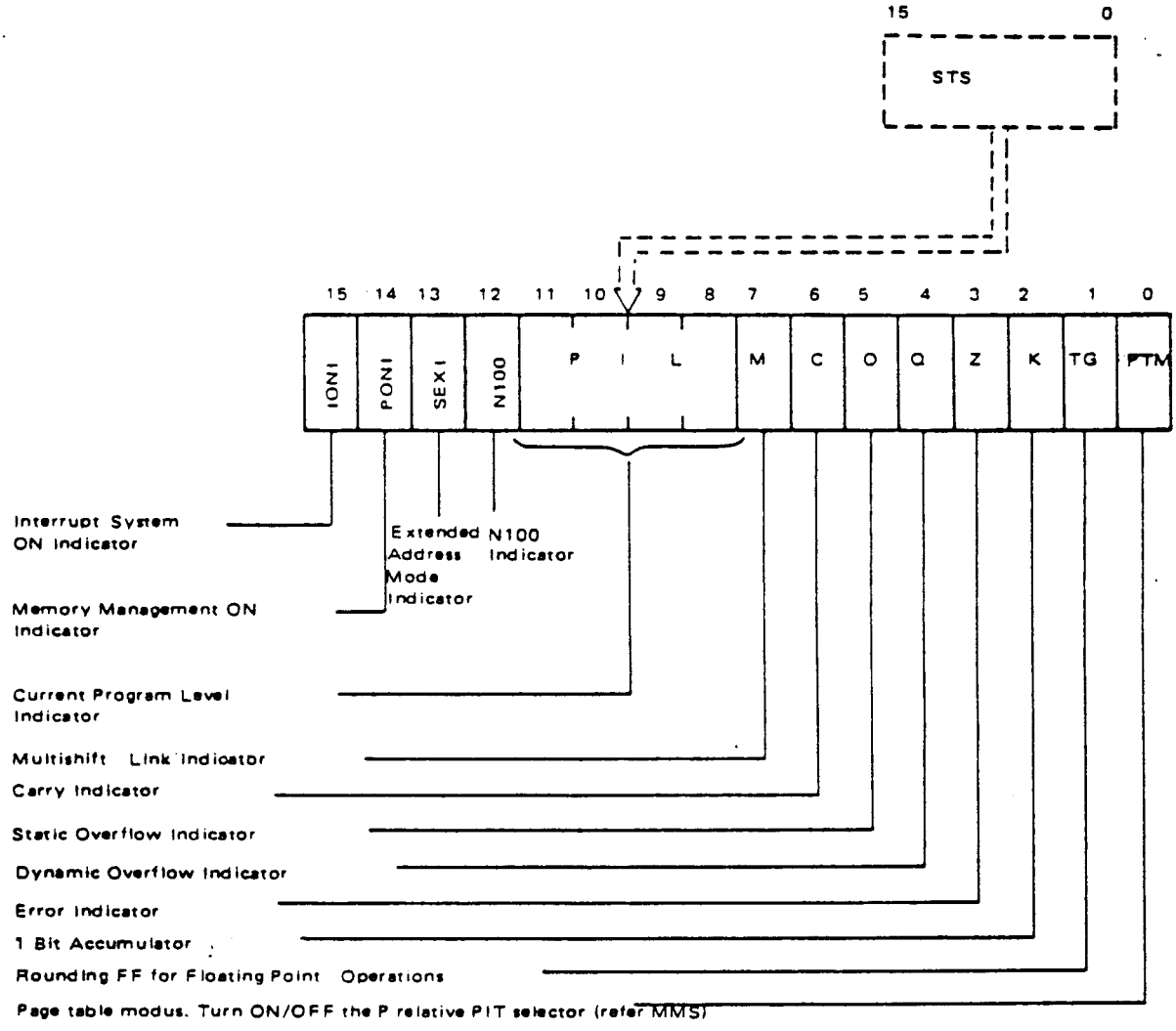


Figure 2.5.2: Status Register Assignment

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 in such a way 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 instructions 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, and not the whole register. Refer to figure 2.5.2.

Figure 2.5.2: 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 ND-100 machine.

PIL — Current program level indicator.

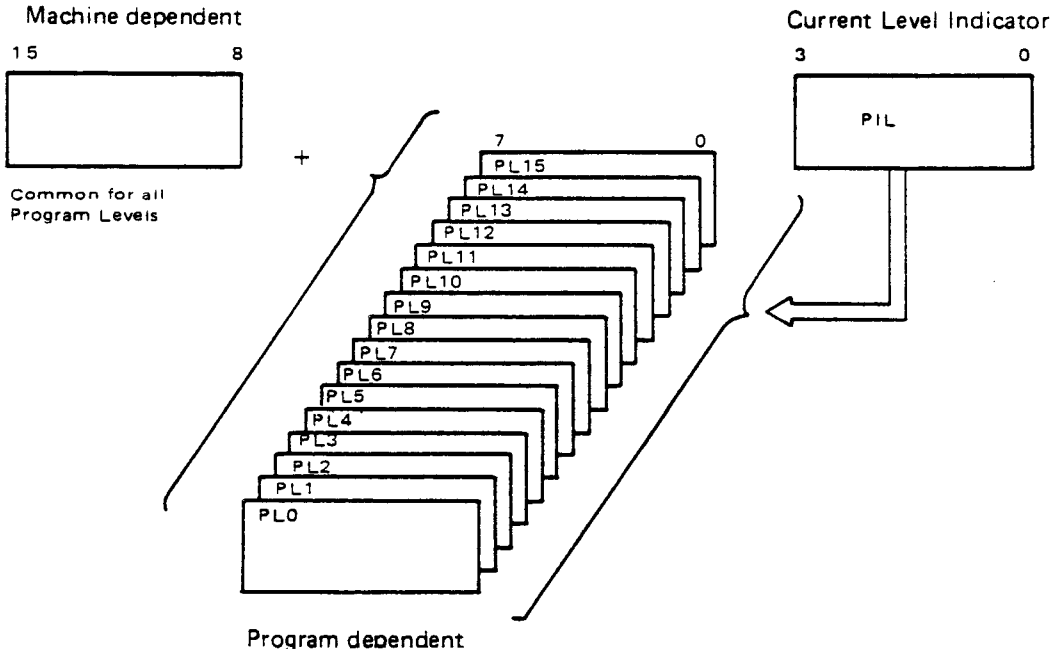


Figure 2.5.3: The Status Register

Figure 2.5.3 shows that STS bits 0-7, as described, have one copy on each level, while STS bits 8-15 are common for all levels.

2.6 MICROPROGRAM SEQUENCER

2.6.1 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: microbranching, micro subroutines and repetitive microinstruction execution.

2.6.2 The Microprogram 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 microinstruction. The microprogram sequencer contains a microaddress 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 microaddress register. This line provides the microaddress register with access either to the current microinstruction (REPEAT), to an address stored in the push/pop stack (RETURN), to the output of the incrementer (NEXT) or to a direct address (JUMP).

The push/pop stack (FILO) 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 simultaneously executing a POP.

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 microaddress, and controls the sequencer stack if a conditional sequence is not specified. These bits indicate the source of the next microaddress:

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 microaddress + 1 into the stack

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

2.6.3 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 microinstruction last executed, 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 the 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.

2.6.4 Functional Flow

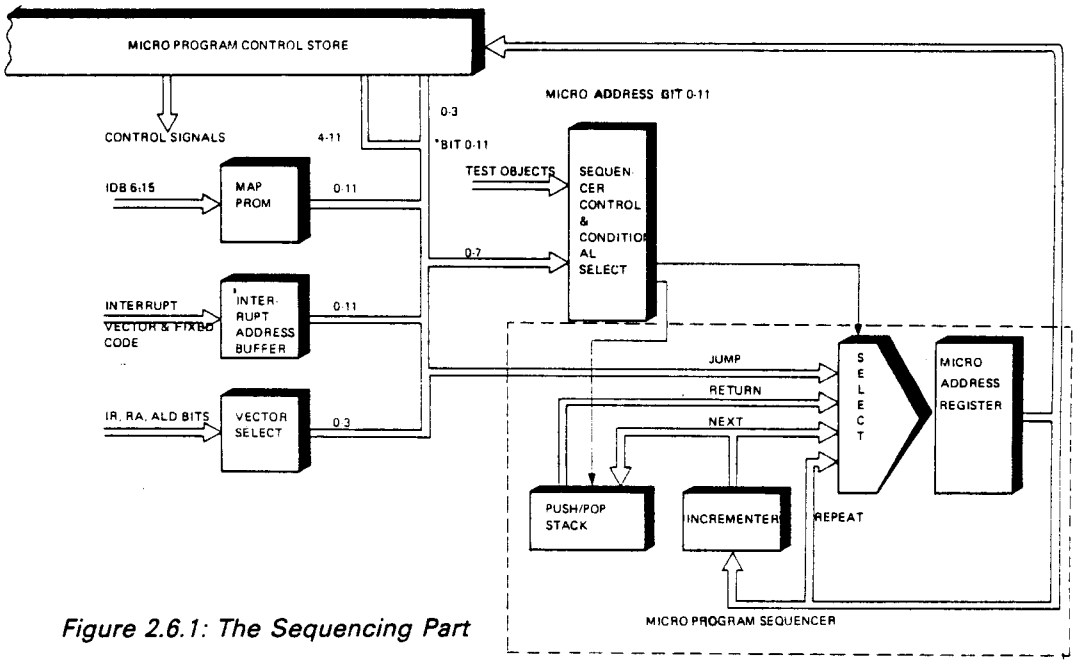


Figure 2.6.1: The Sequencing Part

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 microaddress 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 microprogram 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 is 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 attention is only paid to it at the time when jumps are mapped. When the interrupt is allowed, the 5-bit priority encoded interrupt vector, together with a fixed code, makes a vectored branch in the microprogram. The micro address is generated by the interrupt address buffer, which forces the micro address to the interrupt service vector. Under this operation the map prom and the vector select are disabled.

Example:

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

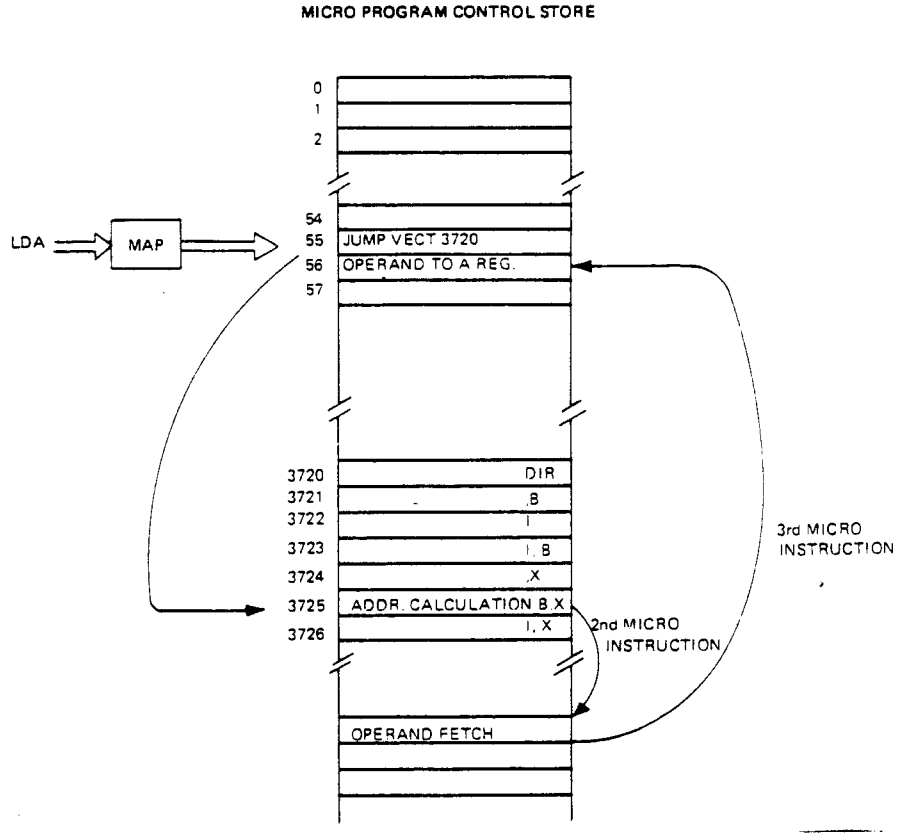


Figure 2.6.2: LDA, B, X Execution

The output of the map PROM will give 55, for instance, 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 contents of the B and the X registers. It also makes a branch to the microinstruction which will generate a read request for the operand. This microinstruction will make a RETURN 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 in location 56.

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

2.7 PIPELINE

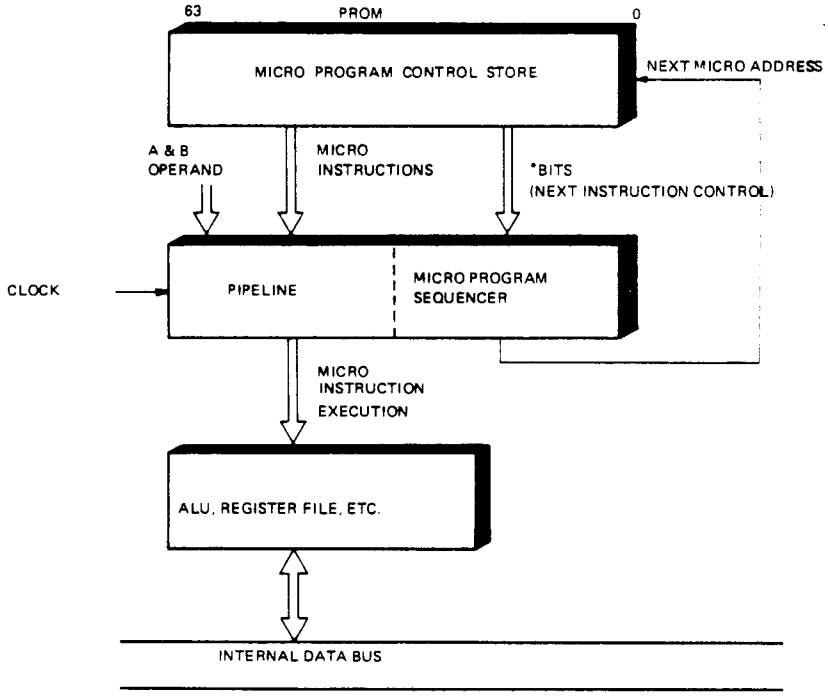


Figure 2.7.1: Pipeline Block Diagram

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, ie., at the input of the pipeline and microprogram sequencer. The bits to the microprogram sequencer are for determination of the next microinstruction to be executed. The part of the microinstruction that goes to the pipeline register is the data manipulation control bits to the different system elements, for control of the data hardware.

When the clock pulse arrives, the pipeline register and the microprogram sequencer are clocked simultaneously.

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 is performed, 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 microinstruction arrives at the input of the pipeline register and the microprogram sequencer. When the clock pulse now arrives, the execution of the next microinstruction is performed and the address for the next microinstruction 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 microcycle. While one microinstruction is executed, the next

microinstruction 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 2.7.2 shows a timing diagram for the events in the pipeline blocks.

The diagram with 'PROM VALID' for the previous microinstruction 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 microaddress 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'. The next microinstruction has now 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 microinstructions 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 sequentially, one after the other. 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.

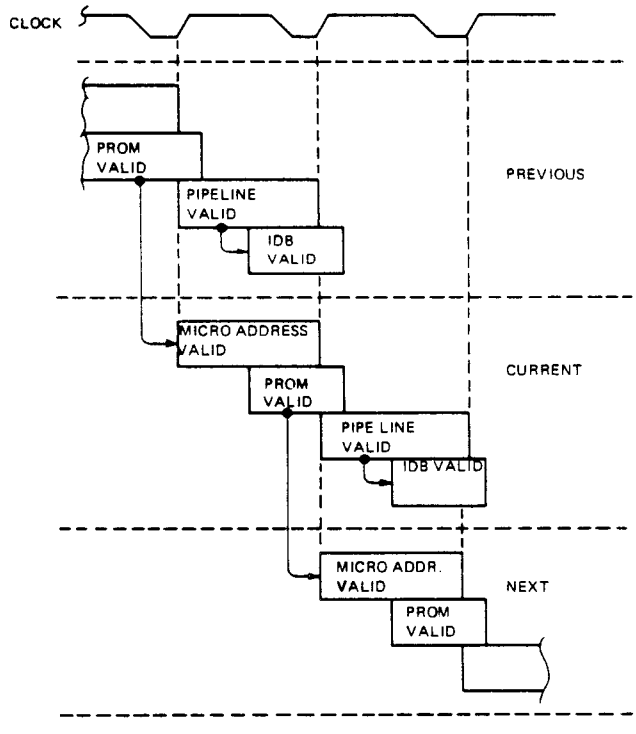


Figure 2.7.2: Pipeline Blocks Timing Diagram

2.8 THE ARITHMETIC LOGIC UNIT

2.8.1 General

The arithmetic logic unit (ALU) is the computing part of the processor. 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 2.8.1 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 of the ALU shift operations with the right in/left out and right out/left in lines.

A operand select and B operand select selects two operands to be operated on in the working register block inside the ALU. 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.

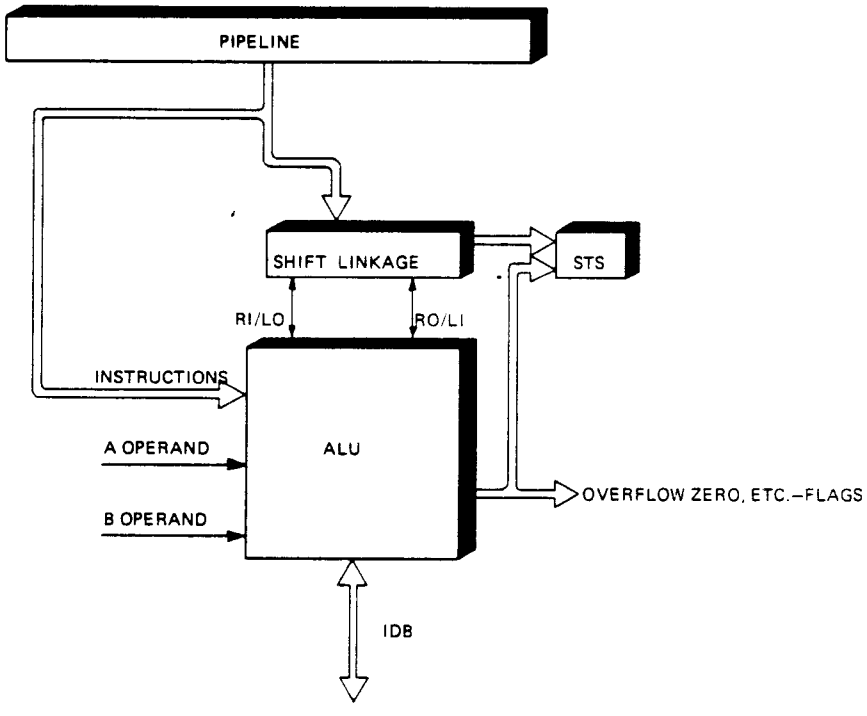


Figure 2.8.1: The Arithmetic Logic Unit

2.8.2 The ALU Bit Slice

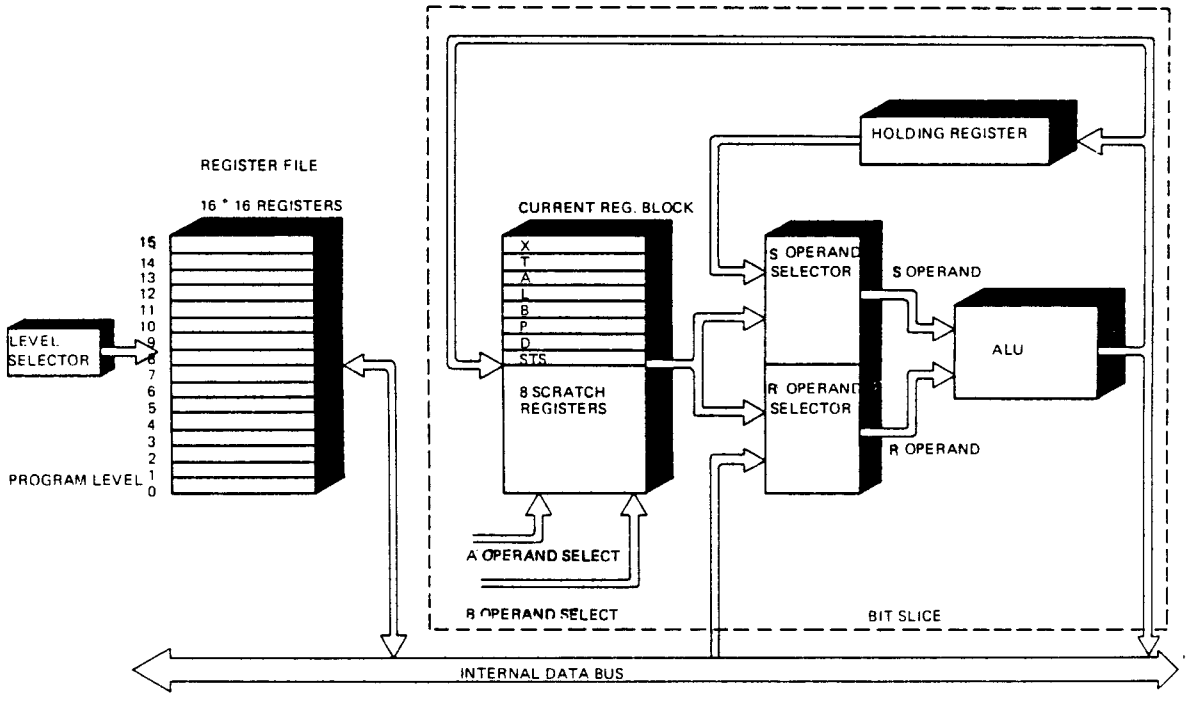


Figure 2.8.2: Register File and Main Arithmetic

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 ND-100, bit slices are driven by a set of control lines (bits) from a microinstruction. This gives the possibility of changing or modifying the microprogram while keeping the existing hardware.

The 4 ALU bit slices in the ND-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. The selection of the inputs to become 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 bit slice registers. The eight scratch registers contain temporary information managed by the microprogram, such as 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 2.8.1. In one clock cycle, two operands can be supplied to and manipulated by the ALU, with one result either placed in the holding register or in 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 of the result of the operation to be the current register block. In the current register set the register to be written into is given by the B source select, which is equal to the B register.

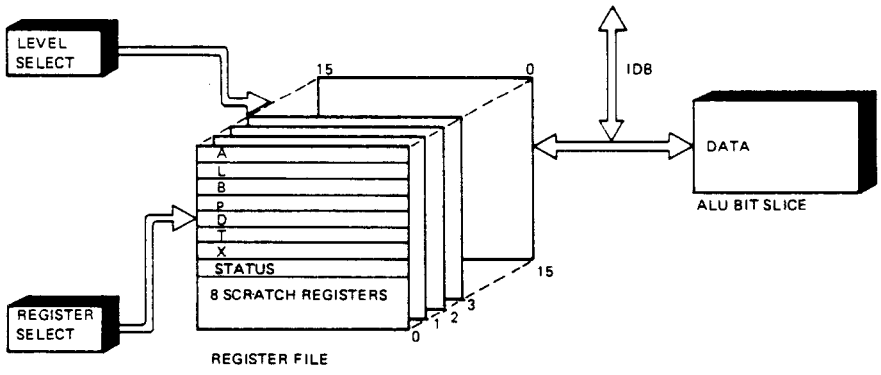


Figure 2.8.3: ALU and Register File Connection

Figure 2.8.3 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 in this way during a level change.

2.9 THE INTERRUPT SYSTEM

2.9.1 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 events, 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'.

2.9.1.1 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, ie., a great part of the CPU time is spent in the polling routines.

2.9.1.2 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 and 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 due to higher system throughput and faster response.

There are several kinds of interrupt systems:

— Single line interrupt system

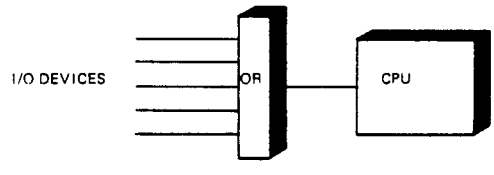


Figure 2.9.1: Single Line Interrupt System

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

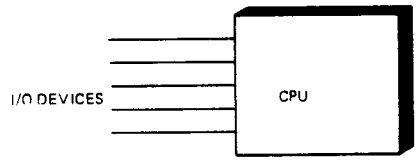


Figure 2.9.2: Multilevel Interrupt System

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

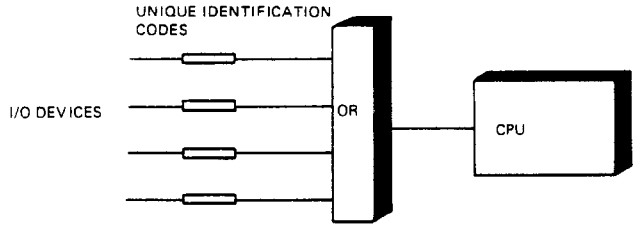


Figure 2.9.3: Vectored Interrupt System

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

2.9.2 ND-100 Interrupt System

2.9.2.1 General

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

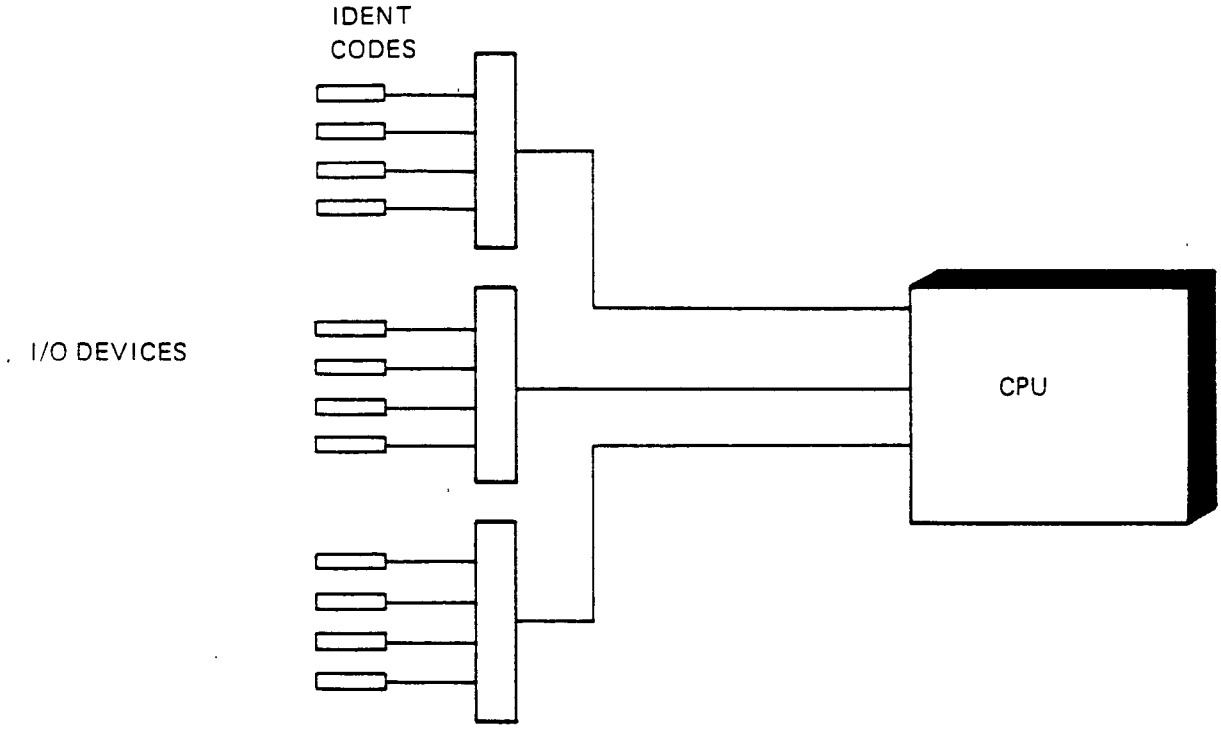


Figure 2.9.4: ND-100 Interrupt System

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 to 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 a 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

A particular interrupt request often 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 ND-100 and therefore, 16 sets of registers and status indicators. Each set consists of A, D, T, L, X and B registers, program counter and a status register with the status indicators O, Q, Z, C, M, K, PTM and TG. There are also 8 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:

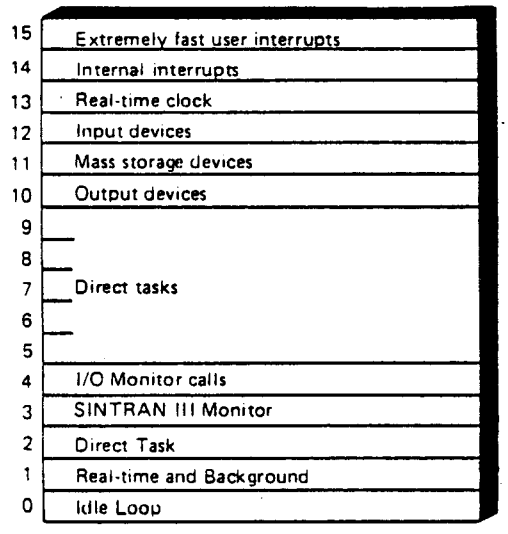


Figure 2.9.5: Level Assignments

The priority increases, program level 15 having the highest priority and 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, resulting in 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 ND 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 so is HDLC output.

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

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

A change from a lower program level to a higher 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 relinquishes its priority.

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

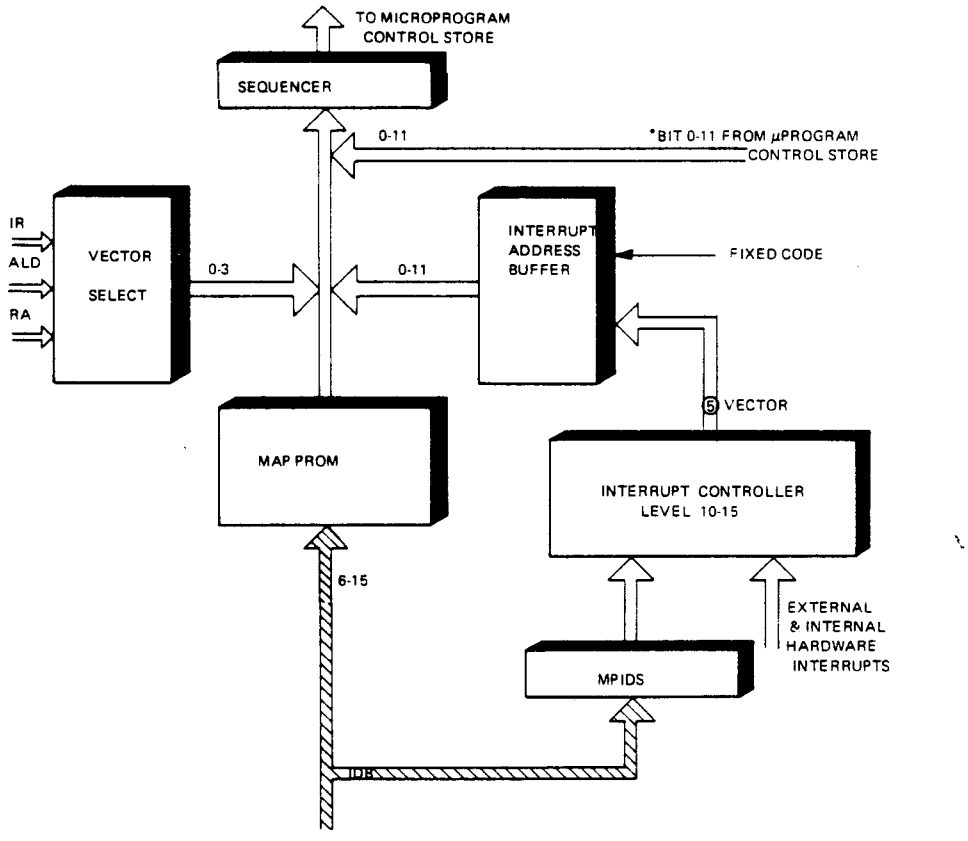


Figure 2.9.6: The Interrupt System Part of the CPU

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 such things 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, go 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 levels 10-15, including all internal interrupts.

2.9.2.2 Functional Operation

Figure 2.9.7 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, ie., 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 to 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 to 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 to service a higher priority device. Service to the lower priority device is resumed automatically upon completion of the higher level servicing.

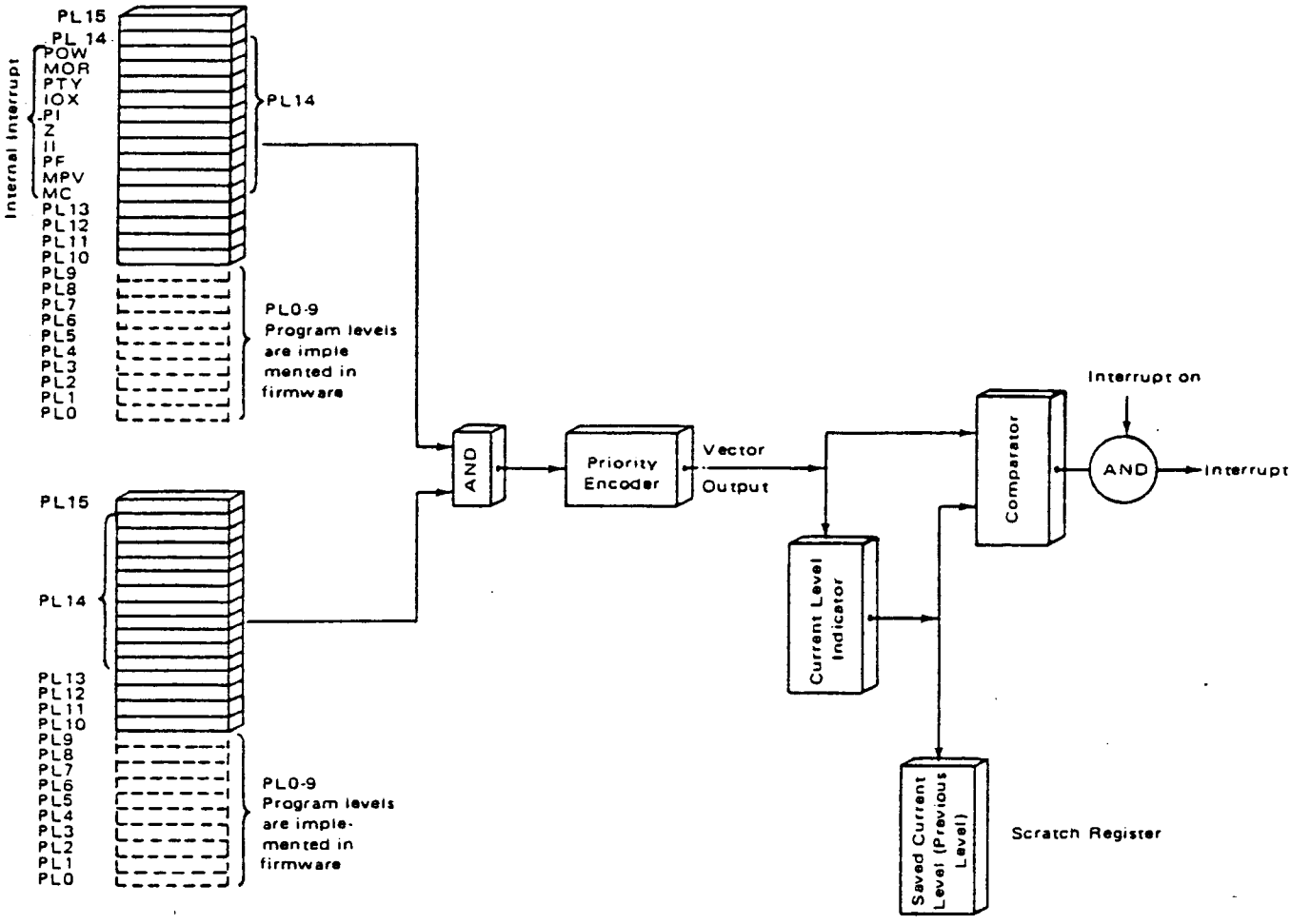


Figure 2.9.7: Priority Interrupt System

2.9.2.3 The External Interrupt System

Figure 2.9.8 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 by 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.

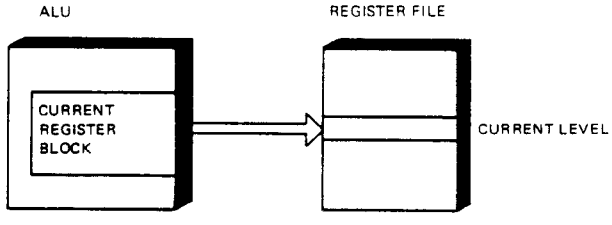
The PIL number is constantly compared to PK. PK always contains the number of the highest program level which has its corresponding bits set in both PIE and PID. Whenever PK is different from PIL, an automatic change of context block will take place through a microprogram sequence.

The CPU will not ask for the next machine instruction but enter a micro program 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.

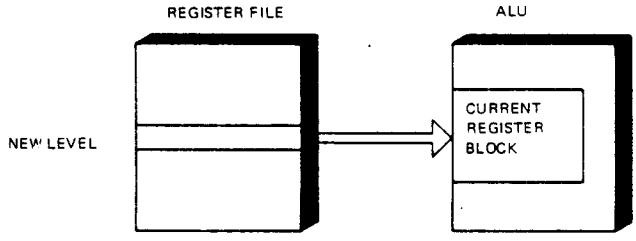
2. The working register block on the current level in the ALU is saved in the register file.



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.

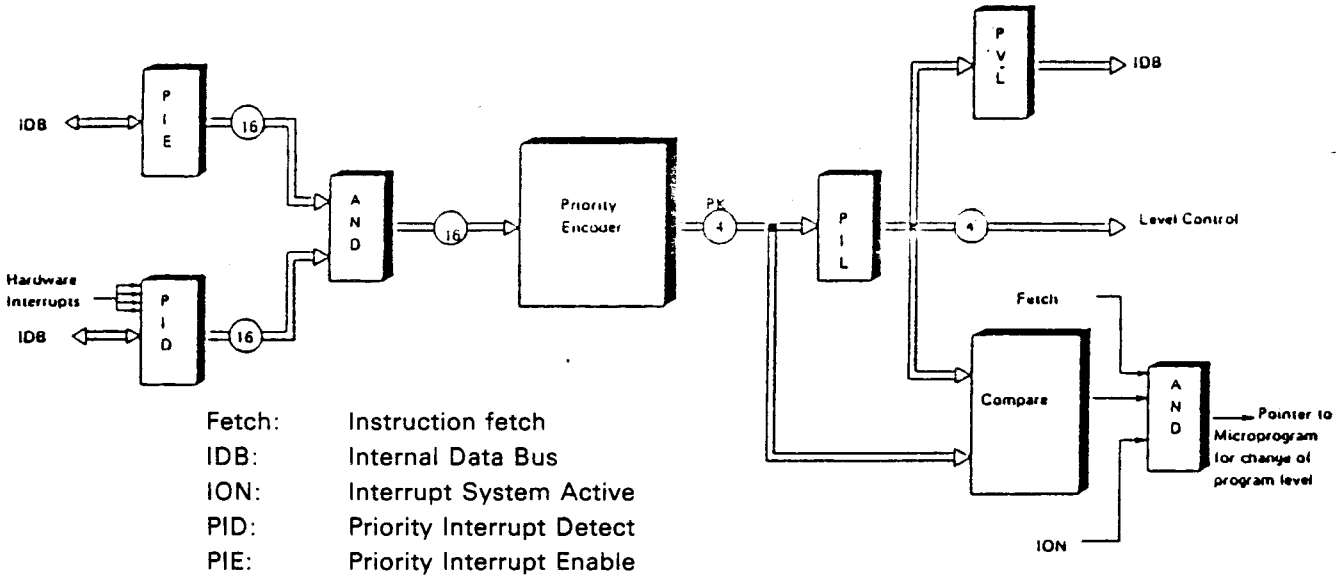
5. The working register block on the new level in the register file is moved to the current register block in the ALU.



6. A fetch is issued, ie., 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.



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

Figure 2.9.8: External Interrupt System

2.9.2.3.1 *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 ND-100 there may be up to 2048 vectored interrupts. Each physical input/output unit will usually 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 give 512 different vectors on each of the four actual levels, providing 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 the 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,      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
```

2.9.2.4 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 for the external one. Refer to figure 2.9.9.

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, ie., 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 0-12 [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.

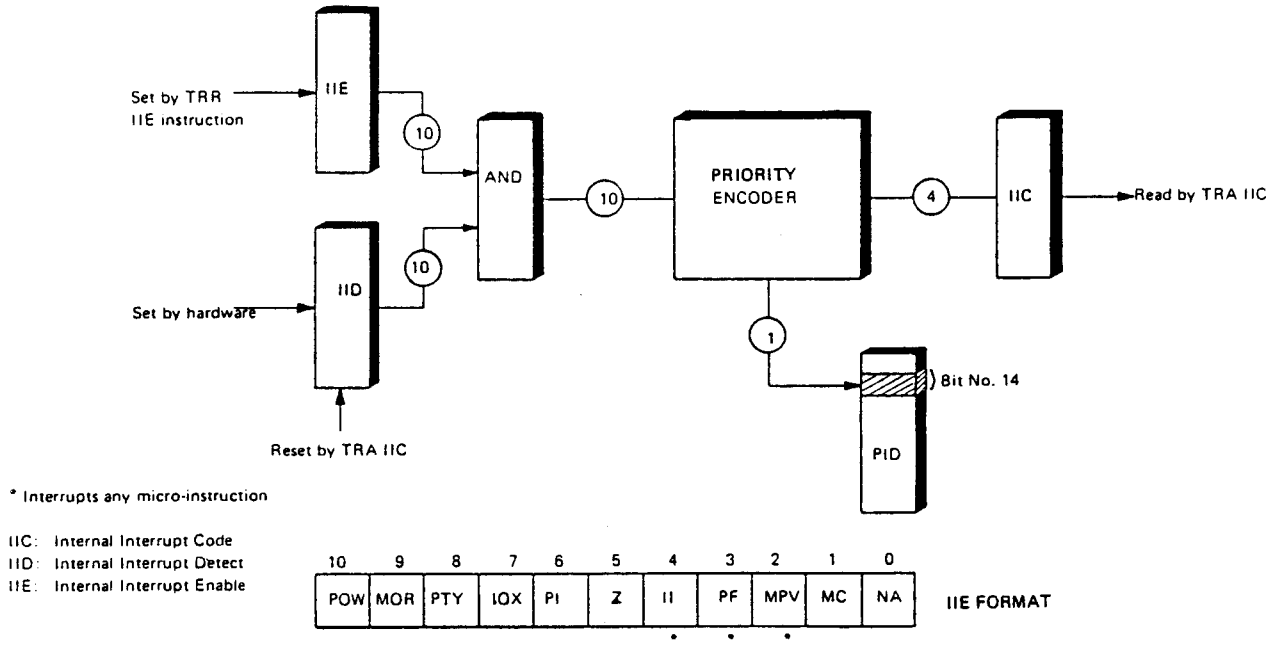


Figure 2.9.9 Internal Interrupt System, Block Diagram

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
MPV	2	2	Memory 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. Instruction not implemented.
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, ie., 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 MPV 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 no priority assigned to the different internal interrupts, as only one condition may arise at a 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.

2.9.2.4.1 *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 a 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 reentered.

Privileged Instruction Interrupt

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

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

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 to 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.)

2.9.2.4.2. *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.

2.9.2.5 Programming Control of the Interrupt System

2.9.2.5.1 *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 the 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 lit by the ION. The ION instruction is privileged.

IOF — Interrupt System off

Format: IOF

The IOF instruction turns off the interrupt system, ie., the mechanisms for a change in program levels are disabled. The computer will continue operation at the program level at which the IOF instruction was executed, ie., 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.

2.9.2.5.2 *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 to set 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. These are 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), 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.

2.9.2.5.3 *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. This will then normally 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.

2.9.2.5.4 *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 (ie., the operating system) wants to know which level was the last, and the value of the program counter on that interrupting level. This may be useful for a number of purposes. It 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 microprogram 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). The A register will now 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

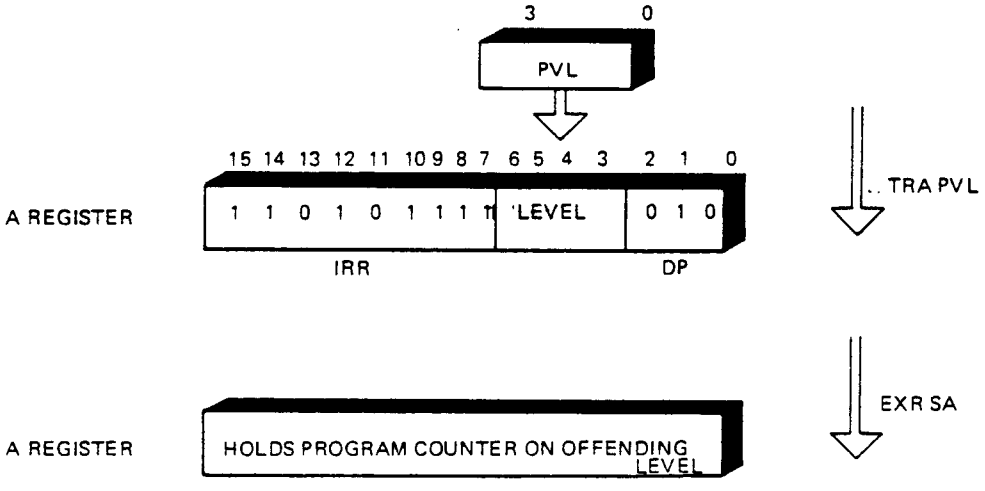


Figure 2.9.10: TRA PVL Instruction Execution

By executing an EXR SA (execute register) at this point, the contents 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).

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

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

3. The saved program counters, on the level to be used, must be initialized, ie., 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 PEA, 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
```

2.10 THE ADDRESS ARITHMETIC

2.10.1 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 microprogram.

Three types of information are stored in memory:

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

Memory, however, regards all of them as information. The CPU must therefore, 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:

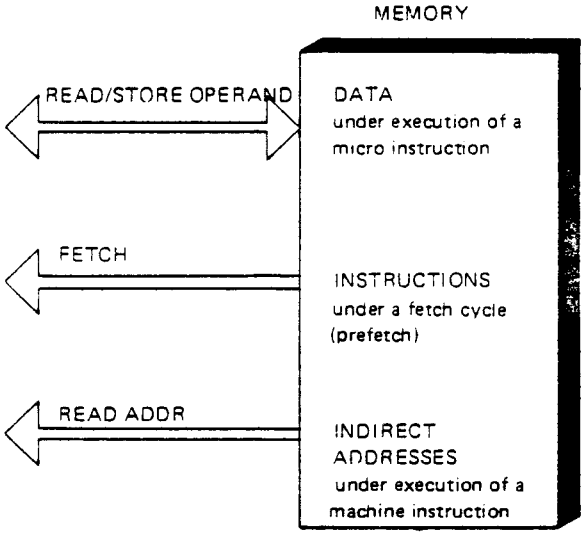


Figure 2.10.1: Different Memory Accesses

2.10.2 Addressing Structure

Memory reference instructions specify operations on words in memory. For all the memory reference instructions in ND-100, the addressing mode is the same with the exception of the conditional jump, the byte, 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 the conditional jump, the byte and the register block instructions are excluded from this.)

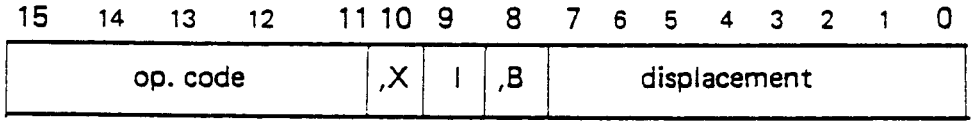


Figure 2.10.2: Format of the Memory Reference Instructions

ND-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 backward 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

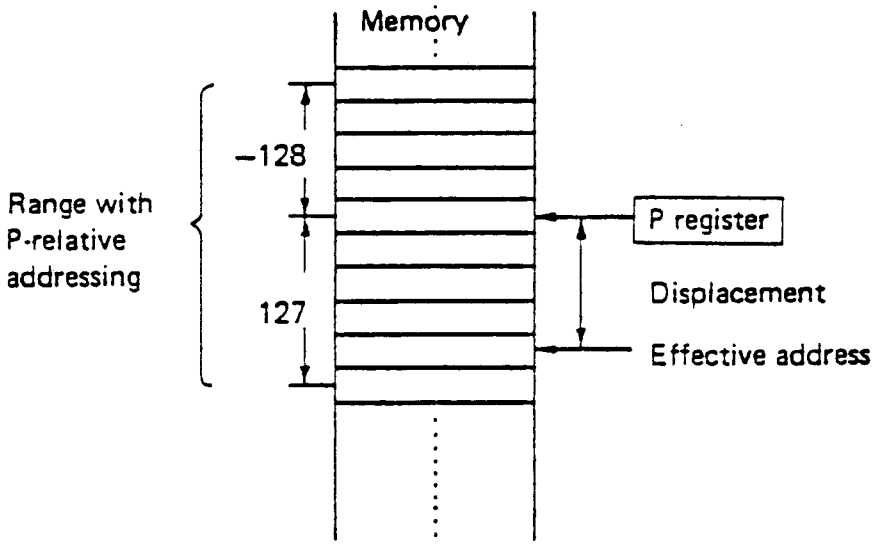


Figure 2.10.3: Schematic Illustration of P Relative Addressing

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 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. However 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

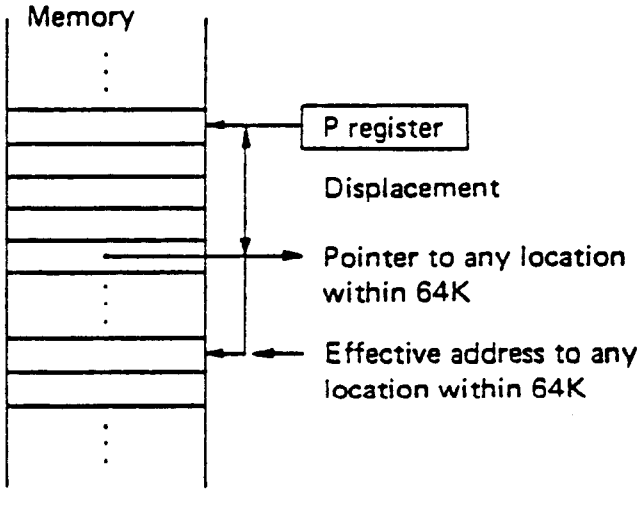


Figure 2.10.4: Schematic Illustration of Indirect P Relative Addressing

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

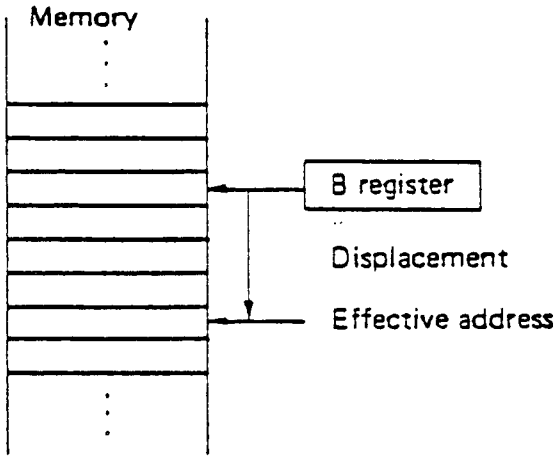


Figure 2.10.5: Schematic Illustration of B Relative Addressing

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

This mode has the same relationship to B relative addressing as 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 to call library routines.

Example:

STA I +2 ,B

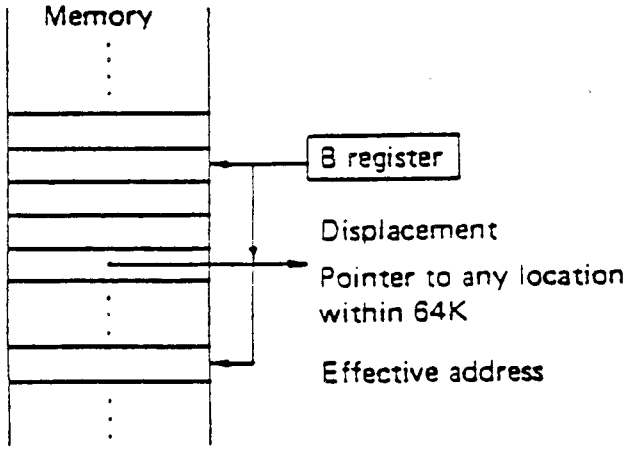


Figure 2.10.6: Schematic Illustration of Indirect B Relative Addressing

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 to access the elements of a block of data.

Example:

STA +2 ,X

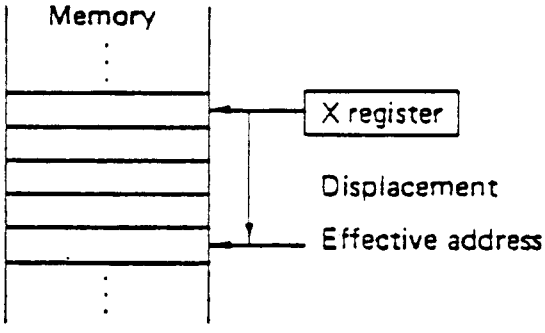


Figure 2.10.7: Schematic Illustration of X Relative Addressing

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 up 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

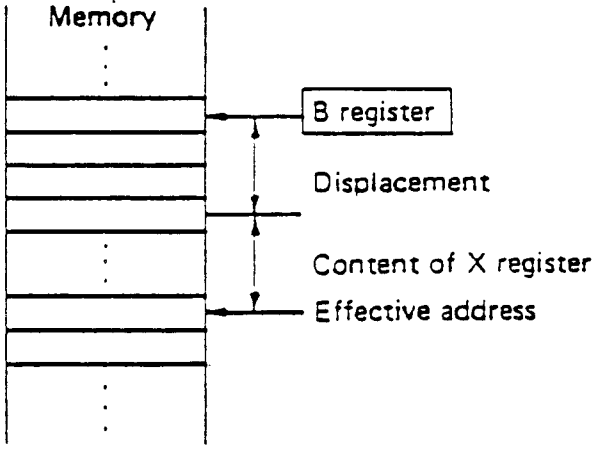


Figure 2.10.8: Schematic Illustration of B Relative Indexed Addressing

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

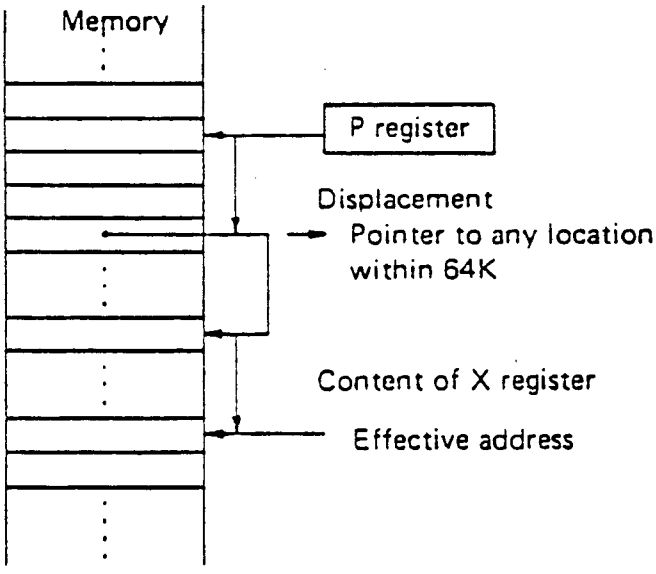


Figure 2.10.9: Schematic Illustration of Indirect P Relative Indexed Addressing

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 are used in the effective address computation instead of the contents of the P register. 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

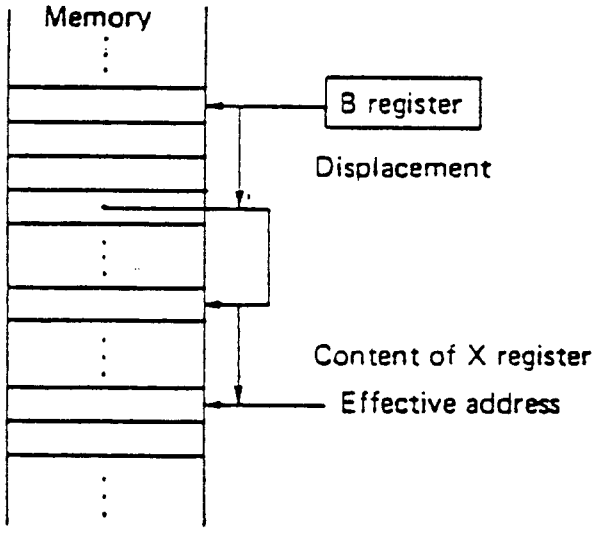


Figure 2.10.10: Schematic Illustration of Indirect B Relative Indexed Addressing

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

2.10.3 Principles of Address Arithmetic

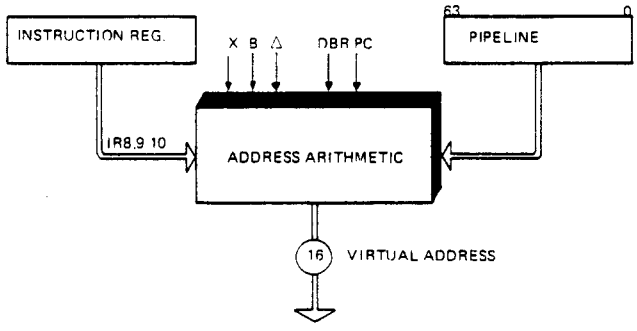


Figure 2.10.11: Address Arithmetic — Input and Control

Data input to the address arithmetic is:

— the X register
— the B register
— the program counter (PC)
— DBR (data bus read register) used for indirect addressing
— the displacement ( ) of the instruction, taken from GPR

Sign extension is performed by setting bits 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.

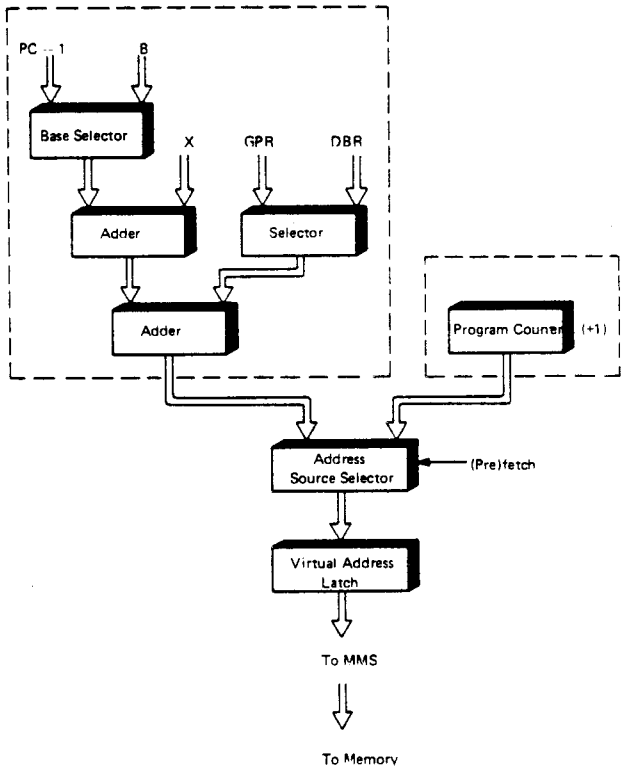


Figure 2.10.12: Address Arithmetic — Functional Operation

Figure 2.10.12 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 by 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 microprogram.

Let us assume that the last microinstruction in a given machine instruction is being processed, and a fetch 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 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 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 , 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, a prefetch cycle will also take place when the execution of this instruction starts.

First Memory Cycle:

To find the indirect address, the B register is added to the displacement (pre-indexing) found in GPR, and goes 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 contents of the X register (post-indexing).

3 THE MEMORY MANAGEMENT SYSTEM

3.1 GENERAL

The hardware memory management module is needed to run 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 the storage.

In order to implement virtual storage, an intelligent addressing translation mechanism must be employed. This mechanism is under the 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.

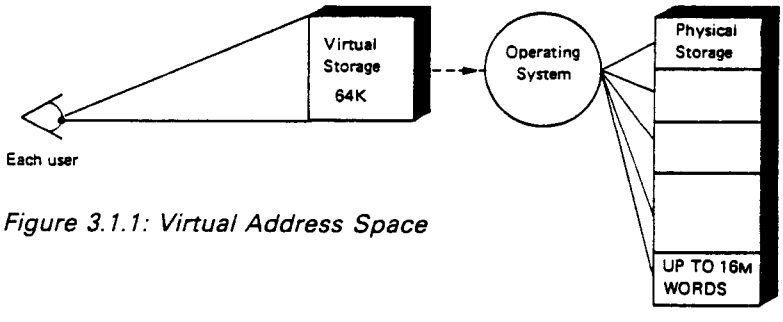


Figure 3.1.1: Virtual Address Space

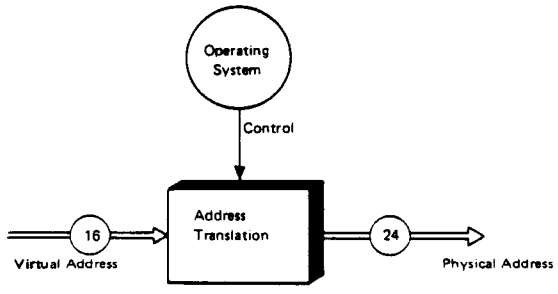


Figure 3.1.2: Virtual to Physical Address Translation

In ND-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.

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 location 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 programs will occur. Programs are broken up in physical storage, and loaded where vacant pages are found.

More parallelism

Also due to the paging system, current parts of a given program only reside momentarily 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.

3.2 IMPLEMENTATION

3.2.1 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.

3.2.2 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 3.2.1 shows the CPU memory management module connection.

In addition to the memory management system, the memory management module contains 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 ND-100 bus, for physical address generation and to receive data to the cache memory.

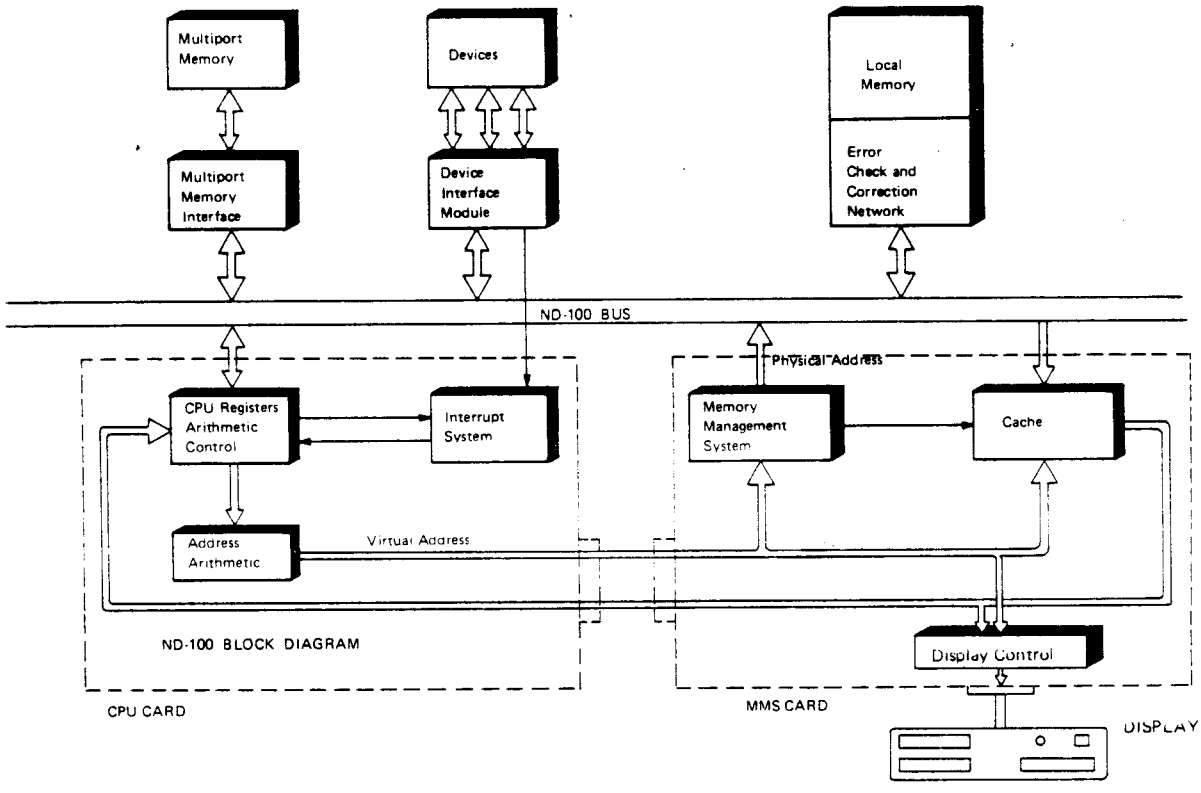
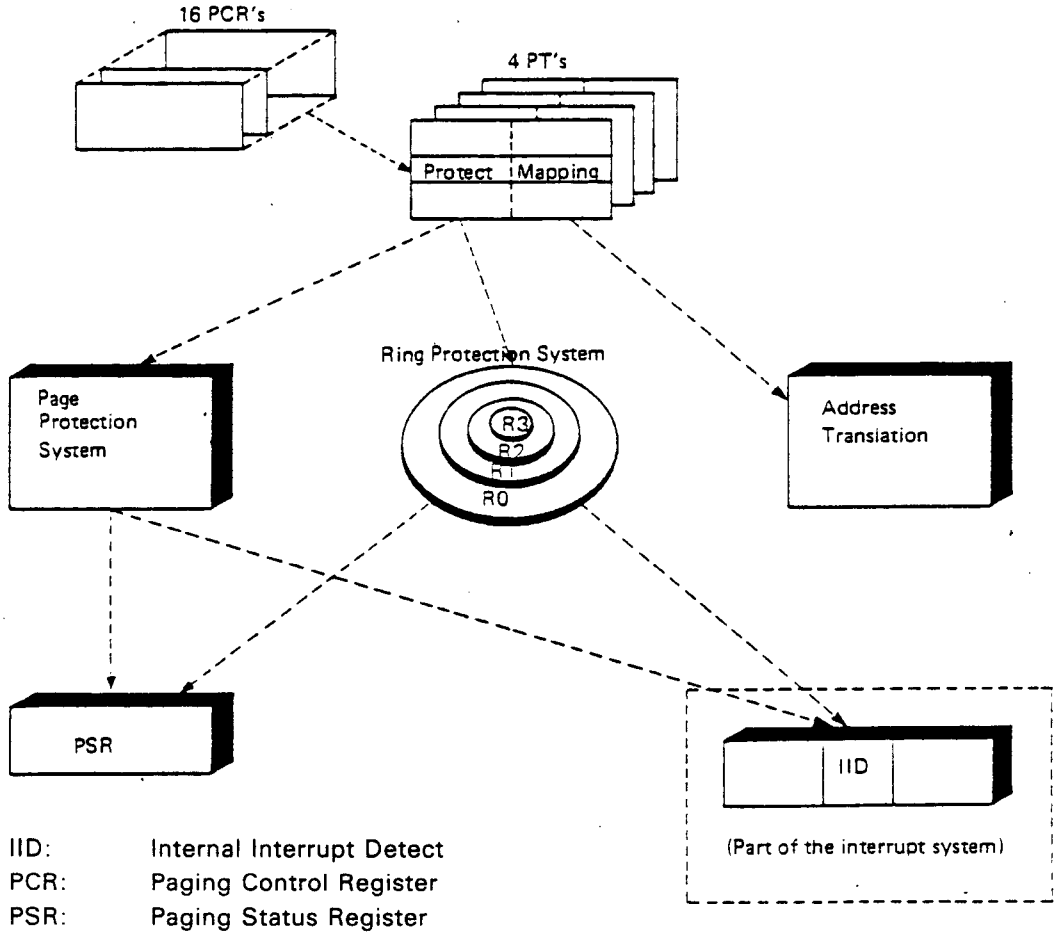


Figure 3.2.1: CPU — MMS Module Connection

3.2.3 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 3.2.2: Memory Management Building Blocks

3.3 ADDRESS TRANSLATION

3.3.1 Virtual to Physical Address Mapping

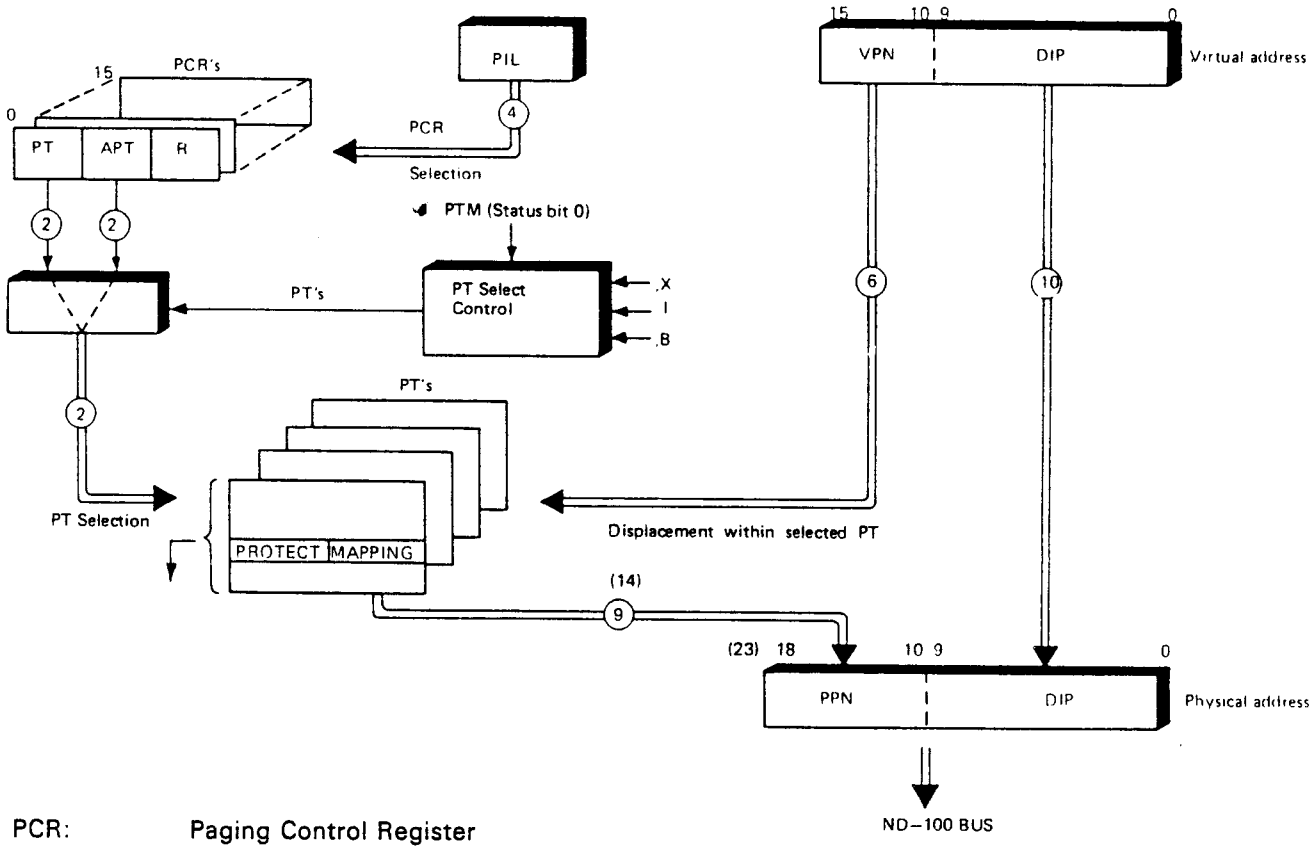
Refer to figure 3.3.1.

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 whether the normal (19 bits) or the extended (24 bits) addressing mode is used. In the following explanations, the extended mode is used. The extended 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 the 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 ND-100 bus. The most significant part of the virtual address (bits 10-15) is 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 is to be used from the selected PCR is determined from the page table modus given status bit 0. The PCR determines which of the 4 page tables to select, and VPN addresses an entry in the selected PT.



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

Figure 3.3.1: 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 described later. Fourteen bits (or 9) from the mapping table, called physical page number (PPN) are transferred to the ND-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 tables in the PT. The address translation is therefore under the control of the operating system.

3.3.2 Page Table Selection

ND-100 has 4 page tables. The one to be used is selected by the paging control register on the current program level. In PCR, the information is either taken from the PT field or from 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 provides an explanation.

Note that indirect addressing involves 2 memory references, where one may go via the PT and the other 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 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 64 K. Normally, however, PT and APT will both point to the same page table, so that the accesses via PT and those via APT will give the same result.

3.3.3 Page Table Assignment

Figure 3.3.2 shows page table assignment under SINTRAN III.

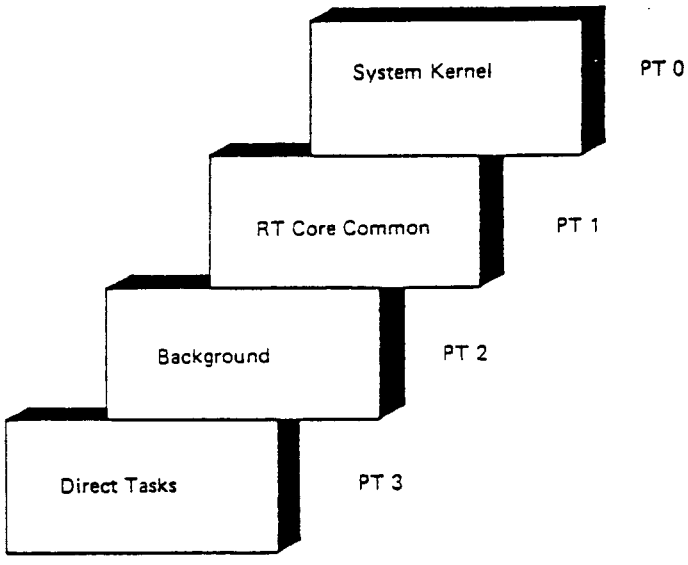


Figure 3.3.2: Page Table Assignments

Page table 0:

Contains physical page numbers which point to the core resident part of the operating system (SINTRAN III), and to 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 current programs using the different pages. The unused table entries contain zero.

3.4 MEMORY PROTECTION SYSTEM

3.4.1 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, ie., complete protection of system from user and of 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, and that both the individual memory protection mode and the ring mode must be satisfied before an operation is performed.

3.4.2 Layout of Page Tables

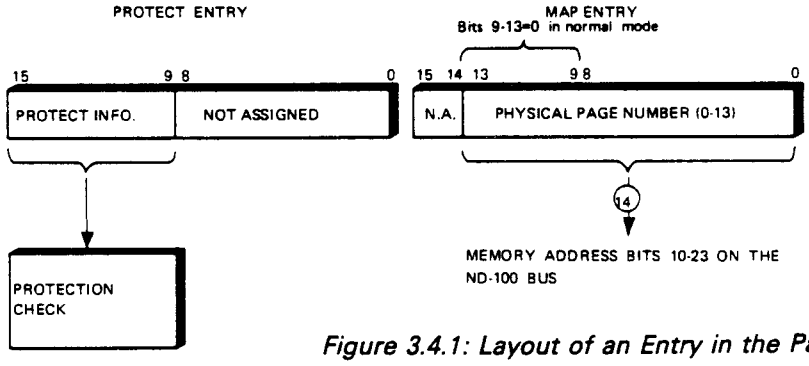


Figure 3.4.1: Layout of an Entry in the Page Table

In the following, it is of vital importance to separate the aspect of the page tables as seen from program (as shadow memory) from how things work physically during the paging process. The paging process described in this section is identical in normal mode and in extended mode. Chapter 3.6 describes the page tables operated as shadow memory, where the handling is very different in normal and in extended mode.

The paging process begins with lookup of an entry selected by the 6 bit VPN (see figure 3.3.1). This entry is 32 bits long for both modes and consists of two parts, *protect* and *map* as shown in figure 3.4.1.

Thus, each 1 K 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 straightforward. It simply tells where in memory the page is placed, ie., the physical page address. Since the address *within* a 1 K page is unaltered (DIP in figure 3.3.1), 24 — 10 = 14 additional bits are needed 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 3.4.1).

3.4.3 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, ie., if it is load (read), store (write), instruction fetch or indirect address.

The three bits from the protect table have the following significance:

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.

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.

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

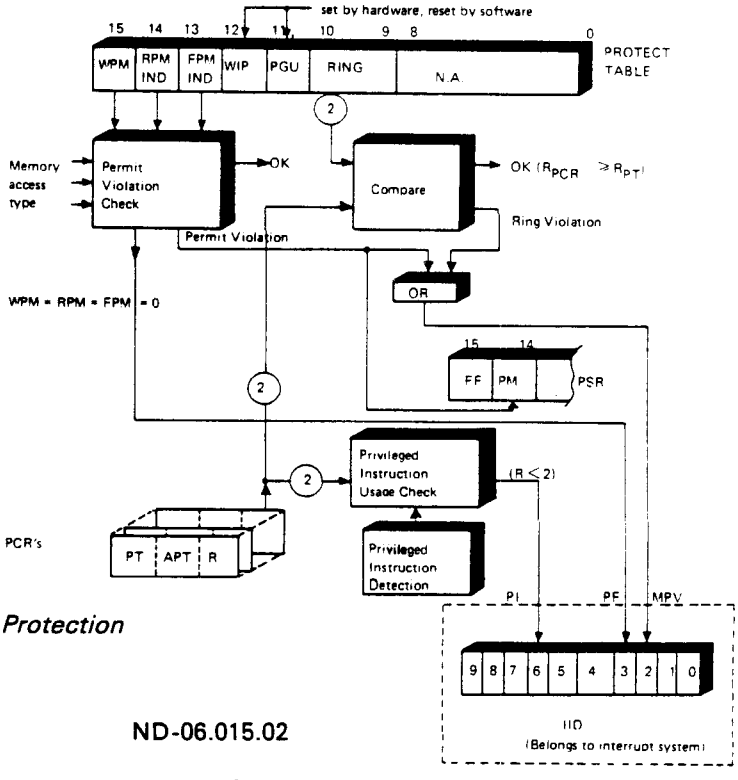


Figure 3.4.2: Memory Protection

3.4.4 Ring Protection System

3.4.4.1 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.*

3.4.4.2 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' who are often abbreviated 'user'. Hence, *user* is the 'manager' of programs.

3.4.4.3 User Ring

The *user* may belong to one of four *rings* depending on whether he is privileged 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.

3.4.4.4 Program

A program is the tool a user has at his/her disposal to perform a specific task. It consists of instruction code and pure data. In this context a *program* is executed by a *user*. The *program* itself cannot execute code, because it *is* code.

3.4.4.5 Program Ring

A program may be good or bad depending on how much harm the user may cause by executing it. To protect delicate yet powerful programs from potentially 'dangerous' 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).

3.4.4.6 Ring Usage

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

The privileges of the four user rings are defined by the paging control register bits 0-1:

PCR bits 1 0

0 0 Ring 0:
Users of this ring may not execute privileged 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 have top quality programs at their disposal, since even a small slip may destroy the system. A ring 0 user can only create problems on his own accessible programs. Other users are not affected by them.

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 bad 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 ring 3 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 takes place when lower ring *instruction codes are executed,* but not when pure data is accessed.

3.4.4.7 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.

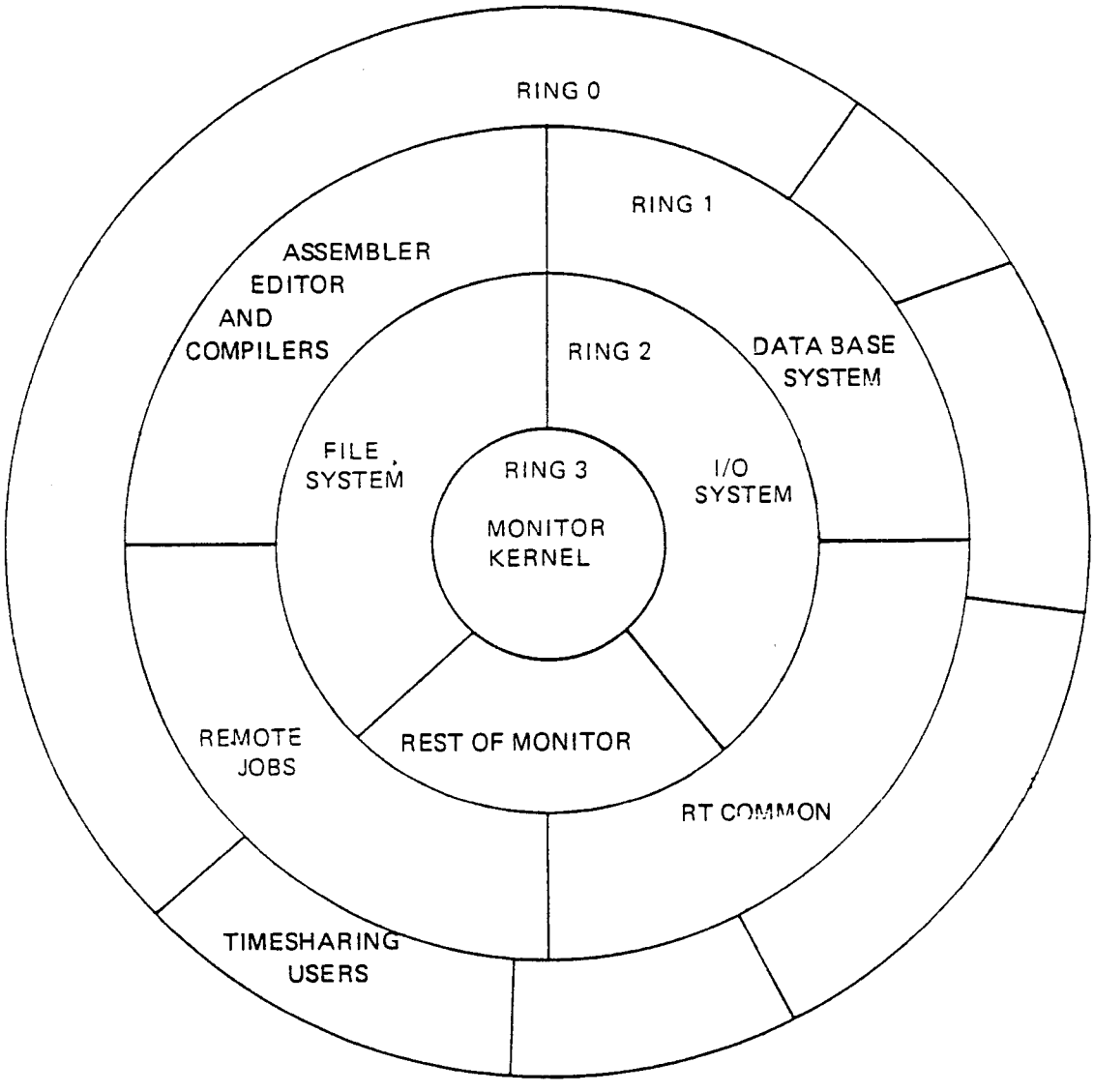


Figure 3.4.3: Ring Assignment

3.4.5 Privileged Instructions

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

3.4.6 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 needs no writing back. This bit is automatically set to one the first time a write occurs, and then remains set. It is cleared by program (whenever a new page is brought from mass storage).

Bit 11: PGU — Page Used

If PGU = 1, the page has been used. The bit is automatically set whenever the page is accessed, and it remains set. The bit is cleared by program. This bit may be used by the operating system to maintain a record of the access frequency of a page. This is used in decisions making the replacement algorithm, ie., to determine which page should be swapped.

3.5 MEMORY MANAGEMENT CONTROL AND STATUS

3.5.1 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 64 K, the address following the POF instruction.

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

3.5.2 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.

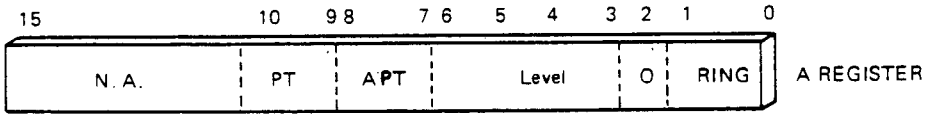
Bit number 13 in the status register is reset, indicating normal address mode.

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

3.5.3 Paging Control Register

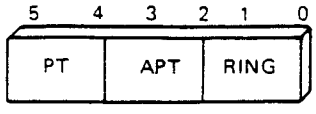
There is one PCR (paging control register) for each level. The setting of the PCRs is done by the operating system prior to the program execution. One PCR may be written into at a time, by the instruction TRR PCR.

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



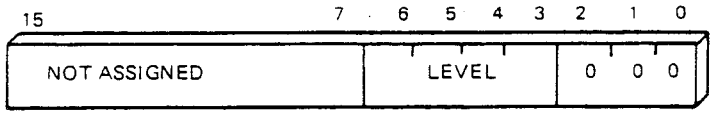
Bits 11-15:	Not assigned
Bits 9-10:	Page table number (0-3)
Bits 7-8:	Alternative page table number (0-3)
Bits 3-6:	Program level (PCR number) (0-15)
Bit 2:	Equals zero
Bits 0-1:	Ring number (0-3)

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

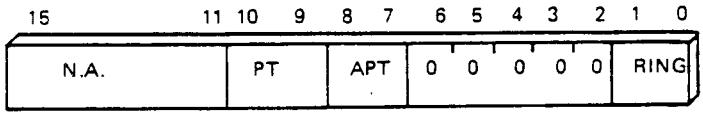


PCR FORMAT

For maintenance purposes, it may be desirable to read back the contents of the 16 PCRs. This is accomplished by the TRA PCR instruction. Before execution the contents of the A register must be:



After execution the contents of the A register will be:



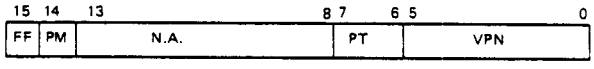
3.5.4 Paging Status Register

Whenever the memory management system reports errors (page fault, memory protection violations), the operating system is alerted through an internal interrupt with the interrupt code equal to the error source. The operating system then reads 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 significance:

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

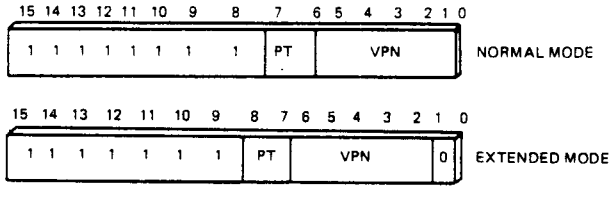
Bits 0-5:

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 3.6).

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

Figure 3.5.1: 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 one 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.

3.6 CONTROL OF PAGE TABLES

3.6.1 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). Pages are taken from other processes if necessary. According to the program properties the protect part is also filled.

As mentioned in section 3.4.2, one must separate the appearance of the page tables as seen from program (shadow memory) from how things work during the paging process. During paging the process is essentially the same whether in normal or in extended mode. There are 4 subtables, PT 0-3, each consisting of 64 entries (see figure 3.6.1).

3.6.2 Shadow Memory

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

To ease reading and writing of the page tables, these 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 thus 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 3.6.1 shows the shadow memory addressing.

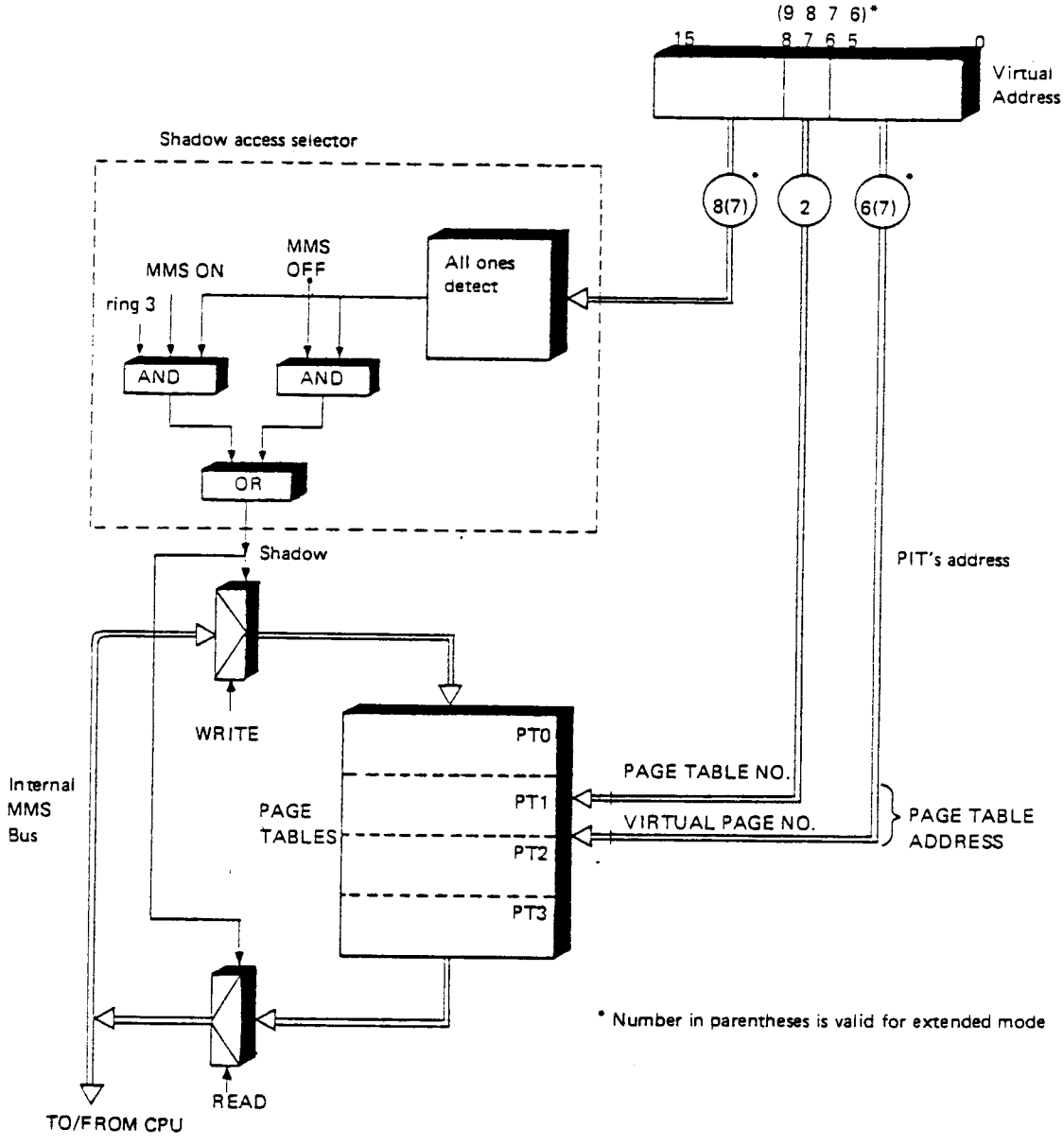


Figure 3.6.1: Shadow Memory Access

3.6.3 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 3.6.2 and 3.6.3. For the sake of simplicity only page table 3 is shown. However, 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 contents in the tables without interpreting it as paging information. When paging is off, the page tables may be used as ± K very fast random access memory.

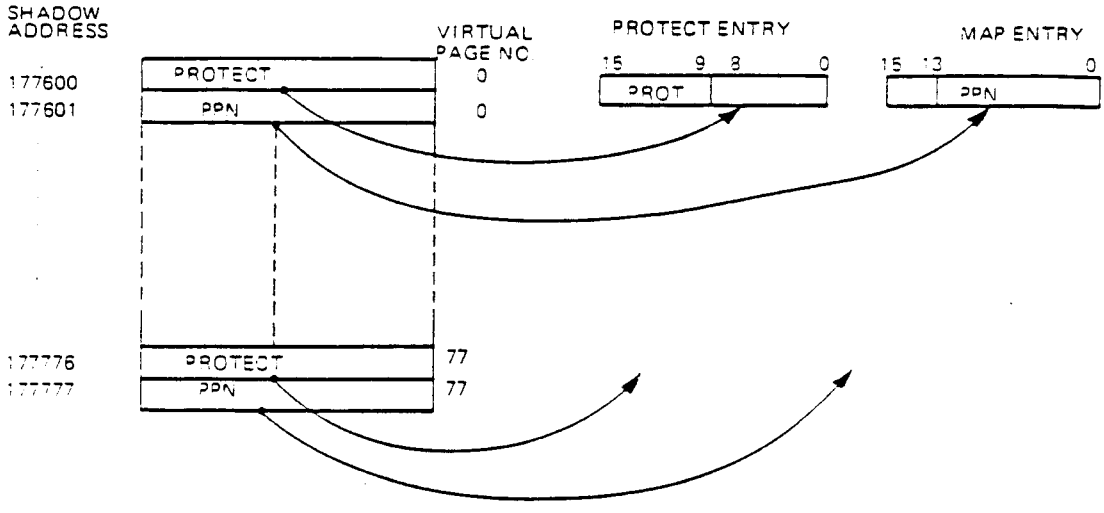


Figure 3.6.2: Reading and Writing Page Table 3 Entries as seen from Program in Extended Mode

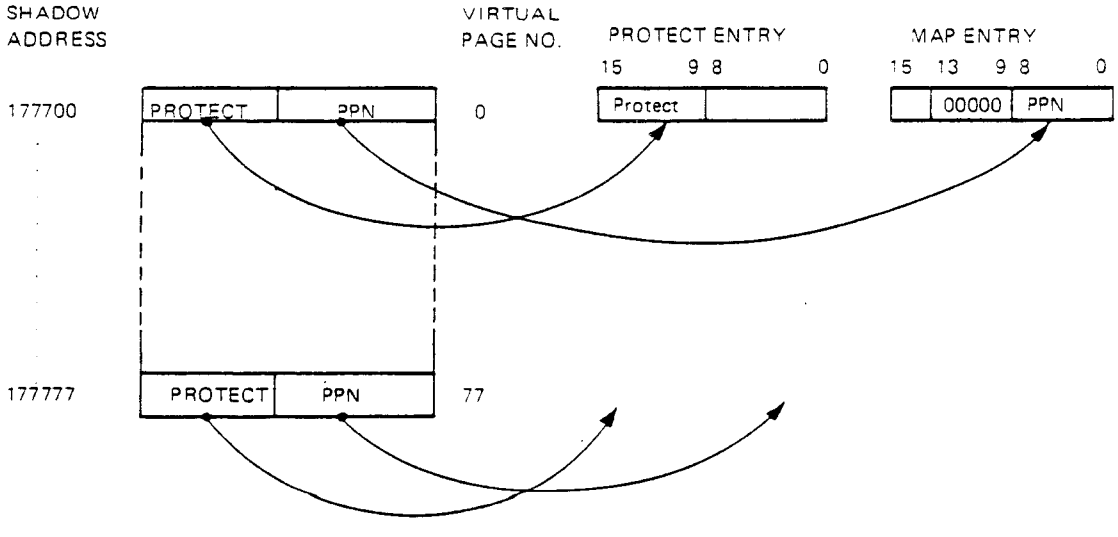


Figure 3.6.3: Reading and Writing Page Table 3 Entries as seen from Program in Normal Mode

3.7 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 contents are in cache. However, if there is no hit in cache, the paging system will introduce 50 ns overhead.

3.8 EXAMPLES

Example 1:

Assume that a user has a 3 K program. The start address is 40000_8, ie., the address space is 40000_8 — 45777_8 since 3 K = 3072_{10} = 6000_8. Refer to figure 3.8.1.

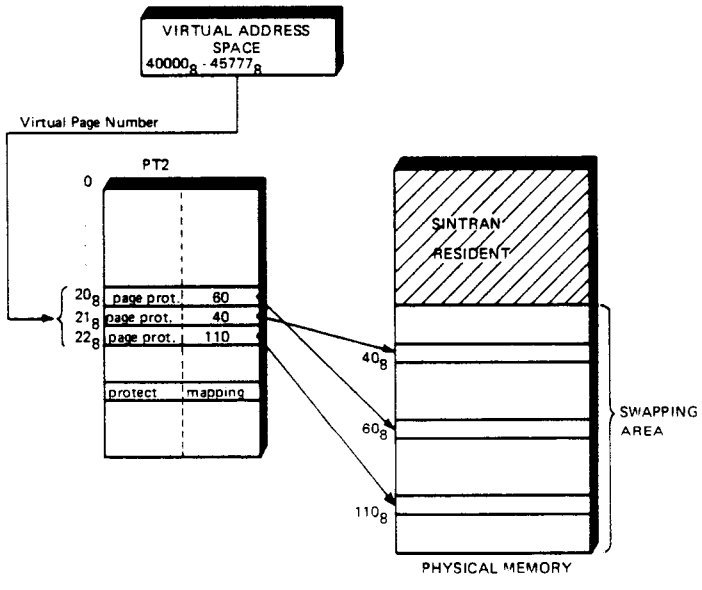


Figure 3.8.1: Virtual to Physical Address Conversion

From figure 3.8.1, it can be seen that the logical or virtual address space of 40000_8 — 45777_8 will always 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 3.8.2 shows the page table selection if the following conditions are fulfilled:

— Status register bit 0 (PTM — page table mode) 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 possibly by a compiler). The data accesses will then use the APT (refer to the table in section 3.3.2).

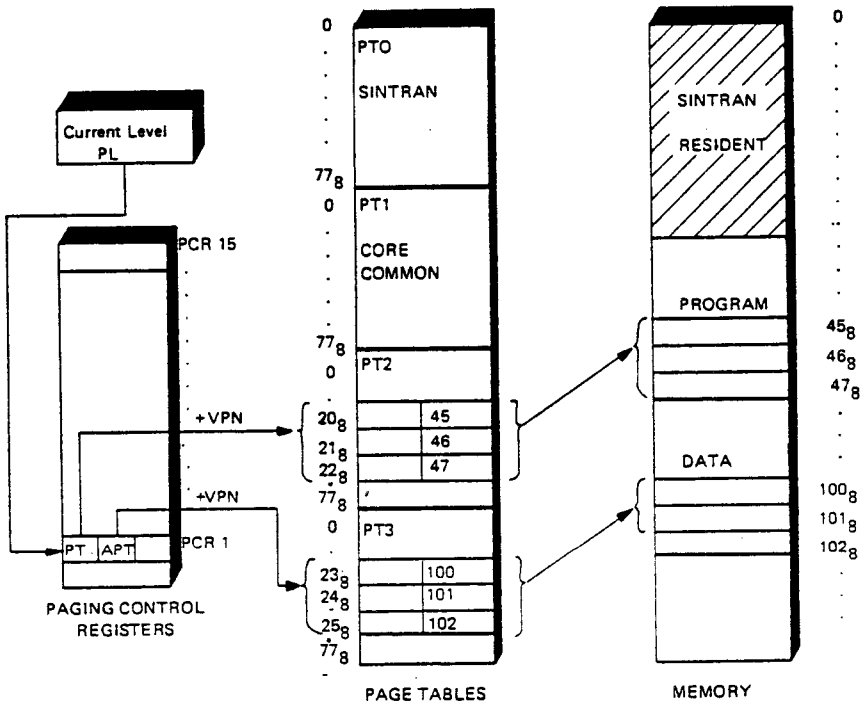


Figure 3.8.2: PCR and PT Usage

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 3.8.3 shows the same situation as in figure 3.8.2 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 (ie., all program systems which run on a lower ring than the operating system).

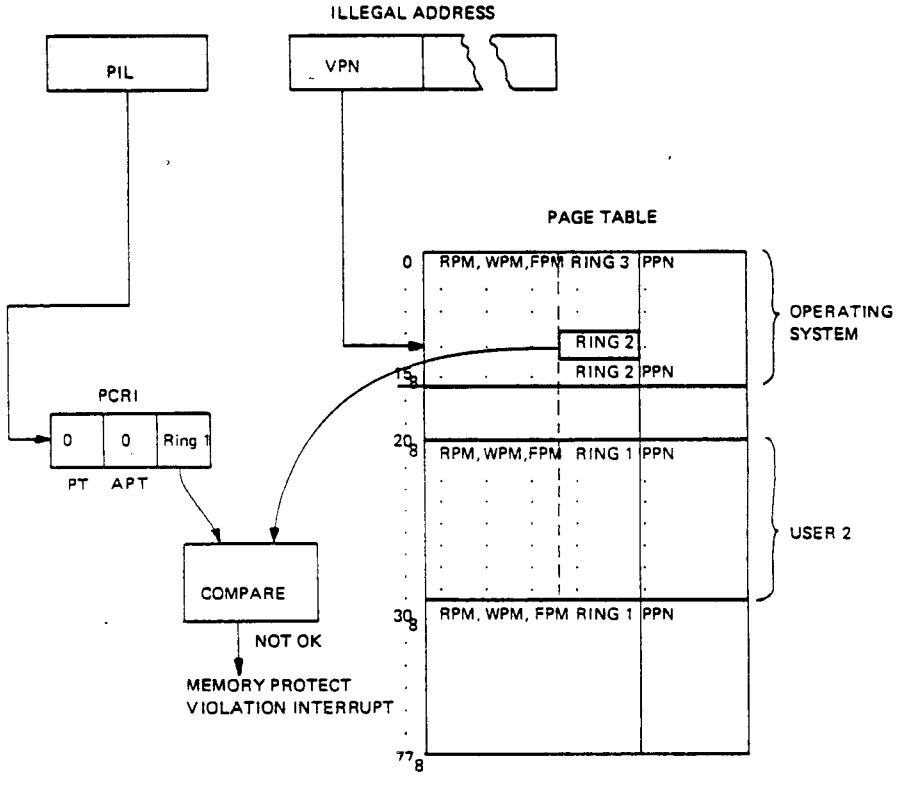


Figure 3.8.3: Page Table Usage, Example

4 ND-100 BUS SYSTEM

4.1 GENERAL

All system components and peripherals are connected to the high-speed ND-100 bus. The ND-100 bus provides the communication path through its bidirectional lines for addresses, data and control lines for devices in the bus. There is an exception, which is communication directly between the CPU and the MMS and cache. Figure 4.1.1 shows the interconnections between the modules. The bus is general purpose, thus all system elements communicate with each other in an identical fashion over this bus, allowing all modules of the ND-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 buses gives a more reliable system

— no overhead in connecting several buses between source and destination makes a faster system

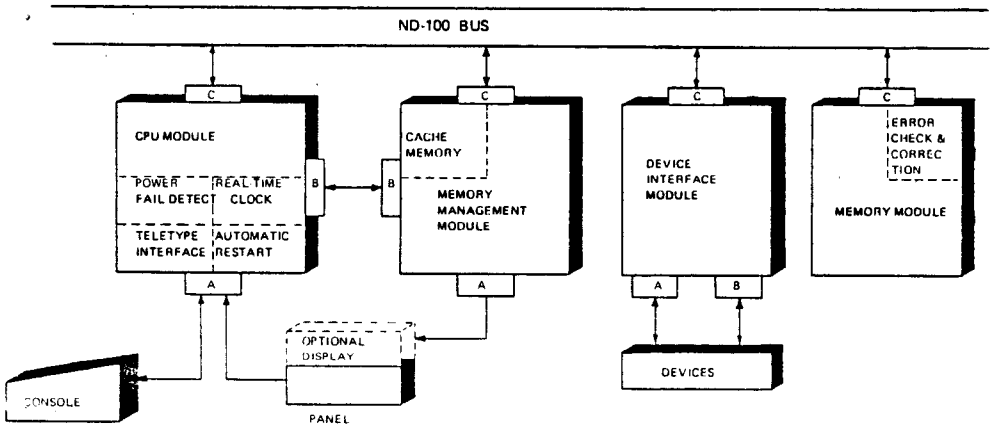


Figure 4.1.1: Interconnection between Modules

4.2 BUS CONTROL

The ND-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:

— allocation of the ND-100 bus to one of the possible requesting bus users

— supervising that the ND-100 bus is released by the authorized user within a certain time limit

Before the ND-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 ND-100 bus, a bus allocation request is sent to the bus control.

The bus control controls the allocation of the ND-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)

If more than one source requests the bus at the same time, the bus control gives the priority. DMA-requests and Refresh-requests are both external requests which have a separate internal priority; Refresh is always served first if they occur at the same time.

If the external request (Refresh or DMA) and the CPU-request are present at the same time, priority is given to the one which does not have the previous cycle (toggled priority).

4.3 PHYSICAL ARRANGEMENT OF THE ND-100 BUS

Physically, the ND-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.

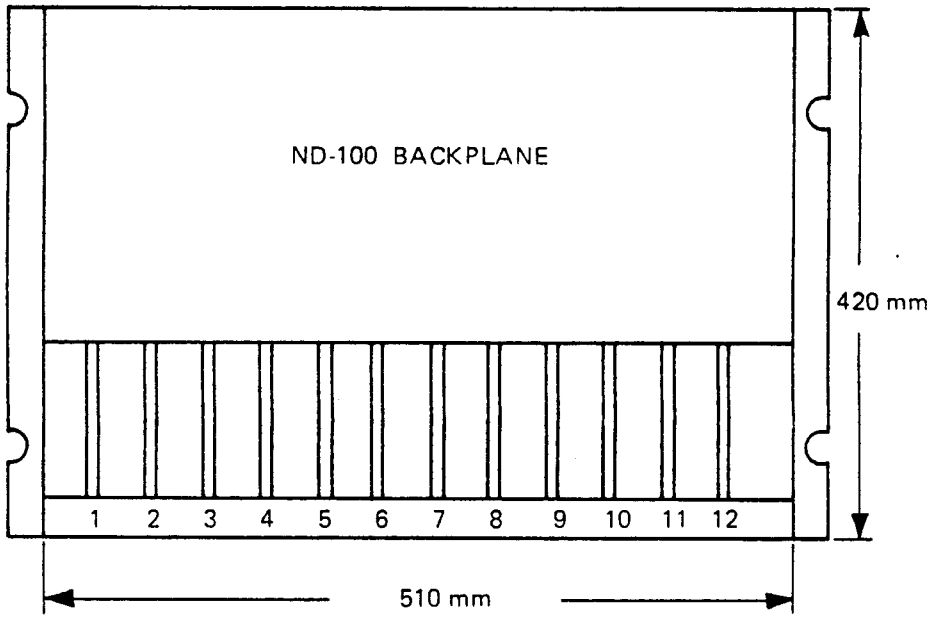


Figure 4.3.1: ND-100 Crate Layout (Top View)

The ND-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, ie., 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).

4.4 ORGANIZATION OF ND-100 MODULES

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

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

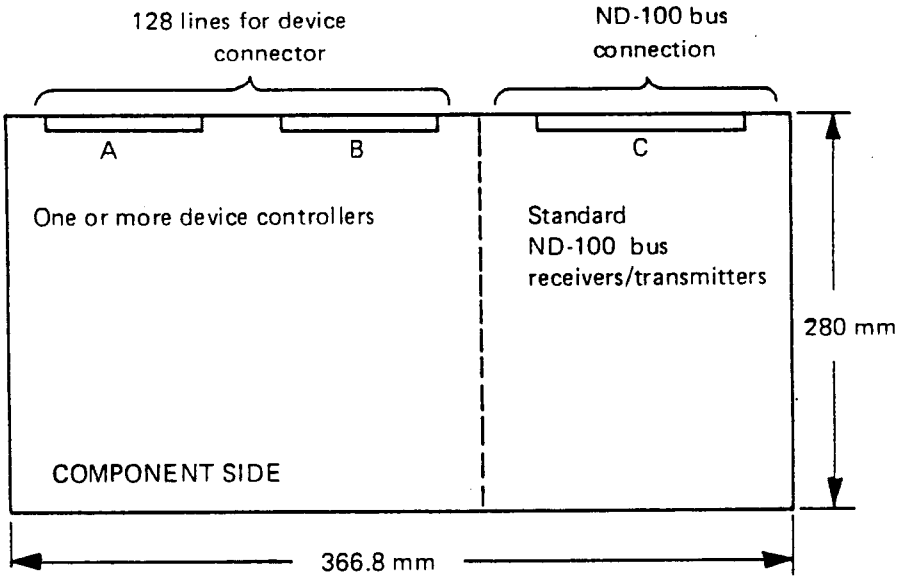


Figure 4.4.1 shows a standard ND-100 I/O module and the connector usage.

Figure 4.4.1: ND-100 Module and Connectors

The plugs are assigned an identification letter as illustrated. The plug used for connection to the ND-100 bus is assigned the letter C. This plug contains 96 pins, each of them defined in accordance with the ND-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 4.4.2 below shows the card crate and the usual placement of modules in a medium sized ND-100 system.

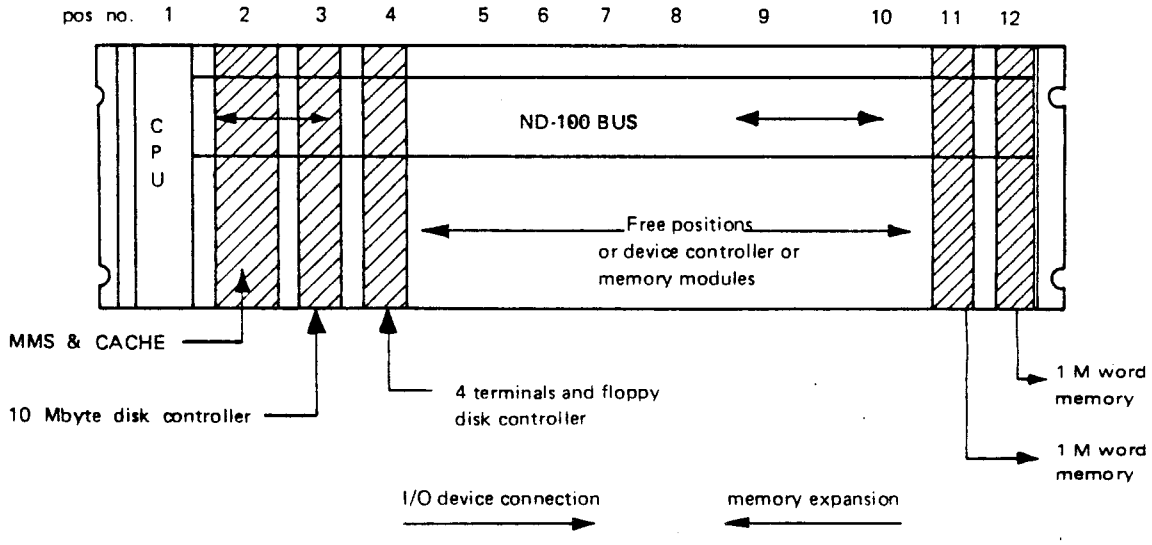


Figure 4.4.2: The Card Crate and Usual Placement of Modules

If the memory management system and the 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 ND-100 bus with 12 new positions, with the help of a bus extender card.

4.5 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 was 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 4.5.1 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.

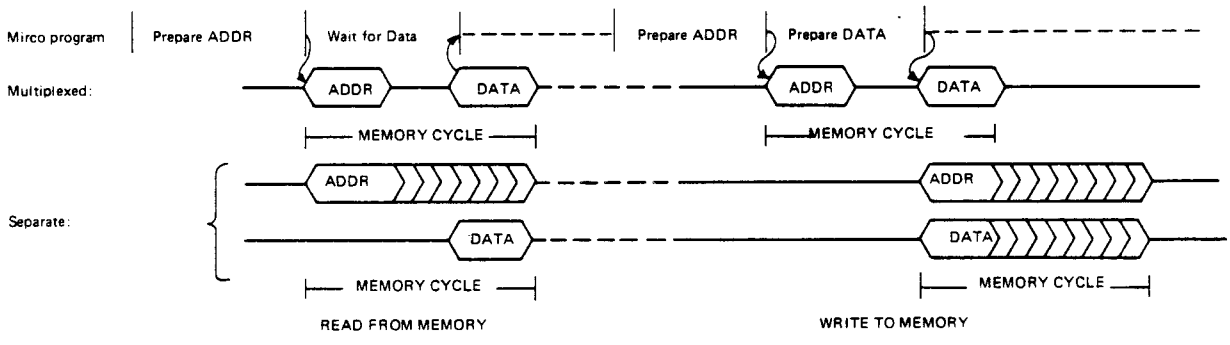


Figure 4.5.1: Comparison of Buses with Multiplexed and Separate Address and Data

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 for typically 550 ns.

5 THE ND-100 STORAGE SYSTEM

5.1 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, ie., it does not discriminate any given type of information.

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

— low access time
— low storage cost
— large capacity

These requirements are usually conflicting.

5.2 THE MEMORY HIERARCHY

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

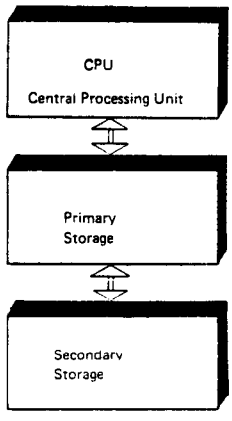


Figure 5.2.1: Two Level Storage System.

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.

5.3 ND-100 MEMORY SYSTEM

In the ND-100 memory system, a compromise between the conflicting requirements is achieved through the implementation of a multilevel hierarchical memory system. Figure 5.3.1 shows the major building blocks in this system.

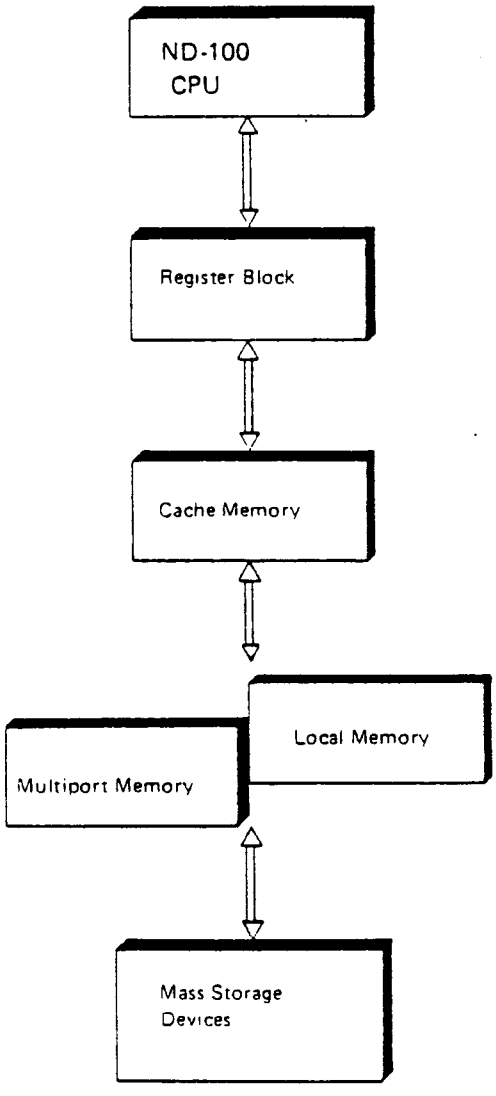


Figure 5.3.1: Multilevel Storage System

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 microprogram.

— 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 of 1 M words (2 M 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 multiport memory may be installed. Multiport memory is accessed through a multiport memory channel transceiver connected to one port in a separate card crate. The multiport memory may have several ports, allowing several sources to access the same physical memory area.
  In principle there is no difference between a local memory and a multiport (remote) memory, except that the access time is longer for the multiport memory. In addition, a multiport memory allows communication between several processors.
  The multiport memory channel is further described in Multiport Memory Channel Specifications (ND-10.006.).

— The next level of the ND-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 be 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 ND-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.

5.4 MOS MEMORY OPERATING PRINCIPLES

5.4.1 General

The read/write memory used in ND-100 is random access MOS memory (RAM). Principally, two categories 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 a very fast memory. Therefore, static RAMs are used in the ND-100 cache memory to have a very short 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, ie., 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 ND-100, and only this type will be discussed in the following.

5.4.2 A Memory Cell

A typical three transistor dynamic RAM cell is depicted in figure 5.4.1.

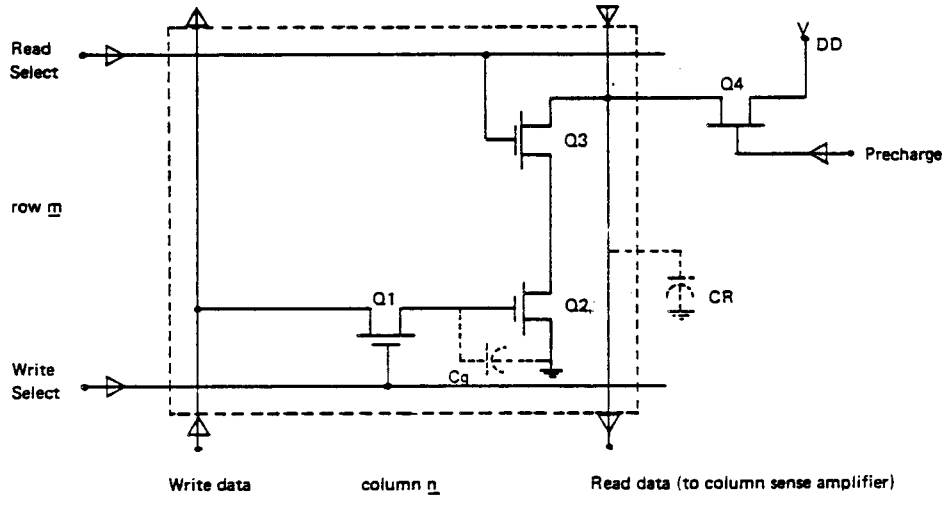


Figure 5.4.1: A Memory Cell

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 capacitance of the read data column 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 impedance 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'.

5.4.3 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 5.4.2.

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

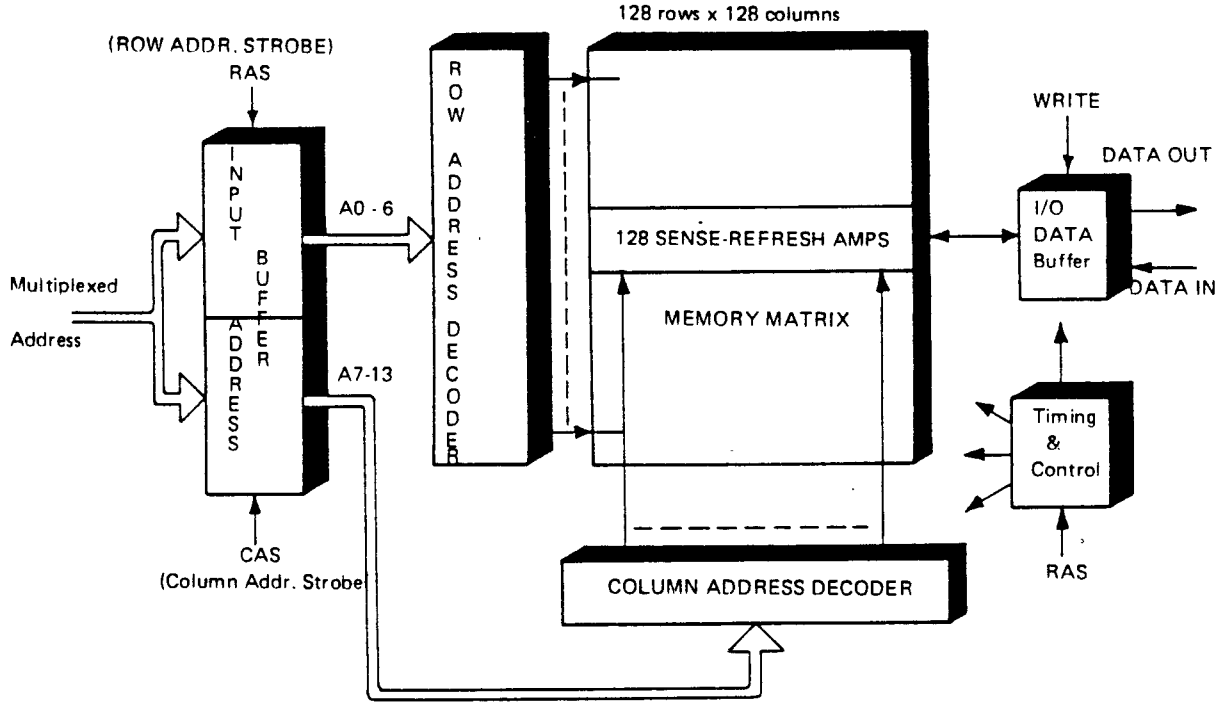


Figure 5.4.2: A Memory Chip — Functional Blocks

5.4.3.1 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 fed back to their respective cells, refreshing the voltage levels in the cells.

5.4.3.2 Read Cycle

Refer to the timing diagram, figure 5.4.3.

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.

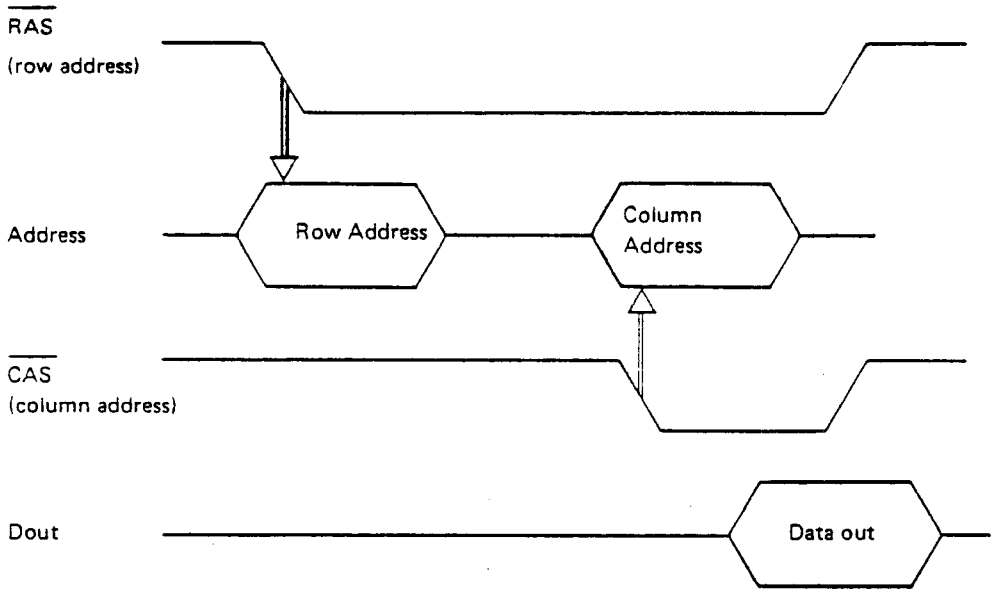


Figure 5.4.3: Read Cycle

5.4.3.3 Write Cycle

Refer to the timing diagram, figure 5.4.4.

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.

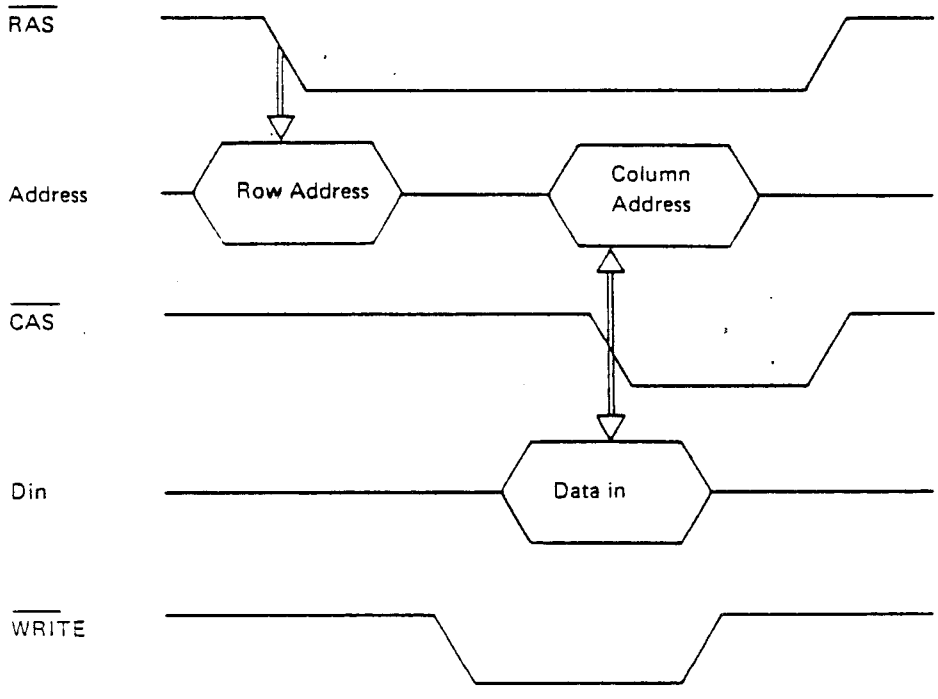


Figure 5.4.4: Write Cycle

5.4.3.4 Refresh Cycle

Refer to the timing diagram, figure 5.4.5.

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 to 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.

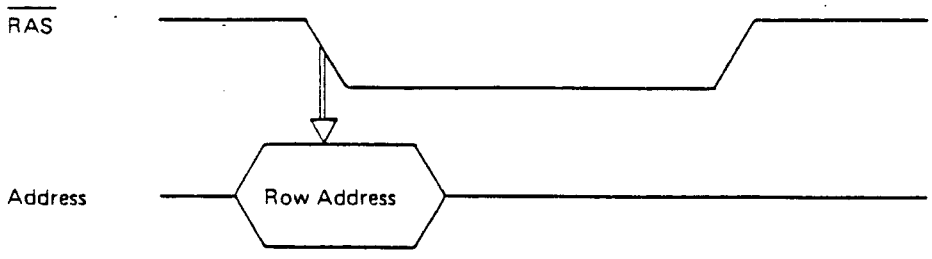


Figure 5.4.5: Refresh Cycle

5.5 CACHE MEMORY

5.5.1 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.

5.5.2 Cache Memory Architecture

5.5.2.1 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.

5.5.2.2 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 5.5.1 shows the hit ratio versus cache size for a given program.

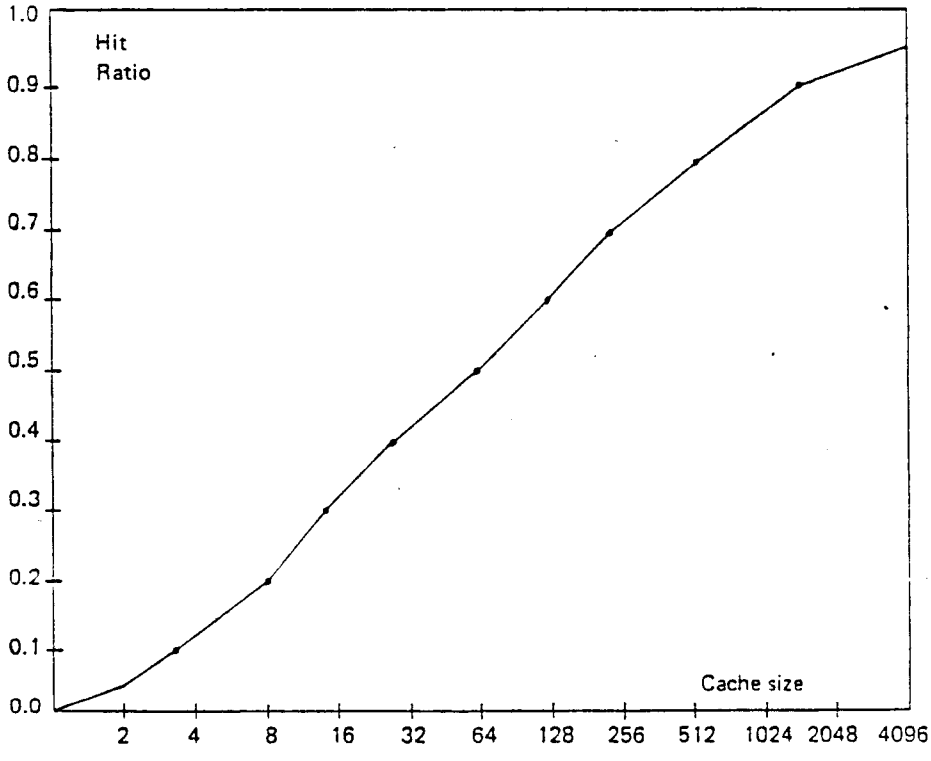


Figure 5.5.1: Hit Ratio versus Cache Size

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 (ie., 1 K words).

5.5.2.3 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 to 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 that it is:

— 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.

5.5.2.4 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 5.5.2 where the abscissa illustrates the number of memory references.

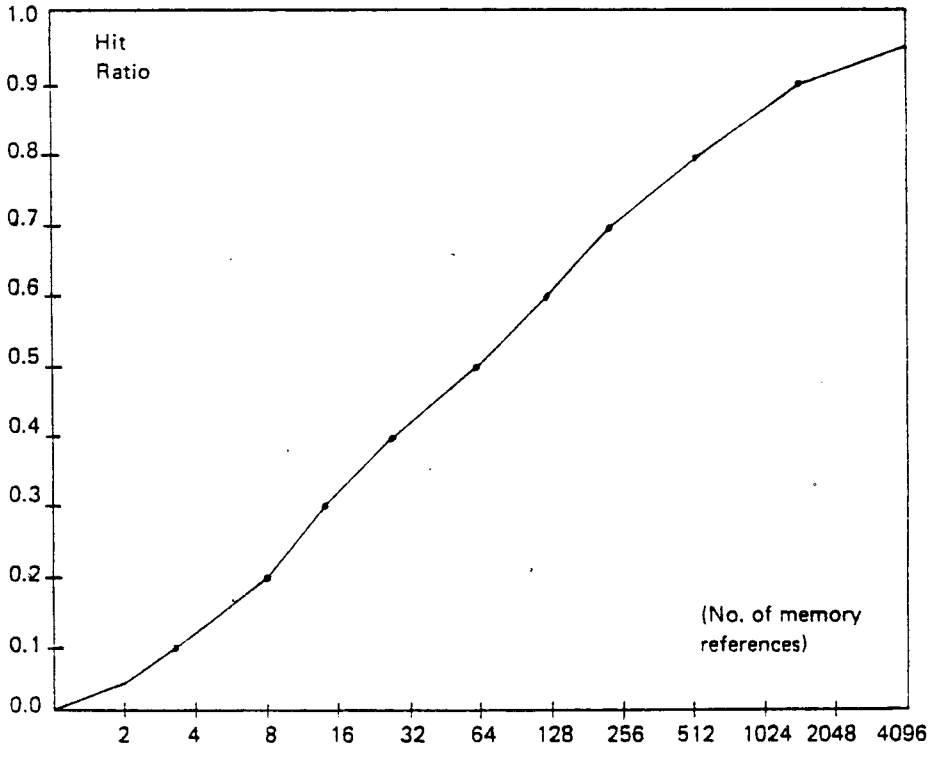


Figure 5.5.2: Hit Ratio versus Number of Memory References

5.5.3 Cache Memory Organization

The cache memory is homogeneous, ie., the cache memory does not discriminate between data words, instructions or indirect addresses which are stored in main memory.

The cache memory is organized as a 1 K by 31 bits lookup table, handling both the normal and the extended mode. See figure 5.5.3. Each word in cache is a copy of a word on one of the physical pages in the 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) indicating 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 by 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 unequal, the cache word belongs to a page other 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 contents of address 17, 2017 and 14017 will all be put in cache address 17. The contents of directory location 17 will be 0, 1 and 6 respectively.

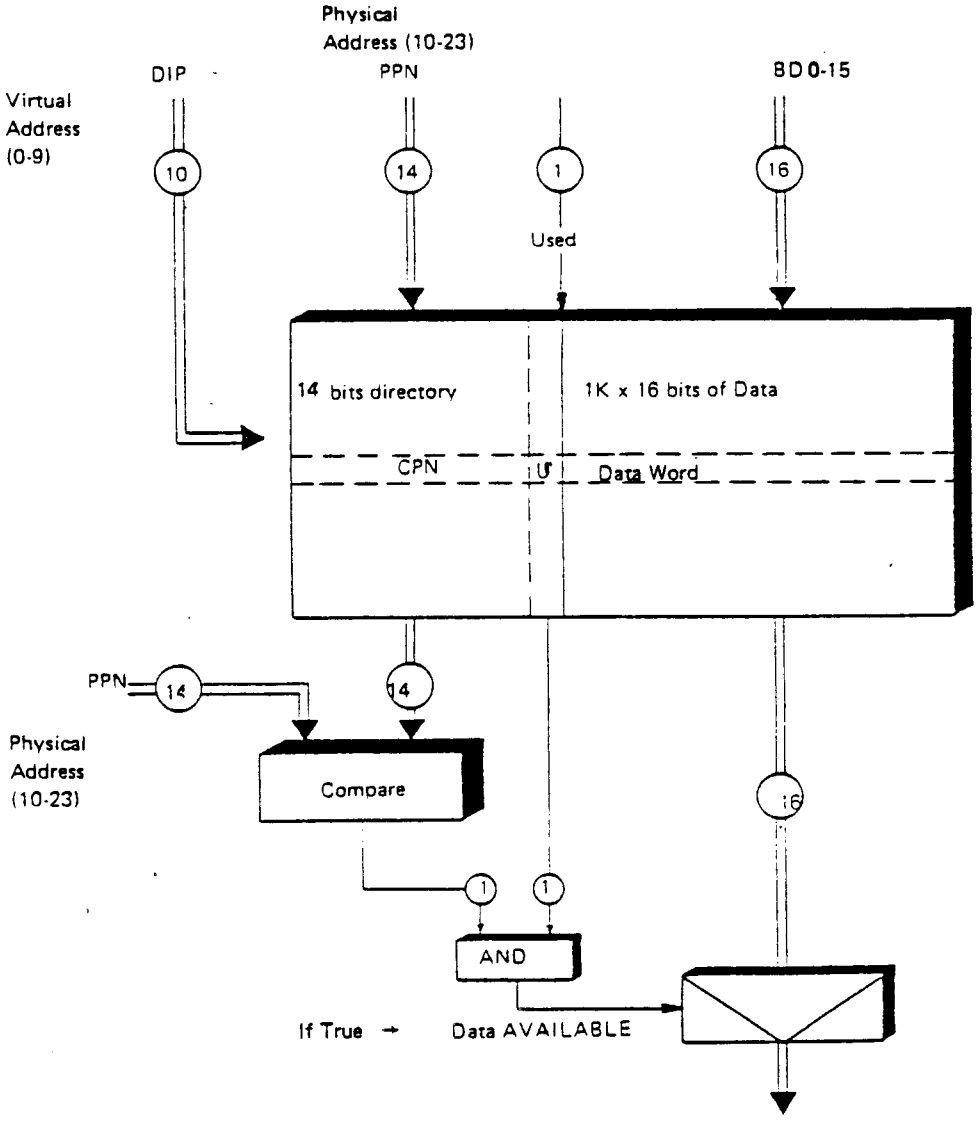


Figure 5.5.3: Cache Operation Principles

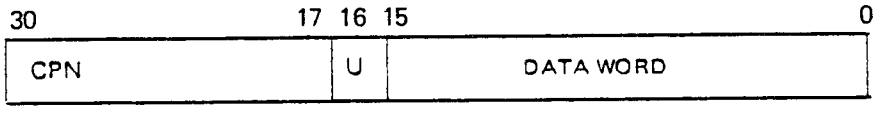


Figure 5.5.4: Format of One Cache Word

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 the main memory.

5.5.4 Cache Memory Access

5.5.4.1 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 5.5.5.

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 lookup is done in parallel with the page table lookup. 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 the address to the ND-100 bus, is then inhibited.

5.5.4.2 Write Access

Refer to figure 5.5.5.

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

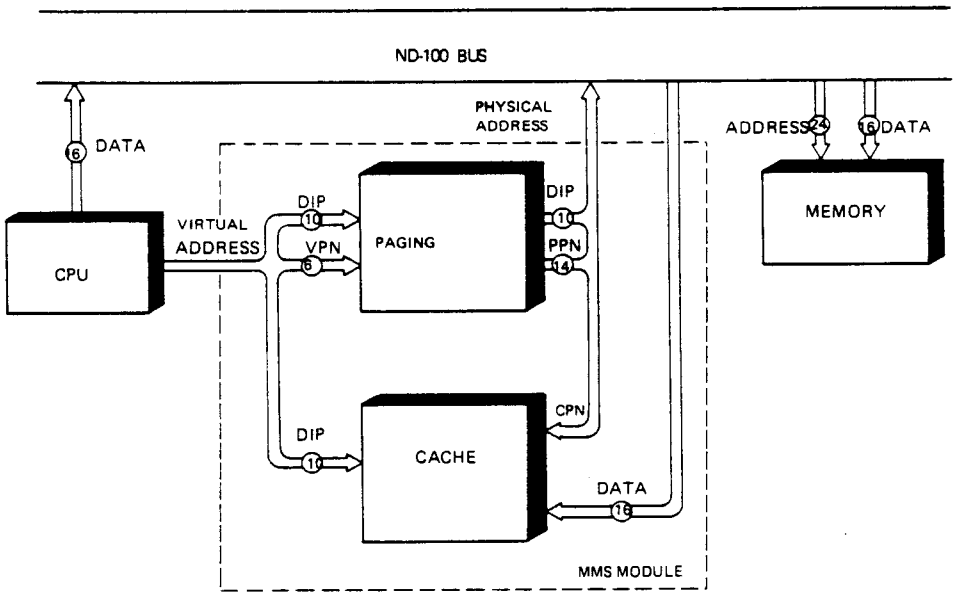


Figure 5.5.5: Cache Memory, Write Access

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.

5.5.4.3 Read Access

Refer to figure 5.5.6.

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

The virtual address, calculated at 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), bits 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 ND-100 bus it is also written into the data part of cache memory, the PPN is 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.

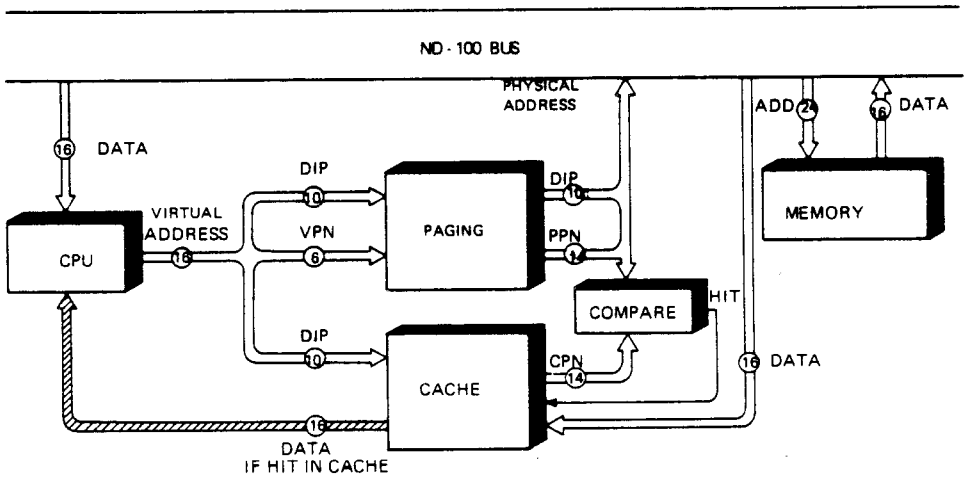


Figure 5.5.6: Chach Memory, Read Access

5.5.5 Cache Control and Status

Cache memory contains:

— 3 registers for control
— 1 status register for feedback

5.5.5.1 Cache Control

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

— clear cache
— set the cache inhibit limit registers

Clearing Cache

The ND-100 cache concept requires that all changes in main memory be updated in cache. This is done automtically when the CPU writes to memory. A DMA transfer will not be mapped through cache, however; hence a DMA transfer would result in different data in cache and memory. Because of this, the operating system will execute 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 60 µs, 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, ie., data in this area will not be copied into the cache when accessed. The inhibited area includes all pages with

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

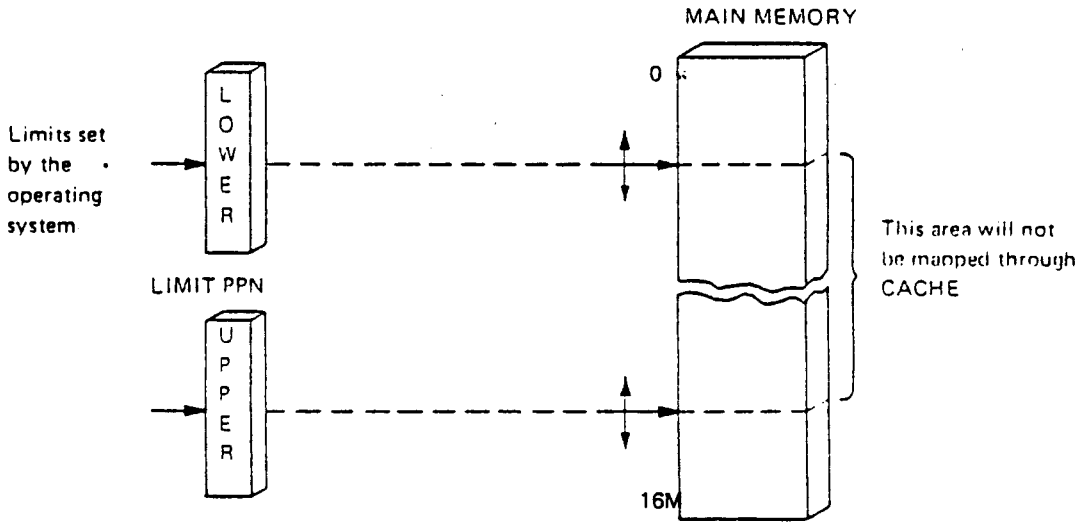


Figure 5.5.7: Cache Inhibit Limits

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 invalid data that might reside in cache.

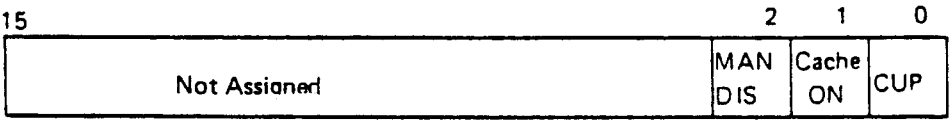
Note that data is not *removed* from cache when the cache inhibit area is expanded. Therefore, expansion of the cache inhibit area must always be accompanied by clear cache. Note that the whole address range is inhibited after master clear.

5.5.5.2 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, ie., 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 60 µs period, following cache clear and master clear. If bit 2, MAN DIS is '1', cache on will be '0'.

Bit 2: MAN DIS

Manual disable of cache.

'1' if disabled.
'0' if not disabled.

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

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

5.6 LOCAL MEMORY

5.6.1 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 1 M words per memory module

— Direct connection to ND-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 ND-100 bus. All communication with memory is performed over this bus. The ND-100 bus connects the memory to CPU, as well as interfaces for direct memory access (DMA) communication.

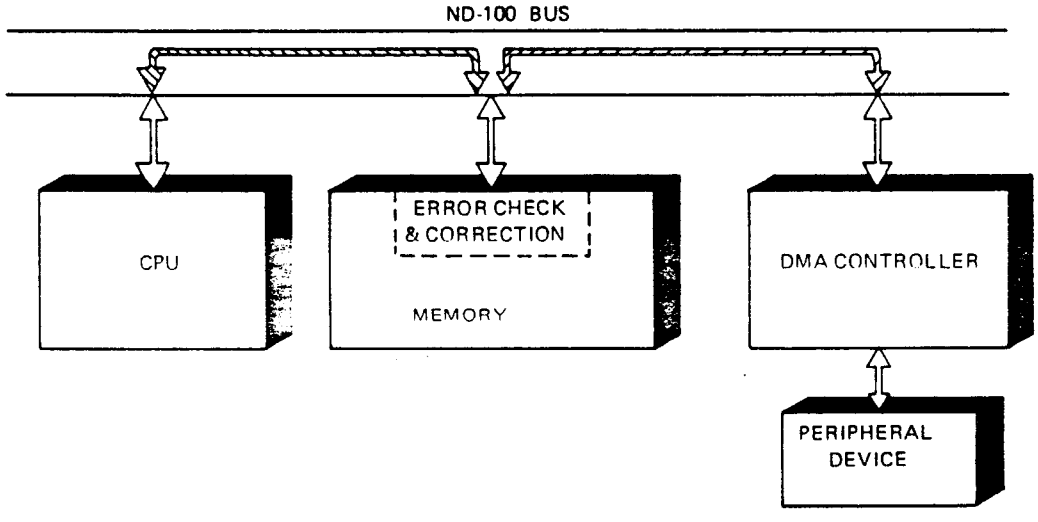


Figure 5.6.1: Memory Interconnection

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

5.6.2 Memory Module Placement

Memory modules should be placed from the rear position (position 12 or 21) in the ND-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

5.6.2.1 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 at 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. Reference appendix D for the different memory modules' thumbwheel setting.

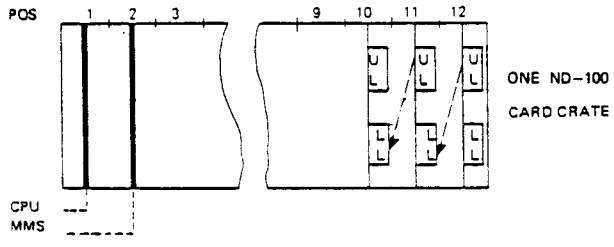


Figure 5.6.2: The Memory Module Placement in the Card Crate

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 settings 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 generated 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 with equal memory size must then be used.

5.6.3 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 ND-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:

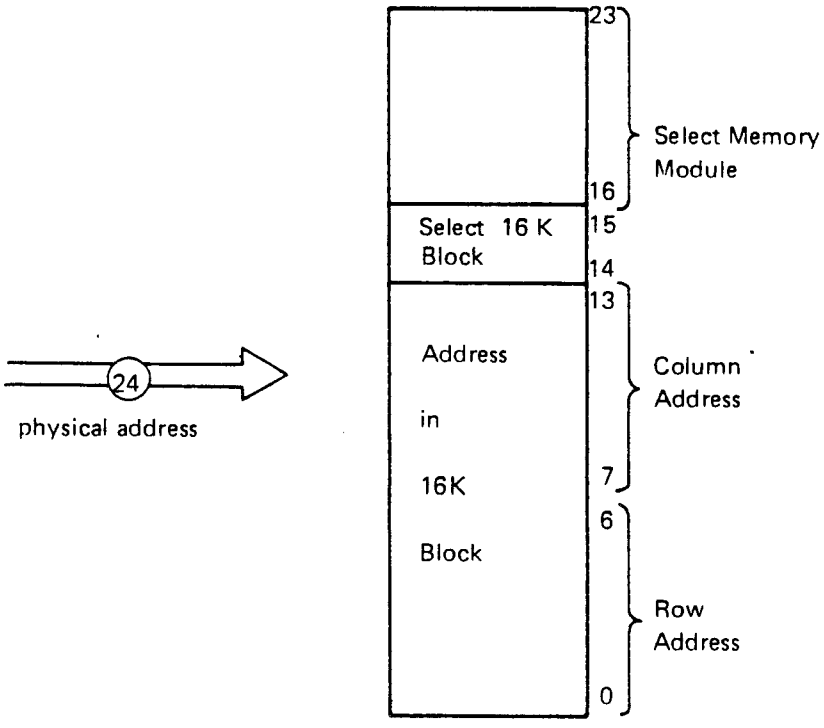


Figure 5.6.3: Address Decoding, Using 64 K Words Memory Modules

Bits 14 - 23 of the physical address select the memory module. Each of the 64 K 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 are 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 5.4.3).

The physical address range for the different modules is determined by the thumbwheel setting and the size of each module, or the position 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 a light in the yellow lamp labelled 0 - 16; block 1 lights the yellow lamp labelled 16 - 32, etc.

5.6.4 Data

The ND-100 bus is 24 bits wide, but data to/from 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 5.6.4.

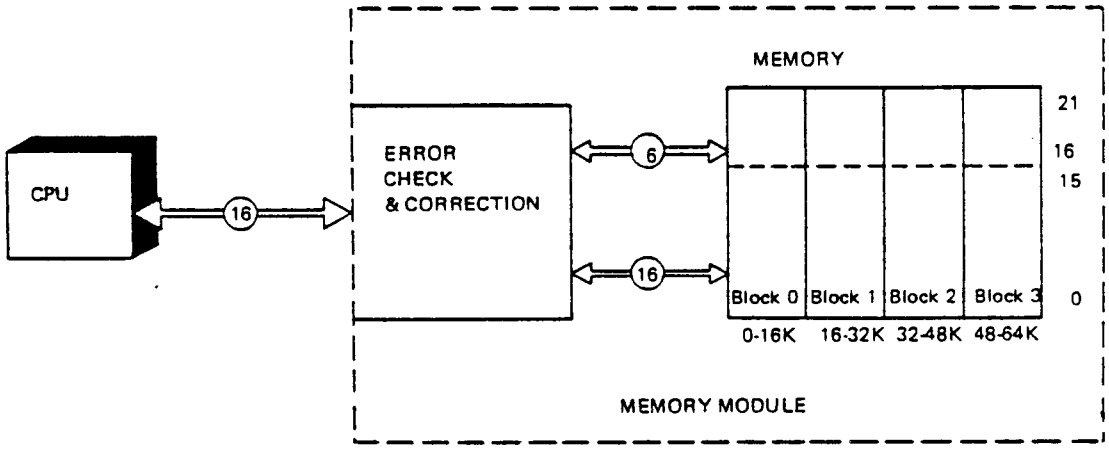


Figure 5.6.4: Error Correction Memory, 64 K Modules

The error correction network will be described in detail in section 5.7.

5.6.5 Memory Access

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

Local memory is available to three sources:

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 the bus control logic located on the CPU module (refer to chapter 4). The different control signals which must be established for proper operation are generated in the bus control.

Figure 5.6.5 illustrates the control signals on the ND-100 bus involved in a data transfer between memory and CPU.

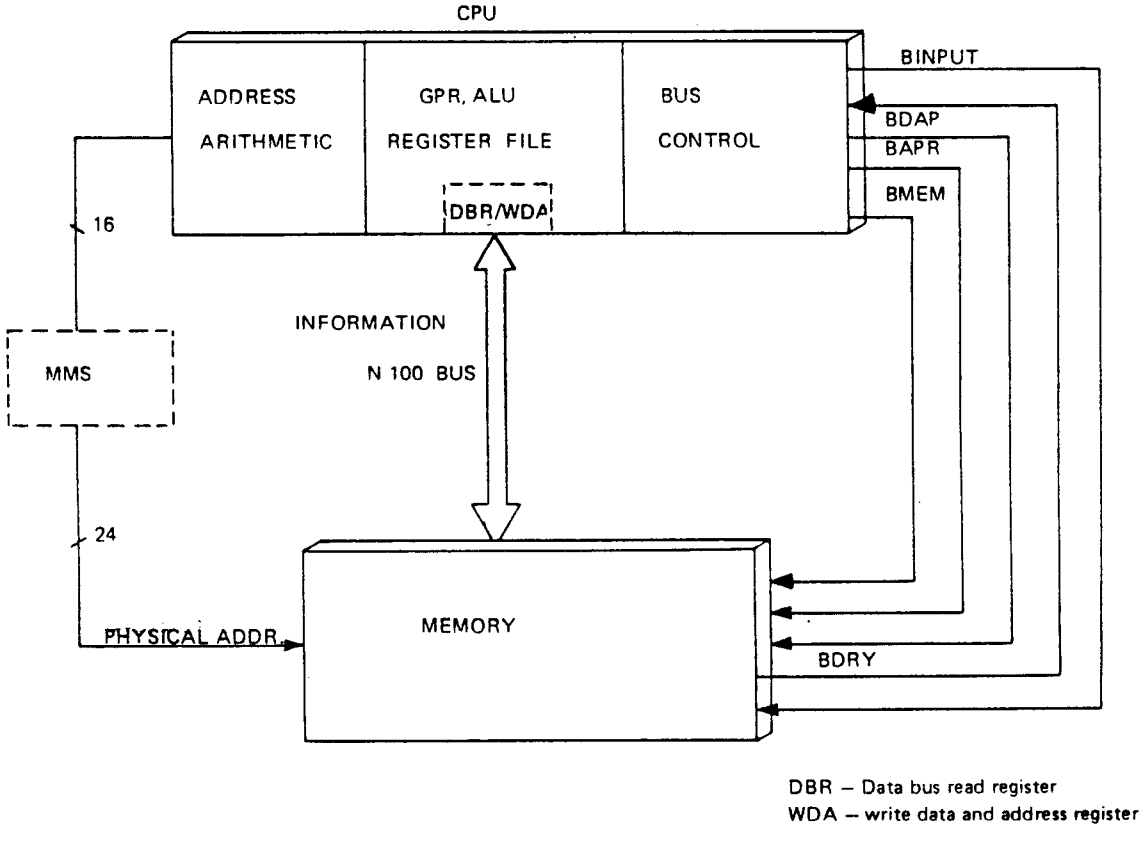


Figure 5.6.5: CPU — Memory Operation

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. A CPU access below is described. 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 ND-100 bus from the CPU, and 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 ND-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 are checked against the limits on all modules, and one is selected. On the selected one, the address is 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 ND-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 ND-100 bus, the memory gives the bus data ready (BDRY) signal to the CPU, indicating that data is ready on the bus. This signal will store the data in the data bus read register (DBR) in the CPU. The data is 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 transfer 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 ND-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.

5.6.6 Refresh

Refer to figure 5.6.6.

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' is sent on the ND-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 section 5.4). 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.

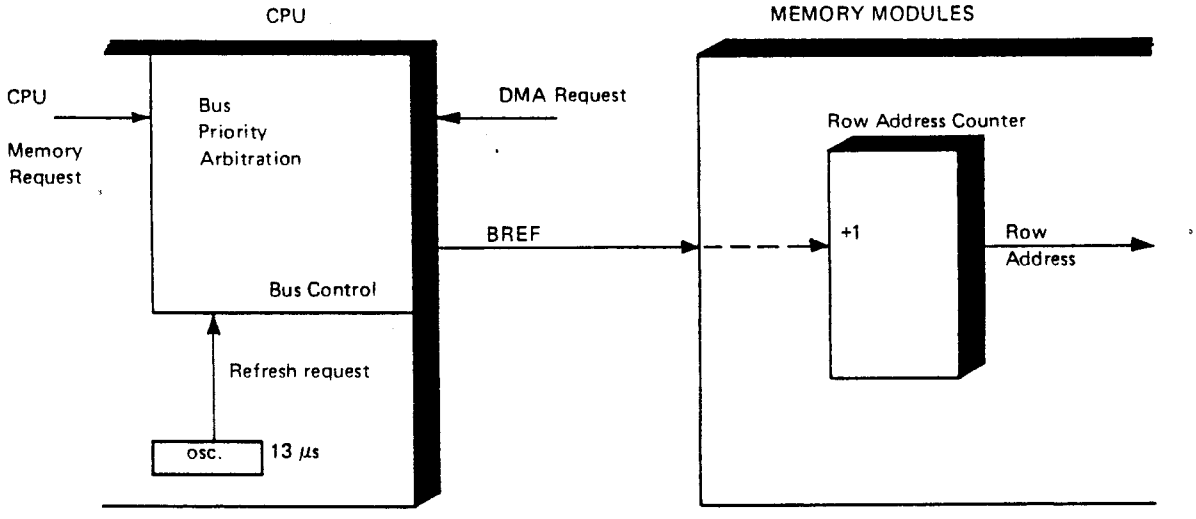


Figure 5.6.6: Refresh

5.7 MEMORY ERROR CHECK AND CORRECTION

5.7.1 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 great 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 (ie., 'soft errors').

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

The ECC used in ND-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 at the cost of:

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

However, great efforts 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 ND-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 noncorrectable memory failure

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

5.7.2. Functional Description

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

5.7.2.1 Parity Checking

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

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 leaves 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:

$P_R = P_R \oplus P_W \oplus P_W$

(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 changed, $S = 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 = 0$. If one bit has 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.

5.7.2.2 Error Correction

Each memory module has its own ECC network which is shown simplified in figure 5.7.1.

For the sake of clarity, parity generation and checking are shown separately, 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, called 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

Thus, it is the total parity of all 22 data and correction bits.

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.

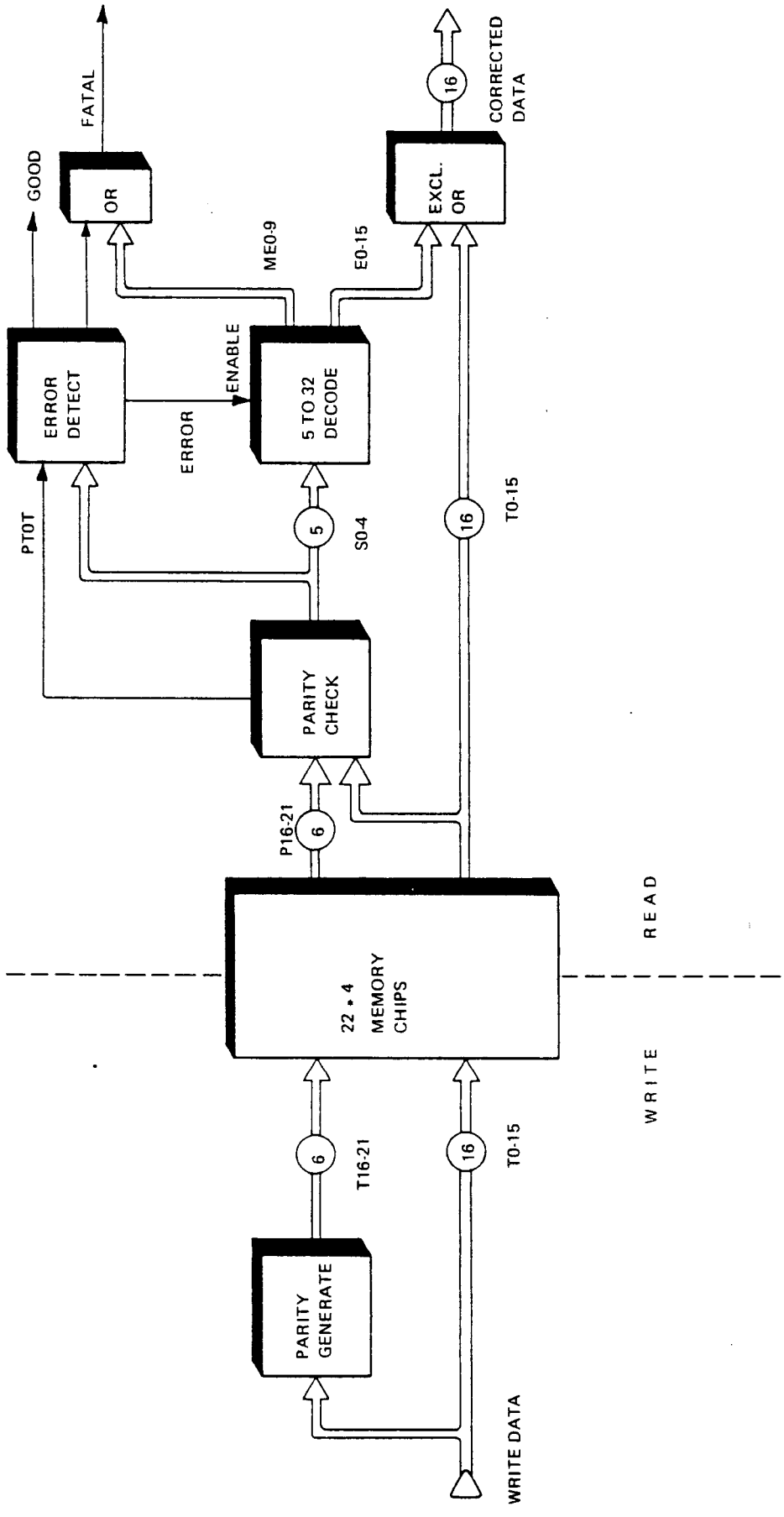


Figure 5.7.1: Error Correction Network During Write and Read

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 (eg., 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 5.7.1 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	Syndrom Bits S5	S4	S3	S2	S1	S0	No Error	Single code Error	Single data Error	Fatal 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	0	1			E9	
23	0	1	0	0	1	1			E10	
24	0	1	0	1	0	0			E11	
25	0	1	0	1	0	1				
26	0	1	0	1	1	0			E12	
27	0	1	0	1	1	1				
30	0	1	1	0	0	0			E13	
31	0	1	1	0	0	1				
32	0	1	1	0	1	0			E14	
33	0	1	1	0	1	1				
34	0	1	1	1	0	0			E15	
35	0	1	1	1	0	1				
36	0	1	1	1	1	0				
37	0	1	1	1	1	1		EC5		
40	1	0	0	0	0	0				Multiple Errors
"	"									
74	1	1	1	0	0	0				
75	1	1	1	1	0	1		Lower byte parity error		For two 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 parity error		

Table 5.7.1: Error Codes as Reported in the PES Register

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

Since 21 bits can produce more than 22 error codes (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 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 $\geqslant 3$ produce the same code as some GOOD and SING error patterns. However, that should be expected because multiple errors must produce *some* code and there are only 64 available.

Table 5.7.2 shows how many occurrences of each code are 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 of 100 K — 1 M words the mean time between triple failure will be of the order 100 - 1000 years.

Revert to figure 5.7.1. The section ERROR DETECT checks syndrome bits and PTOT according to the conditions mentioned. When errors are revealed, a decoder is enabled that decides 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 5.7.2: Distribution of Syndrome Code Occurrences for 0-6 Erroneous Bits

5.7.3 Implementation

The practical implementation of ECC is as outlined in figure 5.7.1, with a few modifications.

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

(The single names on the detailed diagram will therefore differ from figure 5.7.1.)

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 caught by hardware and given the nonfatal code 37.

Some find it confusing that some of the parities are stored in memory in EVEN = true and others 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 to arise 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, and failure may result if it is 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 until the disable switch is operated.

5.8 MEMORY CONTROL AND STATUS

5.8.1 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:

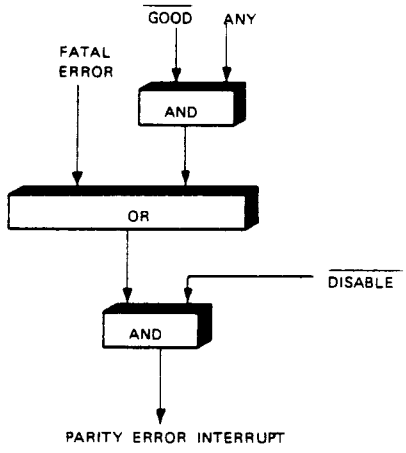


Figure 5.8.1: Parity Error Interrupt

Operation of the test bits produces 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 bits 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:

	TST15	TST6	TST0	Error Code	Interpretation
ECC bit:	1	4	0	PES bit 13-8	from table 5.7.1
	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 produce an error code in PES, indicating that a single-bit error has been corrected. The multiple errors have 'FATAL' set, and generate an interrupt.

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

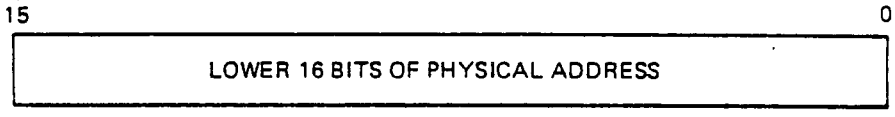
5.8.2 Memory Status Registers

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)

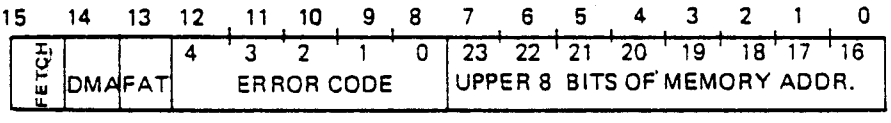
Both registers are read to the A register by the TRA instruction.

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 5.7.1).

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 occurred during DMA reference.

Bit 15: Fetch - error occurred during instruction fetch, an EXAM or a DEPO instruction.

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

5.9 ERROR LOGGING

Using the error correction feature, error correction is automatically done on single-bit errors without the CPU being 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.

6 THE INPUT/OUTPUT SYSTEM

6.1 GENERAL

The purpose of the input/putput system (I/O system) is to ensure the physical communication between connected peripheral equipment and the ND-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 between 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 according to customer requirements. The I/O configuration should be easy to change.

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

— with CPU controlled, Program Input/Output (PIO)

— with Direct Memory Access (DMA)

6.2 ND-100 BUS IN THE I/O SYSTEM

6.2.1 General

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

6.2.2 Organization of an I/O Device Controller Module

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

Figure 6.2.1 shows a standard ND-100 I/O module.

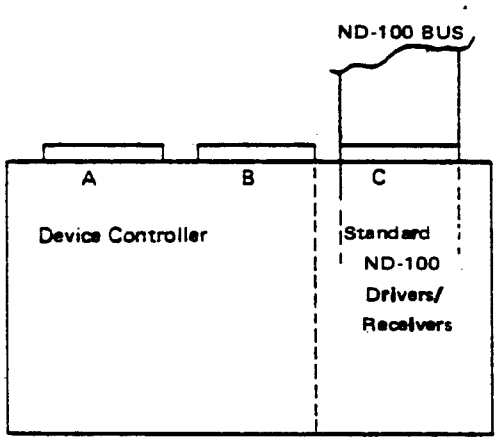


Figure 6.2.1: Standard ND-100 I/O Module

The ND-100 dependent part includes bus handshake control logic. This part is standardized 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)

6.2.3 Allocation of the ND-100 Bus

One of the functions of the Bus Control, is to allocate the ND-100 bus to one of the possible requesting bus users (refer to chapter 4). That is, to:

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

These sources request the ND-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 signal.

CPU Bus Request

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

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

* 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 ND-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.

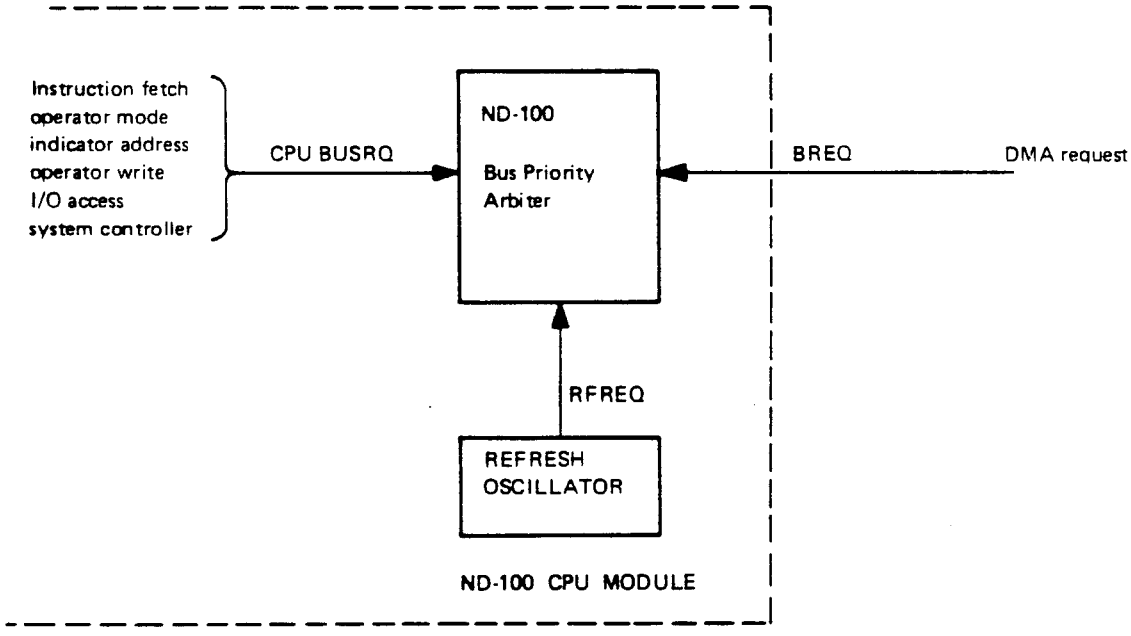


Figure 6.2.2: Requesting Sources to the Bus Control Unit

The ND-100 bus is allocated and released on a cycle basis, ie., for every byte/word to be exchanged. Due to the multiplexed bus (refer to chapter 4) the cycle may be divided into an address cycle and a data cycle.

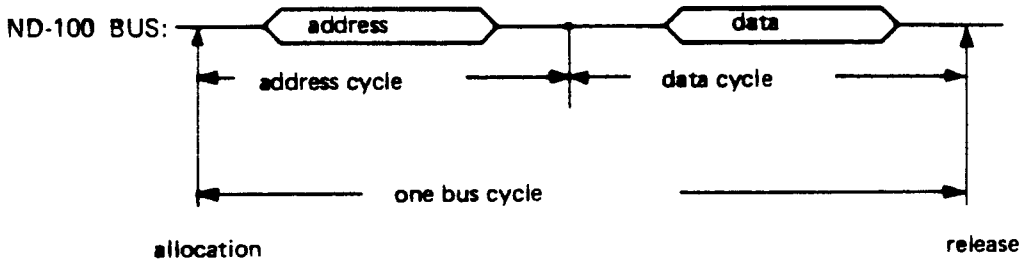


Figure 6.2.3: ND-100 Bus Cycle

The time from allocation to release of the ND-100 bus shall not exceed 15 µs. This is monitored by the Bus Control.

The accessed device, ie., 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 15 µs. The faulty cycle is reported to the CPU as internal interrupt on level 14 (MOR, IOXERR).

6.3 PROGRAMMED INPUT/OUTPUT — PIO

6.3.1 General

External devices may be classified as:

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

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

Data exchanged between the two classes of peripherals and ND-100 falls 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 6.3.1.

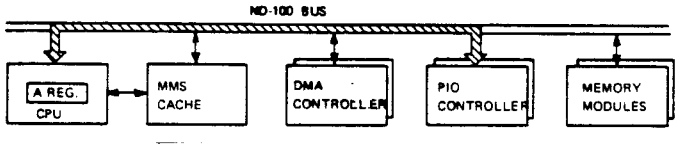


Figure 6.3.1: PIO Data Exchange

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.

6.3.2 The Input/Output Instructions IOX and IOXT

In the ND-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, ie., only privileged users may use these instructions.

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

In the ND-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 *depends 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 a special function to the exchanged information.

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

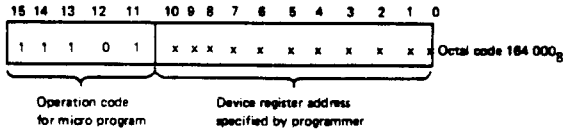
— transfer status information from PIO and DMA interface status registers.

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.

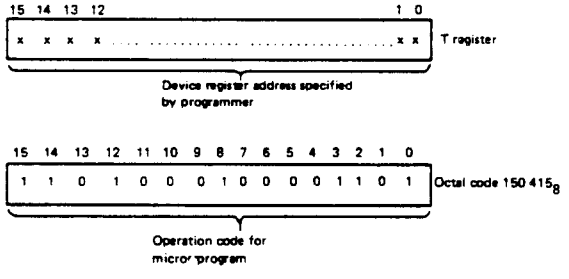
IOX <device register address>



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



6.3.3 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.

6.3.3.1 The IOX Instruction Transfer Direction

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

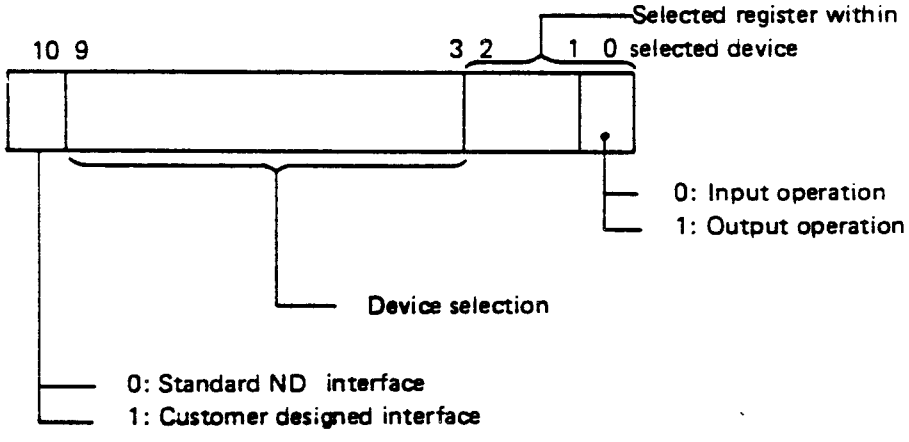


Figure 6.3.2: IOX Device Register Address Decoding Details

Input Transfer

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

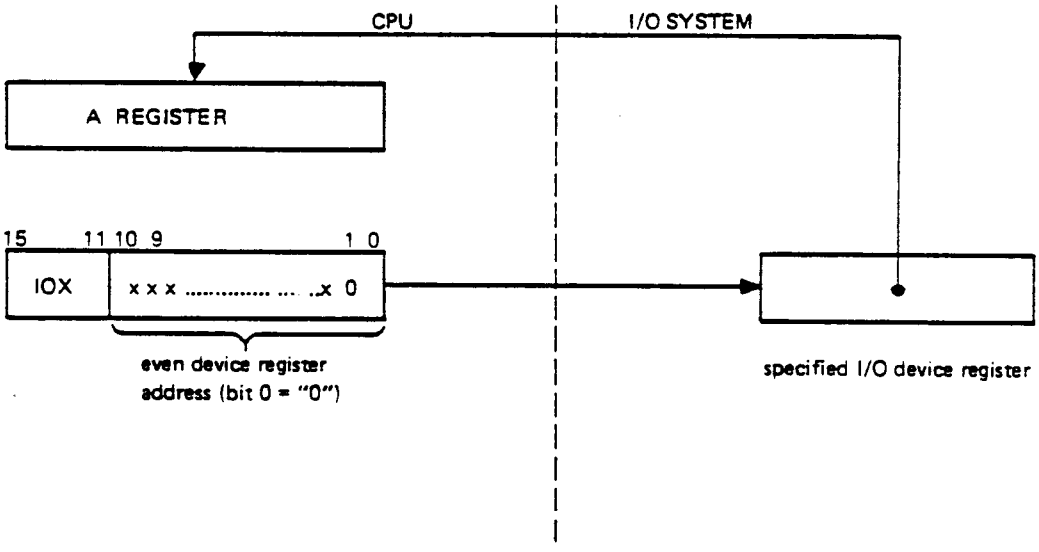


Figure 6.3.3: IOX Input Transfer

Output Transfer

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

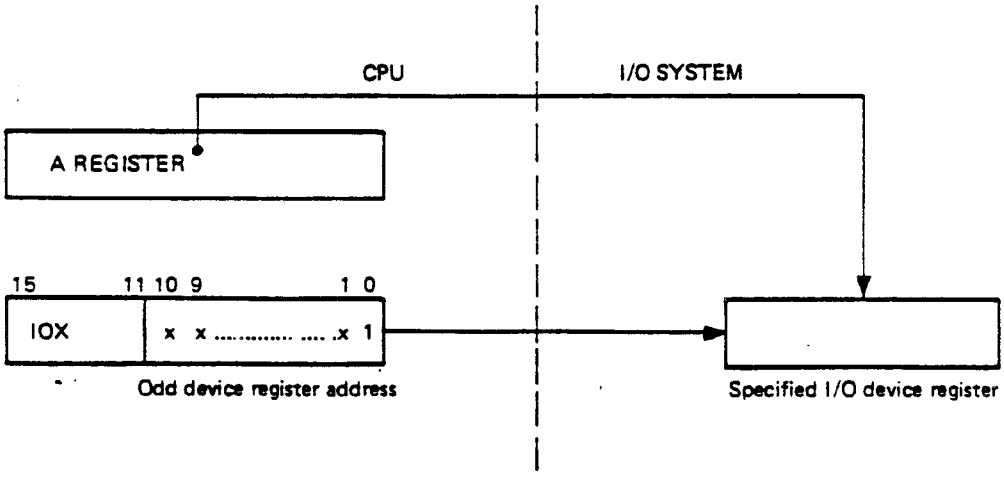


Figure 6.3.4: IOX Output Transfer

6.3.3.2 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.

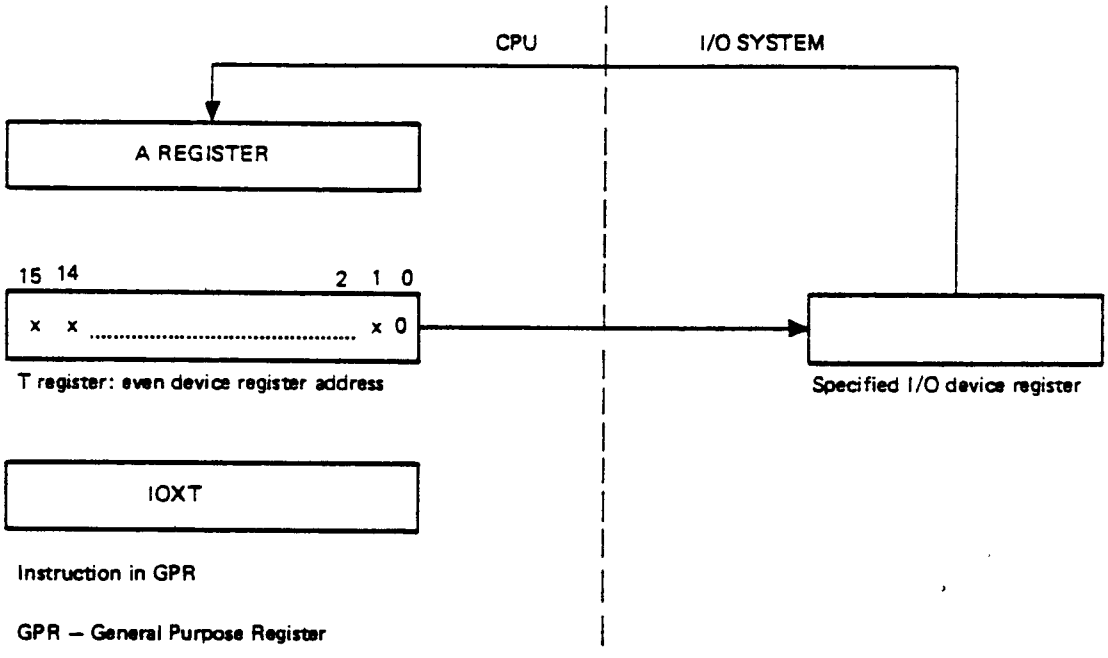


Figure 6.3.5: IOX Input Transfer

Output Transfer:

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

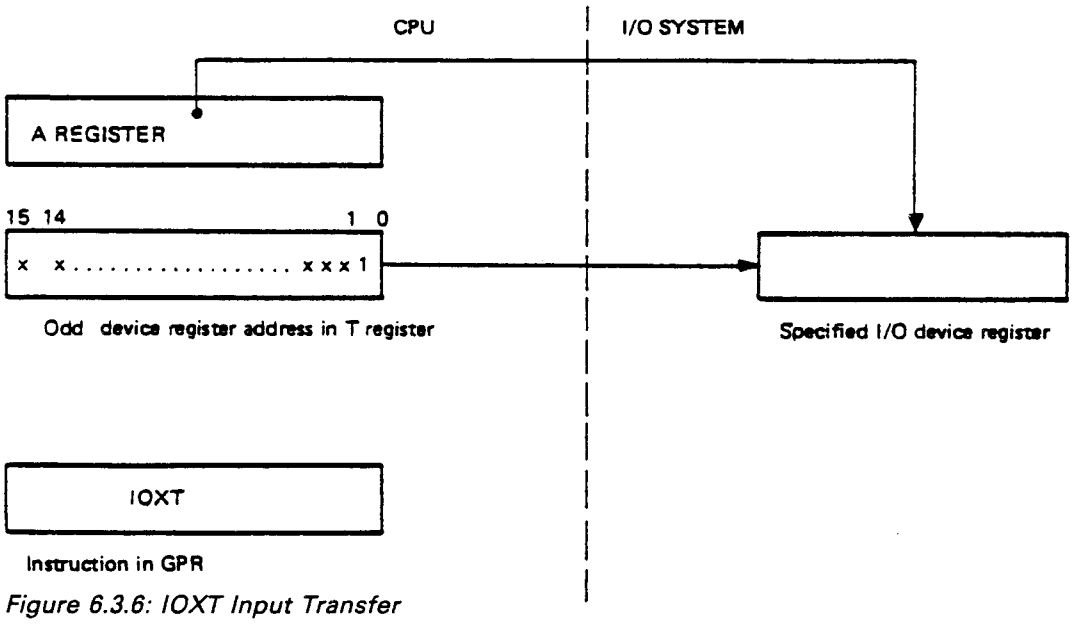


Figure 6.3.6: IOXT Input Transfer

6.3.4 Calculation of the Device Register Address

6.3.4.1 The IOX Instruction Address Range

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

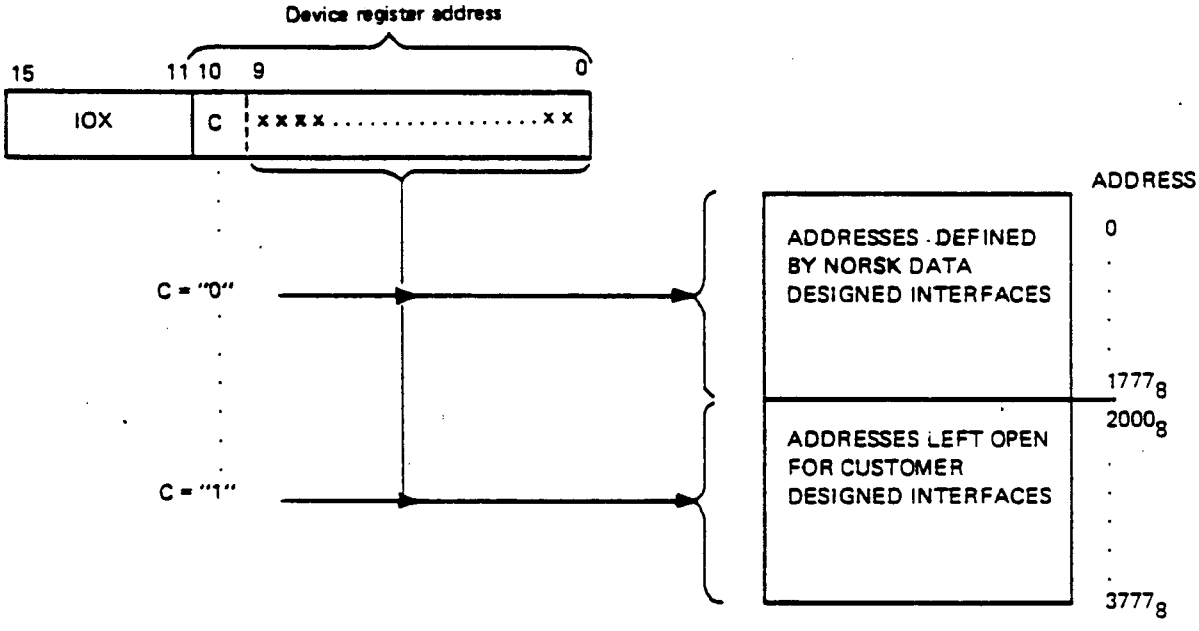


Figure 6.3.7: IOX Address Range

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.

6.3.4.2 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 64 K 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.

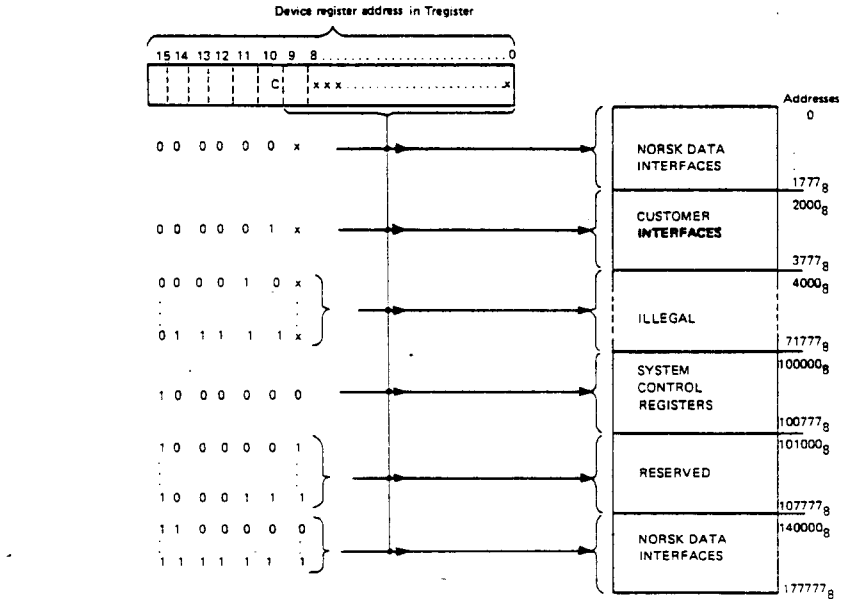


Figure 6.3.8: IOXT Address Range

As indicated, if the bits 11-15 are all zero, the device register address of the IOXT instruction overlaps 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 ND-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 ND-100 bus, the microprogram 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.

6.3.4.3 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 registers 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 into two parts:

— device number (hardware)

— register number. 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

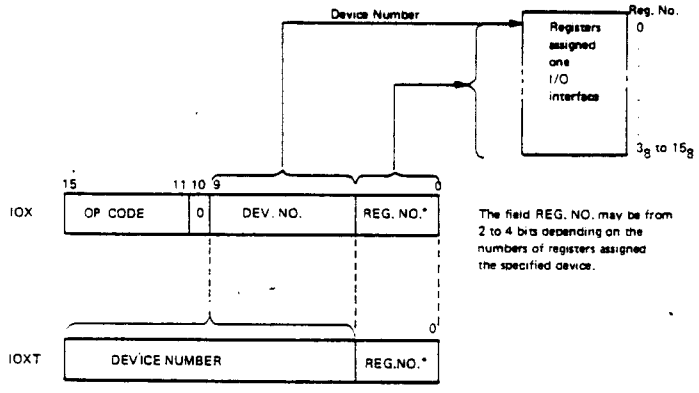


Figure 6.3.9: Specification of Device Register Address

For Norsk Data produced device controllers, both the device number and the register number have been standarized. See appendix C.

The numbers assigned to 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, which is the lowest device register address for terminal number 1. Eight registers are assigned to 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 to one class of peripherals.

Note that there is no relationship between a module device number and its slot number in the ND-100 bus.

6.3.4.4 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 are described (programming specifications for some I/O interfaces are 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 is 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 belong 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 registers assigned to 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,
```

6.4 CONTROL AND STATUS REGISTERS ON ND DESIGNED PIO AND DMA INTERFACES

6.4.1 General

All information exchange between PIO devices and ND-100 CPU, ie., control, status and data, is programmed via the A register by means of IOX/IOXT instructions.

On DMA controllers, control and status is 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.

6.4.2 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 }
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, ie., 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, ie., it will remain on until Master Clear, Device Clear (control register 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 register 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.

6.4.3 Format and Function of the Status Register

Feedback from a device to the CPU is 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 register bit 3 set or not set tells whether an I/O interface channel is ready to operate or not. The significance of '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, ie., 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 bits 5 - 15.

Bits 5 - 15 — Nondefined

Described in the programming specifications for the actual device.

6.5 LOADING SEQUENCE FOR DEVICE REGISTERS ON I/O INTERFACES

6.5.1 The Loading Sequence

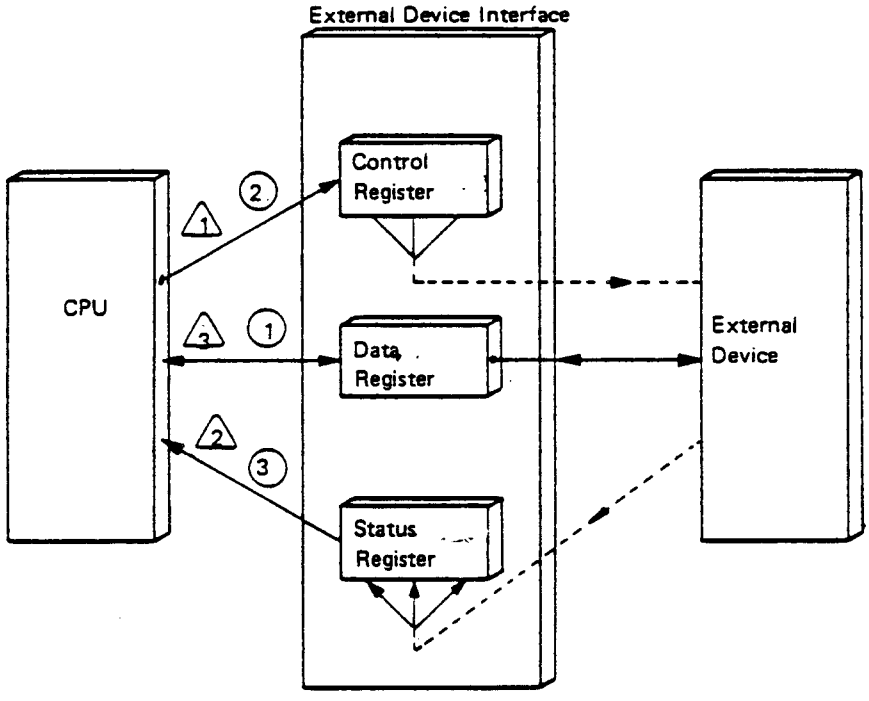


Figure 6.5.1: Loading Sequence for Device Registers

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 register, the CPU will discover that a new transfer can be initiated.

6.5.2 Programming Example of a Noninterruption 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 300 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
```

6.6 ND-100 BUS SIGNALS DURING IOX INSTRUCTIONS

6.6.1 IOX Input

Figure 6.6.1 shows the different control signals and the information present on the ND-100 bus during the execution of an IOX input instruction.

When starting execution of the IOX instruction, the 11-bit device register address is sent out on the ND-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 ND-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 current device interface now sends a 'BINPUT' signal, telling that this is an input request from a device. The bus control, as an answer to 'BINPUT', sends 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 is now enabled onto the ND-100 bus. The interface then issues the 'BDRY' — bus data ready — signal, telling that data is ready on the ND-100 bus. Data on the bus enabled into the A register, and the bus cycle is terminated.

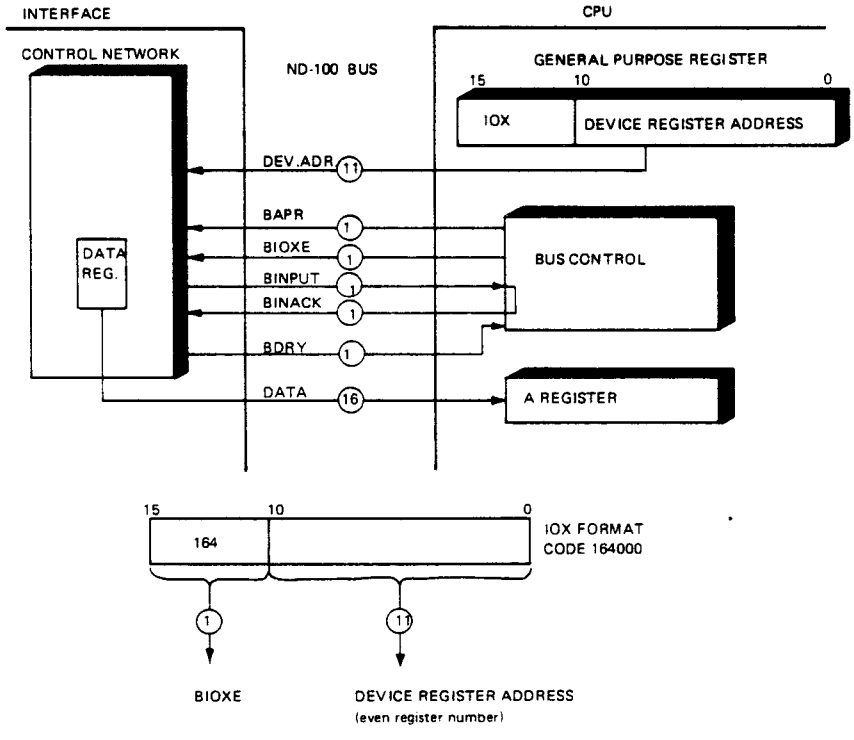


Figure 6.6.1: The Control Signals and Information Flow During an IOX Input Operation

6.6.2 IOX Output

Figure 6.6.2 shows the different control signals and the information present on the ND-100 bus during the execution of an IOX output instruction. The 11-bit device register address is sent out on the ND-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 is now enabled onto the ND-100 bus.

When data is accepted on the device interface, the 'BDRY' (bus data ready control signal) is issued from the interface. This signal terminates the bus cycle.

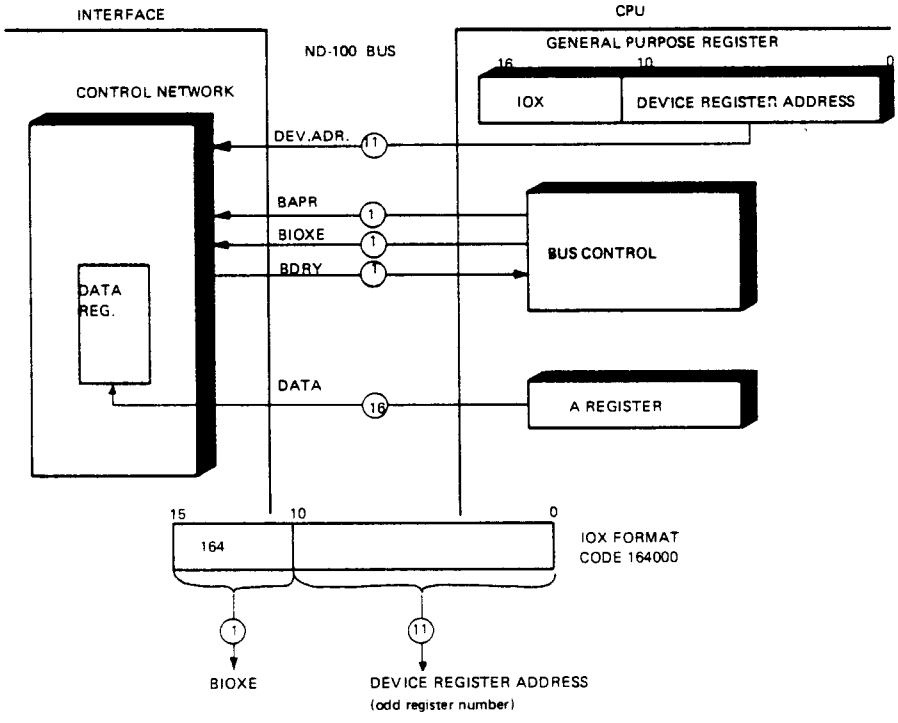


Figure 6.6.2: Control Signals and Information Flow during an IOX Output Operation

If 'BDRY' is not received in the bus control within 10 µs after starting the instruction, an interrupt occurs. 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.

6.7 THE I/O SYSTEM'S CONNECTION TO THE INTERRUPT SYSTEM

6.7.1 General

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

Possible status changes in the I/O system are:

— End of operation interrupt

  If output, this means data is transmitted, can accept next
  If input, this means data is available, please read it (before overrun)

— Error interrupt

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.

6.7.2 The I/O Device Controller Levels

Interrupt levels 10-13 and 15 may be activated by hardware through physical lines available in the ND-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 has been standardized:

— 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

6.7.3 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 occuring in a device sets the local interrupt flip-flop, which drives 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.

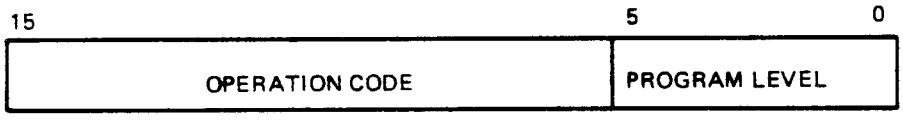
7. The CPU will resume the main program.

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.

6.7.4 The Ident Instruction

6.7.4.1 The Ident Instruction Format

IDENT <PL>



The ident instruction is a privileged machine instruction used in device interrupt identification. When executed, it 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.

6.7.4.2 ND-100 Bus Signals During Ident Instructions

Figure 6.7.1 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 ND-100 bus. The 'BAPR' — bus address present — signal is issued to tell all connected devices that a level code is present on the bus.

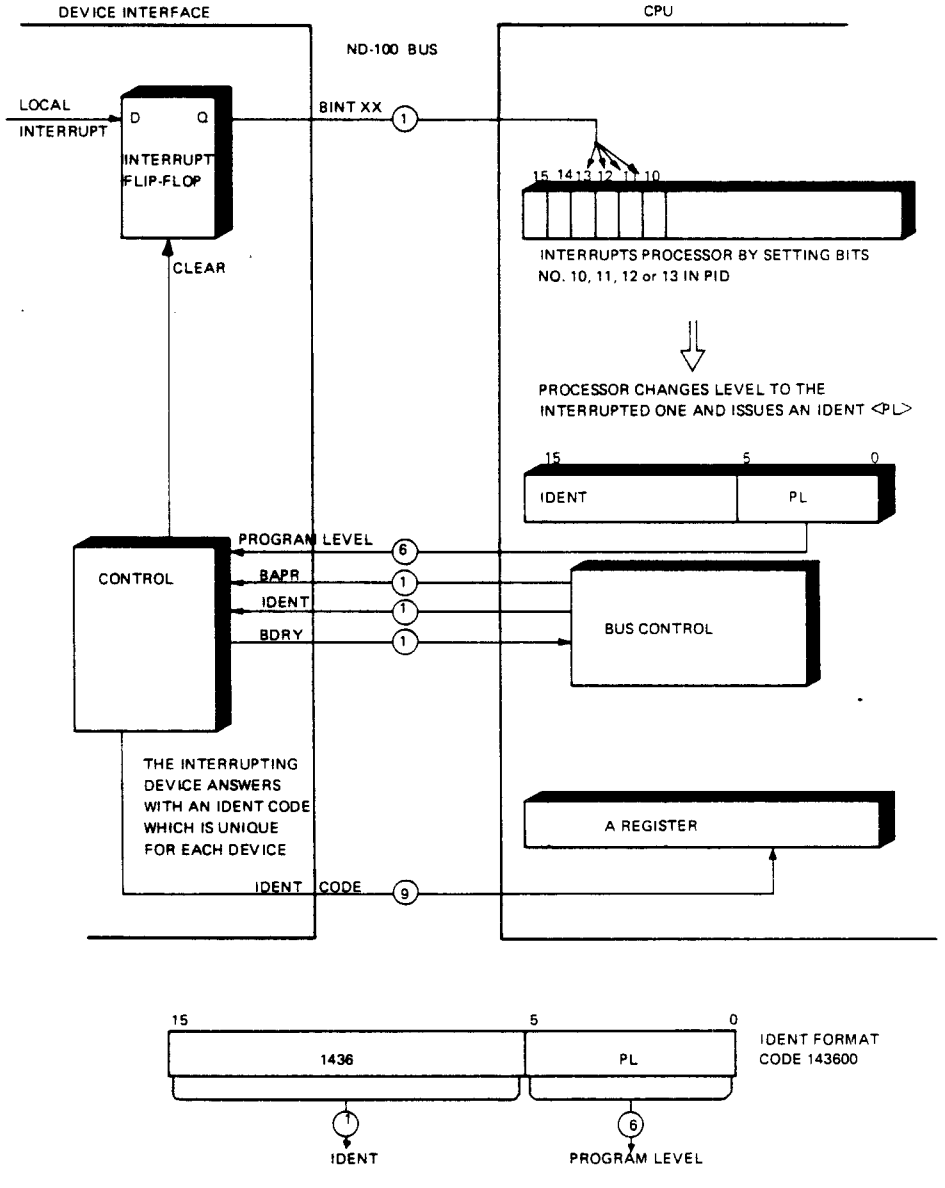


Figure 6.7.1: Control Signals and Information Flow During an IDENT Instruction

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 ND-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 ND-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.

6.7.5 Programming Input/Output Using Interrupt

Programming input/output by using waiting loops, as shown in section 6.5.2, is very ineffective. Most of the computer time will be spent in the input/output loops. This will be avoided by using the interrupt system. An interrupt will occur every time the device is ready for transfer.

Example:

Interrupt controlled driver for reading and printing of 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 enables for interrupt on device ready for transfer and interrupt on errors, and activates 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 enables for interrupt on device ready for transfer. A level 10 interrupt is 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.

6.8 DIRECT MEMORY ACCESS — DMA

6.8.1 General

The second class is for high speed block oriented peripherals, which exchanges data direct with the ND-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 ND-100 bus. This connection is called a DMA-channel.

The DMA interface, which controls the transfer, is DMA activated by a driver program in SINTRAN III. The activation includes telling the DMA 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 control in the CPU.

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 ND-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 6.8.1.

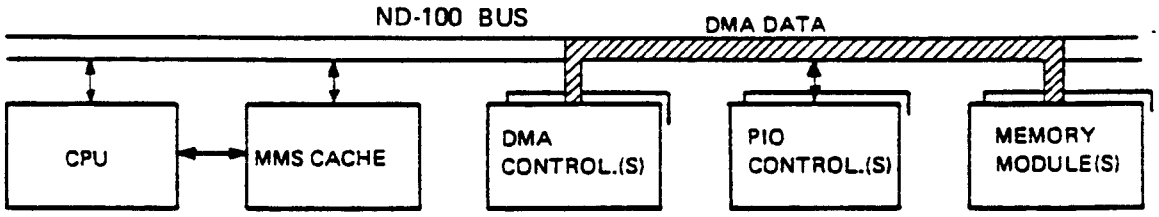


Figure 6.8.1: DMA Data Exchange

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

In ND-100 all DMA controllers have at least 16 words buffering between device and memory. That is, if the DMA channel is occupied for some reason, 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.

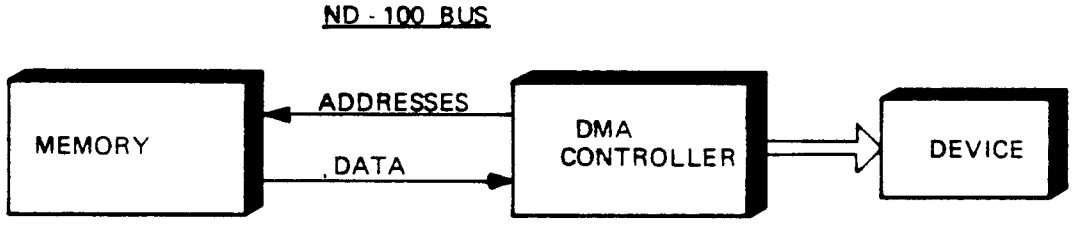


Figure 6.8.2: DMA Output

In a DMA input operation, the DMA controller gives both addresses and data to memory.

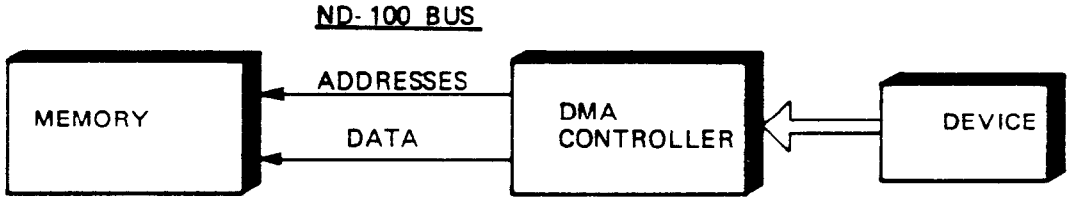


Figure 6.8.3: DMA Input

A DMA transfer may be divided into 3 steps:

— Initialization
— Transfer
— Termination and status check

6.8.2 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 from (DMA input) or write to (DMA input) 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 is given in the hardware programming specifications.

6.8.3 Transfer

After initialization and start are 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.

6.8.4 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 information on the status of the transfer.

6.8.5 ND-100 Bus Signals During a DMA Transfer

6.8.5.1 DMA Input

Figure 6.8.4 shows the different control signals and information which is present on the ND-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 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 ND-100 bus. This is used for enabling of addresses to the memory modules.

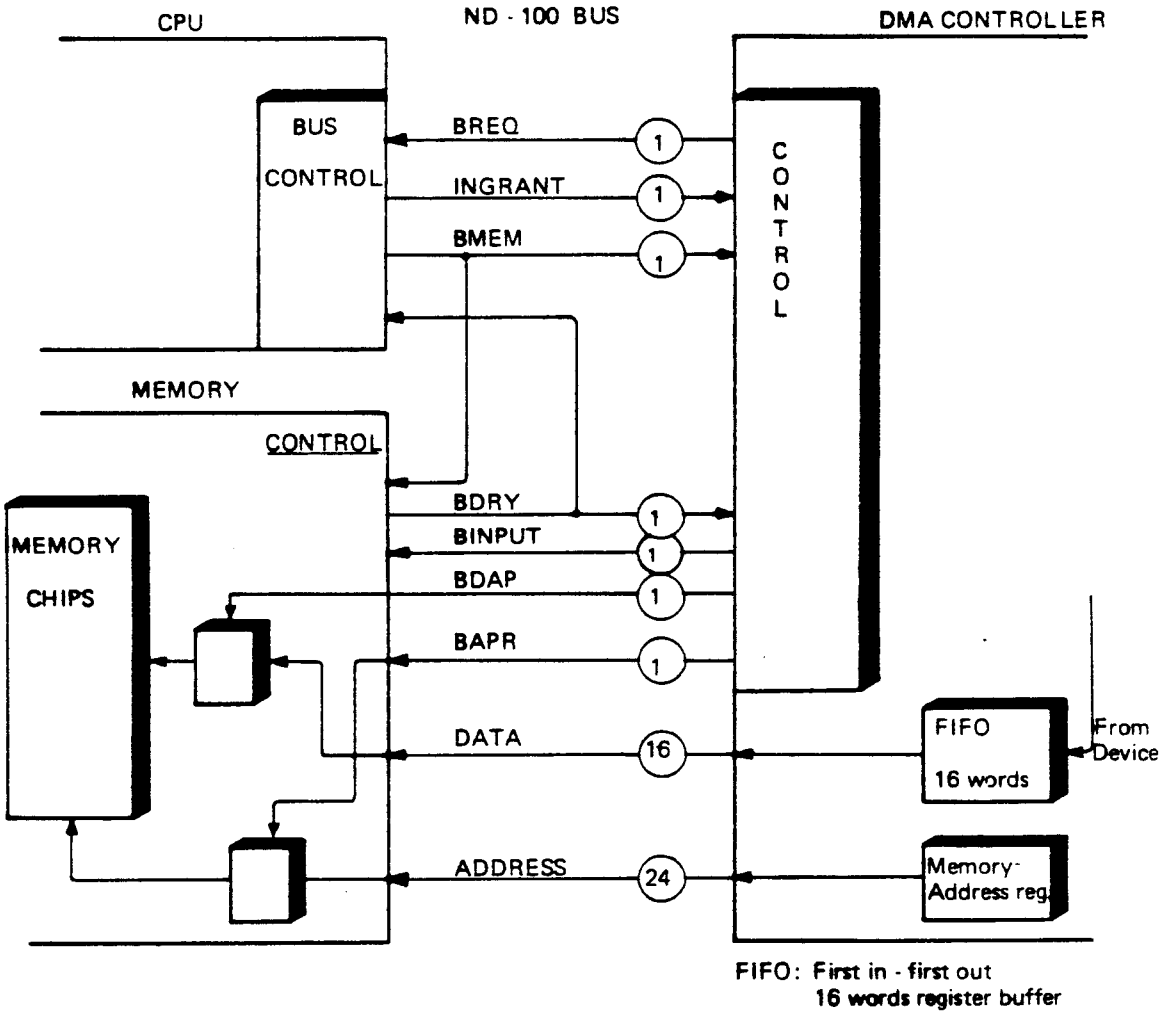


Figure 6.8.4: ND-100 Bus Signals During a DMA Input Transfer

Then the 16 bits of data will be present on the ND-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.

6.8.5.2 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 ND-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 ND-100 bus. Sixteen bits of data are then enabled to the ND-100 bus, and into the FIFO (first in-first out) register buffer on the DMA controller.

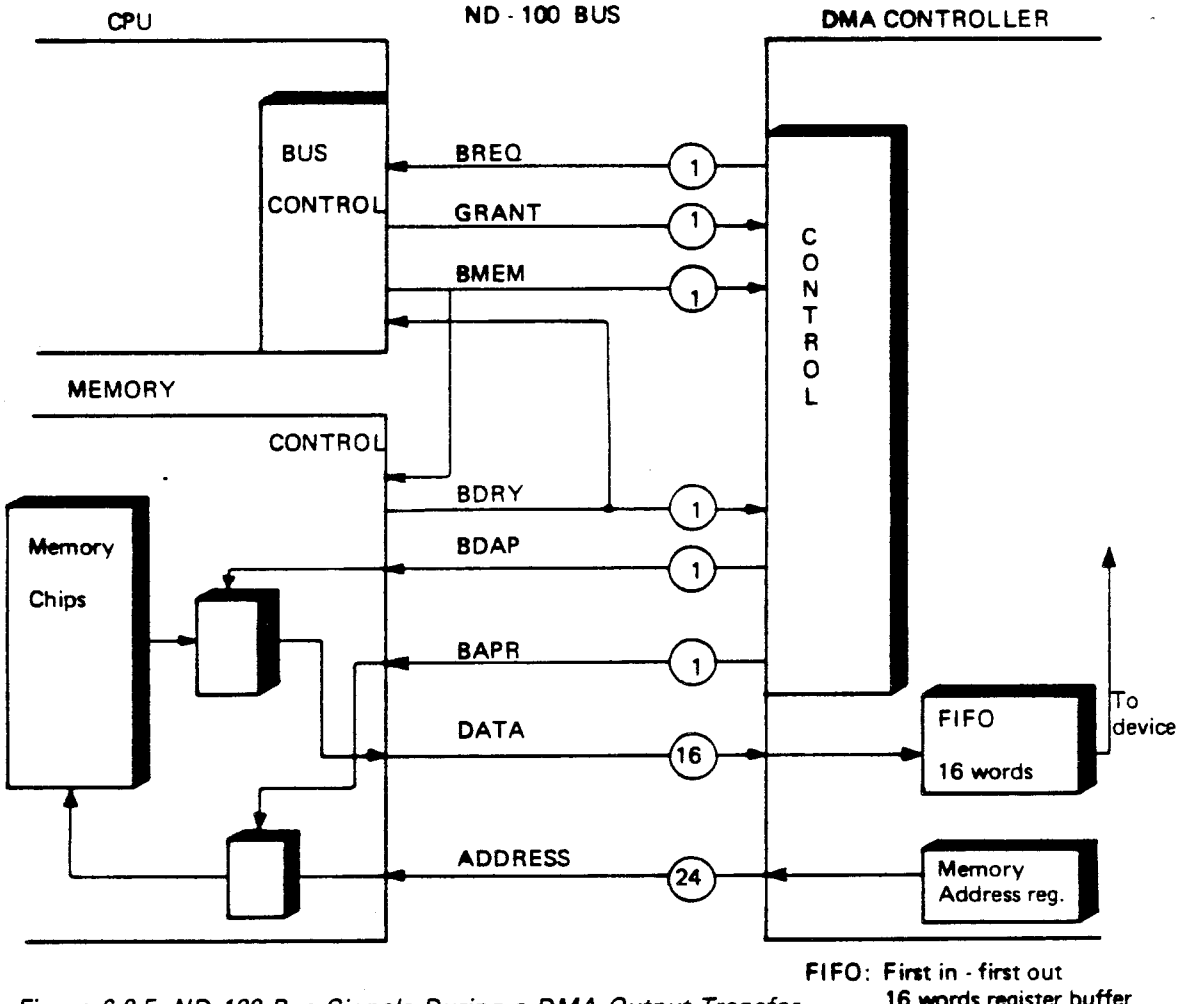


Figure 6.8.5: ND-100 Bus Signals During a DMA Output Transfer

'BDRY' — bus data ready, is then issued for termination of the cycle.

6.8.6 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.

6.9 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 number 300 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.

6.9.1 The Current Loop Interface

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

IOX 300:

Read input data (according to input control word setting). The last character input 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 active.

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.

6.9.2 The Real-Time Clock

The real-time clock on the CPU board has device number 10. The device register address range is 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 instruction is executed repeatedly, the counter will never be incremented, and no clock pulses will occur. This may affect the execution of operator's communication 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, ie., 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.

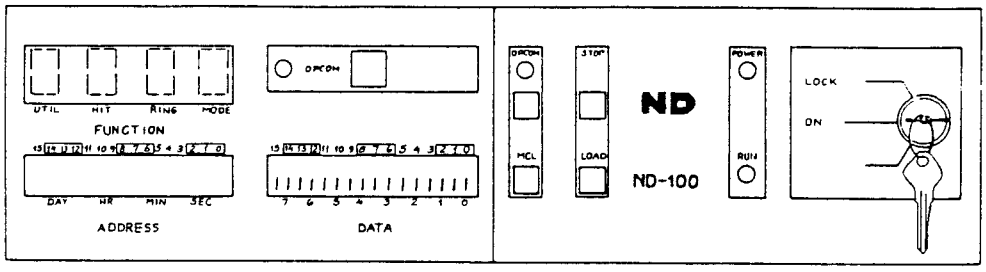
6.10 NORD-10/S MODULES USED IN ND-100

Even if the NORD-10/S modules have a different size, it is possible to use them in the ND-100 computer system.

On a special ND-100 module, two standard NORD-10/S PIO (programmed input/output) modules may be plugged in, and adapted to the ND-100 bus system. By help of this module, the wide range of PIO modules from NORD-10/S may be used in the ND-100.

7 OPERATOR'S INTERACTION

7.1 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 7.2.1), 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.

7.1.1 The Panel Lock Key

The Panel Lock Key has three positions:

1. LOCK

   In this position, the operator's panel control switches are disabled. This is the normal position for an operating machine. Main power is applied to the computer.

   Note: Automatic restart may be initiated after power failure only if the lock key is switched in this position.

2. ON

   In this position, the panel switches can be operated. Main power is applied to the computer.

3. STAND-BY

   In this position, the main power is disabled. Stand-by voltage is applied to memory and display. This position will not be present (or valid) on machines delivered after January 1980.

7.1.2 Status Indicators

POWER ON

Indicates that +5V is present in the rack.

RUN

Indicates that the CPU is running.

OPCOM

Indicates that the operator's 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 from the active program.

7.2 MICROPROGRAMMED OPERATOR'S COMMUNICATION

7.2.1 General Considerations

The ND-100 has a microprogram in the read-only memory for communication between the operator and the machine. This program is called MOPC (Microprogrammed Operator's Communication) and is used for operational control of the ND-100. It includes such functions as memory and register examine and deposit, breakpoint control, bootstrap loading, etc.

Whenever entered, MOPC will perform the necessary communication with the terminal connected to the current loop interface on the CPU printed circuit board. This terminal will be shared as output device between MOPC and other possible programs. As input device, MOPC will receive input from the terminal as long as the OPCOM lamp on the operator's panel is lit.

MOPC will never wait if the terminal is not ready for the transmission of characters. Instead, it will start executing the STOP routine or the running program. MOPC will then be dormant until next time it is entered, and continue with the tasks it had to postpone. The maximum time spent in MOPC is 20 µs. If MOPC does not have any activity to sustain on the terminal, it will use 6 µs every time it is entered.

The ND-100 operator's communication includes bootstrap programs and automatic hardware load from both character oriented devices and mass storage devices.

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

Characters legal in STOP or RUN:

Character:	*Use:*
0 - 7	Octal digits used to specify addresses and data.
A - Y	Letters used to specify commands and register names. Letters typed in succession are acted upon when CR (carriage return) or / is typed. Different letter combinations may have the same effect, because of a scrambling algorithm used to pack the letters.
@ or . (space)	All characters written before this character are ignored (break character).
<	Used to separate lower and upper bounds in dump commands.

/	Specifies memory or register examine.
↵ (carriage return)	Ends a line. Used to terminate commands or to perform a register or memory deposit function.
↑	This character will cause the address of the last examined memory address to be printed.
'escape'	Terminates the communcation between the CPU board terminal and MOPC. This character has no effect if the CPU is in STOP mode.

Characters only legal in STOP:

Character:	*Use:*
!	Start program in main memory command.
Z	Single instruction command.
$ or &	Bootstrap load command.
.	Breakpoint command.
''	Manual instruction command.
#	Start microprogrammed memory test.

All other characters are answered with a ?, and characters written before the erroneous character will be forgotten (as if 'space' had been typed).

7.2.2 Control Functions (Do not affect display)

7.2.2.1 System Control

7.2.2.1.1 *MASTER CLEAR*

When MACL μ 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 - 7.1.

7.2.2.1.2 *STOP*

When STOP μ 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.

7.2.2.1.3 *ALD LOAD*

In the following table the different columns signify:

ALD — Position of the ALD thumbwheel switch on the CPU module (refer to appendix D).

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 & 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

7.2.2.1.4 *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 pressed), and the switch selected ALD has bit 13 equal to '0'.

7.2.2.1.5 *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).

7.2.2.2 Program Execution

7.2.2.2.1 *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.

7.2.2.2.2 *Continue a Program*

!

If the octal number is omitted, the P register is used as start address, ie., 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).

7.2.2.2.3 *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.

7.2.2.2.4 *Instruction Breakpoint*

xxxxxx.

This command starts execution in the same semi-RUN mode as described in section 7.2.2.2.3. 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.

7.2.2.2.5 *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.

7.2.2.2.6 *Single I/O Instruction Function*

xxxxxxIO/

This function executes an IOX instruction with xxxxxx as device number. The output data is taken from the OPR register (see section 7.2.3.2.5). Returned data is printed after the slash, and not stored anywhere. No working registers are affected.

7.2.2.3 Miscellaneous Functions

7.2.2.3.1 *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

7.2.2.3.2 *Delete Entry*

When @ or . (space) is typed, all characters written before this character are ignored.

7.2.2.3.3 *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 is incremented each time carriage return is typed afterwards.

7.2.3 Monitor Functions (Also shown on Display)

7.2.3.1 Memory Functions

7.2.3.1.1 *Physical Examine Mode*

E ⌟

Subsequent examine will be in physical memory with a 24-bit address. Default mode after master clear.

7.2.3.1.2 *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.

7.2.3.1.3 *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 7.2.3.1.1). 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
```

7.2.3.1.4 *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.
```

7.2.3.1.5 *Deposit Rules*

Contents are only changed by zzzzzz ⏎ in STOP mode and by zzzzzzDEP ⏎ in STOP or RUN mode.

Contents are unchanged by ⏎ in STOP or RUN mode and zzzzzz ⏎ in RUN mode (? is answered).

7.2.3.1.6 *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 64 K 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.

7.2.3.2 Register Functions

7.2.3.2.1 *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.

7.2.3.2.2 *Register Deposit*

Format:

xxxxxx ⏎

After a register examine, the contents of the register may be changed by typing an octal number terminated by ⏎. If the CPU is running, 'DEP' must be written between the number and ⏎.

Examples:

A/ 123456 54321 ⏎ % Contents of A register on level 0
% are changed to 054321

7P/ 000044 55 ⏎ % Contents of P register on level 7
% are changed to 000055

7.2.3.2.3 *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.

7.2.3.2.4 *User Register*

U/

The last value written by TRR LMP, is selected as display source.

7.2.3.2.5 *Operator Panel 'Switches'*

OPR/

This selects a scratch register where a code to be read by TRA OPR can be deposited. Contents of OPR can be read and changed from the console.

7.2.3.3 Internal Register Functions

7.2.3.3.1 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 microprogram.
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
```

7.2.3.3.2 *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
```

7.2.3.3.3 *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.

7.2.3.3.4 *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.

7.2.4 Display Functions (Affect only display)

7.2.4.1 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, ie., 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 pushing the DECODE ADDRESS button on NORD-10/S.)

7.2.4.2 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 = DC	CPU Data is displayed
x = 1 = DD	DMA Data is displayed
x = 2 = AC	CPU Address is displayed
x = 3 = AD	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 'ACWR' shown in the FUNCTION field.

7.2.4.3 Display Activity

ACT/

With this display mode active levels, clock and indicator functions are displayed.

7.2.5 Bootstrap Loaders

The ND-100 has bootstrap loaders for both mass storage and character oriented devices. There are two different load formats:

— Binary format load
— Mass storage load

Octal load is not implemented in ND-100.

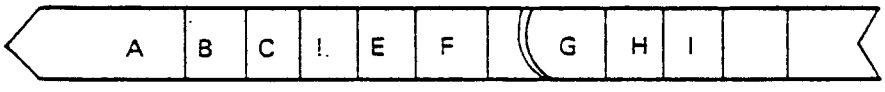
7.2.5.1 Binary Format Load

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), the program is started in the address previously found in the octal number (see above). If I is not a blank, 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, 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.

7.2.5.2 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'.

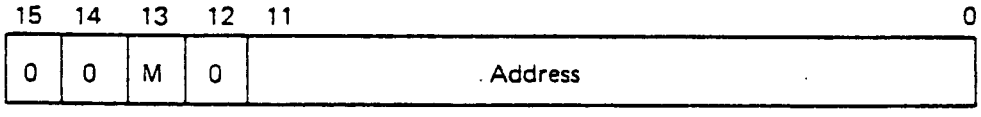
When loading from mass storage, 1 K words will be read from mass storage address 0 into main memory, starting in address 0. After a successful load, the CPU is started in main memory address 0.

The mass storage device must conform with either drum or disk programming specifications.

7.2.5.3 Automatic Load Descriptor

The ND-100 has a thumbwheel switch called the Automatic Load Descriptor (ALD) (CPU card). This switch selects a 16-bit value to use when the LOAD button is pushed or when a single $ or & is typed.

The 16-bit value has the following meaning:



M Mass Storage Load

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

If bit 13 is 0, binary load is taken from the device whose (lowest) address is found in bits 0-10.

7.3 THE DISPLAY

7.3.1 General

The optional display part of the panel is present if the machine has the memory management module installed. This module contains a display processor, in addition to the memory management system and cache memory. 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 (eg., in another room). It is practical therefore to have an 'OPCOM' button on this part of the panel.

More information about controlling the display processor is found in section 9.4 (Panel Processor Programming Specification) and in appendix E (Microprogram Panel Processor Communication).

7.3.2 The Different Display Functions

Figure 7.3.1 shows the normal activity on the display when the machine is running.

The DATA field displays information in binary or octal format (see section 7.2.4.1). 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 contents of the register are shown here.

Memory Contents

When a memory examine is done, the contents of the examined cell is shown here.

Bus Information

If the BUS command is given to display memory accesses on the ND-100 bus, the data present on the bus will be shown here and updated continually. When binary format is selected, the address field is used as extension for bits 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 are also shown here by giving the specific F command to MOPC (see section 7.2.4.1). 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, ie., is, how much time the machine spends on level 0 (idle). The more utility, the less the time spent on level 0 and the more segments on the display are lit up.

Example:

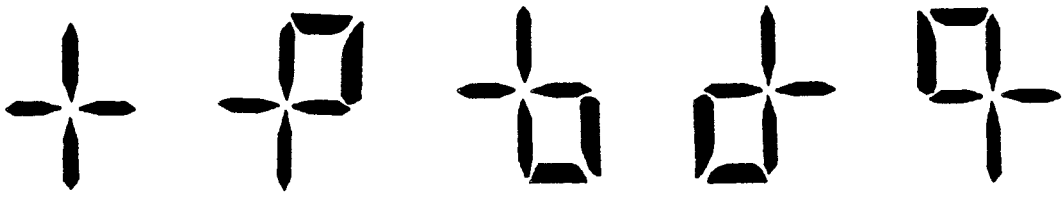
No Activity.

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 0 | 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, possibly 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

The 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.

8 ND-100 POWER SUPPLY

The small and the large cabinets have different power supplies. This section is therefore divided into two main chapters: one describing the power supply in the small cabinet and the other describing the large cabinet's power supply.

8.1 POWER IN THE SMALL CABINET

In the small cabinet you find three types of power: main power, standby power and floppy power. The main power and the standby power are housed in the same power unit located in the card crate. The floppy power is located behind the floppy drive(s).

Standby power holding time.

In case of power failure, the voltages from the standby power will be present for a minimum of 18 minutes with full load on both outputs (5V and 12V) and fully charged battery.

The table 8.1 below lists the powers used in the small cabinet. You find the powers' voltage, current and what they supply. The power supplies are used interchangeably in the small cabinet.

Main Power Manufacturer/Type	Voltage	Current	Used for
Philips/1746/03 Seem/P6003	5VDC	50A	CPU, Input/Output system, memory control logic.

Standby Power Manufacturer/Type	Voltage	Current	Used for
Philips/1746/03 Seem/P6003	5VDC 12VDC	7A 2A	Feeding refresh pulses to the MOS memory under a power break.

Floppy Power Manufacturer/Type	Voltage	Current	Used for
Philips/PE1743	5VDC —5VDC 24VDC	8A 0.3A 2A	Feeding floppy logic and floppy motors.
Power One/CP384-A	5VDC —5VDC 24VDC	9A 1.2A 2A	

Table 8.1.: Power Supplies in the Small ND-100 Cabinet

8.2 POWER IN THE LARGE CABINET

The large cabinets hold three power types plus a power control panel. The three power types are main power, standby power and floppy power. The power supplies and the power control are usually located at the top of the cabinet. See figure 8.1 for details.

Standby power holding time.

In case of power failure, the voltages from the standby power will be present for a minimum of 12 minutes with full load on both outputs (5V and 12V) and fully charged battery.

The table 8.2 below lists the powers used in the large cabinet. You find the powers' voltage, current and what they supply. The power supplies are used interchangeably in the large cabinet.

Main Power Manufacturer/Type	Voltage	Current	Used for
EMI ESP 271 Seem/P6010	5VDC	150A	CPU, Input/Output system, memory control logic.

Standby Power Manufacturer/Type	Voltage	Current	Used for
Philips/1759 Seem/P6005 P6007 P6011	5VDC 12VDC	25A 4A	Feeding refresh pulses to the MOS memory under a powr break.

Floppy Power Manufacturer/Type	Voltage	Current	Used for
Philips/PE1743	5VDC —5VDC 24VDC	8A 0.3A 5A	Feeding floppy logic and floppy motors.
Power One/CP384-A	5VDC —5VDC 24VDC	9A 1.2A 2A	

Power Control Panel Manufacturer/Type	Voltage	Current	Used for
EMI EMP 325 Philips PE 1018	12VDC	2A	Controlling and monitoring the other power supplies in the large cabinet. The unit can control 4 powers + 2 standby powers. The power control panel sends power status information to the computer.

Table 8.2.: Power Supplies in the Large ND-100 Cabinet

8.2.1 Power Location and Distribution in the Large Cabinet

Figure 8.1 illustrates where the power supplies are located and where the power is distributed.

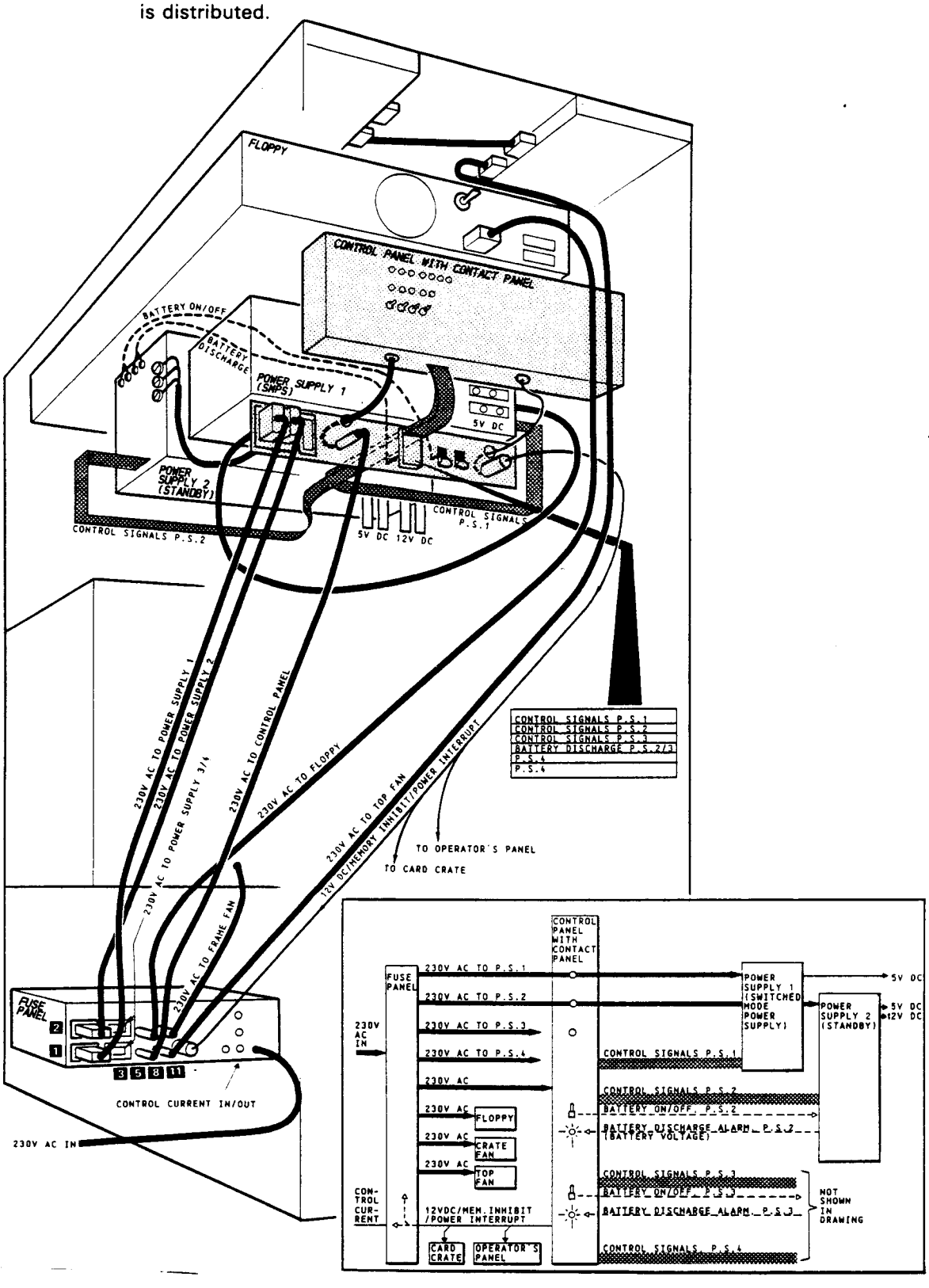


Figure 8.1: Power Location and Distribution in the Large Cabinet

8.3 POWER FAIL AND AUTOMATIC RESTART

8.3.1 General

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 take 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 exectuing programs to resume.

8.3.2 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 capacitor is charged to the peak voltage, and then allowed to discharge towards a sense level given by a Schmitt-trigger, which is set up to approximately half the peak value.

At normal operation, the capacitor is recharged 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 capacitor will discharge below the sense level, driving the power sense line to high and giving a false POWOK signal (refer to figure 8.2).

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.

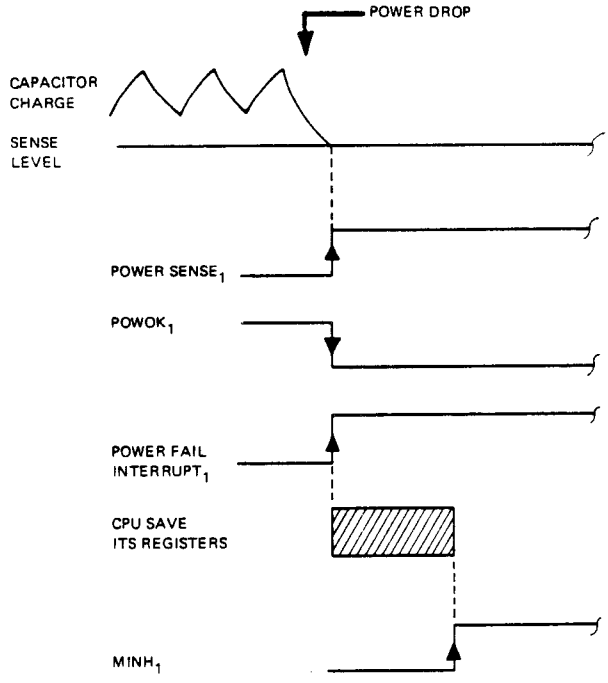


Figure 8.2: Power Fail Sequence

8.3.3 Automatic Restart

When power 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 8.3). 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 7). An automatic restart or an automatic load may then be performed, dependent upon the ALD thumbwheel setting which gives the adequate information about the operation. If the panel is not locked, the machine will enter the stop mode.

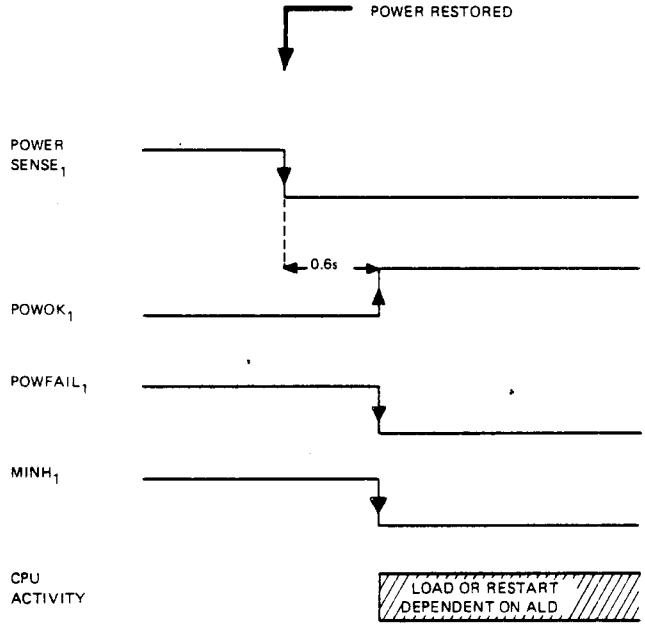


Figure 8.3: Power Up Sequence

9 MISCELLANEOUS

9.1 PRIVILEGED INSTRUCTIONS

9.1.1 General

The instructions termed privileged instructions are available only to:

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

9.1.2 Register Block Instructions

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

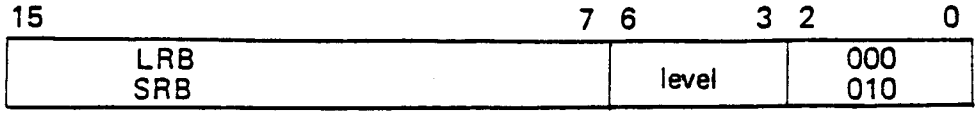
A register block always consists of the following registers in this sequence:

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 which the register block is read or into which it is written.

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 points to the instruction following SRB.

Affected:(EB),+ 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.

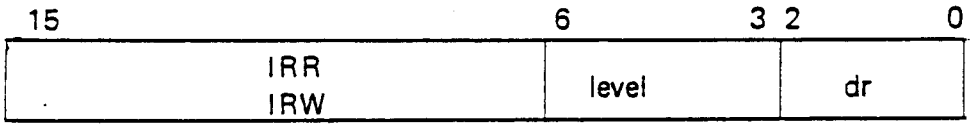
9.1.3 Inter-Level Register Instructions

In the ND-100 there are 16 complete sets of registers and status indicators, one set for each level.

The access to and from registers outside the current program level is achieved 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.

9.1.4 Accumulator Transfer Instructions

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

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

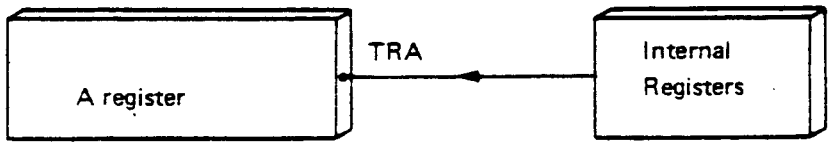
The internal registers controlled by these instructions are described in section 9.2.

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

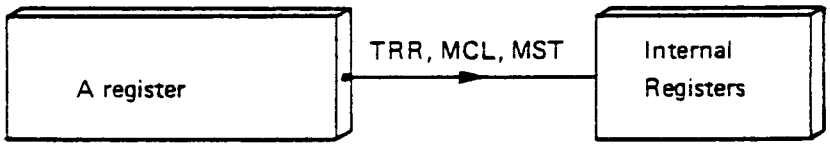
Format: TRR <register name>

The contents of the A register are copied into the register specfied by <register name>. The registers on which TRR may operate 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 on which MCL may operate 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 on which MST may operate 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			

9.1.5 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 category of privileged instructions.

9.1.5.1 Interrupt Control Instructions

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

The arrangement of the 16 program levels is as follows:

15	Reserved for extremely fast user 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 levels 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 section 2.9.

The PID register is controlled both by program and by hardware interrupts. At any time, the highest program level which has its corresponding bits set in both PIE and PID is running, ie., 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 lit 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, ie., the mechanisms for change of program levels are disabled. The computer will continue operation at the program level at which the IOF instruction was executed, ie., the PL register will remain unchanged. The interrupt indicator on the operator's display is reset by the IOF instructions. The IOF instruction is privileged.

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

IIE Internal Interrupt Enable
IIC Internal Interrupt code
PVL Previous level causing internal hardware status interrupt

In ND-100 there are possibilities for 2048 vectored input/output interrupts, where each physical input/output will have its own unique identification code and priority. The IDENT instruction is used to distinguish between vectored interrupts.

IDENT Identify vectored interrupt Code: 143 600

Format: IDENT <program level number>

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

The four instructions are:

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

The identification code of the input/output device is returned in bits 0 - 8 of the A register, with bits 9 - 15 all zeros.

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

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

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

9.1.5.2 Memory Management Control Instructions

A full description of memory management is given in chapter 3. The paging system is controlled by the following privileged instructions:

PON — Memory management on — Code: 150 410

Format: PON

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

The instruction executed after the PON instruction will use the page index table specified by PCR. 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 64 K, ie., the address following the POF instruction.

The CPU will be in an unrestricted mode without any hardware protection features, ie., 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 512 K 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.

9.1.5.3 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 lit. 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. The program level with the highest priority will be entered, which will normally then have a lower priority than the program level which executes the wait instruction. Therefore, the WAIT instruction means 'give up priority'.

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

Note that it is legal to specify WAIT followed by a number less than 400_8. This may be useful to detect in which location the program stopped. The WAIT instruction is privileged.

9.1.5.4 Monitor Call Instruction

MON Monitor Call Code: 153 000

Format: MON <number>

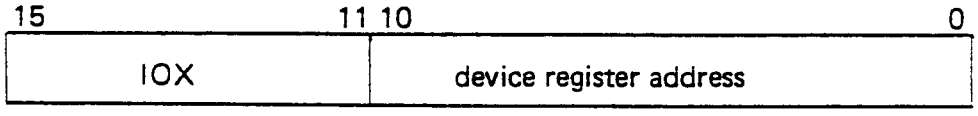
The instruction is used for monitor calls, and causes an internal interrupt to program level 14. The parameter <number> following MON must be specified between -200_8 and 177^8. This provides for 256 different monitor calls. This parameter, sign extended, is also loaded into the T register on program level 14.

9.1.6 Input/Output Control Instructions

9.1.6.1 IOX — Input/Output Execute

IOX Input/Output Execute Code: 164 000

Format: IOX <device register address>



All transfers between the ND-100 and external devices are controlled with 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.

9.1.6.2 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.

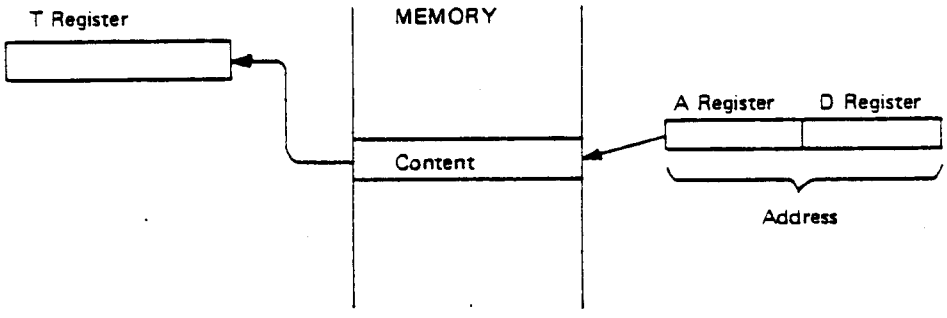
9.1.7 Examine and Deposit

9.1.7.1 Examine

EXAM Examine Code: 150 416

Format: EXAM

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

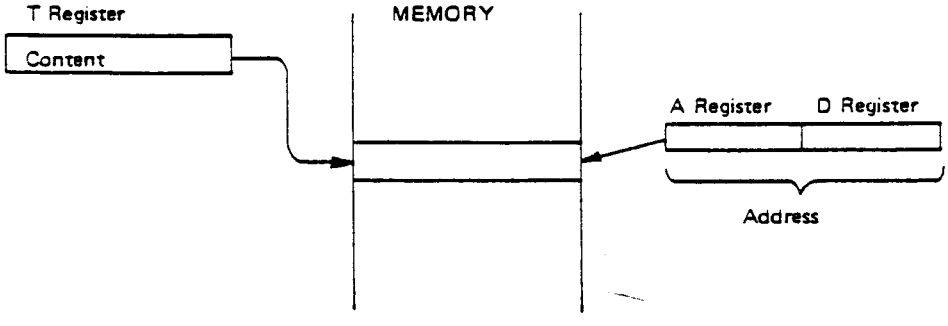


9.1.7.2 Deposit

DEPO Deposit Code: 150 417

Format: DEPO

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

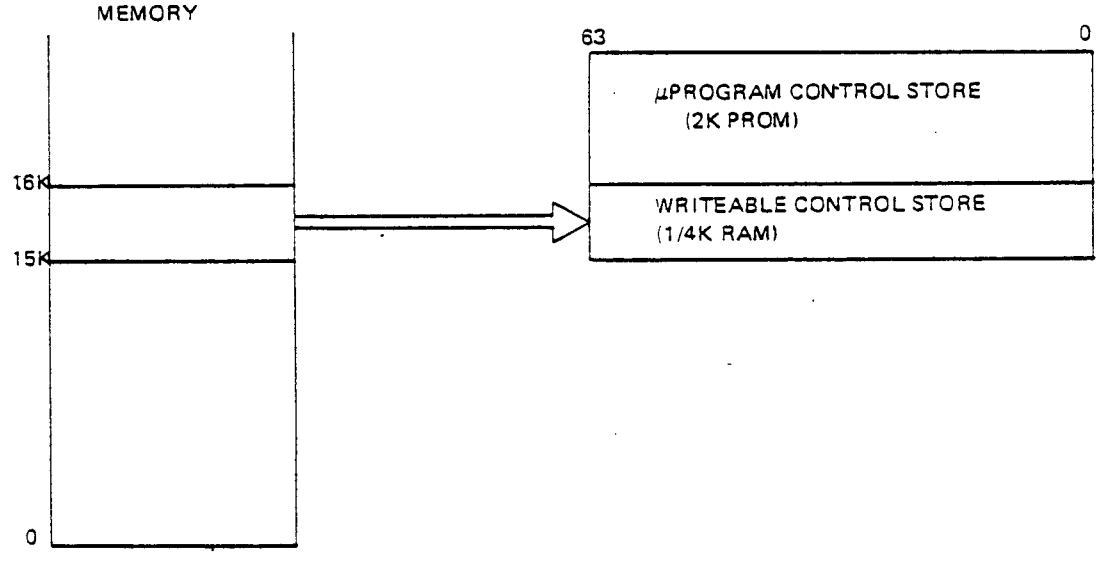


9.1.8. Load Writable Control Store

LWCS Load Writable Control Store Code: 143 500

Format: LWCS

By executing this instruction, the 256 words by 64 bits RAM writable control store will be loaded with the content or memory locations with address from 15 - 16 K in main memory. Micro program 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 Writable Control Store. Therefore, 1 K is needed from main memory.

9.2 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 chapter 7.

The following list shows all the internal registers in ND-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 microprogram. Also used by microprogram.
STS	1	Status Register. Bits 0-7 are level dependent and accessible from user programs, while bits 8-15 are system dependent and only accessible by system (TRA/TRR).
OPR	2	Operator's register. Implemented in firmware.
LMP	2	Display register. Implemented in firmware.
PGS		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
PGC	14	Paging control register read on specified level
PEA	15	Parity error address

9.3 WRITABLE CONTROL STORE

9.3.1 General

Refer to figure 9.3.1.

The illustration shows an instruction readout. The 12 bits ROM address goes into the microprogram control store, which is 2 K words by 64 bits ROM. The microprogram control store may be extended with a 256 words by 64 bits RAM, a so-called Writable Control Store (WCS). This writable control store enables the user to write his own microprogram, extending the ND-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.

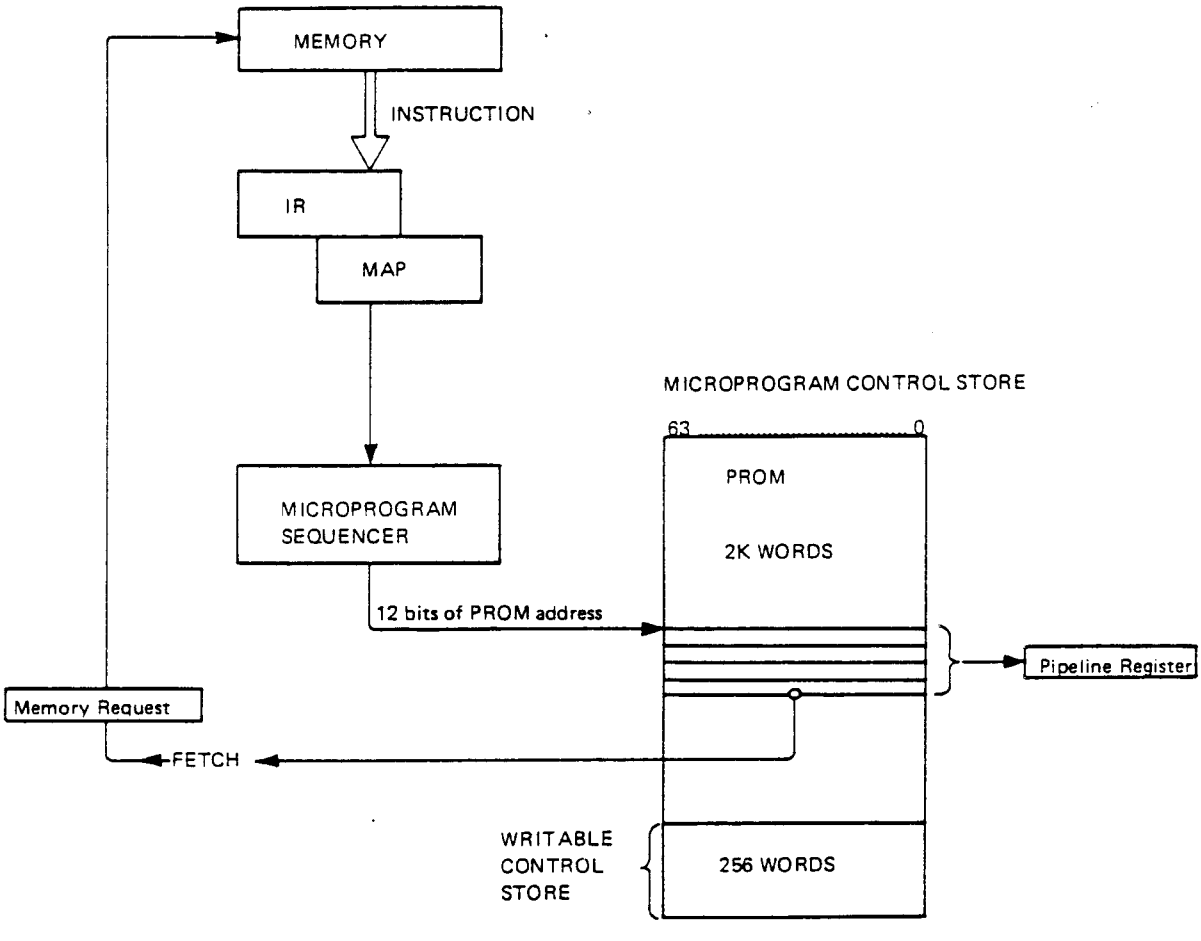


Figure 9.3.1: Instruction Readout

9.3.2 Writing Micro Code for Writable Control Store

In order to write micro code for the writable 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 microinstruction. This requires a good knowledge of the CPU architecture and the structure of the microprogram.

The symbolic microprogram will be most easily 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 ND-100 assembler. The binary representation of the WCS contents must be placed in main memory, in physical addresses 36000 to 37777.

9.3.3 Load the Writable Control Store

To load the writable control store, the privileged instructions LWCS (load writable control store) must be executed. The 256 words by 64 bits RAM (random access memory) writable control store will be loaded with the contents from memory locations with address from 15-16 K in main memory, as shown in figure 9.3.2.

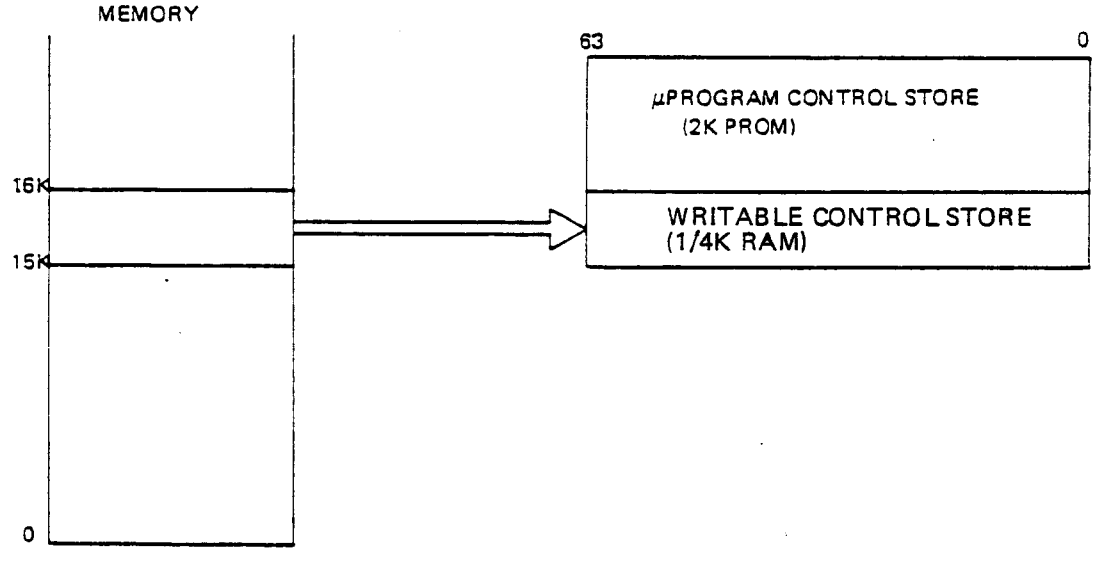


Figure 9.3.2: Loading the Writable Control Store

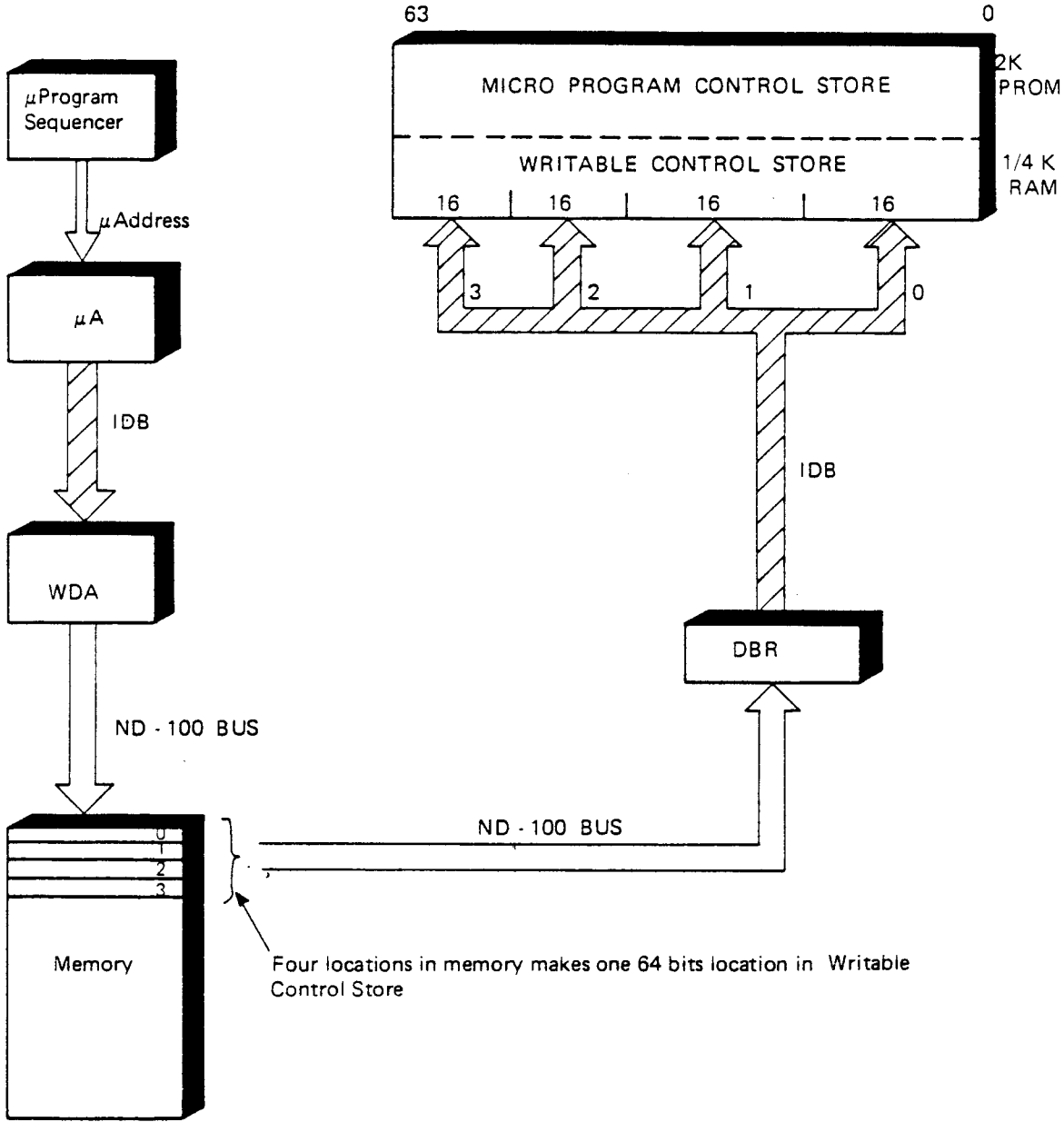


Figure 9.3.3: The Load Procedure of WCS

The LWCS instruction forces the microprogram to generate 1 K of read requests to memory. The addresses are generated by the microprogram sequencer, which enables them onto the internal data bus (IDB) by means of the microaddress register (μ A). Refer to figure 9.3.3.

The write data and address register (WDA) will enable the addresses onto the ND-100 bus. The first addressed location in main memory will go into the writable control store through the data bus read register (DBR). In WCS, these 16-bit words will be placed in bits 0-15 of the WCS word.

The next location in main memory will be placed as bits 16-31 of the WCS word. The third word will be placed as bits 32-47, and the fourth word as bits 48-63 of the WCS word. In this way one 64-bit location in WCS is made by four contignous locations in main memory.

Figure 9.3.4 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 be accessible. All microprogram addresses are then 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 inaccessible and LWCS must be executed in order to allow customer specified instructions to be executed.

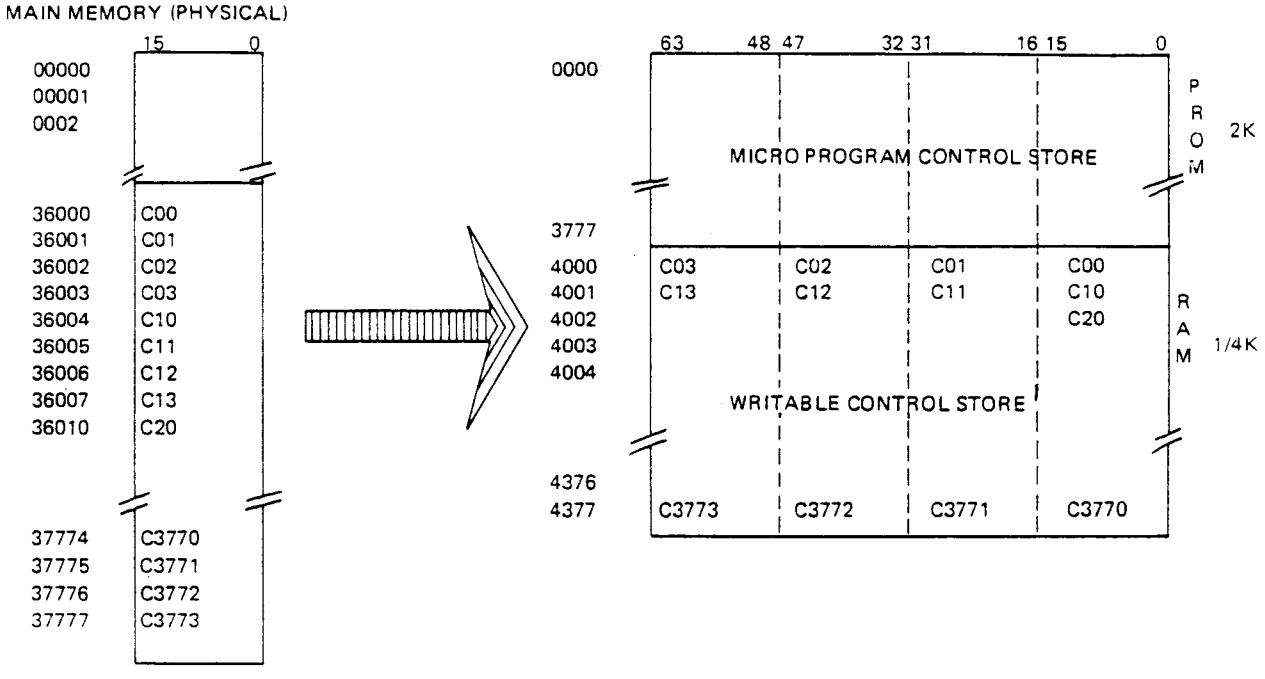


Figure 9.3.4: The addresses used in Main Mamory and Writable Control Store under a LWCS Instruction

9.3.4 Customer Specified Instructions

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

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

These 13 instructions have the following entry points in writable control store:

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

All microinstruction codes are available for new customer specified instructions.

More information about the WCS is found in the 'ND-100 Microprogramming Description' manual.

9.4 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

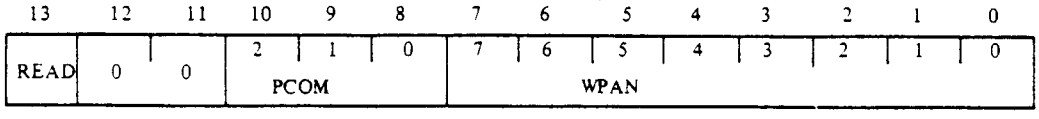
The following privileged instructions apply:

TRA 0 = Transfer PANS to A-register
TRR 0 = Transfer A-register to PANC

9.4.1 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 macro program.
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 macro program!

9.4.2 Panel Status Register

This register is used for handshake with macro program, together with possible input data.

The format is as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAN PRES	PANC $\overline{\text{FULL}}$	READ	COM RDY		2	1	0	7	6	5	4	3	2	1	0
					PCOM			RPAN							

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 command processed. 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.

9.4.3 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)

9.4.3.1 Message on Function Display

From the macro program 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. It can be rotated to display more than 4 characters. 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.

Displayable ASCII-codes are shown in appendix J.

9.4.3.2 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 it has itself lost track of it (at System Start or Power Fail).

The Calendar consists of a 32-bit counter which is incremented every second.

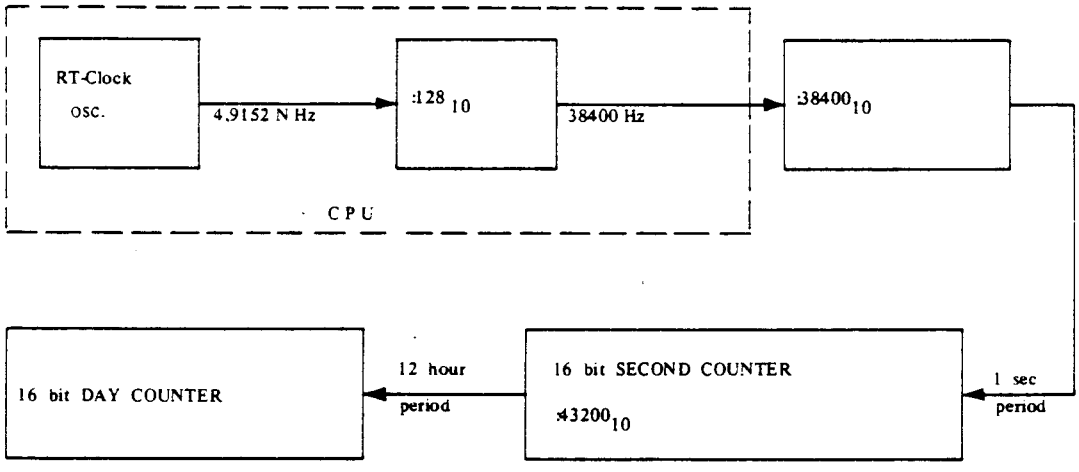


Figure 9.4.1: Calendar Counters

The second counter has shortened count length producing a carry every 12th hour. The Second and Day counters are both write and readable with the commands specified in section 9.4.3. 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

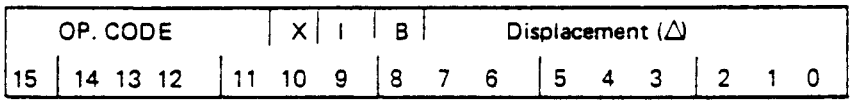
The *displayed* time accounts for leap years.

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 ⏎.

9.5 ND-100 INSTRUCTION CODES

Instruction formats and explanations found in the ND-100 Reference Manual.

MEMORY REFERENCE INSTRUCTIONS



Effective Address:

	000000	Address relative to P;	EL = P + Δ
,X	002000	Address relative to X;	EL=X+Δ
I	001000	Indirect address;	EL=(P+Δ)
,B	000400	Address relative to B;	EL=B+Δ

Δ= 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

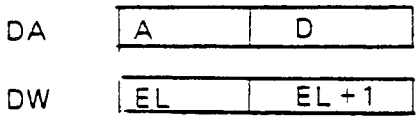
Load Instructions:

LDA	044000	Load A;	A:=(EL)
LDT	050000	Load T;	T:=(EL)
LDX	054000	Load X;	X:=(EL)

Arithmetical and Logical Instructions:

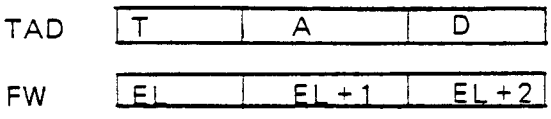
ADD	060000	Add to A (C, O and Q may also be affected);	A:=A+(EL)
SUB	064000	Subtract from A (C, O and Q may also be affected);	A:=A—(EL)
AND	070000	Logical AND to A;	A:=A (EL)
ORA	074000	Logical inclusive OR to A;	A:=A (EL)
MPY	120000	Multiply integer (O and Q may also be affected);	A:=A*(EL)

Double Word Instructions:



STD	020000	Store double word;	(DW):=AD
LDD	024000	Load double word;	AD:=(DW)

Floating Instructions:



STF	030000	Store floating accum.;	(FW):=TAD
LDF	034000	Load floating accum.;	TAD:=(FW)
FAD	100000	Add to floating accum. (C may also be affected);	TAD:=TAD+(FW)
FSB	104000	Subtract from floating accum. (C may also be affected);	TAD:=TAD—(FW)
FMU	110000	Multiply floating accum. (C may also be affected);	TAD:=TAD*(FW)
FDV	114000	Divide floating accum. (Z and C may also be affected);	TAD:=TAD/(FW)

Byte Instructions:

Addressing: EL=(T)+(X)/2

X=1: Right byte
X=0: Left byte

BFILL	140130	Byte fill
MOVB	140131	Move bytes
MOVBF	140132	Move bytes forward
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;	$(sr)=(sr)_0$

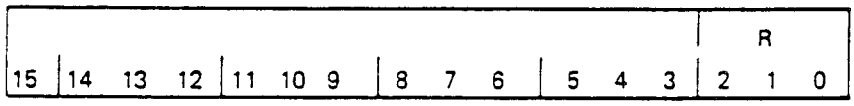
Combined Instructions:

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:

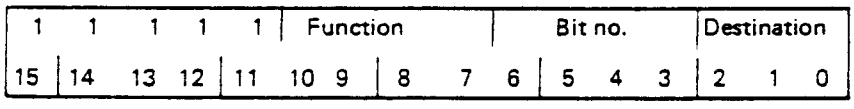
RMPY	141200	Multiply source with destination. Result in double accumulator	AD:=(sr)*(dr)
RDIV	141600	Divide double accumulator with source register. Quotient in A, remainder in D (AD = A*(sr)+D)	A:=AD/(sr)
MIX3	143200	(x)← [(A)-1] •3	

EXECUTE INSTRUCTION



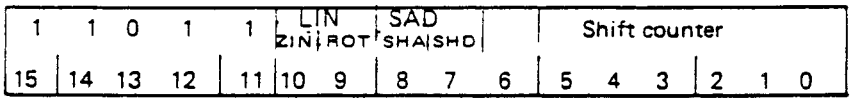
EXR	140600	Execute instruction found in specified register.

BIT INSTRUCTIONS



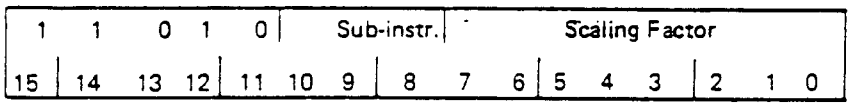
BSKP	175000	Skip next location if specified condition is true;	P:=P+1
BSET	174000	Set specified bit equal to specified condition;	
BSTA	176200	Store and clear K;	(B):=K; K:=0
BSTC	176000	Store complement and set K;	(B):=K_0; K:=1
BLDA	176600	Load K;	K:=(B)
BLDC	176400	Load bit complement to K;	K:=$(B)_0$
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	000000	Shift right; gives negative shift counter.

FLOATING CONVERSION



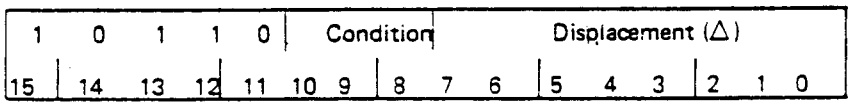
NLZ	151400	Convert the number in A to a floating number in FA.
DNZ	152000	Convert the floating number in FA to a fixed point number in A.
NLZ+20	151420	Integer to floating conversion.
DNZ—20	152360	Floating to integer conversion.

SEQUENCING INSTRUCTIONS

Unconditional Jump:

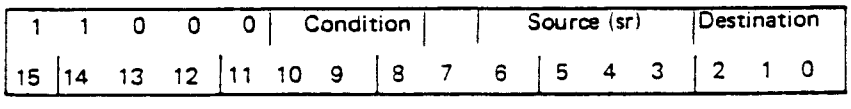
JMP	124000	Jump;	P=EL
JPL	134000	Jump to subroutine;	L=P; P=EL

Conditional Jump:



JAP	130000	Jump if A is positive; P= ± Δ if:	A ⩾ 0
JAN	130400	Jump if A is negative;	A<0
JAZ	131000	Jump if A is zero;	A=0
JAF	131400	Jump if A is nonzero;	A≠0
JXN	133400	Jump if X is negative;	X<0
JXZ	133000	Jump is X is zero;	X=0
JPC	132000	Increment X and jump if positive; X=X+1; P=P+Δ if	X ⩾ 0
JNC	132400	Increment X and jump if negative; X=X+1; P=P+Δ if	X<0

Skip Instructions:

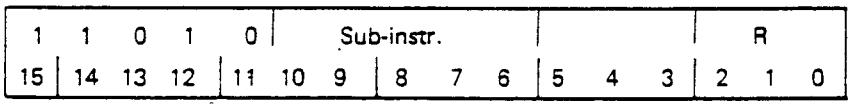


SKP	140000	Skip next location if specified condition is true;	P=P+1

Specified Condition:

EQL	000000	Equal to
UEQ	002000	Unequal to
GRE	001000	Signed greater or equal to, overflow OK
LST	003000	Signed less than, overflow OK
MLST	003400	Magnitude less than
MGRE	001400	Magnitude greater or equal to
IF	000000	May be used freely to
O	000000	obtain easy readability
GEQ	000400	Signed greater than or equal to, overflow **not** OK
LSS	002400	Signed less than, overflow **not** OK

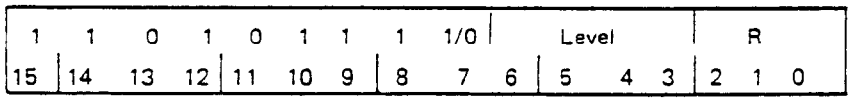
TRANSFER INSTRUCTIONS



Load Independent Instructions:

TRA	150000	Transfer specified internal register to A
TRR	150100	Transfer A to specified internal register
MCL	150200	Masked clear
MST	150300	Masked set

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
MON	153000	Monitor call instruction

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 writable 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 writable 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

IOX	164000	Transfer data to/from specified device
IOXT	150415	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	0	Level			0	0	0
							1	1				0	1	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

PHYSICAL MEMORY READ/WRITE

Read Instructions:

LDATX	143300	A:=(EL)
LDXTX	143301	X:=(EL)
LDDTX	143302	A:=(EL), D:=(EL + 1)
LDBTX	143303	B:=177000 V((EL) + (EL))

Write Instructions:

STATX	143304	(EL):=A
STZTX	143305	(EL):=0
STDTX	143306	(EL):=A, (EL +1):=D

INSTRUCTION IN THE CX-OPTION
(By expanding the microprogram PROM)

Decimal Instructions:

ADDD	140120	Add decimal
SUBD	140121	Subtract decimal
COMB	140122	Compare decimal
PACK	140124	Convert to packed decimal
UPACK	140125	Convert to unpacked decimal
SHDE	140126	Decimal shift

Stack Instructions:

INIT	140134	Initialize stack
ENTE	140135	Enter stack
LEAVE	140136	Leave stack
ELEAV	140137	Error leave stack

SINTRAN III Segment-change Instructions:

SETPT	140300	Set page tables
CLEPT	140301	Clear page tables
CLNREENT	140302	Clear non reentrant
CHREENTPAGES	140303	Change not reentrant pages
CLEPU	140304	Clear page tables, collect PGU information

Other CX Instructions:

MOVEW	1431xx	Move block of words
TSET	140123	Test and set
RDUS	140127	Read does not use cache

APPENDIX A

ND-100 MNEMONICS AND THEIR OCTAL VALUES

(ALPHABETICAL ORDER)

AAA	: 172400	DT	: 000006
AAB	: 172000	DX	: 000007
AAT	: 173000	ECCR	: 000015
AAX	: 173400	ELEAV	: 140137
ADC	: 001000	ENTR	: 140135
ADD	: 060000	EQL	: 000000
ADDD	: 140120	EXAM	: 150416
AD1	: 000400	EXIT	: 146142
ALD	: 000012	EXR	: 140600
AND	: 070000	FAD	: 100000
,B	: 000400	FDV	: 114000
BAC	: 000600	FMU	: 110000
BANC	: 177000	FSB	: 104000
BAND	: 177200	GEQ	: 000400
BCM	: 000400	GRE	: 001000
BLDA	: 176600	I	: 001000
BLDC	: 176400	IDENT	: 143600
BORA	: 177600	IF	: 000000
BORC	: 177400	IIC	: 000005
BSET	: 174000	IIE	: 000005
BSKP	: 175000	INIT	: 140134
BSTA	: 176200	IOF	: 150401
BSTC	: 176000	ION	: 150402
CCLR	: 000010	IOX	: 164000
CHREENT-		IOXT	: 150415
PAGES	: 140303	IRR	: 153600
CLD	: 000100	IRW	: 153400
CLEPT	: 140301	JAF	: 131400
CLEPU	: 140304	JAN	: 130400
CLNREENT	: 140302	JAP	: 130000
CM1	: 000200	JAZ	: 131000
CM2	: 000600	JMP	: 124000
COMD	: 140122	JNC	: 132400
COPY	: 146100	JPC	: 132000
CSR	: 000010	JPL	: 134000
DA	: 000005	JXN	: 133400
DB	: 000003	JXZ	: 133000
DD	: 000001	LBYT	: 142200
DEPO	: 150417	LCIL	: 000011
DL	: 000004	LDA	: 044000
DNZ	: 152000	LDATX	: 143300
DP	: 000002	LDBTX	: 143303

LDD	: 024000	ROT	: 001000
LDDTX	: 143302	RSUB	: 146600
LDF	: 034000	SA	: 000050
LDT	: 050000	SAA	: 170400
LDX	: 054000	SAB	: 170000
LDXTX	: 143301	SAD	: 154600
LEAVE	: 140136	SAT	: 171000
LIN	: 003000	SAX	: 171400
LMP	: 000002	SB	: 000030
LRB	: 152600	SBYT	: 142600
LSS	: 002400	SD	: 000010
LST	: 003000	SETPT	: 140300
LWCS	: 143500	SEX	: 150406
MCL	: 150200	SHA	: 154400
MGRE	: 001400	SHD	: 154200
MIN	: 040000	SHDE	: 140126
MIX3	: 143200	SHR	: 000200
MLST	: 003400	SHT	: 154000
MON	: 153000	SKP	: 140000
MOVEW	: 143100	SL	: 000040
MPY	: 120000	SP	: 000020
MST	: 150300	SRB	: 152402
NLZ	: 151400	SSC	: 000060
ONE	: 000200	SSK	: 000020
OPCOM	: 150400	SSM	: 000070
OPR	: 000002	SSO	: 000050
ORA	: 074000	SSQ	: 000040
PACK	: 140124	SSTG	: 000010
PCR	: 000003	SSZ	: 000030
PEA	: 000015	ST	: 000060
PES	: 000013	STA	: 004000
PGC	: 000014	STATX	: 143304
PGS	: 000003	STD	: 020000
PID	: 000006	STDTX	: 143306
PIE	: 000007	STF	: 030000
PIOF	: 150405	STS	: 000001
PION	: 150412	STT	: 010000
PL10	: 000004	STX	: 014000
PL11	: 000011	STZ	: 000000
PL12	: 000022	STZTX	: 143305
PL13	: 000043	SUB	: 064000
POF	: 150404	SUBD	: 140121
PON	: 150410	SWAP	: 144000
PVL	: 000004	SX	: 000070
RADD	: 146000	TRA	: 150000
RAND	: 144400	TRR	: 150100
RCLR	: 146100	TSET	: 140123
RDCR	: 146200	UCIL	: 000012
RDIV	: 141600	UEQ	: 002000
RDUS	: 140127	UPACK	: 140125
REX	: 150407	WAIT	: 151000
REXO	: 145000	,X	: 002000
RINC	: 146400	ZIN	: 002000
RMPY	: 141200	ZRO	: 000000
RORA	: 145400		

APPENDIX B

PROGRAMMING SPECIFICATION FOR SOME I/O DEVICES

B.1 ND-100 TERMINAL PROGRAMMING SPECIFICATIONS

Terminal Addresses:

The codes below are relevant for the first terminal (Terminal no. 0). The codes for the first 8 terminals are found by adding 10_8.N to the codes given. N = terminal number (0, 1, 2, ..., 7). For the next 8 terminals, the codes are found by adding $(1000 + 10(N - 10))_8$.N = terminal 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 for controlling 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 independent, and are selected by the same IOX instruction (group device number + 1). The contents 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 is 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 contents 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 terminal 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 subsegment to 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 by no peripheral device being 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 PAPER TAPE READER INTERFACE

Device Number (octal) : 0400
Register Address Range (octal) : 0400-0403
Interrupt Level : 12
Ident Code (octal) : 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 (Not used Device Number + 1)
IOX Read character. The same character may be read several times if desired (Device Number + 0)

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 each time IOX (device number + 3) (control word write) is used. It is then possible to test the data patch.

B.3 SPECIFICATION OF THE PAPER TAPE PUNCH INTERFACE

Device Number (octal) : 0410
Device Register Address Range (octal) : 0410-0413
Interrupt Level : 10
Ident Code (octal) : 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 a punch connected.

When bit 3 in the control word is one, the buffer register may be read back by IOX (Device Number + 0). If the interface is activated, it will become 'ready for transfer' after a while.

B.4 ND-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 register 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.

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 timeout
Bit 7: Not used

The following bits are commands to the floppy disk. 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 the 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)

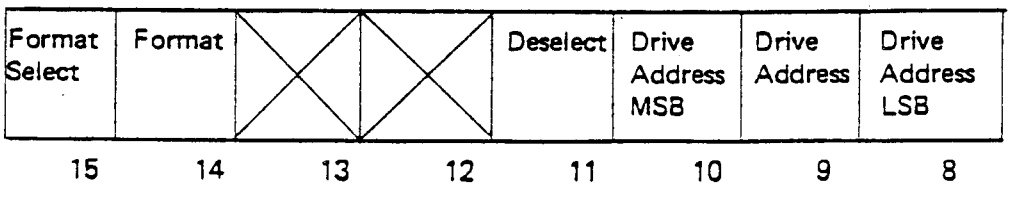
These are 2 instructions, depending on bit 0 in the A register.

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 sectors/track
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 the 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 incremented after each read/write command.

Note: This auto increment is not valid past the last sector of a track.

Auto increment	Sect. MSB	Sect.	Sect.	Sect.	Sect.	Sect.	Sect. LSB
15	14	13	12	11	10	9	8

Read Status Word IOX (Device Number + 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 Number + 1)

Only used for testing the interface without card reader.

Read Data IOX (Device Number + 0)

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 Number + 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 Number + 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 register
                    Exit
%
%
CRER2,
%                   Error, too many or too few columns read
                    Exit
CREOC,              JXZ * + 2; JMP CRER2
                    LDX (CBUF; EXIT AD1
)FILL
CBUF,               0
CPUF + 120/0
```

B.5 ND-100 10MB DISK CONTROLLER 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.

IOX 500 Read Memory Address *
The 24 bits memory address has to be read by two consecutive IOX 500. The least significant 16 bits are read by the first IOX 500. The most significant 8 bits are read by the next IOX 500.

IOX 501 Load Memory Address *
The 24 bits memory address has to be two consecutive IOX 501. The most signifiant 8 bits are loaded by the first IOX 501. The least significant 16 bits are loaded by the next IOX 501.

IOX 502 Read Sector Counter

IOX 503 Load Block Address

IOX 504 Read Status Register

IOX 505 Load Control Word

IOX 506 Read Block Address
This instruction is implemented for maintenance purposes only. By first loading a control word with bit 3 (test mode), the instruction will return the previously loaded block address to the A register.

IOX 507 Load Word Counter Register
The minimum number of words to be transferred is one sector, ie., 200_8 words, and the maximum number of words is one track, ie., 24 sectors.

*

The sequence is initialized by Master Clear, Programmed Clear, Read Status or Activate Device. One transfer is limited to maximum 16 address bits.

Control Word:

Word Counter Register

Bit 0: Enable interrup 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: N.A. (Not Assigned)

Bit 6: N.A.
Bit 7 : N.A.
Bit 8: N.A.

Bit 9 Unit select
Bit 10 Unit select
Bit 11 Device operation

Bit 12 Device operation
Bit 13 Marginal recovery (not used)
Bit 14 N.A. (Not Assigned)

Bit 15 Write format

Unit Select Code:

Bit 10	Bit 9	
0	0	Unit 0
0	1	Unit 1
1	0	Unit 2
1	1	Unit 3

Device Operation Code:

Bit 12	Bit 11	
0	0	Read transfer
0	1	Write transfer
1	0	Read parity
1	1	Compare

To format a disk, both 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 interrup level is 11 and the ident number for the first disk system is 1. The second disk system has ident number 5.

APPENDIX C

STANDARD ND-100 DEVICE REGISTER ADDRESSES AND IDENT CODES

In the following, only the most frequently used Device Names are listed.

Two Device Names may use the same Device Register Address range. In these cases, only the most common Device Name is listed.

Definitions:

Device Register Address = Device Number + Register Number.

Device Number = The lowest Device Register Address for each Device Name.

Register Number = Register Number within the Device (see the manual ND-100 Input/Output System, appendix B).

Interrupt Level 10 = Output Devices (PIO).
Interrupt Level 11 = Mass Storage Devices (DMA).
Interrupt Level 12 = Input Devices (PIO).
Interrupt Level 13 = Real Time Clock.

SINTRAN III Logical Device Number is a unique number for the Device Name.

Ident Code is a code sent from the device interface. The Ident Code tells that the CPU device asked for an interrupt. The Ident Code is unique for the Device Number on a specified Interrupt Level.

Device Reg. Address Range (octal)	*Interrupt Level*	*SINTRAN III Logical Device Numbers* (octal)*	*Ident Code (octal)*	*Device Name*
4- 7			4	Memory Parity N-12/N-42
10- 13	13		1	Real Time Clock 1
14- 17	13		2	Real Time Clock 2
20- 23	13		6	Real Time Clock 3
24- 27	13		7	External Interrupt
30- 33	12		16	NORD-50/1
34- 37	10		16	ACM 5
40- 43	10		15	ACM 1
44- 47	10		25	ACM 2
50- 53	10		40	ACM 3
54- 57	10		41	ACM 4
60- 77				NORD-50/1 Regs.
100-107	10/12	6	4	Sync. Modem 1
110-117	10/12	16	14	Sync. Modem 2
120-127	10/12	30	20	Sync. Modem 3
130-137	10/12	31	24	Sync. Modem 4
140-147	10/12	26	30	Sync. Modem 5
150-157	10/12	27	34	Sync. Modem 6
160-167	10/12		40	Sync. Modem 7
170-177	10/12		10	Sync. Modem 8
200-207	10/12	7	60	Terminal 17
210-217	10/12	17	61	Terminal 18
220-227	10/12	52	62	Terminal 19
230-237	10/12	53	63	Terminal 20
240-247	10/12	54	64	Terminal 21
250-257	10/12	55	65	Terminal 65
260-267	10/12	56	66	Terminal 23
270-277	10/12	57	67	Terminal 24
300-307**	10/12	1	1(120)***	Terminal 1
310-317**	10/12	11	5(121)***	Terminal 2/TET 15
320-327**	10/12	42	6(122)***	Terminal 3/TET 14
330-337**	10/12	43	7(123)***	Terminal 4/TET 13
340-347	10/12	44	44	Terminal 5/TET 12
350-357	10/12	45	45	Terminal 6/TET 11
360-367	10/12	46	46	Terminal 7/TET 10
370-377	10/12	47	47	Terminal 8/TET 9

* A complete list of SINTRAN III Logical Device Numbers is found in SINTRAN III Reference Manual (ND-60.128).

** Terminal no. 1 is implemented on the CPU module. Terminals with device numbers 317, 320-327 and 330-337 are normally not used.

*** Number in parentheses is valid for 4 current loop modules.

Device Reg. Address Range (octal)	*Interrupt Level*	*SINTRAN III Logical Device Numbers* (octal)*	*Ident Code (octal)*	*Device Name*
400- 403	12	2	2	Paper Tape Reader 1
404- 407	12	12	22	Paper Tape Reader 2
410- 413	10	3	2	Paper Tape Punch 1
414- 417	10	13	22	Paper Tape Punch 2
420- 423	12	4	3	Card Reader 1
424- 427	12	14	23	Card Reader 2
430- 433	10	5	3	Line Printer 1
434- 437	10	15	23	Line Printer 2
440- 443	10	10	11	Calcomp Plotter 1
444- 447	10	50	12	Card Punch 1
450- 453	10	35	21	Card Punch 3/Calc. 2
454- 457	10	51	13	Card Punch 2
460- 467	10/12		31	E & Pict. Syste. I/O
470- 477	12			Graphical Pen
500- 507	11		1	Disk System 1
510- 517	11		5	Disk System 2
520- 527	11		3	Mag. Tape 1
530- 537	11		7	Mag. Tape 2
540- 547	11		2	Drum 1
550- 557	11		6	Drum 2
560- 577	12/13	1006	156	HDLC HASP 1
600- 607	11	22	4	Versatec 1
610- 617	11		11	Core-to Core 1
620- 637	11	36	10	CDC I/O Link
640- 647	10/12	1040	124	Terminal 33
650- 657	10/12	1041	125	Terminal 34
660- 667	10/12	1042	126	Terminal 35
670- 677	10/12	1043	127	Terminal 36
700- 707	12	20	11	CATSY 1
710- 717	12	21	21	CATSY 2
720- 727	11	21	21	E & S Pict. Syst. DMA
730- 737	10		10	D/A- Converter
750- 753	13		5	BIG MPM LOG Module
754- 757	12		13	Process Input 5
760- 767	10-11-12-13		100	Test Card
770- 773	12		17	Dig. Reg. 1 Input
774- 777	10		17	Dig. Reg. 1 Output
1000-1003	12		26	Dig. Reg. 2 Input
1004-1007	10		26	Dig. Reg. 2 Output
1010-1013	12		27	Dig. Reg. 3 Input
1014-1017	10		27	Dig. Reg. 3 Output
1020-1023	12		43	Dig. Reg. 4 Input
1024-1027	10		43	Dig. Reg. 4 Output
1030-1033	12		116	NORD 50/2
1034				Watch Dog
1035				Process Output 1
1036				Process Output 2

Device Reg. Address Range (octal)	*Interrupt Level*	*SINTRAN III Logical Device Numbers (octal)*	*Ident Code (octal)*	*Device Name*
1037				Process Output 3
1040-1043	12		15	Process Input 1
1044-1047	12		25	Process Input 2
1050-1053	12		40	Process Input 3
1054-1057	12		12	Process Input 4
1060-1077				NORD-50/2 Reg.
1100-1107	10/12	1044	130	Terminal 37
1110-1117	10/12	1045	131	Terminal 38
1120-1127	10/12	1046	132	Terminal 39
1130-1137	10/12	1047	133	Terminal 40
1140-1147	10/12	1050	134	Terminal 41
1150-1157	10/12	1051	135	Terminal 42
1160-1167	10/12	1052	136	Terminal 43
1170-1177	10/12	1053	137	Terminal 44
1200-1207	10/12	70	70	Terminal 25
1210-1217	10/12	71	71	Terminal 26
1220-1227	10/12	72	72	Terminal 27
1230-1237	10/12	73	73	Terminal 28
1240-1247	10/12	74	74	Terminal 29/PHOTOS. 1
1250-1257	10/12	75	75	Terminal 30/PHOTOS. 2
1260-1267	10/12	76	76	Terminal 31/PHOTOS. 3
1270-1277	10/12	77	77	Terminal 32/PHOTOS. 4
1300-1307	10/12	60	50	Terminal 9
1310-1317	10/12	61	51	Terminal 10
1320-1327	10/12	62	52	Terminal 11
1330-1337	10/12	63	53	Terminal 12
1340-1347	10/12	64	54	Terminal 13
1350-1357	10/12	65	55	Terminal 14
1360-1367	10/12	66	56	Terminal 15
1370-1377	10/12	67	57	Terminal 16
1400-1407	10/12	1054	140	Terminal 45
1410-1417	10/12	1055	141	Terminal 46
1420-1427	10/12	1056	142	Terminal 47
1430-1437	10/12	1057	143	Terminal 48
1440-1443	12		101	A/D Converter 1
1444-1447	12		102	A/D Converter 2
1450-1453	12		103	A/D Converter 3
1454-1457	12		104	A/D Converter 4
1460-1463	12		105	A/D Converter 5
1464-1467	12		106	A/D Converter 6
1470-1473	12		107	A/D Converter 7
1474-1477	12		110	A/D Converter 8
1500-1507	10/12	1060	144	Terminal 49

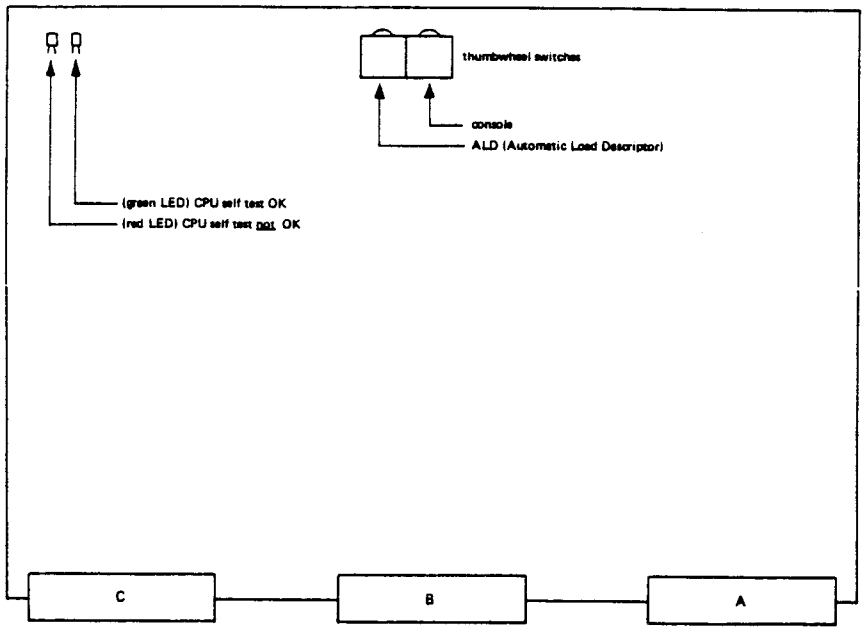
Device Reg. Address Range (octal)	*Interrupt Level*	*SINTRAN III Logical Device Numbers (octal)*	*Ident Code (octal)*	*Device Name*
1510-1517	10/12	1061	145	Terminal 50
1520-1527	10/12	1062	146	Terminal 51
1530-1537	10/12	1063	147	Terminal 52
1540-1547	11		17	Big Disk System 1
1550-1557	11		20	Big Disk System 2
1560-1567	11	1000-1002	21	Floppy Disk System (Unit 0, 1, 2)
1570-1577	11	1003-1005	22	Floppy Disk 2 (Unit 0, 1, 2)
1600-1603	11		14	Versatec 2
1604-1607				HDLC Remote Load 1
1610-1613				HDLC Remote Load 2
1614-1617				HDLC Remote Load 3
1620-1623				HDLC Remote Load 4
1624-1627				HDLC Remote Load 5
1630-1633				HDLC Remote Load 6
1634-1637				HDLC Remote Load 7
1640-1657	12/13		150	HDLC NORD-NET 1
1660-1667	12/13		151	HDLC NORD-NET 2
1700-1717	12/13		152	HDLC NORD-NET 3
1720-1737	12/13		153	HDLC NORD-NET 4
1740-1757	12/13		154	HDLC NORD-NET 5
1760-1777	12/13		155	HDLC NORD-NET 6
100000-100003				Bus Expander 0
100004-100007				Bus Expander 1
100010-100013				Bus Expander 2
100014-100017				Bus Expander 3
100020-100023				Bus Expander 4
100024-100027				Bus Expander 5
100030-100033				Bus Expander 6
100034-100037				Bus Expander 7
100115				ECCR
100200-100203	13/13		20	Bus Controller 1
100204-100207	13/13		21	Bus Controller 2
100210-100213	13/13		22	Bus Controller 3
100214-100217	13/13		23	Bus Controller 4
100220-100223	13/13		24	Bus Controller 5
100224-100227	13/13		25	Bus Controller 6
100230-100233	13/13		26	Bus Controller 7
100234-100237	13/13		27	Bus Controller 8
100240-100243	13/13		30	Bus Controller 9
100244-100247	13/13		31	Bus Controller 10
100250-100253	13/13		32	Bus Controller 11
100254-100257	13/13		33	Bus Controller 12
100260-100263	13/13		34	Bus Controller 13

Device Reg. Address Range (octal)	*Interrupt Level*	*SINTRAN III Local Device Numbers (octal)*	*Ident Code (octal)*	*Device Name*
100264-100267	13/13		35	Bus Controller 14
100270-100273	13/13		36	Bus Controller 15
100274-100277	13/13		37	Bus Controller 16
100300-100303	13/13		40	Bus Controller 17
100304-100307	13/13		41	Bus Controller 18
100310-100313	13/13		42	Bus Controller 19
100314-100317	13/13		43	Bus Controller 20
100320-100323	13/13		44	Bus Controller 21
100324-100327	13/13		45	Bus Controller 22
100330-100333	13/13		46	Bus Controller 23
100334-100337	13/13		47	Bus Controller 24
100340-100343	13/13		50	Bus Controller 25
100344-100347	13/13		51	Bus Controller 26
100350-100353	13/13		52	Bus Controller 27
100354-100357	13/13		53	Bus Controller 28
100360-100363	13/13		54	Bus Controller 29
100364-100367	13/13		55	Bus Controller 30
100370-100373	13/13		56	Bus Controller 31
100374-100377	13/13		57	Bus Controller 32

APPENDIX D

SWITCH SETTINGS FOR THE DIFFERENT ND-100 MODULES

D.1 SWITCHES ON THE CPU MODULE (3002)



D.1.1 ALD — Automatic Load Descriptor

ALD	i 12	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

Figure D.1.1: CPU Module (3002)

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.1.3 The Indicators (The Read and Green LEDs)

LED = Light Emitting Diode

The green LED is lit when the CPU is OK. When the red LED is lit this indicates a CPU error. See section 7 'Operator's Interaction' for more details about the two LEDs.

D.2 SWITCH AND INDICATORS ON THE 10MB DISK CONTROLLER (3004)

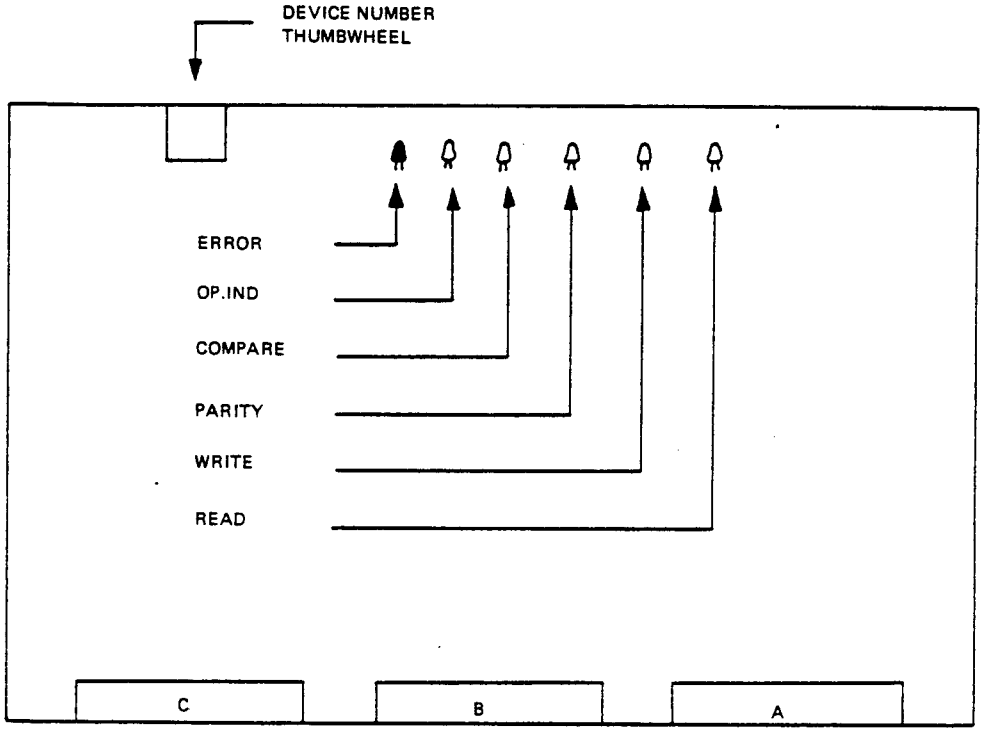


Figure D.2.1: 10 MB Disk Controller (3004)

D.2.1 Device Number Thumbwheel:

Pos:

0	— Disk system 1, Device no. 500, Ident no. 1
1	— Disk system 2, Device no. 510, Ident no. 5
2—15	— Not used

IOX address bit 15 active will inhibit this card.

D.2.2 **Indicators:**

Error	— Red LED, indicating different error conditions like timeout, address mismatch etc.
Op.Ind	— Yellow LED, indicating start or active set by control word bit 2 (activate).
Compare	— Yellow LED, indicating compare transfer.
Parity	— Yellow LED, indicating read parity.
Write	— Yellow LED, indicating write transfer.
Read	— Yellow LED, indicating read transfer.

D.3 SWITCHES AND INDICATORS ON THE DYNAMIC RAM (3005)

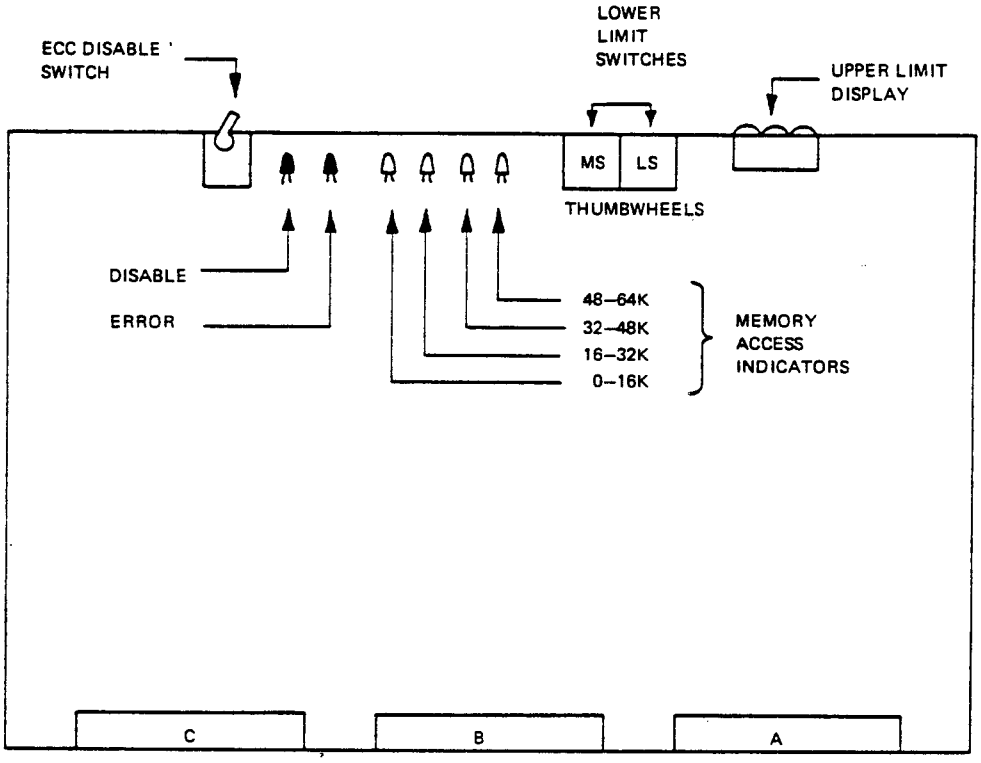


Figure D.3.1: Dynamic RAM (3005)

D.3.1 Lower Limit Switches

The lower limit switches, determining lower memory address for this memory card. This is a two digit hex thumbwheel, giving limit addresses in increments of 16 K units. The octal address will then be:

Octal address = 40.000 x limit.

The upper limit for this card is automatically displayed in the upper limit display in octal representation according to the actual memory size.

Example:

Lower limit	Size	Upper limit	Address range
0 0	32 K	0 2	0 — 32 K
0 0	64 K	0 4	0 — 64 K
0 3	32 K	0 5	48 — 80 K
0 3	64 K	0 7	48 — 112 K
3 4	64 K	4 0	448 — 512 K

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 64 K). The address ranges will be:

Pos.	Lower limit	Size	Upper limit	Address Range
12 (22)	88	64 K	04	0—64 K
11 (21)	88	64 K	10	64—128 K
10 (20)	88	64 K	14	. .
9 (19)	88	64 K	20	. .
8 (18)	88	64 K	24	. .
7 (17)	88	64 K	30	. .
6 (16)	88	64 K	34	384—448 K
5 (15)	88	64 K	40	448—512 K

(The numbers in parentheses are for a 20 position rack.)

etc.

D.3.2 Memory Access Indicators

There are four yellow leds indicating an access in the different memory blocks (16 K blocks) of this card.

If an access in block 0 occurs, the indicator labeled 0—16 is lit up. An access in the next block (No 1) will light up the indicator labeled 16—32 etc.

D.3.3 Error Check and Correction (ECC) Switch and Indicators

The error check and correction network may be disabled with the ECC disable switch. If disabled, the red indicator called Disable is lit.

When the ECC network detects a parity error, the red error indicator is activated. The disable switch or a master clear will reset this indicator.

D.4 SWITCH AND INDICATORS ON THE PERTEC MAGNETIC TAPE CONTROLLER (3006)

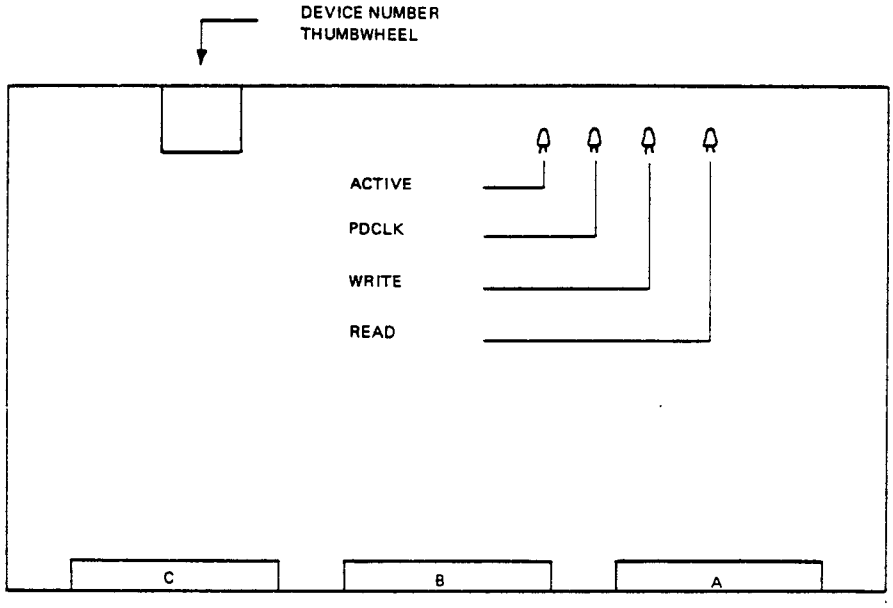


Figure D.4.1: Pertec Magnetic Tape Controller (3006)

D.4.1 Device Number Positions:

2	— Mag.tape 1, Device no. 520, ident.no. 3
3	— Mag.tape 2, Device no. 530, ident.no. 7
4—15	Not used

IOX address bit 15 active will inhibit this card.

D.4.2 Yellow LED Indicators:

Active	— Controller active
PD CLK	— A data transfer is in progress
Write	— Write to the tape
Read	— Read from the tape

D.5 SWITCH SETTING ON THE EUROLINE ADAPTER (3008)

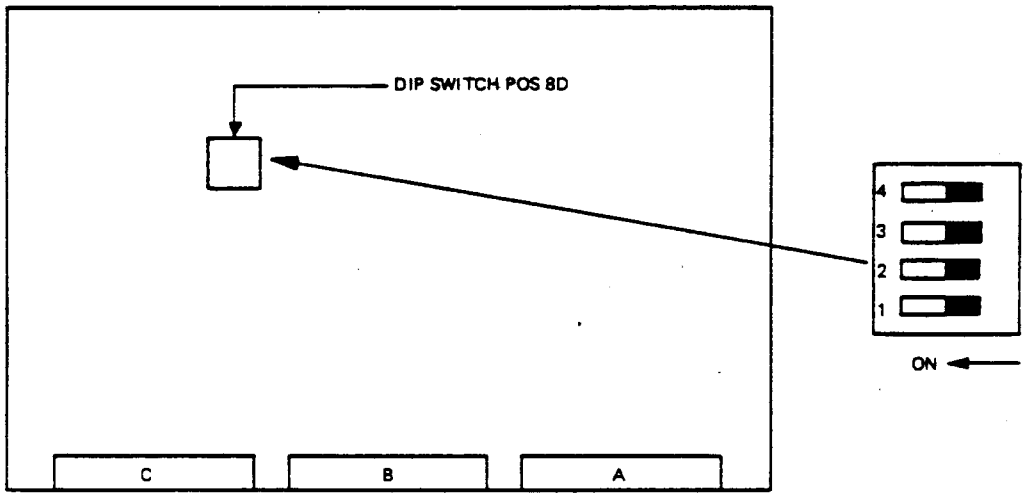


Figure D.5.1: Euroline Adapter (3008)

Switch 1 = off: 0 ⩽ device no. ⩽ 1777, 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.6 SWITCH ON THE LOCAL I/O BUS (3009)

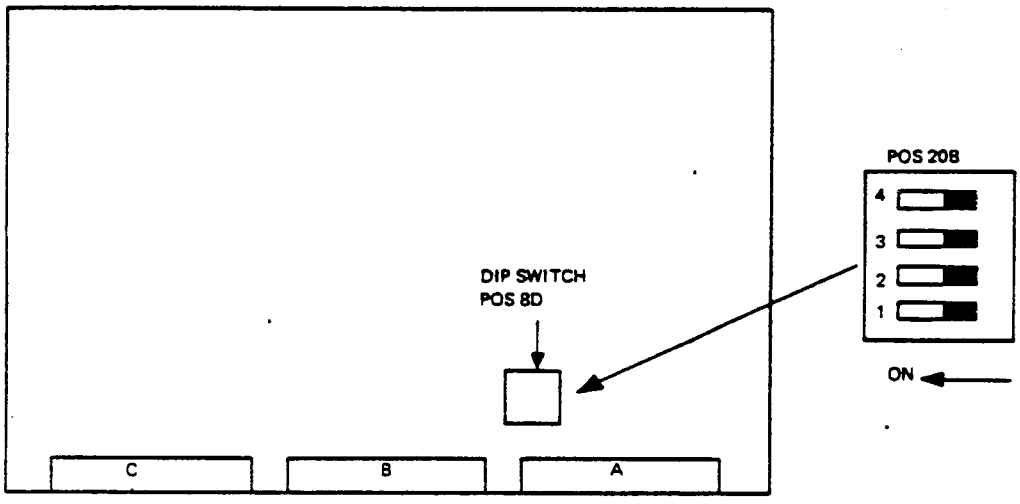


Figure D.6.1: Local I/O Bus (3009)

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 inhibit this card.

D.7 SWITCHES ON THE FLOPPY AND 4 TERMINALS MODULE (3010)

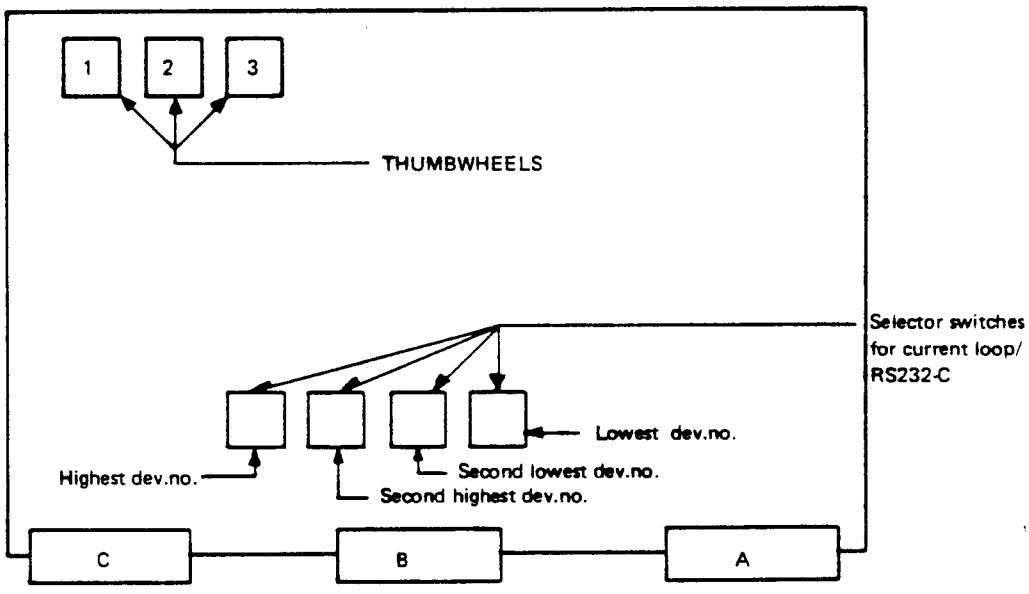


Figure D.7.1:Floppy and 4 Terminals Module (3010)

1 = Floppy disk system
2 = terminal group
3 = initial baud rate for terminals

D.7.1 Floppy Disk System

0 = floppy system no. 1 (Dev. no. 1560, IDENT = 21)
1 = floppy system no. 2 (Dev. no. 1570, IDENT = 22)

2-15 are unused, will answer on IOX 0-7.

D.7.2 **Terminal Group:**

Each group consists of 4 terminals with consecutive device numbers and ident codes.

0 = terminals 1-4	(dev. no. 300-330	IDENT 120-123)
1 = terminals 5-8	(dev. no. 340-370	IDENT 44-47)
2 = terminals 9-12	(dev. no. 1300-1330	IDENT 50-53)
3 = terminals 13-16	(dev. no. 1340-1370	IDENT 54-57)
4 = terminals 33-36	(dev. no. 640-670	IDENT 124-127)
5 = terminals 37-40	(dev. no. 1100-1130	IDENT 130-133)
6 = terminals 41-44	(dev. no. 1140-1170	IDENT 134-137)
7 = terminals 45-48	(dev. no. 1400-1430	IDENT 140-143)
8 = terminals 49-52	(dev. no. 1500-1530	IDENT 144-147)
9 = terminals 53-56	(dev. no. 1640-1670	IDENT 150-153)
10 = terminals 57-60	(dev. no. 1700-1730	IDENT 154-157)
11 = terminals 61-64	(dev. no. 1740-1770	IDENT 160-163)
12 = async. modem 1-4	(dev. no. 200-230	IDENT 60-63)
13 = async. modem 5-8	(dev. no. 240-270	IDENT 64-67)
14 = async. modem 9-12	(dev. no. 1200-1230	IDENT 70-73)
15 = async. modem 13-16	(dev. no. 1240-1270	IDENT 74-77)

D.7.3 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

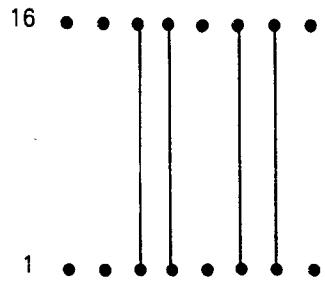
Selector switches for current loop/RS232-C.
Switch set to 0 selects current loop.
Switch set to 1 selects RS232-C.
Switch settings under Terminal Group and Initial baud rate for terminals are also valid for the 8 terminal modules (3013).

This description is correct for the 2 position switch.

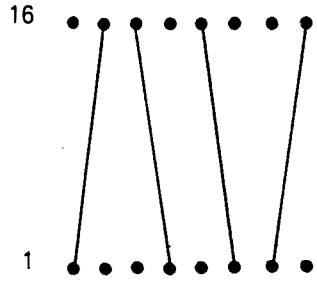
If a hexadecimal switch is used:

0 = current loop
F = RS232-C

If component houses are used:



Selects current loop



Selects RS 232-C

D.8 SWITCH AND INDICATOR ON THE MEMORY MANAGEMENT (3012)

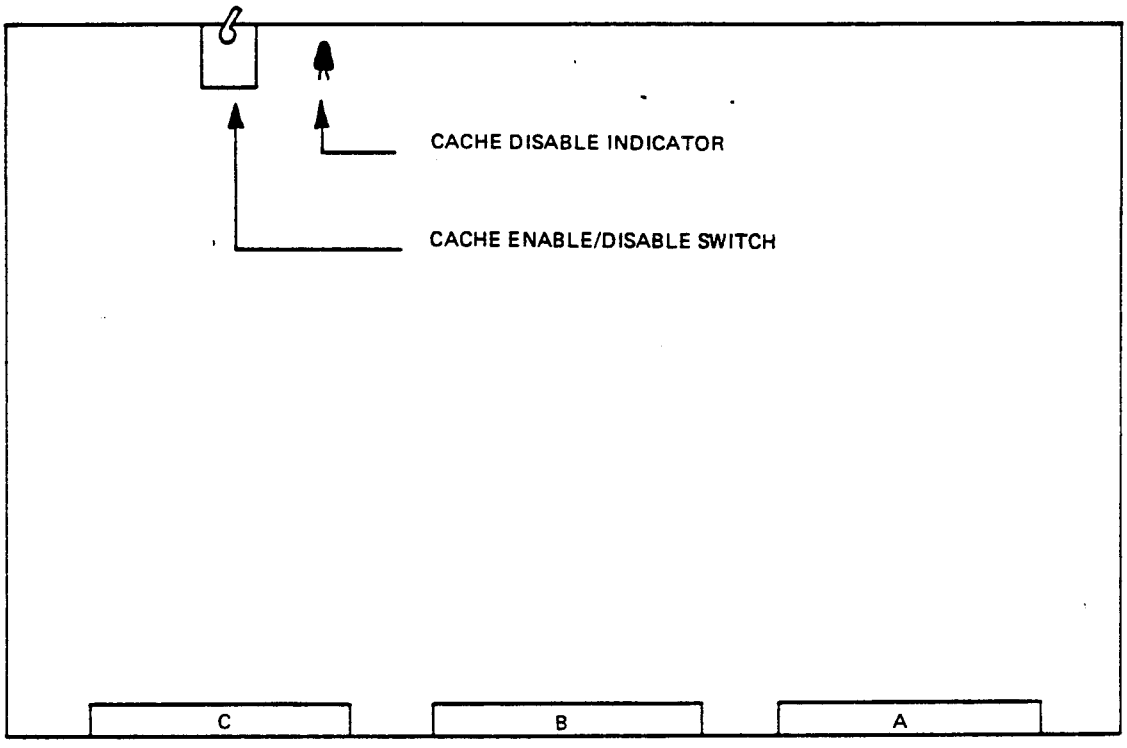


Figure D.8.1: Memory Management (3012)

D.8.1 Cache Memory Enable/Disable Switch and Indicator

This switch will enable or disable the cache memory, which is an option on the memory management card. If the cache memory is disabled, the red LED will be lit up.

D.9 SWITCHES ON THE 8 TERMINAL INTERFACE (3013)

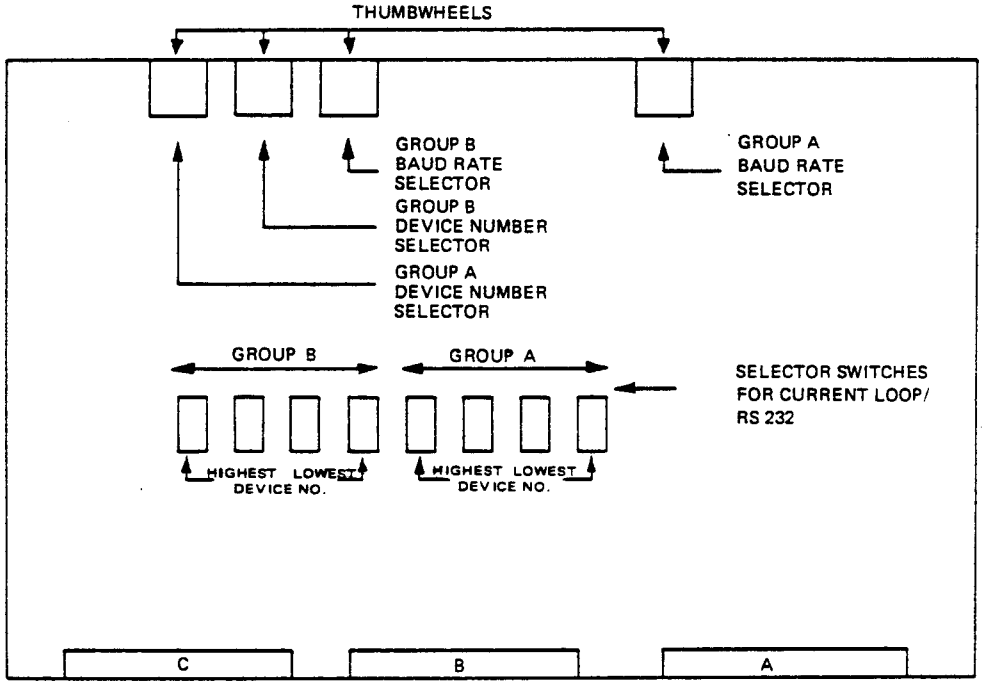


Figure D.9.1: 8 Terminal Interface (3013)

D.9.1 Device Numbers:

Terminals are divided into groups of four terminals. An 8-terminal card consists of two groups, Group A and B. Each group consists of four terminals with consecutive device numbers and ident.codes.

Thumbwheel position:

0 = terminals 1—4	(dev. no. 300—330	IDENT 120—123)
1 = terminals 5—8	(dev. no. 340—370	IDENT 44—47)
2 = terminals 9—12	(dev. no. 1300—1330	IDENT 50—53)
3 = terminals 13—16	(dev. no. 1340—1370	IDENT 54—57)
4 = terminals 33—36	(dev. no. 640—670	IDENT 124—127)
5 = terminals 37—40	(dev. no. 1100—1130	IDENT 130—133)
6 = terminals 41—44	(dev. no. 1140—1170	IDENT 134—137)
7 = terminals 45—48	(dev. no. 1400—1430	IDENT 140—143)
8 = terminals 49—52	(dev. no. 1500—1530	IDENT 144—147)
9 = terminals 53—56	(dev. no. 1640—1670	IDENT 150—153)
10 = terminals 57—60	(dev. no. 1700—1730	IDENT 154—157)
11 = terminals 61—64	(dev. no. 1740—1770	IDENT 160—163)
12 = terminals 17—20	(dev. no. 200—230	IDENT 60—63)
13 = terminals 21—24	(dev. no. 240—270	IDENT 64—67)
14 = terminals 25—28	(dev. no. 1200—1230	IDENT 70—73)
15 = terminals 29—32	(dev. no. 1240—1270	IDENT 74—77)

D.9.2 Baud Rates:

Thumbwheel position:

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

D.9.3 Current Loop/RS232 Selector

Switch set to 0 selects current loop.
Switch set to 1 selects RS232—C.

D.10 SWITCHES ON THE HDLC + AUTOLOAD CONTROLLER (3015)

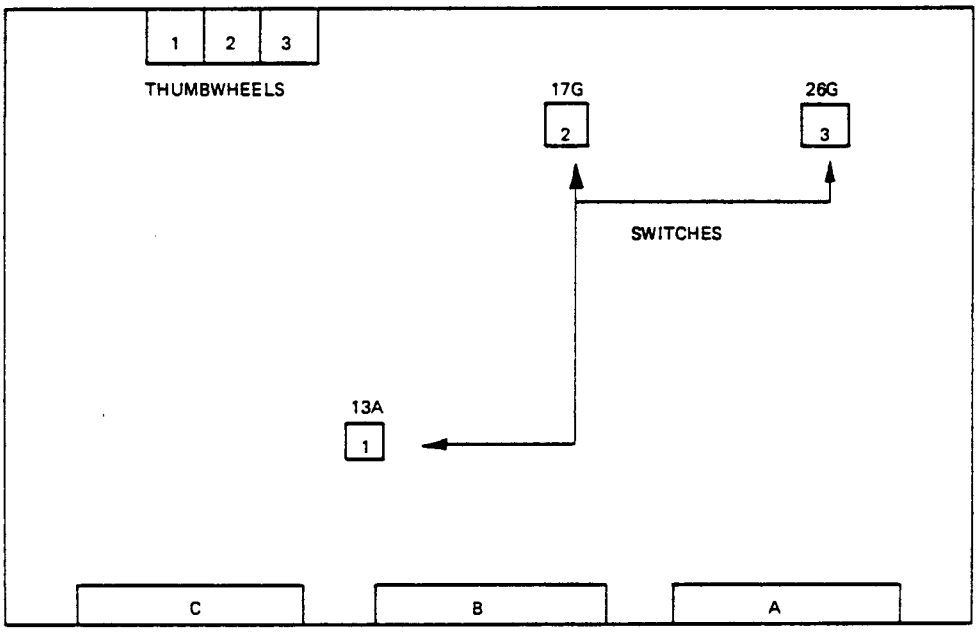


Figure D.10.1: HDLC Controller (3015)

D.10.1 Thumbwheels:

BAUD RATE: THUMBWHEEL NO. 1

Position:

0 =	307 K baud
3 =	1.2 K baud
6 =	9.6 K baud
7 =	38.4 K baud
8 =	153.6 K baud
9 =	76.8 K baud
11 =	19.2 K baud
13 =	4.8 K baud
14 =	2.4 K baud

AUTOLOAD DEVICE NUMBER: THUMBWHEEL NO. 2.

	THUMBWHEEL POSITION	DEVICE NUMBERS	
	(0	1600)	NOT ALLOWED
AUTOLOAD 1	1	1604	NO IDENT
AUTOLOAD 2	2	1610	NO IDENT
AUTOLOAD 3	3	1614	NO IDENT
AUTOLOAD 4	4	1620	NO IDENT
AUTOLOAD 5	5	1624	NO IDENT
AUTOLOAD 6	6	1630	NO IDENT
AUTOLOAD 7	7	1634	NO IDENT
	8—15		NOT ALLOWED

HDLC DEVICE NUMBER: THUMBWHEELS NO. 3

	THUMBWHEEL POSITION	DEVICE NUMBERS	IDENT CODE
HDLC 1	0	1640	140
HDLC 2	1	1660	151
HDLC 3	2	1700	152
HDLC 4	3	1720	153
HDLC 5	4	1740	154
HDLC 6	5	1760	155
HASP 1	6	560	156
HASP 2	7	620	157
HASP 3	8	700	160
HASP 4	9	720	161
HASP 5	10	1500	162
HASP 6	11	1520	163

D.10.2 **Switches:**

SWITCH ON = 1, SWITCH OFF = 0

SWITCHES IN POSITION 13A

4 3 2 1

x x 0 0 X 21

x x 0 1 Computer link

x x 1 0 V 35

x x 1 1 V 24/x 21 bis

SWITCHES IN POSITION 17G

4 3 2 1

0 Disable MCL

0 Disable LOAD

0 Disable RML

0 Disable CLOAD

SWITCHES IN POSITION 26G

4 3 2 1

x x 0 x Disable Abaud rate osc.

x 0 x x Disable main clock

D.11 SWITCH AND INDICATORS ON THE ECC DISK CONTROLLER (3018 AND 3019)

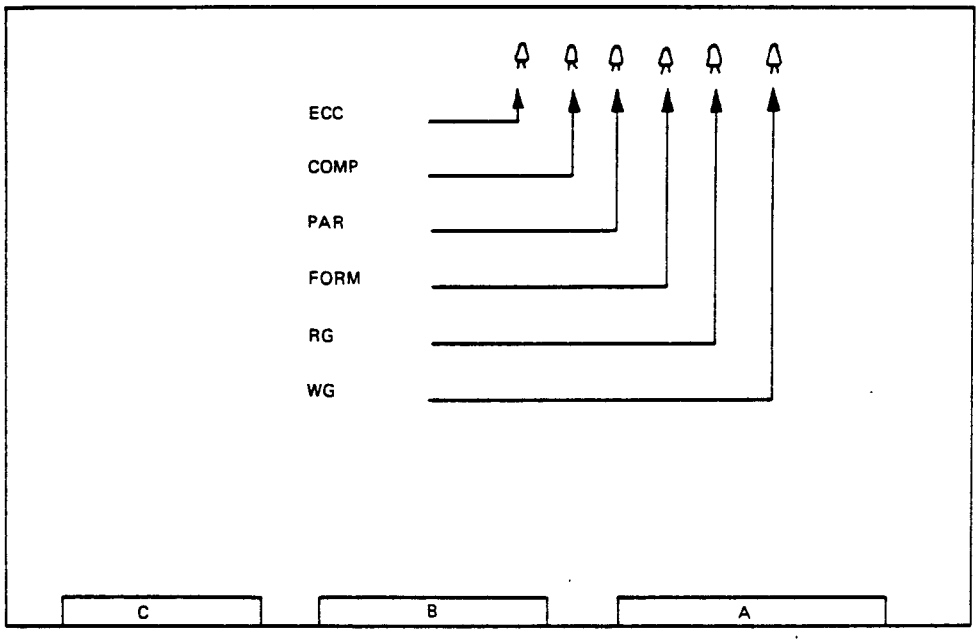


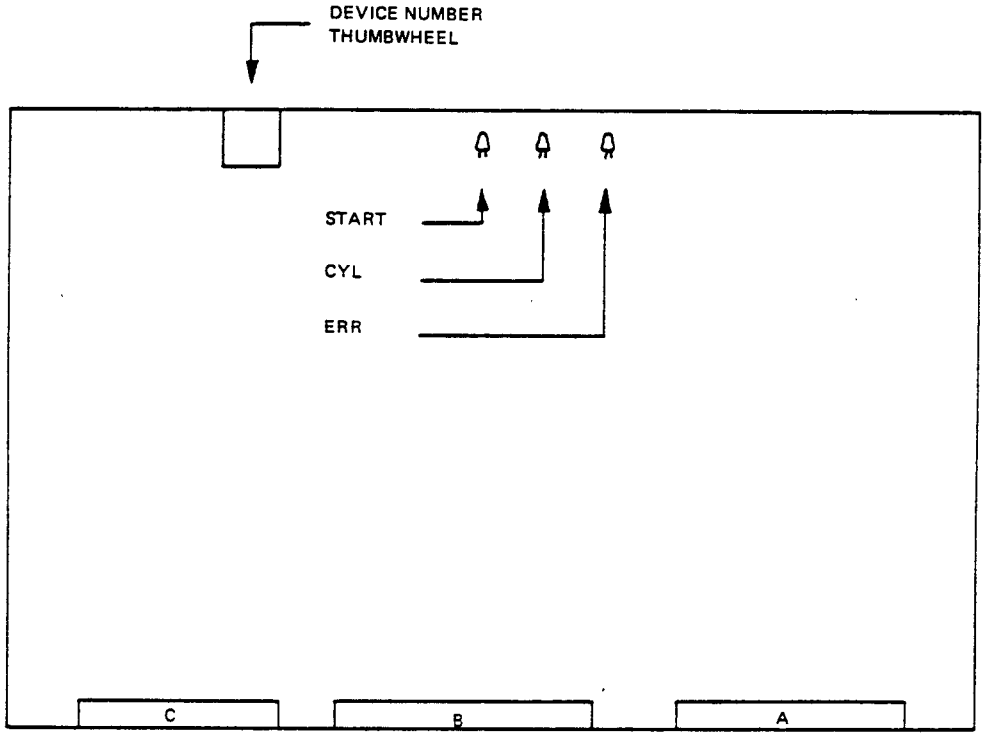
Figure D.11.1: SMD Control (3018)

D.11.1 SMD Control (3018)

All indicators are yellow LEDs with the following meaning:

ECC	—	ECC OPERATION
COMP	—	COMPARE TRANSFER
PAR	—	READ PARITY
FORM	—	WRITE FORMAT
RG	—	READ GATE
WG	—	WRITE GATE

D.11.2 SMD Data (3019)



Figur D.11.2: SMD Data (3019)

Device Number Thumbwheel Position:

8 — Big Disk System 1, Device No. 1540, Ident.No. 17
9 — Big Disk System 2, Device No. 1550, Ident.No. 20
0—7 and 10—15 are unused.

Indicators:

START — Yellow LED, Indicating Controller Active.
CYL — Yellow LED, Indicating on Cylinder.
ERR — Red LED, Indicating an Inclusive OR of Errors.

D.12 SWITCH AND INDICATORS ON THE STC MAGTAPE CONTROLLER (3020)

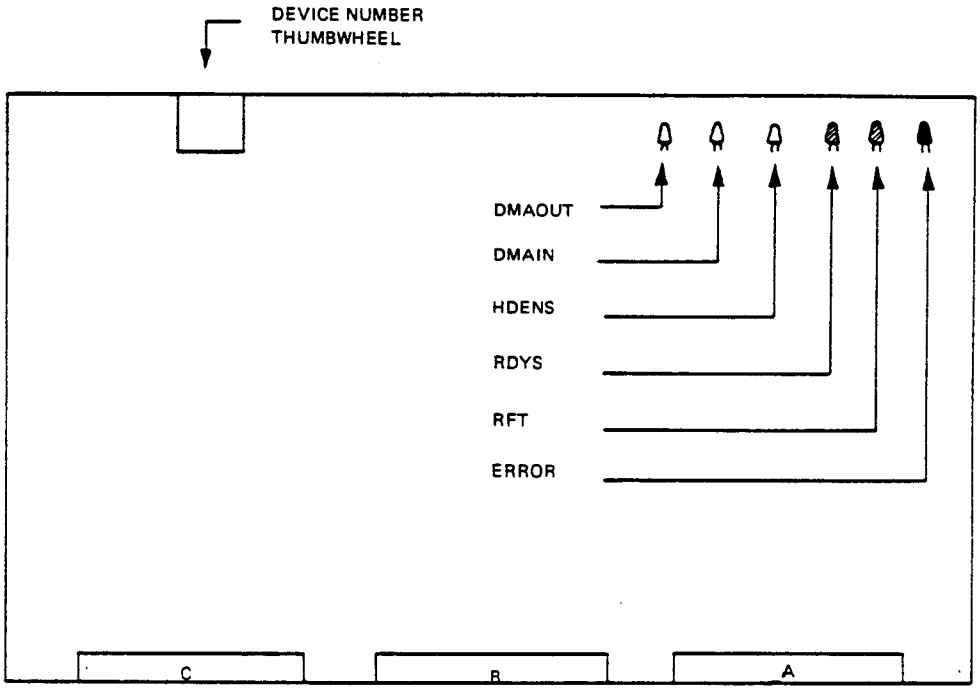


Figure D.12.1: STC Magtape Controller (3020)

D.12.1 Device Number

Thumbwheel Positions:

2 — Magtape System 1 (Dev. no. 520, IDENT=3)
3 — Magtape System 2 (Dev. no. 530, IDENT=7)
Position 0&1 and 4—15 are unused.

D.12.2 **Indicators:**

DMAOUT	—	Yellow LED indicating DMA output ie., write to magtape.
DMAIN	—	Yellow LED indicating DMA input ie., read from magtape.
HDENS	—	Yellow LED indicating high density ie., 6250 BPI (GCR)
RDYS	—	Green LED indicating Ready status from formatter ie., selected drive is ready.
RFT	—	Green LED indicating ready for transfer.
ERROR	—	Red LED indicating an error.

D.13 SWITCHES ON THE ND-500 INTERFACE (3022)

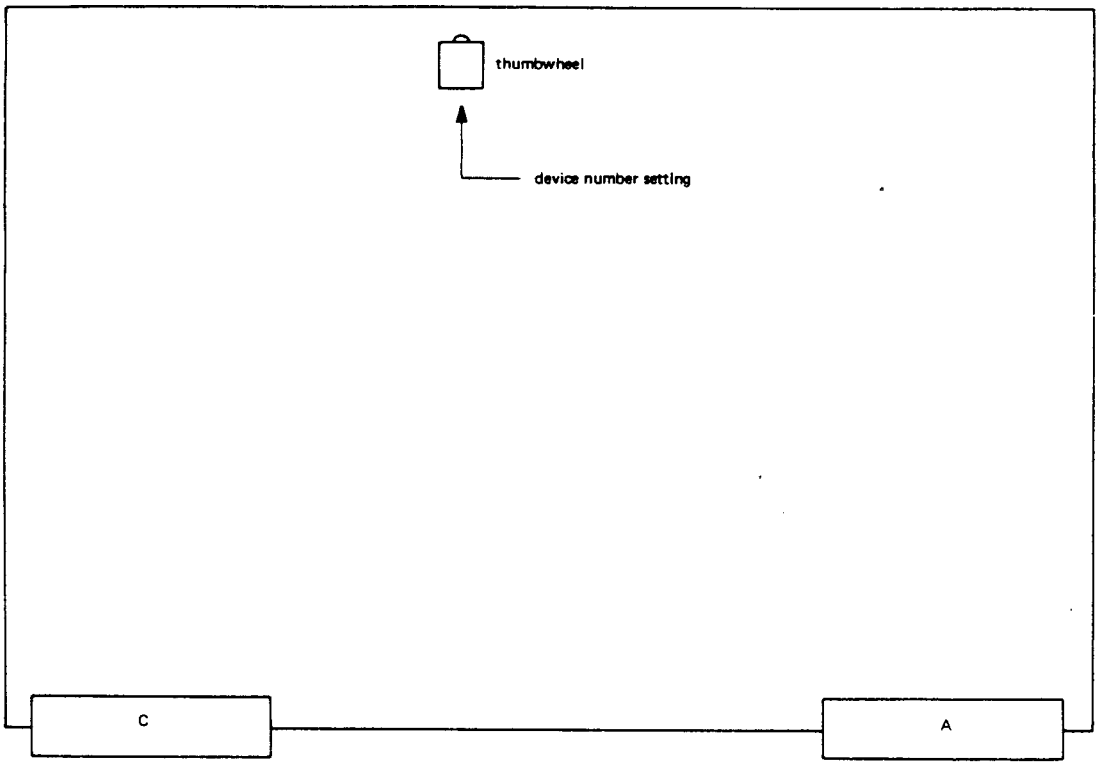


Figure D.13.1: The ND-500 Interface (3022)

D.13.1 Device Number Setting (Thumbwheel)

You may connect a maximum of 5 ND-500 computers to one ND-100. Each ND-500 computer requires a ND-500 interface in the ND-100. If you for example have 3 ND-500 computers connected to the ND-100, the first interface must have the thumbwheel set to 0, the second interface thumbwheel is set to 1 and the third interface thumbwheel is set to 2.

thumbwheel setting	device number (octal)	device register address range (octal)	ident number
0	60	60 - 77	16
1	1060	1060 - 1077	116
2	660	660 - 677	36
3	760	760 - 777	114
4	560	560 - 577	76

Thumbwheel settings 5-15 are not valid.

D.14 SWITCHES AND INDICATORS ON THE MEGALINK INTERFACE (3023)

It is recommended that the Megalink interface have the highest priority.

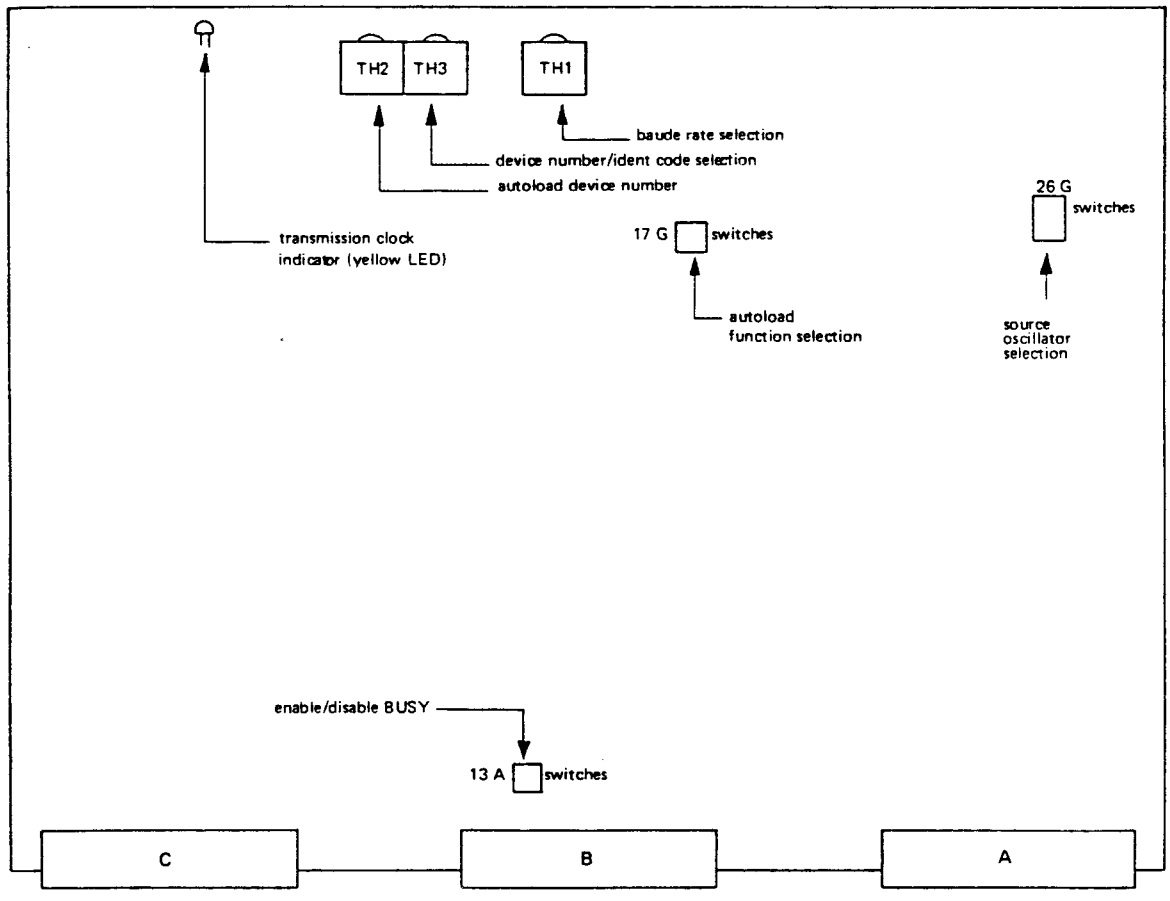


Figure D.14.1: The Megalink Interface (3023)

D.14.1 Baud Rate Selection (Thumbwheel TH1)

Baud rate selection is only relevant if the source oscillator switches are set to 'baud rate selection via TH1', see appendix D.14.4.1.

TH1 setting	baud rate (in kbps)
0	307.2 kbps
3	1.2 kbps
6	9.6 kbps
7	38.4 kbps
8	153.6 kbps
9	76.8 kbps
11	19.2 kbps
13	4.8 kbps
14	2.4 kbps

Thumbwheel settings *not* specified overlap with the specified settings.

D.14.2 Autoload Device Numbers (Thumbwheel TH2)

TH2 setting	autoload number	device register address range (octal)	
0		1600-1603	← not allowable device registers
1	autoload 1	1604-1607	
2	autoload 2	1610-1613	
3	autoload 3	1614-1617	
4	autoload 4	1620-1623	
5	autoload 5	1624-1627	
6	autoload 6	1630-1633	
7	autoload 7	1634-1637	

D.14.3 Device Register Address Range and Ident Code Selection (Thumbwheel TH3)

TH3 setting	megalink number	device register address range (octal)	ident code (octal)
0	megalink 1	1640-1657	150
1	megalink 2	1660-1677	151
2	megalink 3	1700-1717	152
3	megalink 4	1720-1737	153
4	megalink 5	1740-1757	154
5	megalink 6	1760-1777	155

Extension to more than six megalinks is possible on special request.

D.14.4 Switch Settings on the Switch in Position 26G

D.14.4.1 Source Oscillator Selection Switches

switch number						
6	5	4	3	2	1	function
on	on	off	off	on	x	special 983 kbps, normal operation
off	on	off	on	on	x	baud rate via TH1, see chapter D.14.1
off	on	on	off	on	x	special 614.4 kbps

x = irrelevant

D.14.4.2 Enable/Disable the Processor Clock

switch number	
5	function
on	normal operation, processor clock enabled
off	processor clock disabled, (only used for card tester)

D.14.4.3 Baud Rate Oscillator Test

switch number	
2	function
on	normal operation, baud rate oscillator enabled
off	baud rate oscillator disabled, (only used for card tester)

D.14.5 Autoload Function Selection (Switch 17G)

switch number				
4	3	2	1	function
off	off	off	off	normal operation, autoload function disabled
off	on	on	on	only if remote load from line is used

x = irrelevant

D.14.6 BUSY Enable/Disable (Switch 13A)

switch number				
4	3	2	1	function
x	x	off	on	normal operation, BUSY enabled
x	x	on	off	BUSY disabled

x = irrelevant

D.14.7 Transmission Clock Indicator (Yellow LED)

— LED *not* lit :.....transmission clock is running
— LED lit :.....transmission clock is *not* running

D.15 SWITCHES AND INDICATORS ON THE N10 BUS ADAPTER (3024)

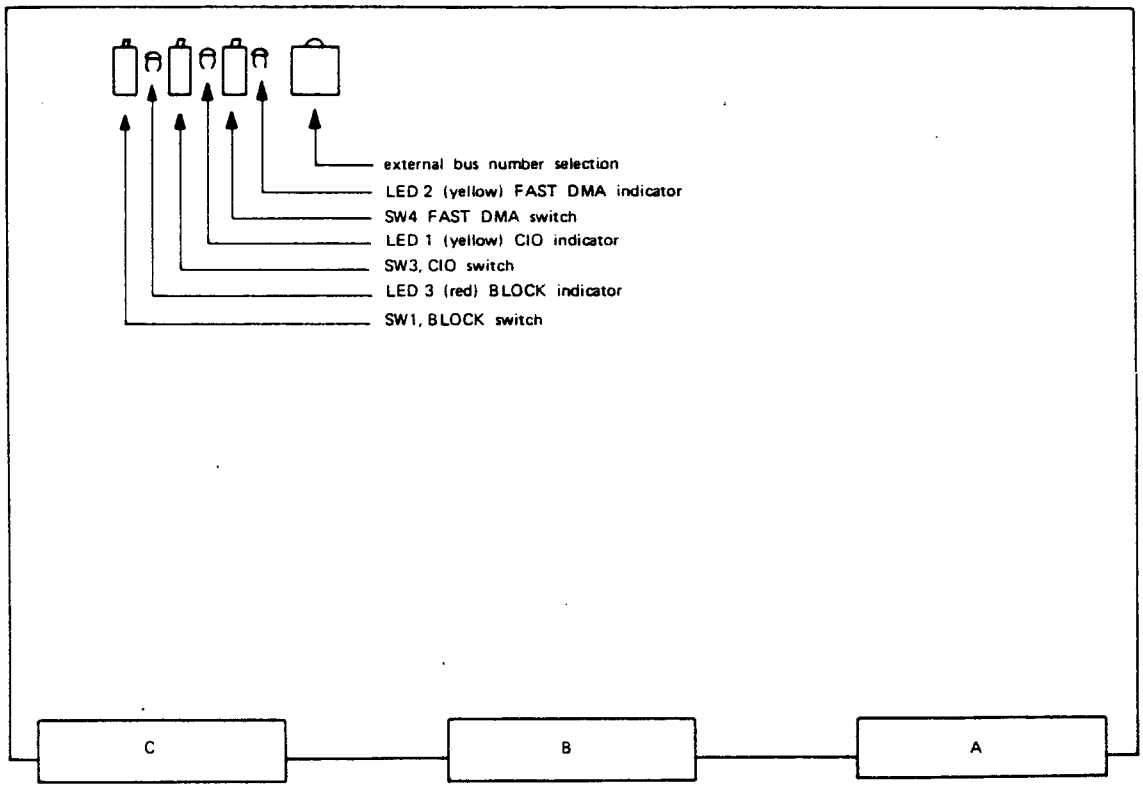


Figure D.15.1: The N10 Bus Adapter (3024)

D.15.1 External Bus Number Selection (Thumbwheel)

thumbwheel setting	external bus number
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15

D.15.2 BLOCK Switch Setting (SW 1) and Reading (LED 3)

This switch has two positions:

0 — Disable. LED 3 (red) is lit. The BLOCK switch inhibits all incoming and all outgoing signals from/to the external bus.

1 — Enable. LED 3 (red) is *not* lit. A *master clear* pulse is sent to the external bus when switching from disable to enable.

Bit 13 in the programmable register simulates the BLOCK switch.

D.15.3 CIO Switch Setting (SW 3) and Reading (LED 1)

This switch has two positions:

0 — LED 1 (yellow) is *not* lit. All IOX addresses (0-077777) will pass and all the ident codes are read. (The 3008 module forced bits 8-15 in the ident codes to 0 (zero) when the switch was inactive).

1 — LED 1 (yellow) is lit. Customer I/O enables only IOX addresses 2000-3777 to the external bus. Bit 8 in the ident code is set to 1. Ident code bits 9-15 are forced to 0 (zero).

D.15.4 The FAST DMA Switch Setting (SW4) and Reading (LED 2)

This switch has two positions:

0 — LED 2 (yellow) is *not* lit. The ND-100 bus is kept reserved until the external bus has finished the DMA cycle (CONNECT no longer true).

1 — LED 2 (yellow) is lit. The ND-100 bus is released to CPU or DMA-devices as soon as the necessary DMA data exchange on the ND-100 bus is finished (arrival of DRY) and while the external bus is ending its DMA cycle.

The switch must be set ot 0 if the DMA bandwidth of the external bus is very critical (the bus needs every DMA cycle in periods). Otherwise the switch should be set to 1 which gives optimum usage of the ND-100 bus.

To be compatible with the 3008 module the switch must be set to 0.

D.16 SWITCHES AND INDICATORS ON THE GPIB CONTROLLER (3026)

GPIB means General Purpose Interface Bus.

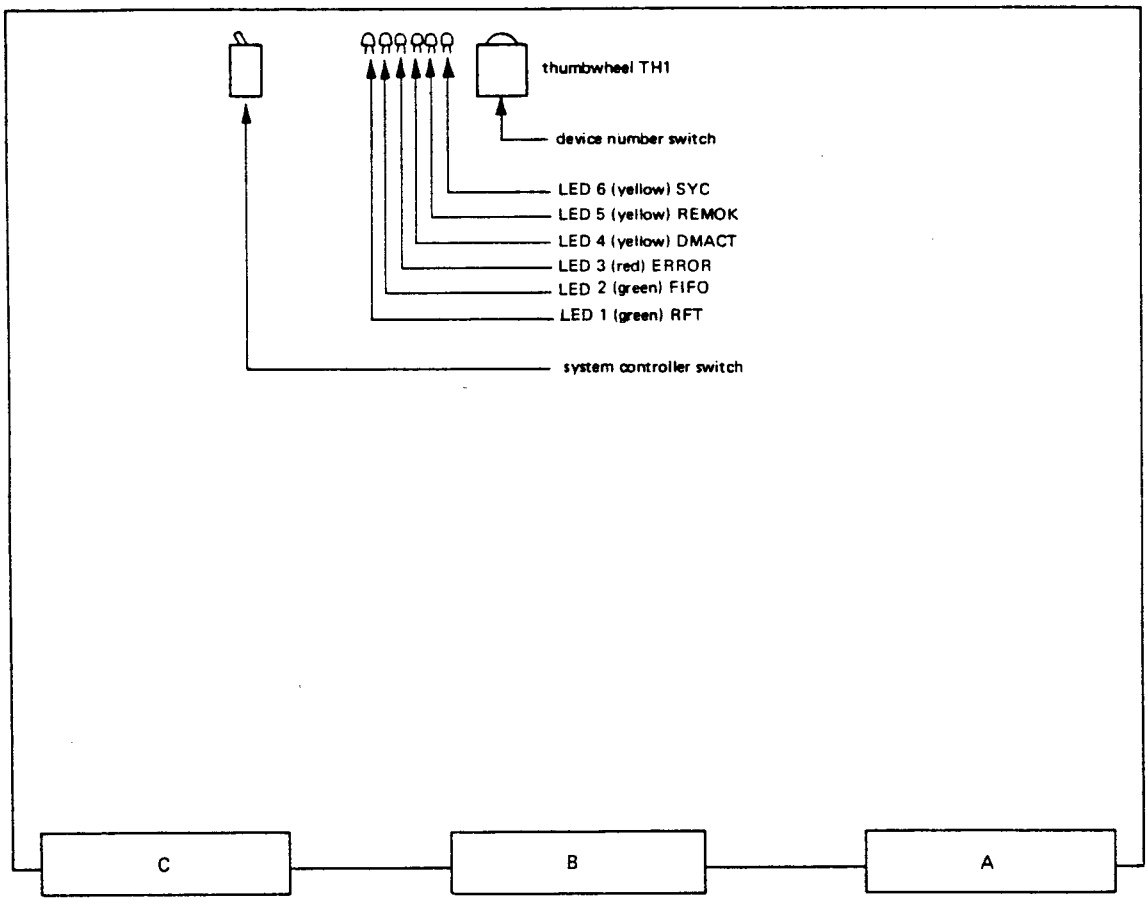


Figure D.16.1: The GPIB Controller (3026)

D.16.1 Device Number Selection (Thumbwheel TH1)

TH1 setting	device	device number (octal)	ident code (octal)	PROM addresses odd hex	odd oct	even hex	even oct
1	GPIB 0	140000	140000	00	000	00	000
2	GPIB 1	140010	140001	01	001	01	001
3	GPIB 2	141400	140140	60	140	60	140
4	GPIB 3	141410	140141	61	141	61	141
5	GPIB 4	141420	140142	62	142	62	142
6	GPIB 5	141430	140143	63	143	63	143
7	GPIB 6	141440	140144	64	144	64	144
8	GPIB 7	141450	140145	65	145	65	145

D.16.2 System Controller Selection

Switch ON — ND-100 is system controller
Switch OFF — ND-100 is *not* system controller

D.16.3 The LED Indicators

LED 1	(green)	RFT,	Ready For Transfer
LED 2	(green)	FIFO,	First In First Out register is empty
LED 3	(red)	ERROR,	Z80 microprogram has detected an error
LED 4	(yellow)	DMACT,	DMA ACTive (DMA-Direct Memory Access)
LED 5	(yellow)	REMOK,	REMote OK, remote option in function
LED 6	(yellow)	SYC,	System controller switch in ON position

D.17 SWITCH AND INDICATOR ON THE FLOPPY DISK CONTROLLER FOR DMA (3027)

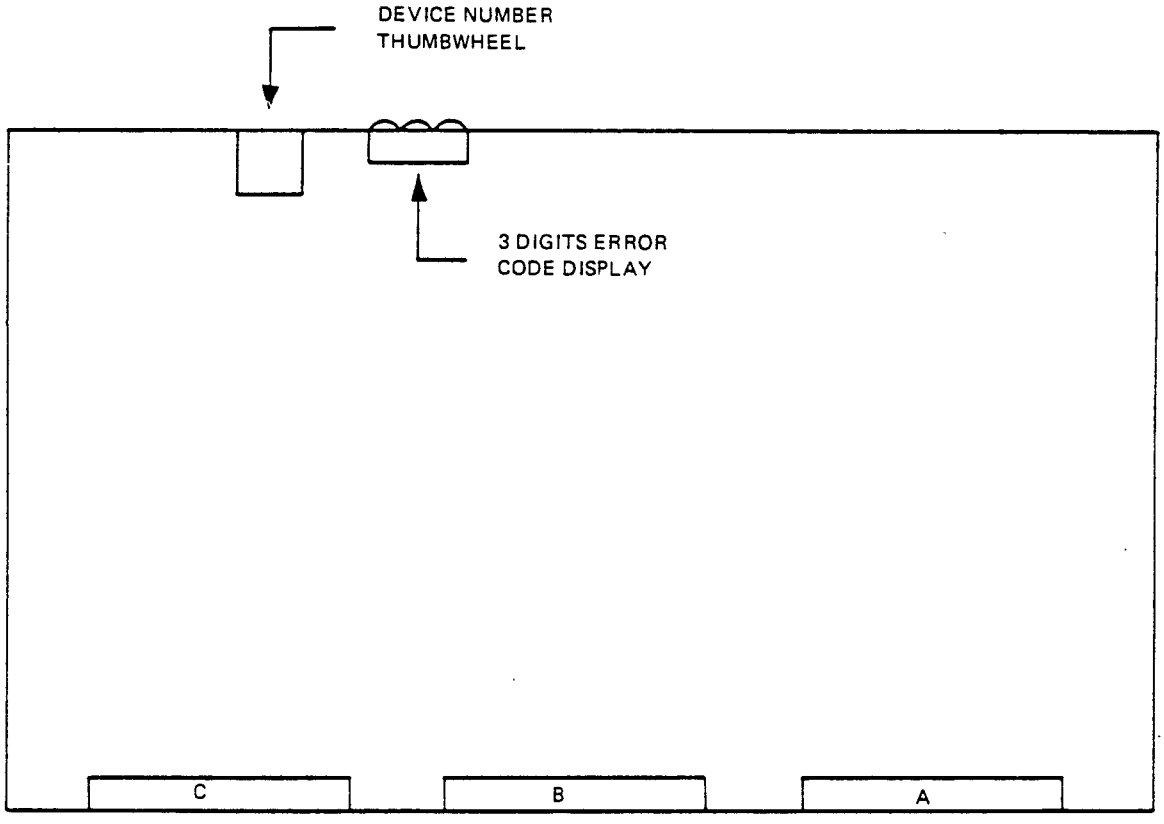


Figure D.17.1: Floppy Disk Controller for DMA (3027)

Thumbwheel pos:
- 0— Floppy Disk System no 1 (dev. no. 1560, IDENT=21)
- 1— Floppy Disk System no 2 (dev. no. 1570, IDENT=22)
- 2—15 Are unused, will answer on IOX 0—7.

Error Code Display: After Master Clear, the Floppy Controller will start a selftest. If successful, 000 will be shown, else an error code is displayed here. During transfer 000 is normally displayed. If an error occurs, an «E» with an error code is displayed.

D.18 SWITCHES AND INDICATORS ON THE BUS EXPANDER (BEX) (3028)

abbreviations.

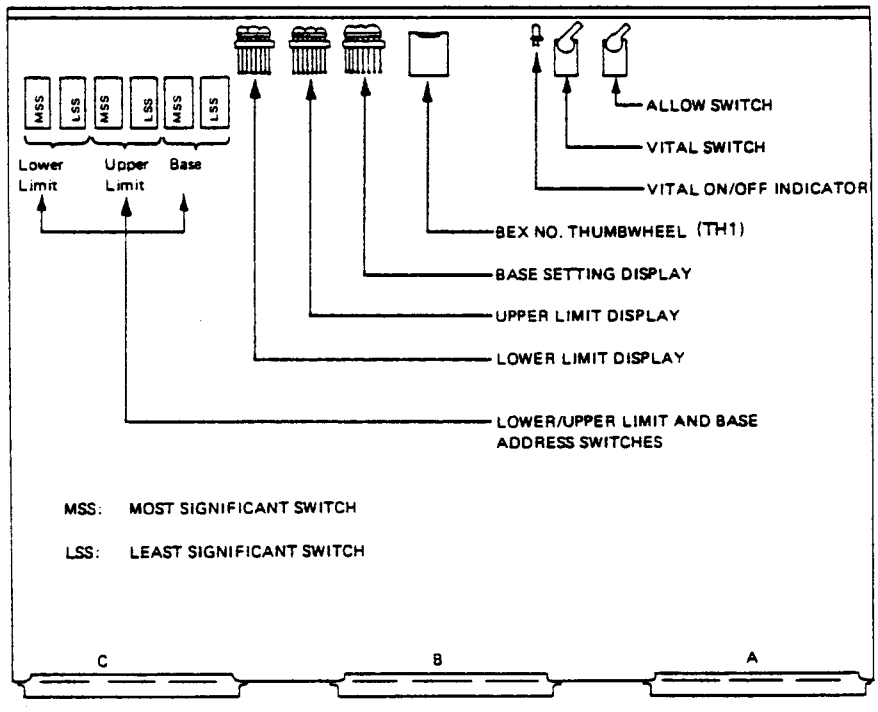


Figure D.18.1: Bus Expander (BEX) (3028)

D.18.1 Limit Switches:

On each BEX connected to a crate with memory, the limit switches are used to specify the memory area covered by the crate. These values could also be set by program.

Lower Limit — Two hex switches for setting of lower memory boundaries for this crate.

Upper limit — Two hex switches for setting of upper memory boundaries for this crate.

Base — The value of the base register is to give a positive offset to the address presented to a card crate. For addresses below 1M word, lower limit is set equal to the base.

The resolution of the switches is 64 K words per number turn on the least significant limit switch.

D.18.2 The Displays

The displays will always show the currently used limit/base setting. Thus, corresponding display and pair of switches will be equal if none of the limit registers has been programmed to another value than that set by the switches.

The Table below gives a complete list of all possible switch combinations.

The numbers given in the table correspond to the displayed value by a given switch setting in either the Lower Limit, Upper Limit or Base registers.

To get the corresponding value in K words, a table entry should be converted to decimal and multiplied by 64 K words.

D.18.3 BEX Numbers

Eight BEX numbers are defined, and the device number thumbwheel may thus take the values from 0,1,...,7.

Thumbwheel pos.	Dev.No.	Int.Level	Ident Code
0*	100000	13	10
1	100004	13	11
2	100010	13	12
3	100014	13	13
4	100020	13	14
5	100024	13	15
6	100030	13	16
7	100034	13	17

*BEX No. 0 is defined as MASTER BEX and should be placed in the A crate.

D.18.4 The Vital Switch and Indicator

The vital switch controls how a BEX (BEX NO ≥ 1) will report a power fail interrupt:

ON — Power fail in the crate will be reported as a level 14 interrupt to the CPU.

OFF — Power fail in the crate will be reported as a level 13 interrupt to the CPU.

The yellow indicator (LED) will be lit when the vital switch is in the OFF position.

D.18.5 The Allow Switch

This function is not applicable, and the switch should always be in the ON position on all BEXs.

D.19 SWITCHES AND INDICATORS ON THE N100 BUS CONTROLLER (3031)

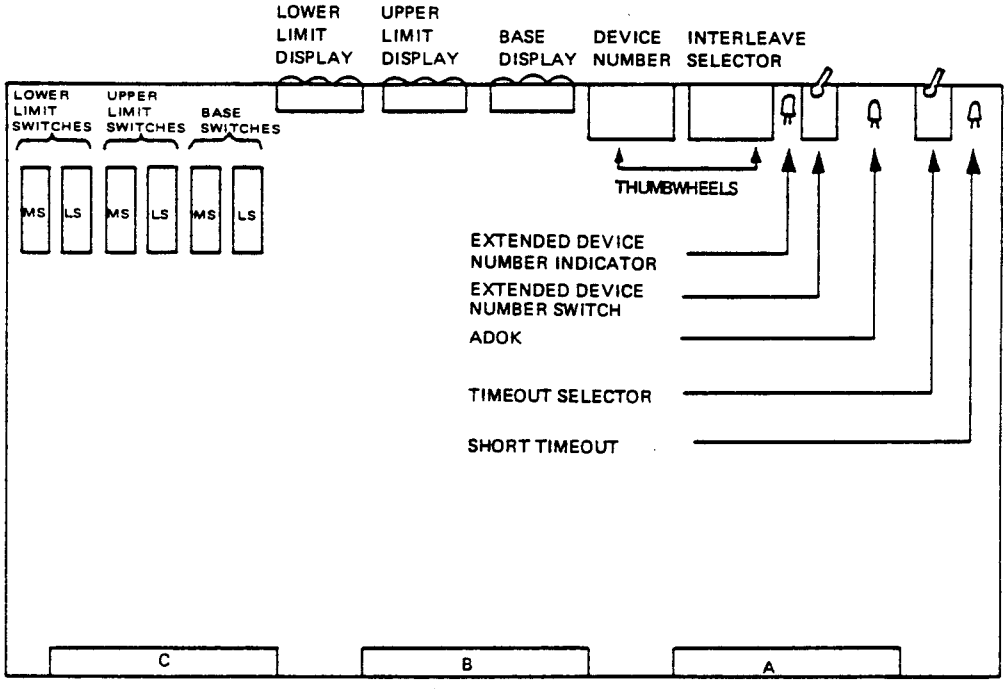


Figure D.19.1: N100 Bus Controller (3031)

D.19.1 Limit Switches

The address area for a bus controller is decided by the setting of lower and upper address limit and legal address being lower ⩽ adress < upper. The limit address increments are 64 K units.

Lower Limit Switches: Two hex switches, one most significant (MS) and one least significant (LS), for setting of lower memory boundaries.

Upper Limit Switches: Two hex switches, one most significant (MS) and one least significant (LS), for setting of upper memory boundaries.

Base Switches: Two hex switches, one most significant (MS) and one least significant (LS), for setting of the base.

D.19.2 **Device Numbers**

There are 32 Device Numbers allocated for the N100 BUS CONTROLLER. The Device Number Thumbwheel has 16 positions, and to allow 32 BUS CONTROLLERS, the Extended Device Number Switch must be used. A unique device number and Ident Code correspond to each position, in accordance with the table below:

Extended Device Number Indicator	Device Number Thumbwheel	Device Number	Ident Code Level 13
not lit	0	100200	20
not lit	1	100204	21
not lit	2	100210	22
not lit	3	100214	23
not lit	4	100220	24
not lit	5	100224	25
not lit	6	100230	26
not lit	7	100234	27
not lit	8	100240	30
not lit	9	100244	31
not lit	10	100250	32
not lit	11	100254	33
not lit	12	100260	34
not lit	13	100264	35
not lit	14	100270	36
not lit	15	100274	37
lit	0	100300	40
lit	1	100304	41
lit	2	100310	42
lit	3	100314	43
lit	4	100320	44
lit	5	100324	45
lit	6	100330	46
lit	7	100334	47
lit	8	100340	50
lit	9	100344	51
lit	10	100350	52
lit	11	100354	53
lit	12	100360	54
lit	13	100364	55
lit	14	100370	56
lit	15	100374	57

D.19.3 **Limit Displays**

The switch settings are octally displayed in the three-digit, seven-segment displays. Each switch setting will be shown in the corresponding display in 64 K word increments. The following table gives the correspondence between switch settings and display presentation.

LEAST \ MOST	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	000	020	040	060	100	120	140	160	200	220	240	260	300	320	340	360
1	001	021	041	061	101	121	141	161	201	221	241	261	301	321	341	361
2	002	022	042	062	102	122	142	162	202	222	242	262	302	322	342	362
3	003	023	043	063	103	123	143	163	203	223	243	263	303	323	343	363
4	004	024	044	064	104	124	144	164	204	224	244	264	304	324	344	364
5	005	025	045	065	105	125	145	165	205	225	245	265	305	325	345	365
6	006	026	046	066	106	126	146	166	206	226	246	266	306	326	346	366
7	007	027	047	067	107	127	147	167	207	227	247	267	307	327	347	367
8	010	030	050	070	110	130	150	170	210	230	250	270	310	330	350	370
9	011	031	051	071	111	131	151	171	211	231	251	271	311	331	351	371
A	012	032	052	072	112	132	152	172	212	232	252	272	312	332	352	372
B	013	033	053	073	113	133	153	173	213	233	253	273	313	333	353	373
C	014	034	054	074	114	134	154	174	214	234	254	274	314	334	354	374
D	015	035	055	075	115	135	155	175	215	235	255	275	315	335	355	375
E	016	036	056	076	116	136	156	176	216	236	256	276	316	336	356	376
F	017	037	057	077	117	137	157	177	217	237	257	277	317	337	357	377

D.19.4 Interleave

The interleave thumbwheel has 16 positions to allow the following selections:

Thumbwheel Position	Interleave	Vital	Delay
0	None	Yes	No
1	2-way	Yes	No
2	4-way	Yes	No
3	8-way	Yes	No
4	None	No	No
5	2-way	No	No
6	4-way	No	No
7	8-way	No	No
8	None	Yes	Yes
9	2-way	Yes	Yes
10	4-way	Yes	Yes
11	8-way	Yes	Yes
12	None	No	Yes
13	2-way	No	Yes
14	4-way	No	Yes
15	8-way	No	Yes

Vital:

If Vital=Yes (ie.,=1), a locally detected Power Fail Interrupt (PFI) will be sent to the Master ND100, the CPU detecting this as a regular power fail interrupt. If VITAL=0, the PFI will result in a level 13 interrupt which will be sent to the Master-ND100.

Delay:

If the Bus controlled by the Bus Controller contains MPM-4 Ports only, the delay is not neccessary. It is only necessary if the Bus contains regular DMA-Devices.

D.19.5 Timeout Selector

The timeout switch is used to select two different timeouts, one long (app.8 µs) and one short (app. 2 µs). The timeout indicator will be lit if short timeout is selected.

D.20 SWITCHES AND INDICATORS ON THE MEMORY PORT—MPM4 (3032)

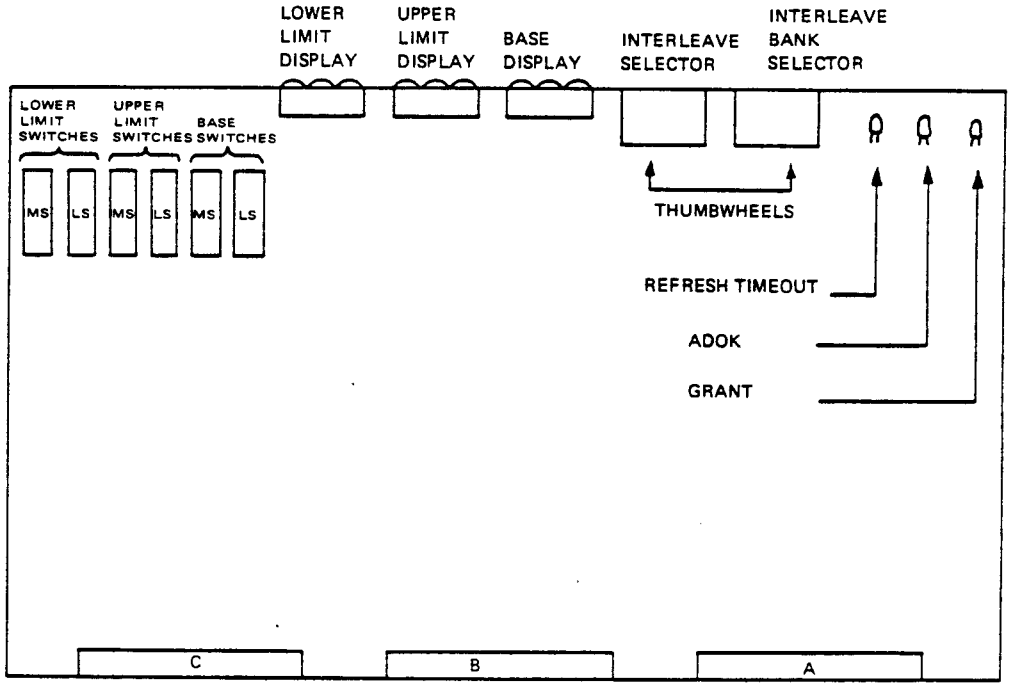


Figure D.20.1: Memory Port — MPM-4 (3032)

D.20.1 Limit Switches:

The address area for the memory port is decided by the setting of lower and upper address limits, and legal address being lower ⩽ address < upper. The limit address increments are 64 K units.

Lower Limit Switches: Two hex switches, one most significant (MS) and one least significant (LS), for setting of lower memory boundaries.

Upper Limit Switches: Two hex switches, one most significant (MS) and one least significant (LS), for setting of upper memory boundaries.

Base Switches: Two hex switches, one most significant (MS) and one least significant (LS), for setting of the base.

D.20.2 **Interleave**

The interleave thumbwheel has 16 positions to allow the following selections:

Thumbwheel position	Interleave	Speedup	Write Parity
0	None	No	No
1	2-way	No	No
2	4-way	No	No
3	8-way	No	No
4	None	Yes	No
5	2-way	Yes	No
6	4-way	Yes	No
7	8-way	Yes	No
8	None	No	Yes
9	2-way	No	Yes
10	4-way	No	Yes
11	8-way	No	Yes
12	None	Yes	Yes
13	2-way	Yes	Yes
14	4-way	Yes	Yes
15	8-way	Yes	Yes

SPEEDUP:

The specification on BMPM address set-up time being 0, it is not neccessary to have the address valid prior to the 'REQL'-signal. If the requesting source is of this type, SPEEDUP should be 0. If, however, the source has the address valid at least 40 nsec before the request is generated, the 'speedup' feature will be used to avoid unneccessary delay in access time.

WRITE PARITY:

If the requesting device is generating odd parity on each byte on write, this feature might be used to detect parity errors on data during write-cycles.

D.20.3 Interleave Bank Selector

This thumbwheel uses 8 positions (0-7) and is used in connection with the interleave selector thumbwheel. The least significant bits of the channel address are used to select the bank, and this thumbwheel selects these bits the following way:

2-ways interleave:	Bit 0 of the channel address selects one of two banks.
4-ways interleave:	Bits 0 and 1 of the channel address select one of four banks.
8-ways interleave:	Bits 0, 1 and 2 of the channel address select one of eight banks.

D.20.4 Refresh Timeout

This LED may have two differrent colours, red and green, indicating two different conditions:

Green:	Normal situation
Red:	Indicating refresh timeout (reset by MCL from master CPU)

D.20.5 ADOK

This yellow LED indicates address OK. This port has received a request on the connected channel. Lit until the next request on the channel.

D.20.6 GRANT

This yellow LED is lit when this port has been allocated a memory cycle.

D.21 SWITCHES ON THE CPU MODULE, INCLUDING THE CX OPTION (3033)

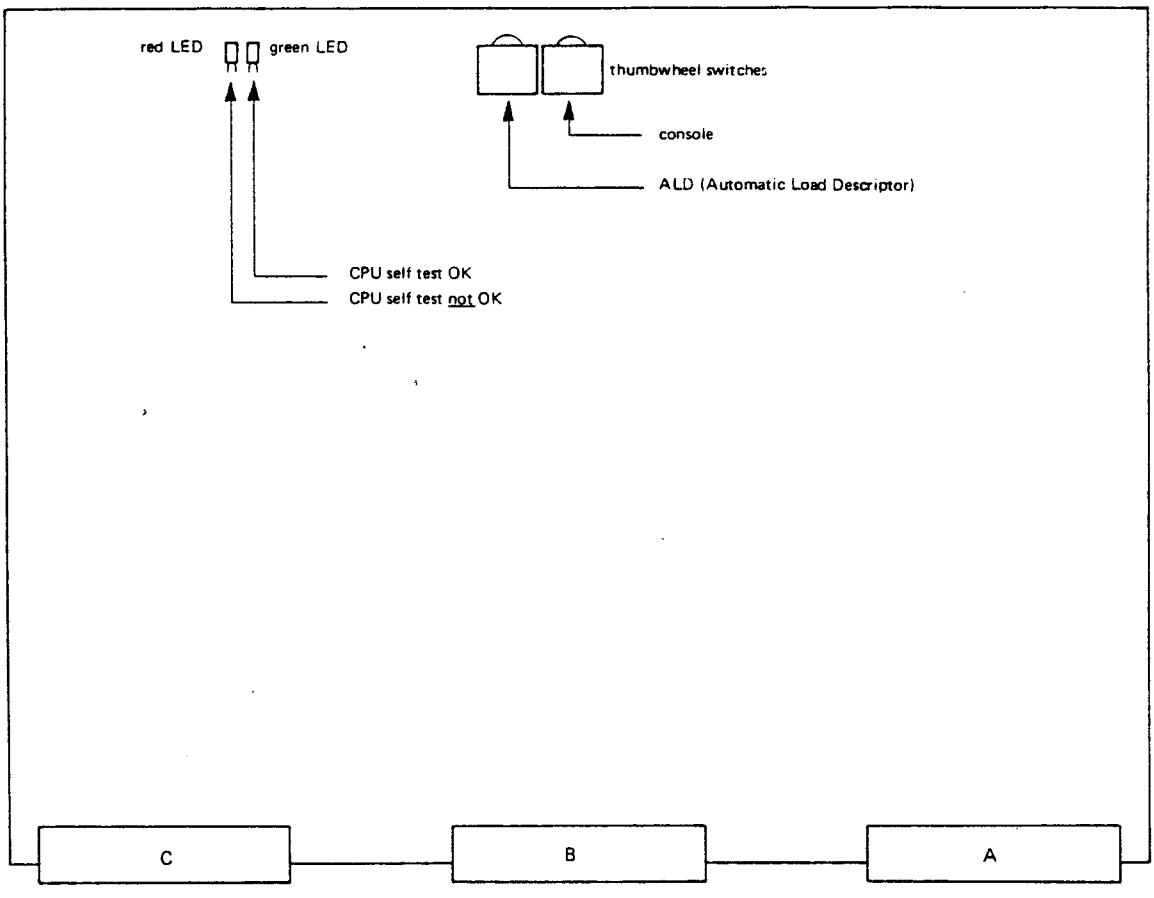


Figure D.21.1: CPU Module Including CX Option (3033)

D.21.1 ALD — Automatic Load Descriptor

ALD	i 12	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 form 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 form 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.21.2 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.21.3 The Indicators (The Red and Green LEDs)

LED = Light Emitting Diode

The green LED is lit when the CPU is OK. When the red LED is lit, this indicates a CPU error. See section 7 'Operator's Interaction' for more details about the two LEDs.

D.22 SWITCHES AND INDICATORS ON THE 256 K MEMORY MODULE (3034)

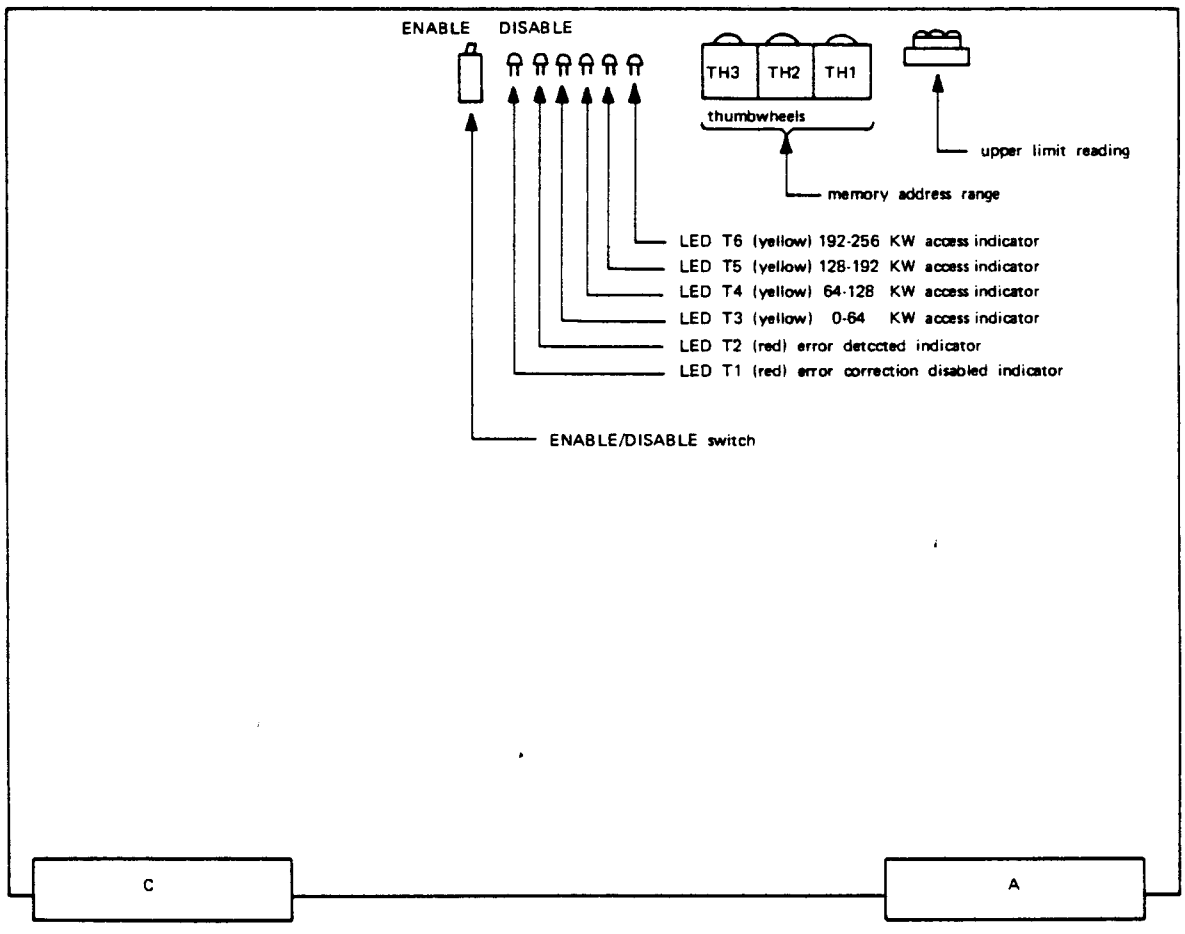


Figure D.22.1: The 256 K Memory Module (3034)

D.22.1 Memory Address Range Setting and Upper Limit Reading

thumbwheel setting (octal) TH3	TH2	TH1	upper limit display reading (octal)	memory address range lower-upper
0	0	0-3	0 2 0	0 - 256 K
0	0	4-7	0 2 4	64 - 320 K
0	1	0-3	0 3 0	128 - 384 K
0	1	4-7	0 3 4	192 - 448 K
0	2	0-3	0 4 0	256 - 512 K
0	2	4-7	0 4 4	320 - 576 K
0	3	0-3	0 5 0	384 - 640 K
•	•	•	•	•
•	•	•	•	•
1	0	0-3	1 2 0	1024 - 1280 K
1	0	4-7	1 2 4	1088 - 1344 K
1	1	0-3	1 3 0	1152 - 1408 K
•	•	•	•	•
•	•	•	•	•
7	7	4-7	7 2 0	6784 - 7040 K

D.22.2 Setting the ENAB/DISAB Switch

- Switch in position 0 (DISAB) does two things:
  1) Disables the error correction indicator (T1)
  2) Clears the error detected indicator (T2)
- Switch in position 1 (ENAB) enables the error detected indicator (T2)

D.22.3 The LED Indicators

LED T6 (yellow)	192-256 KW access indicator
LED T5 (yellow)	128-192 KW access indicator
LED T4 (yellow)	64-128 KW access indicator
LED T3 (yellow)	0- 64 KW access indicator
LED T2 (red)	error detected indicator (cleared by toggling the ENAB/DISAB switch)
LED T1 (red)	error correction disabled indicator (disabled by switch or program)

D.23 SWITCHES AND INDICATORS ON THE 64 K MEMORY MODULE (3036)

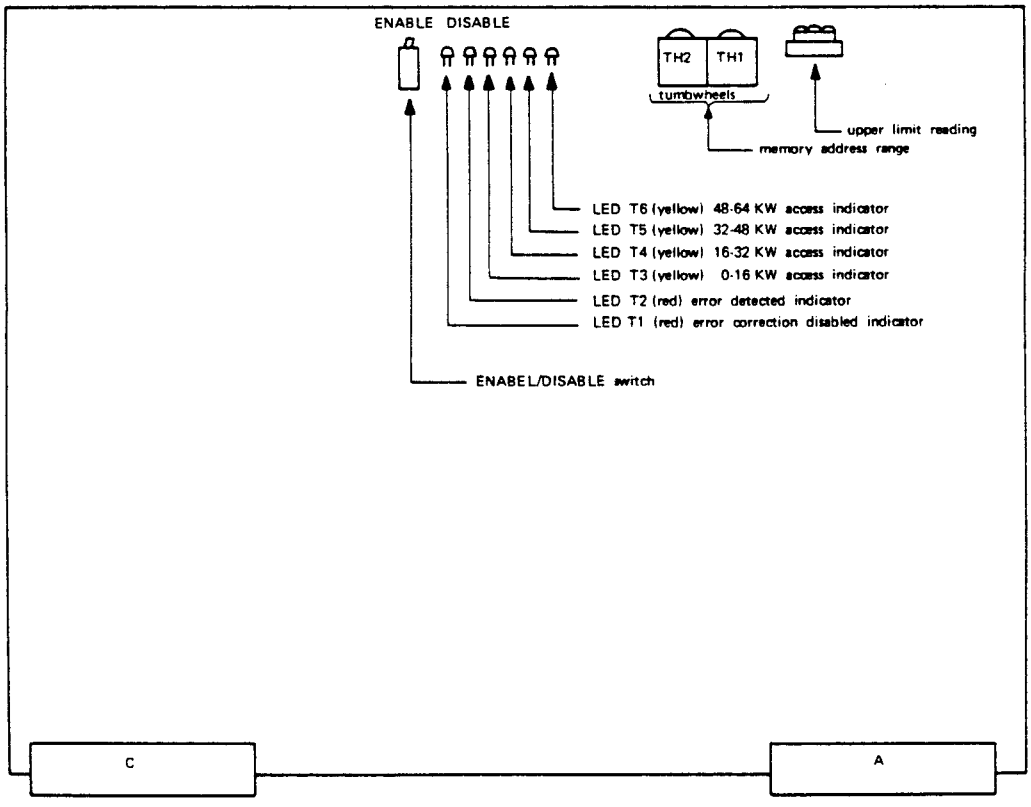


Figure D.23.1: The 64 K Memory Module (3036)

D.23.1 Memory Address Range Setting and Upper Limit Reading

thumbwheel setting (octal) TH2	TH1	upper limit display reading (octal)	memory address range lower-upper
0	0	0 4	0 - 64 K
0	1	0 5	16 - 80
0	2	0 6	32 - 96 K
•	•	•	•
•	•	•	•
1	0	1 4	128 - 192 K
1	1	1 5	144 - 208 K
•	•	•	•
•	•	•	•
7	3	7 7	944 - 1008 K

D.23.2 Setting the ENAB/DISAB Switch

- Switch in position 0 (DISAB) does two things:
  1) Disables the error correction indicator (T1)
  2) Clears the error detected indicator (T2)
- Switch in position 1 (ENAB) enables the error detected indicator (T2)

D.23.3 The LED Indicators

LED T6 (yellow)	48-64 KW access indicator
LED T5 (yellow)	32-48 KW access indicator
LED T4 (Yellow)	16-32 KW access indicator
LED T3 (Yellow)	0-16 KW access indicator
LED T2 (red)	error detected indicator (cleared by toggling the ENAB/DISAB switch)
LED T1 (red)	error correction disabled indicator (disabled by switch or program)

D.24 SWITCHES AND INDICATORS ON THE 8'' DISK CONTROLLER (3038)

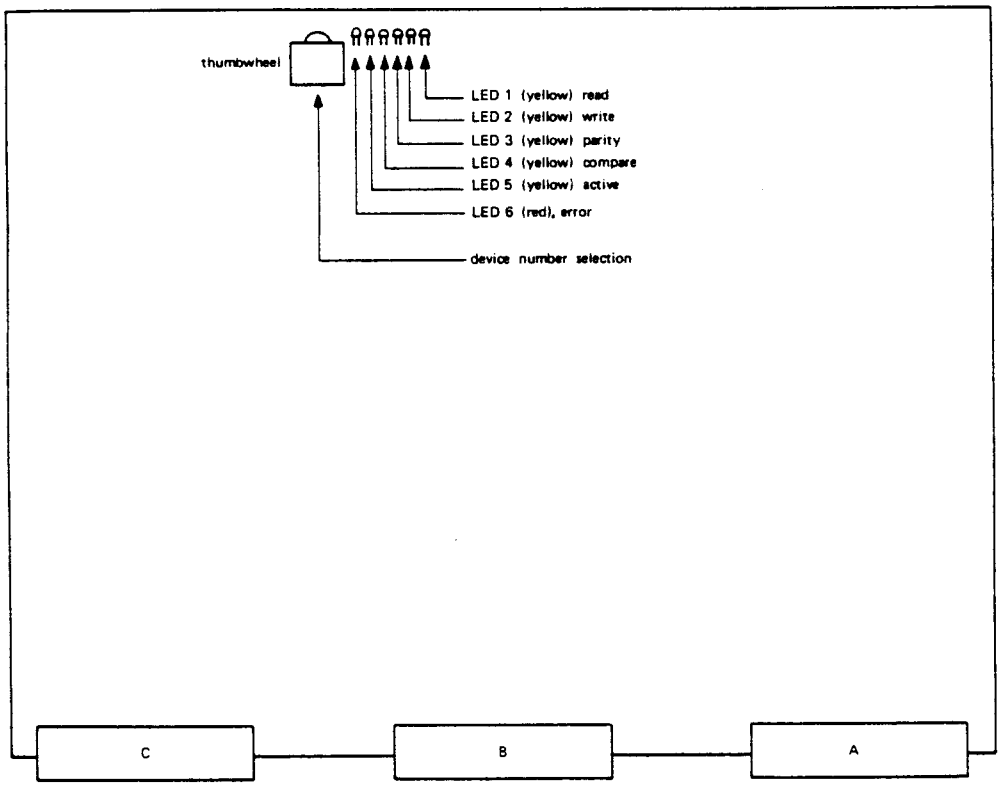


Figure D.24.1: The 8'' Disk Controller (3038)

D.24.1 Device Number Selection (Thumbwheel)

thumbwheel setting	device reg. address range (octal)	ident code (octal)	device name
0	500-507	1	the first 5 1/4'', 8'' or 10MB controller
1	510-517	5	the second 5 1/4'', 8'' or 10MB controller
2	520-527	3	reserved for magtape 1
3	530-537	7	reserved for magtape 2
4	540-547	2	reserved for drum 1
5	550-557	6	reserved for drum 2
6	600-607	4	reserved for versatec 1
7	1600-1607	14	reserved for versatec 2
8	1540-1547	17	reserved for SMD 1
9	1550-1557	20	reserved for SMD 2
10-15	not reserved		

D.24.2 The LED Indicators

LED 1 (yellow) read
LED 2 (yellow) write
LED 3 (yellow) parity
LED 4 (yellow) compare
LED 5 (yellow) active
LED 6 (red) error

D.25 SWITCHES AND INDICATORS ON THE N100 BUS CONTROLLER (3039)

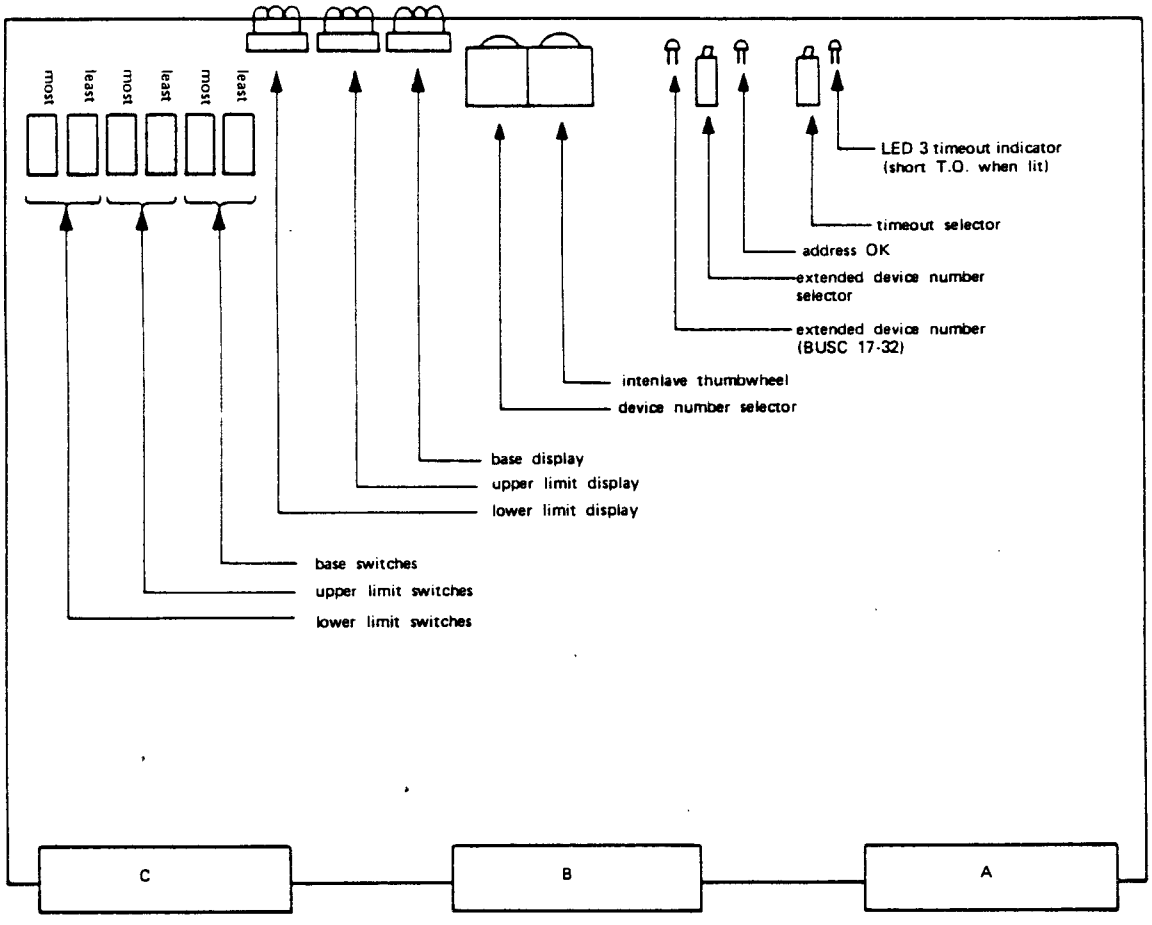


Figure D.25.1: The N100 Bus Controller (3039)

D.25.1 Memory Boundaries Setting and Reading

The memory boundaries are set by using the six *hexadecimal* switches on the module. The six switches are:

- lower limit, most significant
- lower limit, least significant
- upper limit, most significant
- upper limit, least significant
- base, most significant
- base, least significant

The switch settings are *octally* displayed in the three-digit displays. The incremets are 64 K words (128 K bytes). The address area for a BUSC is decided by the setting of lower and upper limit switches. Legal addresses are: lower limit = <address> upper limit. The following table gives the correspondence between switch settings and display presentation.

LEAST \ MOST	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	000	020	040	060	100	120	140	160	200	220	240	260	300	320	340	360
1	001	021	041	061	101	121	141	161	201	221	241	261	301	321	341	361
2	002	022	042	062	102	122	142	162	202	222	242	262	302	322	342	362
3	003	023	043	063	103	123	143	163	203	223	243	263	303	323	343	363
4	004	024	044	064	104	124	144	164	204	224	244	264	304	324	344	364
5	005	025	045	065	105	125	145	165	205	225	245	265	305	325	345	365
6	006	026	046	066	106	126	146	166	206	226	246	266	306	326	346	366
7	007	027	047	067	107	127	147	167	207	227	247	267	307	327	347	367
8	010	030	050	070	110	130	150	170	210	230	250	270	310	330	350	370
9	011	031	051	071	111	131	151	171	211	231	251	271	311	331	351	371
A	012	032	052	072	112	132	152	172	212	232	252	272	312	332	352	372
B	013	033	053	073	113	133	153	173	213	233	253	273	313	333	353	373
C	014	034	054	074	114	134	154	174	214	234	254	274	314	334	354	374
D	015	035	055	075	115	135	155	175	215	235	255	275	315	335	355	375
E	016	036	056	076	116	136	156	176	216	236	256	276	216	336	356	376
F	017	037	057	077	117	137	157	177	217	237	257	277	317	337	357	377

D.25.2 Device Number Setting (Thumbwheel) and Extended Device Number Diode Reading

thumbwheel setting	device register address range (octal)	ident. code (octal) (int. lev. 13)	extended device number LED
0	100200-100203	20	not lit
1	100204-100207	21	not lit
2	100210-100213	22	not lit
3	100214-100217	23	not lit
4	100220-100223	24	not lit
5	100224-100227	25	not lit
6	100230-100233	26	not lit
7	100234-100237	27	not lit
8	100240-100243	30	not lit
9	100244-100247	31	not lit
10	100250-100253	32	not lit
11	100254-100257	33	not lit
12	100260-100263	34	not lit
13	100264-100267	35	not lit
14	100270-100273	36	not lit
15	100274-100277	37	not lit
0	100300-100303	40	lit
1	100304-100307	41	lit
2	100310-100313	42	lit
3	100314-100317	43	lit
4	100320-100323	44	lit
5	100324-100327	45	lit
6	100330-100333	46	lit
7	100334-100337	47	lit
8	100340-100343	50	lit
9	100344-100347	51	lit
10	100350-100353	52	lit
11	100354-100357	53	lit
12	100360-100363	54	lit
13	100364-100367	55	lit
14	100370-100373	56	lit
15	100374-100377	57	lit

D.25.3 Interleave Setting (thumbwheel)

The interleave thumbwheel has 16 positions giving the following selections:

thumbwheel setting	interleave	vital	delay
0	none	yes	no
1	2-way	yes	no
2	4-way	yes	no
3	8-way	yes	no
4	none	no	no
5	2-way	no	no
6	4-way	no	no
7	8-way	no	no
8	none	yes	yes
9	2-way	yes	yes
10	4-way	yes	yes
11	8-way	yes	yes
12	none	no	yes
13	2-way	no	yes
14	4-way	no	yes
15	8-way	no	yes

D.25.4 Timeout Setting and Reading

The timeout selector is used to select the timeout length for the BUSC. A BUSC controlling a memory bank should have a short timeout. A BUSC controlling an I/O bus should have a long timeout. A short timeout is indicated by the timeout indicator (yellow LED) being lit.

D.25.5 The Address OK Indicator (yellow LED)

This LED indicates which bank the last Master-bus memory cycle was accessing. It is reset by the start of the next Master-bus memory cycle.

D.26 SWITCHES AND INDICATORS ON THE 5 1/4'' DISK CONTROLLER (3041)

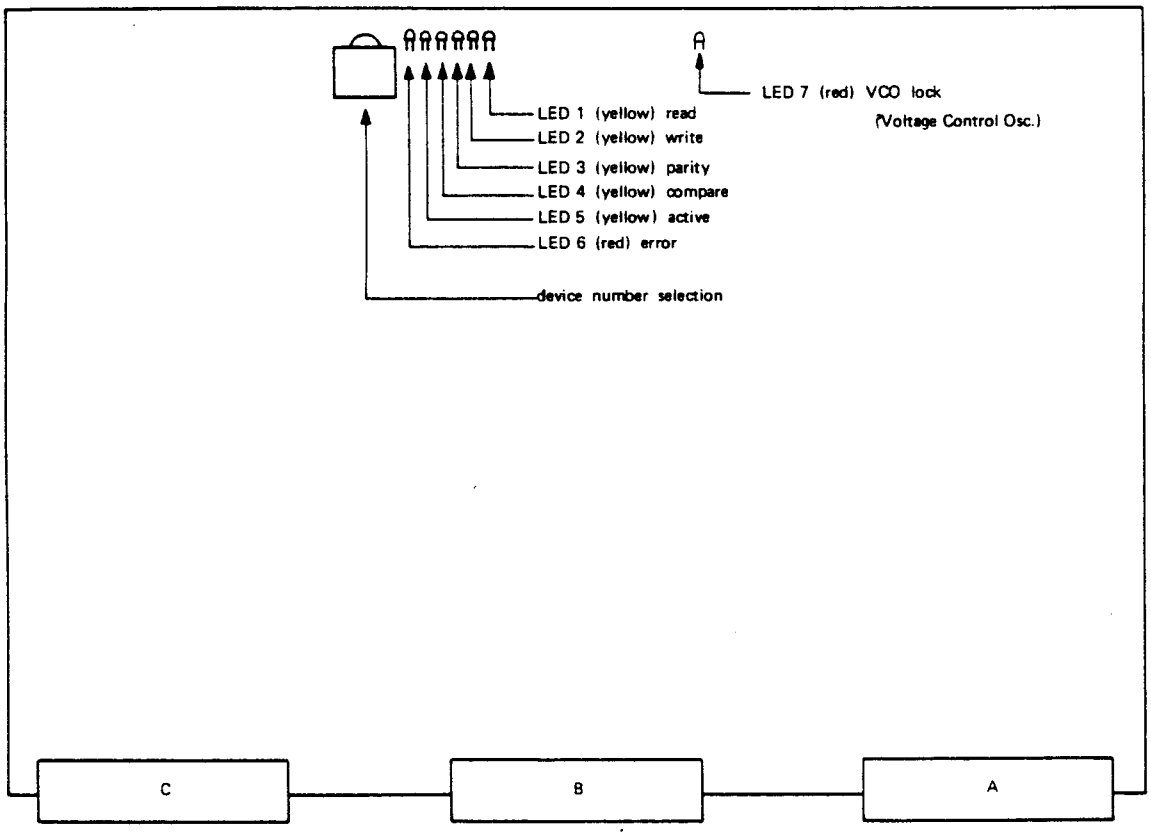


Figure D.26.1: The 5 1/4'' Disk Controller (3041)

D.26.1 Device Number Selection (Thumbwheel)

thumbwheel setting	device reg. address range (octal)	ident code (octal)	device name
0	500-507	1	the first 5 1/4'', 8'' or 10 MB controller
1	510-517	5	the second 5 1/4'', 8'' or 10 MB controller
2	520-527	3	reserved for magtape 1
3	530-537	7	reserved for magtape 2
4	540-547	2	reserved for drum 1
5	550-557	6	reserved for drum 2
6	600-607	4	reserved for versatec 1
7	1600-1607	14	reserved for versatec 2
8	1540-1547	17	reserved for SMD 1
9	1550-1557	20	reserved for SMD 2
10-15	not reserved		

D.26.2 The LED Indicators

LED 1 (yellow)	read
LED 2 (yellow)	write
LED 3 (yellow)	parity
LED 4 (yellow)	compare
LED 5 (yellow)	active
LED 6 (red)	error
LED 7 (red)	VCO lock (Voltage Control OK)

D.27 SWITCHES AND INDICATORS ON THE 15MHz ECC DISK CONTROLLER (3043 AND 3044)

This ECC disk controller replaces the older disk controller (3018 and 3019)

D.27.1 SMD Control

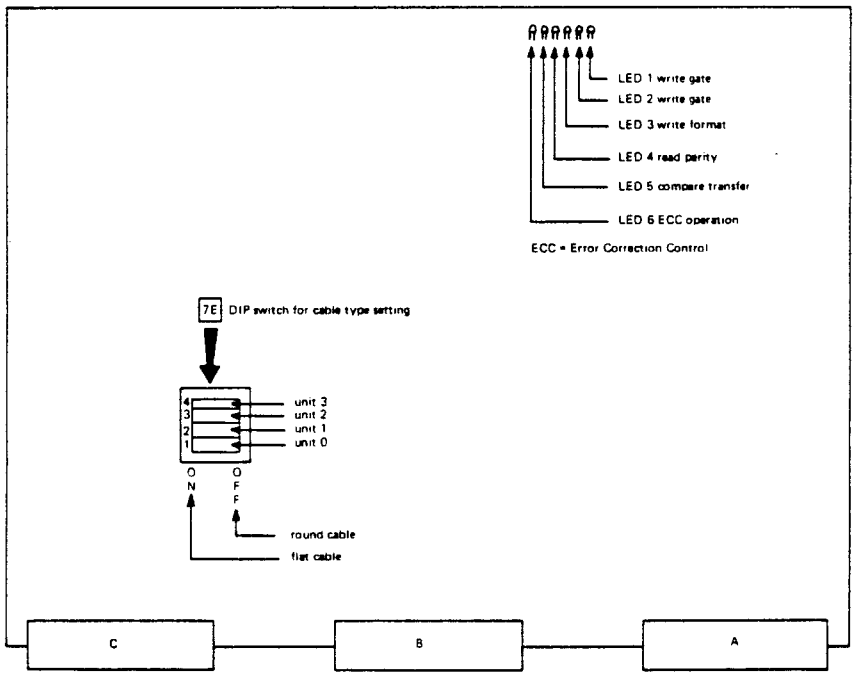


Figure D.27.1: SMD Control (3043)

D.27.1.1 Indicators, LED 1 - LED 6

All indicators are yellow LEDs, with the following meanings:

LED 1	write gate, is lit when the disk head writes.
LED 2	read gate, is lit when the disk head reads.
LED 3	write format, is lit when the controller formats.
LED 4	read parity, is lit when the controller checks parity.
LED 5	compare transfer, is lit when the controller compares data after a read/write.
LED 6	ECC operation, is lit when the controller performs error correction calculation.

D.27.1.2 Cable Type Setting (DIP switch in position 7E)

The switch setting concerns B-cable only (B-cable are the cables going directly from the computer to the disk unit). The daisy chain cable (A-cable) may be flat or round.

Switch 7E consists of 4 switches, one each disk unit. The switch must be OFF when the B-cable is *round*, and ON when the B-cable is *flat*.

unit 0	switch i OFF ON	round B-cable between computer and disk unit 0 flat B-cable between computer and disk unit 0
unit 1	switch 2 OFF ON	round B-cable between computer and disk unit 1 flat B-cable between computer and disk unit 1
unit 2	switch 3 OFF ON	round B-cable between computer and disk unit 2 flat B-cable between computer and disk unit 2
unit 3	switch 4 OFF ON	round B-cable between computer and disk unit 3 flat B-cable between computer and disk unit 3

D.27.2 SMD Data (3044)

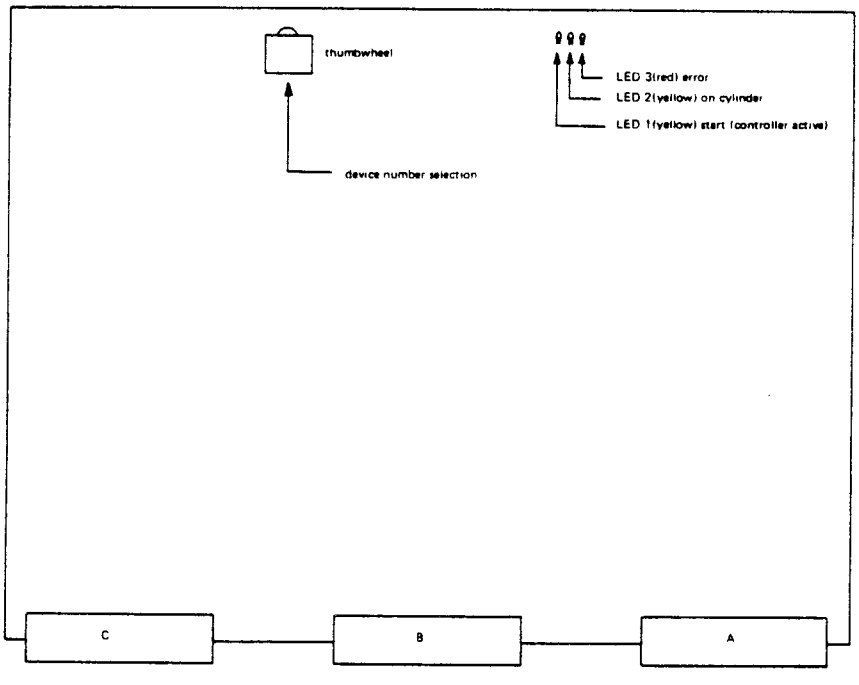


Figure 27.2: SMD Data (3044)

D.27.2.1 Device Number Selection (Thumbwheel)

thumbwheel setting	device register addressr range (octal)	ident. code (octal)	device name
1-7	not used		
8	1540-1547	17	Big Disk System 1
9	1550-1547	20	Big Disk System
10-15	not used		

D.27.2.2 Indicators, LED 1 - LED 3

LED 1 (yellow) start, controller active.
LED 2 (yellow) on cylinder, read/write head on cylinder.
LED 3 (red) error.

D.28 SWITCHES AND INDICATORS ON THE PIOC (3101)

PIOC is an abbreviation for Programmable Input/Output Controller. For more details, see the manual 'PIOC Hardware Reference Manual, ND-02.003'.

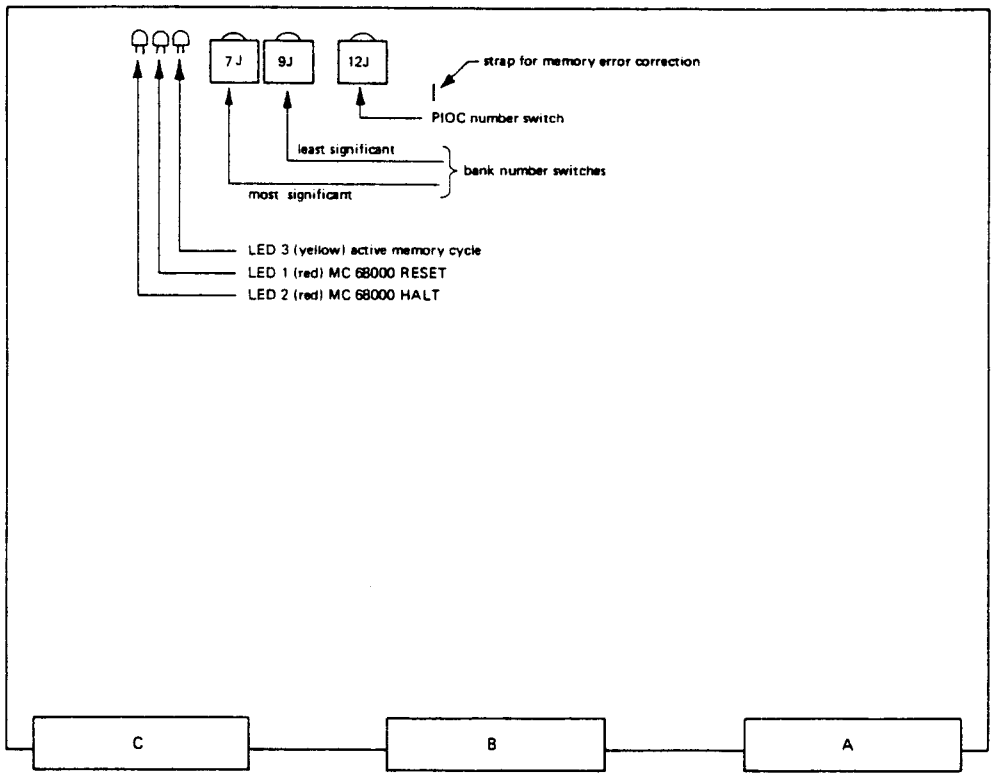


Figure D.28.1: The PIOC (3101)

D.28.1 PIOC Number Setting (Switch 12J)

PIOC number	device number (octal)	ident code (octal)
0	140020	140002
1	140024	140003
2	140030	140004
3	140034	140005
4	140040	140006
5	140044	140007
6-15	not used	not used

D.28.2 Bank Number Selection (Switches 7J and 9J)

The bank number is found by using the formula:
bank number = (7J setting * 16) + 9J setting

Example:
7J is set to 3, 9J is set to 4
bank number = (3 * 16) + 4 = 48 + 4 = *52*

D.28.3 Lowest Address Selection/Reading (Switches 7J and 9J)

The lowest address in PIOC seen from ND-100 is set/found by using the following formula:
lowest address = (7J setting * 2048 K) + (9J setting * 128 K)

Example:
7J is set to 3, 9J is set to 4
lowest address = (3 * 2048 K) + (4 * 128 K) = 6656 K

D.28.4 The LED Indicators

LED 3 (Yellow) is lit when a memory cycle is performed in the PIOC memory either by the MC68000, by ND-100 or by internal PIOC DMA transfer.

LED 1 (Red) is lit when the RESET signal is given to the MC68000. RESET (LED 1) is lit after pushing the ND-100 MCL button and is not lit when MC68000 executes the programs.

LED 2 (Red) is lit when the HALT (stop) is given to MC68000, and it is lit after the ND-100 MCL button is pushed. The LED is not lit when the MC68000 executes programs.

D.29 SWITCH AND INDICATOR ON THE MEMORY MANAGEMENT II (3104)

See also chapter D.8 'SWITCH AND INDICATOR ON THE MEMORY MANAGEMENT (3012)'.

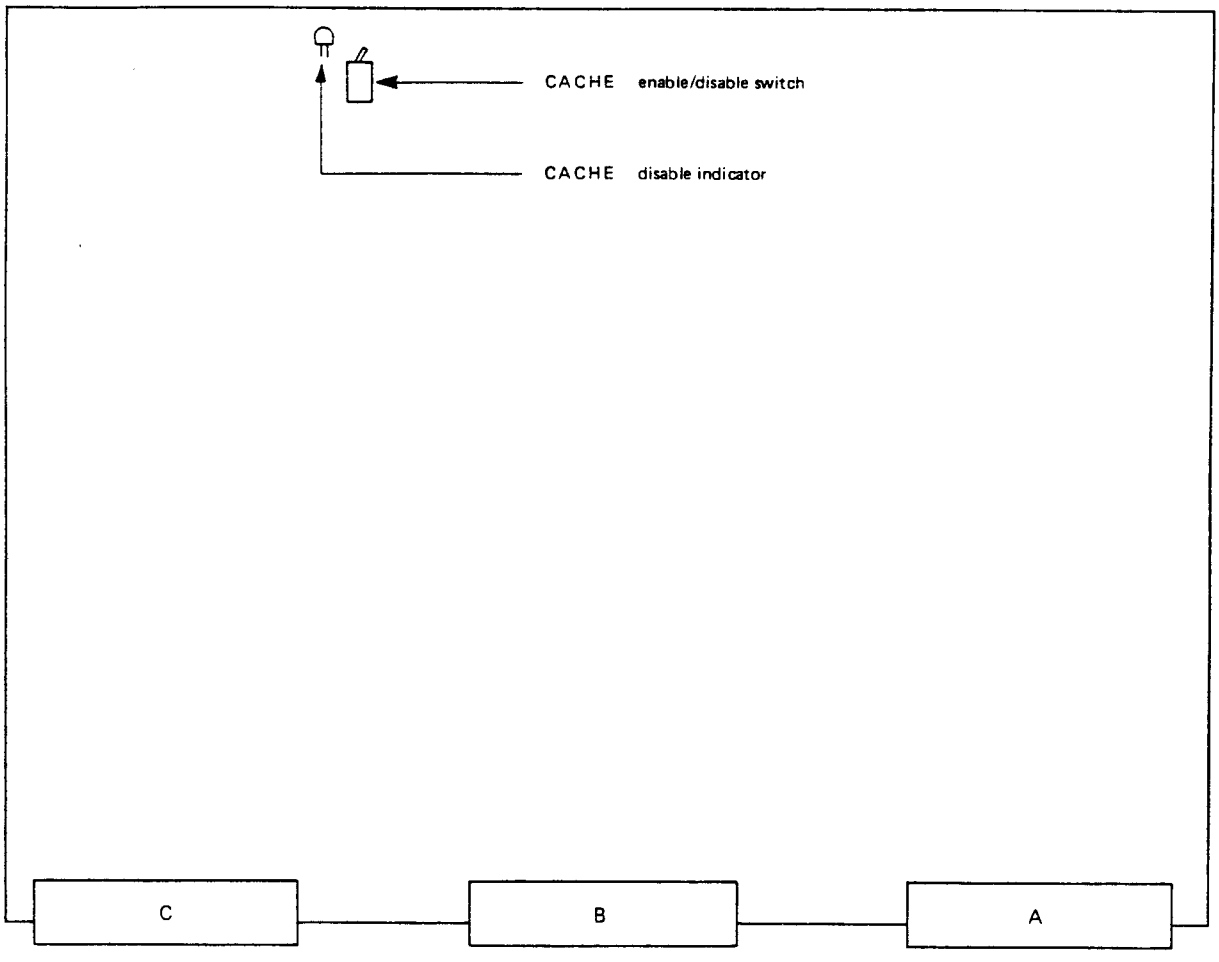


Figure D.29.1: Memory Management II (3104)

D.29.1 Cache Memory Enable/Disable Switch and Indicator

The switch enables or disables the cache memory. This memory is an option on the memory management card. The cache disable indicator (red LED) is lit when the cache memory is disabled.

D.30 SWITCHES ON THE 8-TELEX INTERFACE (3105)

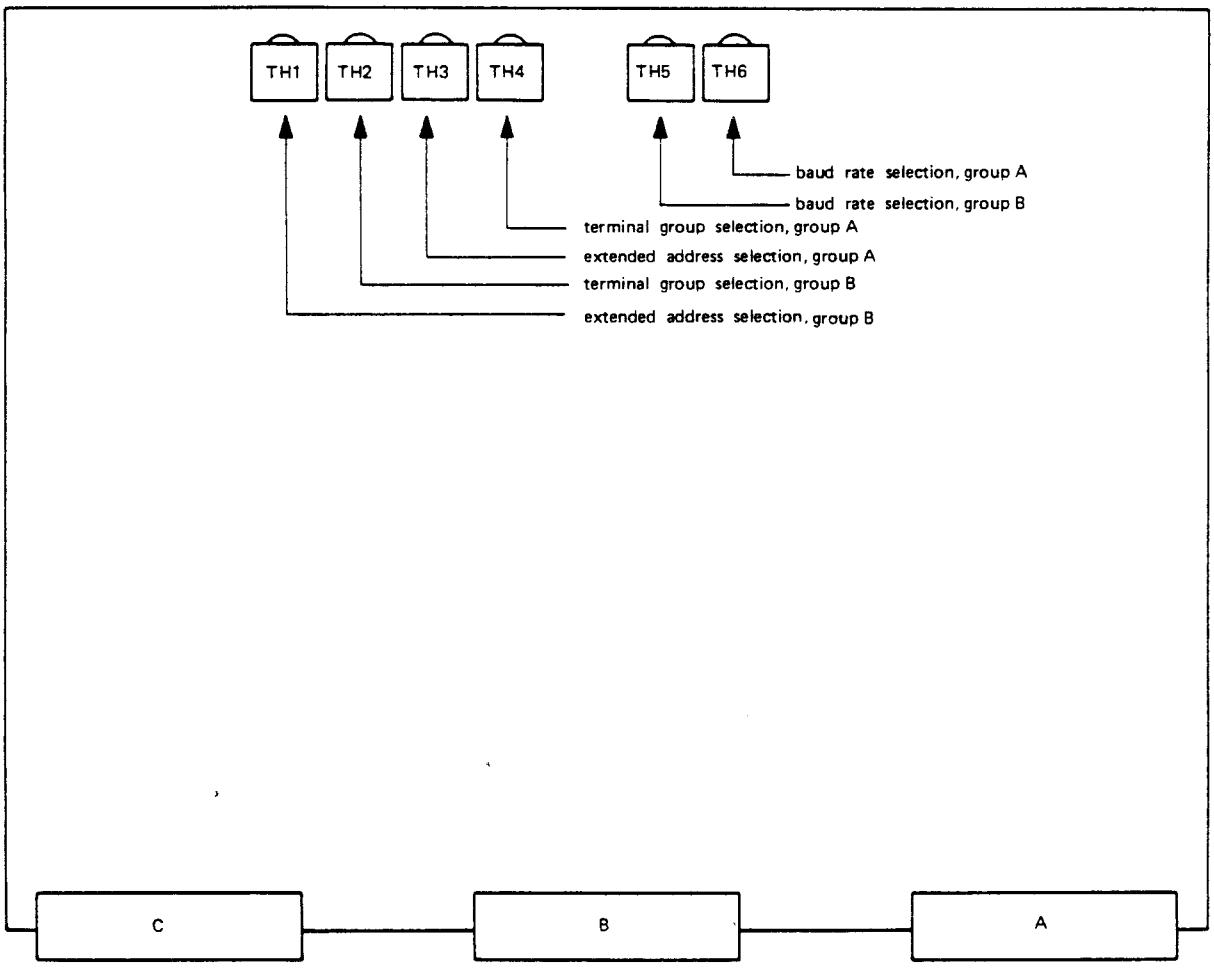


Figure D.30.1: The 8-Telex Interface (3105)

D.30.1 Extended Address Selection (TH1 and TH2)

These thumbwheels have only two valid positions:
0 — normal address range
1 — extended address range

Extended address means terminals 65 to 128. These terminals are only reached by using the ND-100 instruction IOXT.

D.30.2 Terminal Group Selection (TH2 and TH4)

Normal address range selected (terminals 1-64)

thumbwheel setting	group	terminals	device register address range (octal)
0	0	1 - 4	300 - 337
1	1	5 - 8	340 - 377
2	2	9 - 12	1300 - 1337
3	3	13 - 16	1340 - 1377
4	4	33 - 36	640 - 677
5	5	37 - 40	1100 - 1137
6	6	41 - 44	1140 - 1177
7	7	45 - 48	1400 - 1437
8	8	49 - 52	1500 - 1537
9	9	53 - 56	1640 - 1677
10	10	57 - 60	1700 - 1737
11	11	61 - 64	1740 - 1777
12	12	17 - 20	200 - 237
13	13	21 - 24	240 - 277
14	14	25 - 28	1200 - 1237
15	15	29 - 32	1240 - 1277

Extended address range selected (terminals 65-128)

thumbwheel setting	group	terminals	device register address range (octal)
0	0	65 - 68	140400 - 140437
1	1	69 - 72	140440 - 140477
2	2	73 - 76	140500 - 140537
3	3	77 - 80	140540 - 140577
4	4	81 - 84	140600 - 140637
5	5	85 - 88	140640 - 140677
6	6	89 - 92	140700 - 140737
7	7	93 - 96	140740 - 140777
8	8	97 - 100	141000 - 141037
9	9	101 - 104	141040 - 141077
10	10	105 - 108	141100 - 141137
11	11	109 - 112	141140 - 141177
12	12	113 - 116	141200 - 141237
13	13	117 - 120	141240 - 141277
14	14	121 - 124	141300 - 141337
15	15	125 - 128	141340 - 141377

D.30.3 Baud Rate Selection (TH1 and TH2)

thumbwheel setting	baud rate
0	110
1	150
2	300
3	2400
4	1200
5	1800
6	4800
7	9600
8	2400
9	600
10	200
11	134.5
12	75
13	50
14	100
15	100

D.31 SWITCHES ON THE FLOPPY AND STREAMER CONTROLLER (3106)

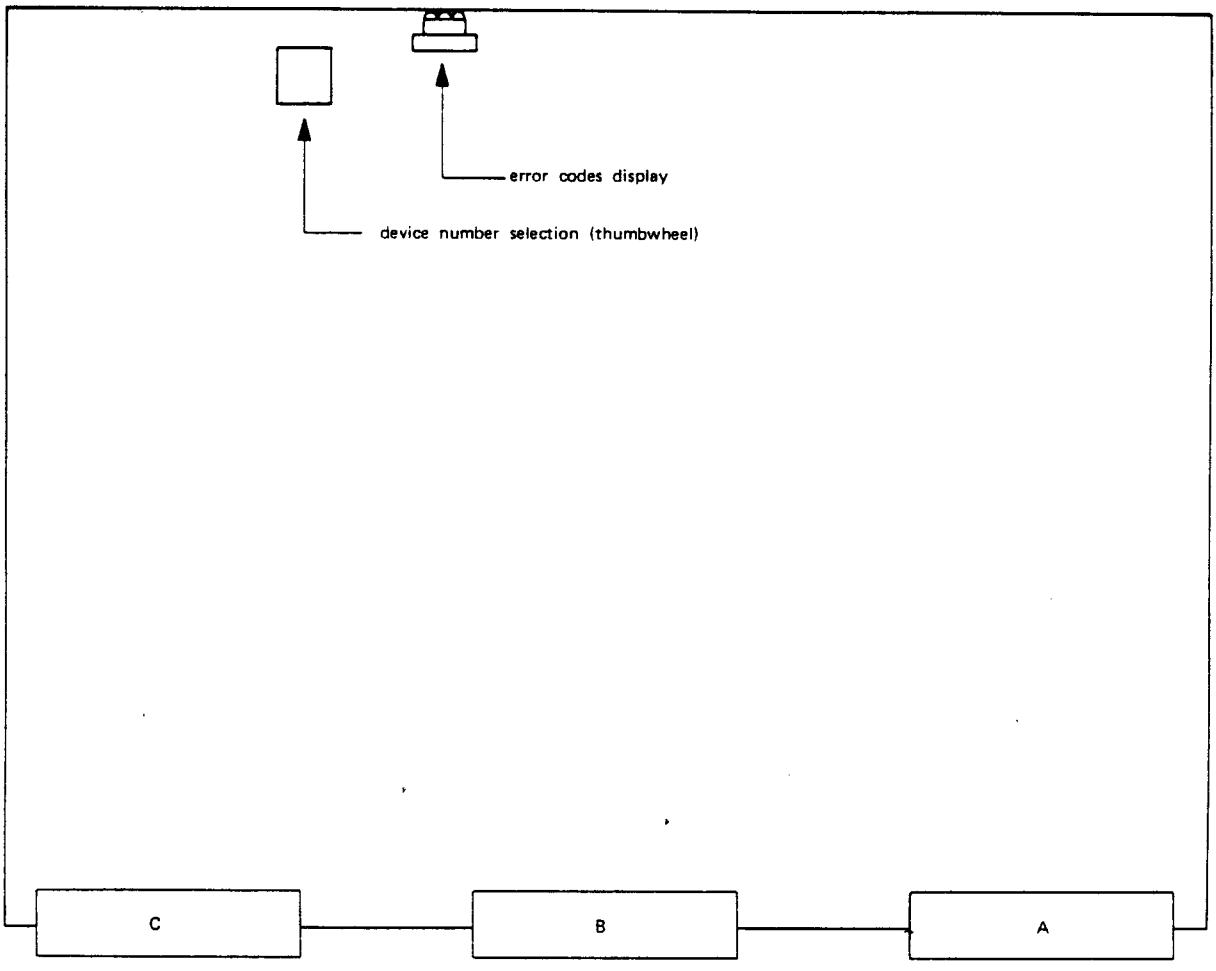


Figure D.31.1: The Floppy and Streamer Controller (3106)

D.31.1 Device Number Selection (Thumbwheel)

thumbwheel setting	device register address range (octal)
0	1560 - 1567
1	1570 - 1577
2-15	not used

D.31.2 Error Codes Display

The error codes are found in bits 9-15 in status word 1. The codes are also displayed in the digital display on the rear edge of the controller.

display reading (octal)	meaning
000	OK
005	CRC error
006	Sector not found
007	Track not found
010	Format not found
011	Diskette defect (impossible to format)
012	Format mismatch
013	Illegal format specified
014	Single-sided diskette inserted
015	Double-sided diskette inserted
016	Write protected diskette
017	Deleted record
020	Drive not ready
021	Controller busy on start
022	Lost data (over or underrun)
023	Track zero not detected
024	VCO (Voltage Controlled Oscillator) frequence out of range
025	Microprogram out of range
026	Timeout
027	Undefined error
030	Track out of range
031	Not used
032	Compare error (during compare of data)
033	Internal DMA errors
034	
035	Not used
036	
037	
040	ND-100 Bus error command fetch
041	ND-100 Bus error status transfer
042	ND-100 Bus error data transfer
043	Illegal command
044	Wordcount not zero
045	Illegal completion (cont. transf.)
046	Addr-reg error
047	Not used
050	No bootstrap found on diskette — Error during
051	Wrong bootstrap (too old flo-mon version) — autoload
052	
053	
054	
055	Not used
056	
057	

display reading (octal)	meaning	
060	Streamer handshake error	
061	Streamer status transfer error	
062	Bad cartridge	
063	No cartridge installed	
064	End of tape, cartridge full	Streamer errors
065	Streamer drive error	
066	Unidentified exception	
067	Illegal command to streamer	
070	PROM checksum error	
071	RAM error	
072	CTC error	
073	DMACTRL error	selftest error
074	VCO error	
075	FLOPPY controller error	
076	Streamer Data register error	
077	ND-100 register error	

D.32 SWITCHES ON THE 8-TERMINAL INTERFACE (3107)

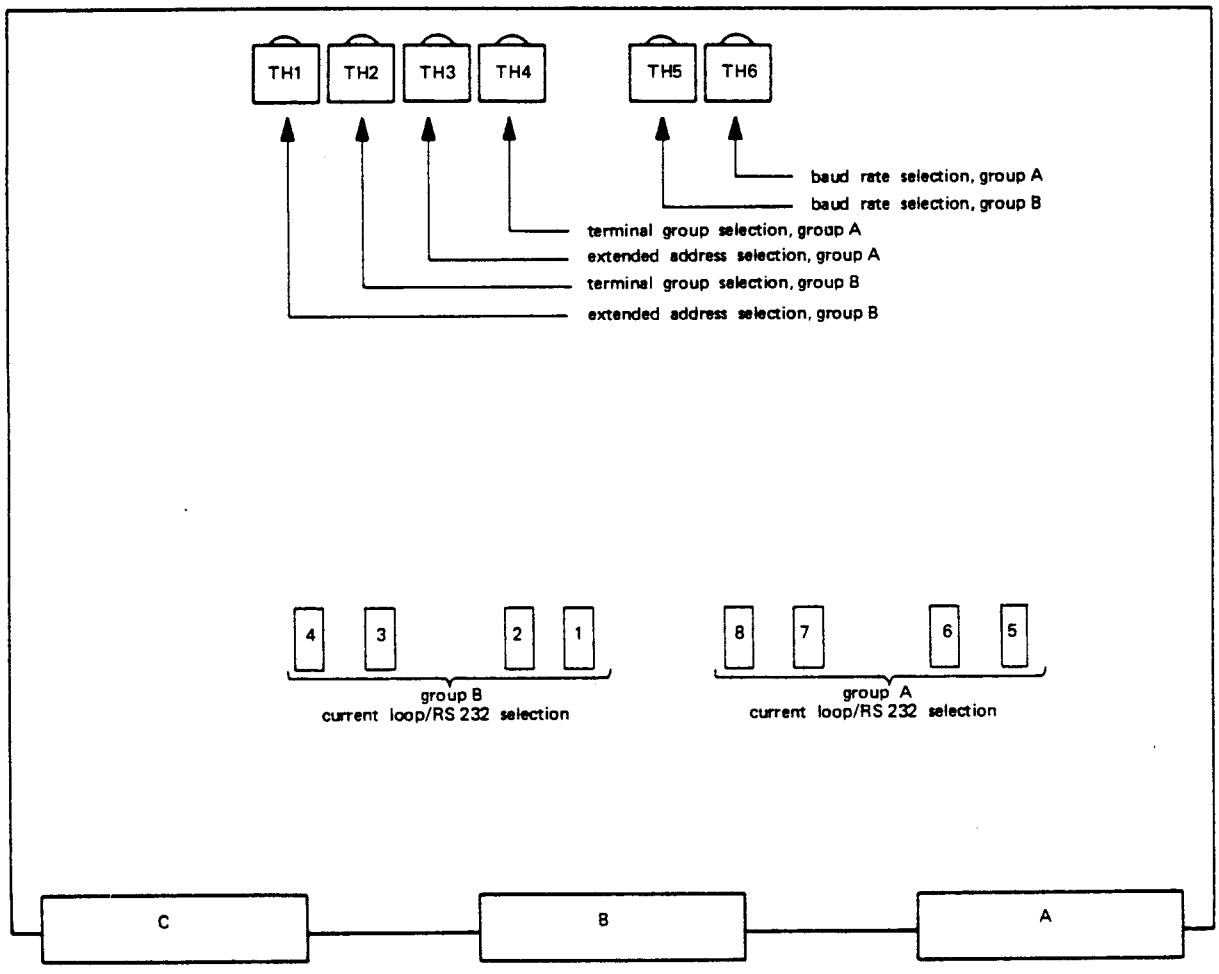


Figure D.32.1: The 8-Terminal Interface (3107)

D.32.1 Extended Address Selection (TH1 and TH2)

These thumbwheels have only two valid positions:
0 — normal address range
1 — extended address range

Extended address means terminals 65 to 128. These terminals are only reached by using the ND-100 instruction IOXT.

D.31.2 Terminal Group Selection (TH2 and TH4)

Normal address range selected (terminals 1-64)

thumbwheel setting	group	terminals	device register address range (octal)
0	0	1 - 4	300 - 347
1	1	5 - 8	340 - 377
2	2	9 - 12	1300 - 1337
3	3	13 - 16	1340 - 1377
4	4	33 - 36	640 - 677
5	5	37 - 40	1100 - 1137
6	6	41 - 44	1140 - 1177
7	7	45 - 48	1400 - 1437
8	8	49 - 52	1500 - 1537
9	9	53 - 56	1640 - 1677
10	10	57 - 60	1700 - 1737
11	11	61 - 64	1740 - 1777
12	12	17 - 20	200 - 237
13	13	21 - 24	240 - 277
14	14	25 - 28	1200 - 1237
15	15	29 - 32	1240 - 1277

extended address range selected (terminals 65-128)

thumbwheel setting	group	terminals	device register address range (octal)
0	0	65 - 68	140400 - 140437
1	1	69 - 72	140440 - 140477
2	2	73 - 76	140500 - 140537
3	3	77 - 80	140540 - 140577
4	4	81 - 84	140600 - 140637
5	5	85 - 88	140640 - 140677
6	6	89 - 92	140700 - 140737
7	7	93 - 96	140740 - 140777
8	8	97 - 100	141000 - 141037
9	9	101 - 104	141040 - 141077
10	10	105 - 108	141100 - 141137
11	11	109 - 112	141140 - 141177
12	12	113 - 116	141200 - 141237
13	13	117 - 120	141240 - 141277
14	14	121 - 124	141300 - 141337
15	15	125 - 128	141340 - 141377

D.32.3 Baud Rate Selection (TH1 and TH2)

thumbwheel setting	baud rate
0	110
1	150
2	300
3	2400
4	1200
5	1800
6	4800
7	9600
8	2400
9	600
10	200
11	134.5
12	75
13	50
14	100
15	100

D.32.4 Current Loop/RS232 Selection

There are 8 switches for selection of current loop or RS232, one for each terminal. Each switch has two positions:

0 — current loop
1 — RS232-C

D.33 SWITCHES AND INDICATORS ON THE PIOC EXPANDED (3108)

PIOC is an abbreviation for Programmable Input/Output Controller. For more details refer to 'PIOC Hardware Reference Manual, ND-865 & ND-867 ND-02.004'.

The new PIOC print, number 3108, is called PIOC/EXP./for 'expanded'. The added function is X21 Clear Detection in hardware for line 0 and 1.

The print is produced in two versions: PIOC/64 and PIOC/256 depending on the local memory size in K words. When equipped with 64 K bit memory chips the memory size is 128 K bytes. When 256 K bit chips are used, the memory size is 512 K bytes.

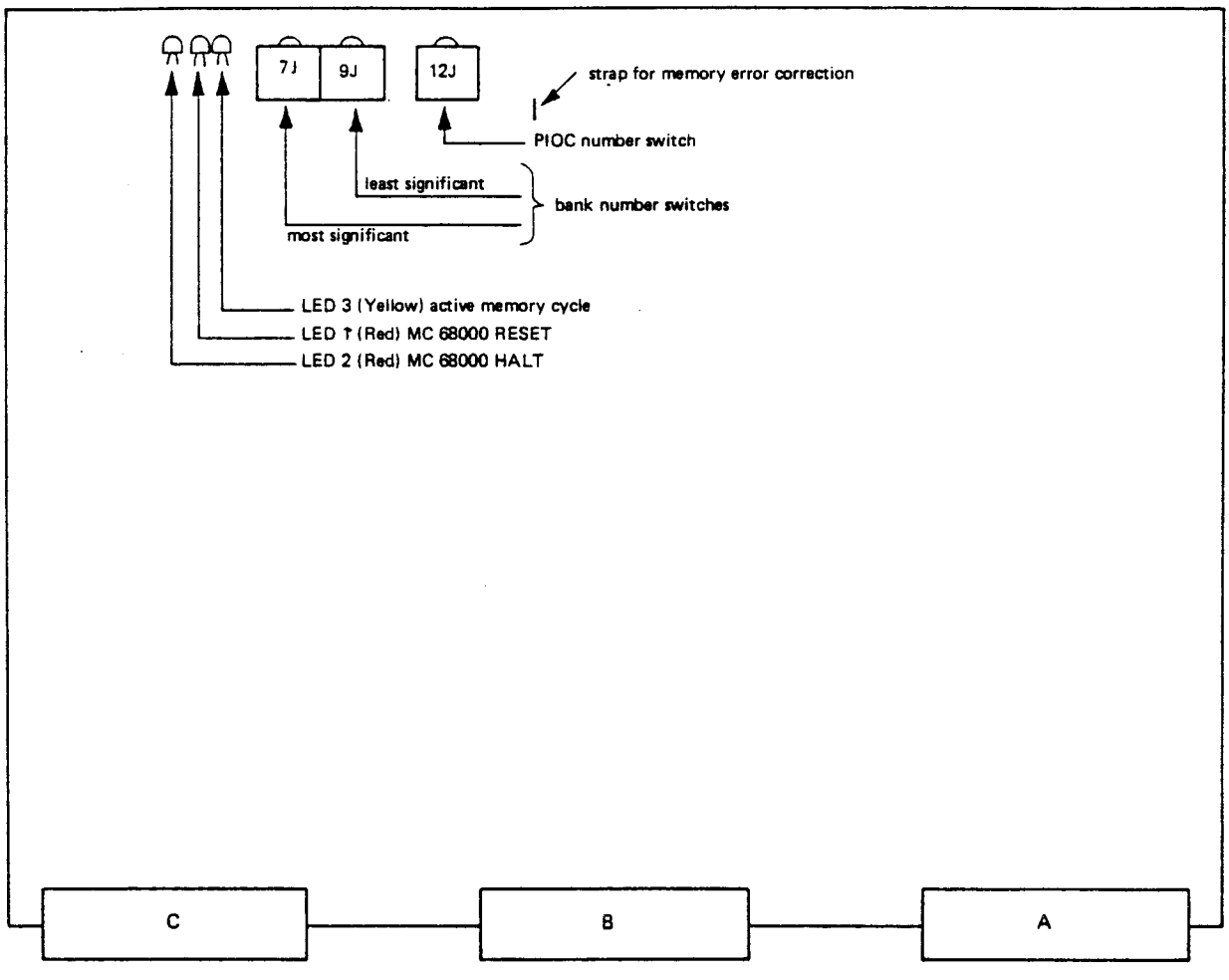


Figure D.33.1: The PIOC Expanded (3108)

D.33.1 PIOC Number Setting (Switch 12J)

PIOC number	device number (octal)	ident code (octal)
0	140020	140002
1	140024	140003
2	140030	140004
3	140034	140005
4	140040	140006
5	140044	140007
6-15	not used	not used

D.33.2 Bank Number Selection (Switches 7J and 9J)

D.33.2.1 Bank Number Selection on PIOC/64

The bank number is found by using the formula:
bank number = (7J setting * 16) + 9J setting

Example:
7J is set to 3, 9J is set to 4
bank number = (3 * 16) + 4 = 48 + 4 = *52*

D.33.2.2 Bank Number Selection on PIOC/256

On PIOC/256 the upper thumbwheel, 7J, is used to select megaword number as on the PIOC/64. The lower two bits of the lower thumbwheel (position 9J) are used differently from the PIOC/64.

When the lower bank select thumbwheel is set to an odd number, only the lower half of the PIOC/256's local RAM can be reached from the ND-100 CPU or from the DMA devices. The upper 256 K bytes in PIOC/256 will be private. The lowest bank number in PIOC/256 seen from the ND-100 bus is a multiple of two.

When the lower bank select thumbwheel is set to an even number, the lower two bank number bits from the ND-100 backplane are ignored. This means that the lowest bank number in PIOC/256 is a multiple of four.

ND-100 bus small window:	0-128 KW	(thumbwheel 9J = odd number)
ND-100 bus large window:	0-256 KW	(thumbwheel 9J = even number)
68000	0-256 KW	

lower (9J) thumbwheel setting	K words	ND-100 hex address	ND-100 octal address	
0	0-256	0-3FFFF	0-7777777	
2	"	"	"	
4	256-512	40000-3FFFF	1000000-1777777	
6	"	"	"	large
8	512-768	80000-BFFFF	2000000-27777777	window
10	"	"	"	
12	768-1024	C0000-FFFFF	3000000-3777777	
14	"	"	"	
1	0-128	0-1FFFF	0-377777	
3	128-256	20000-3FFFF	400000-777777	
5	256-384	40000-5FFFF	1000000-1377777	
7	384-512	60000-7FFFF	1400000-1777777	small
9	512-640	80000-9FFFF	2000000-2377777	window
11	640-768	A0000-BFFFF	2400000-2777777	
13	768-896	C0000-DFFFF	3000000-3377777	
15	896-1024	E0000-FFFFF	3400000-3777777	

The upper thumbwheel selects one out of 16 areas of 1 megaword each. The table above gives the offset from the megaword setting.

D.33.3 Lowest Address Selection/Reading (Switches 7J and 9J)

The lowest address in PIOC seen from ND-100 is set/found by using the following formula:
lowest address = (7J setting * 2048 K) + (9J setting * 128 K)

Example:
7J is set to 3, 9J is set to 4
lowest address = (3 * 2048 K) + (4 * 128 K) = *6656 K*

D.33.4 The LED Indicators

LED 3 (Yellow) is lit when a memory cycle is performed in the PIOC memory either by the MC 68000, ND-100 or internal PIOC DMA transfer.

LED 1 (Red) is lit when the RESET signal is given to the MC68000. RESET (LED 1) is lit after pushing the ND-100 MCL button and is not lit when MC68000 executes the programs.

LED 2 (Red) is lit when the HALT (stop) is given to MC68000, and it is lit after the ND-100 MCL button is pushed. The LED is not lit when the MC68000 executes programs.

D.34 SWITCHES ON THE 8-TERMINAL INTERFACE WITH BUFFER (3111)

This interface is similar to the 3013 and the 3107, except that it has input/output FIFO buffers.

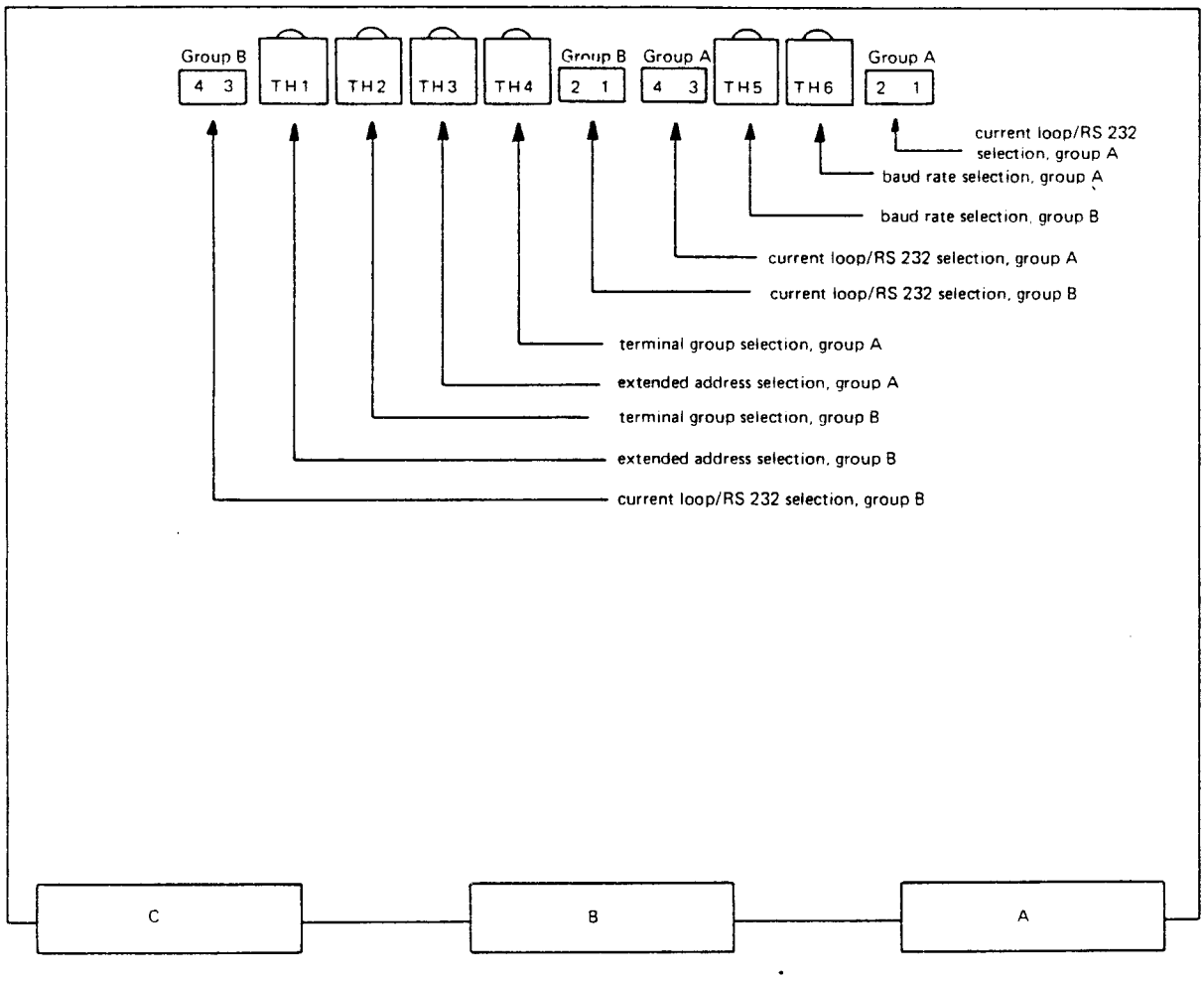


Figure D.34.1: The 8-Terminal Interface with Buffer (3111)

D.34.1 Extended Address Selection (TH1 and TH2)

These thumbwheels have only two valid positions:

0 - normal address range
1 - extended address range

Extended address means terminals 65 to 128. These terminals are only reached by using the ND-100 instruction IOXT.

D.34.2 Terminal Group Selection (TH2 and TH4)

Normal address range selected (terminals 1 - 64)

thumbwheel setting	group	terminals	device register address range (octal)
0	0	1 - 4	300 - 337
1	1	5 - 8	340 - 377
2	2	9 - 12	1300 - 1337
3	3	13 - 16	1340 - 1377
4	4	33 - 36	640 - 677
5	5	37 - 40	1100 - 1137
6	6	41 - 44	1140 - 1177
7	7	45 - 48	1400 - 1437
8	8	49 - 52	1500 - 1537
9	9	53 - 56	1640 - 1677
10	10	57 - 60	1700 - 1737
11	11	61 - 64	1740 - 1777
12	12	17 - 20	200 - 237
13	13	21 - 24	240 - 277
14	14	25 - 28	1200 - 1237
15	15	29 - 32	1240 - 1277

Extended address range selected (terminals 65 - 128)

thumbwheel setting	group	terminals	device register address range (octal)
0	0	56 - 68	140400 - 140437
1	1	69 - 72	140440 - 140477
2	2	73 - 76	140500 - 140537
3	3	77 - 80	140540 - 140577
4	4	81 - 84	140600 - 140637
5	5	85 - 88	140640 - 140677
6	6	89 - 92	140700 - 140737
7	7	93 - 96	140740 - 140777
8	8	97 - 100	141000 - 141037
9	9	101 - 104	141040 - 141077
10	10	105 - 108	141100 - 141137
11	11	109 - 112	141140 - 141177
12	12	113 - 116	141200 - 141237
13	13	117 - 120	141240 - 141277
14	14	121 - 124	141300 - 141337
15	15	125 - 128	141340 - 141377

D.34.3 Baud Rate Selection (TH5 and TH6)

thumbwheel setting	baud rate
0	110
1	150
2	300
3	2400
4	1200
5	1800
6	4800
7	9600
8	2400
9	600
10	200
11	134.5
12	75
13	50
14	100
15	100

Baud rate 19200 is available when ordering ND 274.

D.34.4 Current Loop/RS 232 Selection

There are 8 switches for the selection of current loop or RS 232, one for each terminal. Each switch has two positions:

0 - current loop
1 - RS 232-C

D.35 SWITCHES ON THE PLOTTER/PRINTER DMA INTERFACE (3114)

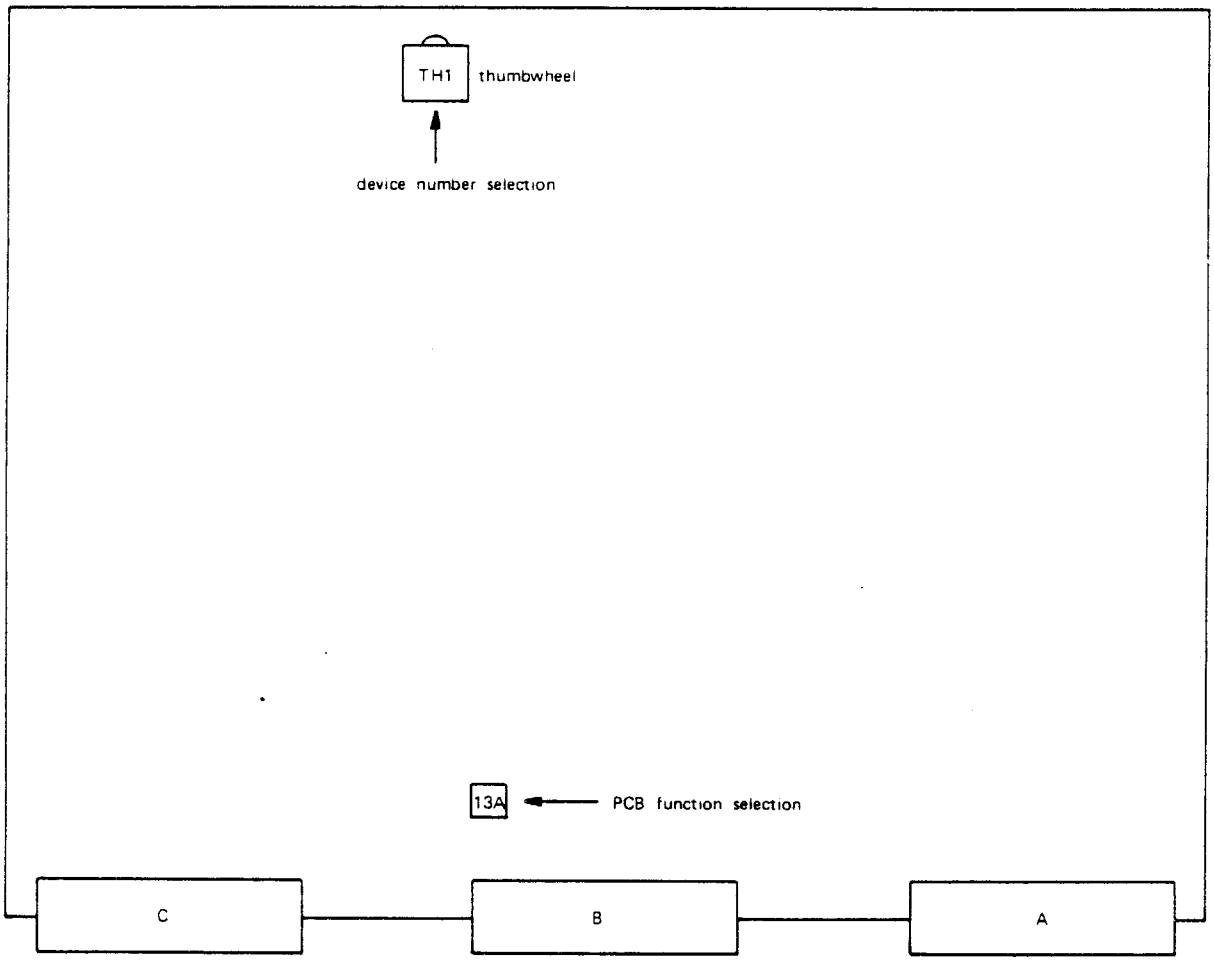


Figure D.35.1: The Printer/Plotter DMA Interface (3114)

D.35.1 PCB Function Selection (DIP Switch in POs. 13A)

PCB = Printed Circuit Board

switch 4	3	2	1	function
on	x	x	on	Versatec, A-connector, TTL
on	x	x	off	Printer, A-connector, TTL
off	x	x	on	Versatec, B-connector, V80

D.35.2 Device Number Selection (Thumbwheel)

thumbwheel setting	device reg. address range (octal)	ident code (octal)	device name
0	142170-412177	140230	Printer/plotter no.1
1	142200-142207	140231	Printer/plotter no. 2
2	142210-142217	140232	Printer/plotter no.3
3	142220-142227	140233	Printer/plotter no.4
4	600-607	4	Printer/plotter no.1 (old)
5	1500-1607	14	Printer/plotter no.2 (old)
6-15	not used		

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 bits wide, and information is sent in packets consisting of 2-6 bytes. The first byte is a command code telling what the succeeding bytes mean. 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 the 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 micro program believe that there is no panel present, and then it functions as explained in appendix E.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:

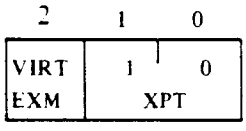
Bit 15 = Panel Present

At each clock interrupt (normally every 20ms) the microprogram reads 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 be any panel interrupt. Therefore, the MOPC routine is entered each 20ms in the absence of a Panel Processor.

APPENDIX F
MICROPROGRAMMABLE REGISTERS ON MEMORY MANAGEMENT

F.1 WRITE ONLY REGISTERS

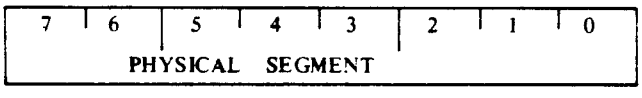
IR NO. 21 Examine Mode



Bit 2 = 1 Enable virtual examine via page table no. XPT
= 0 Physical examine with address bits 16-23 from segment register.

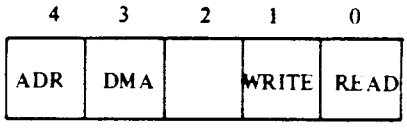
Bits 0-1 = XPT, Page table number used at virtual examine.

IR NO. 22 Segment Register



Bits 0-7 = PSEG. Number of 64 K segment at physical examine.
PSEG. 0-7 are used directly as memory address 16-23.

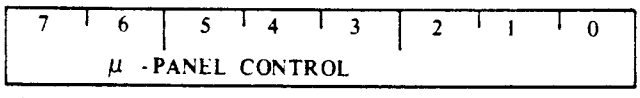
IR NO. 23 Select Memory Bus Register



Selects the 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

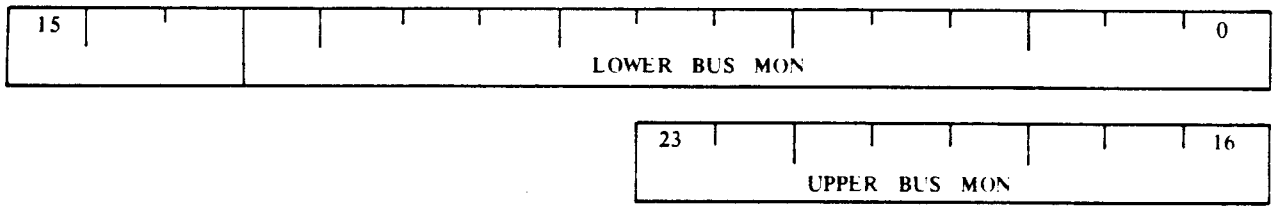
IR NO. 40 Panel Control Register



Bit 0-7 = μ PANC. Used by panel processor for display information and control. See appendix E.

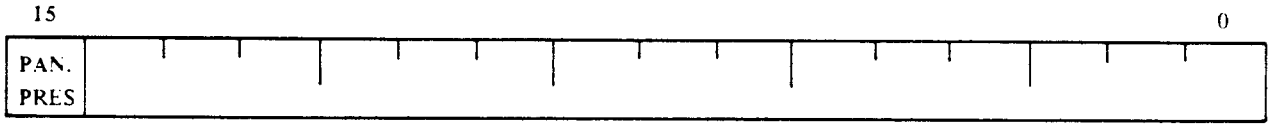
F.2 READ ONLY REGISTERS

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.

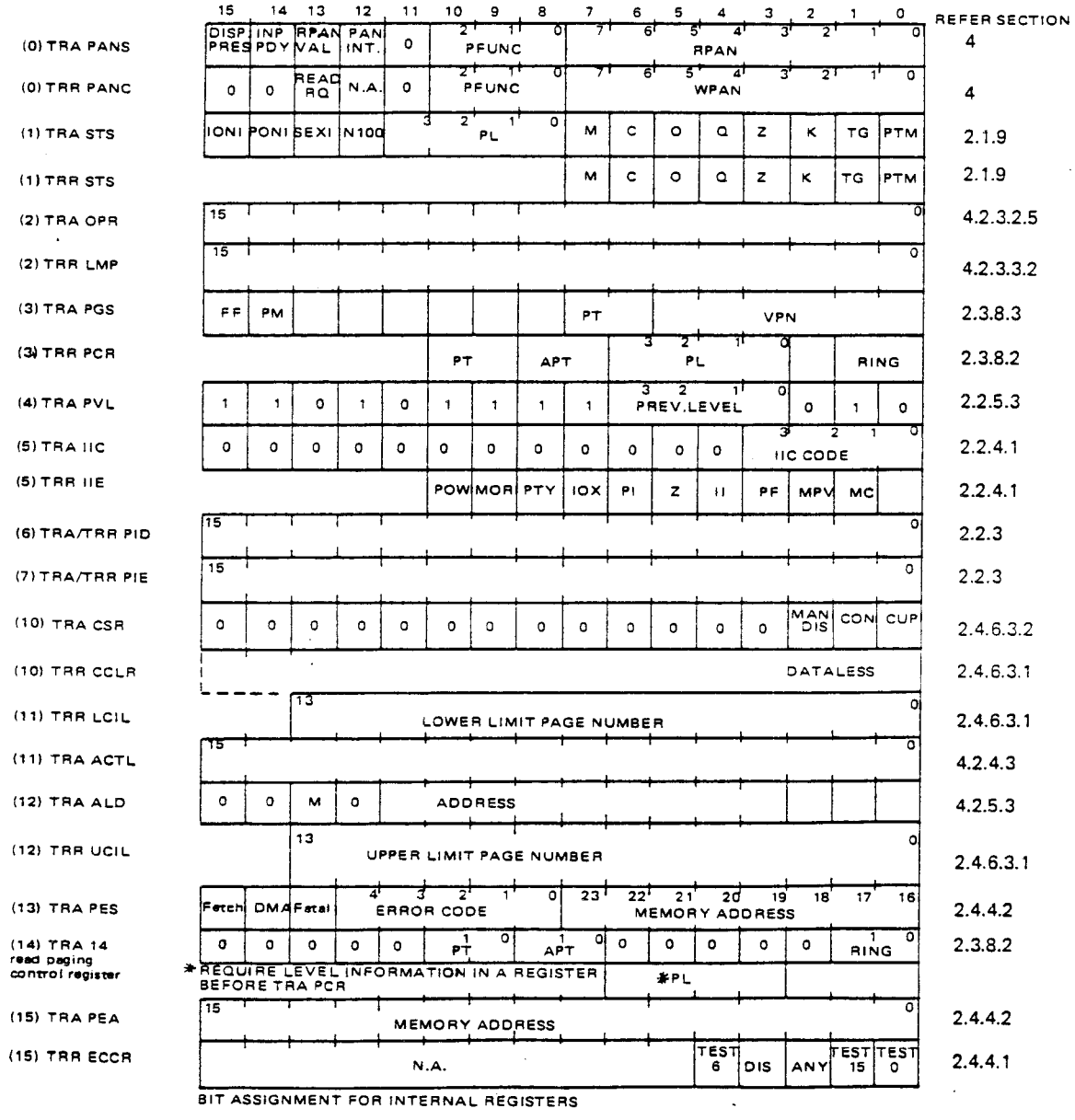
= 0 Panel not present. MOPC is triggered by 20ms clock interrupt.

Reading of panel status also clears panel request.

APPENDIX G

INTERNAL REGISTERS AND THEIR BIT ASSIGNMENT

UCILR	12	Upper cache inhibit limit register
PES	13	Parity error status
PGC	14	Paging control register read on specified level
PEA	15	Parity error address



BIT ASSIGNMENT FOR INTERNAL REGISTERS

APPENDIX H

OPERATOR'S COMMUNICATION INSTRUCTION SURVEY

H.1 CONTROL FUNCTIONS (Do 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 Operator's Panel 'Switches' from n to z

H.2 DISPLAY FUNCTIONS (Affect 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 ND-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 Contents of memory address x from n to z. x is 24 bits at Physical and 16 bits at Virtual Examine.
xxxxxx < yyyyyy ⏎	Dump Contents of memory from address x to address y. Select 64 K area of last Examine.

Registers:

xxRy/ nnnnnn zzzzzz ⏎	Examine and Change Contents 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	Contents of User Register are n
OPR/nnnnnn zzzzzz ⏎	Change Operators Panel "Switches" from n to z

Internal Registers:

Ixx/ nnnnnn	Contents of Internal Register No. x are n
	x (4 bits): 0 = PANS 1 = STS 2 = OPR 3 = PGS 4 = PVL 5 = IIC 6 = PID 7 = PIE 10 = CSR 11 = ACTL 12 = ALD 13 = PES 14 = PGC 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 = UNCIL 15 = ECCR
IRD ⏎	Dump Internal Registers 0 -15 (only in STOP)
xx < yyRDE ⏎	Dump Scratch Registers from level x to level y

Deposit Rules:

Contents are only changed by zzzzzz ⏎ in STOP mode and by zzzzzz ⏎ in STOP or RUN mode.

Contents are 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 J

ASCII CHARACTER SET (ANSI X3.4 - 1968)

Octal		000	010	020	030	040	050	060	070	100	110	120	130	140	150	160	170
Decimal		000	008	016	024	032	040	048	056	064	072	080	088	096	104	112	120
A	0	NUL	BS	DLE	CAN		(	0	8	@	H	P	X	'	h	p	x
d	1	SOH	HT	DC1	EM	!	)	1	9	A	I	Q	Y	a	i	q	y
d	2	STX	LF	DC2	SUB	"	*	2	:	B	J	R	Z	b	j	r	z
e	3	ETX	VT	DC3	ESC	#	+	3	;	C	K	S	[	c	k	s	{
r	4	EOT	FF	DC4	FS	S	,	4	<	D	L	T	\	d	l	t	|
s	5	ENQ	CR	NAK	GS	%	-	5	=	E	M	U	]	e	m	u	}
	6	ACK	SO	SYN	RS	&	.	6	>	F	N	V	^	f	n	v	~
	7	BEL	SI	ETB	US	'	/	7	?	G	O	W	_	g	o	w	DEL

Example: Finding ASCII value for A

A (Octal) = 100 + 1 = 101_8
A (Decimal) = 64 + 1 = 65_{10}
adder

ASCII Chracter set (ANSI X3.4-1968)

NUL		Null	DLE	(CTRL+P)	Data Link Escape
SOH	(CTRL+A)	Start Of Heading	DC1	(CTRL+Q)	Device Control 1
STX	(CTRL+B)	Start Of Text	DC2	(CTRL+R)	Device Control 2
ETX	(CTRL+C)	End Of Text	DC3	(CTRL+S)	Device Control 3
EOT	(CTRL+D)	End Of Transmission	DC4	(CTRL+T)	Device Control 4
ENQ	(CTRL+E)	Enquiry	NAK	(CTRL+U)	Negative Acknowledge
ACK	(CTRL+F)	Acknowledge	SYN	(CTRL+V)	Synchronous Idle
BEL	(CTRL+G)	Bell	ETB	(CTRL+W)	End Of Transmission Block
BS	CTRL+H)	Backspace	CAN	(CTRL+X)	Cancel
HT	(CTRL+I)	Horizontal Tabulation	EM	(CTRL+Y)	End Of Medium
LF	(CTRL+J)	Line Feed	SUB	(CTRL+Z)	Substitute
VT	(CTRL+K)	Vertical Tabulation	ESC	(CTRL+[)	Escape
FF	(CTRL+L)	Form Feed	FS	(CTRL+\)	Fiel Separator
CR	(CTRL+M)	Carriage Return	GS	(CTRL+])	Group Separator
SO	(CTRL+N)	Shift Out	RS	(CTRL+^)	Record Separator
SI	(CTRL+0)	Shift In	US	(CTRL+ _)	Unit Separator
			DEL		Delete (Rubout)

The character CTRL+X (control-X) is produced by holding down the CTRL key and at the same time pressing the X character key.

APPENDIX I

ND-100 TECHNICAL SPECIFICATIONS

I.1 SPECIFICATIONS

Processor:

Microprocessor cycle time:	190 ns/150 ns (fast option)
CACHE memory size:	1 K/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.
Error checking and memory correction:	Add 40 ns if correction. 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

ND-100 CPU Crate, Rack Mountable:

Dimensions

Height:	400 mm
Width:	482 mm
Depth:	505 mm

Can be mounted in 19-inch cabinets of various heights, depending on configuration.

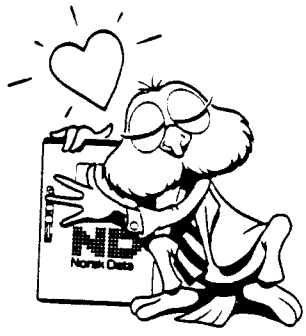
Power:	230V, range 198 - 264V (115V, range 90-132V) 47-63 Hz Max. 2 Amp. 230V

INDEX OF ABBREVIATIONS

MOR	MEMORY OUT OF RANGE	2.9.2.4
MOS	METAL OXIDE SEMICONDUCTOR	5.4
MPM	MULTIPORT MEMORY	5.10
MPV	MEMORY PROTECT VIOLATION	3.4.3
MSB	MOST SIGNIFICANT BIT	2.6.4
MSI	MEDIUM SCALE INTEGRATION	2.3.2
MTBF	MEAN TIME BETWEEN FAILURE	5.7.1
NA	NOT ASSIGNED	
O	STATIC OVERFLOW INDICATOR	2.5.2
OPR	OPERATOR'S PANEL SWITCH REGISTER	7.2.3.2.5
P	P-REGISTER (PROGRAM COUNTER)	2.5.1
PANC	PANEL CONTROL REGISTER	9.4
PANS	PANEL STATUS REGISTER	9.4
PC	PROGRAM COUNTER	2.4.2
PCR	PAGING CONTROL REGISTER	3.5.3
PEA	PARITY ERROR ADDRESS REGISTER	5.8.2
PES	PARITY ERROR STATUS REGISTER	5.8.2
PF	PAGE FAULT	3.4.3
PGU	PAGE USED	3.4.3
PI	PRIVILEGED INSTRUCTION	2.9.2.4
PID	PRIORITY INTERRUPT DETECT	2.9.2.3
PIE	PRIORITY INTERRUPT ENABLE	2.9.2.3
PIL	CURRENT LEVEL INDICATOR	2.9.2.3, 3.3.1
PIO	PROGRAMMED INPUT/OUTPUT	6.3
PK	PRIORY CODE	2.9.2.3
PM	PERMIT FLAGS	3.3.1
PONI	MEMORY MANAGEMENT ON INDICATOR	2.5.2
POW	POWER FAIL INTERRUPT	2.9.2.4
PPN	PHYSICAL PAGE NUMBER	3.3.1
PROM	PROGRAMMED READ ONLY MEMORY	2.3.2
PSR	PAGING STATUS REGISTER	3.5.4
PT	PAGE TABLE	3.2.3, 3.3.1
PTM	PAGE TABLE MODUS	2.5.2
PTS	PAGE TABLE SELECT FLAG	3.3.1
PTY	MEMORY PARITY ERROR	2.9.2.4
PVL	PREVIOUS PROGRAM LEVEL	2.9.2.3
Q	DYNAMIC OVERFLOW INDICATOR	2.5.2
R	RING	3.3.1
RADDR	READ INDIRECT ADDRESS	2.10.1
RAM	RANDOM ACCESS MEMORY	5.4.1
RAS	ROW ADDRESS STROBE	5.4.3.1
ROM	READ ONLY MEMORY	2.1
ROP	READ OPERAND	2.10.1
RPM	READ PERMITTED	3.4.3
RT	REAL TIME PROGRAM	3.3.3
SEXI	EXTENDED ADDRESS MODE INDICATOR	2.5.2
STO	STORE OPERAND	2.10.1
STS	STATUS REGISTER	2.5.1, 2.5.2
T	T-REGISTER (TEMPORARY)	2.5.1
TG	FLOATING ROUNDING INDICATOR	2.5.2
U	USED (VALID DATA) IN CACHE	
UCIL	UPPER CACHE INHIBIT LIMIT REGISTER	5.5.5.1
UL	UPPER LIMIT	5.6.2.1
V	VOLT	8
VA	VIRTUAL ADDRESS	3.1
VPN	VIRTUAL PAGE NUMBER	3.3.1
WCS	WRITABLE CONTROL STORE	9.3
WDA	WRITE DATA AND ADDRESS REGISTER	5.6.5
WIP	WRITTEN IN PAGE	3.4.3
WPM	WRITE PERMITTED	3.4.3
WT	WRITE THROUGH	5.5.2.3
X	X-REGISTER (POST INDEX REGISTER)	2.5.1
Z	ERROR INDICATOR	2.5.2, 2.9.2.4

********** SEND US YOUR COMMENTS!!! **********

Are you frustrated because of unclear information in this manual? Do you have trouble finding things? Why don't you join the Reader's Club and send us a note? You will receive a membership card — and an answer to your comments.



Please let us know if you

- find errors
- cannot understand information
- cannot find information
- find needless information

Do you think we could improve the manual by rearranging the contents? You could also tell us if you like the manual!

********** HELP YOURSELF BY HELPING US!! **********

Manual name: ND-100 Functional Description Manual number: ND-06.015.02

What problems do you have? (use extra pages if needed) _____

_____

_____

_____

_____

Do you have suggestions for improving this manual ? _____

_____

_____

_____

_____

_____

Your name: _____ Date: _____

Company: _____ Position: _____

Address: _____

_____

What are you using this manual for ? _____

_____

NOTE!
This form is primarily for documentation errors. Software and system errors should be reported on Customer System Reports.

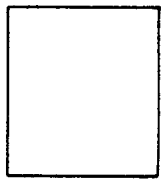
Send to:
Norsk Data A.S
Documentation Department
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

→

Norsk Data's answer will be found on reverse side

Answer from Norsk Data: _____

Answered by: _____ Date: _____



Norsk Data A.S
Documentation Department
P.O. Box 25 BOGERUD
N - 0621 OSLO 6 - Norway