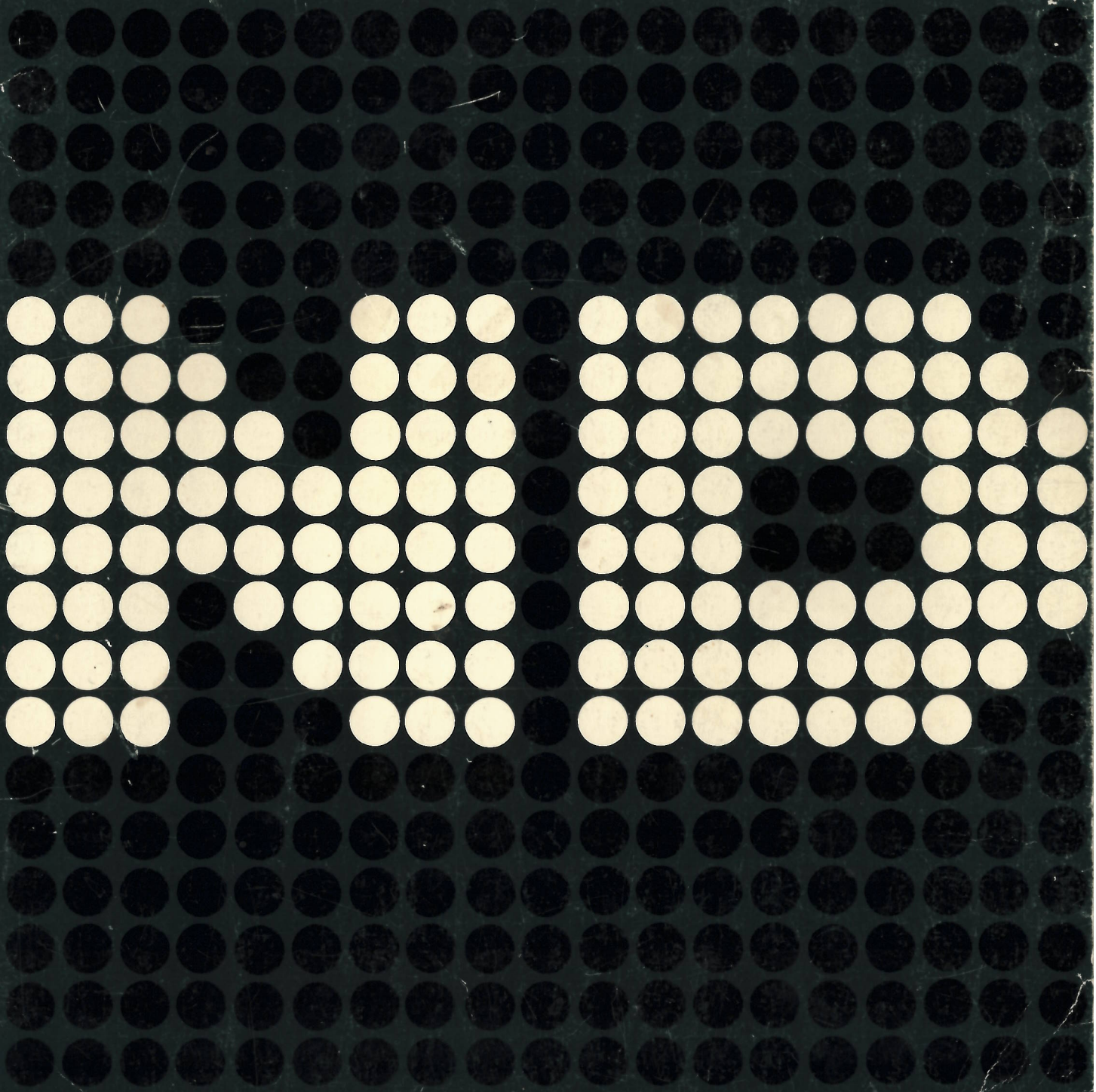
NORD–10/S

Functional Description

NORSK DATA A.S



NORD-10/S
Functional Description

REVISION RECORD

Revision	Notes
08/77	Original Printing
10/78	Revision A
	Typing errors and changed examples in Section II.8.
	We have also added a new section – Section II.7.9.

NORD-10/S – Functional Description
Publication No. ND-06.009.01

NORSK DATA A.S.
Lørenveien 57, Postboks 163 Økern, Oslo 5, Norway

MAIN CONTENTS

+ + +

Section:

I THE CPU

I.1 NORD-10/S Architecture
I.2 The Interrupt System
I.3 The Input/Output System
I.4 Operator's Communication
I.5 Operator's Panel
I.6 NORD-10/S Power Unit

II THE STORAGE SYSTEM

II.1 The Storage System
II.2 The Memory Hierarchy
II.3 NORD-10/S Storage System
II.4 MOS – Memory Operating Principles
II.5 Local Memory
II.6 Error Checking and Correction
II.7 Multiport Memory System
II.8 Memory Management System
II.9 Cache Memory
II.10 Power Fail/Automatic Restart

III SPECIFICATIONS

III.1 System Control Programming
III.2 NORD-10/S Instructions
III.3 Programming Specification for I/O Devices
III.4 Specifications

Appendix

A LIST OF ABBREVIATIONS

B NORD-10/S CARD ASSEMBLY

A detailed "Table of Contents" is at the beginning of each section.

INDEX TO FIGURES

+ + +

Figure:		*Page:*
Section I		
I.1.1	NORD-10/S Block Diagram	I–1–2
I.1.2	MMS Main Functional Blocks	I–1–5
I.1.3	Register Block	I–1–8
I.1.4	NORD-10/S Bus Structure	I–1–10
I.1.5	CPU – Control Section	I–1–12
I.1.6	CPU Block Diagram	I–1–14
I.1.7	Status - Bit Assignment	I–1–16
I.2.1	Register Block, Level Usage	I–2–4
I.2.2	Level Assignments	I–2–5
I.2.3	External Interrupt System	I–2–7
I.2.4	Internal Interrupt System, Block Diagram	I–2–9
I.2.5	TRA PVL, Flow Diagram	I–2–19
I.3.1	Storage Interconnection	I–3–3
I.3.2	I/O Architecture, Block Diagram	I–3–5
I.3.3	I/O Card Crate, Physical Layout	I–3–5
I.3.4	I/O Crate Control	I–3–6
I.3.5	IOX <Device Address> Decoding	I–3–7
I.3.6	IOX Instruction Execution	I–3–8
I.3.7	Loading Sequence for Interface Registers	I–3–13
I.4.1	Binary Load Format	I–4–8
I.5.1	Operator's Panel – Physical Layout	I–5–2
I.5.2	Panel Data and Control	I–5–9
I.6.1	NORD-10/S Power Unit	I–6–2

Figure: *Page:*

Section II

II.1.1	A Simple Memory Model	II–1–2
II.2.1	Two Level Storage System	II–2–1
II.3.1	Multilevel Storage System	II–3–1
II.3.2	Storage Interconnection	II–3–3
II.3.3	A Simple Memory Model	II–3–4
II.3.4	Memory Information and Control - Principal	II–3–5
II.3.5	Memory Information and Control – Functional	II–3–6
II.3.6	Address Arithmetic – Input and Control	II–3–9
II.3.7	Address Arithmetic – Functional Operation	II–3–10
II.4.1	A Memory Cell	II–4–2
II.4.2	A Memory Chip – Functional Blocks	II–4–4
II.4.3	Read Cycle	II–4–5
II.4.4	Write Cycle	II–4–6
II.4.5	Refresh Cycle	II–4–6
II.5.1	Address Decoding	II–5–2
II.5.2	Error Correction	II–5–2
II.5.3	Memory Control	II–5–4
II.5.4	Data and Address Ready Timing	II–5–5
II.5.5	Refresh	II–5–7
II.6.1	Error Detection and Correction	II–6–3
II.6.2	Parity Generation and Detection	II–6–7
II.6.3	Parity Error Reporting	II–6–9
II.6.4	Memory Out of Range – Generation	II–6–10
II.7.1	Storage Interconnection	II–7–2
II.7.2	NORD-10/S Two-Processor System	II–7–3
II.7.3	Card Crate Principal Layout	II–7–5
II.7.4	Physical Memory Space	II–7–7
II.7.5	Bank Selection	II–7–9
II.7.6	Channel to Bank Address Conversion	II–7–10
II.7.7	Interleave Illustration	II–7–12
II.7.8	Control and Information Flow	II–7–13
II.7.9	Multiport Configuration – Example	II–7–17
II.7.10	Multiport Memory (1 card read)	II–7–18
II.7.11	BMPM Internal Bank Address	II–7–20
II.7.12	Channel Signals	II–7–22
II.7.13	Scan Error Log	II–7–23
II.7.14	Scan Port Status	II–7–24
II.7.15	Write ECC Register	II–7–25

Figure:		*Page:*
Section II, continued		
II.7.16	Execute Test Access	II–7–26
II.7.17	Bank Selection/Channel to Bank Address Conversion	II–7–28
II.7.18	Storage to Port Communication	II–7–29
II.7.19	Shifting of Address Bits	II–7–30
II.7.20	Banks as seen from NORD-10/S	II–7–31
II.7.21	Banks as seen from NORD-50	II–7–31
II.7.22	Data Channel Format	II–7–32
II.7.23	Syndrome Decoding	II–7–34
II.7.24	Data Flow on Write	II–7–35
II.7.25	Reading Data (no error)	II–7–36
II.7.26	Reading Data (Single Error)	II–7–37
II.7.27	Reading Data (Multiple Error)	II–7–37
II.7.28	I/O Interrupt System in NORD-10/S	II–7–39
II.7.29	Big MPM Data (1143) Data Flow	II–7–40
II.8.1	Virtual to Physical Address Translation	II–8–2
II.8.2	Memory Management Building Blocks	II–8–4
II.8.3	Virtual to Physical Address Mapping	II–8–6
II.8.4	Virtual to Physical Address Conversion	II–8–7
II.8.5	PCR and PT Usage	II–8–9
II.8.6	Page Table Assignments	II–8–10
II.8.7	Memory Protection	II–8–12
II.8.8	Virtual Address Space to Physical Memory	II–8–13
II.8.9	Ring Assignment	II–8–15
II.8.10	Page Index Table Usage, example	II–8–17
II.8.11	PSR Format	II–8–20
II.8.12	TRR PCR Flow Diagram	II–8–22
II.8.13	Shadow Memory Access	II–8–24
II.9.1	Hit Ratio versus Cache Size	II–9–2
II.9.2	Cache Memory Organization	II–9–4
II.9.3	Address and Data Flow	II–9–6
II.9.4	Cache Limits	II–9–8
II.9.5	Write Data, Write Cache	II–9–11
II.9.6	Read Data, Read Cache (Data found in cache)	II–9–12
II.9.7	Read Data, Write Cache (Data not found in cache)	II–9–13
II.10.1	Power Fail Sequence	II–10–2
II.10.2	Automatic Restart	II–10–3
Section III		
III.1.1	Internal Register Transfer Instructions	III–1–4

INDEX TO TABLES

+ + +

Table:		*Page:*
I.4.1	ALD Setting	I–4–11
II.3.1	Addressing Modes	II–3–8
II.3.2	Address Source vs. Mode	II–3–9
II.5.1	Physical Address Range	II–5–1
II.6.1	Correction Codes Usage	II–6–2
II.6.2	Control Code Generation	II–6–4
II.6.3	Syndrome Decoding	II–6–5
II.6.4	Parity Generation	II–6–6
II.8.1	Page Table Selection	II–8–8
II.8.2	PT's Address Range	II–8–23

DETAILED CONTENTS

+ + +

SECTION I

Section:		*Page:*
I.1	**NORD-10/S Architecture**	I–1–1
I.1.1	Introduction	I–1–1
I.1.1.1	Instruction Set	I–1–1
I.1.1.2	Addressing Modes	I–1–1
I.1.2	Functional Modules	I–1–3
I.1.2.1	The Memory System	I–1–3
I.1.2.1.1	Local Memory	I–1–3
I.1.2.1.2	The Multiport Memory System	I–1–3
I.1.2.1.3	Cache Memory	I–1–4
I.1.2.1.4	Memory Management System	I–1–4
I.1.2.2	The Input/Output System	I–1–5
I.1.2.3	The Interrupt System	I–1–6
I.1.3	Arithmetic, Registers and Control	I–1–6
I.1.3.1	Register Block	I–1–6
I.1.3.2	The Main Arithmetic	I–1–7
I.1.3.3	Control	I–1–7
I.1.4	Internal Communication	I–1–9
I.1.5	Control Section	I–1–11
I.1.5.1	General	I–1–11
I.1.5.2	μ-program Storage	I–1–11
I.1.5.3	Machine Instruction Execution	I–1–11
I.1.6	Data Flow and Main Arithmetic	I–1–13
I.1.6.1	The Main Arithmetic	I–1–15
I.1.7	Status	I–1–15

Section:		*Page:*
I.2	**The Interrupt System**	I–2–1
I.2.1	General	I–2–1
I.2.2	Interrupt or Polling	I–2–1
I.2.3	Interrupt Technique	I–2–2
I.2.3.1	Single Line Interrupt System	I–2–2
I.2.3.2	Multilevel Interrupt System	I–2–2
I.2.3.3	Vectored Interrupt	I–2–2
I.2.4	NORD-10/S Interrupt System	I–2–3
I.2.4.1	General Description	I–2–3
I.2.4.2	Program Level Usage	I–2–3
I.2.4.3	The Vector Mechanism	I–2–5
I.2.4.4	Functional Operation	I–2–6
I.2.5	The Internal Interrupt System	I–2–8
I.2.5.1	IID – Internal Interrupt Detect Register Bit Assignment	I–2–10
I.2.6	Initialization of the Interrupt System	I–2–13
I.2.7	Control of the Interrupt System	I–2–14
I.2.8	External Interrupt Identification	I–2–14
I.2.8.1	Device Priority	I–2–15
I.2.9	Interrupt Response Time	I–2–16
I.2.10	Leaving the Interrupting Level	I–2–16
I.2.11	Interrupt Program Organization	I–2–16
I.2.12	Internal Interrupt Identification	I–2–17
I.2.13	Programmed Interrupts	I–2–18
I.2.14	Use of the PVL Register	I–2–18
I.2.15	Problems	I–2–20
I.3	**The Input/Output System**	I–3–1
I.3.1	General	I–3–1
I.3.2	Direct Memory Access – DMA	I–3–2
I.3.2.1	Multiport Memory Access	I–3–2
I.3.3	Programmed Input/Output – PIO	I–3–4
I.3.4	NORD-10/S I/O Architecture	I–3–4
I.3.4.1	I/O Card Crate – Physical Layout	I–3–4
I.3.5	The Input/Output Instruction –IOX	I–3–6
I.3.5.1	General	I–3–6
I.3.5.2	Operation	I–3–7

Section:		*Page:*
I.3.6	Device Interrupt	I–3–9
I.3.6.1	Device Interrupt Identification	I–3–9
I.3.6.2	Analyzing the Interrupt	I–3–10
I.3.6.3	Interrupt Sequence	I–3–10
I.3.7	Control and Status	I–3–11
I.3.7.1	Format of Status and Control Word	I–3–11
I.3.8	Programming Input/Output	I–3–12
I.3.8.1	Input/Output Programming Without Use of Interrupt	I–3–12
I.3.8.2	Programming Input/Output Using Interrupt	I–3–14
I.3.9	Programming of a Direct Memory Access Channel	I–3–16
I.4	**Operator's Communication**	I–4–1
I.4.1	Functions	I–4–2
I.4.1.1	Start a Program	I–4–2
I.4.1.2	Memory Examine	I–4–2
I.4.1.3	Memory Deposit	I–4–3
I.4.1.4	Register Examine	I–4–3
I.4.1.5	Register Deposit	I–4–4
I.4.1.6	Internal Register Examine	I–4–4
I.4.1.7	Internal Register Deposit	I–4–5
I.4.1.8	Current Location Counter	I–4–6
I.4.1.9	Break Function	I–4–6
I.4.1.10	Bank Number	I–4–7
I.4.2	Bootstrap Loaders	I–4–7
I.4.2.1	Octal Format Load	I–4–7
I.4.2.2	Binary Format Load	I–4–8
I.4.2.3	Mass Storage Load	I–4–9
I.4.2.4	Automatic Load Descriptor	I–4–9
I.4.2.5	Examples	I–4–11
I.4.3	NORD-10/S Microprogrammed Memory Test	I–4–12
I.5	**Operator's Panel**	I–5–1
I.5.1	Panel Elements	I–5–1
I.5.2	18-bit Switch Register	I–5–1
I.5.3	18-bit Light Emitting Diode Register	I–5–1
I.5.4	16 Selector Push-buttons and 16 Associated Light Emitting Diodes	I–5–3

Section: *Page:*

I.5.5	Display Level Select	I–5–5
I.5.6	Control Buttons	I–5–5
I.5.6.1	Master Clear	I–5–5
I.5.6.2	Restart	I–5–5
I.5.6.3	Load	I–5–6
I.5.6.4	Decode Address	I–5–6
I.5.6.5	Set Address	I–5–6
I.5.6.6	Deposit	I–5–6
I.5.6.7	Enter Register	I–5–7
I.5.6.8	Single Instruction	I–5–7
I.5.6.9	Continue	I–5–7
I.5.6.10	Stop	I–5–7
I.5.7	Mode Indicators	I–5–8
I.5.8	Data and Control	I–5–8
I.6	**NORD-10/S Power Unit**	I–6–1
I.6.1	General	I–6–1
I.6.2	Description	I–6–3
I.6.2.1	Indicators	I–6–3
I.6.2.2	Switches	I–6–3
I.6.2.3	Adjustments	I–6–4
I.6.2.4	Input Voltage	I–6–5
I.6.2.5	Output Voltage	I–6–5
I.6.2.6	Fuse	I–6–5
I.6.2.7	Time Meter	I–6–5

I.1 NORD-10/S ARCHITECTURE

I.1.1 *INTRODUCTION*

NORD-10/S is a 16 bit general purpose computer. The maximum address space is 128 Kbytes without the Memory Management System (MMS) and 512 Kbytes with MMS. The Memory Management System offers an efficient paging system including extensive memory protection through a permit protect system and a ring protect system. A CACHE memory system is also available for increased performance.

I.1.1.1 *Instruction Set*

The NORD-10/S has a comprehensive instruction set which includes bit, byte, word, double word and triple word instructions. Integer arithmetical operations include single precision memory to register operations and double precision register to register multiply and divide.

The floating point instructions add, subtract, multiply and divde use a 32 bit mantissa and a 16 bit exponent (2 bits for sign of exponent and mantissa).

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

The NORD-10/S is microprogrammed and all instruction execution is in firmware using a 32 bit Read Only Memory – ROM. To maintain processor speed, the address arithmetic is implemented in hardware.

The ROM has provisions for user extensions of the NORD-10/S instruction set, by allowing generation of different entry points in the ROM.

I.1.1.2 *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

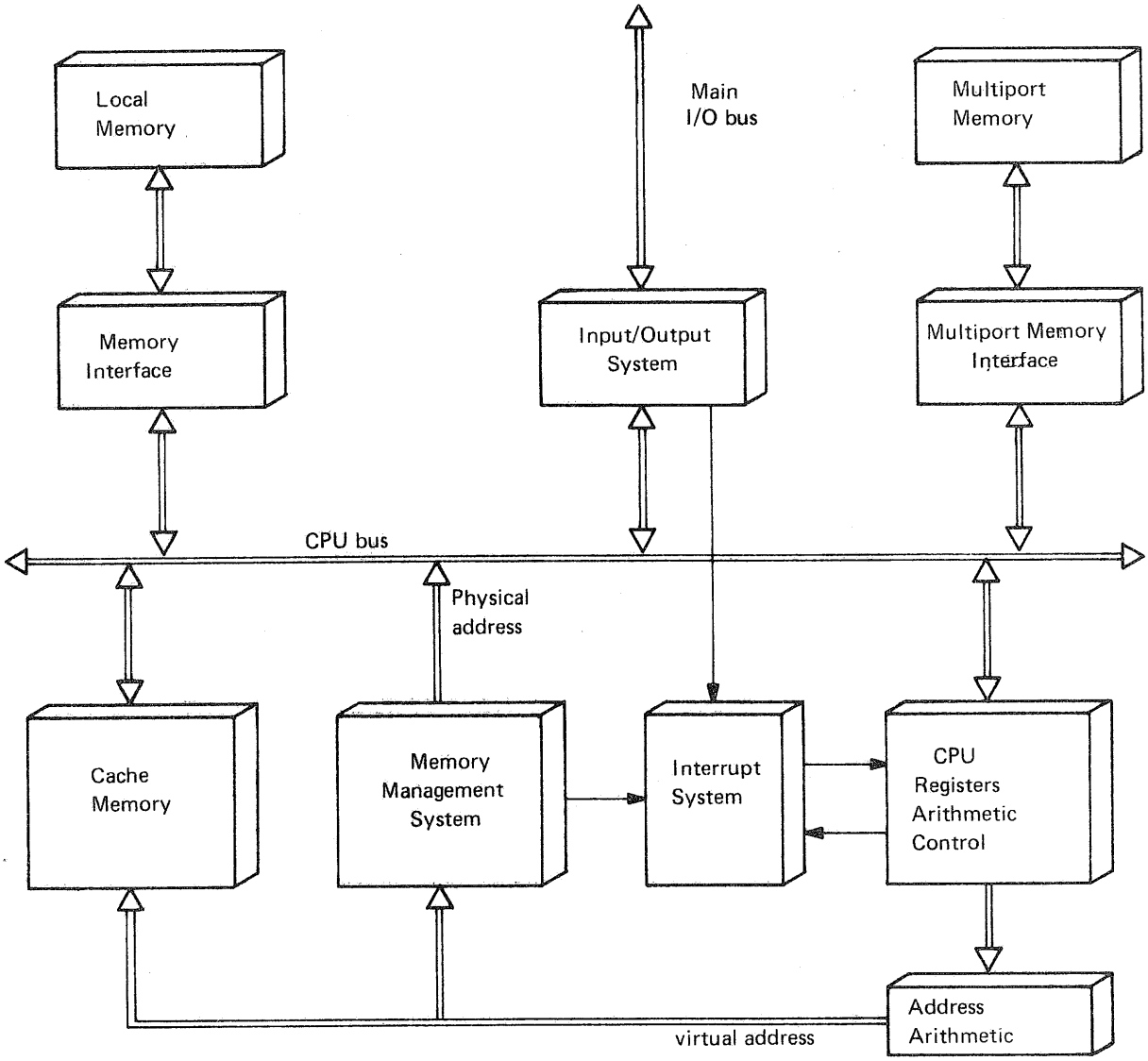


Figure I.1.1: *NORD-10/S Block Diagram*

I.1.2 *FUNCTIONAL MODULES*

Figure I.1.1 depicts NORD-10/S block diagram.

I.1.2.1 *The Memory System*

The Memory System is a flexible multi-level memory system.

The Memory System includes:

– 2 Kbytes CACHE memory
– Up to 256 Kwords local memory, or
– Up to 256 Kwords Multiport Memory System

I.1.2.1.1 Local Memory

The Local Memory is physically located next to the CPU where 8 slots are reserved. Maximum memory size is 512 Kbytes (8 modules of 64 Kbytes each). 16 Kbytes memory modules may also be used.

– A parity error occurring on an 18 bit module will be reported to the Internal Interrupt System.

– A single bit error occurring on a 21 bit module will be corrected and the error recorded.

- Multiple bit errors occurring on a 21 bit module will be reported to the internal interrupt system which interrupts the CPU.

I.1.2.1.2 The Multiport Memory System

For maximum flexibility a Multiport Memory System may be used. The Multiport Memory System consists of 4 independent ports, control and priority logic, and from 1 to 4 independent memory banks (each expandable up to 128 Kbytes).

If Direct Memory Access devices with high transfer rate are to be used, the Multiport Memory System should be employed to avoid cycle stealing from the CPU.

I.1.2.1.3 Cache Memory

The high speed CACHE memory will reduce the average memory access time significantly. The contents of the CACHE holds the most actual data and instructions to be processed.

The CACHE memory is organized as a 1K by 25 bit look-up table. A word in CACHE is identified with the main memory word of which it is a copy and by its main memory physical address – the physical page number.

I.1.2.1.4 Memory Management System

Refer to Figure I.1.2.

The Memory Management System includes two major subsystems:

- The Paging System
- The Memory Protection System

The paging system maps a 16 bit virtual address into an 18 bit physical address, extending the physical address space from 128 to 512 Kbytes. Four page tables of 64 words each, located in high-speed registers, reduce paging overhead to practically zero. Data and instruction pages may be allocated anywhere in memory without restriction. The page size is 1024 words.

The Memory Protection System may be divided into two subsystems:

- The Page Protection System
- The Ring Protection System

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

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

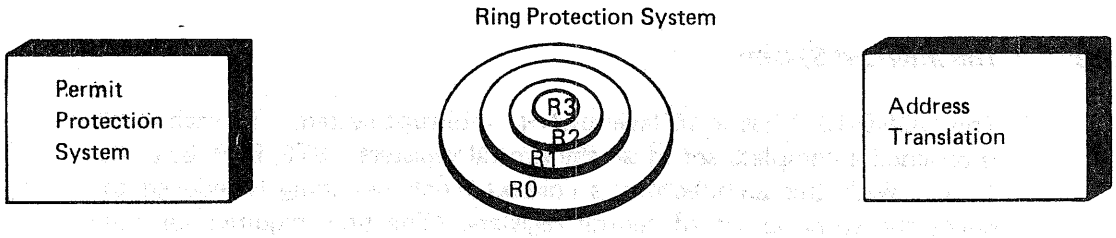


Figure I.1.2: *MMS Main Functional Blocks*

I.1.2.2 *The Input/Output System*

General:

The input/output system facilitates communication with external devices. The external devices can be input, output or mass storage devices.

Programmed Input/Output – PIO:

Program controlled input/output operates via the A register which implies that each word of input/output has to be programmed via this register. The PIO interfaces are always controlled by the CPU.

Direct Memory Access – DMA:

A Direct Memory Access – DMA – channel is used to obtain high transfer rates to and from main memory. CPU and DMA transfers may thus be performed simultaneously. The DMA channel is connected to main memory via the CPU bus on a cycle steal basis or for higher performance on a separate port on the Multiport Memory system. More than one DMA device may be active on the DMA channel at the same time,sharing the channel's total band width.

Bootstrap Loading:

Bootstrap loading is under microporgram control and makes available the following facilities:

- octal or binary load, from character oriented devices, i.e., from floppy disk, paper tape or communication line.
- system loading from block oriented devices, i.e., disk.

1.1.2.3 *The Interrupt System*

The NORD-10/S has a 16 level priority interrupt system. To each level is assigned a complete set of all the central registers: STS, D, P, B, L, A, T, X. With this architecture, a context block switching is reduced to select the working set of central registers. The time required for this operation is 1 μs.

All program levels may be activated by software. In addition, each of the levels 10, 11, 12 and 13 may be activated by 512 vectored I/O interrupts. An IDENT instruction is used to identify the interrupting device. Program level 15 may only have one I/O interrupt source. (Program level 15 is not used by standard NORD equipment or software, but is available for users who need immediate access to the CPU.)

The Internal Interrupt System will report 10 different internal conditions.

1.1.3 *ARITHMETIC, REGISTERS AND CONTROL*

1.1.3.1 *Register Block*

The CPU has 16 program levels, each level has the following 8 registers:

Number:

0 STS Status. This register holds different status indicators.

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

2 P 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.

3 B Base register or second index register. In connection with indirect addressing, it causes pre-indexing.

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

5 A This is the main register for arithmetical and logical operations together with operands in memory. The register is also used for CPU controlled I/O communication.

6 T Temporary register. In floating point instructions it is used to hold the exponent part.

7 X Index register. In connection with indirect addressing, it causes post-indexing.

Refer also to Figure I.1.3.

I.1.3.2 *The Main Arithmetic*

The main arithmetic consists of two identical 16 bit modules.

During all integer and logical operations only one arithmetic unit is active. For floating point operations, however, the two modules are connected to form a full 32 bit wide arithmetic.

I.1.3.3 *Control*

A simple time counter together with a sequential execution of the μ-instructions controls the operation of the CPU.

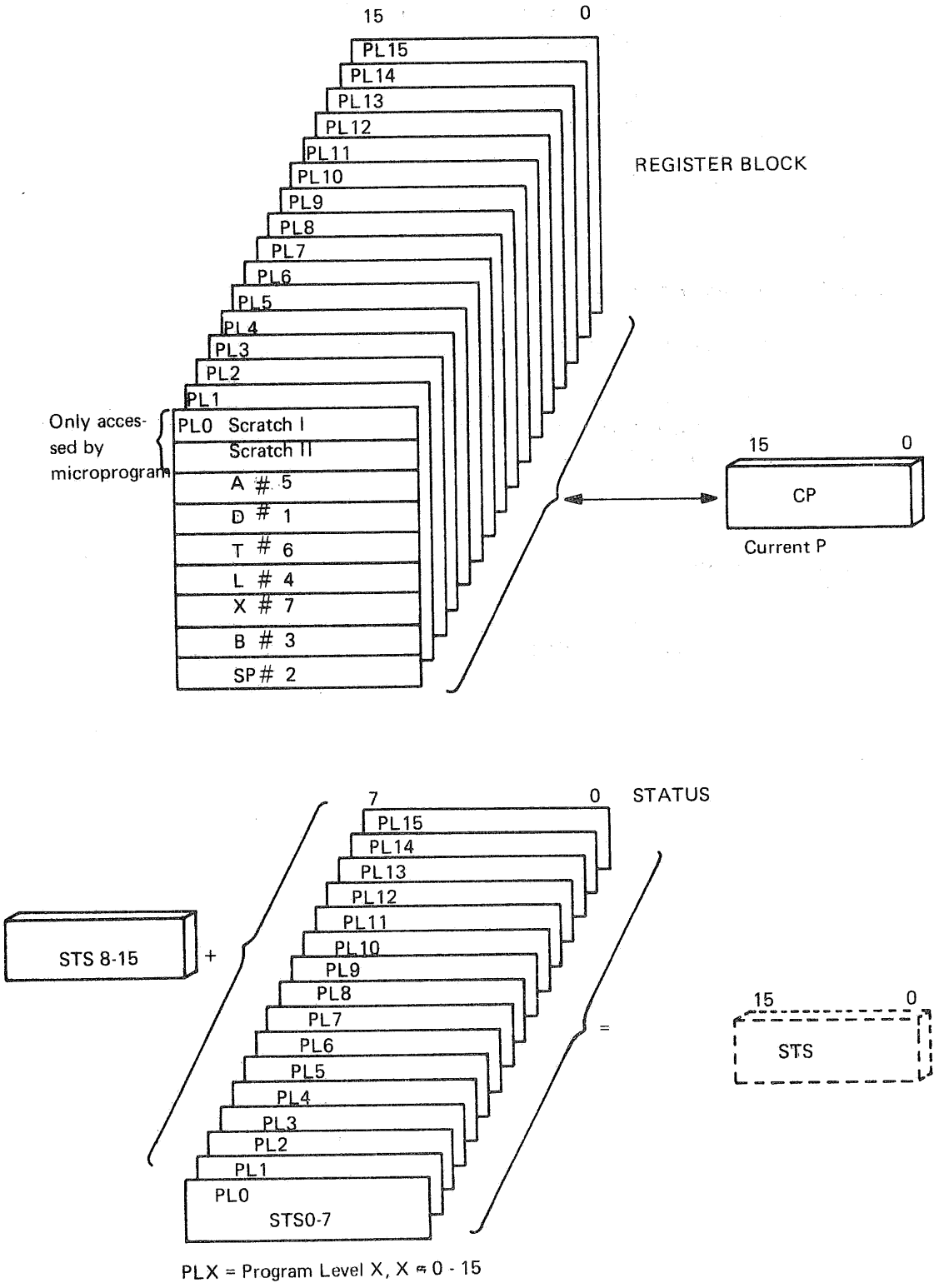


Figure I.1.3: *Register Block*

I.1.4 *INTERNAL COMMUNICATION*

For the following discussion refer to Figure I.1.4.

The internal communication is performed over the CPU bus system.

A bus is a highway for information where only one word of information may travel at the time, i.e., communication is time multiplexed. The different modules are provided with either transmitters or receivers or both in order to communicate over the bus.

The CPU bus consists of two buses. The Information Bus (IB) is for data communication and is 16 bits wide. The Memory Address Bus (MR) is physical memory address and is 18 bits wide.

Figure I.1.4 gives a block diagram presentation of the different functional modules and their inter-communication over the CPU bus.

The CPU with its central registers has a two-way communication with IB for loading and storing of the central registers. The output of the address arithmetic (16 bits) goes to the Memory Management System. In the case of a non-paging machine, the output of the address arithmetic will be transferred directly to MR via a *paging surrogate module.*

The Memory Management System is a source of the 18 bits physical memory address. The transformation from a 16 bits virtual address to an 18 bits physical address is implemented by the Page Table (PT) and will be described later.

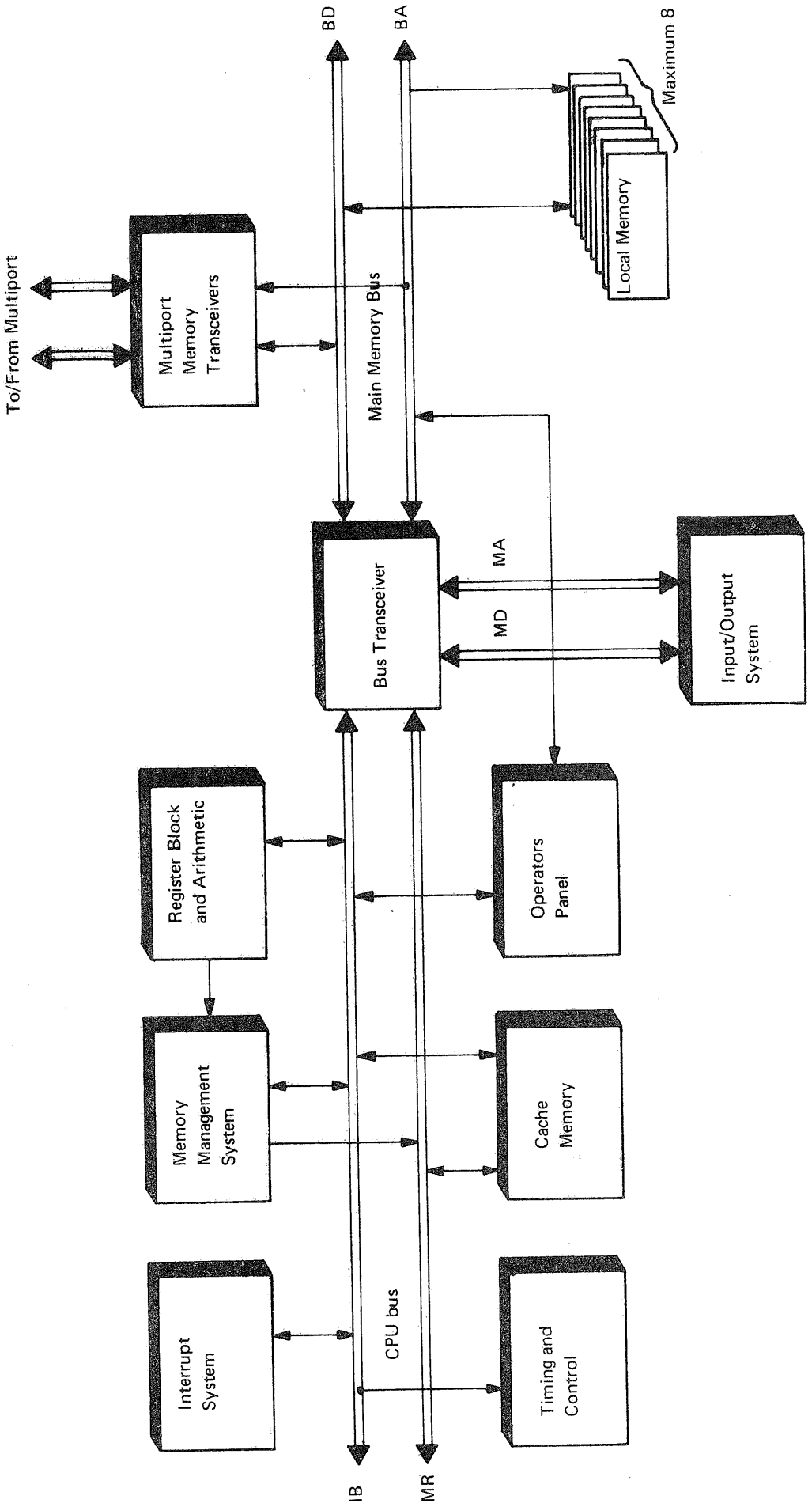
To read and write the PT, it is required with a two-way communication with IB.

Data can be stored in local memory or multiport memory or both. Both memories have a two-way communication with the information bus and will accept addresses from the address bus.

The I/O system has a two-way communication with IB for data to/from the I/O system and is a source for an 18 bits memory address during Direct Memory Access transfers.

The bus transceiver is an isolation stage between the internal bus system, the memory bus and the main I/O bus.

The Interrupt System has a two-way communication with the IB. Within the system, there are three registers which may be loaded from IB (Priority Interrupt Enable – PIE, Priority Interrupt Detect – PID, and Internal Interrupt Enable – IIE).



ND-06.009.01

Figure I.1.4: *NORD-10/S Bus Structure*

The operator's panel has a two-way communication with the internal data bus, IB, and the memory address bus, BA.

The timing and control will receive machine instructions from the internal information bus IB.

The cache memory has a two-way communication with the internal information bus, IB, for reading and writing data. Part of the internal address bus, MR, is used for look-up and the other part stored with the information as a directory.

I.1.5 *CONTROL SECTION*

I.1.5.1 *General*

The NORD-10/S is micro-programmed, and all instruction execution is implemented by a 32 bits Read Only Memory (ROM) and a microprocessor.

To maintain processor speed, the address arithmetic is not implemented in microprogram but in hardware. This means that the addresssing structure of NORD-10/S cannot be changed by rewriting the microprogram.

I.1.5.2 *μ-program Storage*

The μ-program is stored in a 1K x 32 bits Read Only Memory – ROM.

This memory is logically divided into the following sections:

- μ-programmed execution of NORD-10/S instruction reportoire
- μ-program operators panel driver
- MOPC – μ-programmed operator communication in stop mode.
- μ-programmed bootstrap loader.
- μ-programmed memory check.

I.1.5.3 *Machine Instruction Execution*

Refer to Figure I.1.5.

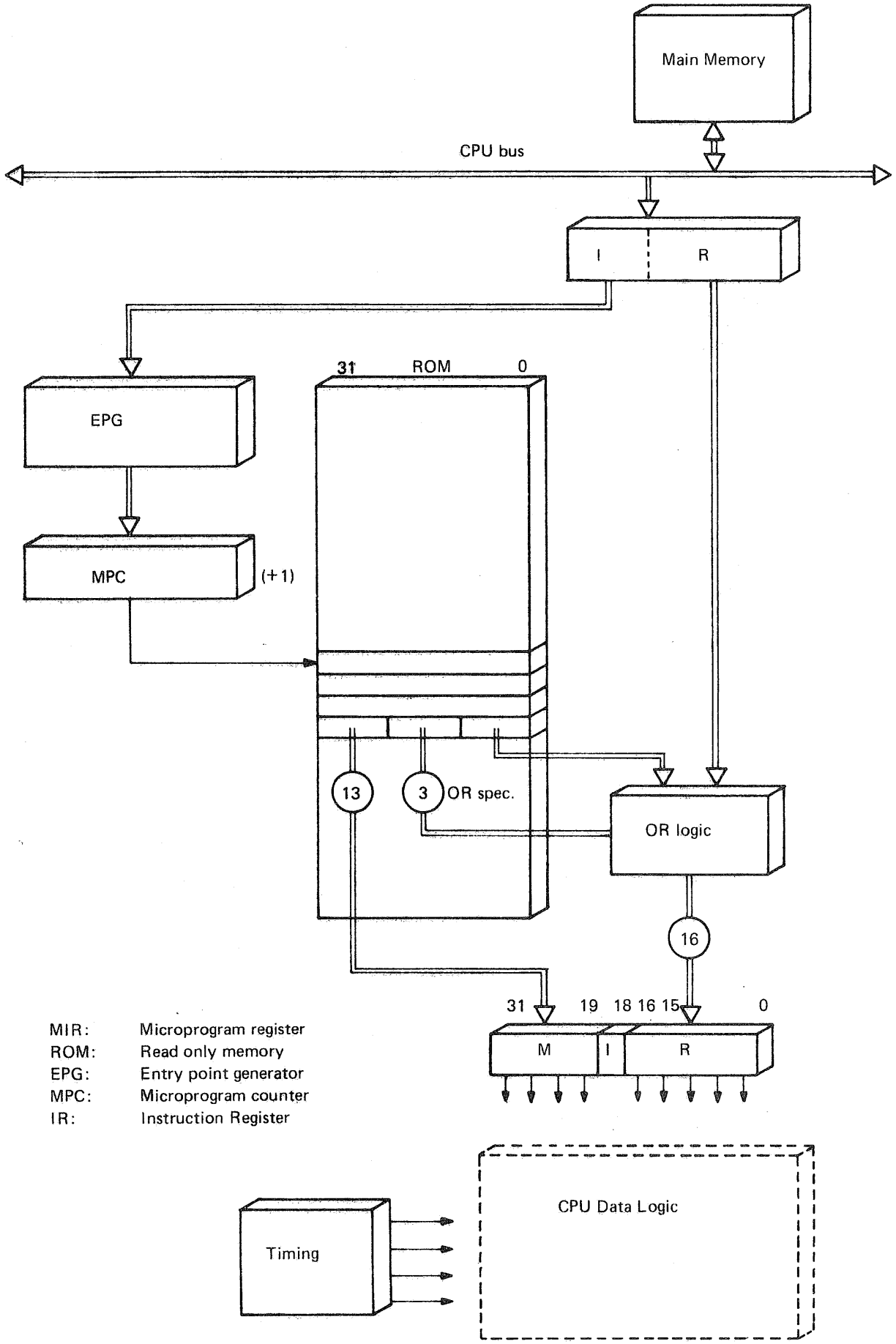


Figure I.1.5: *CPU — Control Section*

The machine instruction to be executed resides in memory. The program counter, CP, is enabled onto the address bus and a request is sent to memory. The instruction will be clocked into the instruction register IR. Since the machine instruction is executed by a number of μ-instructions residing in Read Only Memory (ROM) an instruction dependent address should be performed. This is the task of the entry point generator.

The address formed by the entry point generator is clocked into the microprogram counter (MPC). This address is sent to ROM and the first μ-instruction is clocked into the Micro Instruction Register (MIR). This register will, together with a timing module, control the operation of the CPU. The operation specified by one μ-instruction normally takes 260 ns. This time it is referred to as μ-cycle.

When the first μ-cycle has been completed the next has already been requested and is ready for execution. The microprogram counter is incremented to point at the next μ-instruction.

The number of μ-instructions to be executed is determined by the complexity of the machine instruction.

When a certain flag (CFC) is reached in the sequence of μ-instructions, the sequence is terminated and a request for the next machine instruction is made. The same sequence of events will be repeated.

I.1.6 *DATA FLOW AND MAIN ARITHMETIC*

Figure I.1.6 gives a block diagram presentation of the register block, main arithmetic and address arithmetic.

All operands fetched from memory will enter the processor via the H register.

All instructions fetched from memory will enter the control section via the instruction register. A copy of the instruction will also enter the H register. If the instruction is a memory reference instruction, the lower part of the instruction contains a memory displacement. This displacement is available to the address arithmetic from the lower part of the H register.

An indirect address enters the address arithmetic via the H register.

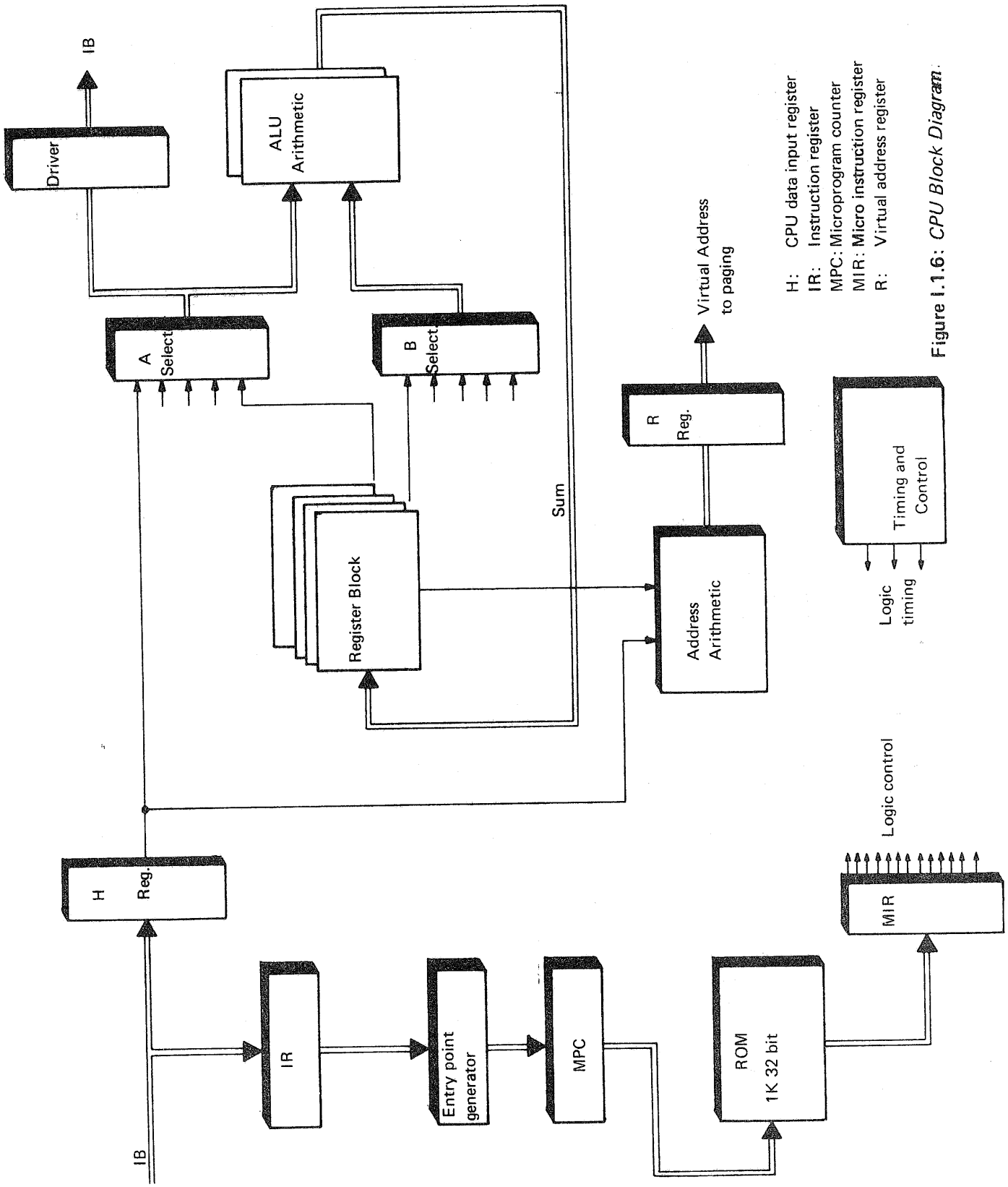


Figure I.1.6: *CPU Block Diagram*

I.1.6.1 *The Main Arithmetic*

The Main Arithmetic consists of two identical 16 bits modules. Each module can perform all the arithmetical and logical operations as specified in the instruction set.

During all integer and logical operations only one arithmetic unit is active. For floating point operations, however, the two modules are connected together to form a full 32 bits wide arithmetic.

All setting of status flip flops as static and dynamic overflow, etc. is done by the most significant arithmetic module.

The two selectors, A and B, are selecting the two operands for the arithmetic. Both operands are selected at the same time.

An operand may be any of the central registers (on the current level), zero, memory or some other registers (R, CP, SCR, SSTS).

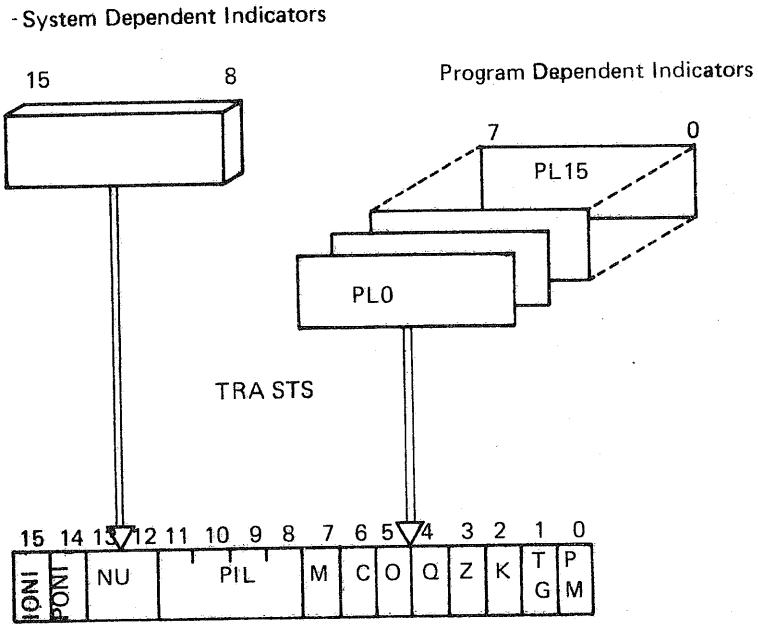
The B operand may be any of the 8 registers (except STS), zero or some internal registers.

The result of the arithmetic is always going to the register block to be written into A, A and D or A, D and T registers or any of the other programmable registers, depending upon the instruction type executed.

I.1.7 *STATUS*

Each of the 16 program levels has access to its own STS register. This part of the status register is only 8 bits wide. The upper part (8 bits) is common for all program levels.

Figure I.1.6 illustrates the bit assignment of the status register.



Bit 15 – IONI: Interrupt system on indicator
Bit 14 – PONI: Memory management on indicator
Bits 13-12: Not assigned
Bits 11-8: Current program level indicator

Figure I.1.7. *Status – Bit Assignment*

Bit 7 – M: Multi-shift link indicator. This indicator is used as temporary storage for discarded bits in shift instructions in order to ease the shifting of multiple precision words.

Bit 6 – C: Carry indicator. The carry indicator is dynamic.

Bit 5 – O: Status overflow indicator. This indicator remains set after an overflow condition until it is reset by program.

Bit 4 – Q: Dynamic overflow indicator.

Bit 3 – Z: Error indicator. This indicator is static and remains set until it is reset by program. The Z indicator may be internally connected to an interrupt level such that an error message routine may be triggered.

Bit 2 – K: One bit accumulator. This indicator is used by the BOP bit operations, instructions operating on one-bit data.

Bit 1 – TG: Rounding indicator for floating point operations.

BIT 0 – PM: Page index modus. Enables use of the alternate page table.

The lower 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.

1.2 THE INTERRUPT SYSTEM

1.2.1 *GENERAL*

What is interrupt? When used in context of a computer an interrupt can be defined as suspension of normal program execution in order to handle a sudden request for service. At completion of the interrupt service, the computer resumes the interrupted program from the point where it was interrupted. One CPU can handle many simultaneous activities, but a CPU can only be involved in one process at a time.

Interrupt may be generated from internal or external sources. Generally, an interrupt is generated when a change of condition occurs in an external device (external interrupt) or some kind of error condition occurring in the CPU itself (internal interrupt). Since, almost without exception, all external processes are asynchronous, the CPU, or operating system, cannot predict in which sequence the different processes should be carried out.

(By an asynchronous process [device] we mean a process [device] in which the speed of operation is not related to any frequency in the system to which it is connected.) An interrupt system will therefore put the different external processes in the correct order at the time of execution.

1.2.2 *INTERRUPT OR POLLING*

Is an interrupt system required?

When communicating with I/O devices both modes of operation are possible. Polling is a periodic interrogation of each I/O device connected to the input/output system to determine whether it requires servicing.

Each device must be polled (looked at) at a frequency of its maximum transfer rate, or higher.

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

Some processes have, at critical points, a demand for quick response. This can only be accomplished through a well organized interrupt system.

1.2.3 *INTERRUPT TECHNIQUE*

Existing interrupt system may be divided into 3 categories:

- Single line interrupt system
- Multilevel interrupt system
- Vectored interrupt

1.2.3.1 *Single Line Interrupt System*

In the single line interrupt system all I/O devices are OR-ed 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

1.2.3.2 *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

1.2.3.3 *Vectored Interrupt*

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

I.2.4 *NORD-10/S INTERRUPT SYSTEM*

I.2.4.1 *General Description*

The NORD-10/S interrupt system is a combination of:

— a multilevel interrupt system

and

— a vectored interrupt system.

This will then form a fast and flexible interrupt system. NORD-10/S also employs an internal interrupt system which momentarily reports internal CPU failures.

I.2.4.2 *Program Level Usage*

The NORD-10/S has a 16 level priority interrupt system. To each level is assigned a complete set of the central registers: D, P, B, L, A, T, X and STS. Thus, there are 16 x 8 = 128 central registers in the register block With this architecture, a context block swtiching amounts to selecting another working set of central registers.

This is automatically accomplished by the interrupt system in giving the PL, Program Level, register a new value. (Refer to Figure I.2.1.) A program can therefore normally run on any level.

Figure I.2.2 illustrates the program level usage.

All program levels may be activated by software. In addition, levels 10, 11, 12, 13 and 15 may be activated by I/O interrupts. Program level 14 is used by the Internal Interrupt System, which monitors error or trap conditions in the CPU.

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

Program level 13_{10} is wired up to the "Real Time Clock". This clock gives an interrupt every 20 ms if level 13 is enabled in PIE. The interrupt will be reset by a WAIT instruction.

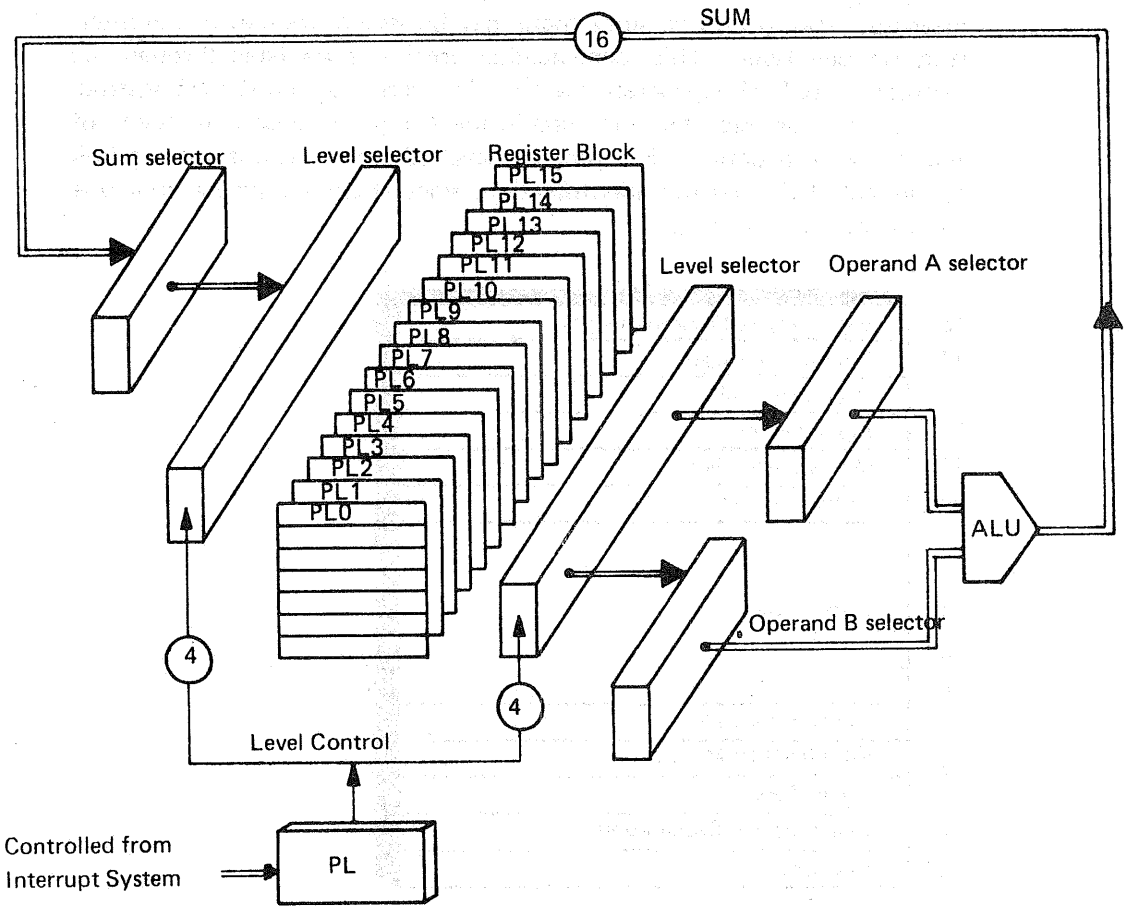


Figure I.2.1: *Register Block, Level Usage*

I.2.4.3 *The Vector Mechanism*

For program levels 10, 11, 12 and 13, it is required to identify the I/O interrupt when it occurs, since there may be several sources to the interrupt on one level. This identification process is available through the instruction IDENT <program level>. The result of the IDENT instruction is a unique identification code in the A register upon completion of the IDENT instruction. For program level 15, however, it is not required to do any I/O interrupt identification, since there should be only *one* source for this interrupt.

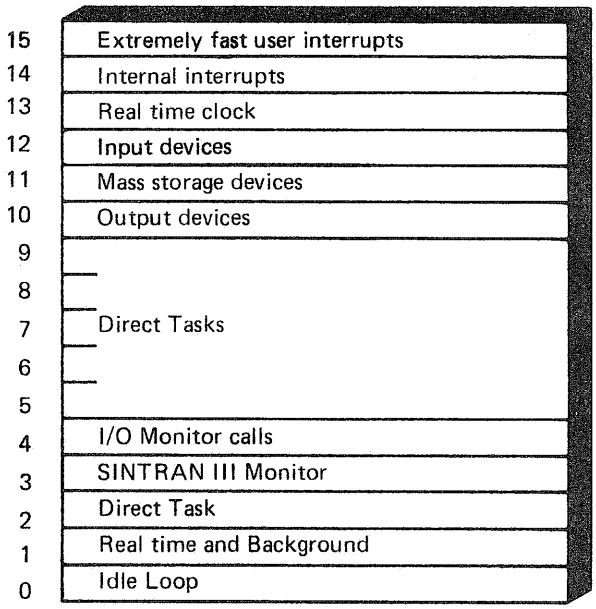


Figure I.2.2: *Level Assignments*

I.2.4.4 *Functional Operation*

Figure I.2.3 gives a block diagram presentation of the external interrupt system.

The Priority Interrupt Enable Register (PIE) is a 16 bits register which is programmable via the A register. The PIE register is used for *enabling* the different program levels.

The Priority Interrupt Detect (PID) register detects the physical interrupts. Any interrupt is "locked" into the PID register until reset by software (wait instruction). This register is also programmable via the A register.

The 16 AND gates are looking for coincidence between corresponding bits in PIE and PID. The Priority Encoder gives a 4 bits value of the highest bit position (priority) where corresponding bits in PID and PIE are set.

This code is referred to as PK – Priority Interrupt Code.

The current Program Level indicator (PL) always contains the value of the current program level.

The comparator continuously checks PK against PL. If PK $\neq$ PL the output of the comparator will be activated. If the interrupt system is active and the current instruction is completed, the interrupt signal will reach the central CPU.

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

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

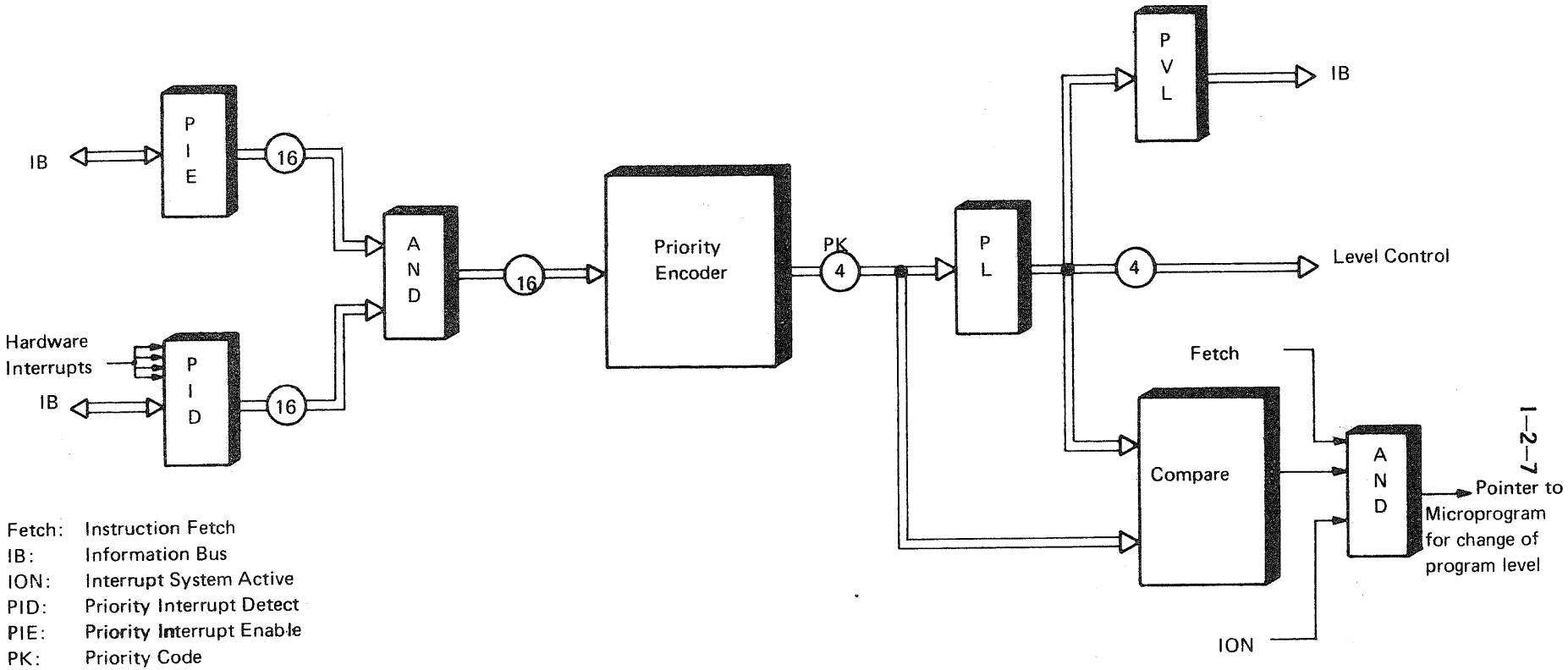
2. The program counter (CP) is copied to the saved program counter (SP) on the current level.

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

4. Copy the PK priority code into the PL (program level) register (new level). (The CPU has at this moment changed levels.)

5. Copy the $SP_{n.l}$ saved program counter on the new level to the CP (current program) counter.

6. Issue a fetch, i.e., ask for the first machine instruction on the new level.



Fetch: Instruction Fetch
IB: Information Bus
ION: Interrupt System Active
PID: Priority Interrupt Detect
PIE: Priority Interrupt Enable
PK: Priority Code
PL: Program Level
PVL: Previous Program Level

Figure I.2.3: *External Interrupt System*

I.2.5 *THE INTERNAL INTERRUPT SYSTEM*

This system is also a vectored system. The functional operation of the internal interrupt system is basically the same as the external one, and is illustrated in Figure I.2.4.

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

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).

Example:

If MON 2 instruction is executed on a level less than 14, T_{14} will be equal to 2.

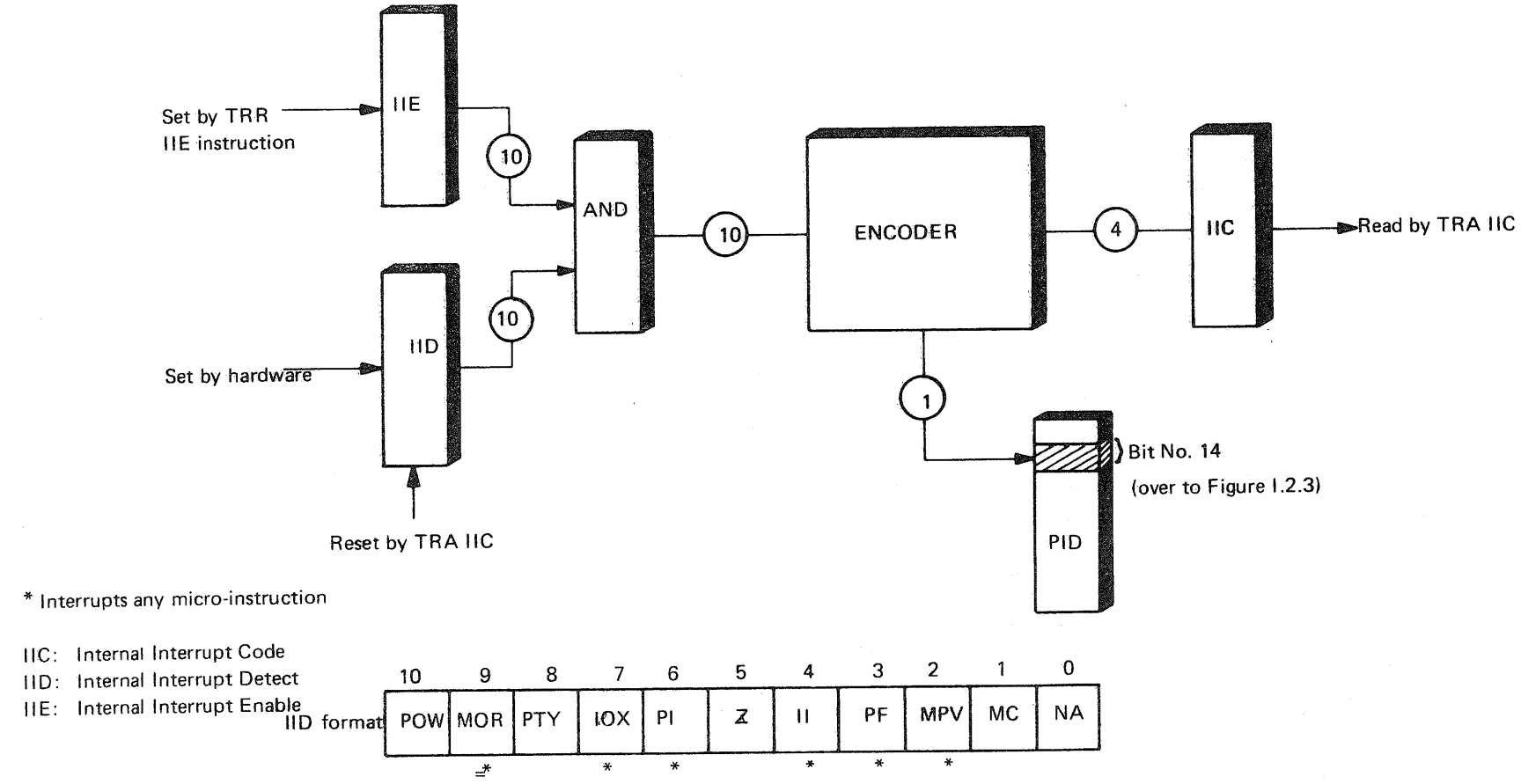


Figure I.2.4: *Internal Interrupt System, Block Diagram*

I.2.5.1 *IID -- Internal Interrupt Detect Register Bit Assignment*

The internal conditions which may cause internal interrupts and their associated vectors are listed below:

IIC Code:	Abbreviation:	Explanation:
0_8	–	This code is not legal and indicates a malfunction in the interrupt system.
1_8	MC	A monitor call instruction has been executed. The monitor call number is found in the T register on level 14. Note that this number is 8 bit with sign extension (i.e., the range -200_8 to 177_8).
2_8	MPV	Memory Protect Violation. Two types of violations are possible: 1. Memory Protect Violation This means that an illegal reference type (Read, Write, Fetch or Indirect) has been attempted. 2. Ring Violation This means that a program has attempted to access an area with higher ring status. Details regarding this interrupt is found in the Paging Status register (PGS, PSR).
3_8	PF	The program attempted to reference a page that presently is not in core, indicated by all the protect bits in PIT being zeros. Information regarding page number, etc. is found in the Paging Status Register.
4_8	II	Illegal Instruction Interrupt Attempted execution of an instruction that is not implemented causes this interrupt.

IIC Code:	Abbreviation:	Explanation:
5_8	Z	Error Indicator Interrupt The Z indicator in the STS register has been set. This may be caused by several conditions: 1. FDV with 0.0 2. EXR of an EXR instruction 3. DNX overflow 4. RDIV overflow 5. Programmed setting of Z (BSET, MST or TRR) Note: Level 14 must always reset the Z indicator on the offending level, otherwise a new interrupt will occur as the level is reentered.
6_8	PI	Privileged Instruction Interrupt A program running on ring 0 or 1 is trying to execute a privileged instruction as listed below: OF, ION, POF, PON, WAIT, IOT, IOX, IDENT, TRA, TRR, MCL, MST, LRB, SRB, IRW, IRR.
7_8	IOX	IOX Error Interrupt The addressed I/O device fails to return a CONNECT (response) within approximately 6 μs from start execution of an IDENT or IOX instruction. This may be due to a malfunctioning or missing device interface.
10_8	PTY	Memory Parity Error Interrupt A memory parity error has occurred. The lower 16 bits address is found by reading the PEA – Parity Error Address register (TRA PEA). The two upper two address bits and further information regarding the error is found by reading the PES – Parity Error Status register.

IIC Code:	Abbreviation:	Explanation:
11_8	MOR	Memory Out of Range Interrupt This interrupt occurs when a program tries to access non-existing memory. The PEA and PES will hold information regarding the error.
12_8	POW	Power Fail Interrupt This interrupt is given by the Power Sense Unit. It is possible that this interrupt occurs simultaneously with some other internal interrupts. In this case, the Power Fail Interrupt has priority.

Comments:

1. The interrupt system must be turned on in order to receive an external or internal interrupt.

2. The PIE bit number 14 has to be set in order to enable the internal interrupts.

3. The appropriate bit mask must be set up in IIE – Internal Interrupt Enable.

4. The paging system has to be turned on in order to receive a MPV, PF or PI interrupt.

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

6. If MOR, PF or MPV occur during an RNI or Fetch cycle, the CP is *not* incremented. In all other cases, CP points to the next machine instruction.

7. There is no priority assigned to the different internal interrupts as only one condition may arise at any time (except if Power Fail occurs in which case POW has the highest priority).

1.2.6 *INITIALIZATION OF THE INTERRUPT SYSTEM*

Before use of the interrupt system it must be initialized. After power-up, PID, 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 of PIE. Priority Interrupt Enable.

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

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

4. If the Z indicator is enabled for (IIE bit number 5), care should be taken that this indicator is not set in the status register (bit number 3) on the levels in question.

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 IIC the IIC register is unblocked and ready for use.

6. The interrupt system is turned ON.

Example:

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

1.2.7 *CONTROL OF THE INTERRUPT SYSTEM*

Two system control instructions are used for controlling the interrupt system. Both instructions are classified as privileged instructions.

ION Interrupt System ON

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 level exists, the ION instruction will be the last instruction executed at the old program level, and the P register at the old program level will point to the instruction after ION. The interrupt indicator on the operator's panel is lighted by the ION. The ION instruction is privileged.

IOF Interrupt System OFF

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

1.2.8 *EXTERNAL INTERRUPT IDENTIFICATION*

Four levels are assigned for external devices. The standard way of using the external interrupt levels are:

Level 13: Real Time Clock
Level 12: Input Devices
Level 11: Mass Storage Devices
Level 10: Output Devices

When an interrupt occurs on one of the above listed levels, the CPU is forced to do a fast level change.

To find the interrupt vector, i.e., to identify the device an IDENT instruction is used. The instruction has the following format.

IDENT <Program level>

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

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

Example:

```
LEV12,  IDENT   PL12      % Identify interrupting device on level 12
                          % The code is written into the A register
        RADD    SADP      % P: = P + A
        JMP     ERROR     % Branch
        JMP     DRIV1     % to
        JMP     DRIV2     % proper
        :
        :
        JMP     DRIVn     % driver

ERROR,  ...               % Ident code equal to zero
        ...               % is not legal

DRIV1,  ...
        ...

etc.
```

I.2.8.1 *Device Priority*

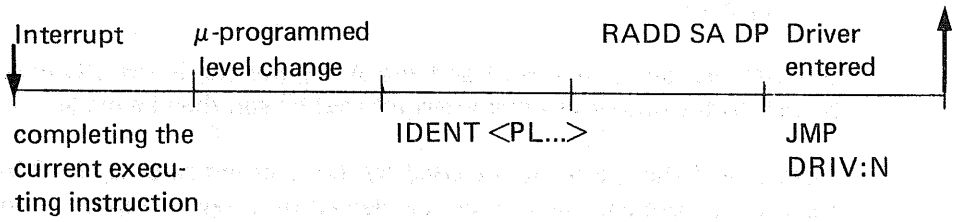
If devices on different levels issue simultaneous interrupts, the interrupt system will sort out the priority – devices attached to the higher level have higher priority.

However, if more than one device on the same level generates interrupts, which one should be treated first? The physical location of the device interface will, in this case, determine the priority. The location to the left in the I/O card crate gives the higher priority.

I.2.9 *INTERRUPT RESPONSE TIME*

Note that only 3 instructions are required from the time of interrupt until the proper driver is entered.

The interrupt response time is the time between the interrupt occurs in the device until the first instruction of the interrupt service program is entered. The interrupt response time is illustrated below:



I.2.10 *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 instruction will reset the bit in PID register corresponding to the level where the WAIT instruction is executed.

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

I.2.11 *INTERRUPT PROGRAM ORGANIZATION*

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

```
ENTRX'  –                   % First entry point
        –
        –
        WAIT                % Give up priority
        JMP ENTRX           % Next time program level is activated,
                            % the entry point is here
```

1.2.12 *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 and lock the register preventing overwriting.

After executing a

```
TRA IIC
```

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

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 the program level 14. (Otherwise, another interrupt will occur immediately.)

Example:

The IIC code may be analyzed by the following routine.

```
LEV14,  TRA     IIC       % Also resets error lock
        RADD    SA DP     % computed go to
        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
```

CP will point here when LEV14 is re-entered.

I.2.13 *PROGRAMMED INTERRUPTS*

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

Example 1:

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

```
        SAA 0
        BSET ONE 110 DA    % Set bit 9 to one
        MST PID            % Set PID bit 9
NEXT,   ...
```

Example 2:

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

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

Note that the SP on the "old" level will point to the NEXT instruction.

I.2.14 *USE OF THE PVL REGISTER*

In some cases after being forced to level 14_{10} the CPU (operation system) wants to know which level was the last one.

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

The PVL register holds the previous level information. When reading this register a rather unusual format is used.

Figure I.2.5 will help illustrate.

The A register will hold the instruction

IRR <previous level 10_8> DP

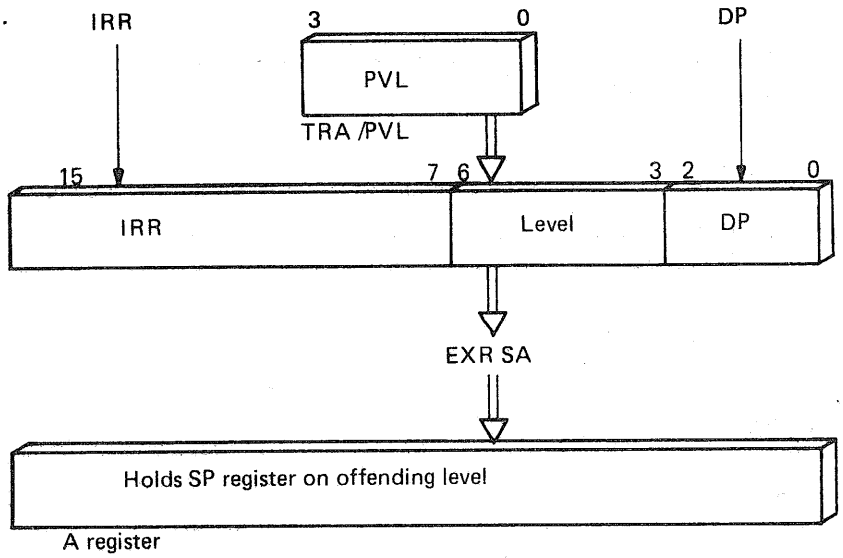


Figure I.2.5: *TRA PVL, Flow Diagram*

After executing the content of the A register as an instruction, the A register will be loaded with the content of the SP (Saved Program) counter of the offending level.

Example:

```
TRA     PVL     % A=: IR level no. DP
EXR     SA      % A:= P register on offending level
```

A decision should now be taken if P has been incremented or not. If not, the following two instructions should be executed:

```
COPY    SA DX   % A → X
LDA     ,X      % A offending instruction
```

If P has been incremented, the following two instructions should be executed:

```
COPY    SA DX   % A → X
LDA     −1,X    % A offending instruction
```

Note: The program counter has not been incremented if the violation occurred during instruction fetch. In all other cases it has been incremented.

This information is found in PSR (Paging Status Register) bit 15.

1.2.15 *PROBLEMS*

1. Operator's panel 18 bits switch register is set to 76051_8.

   a) How do we get this transferred to PIE?
   b) What levels will now be enabled?

2. 300/MON 37
   JMP + 0

   The above program is executed. Internal interrupt is generated. On program level 14_{10}, the following instruction is executed:

   1200/TRA IIC

   What is the content of the A and T registers on level 14_{10}?

3. Program Counter on level 2 = 1301_8.

   A program is executed on level 0. Program level 2 is activated by an interrupt. The first instruction on this level causes a MPV – Memory Protect Violation.

   a) What is the Previous Level Register – PVL?
   b) What is P register on level 2?
   c) How can the program on level 14_{10} find the instruction that caused the violation?

I.3 THE INPUT/OUTPUT SYSTEM

I.3.1 *GENERAL*

The input/output system (also referred to as the I/O system) facilitates a two-way communication between the CPU and the external devices. The external devices may be classified as:

- Input Devices
- Output Devices
- Mass Storage Devices

The mass storage devices may rather be considered as an extension of main memory than an I/O device.

What do we expect from a good I/O system?

- Reliability

  Information after an I/O transport is identical to the information before the I/O transport took place.

- High Band Width

  A high band width I/O system enables for high transfer rate of data.

- Parallelism

- Flexibility

  The I/O system should easily be expanded as gradually as the customer requires.

- Modularity

  I/O configuration changes should be carried out without difficulties.

- Simplicity

  A simple I/O system gives higher reliability and lower cost.

- Low System Overhead

  Device identification and service routines should not significantly reduce system speed.

- Low Cost Interfacing

  Low cost devices require low cost interfacing.

I.3.2 *DIRECT MEMORY ACCESS – DMA*

A Direct Memory Access channel – DMA – is normally used for mass storage devices, where a high data transfer rate and low access time are desired.

Examples of mass storage devices are:

mag. tape, disk, drum and floppy disk.

The CPU and the mass storage devices are then sharing memory and its bus system as a common resource. If simultaneous requests occur, the DMA channel is given the higher priority. The DMA is connected to main memory via the CPU bus (address + data + control) on a "cycle steal" basis.

The maximum speed of a DMA channel is close to 1 Mwords/second. Simultaneous operation on more than one DMA device is permitted, and no extra system overhead is introduced. The limitation is the channel band width.

Prior to a DMA transfer an initialization process is carried out. The initialization informations are given by the CPU via the A register (IOX instructions). The types of information given are: unit selection, transfer direction (read or write), number of words to be transferred, start address in memory and start address on mass storage device, etc.

The DMA operation takes place without CPU supervision, i.e., the CPU can be busy with other activity during the transfer time. The DMA activity is normally completed by an "end of operation interrupt" (word counter equals zero).

I.3.2.1 *Multiport Memory Access*

Assume that main memory has the same band width as the I/O system. Further assume that DMA activities use close to the full band width of the I/O system. Since the CPU also attempts to access the main memory, the CPU activity will be significantly reduced.

To overcome this problem, a direct multiport bus can be established.

If the CPU and DMA devices are using different banks, a complete independence and parallelism is obtained.

Refer to Figure I.3.1 and Chapter II.7 for multiport memory description.

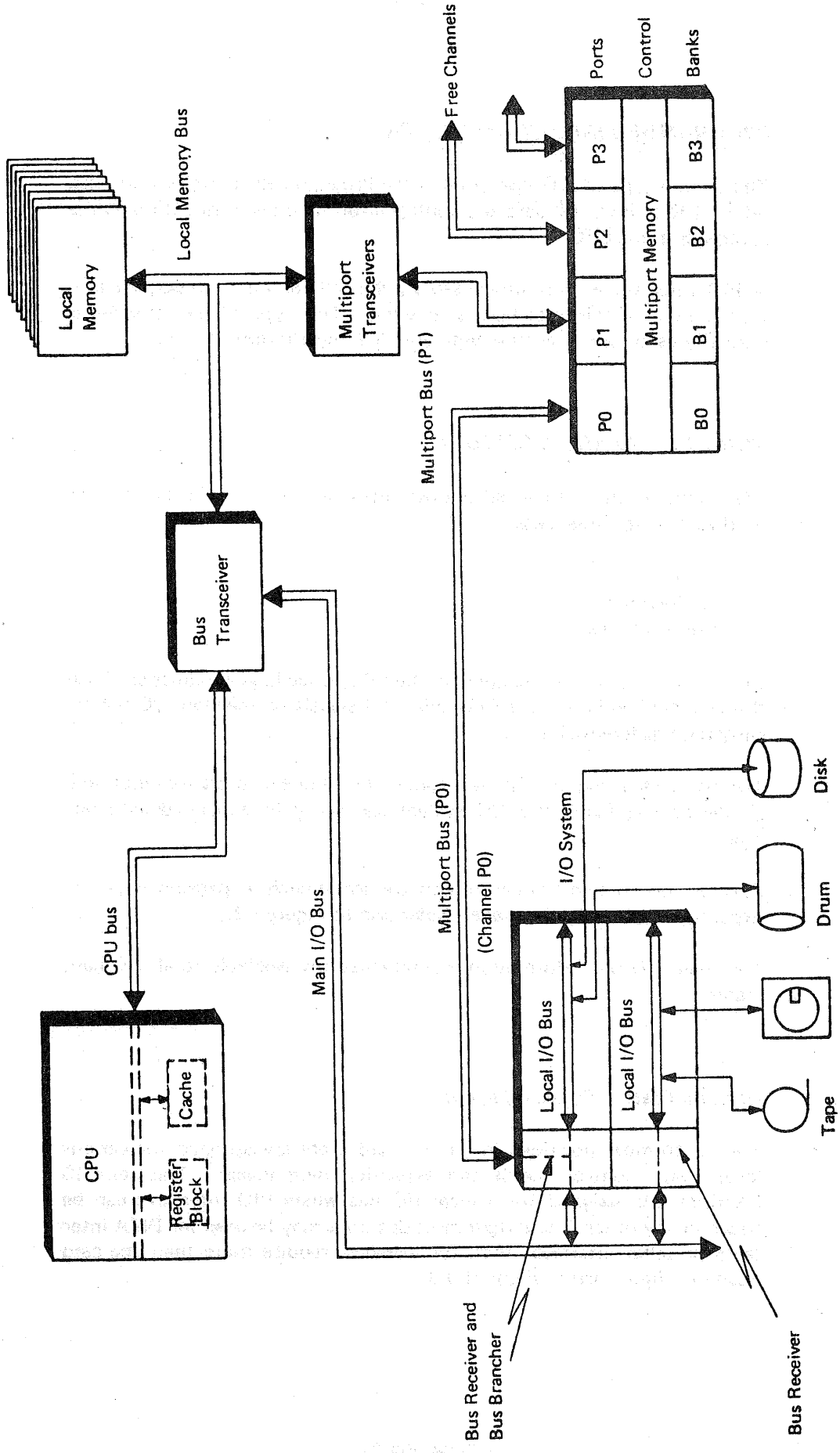


Figure I.3.1: *Storage Interconnection*

I.3.3 *PROGRAMMED INPUT/OUTPUT – PIO*

The second type of I/O operation is the Programmed Input/Output, also referred to as PIO. All data and control information is routed through the A register in the CPU.

A PIO interface is thus supervised by the CPU and as a consequence of that, a more simple interface is required. This type of data transfer is normally used for rather slow byte oriented input/output devices.

I.3.4 *NORD-10/S I/O ARCHITECTURE*

I/O communication is carried out over the main I/O bus. The bus may be subdivided into three parts:

- Data Bus
- Address Bus
- Control Lines

In the Bus Transceiver located in the CPU, three logic modules establish the communication on the CPU side. All signals on the main I/O bus are carried on differential lines.

The I/O system requires its own card crate. If one card crate cannot hold all device interfaces, the I/O system can easily be expanded with one more.

Furthermore, if two I/O card crates are not enough, expansion to an I/O expansion cabinet can be made. Refer also to Figure I.3.2.

The main I/O bus information is simultaneously available to all I/O card crates.

I.3.4.1 *I/O Card Crate – Physical Layout*

The 8 left-most positions in an I/O card crate are assigned for various local crate "control" cards and expansion plug spaces. The next 16 positions are assigned for a local I/O bus where PIO interfaces can be placed in any order. The 8 right-most positions may be used for DMA interfaces or other interfaces that, for instance, require more than one card position. Refer also to Figure I.3.3.

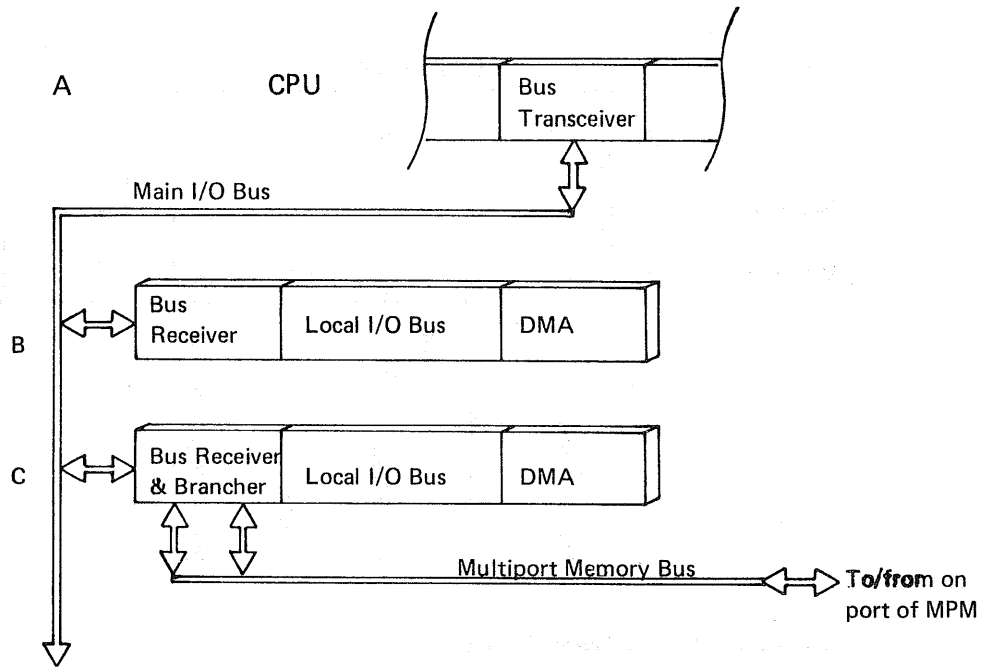


Figure I.3.2: *I/O Architecture, Block Diagram*

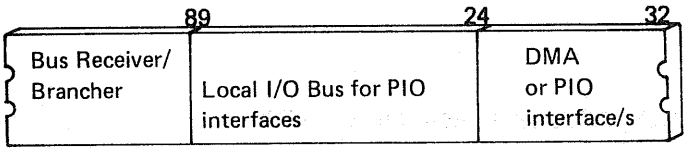


Figure I.3.3: *I/O Card Crate, Physical Layout*

- If the I/O card crate contains only PIO interfaces, the simplest state is used.
- If the DMA controller is located in the eight right-most positions a DMA Address Register must be inserted. The DMA data transport will then go over the CPU bus on a cycle steal basis.
- If a DMA data transport to memory over its own bus is desired, another card is added to handle the data to/from multiport.

Refer also to Figure I.3.4.

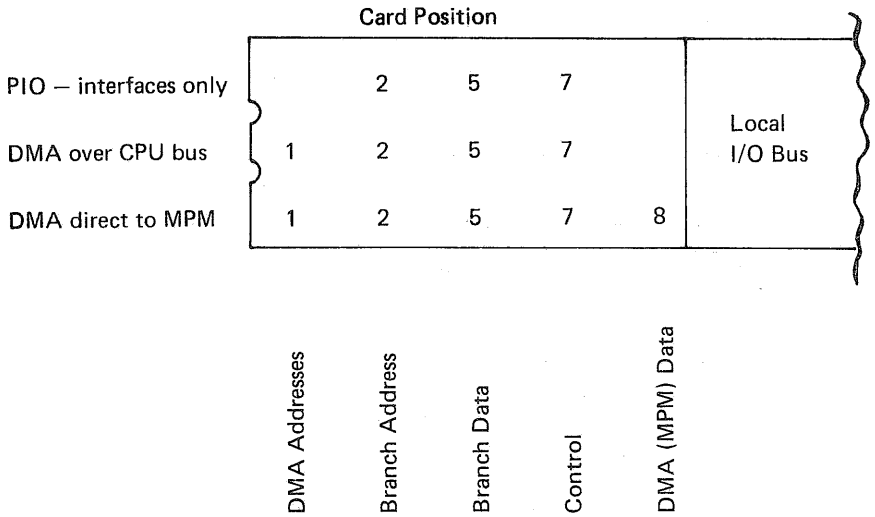


Figure I.3.4: *I/O Crate Control*

Note: Positions 3, 4 and 6 are assigned for plugs to the next card crate.

All signals on the local I/O bus are in tri-state form.

I.3.5 *THE INPUT/OUTPUT INSTRUCTION – IOX*

I.3.5.1 *General*

The communication with the I/O system for PIO and initialization of a DMA channel are controlled by the input/output instruction IOX. The IOX instruction has an 11 bits device register address, such that the format of the IOX instruction is:

IOX <register address>

The device register address may take the values $0\text{-}3777_8$. The IOX instruction itself does not carry any information about what kind of transfer should be performed. This kind of information is found in the addressed device register. When addressing a register, the register itself "knows" if an input or an output should take place.

If the addressed register is an input register, this will signal the control logic in the interface to set up the necessary control functions for an input operation. The A register will, when the IOX instruction is completed, contain a copy of the information in the addressed input register.

If an output transfer was desired, the data to be transferred must be loaded into the A register prior to execution of the IOX instruction. The addressed register will signal the control logic in the interface which in turn will strobe the data at some certain point in time during the IOX instruction. The data is enabled out from the A register and the addressed register will hold a copy of the A register when IOX is executed.

I.3.5.2 *Operation*

The IOX instruction is carried out by 3 μ-instructions. Refer also to Figure I.3.6 for the following discussion.

The first μ-instruction drives the content of the A register out to an I/O data register. The data contained in this register will be driven out on the I/O data bus which through the local I/O bus(es) is available to all device interfaces.

At the same time, the address part of the IOX instruction is driven out on the main I/O address bus. This address is available to all I/O interfaces through the local I/O bus(es). A strobe signal (IOXE) is issued and forces all device interfaces to "look" at the address bus. The one that recognizes the address (device number) will answer with a CONNECT. Figure I.3.5 shows the decoding of the device address.

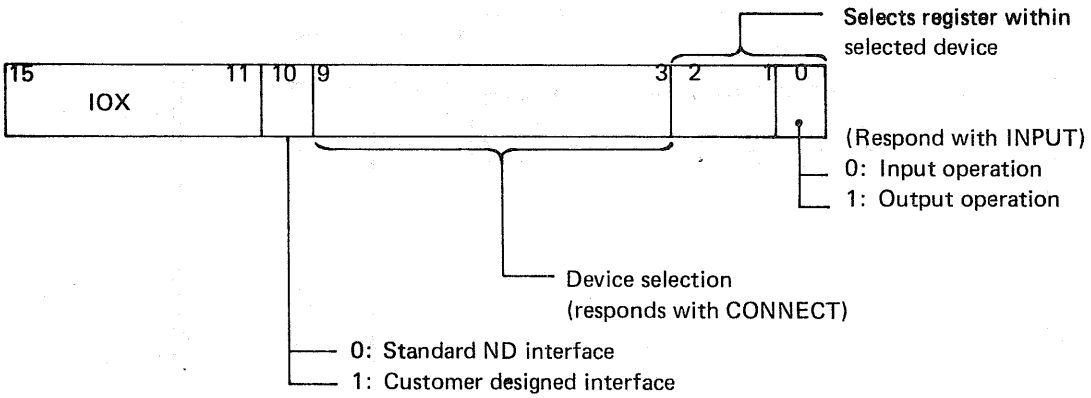


Figure I.3.5: *IOX <Device Address> Decoding*

If this IOX instruction performs an output operation (address bus bit number 0 = 1) the lower three bits of the address will select the proper register within the interface, and this register will accept the data on the local I/O data bus.

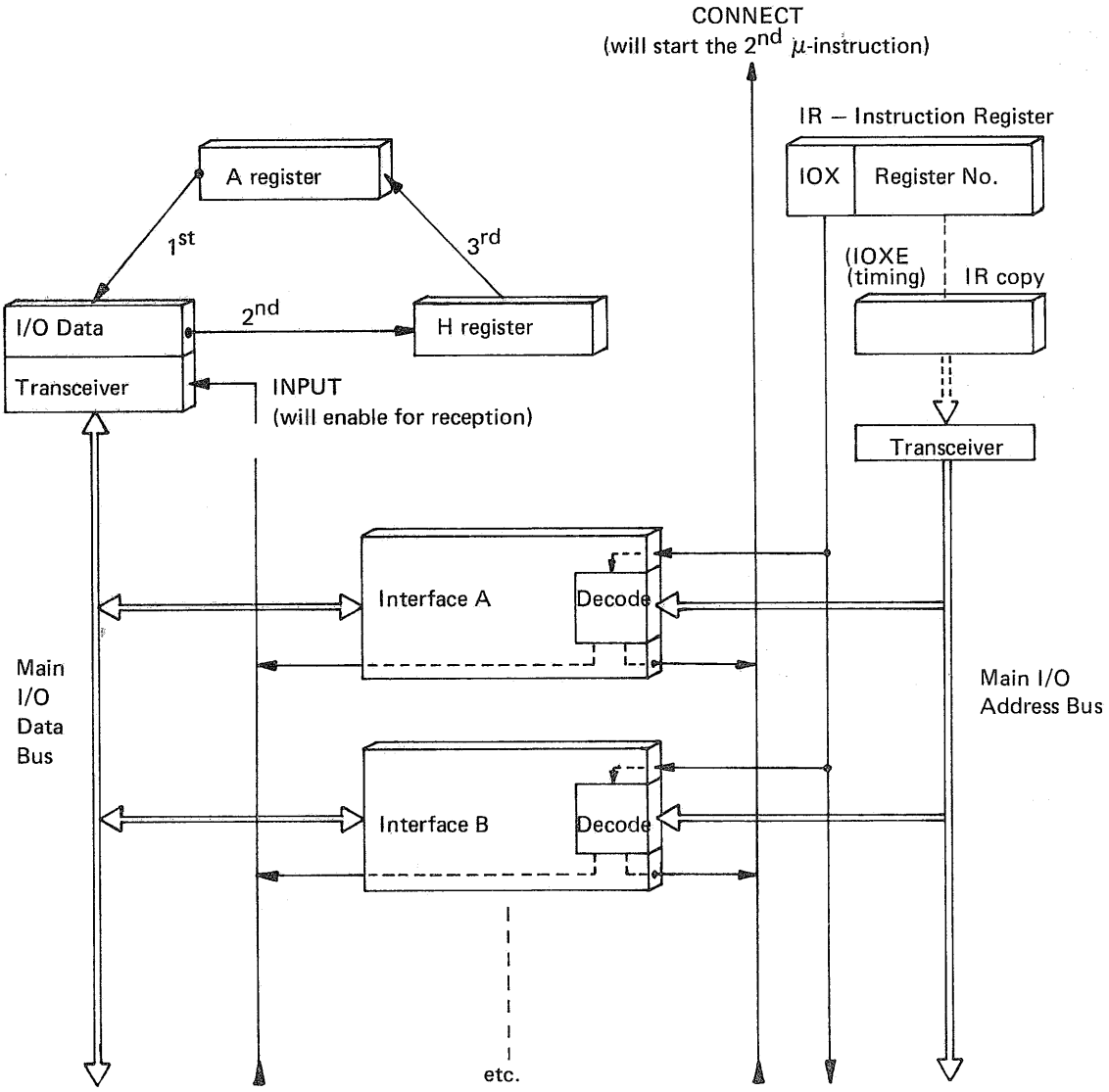


Figure I.3.6: *IOX Instruction Execution*

On the other hand, if this IOX instruction performs an input operation (address bus bit number 0 = 0), an INPUT signal is generated on the selected interface together with the CONNECT signal. The INPUT signal will force the transceiver associated with the I/O data register to receiver data from the main I/O data bus.

When the CPU sees the CONNECT signal the second μ-instruction will be started and will take the information in the I/O data register and move to the H register. The third μ-instruction will move the content of the H register back to the A register.

Note: The content of the A register is unchanged after an output operation.

I.3.6 *DEVICE INTERRUPT*

The interrupt system is described in Chapter I.2. A device interrupt can be divided into two classes:

- End of operation interrupt

  and

- Error interrupt.

Both indicate a change in the conditions in the interface and the CPU should be updated with this information.

Both types of interrupt from a device will drive the same interrupt line.

I.3.6.1 *Device Interrupt Identification*

However, since a vectored interrupt is used, i.e., more than one device can drive the same interrupt line, the vector or ident code is found by using an IDENT <program level> instruction. The IDENT instruction will be used on levels 10, 11, 12 and 13.

If level 15 is used, only one device is allowed and no vector is required, i.e., no IDENT instruction is required for this level.

When an IDENT 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 indicated level having an interrupt condition, will respond with a 9 bits identification code. This code will be received by the A register. The IDENT instruction is similar to an IOX input instruction. The differences are:

1. The lower 6 bits of the main I/O address bus will hold a level code indicating the level on which the device search is performed.

2. To indicate that an IDENT instruction is being executed, a timing signal IDENT is used.

3. The IDENT instruction will always read the IDENT code for the first device found (on the indicated level) giving an interrupt.

When the interrupting device responds with the ident code, the interrupt conditon will be reset.

I.3.6.2 *Analyzing the Interrupt*

Having identified the device, the CPU should find the reason for the interrupt. This information is found in the status register located in the interrupting device. Knowing the reason for the interrupt, the CPU will branch to the proper routine.

I.3.6.3 *Interrupt Sequence*

1. An interrupt condition occurs in a device. The local interrupt FF sets driving the interrupt line (10, 11, 12 or 13).

2. Provided the CPU is operating on a lower level, the CPU is forced to the interrupting level.

3. An IDENT instruction is issued. The IDENT CODE is received in the A register.

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.

I.3.7 *CONTROL AND STATUS*

In addition to a data register, an interface must also have:

- a Status Register

  and

- a Control Register.

Orders to a device are given through the control register while the feed-back to the CPU goes through the status register.

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

I.3.7.1 *Format of Status and Control Word*

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

Control Word

Bit 0	Enable interrupt on device ready for transfer
Bit 1	Enable interrupt on errors
Bit 2	Activate device
Bit 3	Test mode
Bit 4	Device clear

Bit 5 Address bit 16
Bit 6 Address bit 17
Bit 7 Not assigned
Bit 8 Not assigned
Bit 9 Unit
Bit 10 Unit
Bit 11 Device operation
Bit 12 Device operation
Bit 13 Device operation
Bit 14 Device operation
Bit 15 Device operation

The meaning of "device operation" may vary from one equipment to another. For instance, Read Transfer, Write Transfer, Stop Code, etc.

For more specific information, refer to Chapter III.3, "Programming Specifications for I/O Devices".

I.3.8 *PROGRAMMING INPUT/OUTPUT*

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.

I.3.8.1 *Input/Output Programming Without Use of Interrupt*

Figure I.3.7 illustrates the usage of the data, control and status words in a PIO interface.

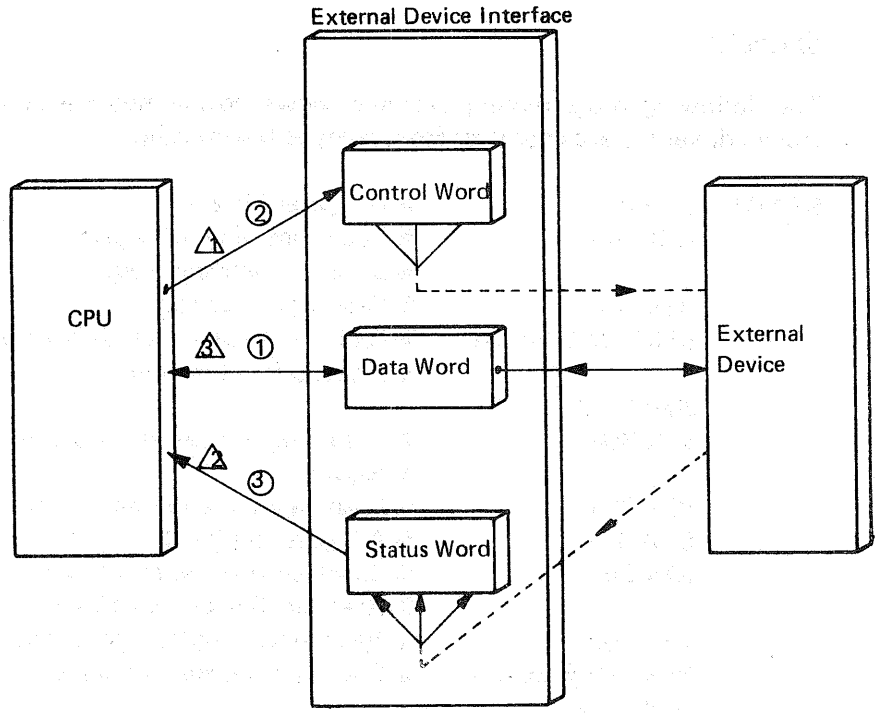


Figure I.3.7: *Loading Sequence for Interface Registers*

We notice that a control word can only be loaded from the CPU and that the status word can only be read. A data word has a two-way communication with the CPU.

We also notice that if the external device is capable of doing input and output, another set of registers is required (in the figure, the same set is used).

Input: ⚠1 The control word must be loaded. The control word will enable the input transfer.

⚠2 The status will change when the external device transfers data to the data word. By reading and analyzing the status word, the CPU will "know" this.

⚠3 The input of the data word will take place.

Output ① Output of the data word is performed.

② The control word is loaded to tell the external device to take the data word from the interface.

③ When the data transfer has taken place, the status will change, and by reading and analyzing the status word, the CPU "knows" that a new transfer can be initiated.

Example:

The following programming example shows how a non-interrupt controlled driver reads a character from teletype 0 with echo.

```
START,  SAA 4           % A register bit 2 = 1
        IOX 303         % Load control word register
                        % Bit 2 = 1 (activate read)
        IOX 302         % Read status register
        BSKP ONE 30 DA  % Is bit 3 = 1 (Device ready for transfer).
                        % If yes, skip 1 location.
        JMP * --2
        IOX 300         % Read data register, character in A
                        % register
        IOX 305         % Load data write register, echo
        SAA 4           % A register bit 2 = 1
        IOX 307         % Load control word bit 2 = activate
                        % (transfer character to TTY)
        IOX 306         % Read status register for output device
        BSKP ONE 30 DA  % Character transfer completed?
        JMP * --2
        JMP START       % Repeat
```

I.3.8.2 *Programming Input/Output Using Interrupt*

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

Example:

Interrupt controlled driver for reading and printing a character on teletype.

Initialize Interrupt System on levels: 0, 1, 10 and 12.

```
LEV,0   SAA 2
        MST PID         % interrupt to level 1
        JMP *  0

LEV,1   SAA 7
        IOX CONTW       % Set control  word for input (303)
        WAIT            % Give up priority
        LDA BUFF        % Get data
        IOX WDATA       % Cond. data write register (305)
        SAA 7
        IOX CONTW       % Set control used word for output (307)
        WAIT
        JMP LEV1        % Give up priority
```

```
LEV10,  IDENT PL 10          % Identify interrupt

        CHECK IDENT CODE

        IOX STATUS           % Get output status word (306)
        BSKP ZERO 40 DA
        JMP ERROR            % Status not ok – error routine
        SAA 2
        MST PID              % interrupt to level 1
        WAIT                 % give up priority
        JMP LEV10

LEV12,  IDENT PL12           % identify interrupt

        CHECK IDENT CODE

        IOX STATUS           % read input status word (302)
        BSKP ZERO 40 DA
        JMP ERROR            % status not ok – error routine
        IOX RDATA            % get data (300)
        STA BUFF             % save data
        SAA 2
        MST PID              % interrupt level 1
        WAIT                 % give up priority
        JMP LEV12

BUFF,   0
ERROR,  JMP * 0
```

I.3.9 *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
STATUS EQUAL TO READY
     AND ON CYLINDER
LDA MEMADDR
IOX CORE ADDR              % Load core address register
LDA WORDCOUNT
IOX WORDCOUNT REG          % Load word counter
LDA DISK ADDR
IOX DISK ADDR              % Load disk address register
LDA CONTROLWORD
IOX CONTROL WORD           % Load control word register
:                          % Enable for interrupt upon transfer
:                          % completion, select unit, specify read
:                          % or write and activate device
:
INTERRUPT ON COMPLETION OR ERROR:

IDENTIFY INTERRUPT
READ STATUS
TEST STATUS FOR NORMAL COMPLETION
READ CORE ADDR REG
CORE ADDR REG AS EXPECTED?
:                          % Core address register before transfer
:                          % and word count should be equal to
:                          % core address register after transfer
:                          % completion
:
```

I.4 OPERATOR'S COMMUNICATION

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

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

The NORD-10/S may either be controlled from the NORD-10/S operator's panel or from a Teletype or visual display unit. The micro-program is designed in such a way that either the operator's panel or the Teletype (visual display unit) may control the NORD-10/S.

The NORD-10/S 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:

Character:	Use:
0, 1, 2, 3, 4, 5, 6, 7	Octal digits used to specify addresses and data
@	Restart MOPC; clear PIE
$	Octal load
&	Binary load
!	Start program in main memory
/	Specifies register or memory cell examine
CR (carriage return)	Terminator of line
LF (line feed)	Echoed, no other effect
/___/ (space)	Octal number before the space is ignored
B	Used to specify 64K bank number (page table number when paging is on)
I	Internal register examine

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

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

I.4.1 *FUNCTIONS*

I.4.1.1 *Start a Program*

Format:

<octal number> !

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

I.4.1.2 *Memory Examine*

Format:

<octal number> /

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

When the paging system is on, the Bank number (see Section I.4.1.8) specifies which page table is used, and page faults and protected violations are ignored. In this case, <octal number> specifies a virtual address.

Example:

```
717/003456              % EXAMINE ADDRESS 717

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

I.4.1.3 *Memory Deposit*

Format:

<octal number> (CR)

After a memory examine, the contents of the memory cell may be changed by typing an octal number terminated by CR.

Example:

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

I.4.1.4 *Register Examine*

Format:

<octal number> R <octal number> /

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

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

0 Status register, bits 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 is printed out.

Examples:

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

I.4.1.5 *Register Deposit*

Format:

<octal number> (CR)

After a register examine, the contents of the register may be changed by typing an octal number terminated by CR.

Examples:

```
R5/ 123456 54321 (CR)   % CONTENT OF A REGISTER ON
                        % LEVEL 0 IS CHANGED TO 054321

7R2/ 000044 55 (CR)     % CONTENT OF P REGISTER ON
                        % LEVEL 7 IS CHANGED TO 000055
```

I.4.1.6 *Internal Register Examine*

Format:

| <octal number> /

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

0	PANS	Operator's Panel Status, used by operator's panel microprogram only.
1	STS	Status register, program level is contained in bits 8-11, bit 14 = PONI and bit 15 = IONI.
2	OPR	Operator's panel switch register
3	PGS	Paging status register
4	PVL	Previous program level (GETR PVL DP)
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	Active level, decoded
12	ALD	Automatic load descriptor
13	PES	Memory error status
14	MPC	Microprogram counter (will show a constant)
15	PEA	Memory error address
16	IO	I/O transfer. Do not use.
17	–	Will show an arbitrary register. Do not use.

I.4.1.7 *Internal Register Deposit*

Format:

I <octal number> (CR)

After an internal register examine the contents of the internal register with the same internal register code. It may be changed by typing an octal number terminated by CR. For deposit, the following internal register codes apply:

0	PANC	Operator's panel control, used by operator's panel micro-program only.
1	STS	Status register, only bits 0-7 will be changed.
2	LMP	Operator's panel lamp register (will be overwritten unless U register is selected)
3	PCR	Paging control register
4	MISC	"Miscellaneous" register (used by micro-program to control IONI, PONI, MCALL and MOPC)
5	IIE	Internal interrupt enable
6	PID	Priority interrupt detect

7	PIE	Priority interrupt enable
10	CCLR	Cache Clear
11	–	Not used
12	CILR	Cache inhibit limits register
13	CAR	Instruction register, used by microprogram subroutine only
14	IR	Instruction register, used by the EXR instruction only.
15	ECCR	Error correction control register
16	IO	I/O transfer. Do not use.
17	–	Will change an arbitrary register. Do not use.

Examples:

```
17/ 030013 0 (CR)        % EXAMINE PIE AND CHANGE TO
                         % 000000

112/ 021540 20044        % EXAMINE ALD AND CHANGE
                         % CILR TO 020044
```

I.4.1.8 *Current Location Counter*

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

I.4.1.9 *Break Function*

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

1.4.1.10 *Bank Number*

Format:

<octal number> B

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

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

1.4.2 *BOOTSTRAP LOADERS*

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

- Octal format load
- Binary format load
- Mass storage load

1.4.2.1 *Octal Format Load*

Octal load is (normally) started by typing:

<physical device address> $

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

During octal load there is no echoing of characters. All legal operators' commands are accepted. Illegal commands terminate the loading and "?" is typed on the console. (In installations without console an attention lamp is turned on.) Normally, @ or ! is used to terminate an octal load.

If no device address precedes the $ command, then $ is nearly equivalent to pushing the LOAD button on the operator's panel. (See also Section 1.4.2.4.)

I.4.2.2 *Binary Format Load*

Binary load is (normally) started by typing:

<physical device address> &

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

The binary information must obey the following format:

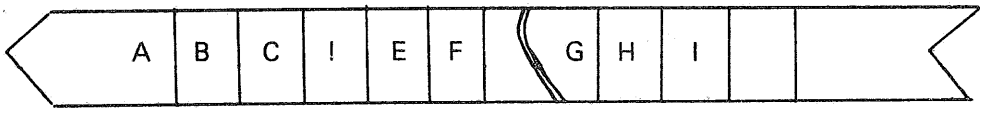


Figure I.4.1: *Binary Load Format*

A Any types not including ! (ASCII 41_8)

B (Optional) octal number (any number of digits) terminated with a non-octal character *:

C (Optional) octal number terminated with the character ! (see below)

! Signals start of binary information (ASCII 41_8)

E Block start address. Presented as two bytes (16 bits), most significant byte first.

F Word count. Presented as two bytes (16 bits), most significant byte first. (E, F and H is not included in F.)

G Binary information. Each word (16 bits) presented as two bytes, most significant byte first.

H Checksum. Presented as two types (16 bits), most significant byte first. The checksum is the 16-bit arithmetic sum of all words in G.

I Action code. If I is a blank (zero), then the program is started in the address previously found in the octal number B (see above). If B is not specified, B = 0 is assumed. If I is not a blank, then control is returned to the operator's communication, which decodes I. (The number B will be found in the P register on level 0.)

---

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

If no device address precedes the & command, then the & is nearly equivalent to pusing the LOAD button on the operator's panel (see Section I.4.2.4).

If a checksum error is detected, "?" is typed (in installations without console an attention lamp is turned on) 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.

I.4.2.3 *Mass Storage Load*

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

If an error occurs, the loading is terminated and "?" is typed on the console and control is returned to the operator's communication. (Note: in installations without console, an attention lamp is turned on.)

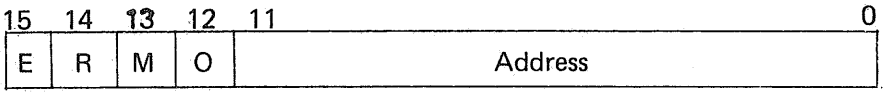
The actual mass storage must conform with either dump or disk programming specification.

Mass storage load must be started by typing $ or &, or pushing the LOAD button on the operator's panel. However, this requires a special setting of the ALD. Refer to Section I.4.2.4 for details.

I.4.2.4 *Automatic Load Descriptor*

The NORD-10/S has a 16-bit switch register called Automatic Load Descriptor (ALD) (located on the Panel Driver Card). This register specifies the load procedure to use when the LOAD button is pushed or when a single $ or & is typed.

The ALD format is as follows:



Automatic Load Descriptor (ALD) Format

E Extensions

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

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

R Restart*

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

Address = 4 * (ALD bits 0-13)

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

M Mass Storage Load

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

O Octal Format Load

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

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

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

---

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

I.4.2.5 *Examples*

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

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

Table I.4.1: *ALD Setting*

I.4.3 *NORD-10/S MICROPROGRAMMED MEMORY TEST*

The NORD-10/S microprogram (1K ROM) contains a special program that tests the main memory. This is a very useful feature as it quickly decides whether an error is in the CPU or the memory.

The program first stores and checks pattern 1, 2, 4 - - - 100000 and then the complement (seen by pushing DATA on operator panel). Afterwards, the address is stored as data in the corresponding addresses and the result is checked. The operation is repeated 16 times.

Initiate:

Press MASTER CLEAR
On TTY 1 (Console TTY) type:

R3/XXXXX (lower address to be tested) RETURN B register
R7/YYYYY (upper address to be tested) RETURN X register

Start:

Type: 101657$ (it is advisable to push RETURN to get an audible stop indication)

If memory is ok, the LOAD light is turned off when the test is finished.

If memory is NOT OK, the:

MASTER CLEAR lights up and ? is typed on the TTY.

The following registers give additional error information:

R2/XXXXX	P register shows failing address
R1/YYYYY	D register shows pattern read from memory
R4/NNNNN	L register shows pattern stored (test pattern)
R6/MMMMM	T register shows failing bit (L⊻D → T)

Note: This test is not implemented in machines with 32 bits floating point format.

I.5 OPERATOR'S PANEL

I.5.1 *PANEL ELEMENTS*

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

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

The operator's panel physical layout may be depicted in Figure I.5.1.

I.5.2 *18-BIT SWITCH REGISTER*

This register is used to present 18 bit data to the CPU. Normally, only 16 of these are used. The switches may be read from program with the TRA OPR instruction. In installations with big memory (more than 64K) 18 switches and lamps may be needed to represent the possible 18 bit addresses for the "examine memory" function. When the paging system is on, switches 16 and 17 select page table number.

I.5.3 *18-BIT LIGHT EMITTING DIODE REGISTER*

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

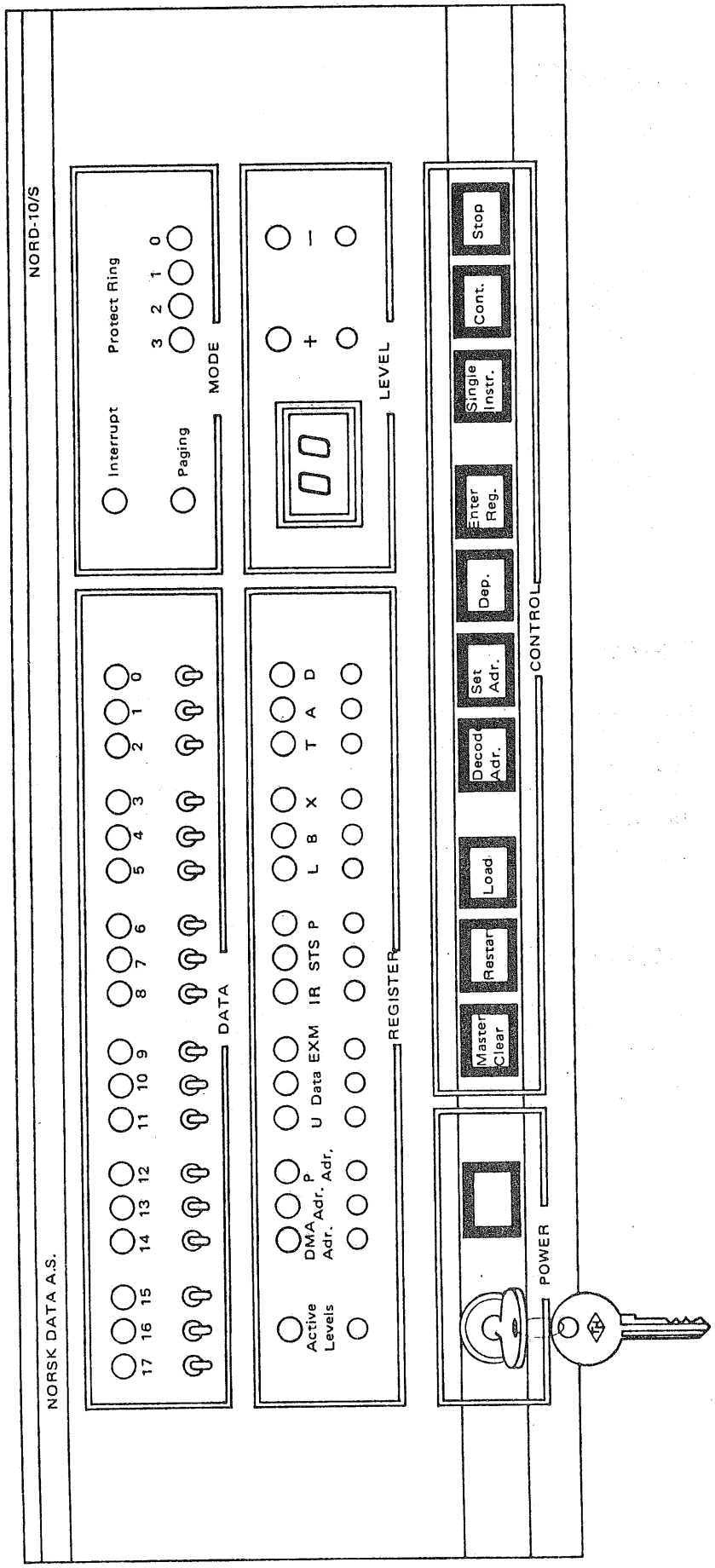


Figure I.5.1: *Operator's Panel — Physical Layout*

I.5.4 *16 SELECTOR PUSH-BUTTONS AND 16 ASSOCIATED LIGHT EMITTING DIODES*

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

The possible register selections are:

ACTIVE LEVELS

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

DMA ADR

If this button is pushed, the data display will show the active DMA (Directory Memory Access) address. See also Section I.5.6.4.

ADR

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

P ADR

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

U

This is the user register set by the TRR LMP instruction.

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

DATA

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

EXM

This selection has two uses:

CPU in STOP

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

CPU runs

The data display will show the contents of the memory location whose address is set in the switch register. The memory location is sampled after each panel interrupt (about every 2-3 ms). The panel interrupt is handled directly by microprogram.

IR

This selection will display the CPU instruction register.

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

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

I.5.5 *DISPLAY LEVEL SELECT*

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

I.5.6 *CONTROL BUTTONS*

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

I.5.6.1 *Master Clear*

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

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

I.5.6.2 *Restart*

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

I.5.6.3 *Load*

The LOAD button starts automatic program load from a device. The device may be an input/output device or a mass storage device, depending on the setting of a switch register (ALD) on the Panel Control Card.

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

I.5.6.4 *Decode Address*

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

I.5.6.5 *Set Address*

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

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

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

I.5.6.6 *Deposit*

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

I.5.6.7 *Enter Register*

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

I.5.6.8 *Single Instruction*

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

I.5.6.9 *Continue*

When this button is pressed, the machine starts running from the address specified by the P register. The level is given by the contents of the PIE and PID registers. If the MASTER CLEAR is first pressed, PIE is cleared and the program is started on level 0.

If the light on the CONTINUE button is on, it indicates that the CPU is running.

I.5.6.10 *Stop*

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

I.5.7 *MODE INDICATORS*

INTERRUPT

Indicates that the interrupt system is turned on, i.e., an ION instruction has been executed.

PAGING

Indicates that the paging system is turned on, i.e., a PON instruction has been executed.

RING

Four indicators show active program protect rings. These indicators are provided with after-glow so that it is possible to observe even the shortest execution run on each ring.

I.5.8 *DATA AND CONTROL*

A special part of the μ-program residing in Read Only Memory (ROM) establishes the communication between the panel and the CPU. The panel interface consists of3 logic modules located in the CPU card crate. The data between the panel and its interface is partly in parallel, partly in serial transmission form.

A panel interrupt indicates for the processor that the panel needs service from the μ-program. The next machine instruction will be temporarily halted, and the panel μ-program is initiated. Refer to Figure I.5.2.

A Panel Status register (PANS) located in the panel interface will indicate for the μ-program what kind of service is required. It is read by the μ-program by a TRA PANS operation.

Via the Panel Control register (PANC), located in the panel interface, the μ-program will give feed-back to the panel example bit indicators, etc. This is accomplished by a TRR PANC operation.

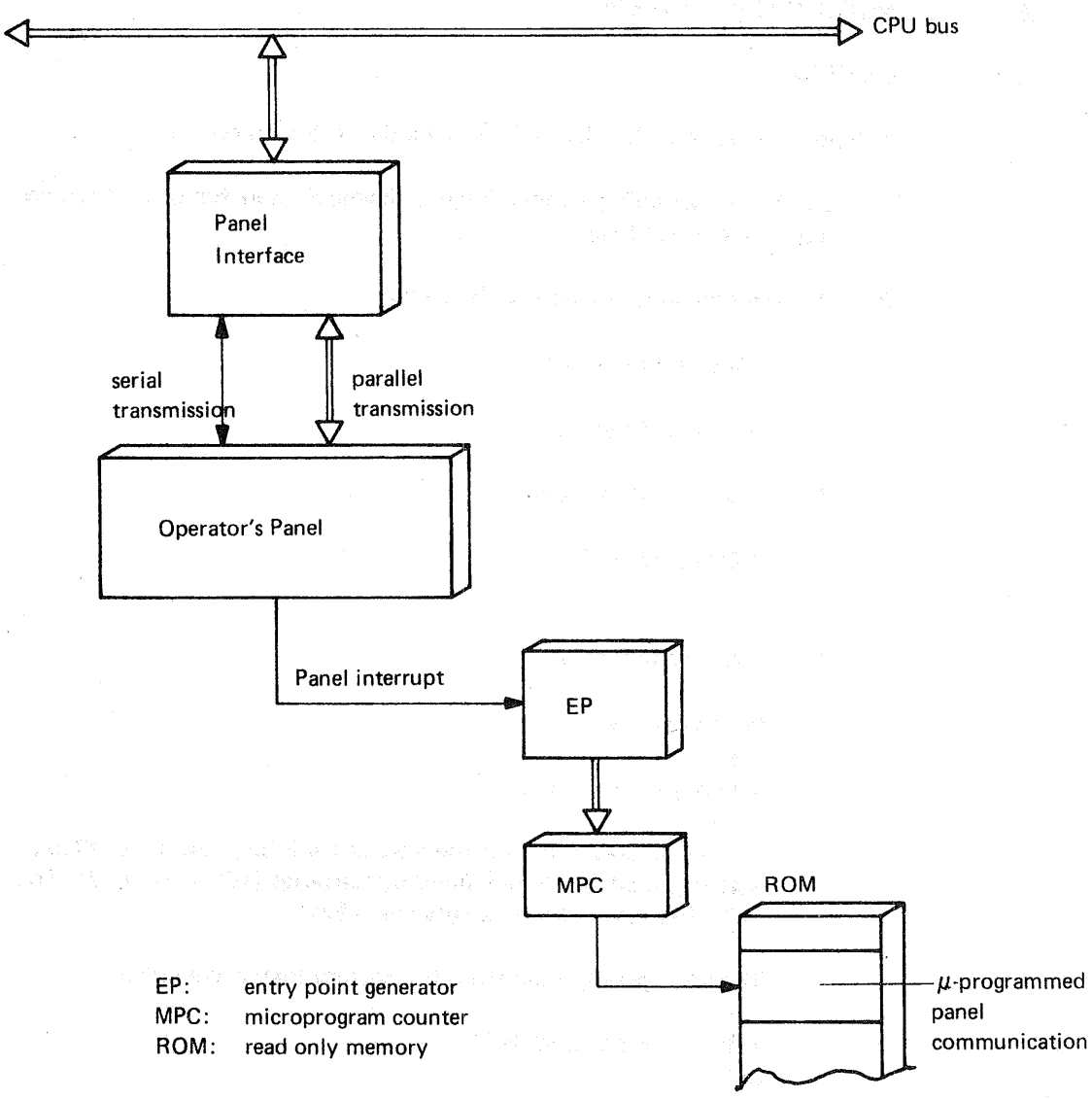


Figure I.5.2: *Panel Data and Control*

I.6 NORD-10/S POWER UNIT

I.6.1 *GENERAL*

The power system in NORD-10/S is divided into two parts:

1. 5V 100A switching supply (high efficiency) used for CPU, I/O and memory control logic.

2. A serial regulated power supply giving:

   * +5V stand by 8A

   * +12V stand by 2A

   * –12V stand by 70mA

   □ +24V stand by 2A

   * voltage applied to:

     MOS memory
     and
     memory refresh logic

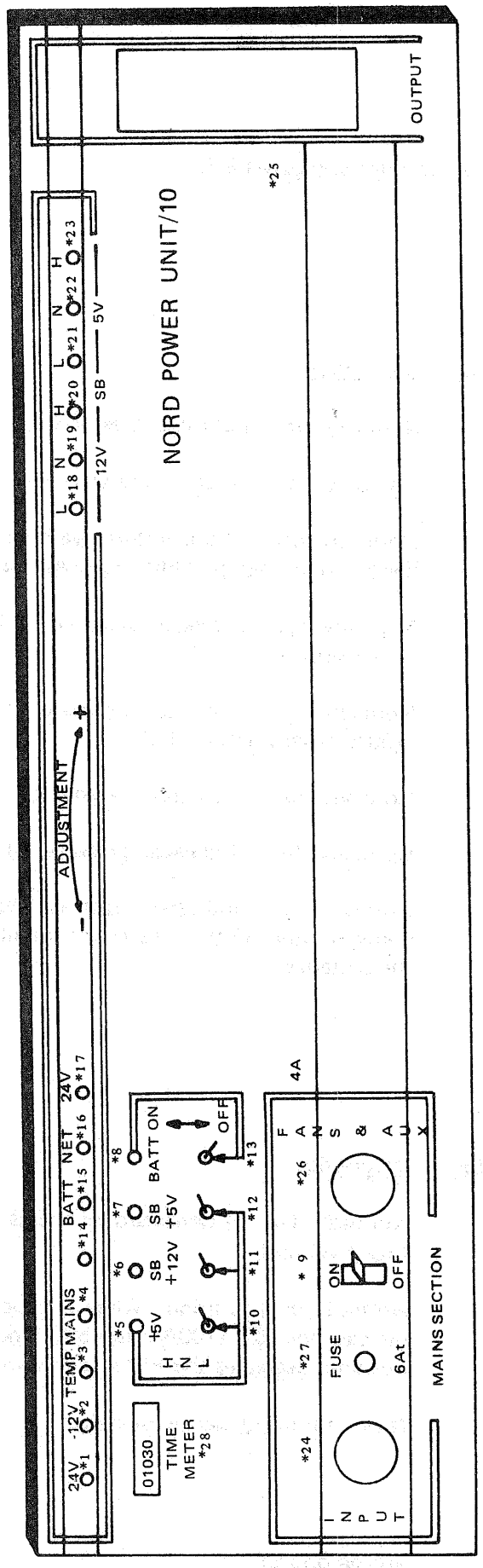
   * in case of power failure the voltages will be present for 30 minutes supplied from two stand by batteries (+5V and +12V). The +12V battery will also supply the –12V.

   * The voltages supplied from the battery may drive up to:

     128K with 8K modules
     or
     256K with 16K modules

   □ voltage may be used in I/O system for current loop interfaces.

   □ up to ± 10% variation on mains input voltage is accepted



ND-06.009.01

Figure I.6.1: *NORD-10/S Power Unit*

I.6.2 *DESCRIPTION*

For the following description refer to Figure I.6.1.

I.6.2.1 *Indicators*

Label	Location	Description
+24V	*1	Normally lit. Indicates presence of +24V.
–12V	*2	Normally lit. Indicates presence of –12V.
Temp	*3	Normally off. Temperature warning light lights when temperature reaches 55° C.
Mains	*4	Normally lit. Indicates presence of Mains (input voltage).
+5V	*5	Normally lit. Indicates presence of +5V (100A) from switching P.S.
SB+12V	*6	Normally lit. Indicates presence of +12V.
SB+5V	*7	Normally lit. Indicates presence of +5V.
Batt	*8	Normally off. Indicates, when lit, that the voltages (+5V, +12V, –12V) are supplied by the batteries.

I.6.2.2 *Switches*

Label	Location	Description
ON-OFF	*9	Normally ON. Connects/disconnects mains (input voltage).
H–N–L+5V	*10	Normally in N position. When in H position, increase the +5V (100A) with 5%. When in L position, decreases the +5V stand by with 5%. Note: for maintenance use only.

Label	Location	Description
H–N–L+12V SB	*11	Normally in N position. When in H position increases the +12V stand by with 5%. When in L position, decreases the + 12 V stand by with 5%. Note: for maintenance use only.
H–N–L+5V SB	*12	Normally in N position. When in H position, increases the +5V stand by with 5%. When in L position, decreases the +5V stand by with 5%. Note: for maintenance use only.
Batt ON-OFF	*13	Normally in ON position. Connects/disconnects the charge circuits for voltage back-up batteries.

I.6.2.3 *Adjustments*

Note: Before performing the adjustments listed below, block plastic caps must be removed.

Label	Location	Description
N.A.	*14	Spare.
Battery	*15	Adjustment of batteries charge current. Factory adjusted.
Net	*16	Power failure threshold adjustment. Factory adjusted.
24V	*17	24V adjustment.
12V SB L	*18	Adjustment of –5%. Refer to *11.
12V SB N	*19	Adjustment of 12V stand by. Refer to *11.
12V SB H	*20	Adjustment of +5%. Refer to *11.
5V SB L	*21	Adjustment of –5%. Refer to *12.
5V SB N	*22	Adjustment of 5V stand by. Refer to *12.
5V SB H	*23	Adjustment of +5%. Refer to *12.

I.6.2.4 *Input Voltage*

Label	Location	Description
INPUT	*24	Mains input, fused and filtered power. (Main fuses and filter located in front/bottom of main cabinet).

I.6.2.5 *Output Voltage*

Label	Location	Description
OUTPUT	*25	All above listed voltages to various logic destinations.
Fans Aux	*26	220V AC for main cabinet fans. Can also control a mains relay connecting cabinets. (Bus switch, I/O, etc.). Maximum load 4A.

I.6.2.6 *Fuse*

Label	Location	Description
Fuse	*27	Mains input fuse. Type: slow blow (T register). Size: 6A.

I.6.2.7 *Time Meter*

Label	Location	Description
Time Meter	*28	Indicates CPU. Power ON Hours – POH.

DETAILED CONTENTS

+ + +

SECTION II

Section:		*Page:*
II.1	**The Storage System**	II–1–1
II.1.1	General	II–1–1
II.1.2	A Simple Memory Model	II–1–2
II.2	**The Memory Hierarchy**	II–2–1
II.2.1	Still Cheaper and Faster	II–2–2
II.3	**NORD-10/S Storage System**	II–3–1
II.3.1	Memory Communication Principles	II–3–4
II.3.2	Control Signals	II–3–4
II.3.3	The Address Arithmetic	II–3–7
II.3.3.1	General	II–3–7
II.3.3.2	Addressing Structure	II–3–7
II.3.3.3	Address Arithmetic Operation	II–3–11
II.4	**MOS – Memory Operating Principles**	II–4–1
II.4.1	General	II–4–1
II.4.2	A Memory Cell	II–4–2
II.4.3	Write Operation	II–4–3
II.4.4	Read Operation	II–4–3
II.4.5	Refresh Operation	II–4–3
II.4.6	4K x 1 Bit Memory Chip	II–4–3
II.4.7	Chip Cycles	II–4–4
II.4.7.1	Read Cycle	II–4–4
II.4.7.2	Write Cycle	II–4–5
II.4.7.3	Refresh Cycle	II–4–6
II.4.8	16K x 1 Bit Memory Chip	II–4–7

Section:		*Page:*
II.5	**Local Memory**	II–5–1
II.5.1	General	II–5–1
II.5.2	Addressing	II–5–1
II.5.3	Data	II–5–2
II.5.4	Control	II–5–3
II.5.5	Refresh	II–5–6
II.6	**Error Checking and Correction**	II–6–1
II.6.1	General	II–6–1
II.6.2	Error Checking and Correction – Functional Description	II–6–1
II.6.2.1	Comments to Tables II.6.2 and II.6.3	II–6–6
II.6.3	Parity	II–6–6
II.6.4	Error Correction or Parity	II–6–8
II.6.5	Error Correction – Control	II–6–8
II.6.6	Memory Out of Range	II–6–10
II.6.7	CPU Feedback Information	II–6–10
II.6.8	Error Logging	II–6–12
II.7	**Multiport Memory System**	II–7–1
II.7.1	General	II–7–1
II.7.2	Basic Unit	II–7–4
II.7.2.1	Bank	II–7–4
II.7.2.2	Port	II–7–4
II.7.2.3	Channel	II–7–4
II.7.3	Card Crate	II–7–5
II.7.3.1	Expansion	II–7–5
II.7.3.2	Physical Memory Space Organization	II–7–5
II.7.4	Addressing	II–7–8
II.7.4.1	Bank Selection	II–7–8
II.7.4.2	Bank Address	II–7–8
II.7.4.3	Example	II–7–10
II.7.5	Interleave	II–7–11
II.7.5.1	Two-way Interleave	II–7–11
II.7.5.2	Example	II–7–12

Section: *Page:*

II.7.6	NORD-10/S to NORD-50 Memory Communication	II–7–13
II.7.7	Control and Information Flow	II–7–13
II.7.7.1	Write Operation	II–7–14
II.7.7.2	Read Operation	II–7–14
II.7.8	Refresh	II–7–14
II.7.9	Big Multiport Memory System	II–7–15
II.7.9.1	General	II–7–15
II.7.9.2	The Memory System	II–7–15
II.7.9.3	System Parts	II–7–16
II.7.9.3.1	Crate	II–7–16
II.7.9.3.2	Bank	II–7–19
II.7.9.3.3	Controller (1144)	II–7–19
II.7.9.3.4	Storage (1132's)	II–7–20
II.7.9.3.5	Port (1142, 1143)	II–7–21
II.7.9.3.6	Channel	II–7–21
II.7.9.3.7	Source	II–7–22
II.7.9.3.8	Service Channel	II–7–23
II.7.9.4	Addressing	II–7–27
II.7.9.4.1	Address Conversion	II–7–27
II.7.9.4.2	Bank Selection	II–7–29
II.7.9.4.3	Interleave (Shifted Address)	II–7–30
II.7.9.5	Data	II–7–32
II.7.9.5.1	Format	II–7–32
II.7.9.5.2	Data Protection	II–7–32
II.7.9.5.3	General about Error Detection and Correction	II–7–32
II.7.9.5.3.1	Write	II–7–35
II.7.9.5.3.2	Read	II–7–35
II.7.9.5.3.3	Single Data Error	II–7–35
II.7.9.5.3.4	Single Control Code Error	II–7–36
II.7.9.5.3.5	Multiple Error	II–7–36
II.7.9.5.4	Control and Error Reporting	II–7–38
II.8	**Memory Management System**	II–8–1
II.8.1	General	II–8–1
II.8.2	Realization	II–8–3
II.8.3	Address Translation	II–8–5
II.8.4	Page Table Selection	II–8–8
II.8.5	Memory Protection	II–8–10
II.8.5.1	Permit Protection System	II–8–10
II.8.5.2	Ring Protection System	II–8–14

Section:		*Page:*
II.8.6	Privileged Instructions	II–8–18
II.8.6.1	Status	II–8–18
II.8.6.2	Replacement Algorithms	II–8–19
II.8.6.3	Paging Status Register	II–8–19
II.8.7	Control of Memory Management System (MMS)	II–8–21
II.8.7.1	General	II–8–21
II.8.7.2	Control of Paging Control Registers – PCR's	II–8–21
II.8.7.3	Control of Page Tables – PT's	II–8–22
II.8.7.4	Timing	II–8–25
II.8.8	Problems	II–8–25
II.9	**Cache Memory**	II–9–1
II.9.1	General	II–9–1
II.9.2	Cache Memory Architecture	II–9–1
II.9.2.1	Location	II–9–1
II.9.2.2	Type	II–9–1
II.9.2.3	Size	II–9–2
II.9.2.4	Placement/Replacement Algorithms	II–9–3
II.9.2.5	Program Start	II–9–3
II.9.2.6	Implementation	II–9–3
II.9.3	Cache Memory Accesses	II–9–7
II.9.3.1	Cache Addressing	II–9–7
II.9.3.2	Read Access	II–9–7
II.9.3.3	Write Access	II–9–7
II.9.3.4	Cache Inhibit Area	II–9–8
II.9.4	Control of the Cache Memory	II–9–9
II.9.4.1	Setting of Cache InhibitsLimits	II–9–9
II.9.4.2	Cache Initialization	II–9–9
II.9.4.3	Cache Status Register	II–9–10
II.9.5	Cache Timing	II–9–10
II.10	**Power Fail/Automatic Restart**	II–10–1
II.10.1	General	II–10–1
II.10.2	Power Fail	II–10–1
II.10.3	Automatic Restart	II–10–3

II.1 THE STORAGE SYSTEM

II.1.1 *GENERAL*

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

It is used to hold program (instructions), indirect address, operands and results. Memory will regard all as information, i.e., it does not descriminate the different types of information. What do we expect from a good storage system?

1. Reliability

   Information read is the same as being stored on an earlier stage.

2. Low Storage Cost

   To be part of an efficient computer system, cost per bit stored should be as low as possible.

3. Low Access Time

   Computer performance is to a great extent given by an efficient memory system. Information should be presented as soon as the demand for it arises, while on the other hand, information should be stored as quickly as possible to enable for new memory referer-ances.

4. Small Physical Dimension

   Higher storage density enables a more practical physical layout giving direct benefits as lower storage cost.

5. Great Capacity

   Storage limit will to a great extent set the limit for computer performance.

6. Modularity

   Desired storage capacity changes should be carried out without difficulty. Malfunctioning memory sections should easily be replaced.

7. Low Power Requirement

   Low power requirement will indirectly give lower storage cost through smaller dimensioned power supplies. Lower power requirement will give lower heat dissipation which again gives higher reliability.

8. Simplicity

   Simplicity reduces the cost and increases the reliability.

II.1.2 *A SIMPLE MEMORY MODEL*

A simple memory system is illustrated in Figure II.1.1.

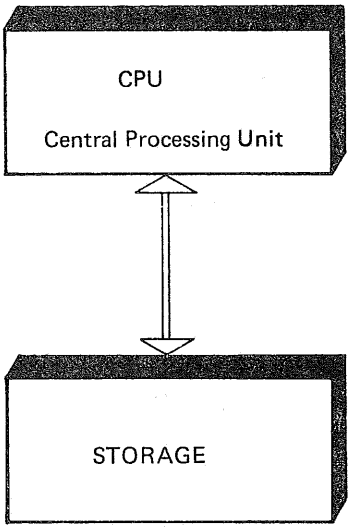


Figure II.1.1: *A Simple Memory Model*

Due to the fact that some of the demands for a desired storage system are opposing, the storage system simplicity must, to a certain extent, be overlooked in favour of demands for speed and low cost.

Even the two major requirements, speed and cost, are opposing. The question then will be: Which one of the two systems should be prefered?

1. Slow and cheap
   or
2. Fast and expensive

Our goal is to combine: cheap and fast!!

II.2 THE MEMORY HIERARCHY

In an effort of approaching this goad (fast and cheap), a memory system, as illustrated below, is employed.

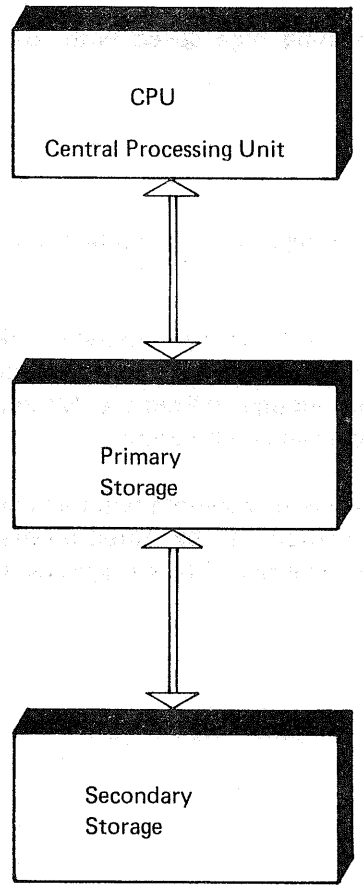


Figure II.2.1: *Two Level Storage System*

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, i.e., 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 is transported from secondary storage 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.

The right approach is taken – combining high speed with low cost.

II.2.1 *STILL CHEAPER AND FASTER*

Looking at the usage of the primary storage, only a fraction is dynamically used within a short time period.

If a new level of high speed storage could be inserted between the CPU and primary storage with the feature of being so small that it would not influence the total storage cost, but big enough to hold the dynamic part of the primary storage, this would be an ideal combination.

Some of the information to be processed is very seldom required and does not have to be "on-line". This information can be stored on magnetic tape, disk packs or floppy disks in a data library. The storage cost will be low.

II.3 NORD-10/S STORAGE SYSTEM

NORD-10/S employs a multi-level storage concept as briefly described earlier.

Refer to Figure II.3.1 for the following discussion.

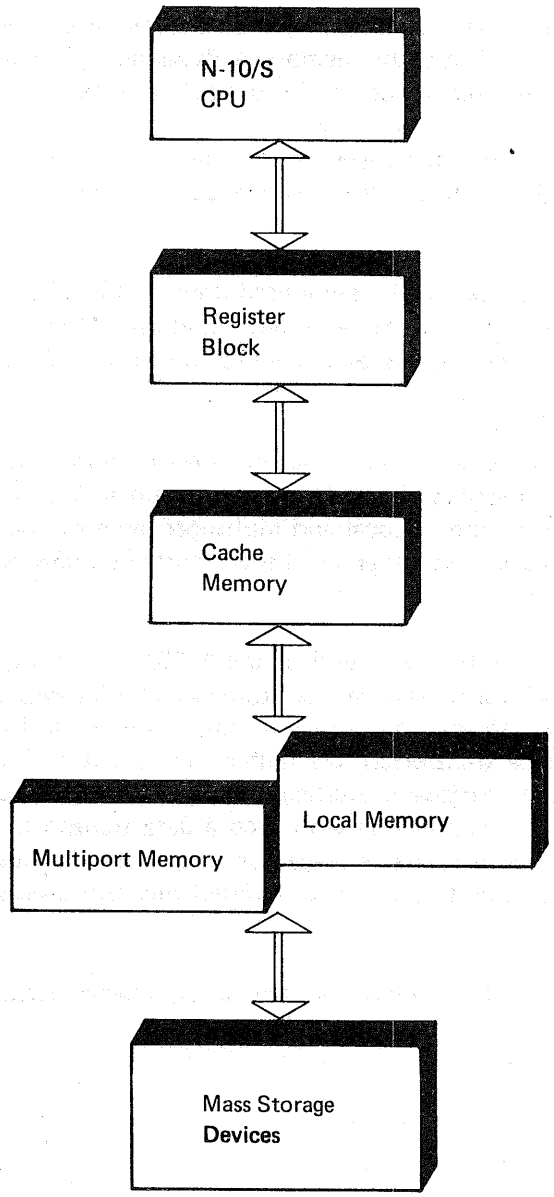


Figure II.3.1: *Multilevel Storage System*

– The first level in the memory system is the register block, holding 128 programmable registers. Refer to Section I.1.

  Since the register block is integrated with the CPU, a very short access time is achieved.

– The Cache memory is located close to the CPU working as a selective high speed memory. The cache memory is dynamically updated to hold the most actual information. Refer to Section II.9.

  Note: Cache memory introduces no overhead in the system by operating in parallel with the Memory Managment System.

– Local memory is located in the same card crate as the CPU. Using 32K memory modules, local memory may hold up to 256K words. Error correction may also be used in connection with local memory. Refer to Section II.6.

– On the same level as local memory, multiport memory may be installed. Multiport memory is physically located in up to 2 card crates in the rear of the cabinet. Local and Multiport memory may both be installed in the same system and will then share the 256K physical address space.

– Mass storage device is the next level of the N-10/S Storage System. Large amounts of information can be stored 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).

  Figure II.3.2 shows the memory system in its logical perspective.

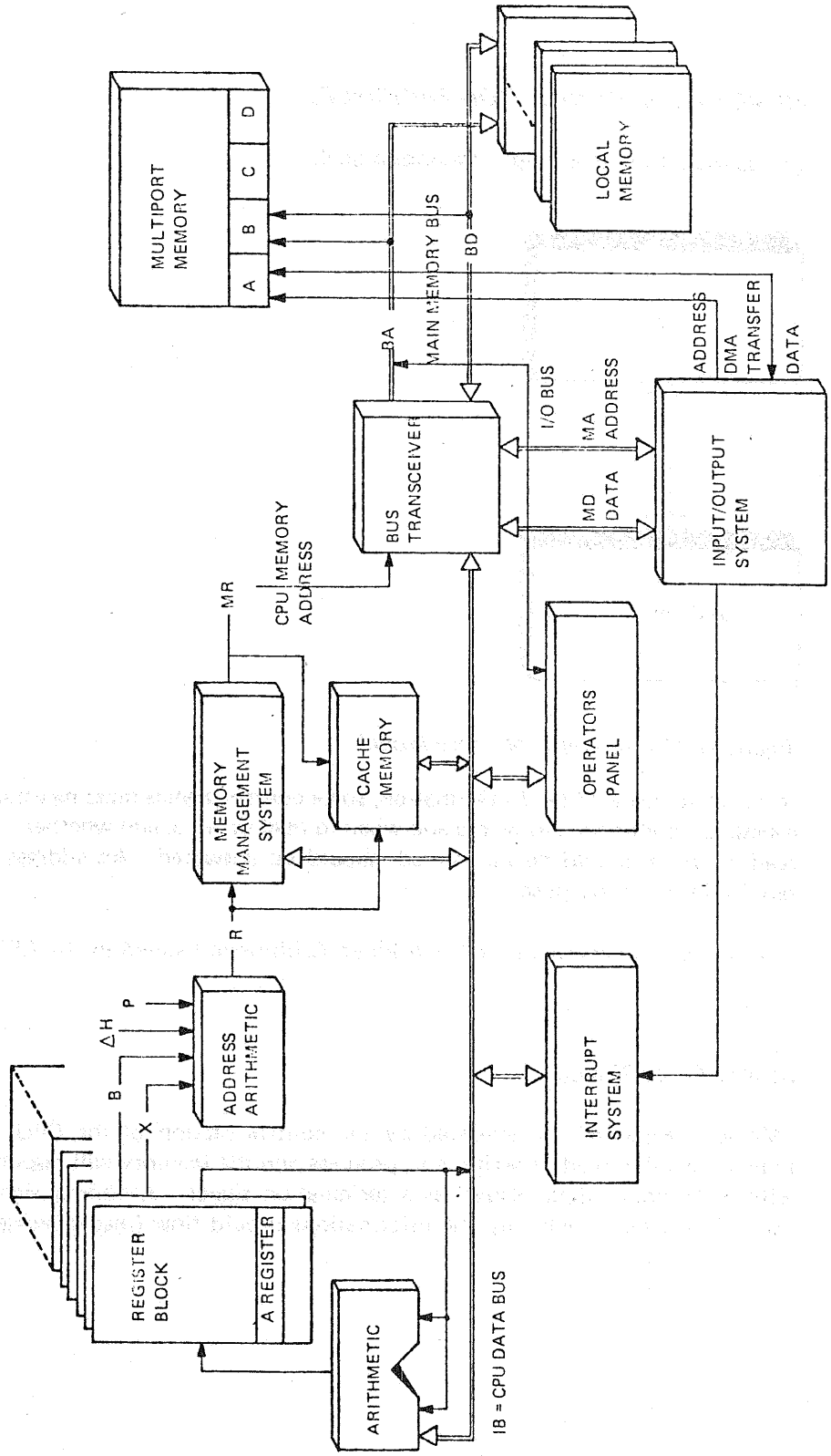


Figure II.3.2: *Storage Interconnection*

II.3.1 *MEMORY COMMUNICATION PRINCIPLES*

Let us again look at a simple memory model.

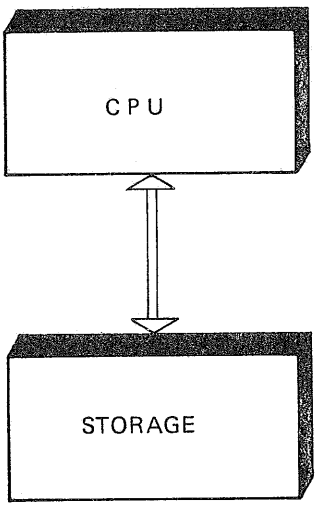


Figure II.3.3: *A Simple Memory Model*

In order to store or read information, some control signals must be established. Questions such as where and when to read or write and whether a read or write should be performed, should be answered. An address is required for that purpose.

The address is generated by the Address Arithmetic located in the CPU.

II.3.2 *CONTROL SIGNALS*

"Memory Request" is generated by the control section of the CPU to indicate when a read or write is in progress and the Memory will respond with a "Memory data ready" as a termination signal. A control signal "write" will tell which way the informations should flow (read or write).

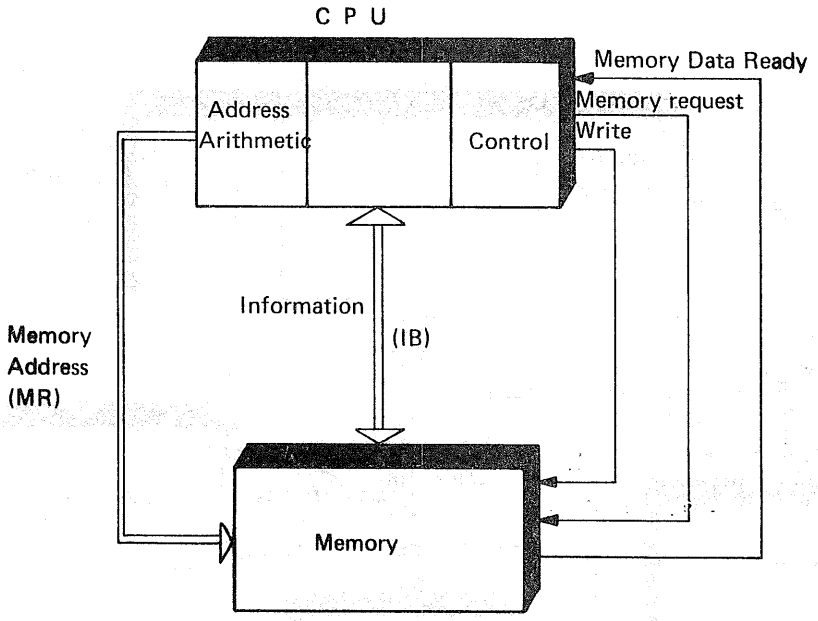


Figure II.3.4: *Memory Information and Control – Principal*

More precisely, "Memory data ready" means:

- for a write operation that memory has accepted the data
- for a read operation that information is available for the CPU on the Information Bus.

For optimum memory utilization, a Memory Management system is employed. Refer to Section II.8 for a detailed description. The 16 bits virtual addresses issued from the Address Arithmetic will here be dynamically converted to an 18 bits physical address. Refer to Figure II.3.5.

For increased performance a 1K word cache memory is inserted. This memory will operate in parallel with the Memory Management system and thus introduces no overhead or delay in the system. For a detailed description see Section II.8.

For increased local storage reliability, error correcting circuitry is employed. Refer to Section II.6 for details.

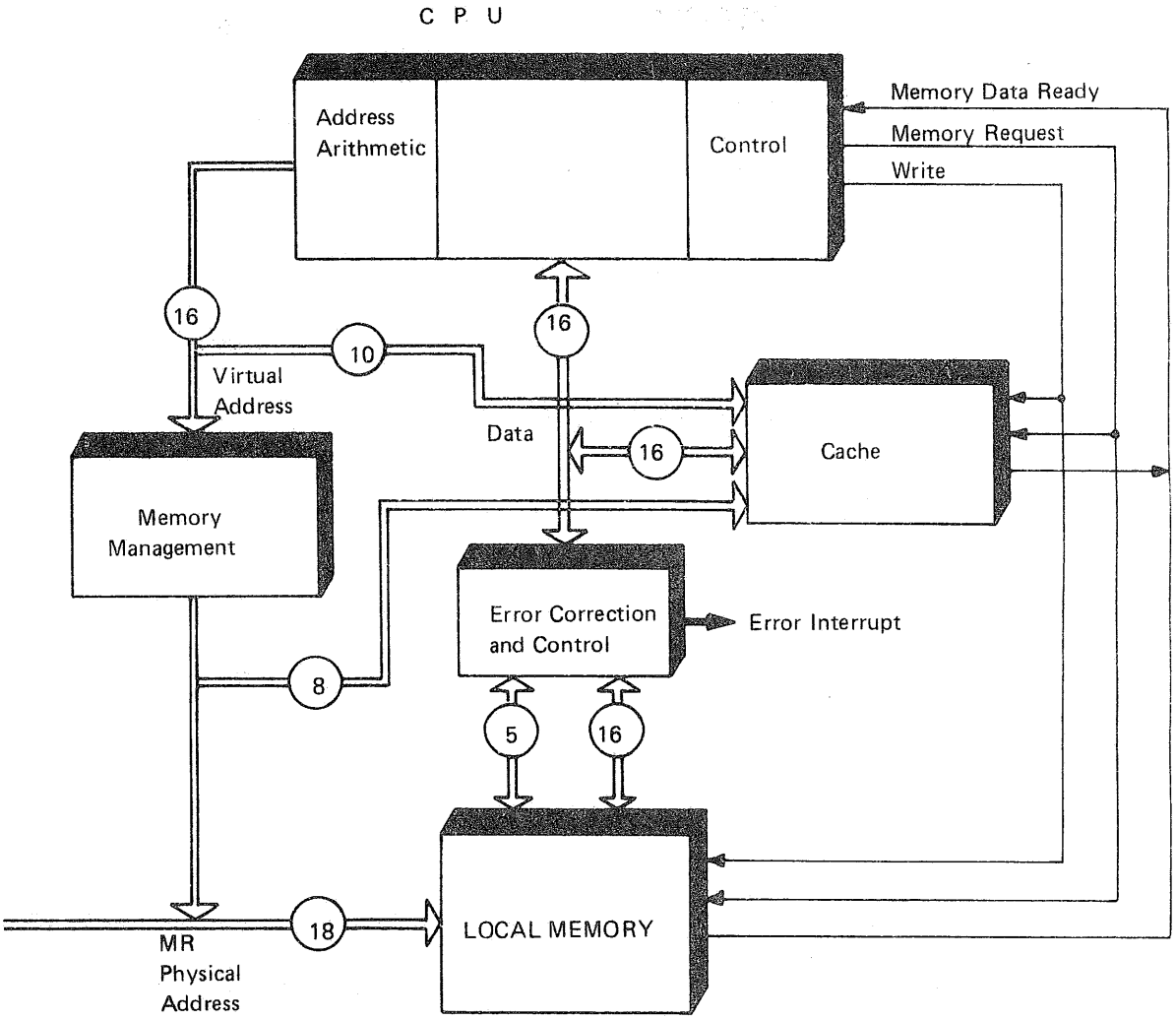


Figure II.3.5: *Memory Information and Control – Functional*

II.3.3 *THE ADDRESS ARITHMETIC*

II.3.3.1 *General*

In order to communicate with memory, an address must be formed. This is the task of the Address Arithmetic.

Three types of information are stored in memory.

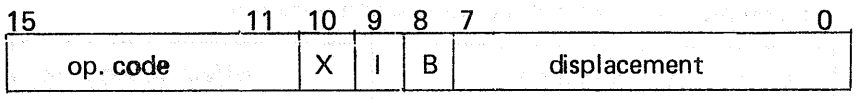
1. Data (operands or results)
2. Instructions (program elements)
3. Indirect addresses

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

1. Fetch – Fetch next instruction
2. ROP – Read operand
3. RADDR – Read indirect address
4. STO – Store operand

II.3.3.2 *Addressing Structure*

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



NORD-10/S 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 register.

The three addressing mode bits called, ",X", "I" and ",B" providing 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.

---

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

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

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

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

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

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

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

Table II.3.1: *Addressing Modes*

Addressing examples are given in NORD-10/S Referance Manual (ND-06.008).

The principal operation is shown in Figure II.3.6.

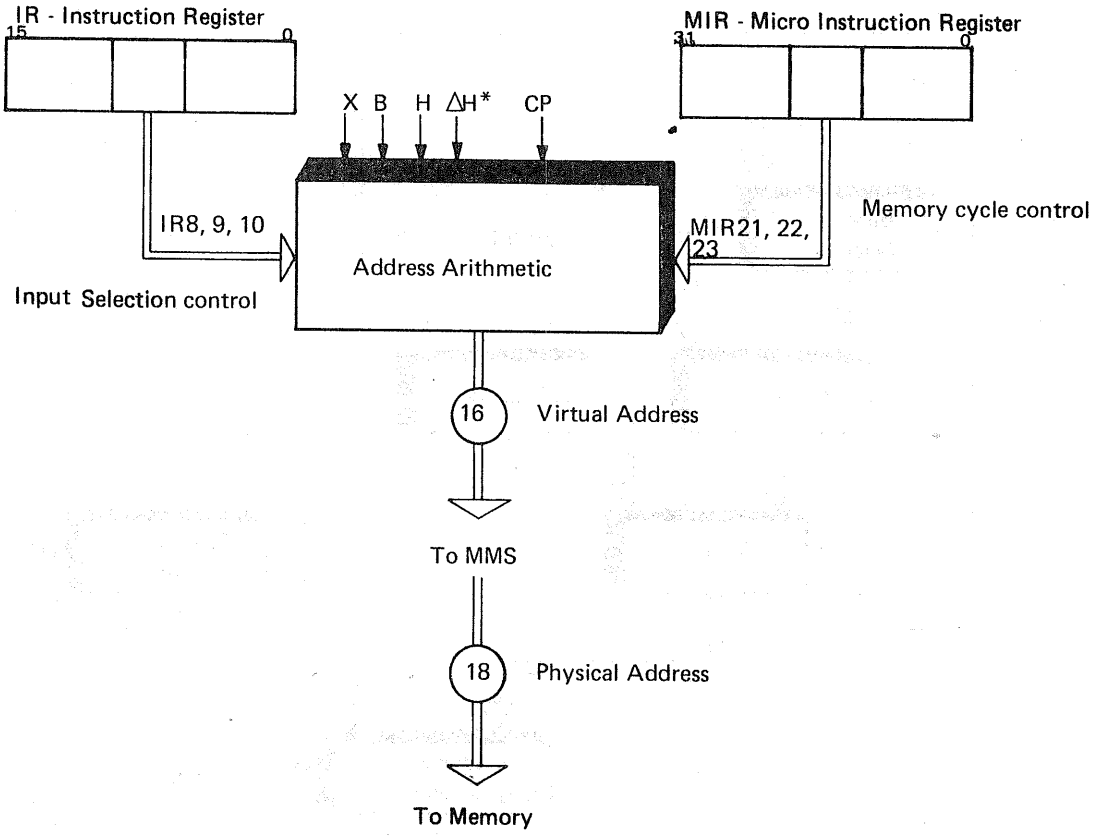


Figure II.3.6: *Address Arithmetic – Input and Control*

The data inputs to the address arithmetic (X, B, H, ΔH, CP) are selected by the 3 bits from the instruction register. The type of memory cycle that should be performed is given by the microprocessor through the Micro Instruction Register (MIR 21, 22, 23).

*ΔH is sign extended

Mainly, we have two sources for the virtual addresses sent to memory. The current program counter, CP, will always be selected and read (and incremented by one) when a new instruction is requested (cycle fetch). In all other memory referances the address will be calculated by the Address Arithmetic consisting of two adders and associated selectors. Figure II.3.7 and Table II.3.2 will help explain.

Address Source	Mode	Direction
CP	Fetch	Read
Address arithmetic	ROP	Read
Address arithmetic	RADD	Read
Address arithmetic	STO	Write

Table II.3.2: *Address Source vs. Mode*

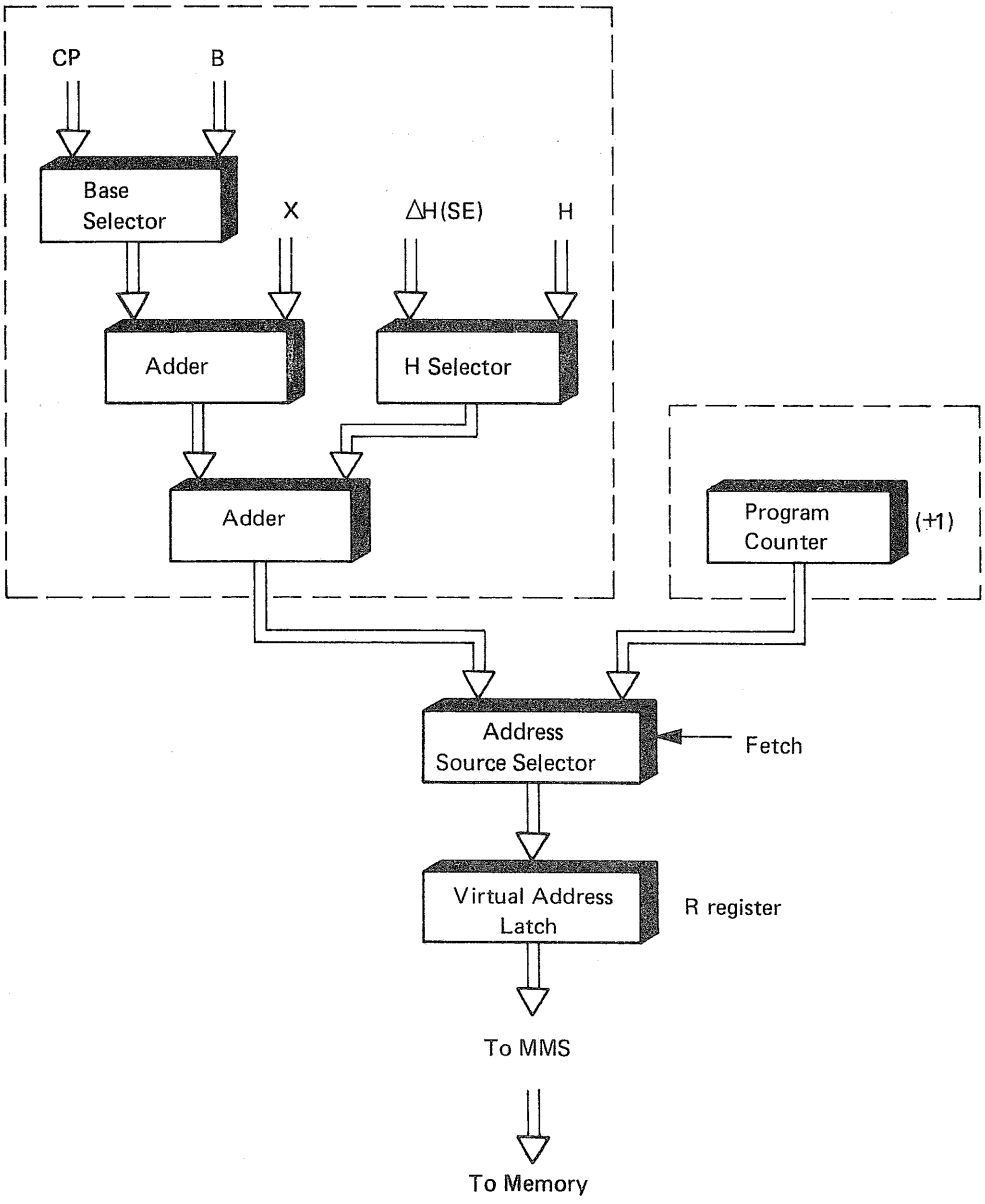


Figure II.3.7: *Address Arithmetic – Functional Operation*

II.3.3.3 *Address Arithmetic Operation*

Let us assume that the last microinstruction in a given instruction is being processed. The next machine instruction will be called for by entering the Fetch mode. The program counter (CP) is selected and clocked into the R register (Virtual Address Latch) and a memory request is made.

The instruction fetched from memory will be clocked into the Instruction and H registers.

Example 1:

Let us also assume that this instruction is an

LDA Δ - instruction

Since indexed addressing is not used, the effective address is found by adding the program counter to the displacement (Δ H).

Example 2:

Let us instead assume that the instruction was an

LDA, B I, X Δ - instruction

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

First memory cycle:

In order to find the indirect address, the B register is added to the displacement (pre-indexing). The indirect address will be loaded into the H register.

Second memory cycle:

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

II.4 MOS – MEMORY OPERATING PRINCIPLES

II.4.1 *GENERAL*

The Read/Write memory used in N-10/S is random accessed MOS memory (RAM). Principally, two classes of RAMs exist: the static and dynamic type.

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

Dynamic RAMs are devices in which the information is stored in the form of electric charge on the gate-to-substrate capacitance of a MOS transistor. This charge dissipates in a few milliseconds, and the element must be refreshed, i.e., capacitance recharged periodically.

Dynamic RAMs are important because fewer elements are involved in storing one bit of information, so that more bits can be packed into a given physical area. They also consume less power than static RAMs in the quiescent state.

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

Due to the higher packing density for dynamic RAMs as compared to the static one (ratio 4:1), dynamic memory turns out to be more suitable for memory sizes over a given limit.

Only the dynamic RAM is used in N-10/S, and only this type will be discussed in the following sections.

II.4.2 *A MEMORY CELL*

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

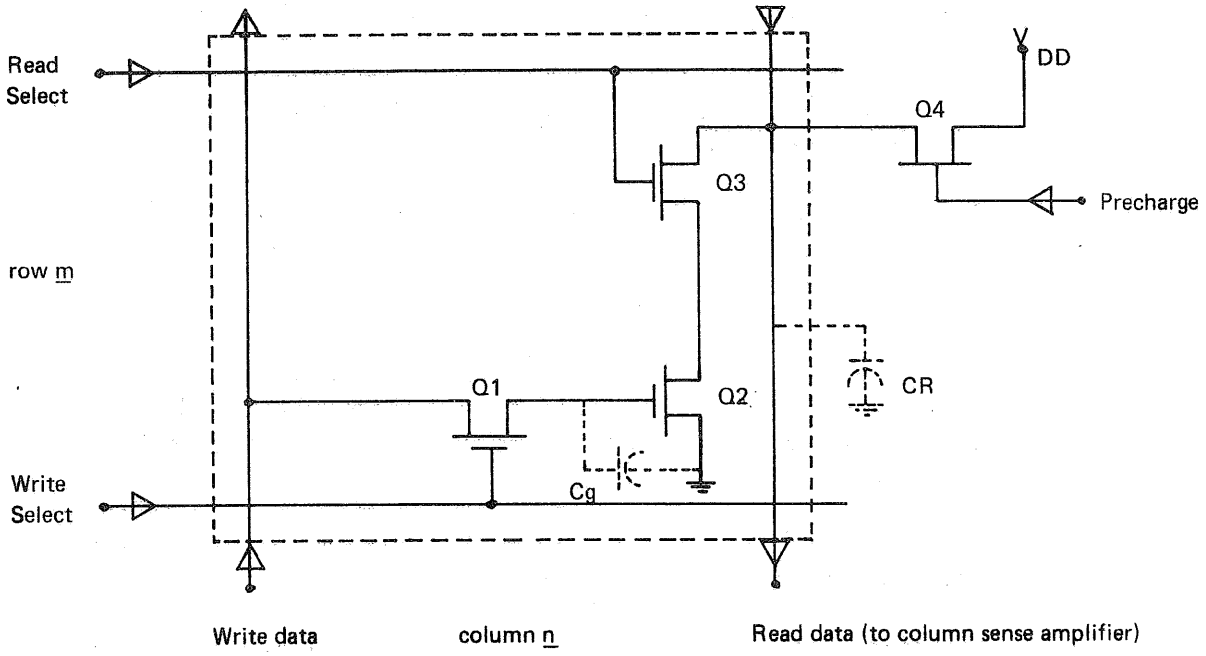


Figure II.4.1: *A Memory Cell*

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 capasitor (CR). CR is implemented by the parasitic capasitance of the ("Read Data") column line. The basic memory cell consists of a capasitor Cg formed by the gate-to-substrate capasitance 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.

II.4.3 *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.

II.4.4 *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 column line (Read Data line).

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

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

II.4.5 *REFRESH OPERATION*

Even though MOS transistors with high input impedence is 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".

II.4.6 *4K x 1 BIT MEMORY CHIP*

A memory chip is built up around a matrix of elements as previously described. For a 4K chip the matrix consists of 64 rows x 64 columns.

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 II.4.1.

Due to pin limitation, the required 12 bits address is multiplexed. Refer to Figure II.4.2.

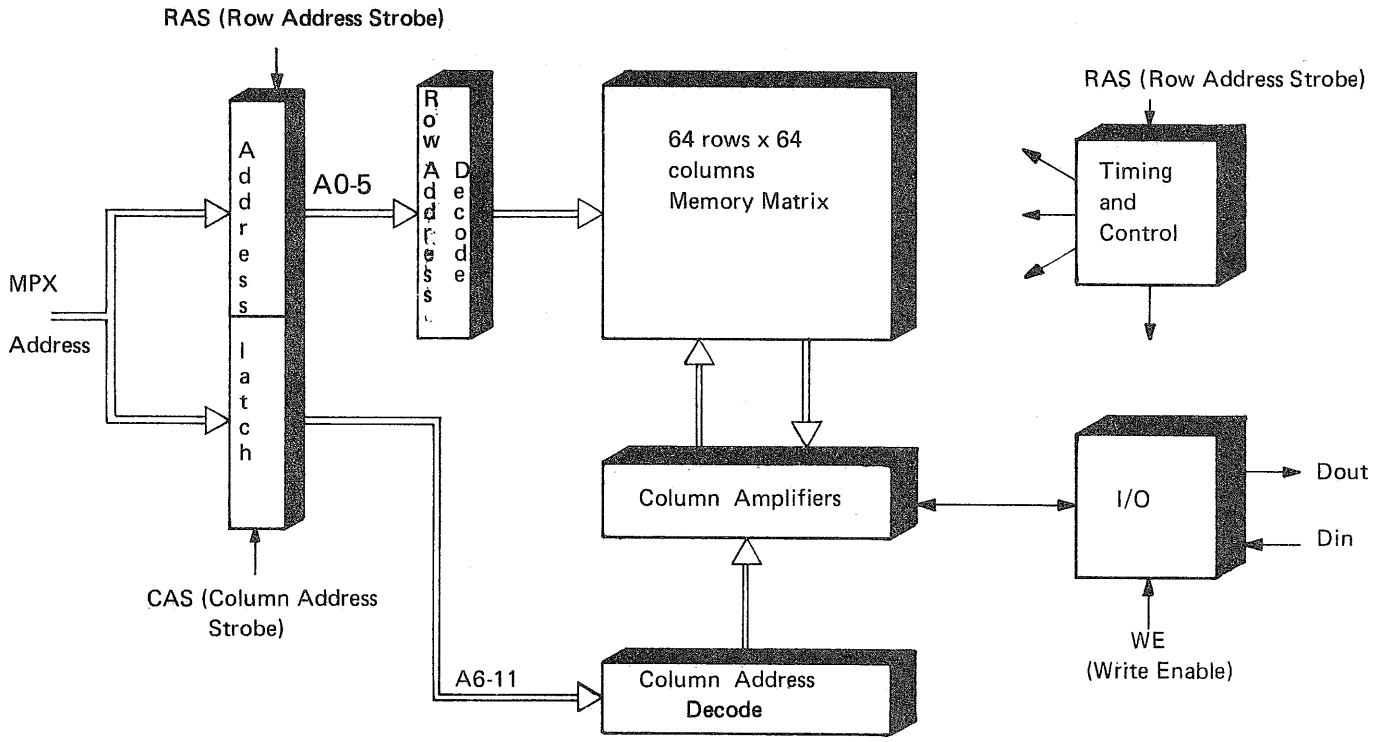


Figure II.4.2: *A Memory Chip – Functional Blocks*

II.4.7 *CHIP CYCLES*

For the following discussion, refer to block diagram II.4.2 and timing diagram II.4.3, II.4.4 and II.4.5 describing a read, write and refresh cycle respectively.

II.4.7.1 *Read Cycle*

1. Row Address Strobe (RAS) will initiate a memory cycle. A series of internal clocks will:

   a) Strobe the Row address into the Row address Latch.

   b) The proper row within the Memory matrix will be selected.

   c) All 64 columns in the selected row will be read by the column (sense amplified, latched and restored). (This action is also referred to as a refresh operation.)

2. "Column Address Strobe" (CAS) will trigger off a sequence of internal clocks which will:

   a) Strobe the Column address into the Column address Latch.

   b) The Column address is decoded.

   c) The Proper column information will be coupled to the I/O stage and latched. One bit of data has been read to the Memory board.

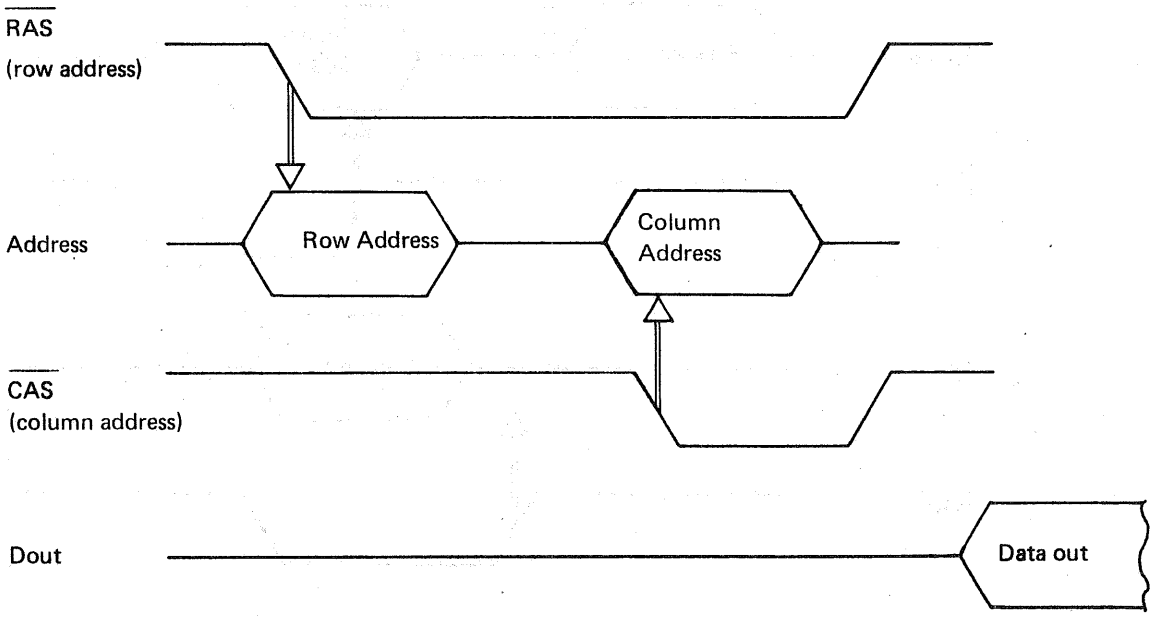


Figure II.4.3: *Read Cycle*

II.4.7.2 *Write Cycle*

1. Same as read cycle step 1.

2. "Write" is activated.

3. "Column Address Strobe" (CAS) will trigger off a sequence of internal clocks which will:

   a) Strobe the column address into the column address latch

   b) Decode the column address

   c) Latch data into Data Input Buffer.

d) The input data is forced to the selected column line and stored into the selected storage cell.

One bit of information has been stored.

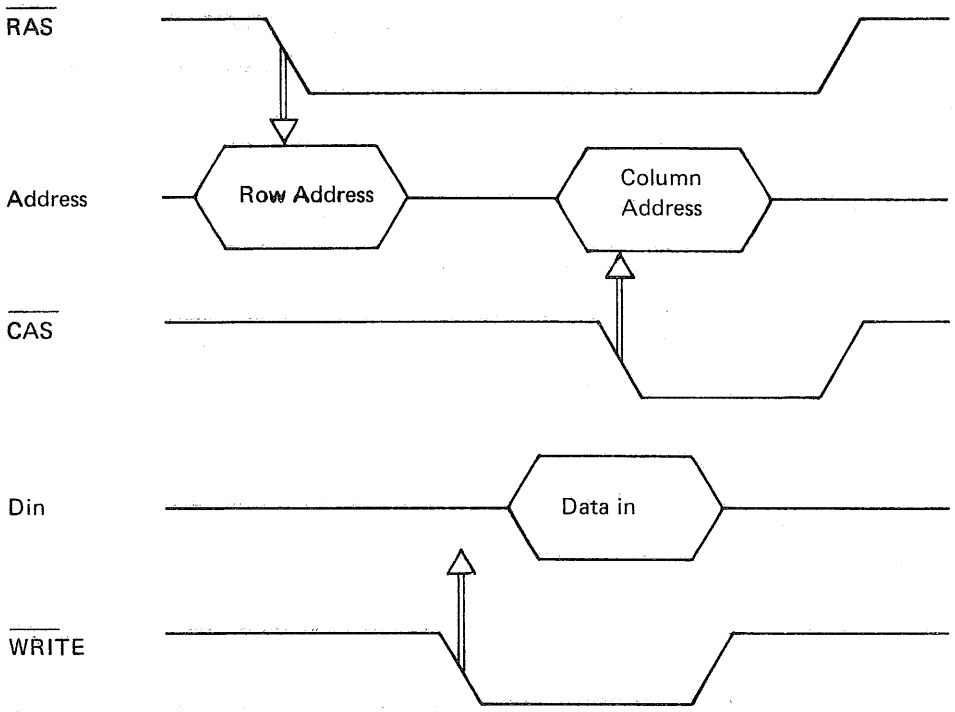


Figure II.4.4: *Write Cycle*

II.4.7.3 *Refresh Cycle*

A refresh cycle is equivalent to step 1 (only) in a read or write cycle, i.e., only the row address will be required.

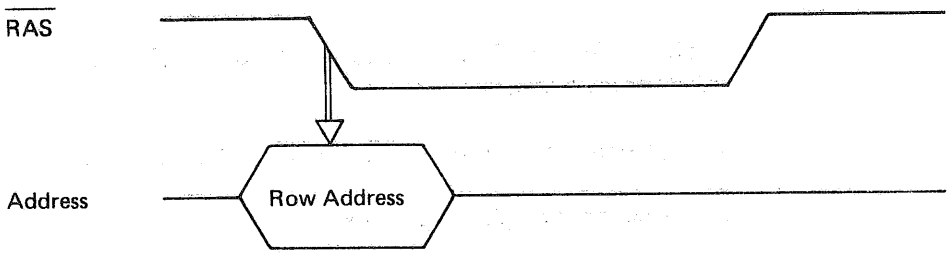


Figure II.4.5: *Refresh Cycle*

II.4.8 *16K x 1 BIT MEMORY CHIP*

Principally, 4K and 16K memory chips are alike. Since the memory matrix for a 16K chip is 128 rows + 128 columns a 14 bits multiplexed address is required. For principal operation, see Section II.4.6.

II.5 LOCAL MEMORY

II.5.1 *GENERAL*

Eight slots in the CPU card crate are reserved for Local Memory.

Two sizes of memory modules are offered, 8K words and 32K words, giving a maximum local physical address space of 64K words and 256K words respectively.

Two kinds of memory modules are available:

– 18 bits: 16 data + 2 parity bits

or

– 21 bits: 16 data + 5 error correction control bits (32K module only)

The principal operation on the memory modules are similar – the parity or error checking is performed separately. Refer to Chapter II.6.

The 8K words module uses two blocks of 4K chips, while the 32K module uses two blocks of 16K chips.

In the following discussion, we assume that 32K x 21 bits memory modules are used.

II.5.2 *ADDRESSING*

Available to all memory modules is a 18 bits physical address strobed by a "Memory Request" signal.

The three upper bits will select one of the eight modules. The address area for each module is defined by its physical position (backwiring) starting from right to left.

Table II.5.1 defines the address range for the 8 memory positions.

POS:	Address Range:
A-32	$0 - 77\,777_8$ (0 – 32K)
A-31	$100\,000_8 - 177\,777_8$ (32 – 64K)
A-30	$200\,000_8 - 277\,777_8$ (64 – 96K)
A-29	$300\,000_8 - 377\,777_8$ (96 – 128K)
A-28	$400\,000_8 - 477\,777_8$ (128 – 160K)
A-27	$500\,000_8 - 577\,777_8$ (160 – 192K)
A-26	$600\,000_8 - 677\,777_8$ (192 – 224K)
A-25	$700\,000_8 - 777\,777_8$ (224 – 256K)

Table II.5.1: *Physical Address Range*

The least significant address bit determines block x or y within the selected memory module. Address bits 1 thorugh 7 will be used as row address while bits 8 – 14 will be used as column address. Refer to Figure II.5.1.

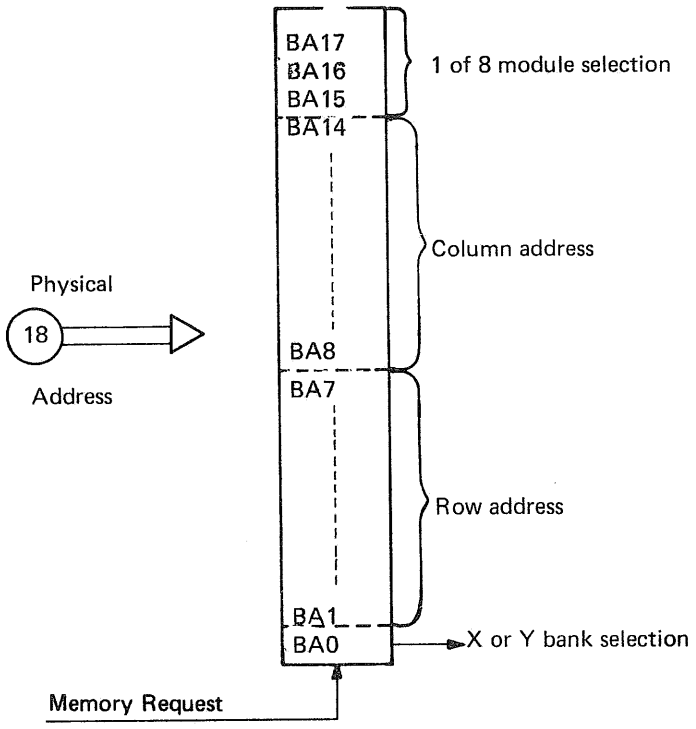


Figure II.5.1: *Address Decoding*

II.5.3 *DATA*

The memory data bus is 21 bits wide. All 21 bits are used if a 5 bits error correction code is attached to the 16 data bits. Otherwise, 2 parity bits are used. Refer to Figure II.5.2. Refer also to Section II.6.2, describing error correction.

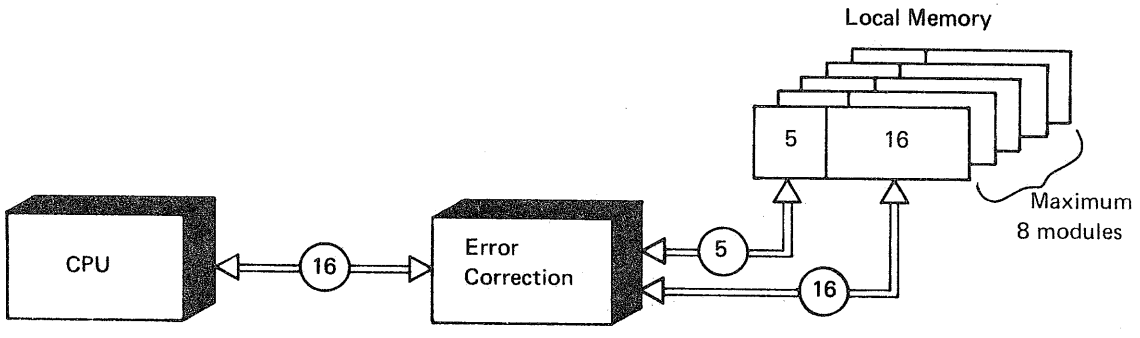


Figure II.5.2: *Error Correction*

II.5.4 *CONTROL*

Local memory operates asynchronous with the CPU. Each memory module has its own timing, allowing modules of different speed in Local memory.

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

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

All those sources again operate asynchronously with respect to each other.

A priority and allocation circuitry is therefore required.

For proper operation, different control signals must be established.

However, the CPU is not willing to given memory this attention, a Memory Interface is therefore required. The Memory Interface has the Error Correction Network as its co-worker. This will be described separately in Section I.6.

Figure II.5.3 illustrates the control signals involved in CPU to Memory communication for transfer of 1 word of data.

A memory reference starts with a "CPU Request" along with a "WRITE" signal defining data direction flow. The "Request" signal may be stopped if the desired data is found in "Cache" memory. Refer to Section II–9. Accompanied with the request is an 18 bits valid address formed by the CPU via the Memory Management System (MMS). (Refer also to Section II–8.

If Local Memory is in an idle state ($\overline{\text{Memory Busy}}$), the memory cycle is started. On the other hand, if Memory is busy the request will be put in a hardware waiting queue, ranked in accordance with priority. When the memory module has been selected and the address latched. "Address Ready" is returned to the Memory Interface allowing a new request to be processed.

For a store operation, "Data Ready" from memory would occur at the same time generating "Memory Data Ready" for the CPU indicating the termination of the store cycle.

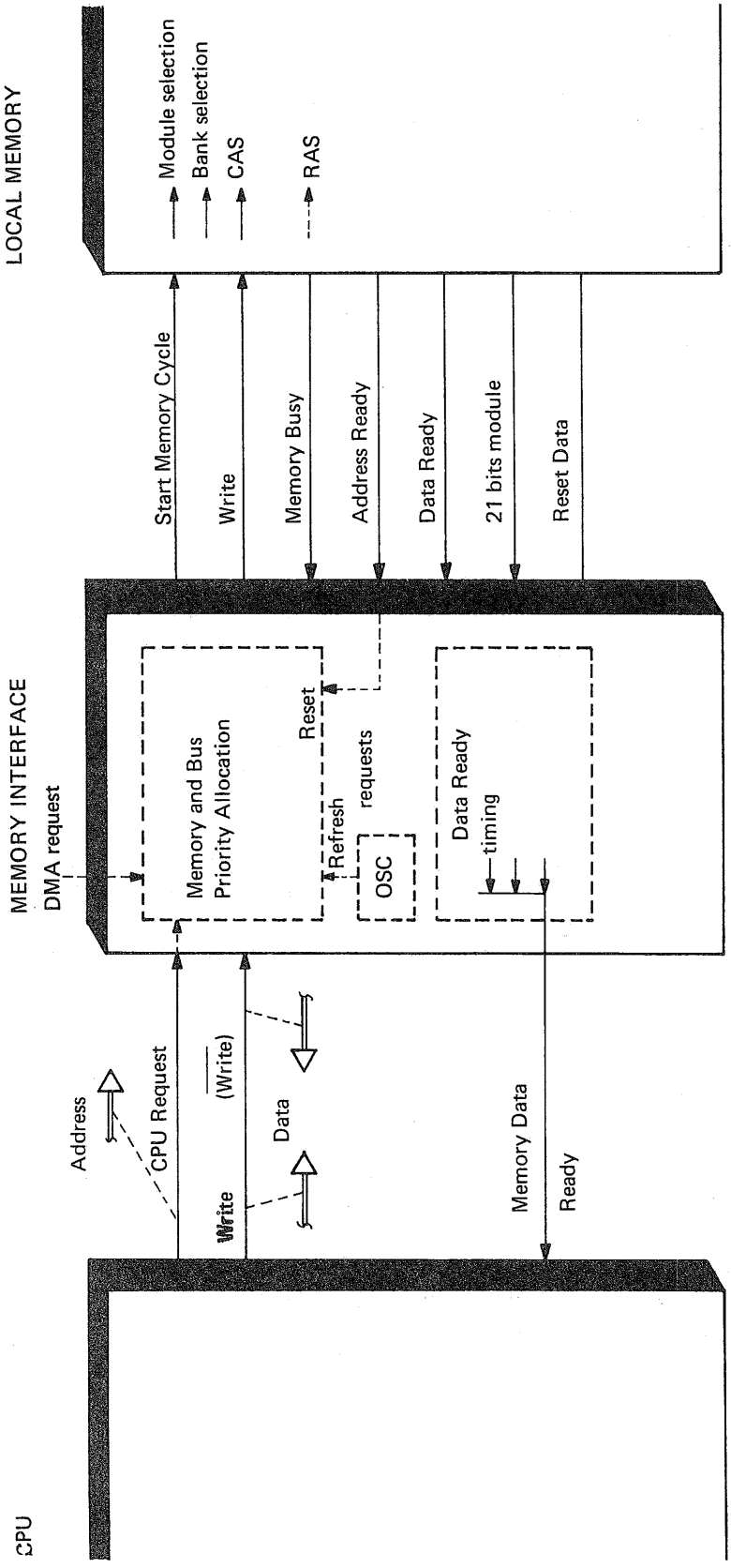


Figure II.5.3: *Memory Control*

If this was a read operation ($\overline{\text{Write}}$), "Address Ready" would open for a new request. After a short access delay, data from Memory is presented indicated with "Data Ready". Data will be parity checked (18 bits) or error checked (21 bits) and corrected if necessary (refer to Section I.6), in accordance with the simultaneous state of "21 bits module" line.

After a delay the Memory Interface will send "Memory Data Ready" indicating valid data on the input to CPU. "Reset data" will be sent to Memory to drop the data.

By using "Address Ready" and "Data Ready" lines from Local Memory, the memory band width is increased. Figure II.5.4 will help explain by showing two consequtive requests processed with or without use of "Address Ready".

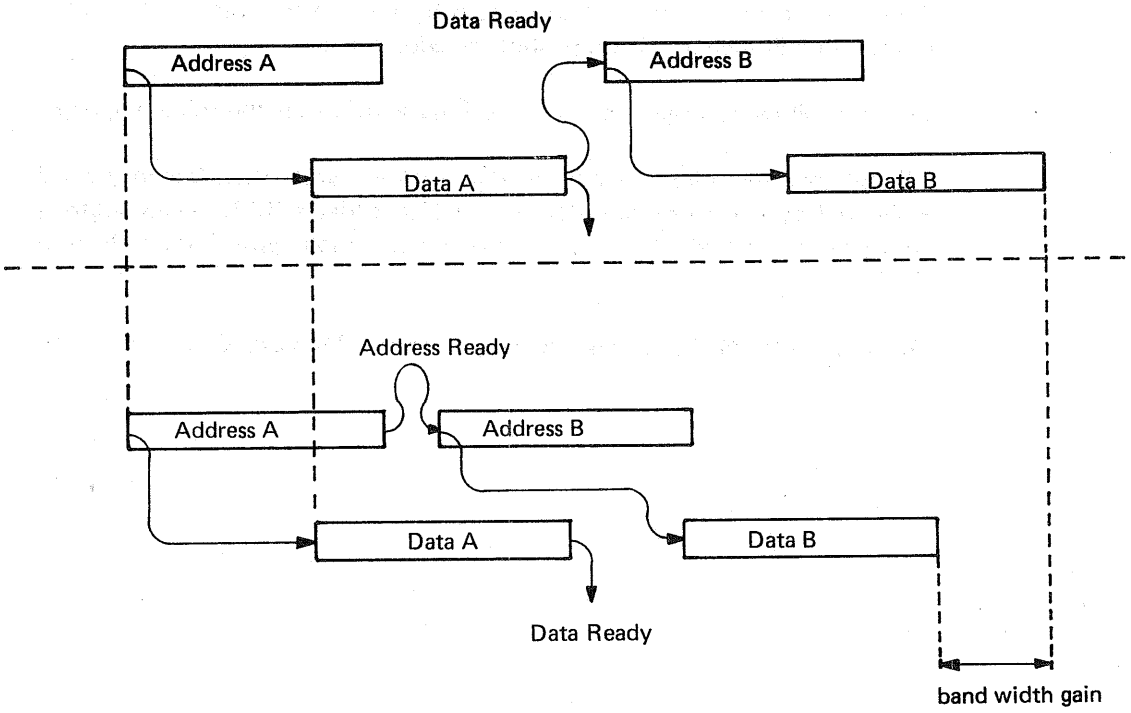


Figure II.5.4: *Data and Address Ready Timing*

II.5.5 *REFRESH*

As previously mentioned, dynamic MOS memory must be refreshed periodically.

An oscillator located in the Memory Interface generates the "Refresh Requests".

A refresh operation works simultaneously on one "row" in both blocks in all memory modules.

Since a 16K chip has 128 rows (= 2^7), 7 bits are sufficient to address all rows.

A circular request counter incremented by the "refresh requests" located in the Bus Brancher accomplishes the addressing.

For the following discussion refer to Figure II.5.5 on the following page.

All memory modules must be in the quiescent state (i.e., Memory Busy = 0), before a refresh can start. A 7 bits address BA1-7 (row address counter) is enabled onto the address bus along with "Start Refresh Cycle".

Memory is ready for a new operation when "Memory Busy" goes off.

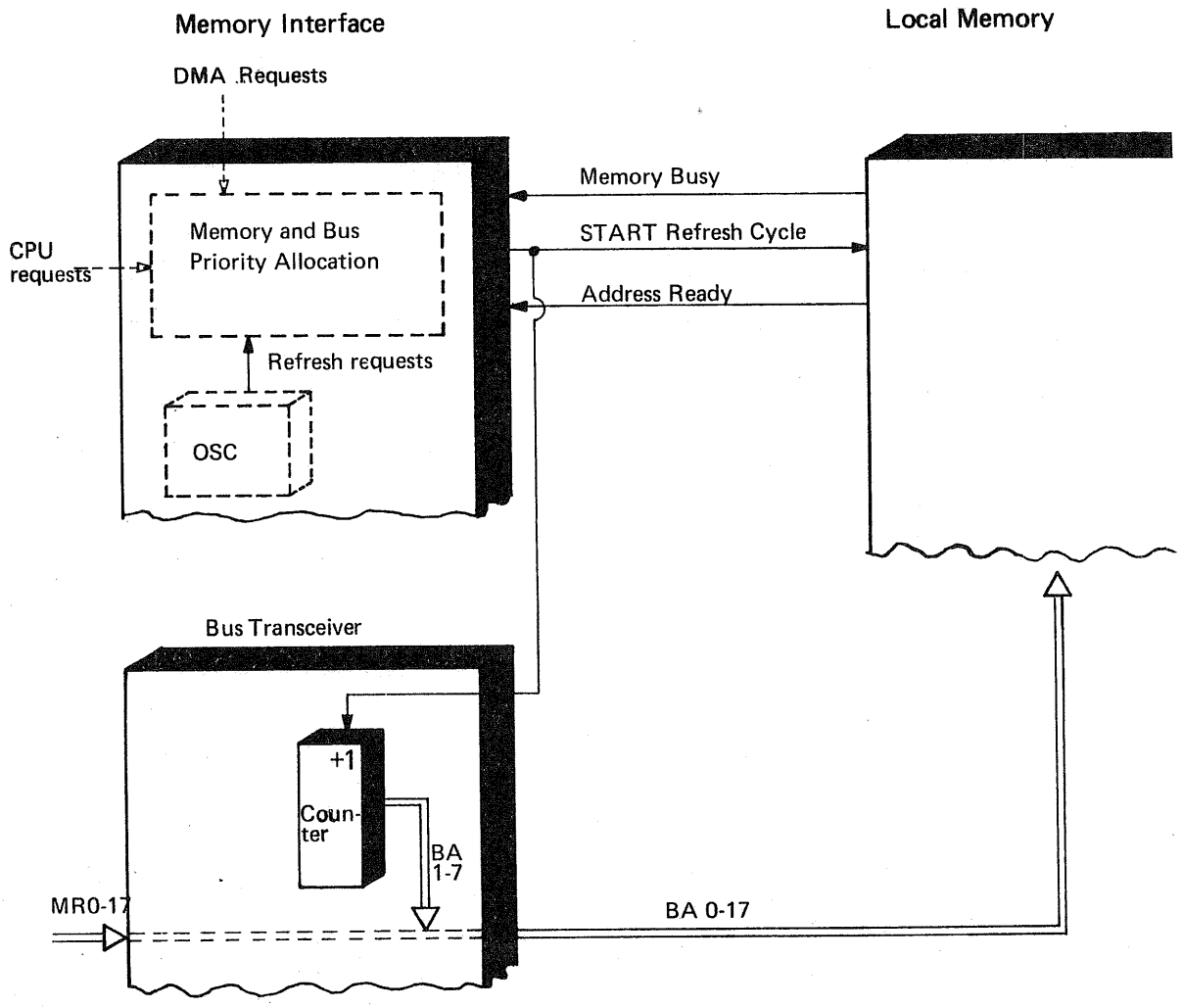


Figure II.5.5: *Refresh*

II.6 ERROR CHECKING AND CORRECTION

II.6.1 *GENERAL*

The increased demand for more reliable data storage systems has lead to development of more sophisticated error detection and correction schemes. 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.

Estimating 8 to 9 of 10 failures in a computer system, excluding I/O, to be caused by memory, a single error correction system would then improve Mean Time Between Failure (MTBF) 5 to 10 times.

Error detection and correction in a memory system is adopting a modified Hamming code working on 16 bits in parallel.

The big advantages of error correction are:

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

This can be implemented with the cost of:

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

However, great efforts have been made in using fast hardware which is reduced to a minimum. The overhead is the same as the more traditional parity circuits.

II.6.2 *ERROR CHECKING AND CORRECTION – Functional Description*

The Error Correction Network is located in the Bus Transceiver between Local Memory and the CPU. Refer to Figure II.5.2. Error detection and correction is realized by generation of a code obtained by a parity matrix working on various combinations of bits from data field.

This code is attached to the data field and stored in memory. When reading, this code is again generated from the data field. This code is then compared with stored code accompaning the data.

If the two codes are equal, the data read has been the same as data stored, i.e., no error has occurred. This is indicated by generating a syndrome equal to zero. The syndrome is obtained by exclusive OR-ing the two control codes. However, if an error occurred the two codes are non-equal giving a non-zero syndrome which also indicates if the error can be corrected and which bit was in error (for single errors).

Since an error may also occur to the code bits, the syndrome should also indicate which code bits were in error. Minimum syndrome combinations will then be as depicted in Table II.6.1.

	Minimum Codes	Codes in Use
No error:	1	1
Single data error:	16	16
Single code errors:	5	5
Multiple error:	1	10
	23	32

Table II.6.1: *Correction Codes Usage*

Since a 5 bits code is used, that gives $2^5 = 32$ combinations.

The Error detection and correction network may be divided into 4 sections as depicted in Figure II.6.1.

The 5 bits code generator is implemented by 5 parity generators, each working on 8 carefully selected data bits. The bits involved in each parity generator can be depicted from Table II.6.2.

A combination of odd and even parity is used to prevent catastrophic failures as all 21 bits are 1's or 0's. The 5 parity bits, are referred to as a Control Code which is attached to the data sent to memory (16 + 5 = 21 bits).

When reading, the stored Control Code (old) will bit for bit be compared with the generated code (new). If they match, the 5 syndrome bits will be zero. A non-zero syndrome, indicating an error, will be decoded by the Syndrome Decoder to tell which error occurred, and if the error can be corrected or not.

Assuming that a single data bit was in error, the corresponding E-line (E0-15) will be activated. The Data Corrector is implemented by 16 exclusive OR gates which may be regarded as 16 controlled inverters, the E-lines being the control lines.

Thus, the data bit going through the Exclusive OR where the E-line is activated, will be inverted. Working in the binary system, a wrong bit will be a correct one.

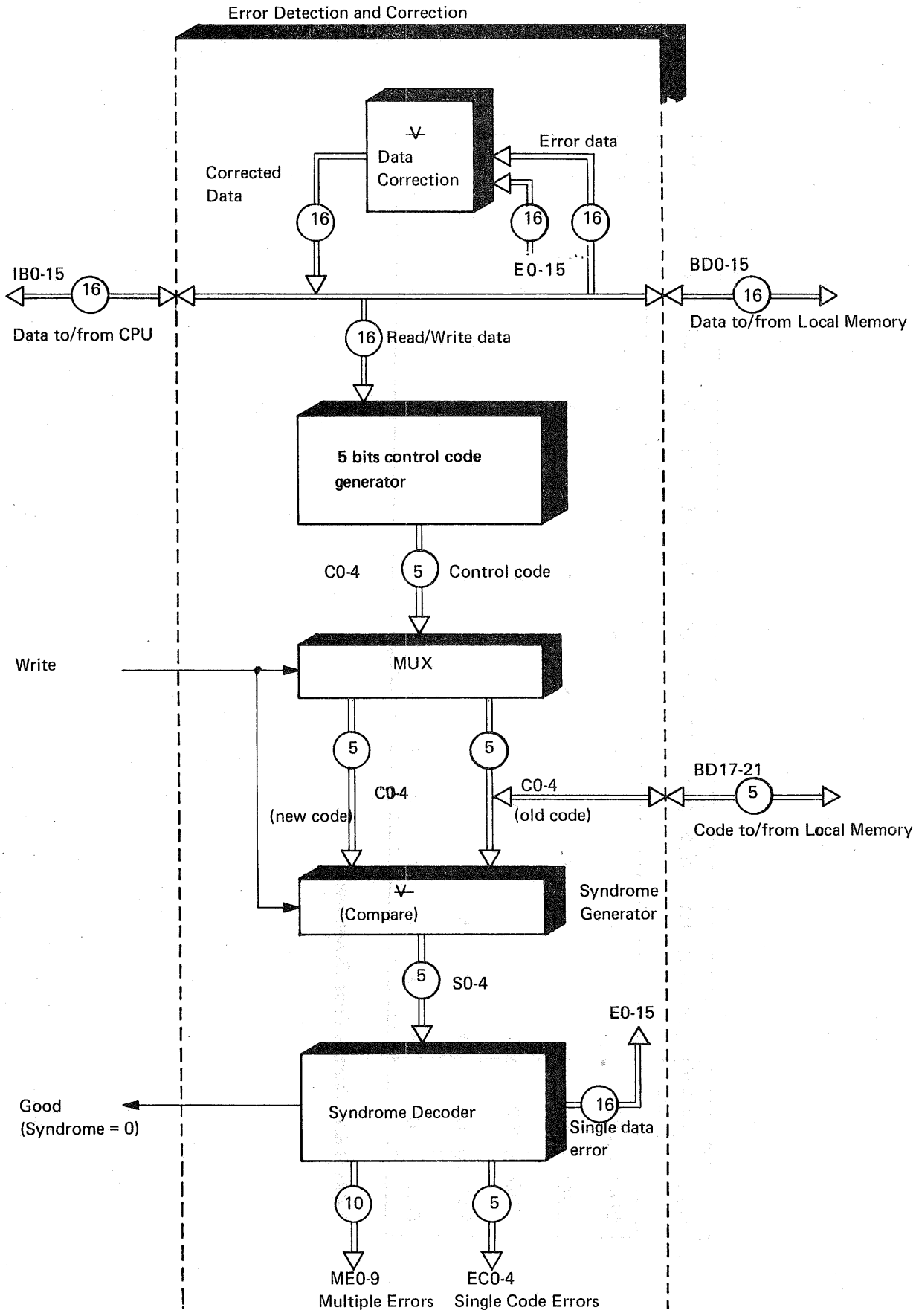


Figure II.6.1: *Error Detection and Correction*

ND-06.009.01

Parity	Control Code	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
ODD	C0						x		x		x		x	x	x	x	x
EVEN	C1		x		x		x	x		x			x		x		x
ODD	C2	x			x	x				x	x	x			x	x	
EVEN	C3	x	x	x						x	x	x	x	x			
ODD	C4	x	x	x	x	x	x	x	x								

Table II.6.2: *Control Code Generation*

S4	S3	S2	S1	S0	No Error	Single Data Error	Single Code Error	Multiple Errors
0	0	0	0	0	Good			
0	0	0	0	1			EC0	
0	0	0	1	0			EC1	
0	0	0	1	1		E0		
0	0	1	0	0			EC2	
0	0	1	0	1		E1		
0	0	1	1	0				ME0
0	0	1	1	1		E2		
0	1	0	0	0			EC3	
0	1	0	0	1		E3		
0	1	0	1	0				ME1
0	1	0	1	1		E4		
0	1	1	0	0		E5		
0	1	1	0	1		E6		
0	1	1	1	0		E7		
0	1	1	1	1				ME2
1	0	0	0	0			EC4	
1	0	0	0	1		E8		
1	0	0	1	0		E9		
1	0	0	1	1		E10		
1	0	1	0	0		E11		
1	0	1	0	1				ME3
1	0	1	1	0		E12		
1	0	1	1	1				ME4
1	1	0	0	0		E13		
1	1	0	0	1				ME5
1	1	0	1	0		E14		
1	1	0	1	1				ME6
1	1	1	0	0		E15		
1	1	1	0	1				ME7
1	1	1	1	0				ME8
1	1	1	1	1				ME9

Table II.6.3: *Syndrome Decoding*

Data has been corrected and can be presented for the CPU. Table II.6.3 shows the errors indicated with the syndrome bits. The 5 syndrome bits are contained in PES (Parity Error Status) bits 9 through 13 and may be read by the CPU.

II.6.2.1 *Comments to Tables II.6.2 and II.6.3*

- From Table II.6.2 we notice that a given data bit is processed by 2 or 3 parity generators. An erroneous data bit will thus produce a syndrome code with 2 or 3 bits set, respectively. For example, an error occurred to data bit 0, S0 and S1 will be set producing a code equal to 3. Refer also to Table II.6.3.

- If a control bit is in error, only one of the 5 bits will be set. Refer to Table II.6.3.

- All other codes represent multiple errors, and data cannot be corrected.

II.6.3 *PARITY*

The more ordinary data reliability control is accomplished by parity checking/generation. Two parity bits are used, one for upper byte and one for lower byte. No correction can be made to data accompanied by parity bits.

Table II.6.4 and Figure II.6.2 help illustrate.

- It should be noticed that error occurring on the parity bit(s) will be interpreted as parity error on data.

- Simultaneous errors occurring to even number of data bits within a byte including the parity bit, will not be discovered.

Parity	Parity Bits	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
ODD	PO(C4)	x	x	x	x	x	x	x	x								
ODD	PI									x	x	x	x	x	x	x	x

Table II.6.4: *Parity Generation*

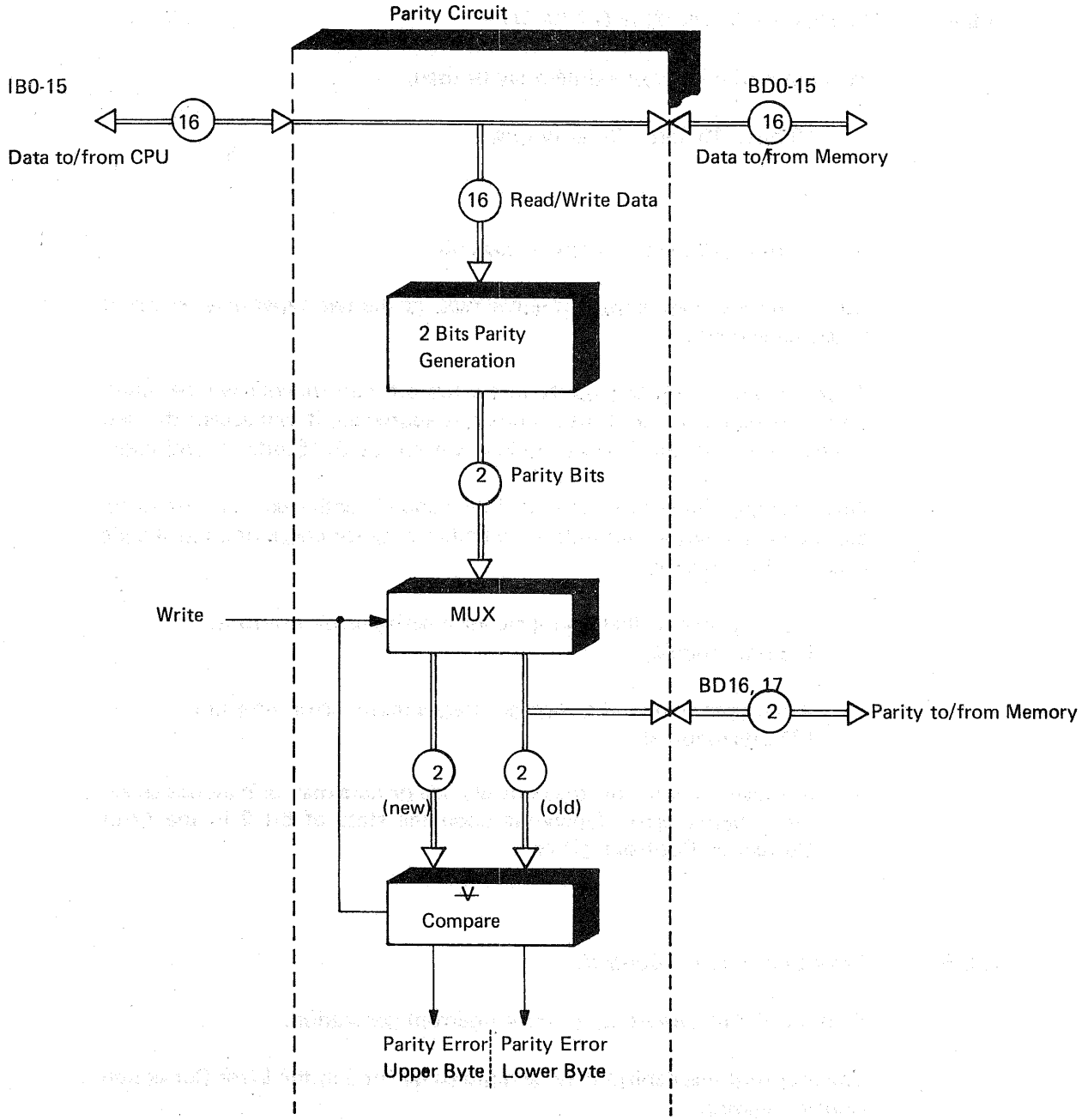


Figure II.6.2: *Parity Generation and Detection*

II.6.4 *ERROR CORRECTION OR PARITY*

Two kinds of memory modules may be used:

– 18 bits: 16 data + 2 parity bits

or

– 21 bits: 16 data + 5 control code bits

Local memory may consist of either type, or the two types may be mixed in the same memory.

When writing, a two bits parity and a five bits control code will be generated in parallel. If an 18 bits module is addressed, it will accept the two parity bits while the 21 bits module will accept the 5 bits control code.

When reading the control line "21 bits module", activated from the accessed memory module will indicate whether a parity check or control code check is the proper one.

– A parity error will always generate a parity error interrupt. (18 bits module)

– A multiple error will always generate a parity error interrupt (21 bits module)

– A single bit error on the control code or data may or may not generate a parity error depending upon the state of bit 2 in the Error Correction Control register.

II.6.5 *Error Correction – Control*

Figure II.6.3 illustrates parity error interrupt generation.

The interrupt mechanism may be disabled by bit 3 in the Error Correction Control register.

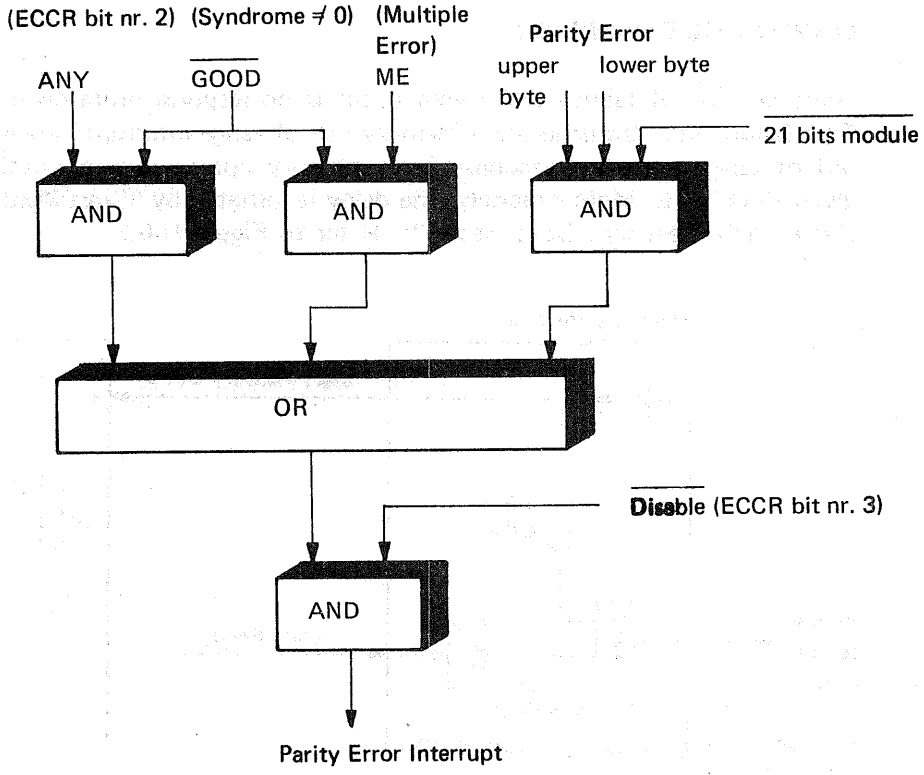


Figure II.6.3: *Parity Error Reporting*

The Error Correction Control Register is set by executing a TRR ECCR instruction. The format and bit assignment is given below:

	3	2	1	0
NA	DISB	ANY	TEST 15	TEST 0

ECCR – Error Correction Control Register

Note: Bits 0, 1 and 3 are used for test purposes only.

Bit 0: will force data bit 0 to a one, thus giving parity error

Bit 1: will force data bit 15 to a one, thus giving parity error

Bit 2: Parity interrupt control bit
If this bit is a zero, only multiple errors will generate a parity error interrupt. If this bit is a one, all errors will generate a parity error interrupt.

Note: This bit has effect on 21 bits memory modules only.

Bit 3: Disable
When this bit is set, error correction and parity error interrupt are disabled.

II.6.6 *MEMORY OUT OF RANGE*

Another type of failure that might occur is no response from memory. This failure will also generate a Memory out of range interrupt. Memory out of range means no response from memory upon a request within a given time limit. More precisely, the delay is initiated by "Start Memory Cycle" and reset by "Data Ready". Refer to Figure II.6.3.

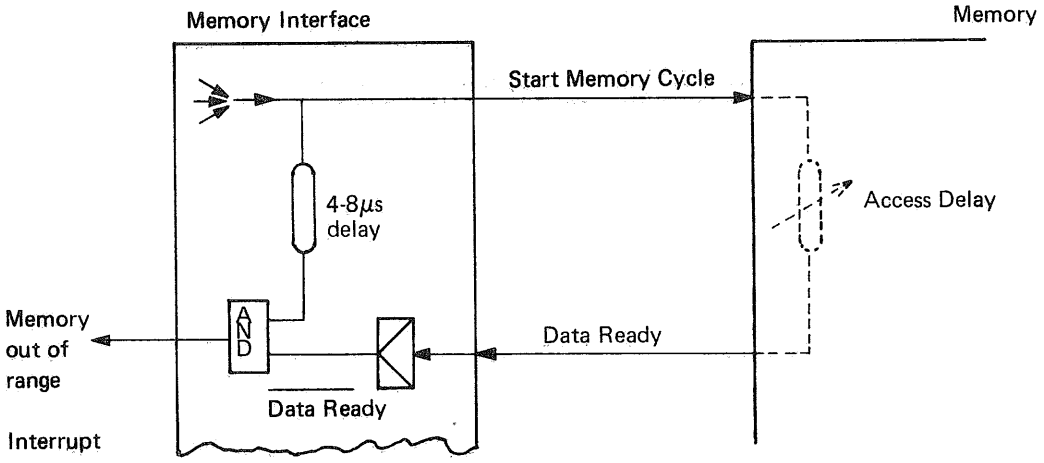


Figure II.6.4: *Memory Out of Range – Generation*

II.6.7 *CPU FEEDBACK INFORMATION*

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

PES – Parity Error Status

and

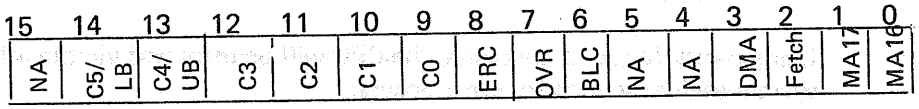
PEA – Parity Error Address

The registers will be read by executing TRA PES and TRA PEA instructions.

PEA (Parity Error Address) holds the lower 16 bits address of the latest memory reference, while PES (Parity Error Status) holds further information regarding the error.

As soon as a memory error occurs, the PES and PEA will be blocked, thus preventing overwriting.

The PES has the following format:



PES – Parity Error Status

Bits 0-1: Address bits 16 and 17 of the offending address

Bit 2: Error occurred during instruction fetch.

Bit 3: Error occurred during a DMA reference.

Bits 4-5: Not assigned.

Bit 6: Blocked. Memory error has occurred.

Bit 7: Overrun, i.e., MOR after a parity error. Overrun given no information that more than one parity error has occurred before the interrupt is honored.

Bit 8: Error Correction.
Error has occurred on a 21 bit module. Bit 9-13 will hold a code giving additional Error information.

Note: If a parity error has occurred on an 18 bits module, this bit will remain a zero. Bits 9-12 do not give any relevant information, while bits 13 and 14 indicate which byte the parity error occurred in.

Bits 9-13: If bit 8 is set, the code given by these bits will indicate what kind of error occurred. If bit 8 is cleared, bits 9-12 do not hold valid information, while bit 13 indicates parity error in upper byte.

Bit 14: Parity error occurred in lower byte.

Bit 15: Not assigned.

The blocking of PES and PEA is released as PEA is read (TRA PEA). This means that PES should always be read before PEA.

II.6.8 *ERROR LOGGING*

Using parity for error checking, the CPU will be informed via the interrupt system each time a parity error occurs.

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

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

A periodic RT program, belonging to the Operating System, will operate on the ANY-bit (Error Correction Control bit 2). Single error information may then be put on an error file. This file is useful from a maintenance point of view.

II.7 MULTIPORT MEMORY SYSTEM

II.7.1 *GENERAL*

A multiport memory system can be used for two major applications:

1. Increase memory band width to enable for higher data transfer rates to and from main memory.

2. Multiprocessor system communication through a common memory system.

Figure II.7.1 shows an example of multiport memory usage. In this example the two DMA devices, disk and drum, which are regarded as one channel, are physically connected to port 0. The CPU channel is connected to port 1. This figure also depicts that local memory may be used together with the multiport memory system. The CPU and DMA devices using the main I/O bus (does not have its own multiport channel) may use either local memory or port 1.

Figure II.7.2 shows a two processor system each having local memory (64K) and common multiport memory consisting of 2 banks each of 64K.

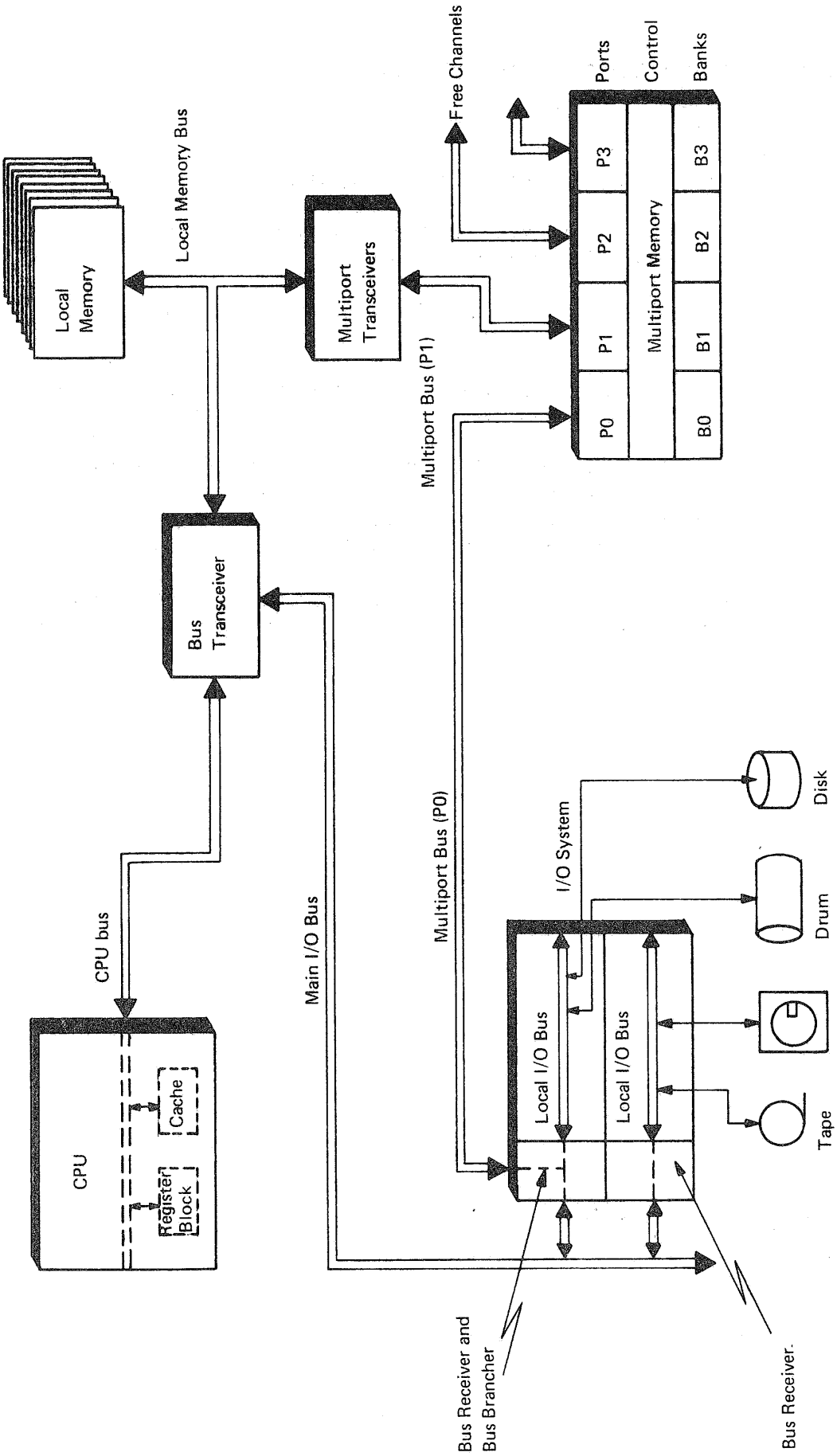


Figure II.7.1: *Storage Interconnection*

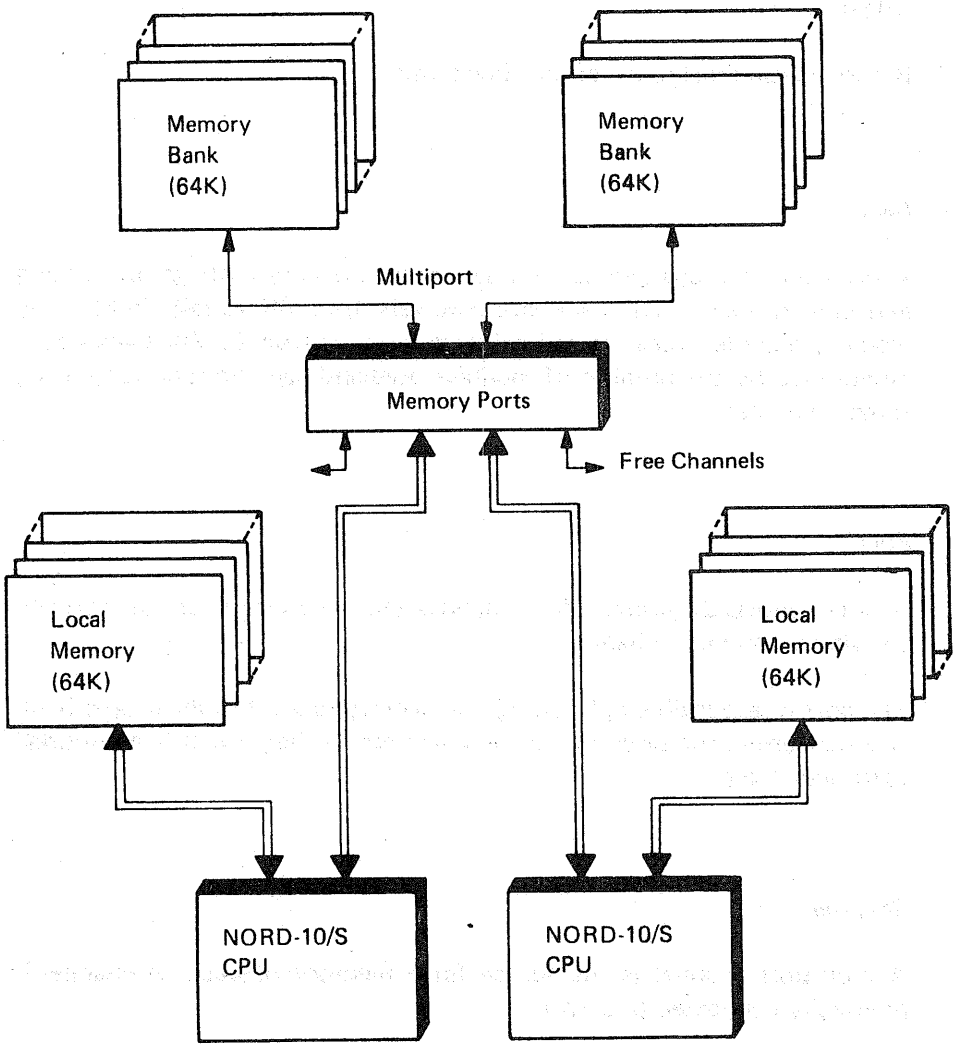


Figure II.7.2: *NORD-10/S Two-Processor System*

II.7.2 *BASIC UNIT*

Before continuing let us define a few terms.

II.7.2.1 *Bank*

A bank may be regarded as an independent memory with its own timing and control logic. The bank size may vary from 8K to 64K in 8K (one memory module) steps. Each bank address starts from 0. The bank size is determined by the number of modules used and bank limit switch setting (described later).

II.7.2.2 *Port*

A port consists of receivers for addresses and transceivers for data to/from a multiport memory channel.

The port is responsible for routing the memory request to the proper bank and converting the physical memory address to the correct bank address (described later).

II.7.2.3 *Channel*

A multiport channel is the source for a memory request. A channel is physically connected to a port.

The basic unit consists of one card crate located in the rear of the cabinet. It is prewired for two memory banks and four ports.

The advantage of higher memory band width is obtained by using minimum two memory banks and minimum two ports.

A given port has access to both banks within the unit. The bank selection is determined by the address range.

If two or more ports try to access the same bank simultaneously, a priority network, located in the control logic for each bank, will solve the problem. The priority is fixed and linear. This enables a high priority channel to run at maximum transfer rate on one bank, while blocking accesses from channels with lower priority to that bank.

II.7.3 *Card Crate*

Figure II.7.3 shows the principal layout of the card crate.

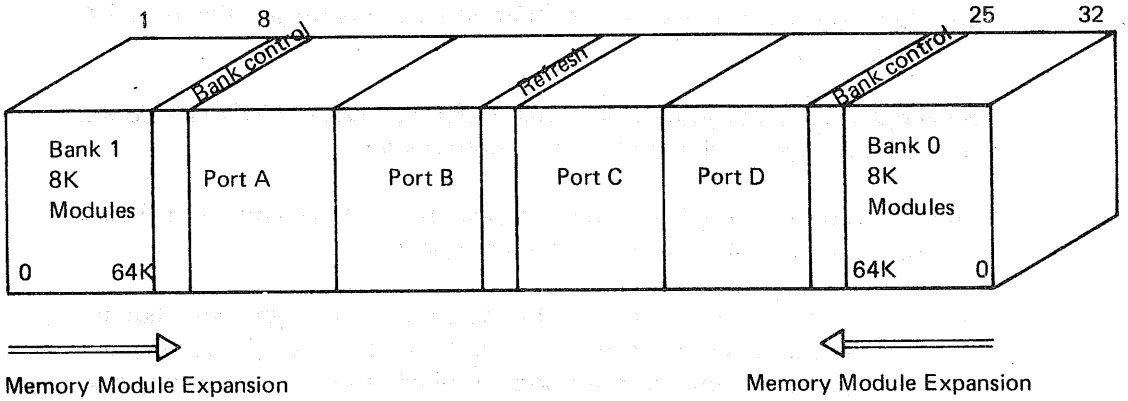


Figure II.7.3: *Card Crate Principal Layout*

Each bank may be expanded up to 64K.

II.7.3.1 *Expansion*

If more than 128K of memory or more than two banks are required, another card crate will be installed. A given channel will then be connected to the same given port in the two card crates. A given channel has then access to all four memory banks. If each bank is limited to 32K, a maximum of four card crates may be installed giving a system of four independent channels and eight independent and parallel accessible banks.

The maximum address space for each channel is limited by the 18 bits address bus to 256K.

II.7.3.2 *Physical Memory Space Organization*

As depicted in Figure II.7.1, local and multiport memory may be used in the same system. Local memory has the advantage of shorter access time and error correction facilities while multiport has the advantages mentioned in the general discussion.

Since the address bus is limited to 18 bits, total physical memory may not exceed 256K. The CPU and the DMA devices using the main I/O bus may use local memory as well as multiport memory, while the other multiport channels only have access to the multiport address range. Figure II.7.4 may help explain.

Switches in the multiport transceiver (refer to Figure II.7.1) will define the address range for the multiport memory system.

When the address issued is within the predefined range, control and data are routed to/from the multiport memory system.

Note: Control and data will also be sent to local memory but no response will be given since local memory should not hold any physical memory in the address range of the multiport memory.

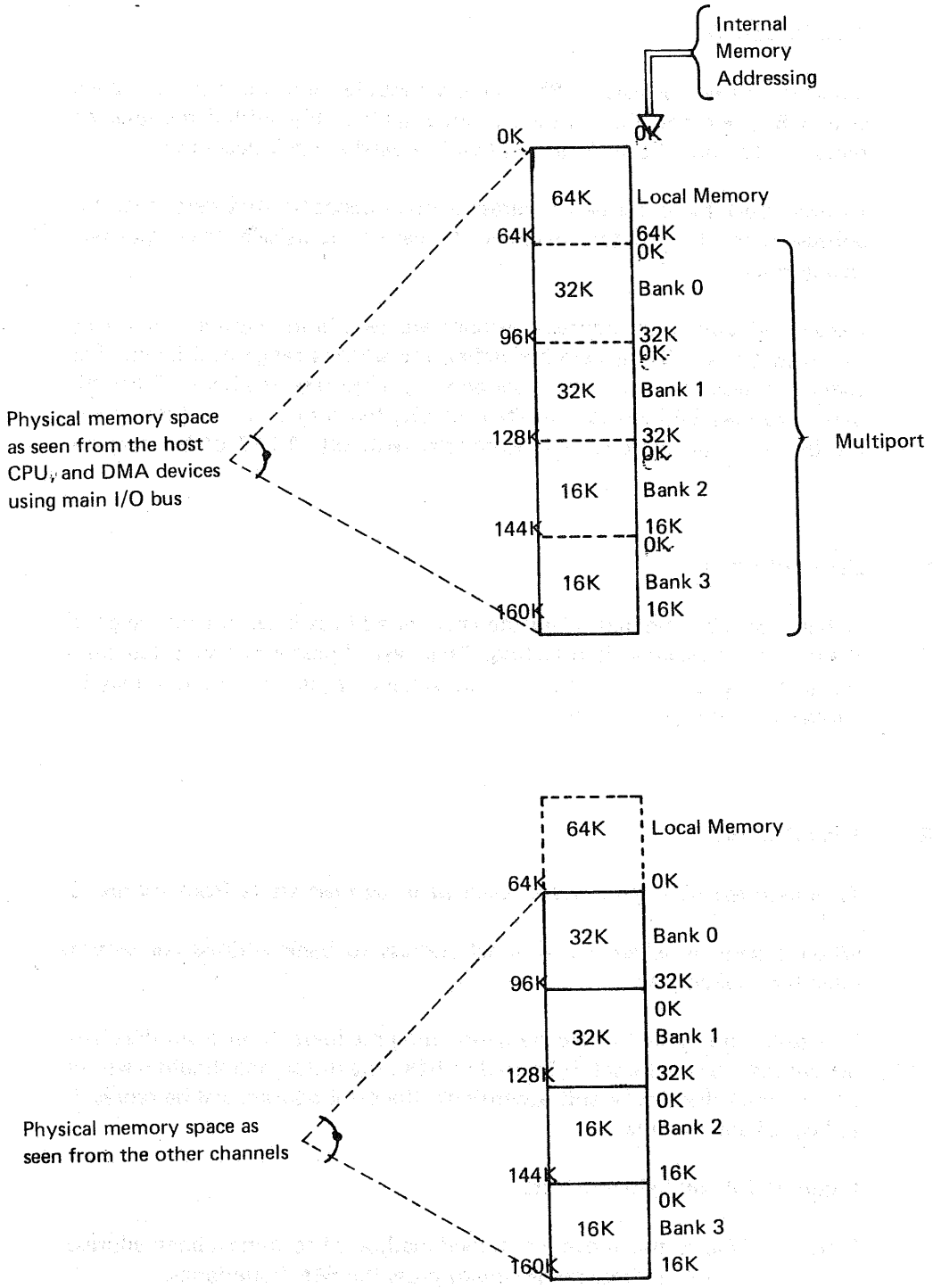


Figure II.7.4: *Physical Memory Space*

II.7.4 *ADDRESSING*

Since the bank consists of 8K memory modules and the minimum bank size is 8K, address bits 0-12 are infernal address bits within the selected module. The upper bits, bits 13-17 will be used as bank decoding.

In each port there are two separate address decoders that determine the address space for the two banks in the card crate as seen from that particular port.

Associated with each address decoder are two limit registers set up by micro-switches. Those switches define the address range of a bank. The two limit registers set up the lower and upper address of a bank. Since the bank size may be increased in steps of 8K, the resolution of the switches are 8K, i.e., the switches are associated with bits 13-17 of the address.

II.7.4.1 *Bank Selection*

A bank will be selected when the channel address is within the range of the upper and lower switch setting. This test if performed simultaneously for all banks upon a channel request. A bank request or selection may be represented in Figure II.7.5.

II.7.4.2 *Bank Address*

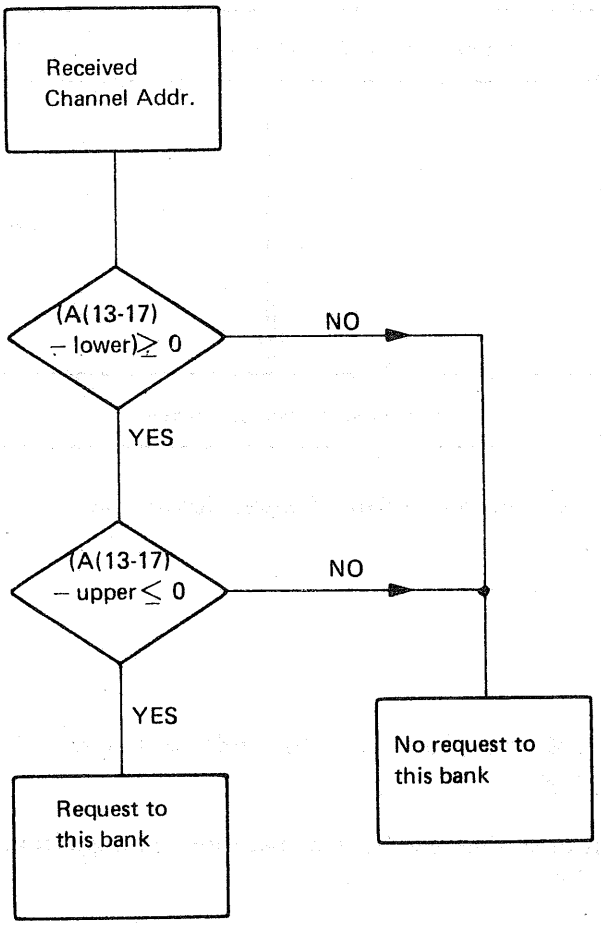
As illustrated in Figure II.7.4, each bank address starts from address 0.

When a bank is selected a channel address to bank address convention must have taken place.

The bank address is formed by subtracting the lower limit from the channel address. Since a bank is limited to 64K, the difference should never be greater than this figure and accordingly the bank address will be represented by 16 address bits.

Figure II.7.6 will help illustrate.

Note: Due to the above mentioned method of forming a bank address, a bank address range *cannot* cross the 64K boundaries.



A(13-17): Upper channel address
Lower: Lower limit switch setting (bits 13-17)
Upper: Upper limit switch setting (bits 13-17)

Figure II.7.5: *Bank Selection*

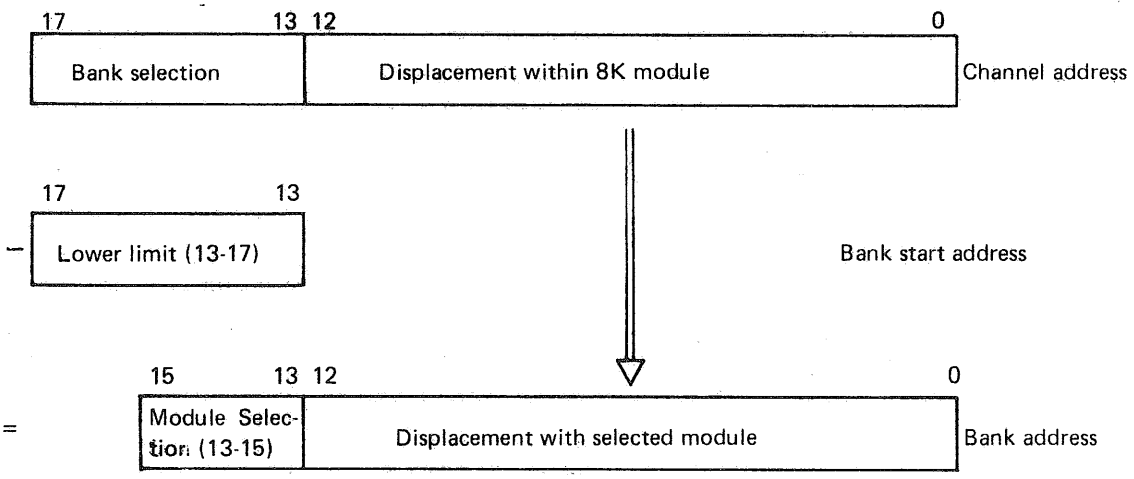


Figure II.7.6: *Channel to Bank Address Conversion*

II.7.4.3 *Example*

During a DMA transfer from a disk (refer to Figure II.7.1) an address of 375034_8 is issued.

Let us assume that we have a multiport configuration as indicated in Figure II.7.4.

The bank selection test, as indicated in Figure II.7.5 will be performed simultaneously for all four banks.

The limit switches for the banks will be set as follows:

	17	16	15	14	13		
64K	0	1	0	0	0	Lower limit	Bank 0
	0	1	0	1	1	Upper limit	
96K	0	1	1	0	0	Lower limit	Bank 1
	0	1	1	1	1	Upper limit	
128K	1	0	0	0	0	Lower limit	Bank 2
	1	0	0	0	1	Upper limit	
144K	1	0	0	1	0	Lower limit	Bank 3
	1	0	0	1	1	Upper limit	

Extracting the upper 5 bits of 375034_8 will give 01111_2 which is equal to upper limit of bank 1.

The bank 1 address will then be:

	011.11	1.101.000.011.100	Channel address
–	011.00		Bank start address
=	11	1.101.000.011.100	Bank address

The upper bits, 11_2, indicate that the fourth (last) module within the bank will be accessed.

As indicated, the banks are defined in the ports. It is therefore possible that the banks may be differently defined as seen from the different ports. This will normally introduce great complexity in the system.

However, it might be desireable that a given channel has limited access in memory. This can be accomplished by leaving some banks undefined (lower limit > upper limit) as seen from that particular port.

II.7.5 *INTERLEAVE*

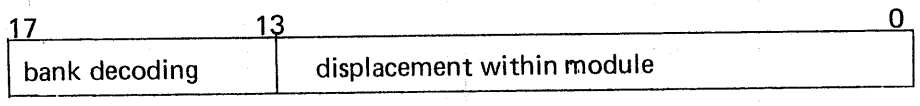
To increase the memory band width an interleave technique can be used. In order to use this method, two or more independent memory banks must be available. Consecutive addresses from a channel will access through all banks in a circular manner. This is accomplished by using the lower address bits for bank selection.

II.7.5.1 *Two-way Interleave*

A two-way interleave means that two banks will alternately be accessed as consecutive addresses are issued.

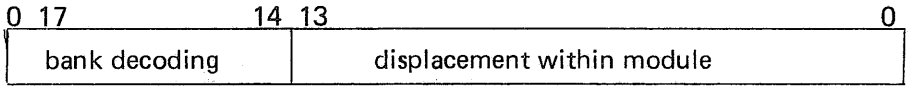
As described earlier, a channel address will be used as follows:

Normal:



In the address cable, wire holding bit 0 will be moved to position 17 and all the other wires moved one place to the right. The interleaved address, also referred to as the shifted address will look like this.

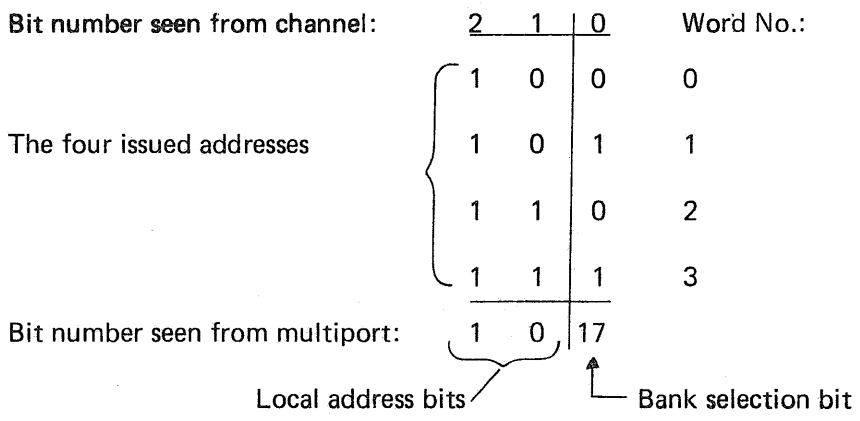
Shifted Address:



In this case, the difference in start address for the two banks in question is 128K. However, if bit 16 had been replaced by bit 0, the difference in start address would have been 64K.

II.7.5.2 *Example*

Four words should be stored in memory in address 4 through 7. (100_2 – 111_2.) The addresses will then be decoded as depicted in Figure II.7.7.



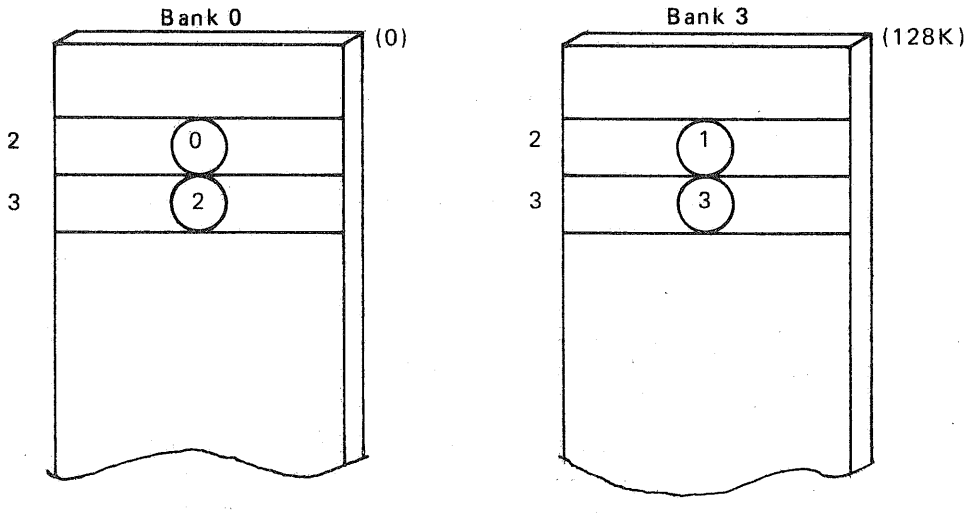


Figure II.7.7: *Interleave Illustration*

II.7.6 *NORD-10/S TO NORD-50 MEMORY COMMUNICATION*

Using multiport memory as a common storage, a NORD-10/S to NORD-50 Memory Communication can be established.

However, the NORD-50 is a 32 bits computer and likes to see a memory system with its own word length. This can be done by giving two banks the same address space as seen from the NORD-50 channel, while the two banks are interleaved as seen from the NORD-10/S.

If we regard the last example again, the CPU has stored four words in memory, two words in bank 0 and two words in bank 2. The first and second word are stored in the same local address in the two banks (refer to Figure II.7.7) and the third and fourth words are stored in the next location in the two banks.

When NORD-50 later reads this information (4 words), it does so with 2 requests only.

NORD-50 sees the two banks as one bank with double word length.

Note: NORD-50 must use the NORD-10/S' shifted start address.

II.7.7 *CONTROL AND INFORMATION FLOW*

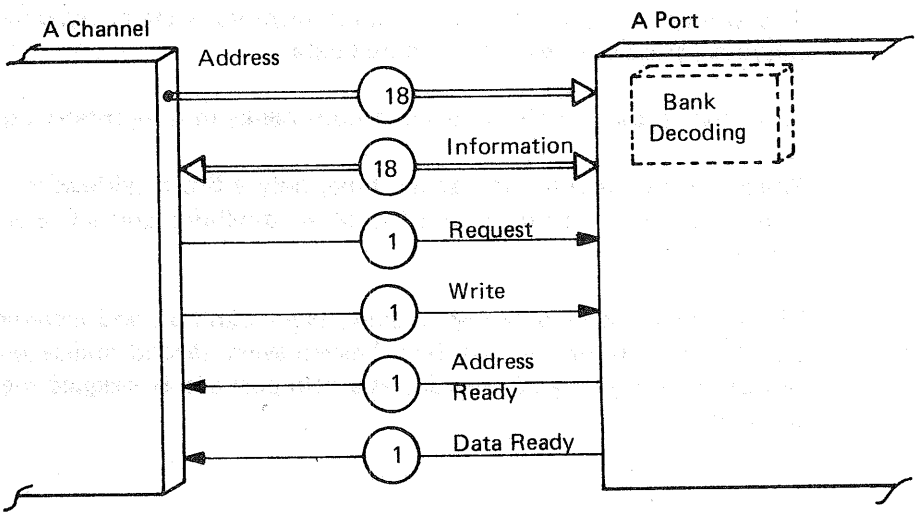


Figure II.7.8: *Control and Information Flow*

II.7.7.1 *Write Operation*

During a write operation the *write* line is activated and an 18 bits *address* is sent along with a *request*. The 18 bits of *information* (16 data + 2 parity bits) must also be present on the bus. From the upper bits of the 18 bits address the bank selection is performed. If the bank is set busy from another port, this request must wait until the previous one is completed.

When the selected bank has accepted the address, the *Address Ready* will also drop indicating that the bank has accepted the data.

II.7.7.2 *Read Operation*

A read operation is indicated by a false *write* line. A *request* is sent along with the *address*. When the selected bank has accepted the address, *Address Ready* is activated and the channel may drop the address. After the access delay, the data will be ready indicated by the *Data Ready* line. Seeing this line being activated, the channel will accept the data.

II.7.8 *REFRESH*

The storage elements in the multiport memory system, dynamic RAM, requires periodic refresh of the stored data.

A refresh is done to all modules in both banks in a multiport card crate.

Since one row is refreshed at the time, only a 6 bits address is required. The refresh logic basically consists of an oscillator and a 6 bits circular counter.

The oscillator gives a refresh request every 32nd μs and increments the counter, i.e., every row will be refreshed every second millisecond. The refresh circuitry may be regarded as a fifth port and is assigned the highest priority.

II.7.9 *BIG MULTIPORT MEMORY SYSTEM*

II.7.9.1 *General*

The Big Multiport Memory System (referred to as BMPM) is a high speed flexible and modular memory system used for two major applications:

1. Increased memory band width to enable for higher total data transfer rates to and from memory.

2. Allow for multiprocessor and/or multi-device communication via a common memory system.

This is accomplished by using a number of independent memory banks accessed by a number of independent memory ports. The system is modular with respect to banks, channels and storage capacity.

Each memory channel has an address range of 2048K words and may be connected from 1 to 10 ports.

Single bit error correction and multiple error detection is standard.

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

II.7.9.2 *The Memory System*

A memory system (refer to Figure 1.1) is a collection of independent *banks,* each covering a definite address space. Every bank is linked together and to the outside world by *Channels.* The entrance for channels is called *Ports.* The Channels in the system are driver from *Sources.*

The Sources are interfaces between the memory system and the unit demanding access. In order to maintain a well-defined intersection the sources are not considered part of the memory system.

In addition to the ordinary channels a *special service channel* exists. All banks are linked to the service channel which is driven from an I/O interface.

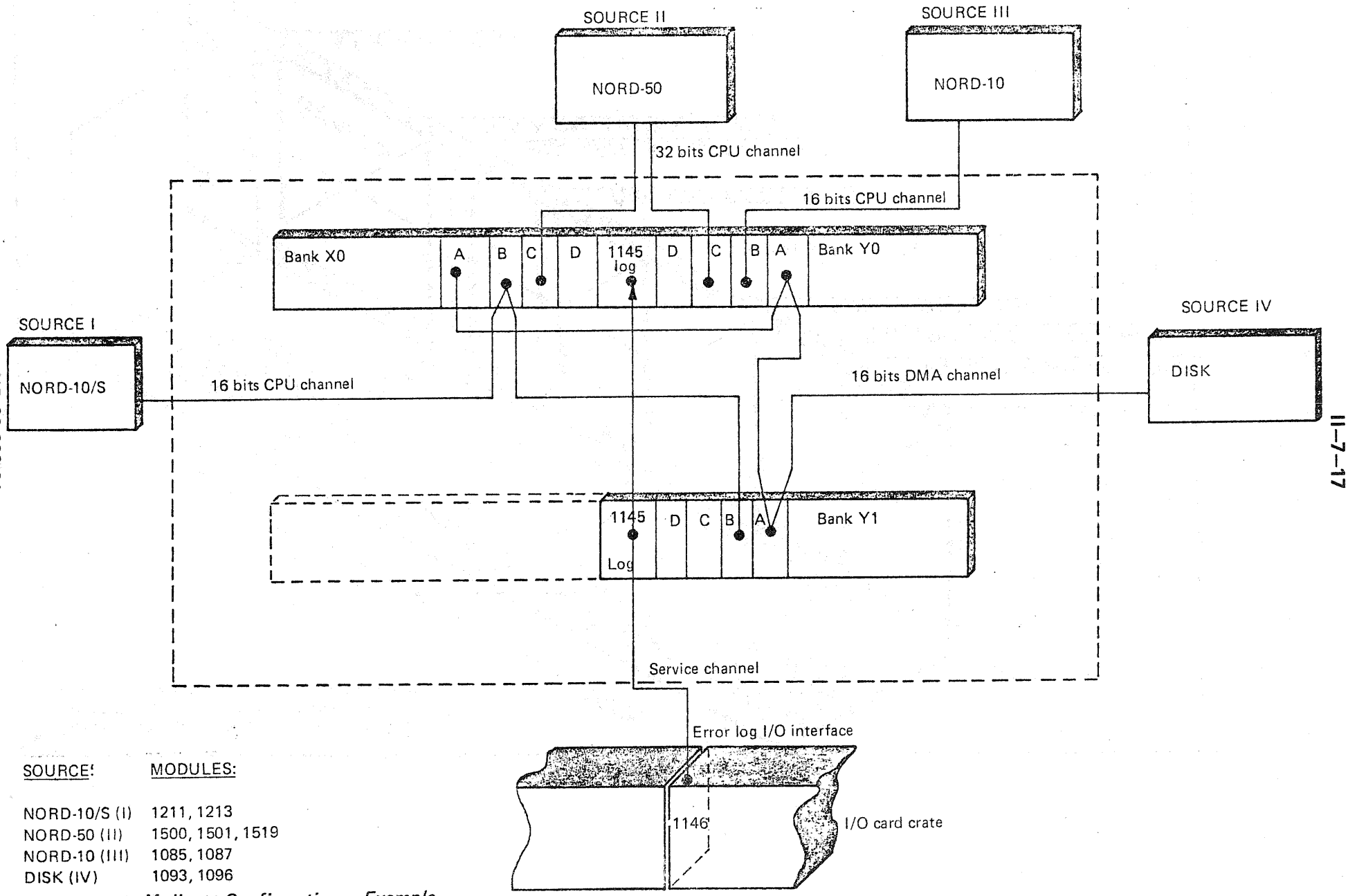
II.7.9.3 *System Parts*

II.7.9.3.1 CRATE

For the following discussion refer to Figure 1.2.

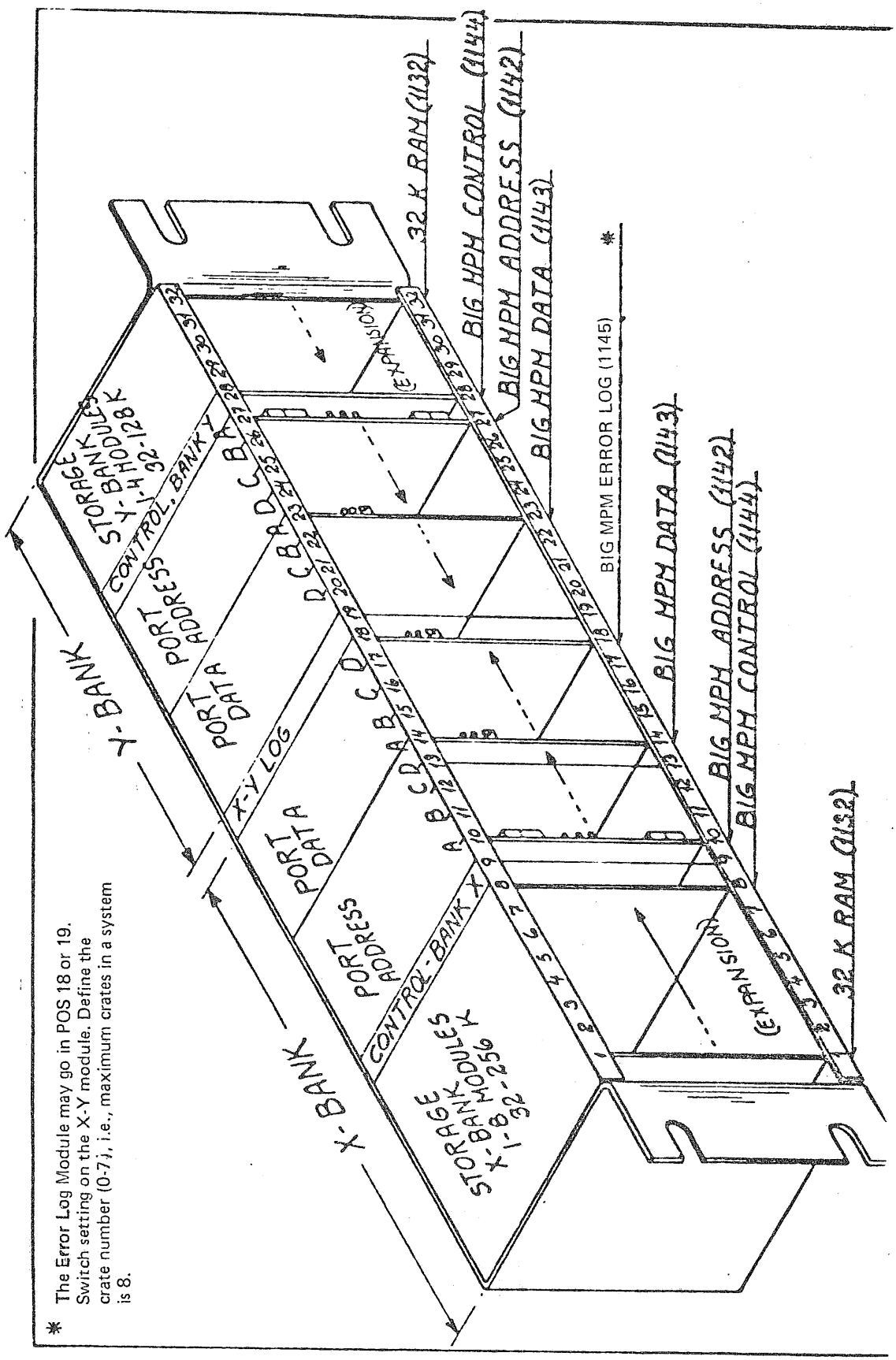
Although the bank is logically the basic building block, the card rate is physically the basic unit, containing one or two banks, an X-bank and a Y-bank. This is done purely for space and wiring reasons. In BMPM there is no logic in common between the two banks as it is on the former model. Yet the service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

A maximum of 8 crates can be used in a Memory system.



SOURCE:	MODULES:
NORD-10/S (I)	1211, 1213
NORD-50 (II)	1500, 1501, 1519
NORD-10 (III)	1085, 1087
DISK (IV)	1093, 1096

Figure II.7.9: *Mulipott Configuration – Example*



* The Error Log Module may go in POS 18 or 19. Switch setting on the X-Y module. Define the crate number (0-7), i.e., maximum crates in a system is 8.

Figure II.7.1U: *Multiport Memory (1 card crate)*

II.7.9.3.2 BANK

A bank is an independent memory unit that can cycle by itself. It also has it's own timing and control logic.

Two banks can be located in one card crate, one X-bank and one Y-bank.

A complete bank consists of the following parts:

* Controller (1144)
* Storage (1132's)
* Ports (1142, 1143)
* Error logic (1145)

II.7.9.3.3 CONTROLLER (1144)

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

The controller solves two major tasks.

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

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

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

The ports and refresh are assigned a fixed priority which is, beginning at the highest priority:

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

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

Note! It is not possible for one high priority port to lock out the lower priority ones. The reason is that the request must toggle on and off.

Even if a channel is capable of absorbing everything it could get from memory, it must withdraw the request for a moment to prepare for the next. At that very moment the lower priority port will be granted access.

However, *two* high priority ports are able to lock the bank for lower priority ports.

Periodic refresh is necessary in MOS memories to avoid decline of information. This is accomplished by regularly accessing all memory cells. These accesses (one each 15 *vs*) may be treated similar to accesses from the ports.

The refresh request circuitry and refresh address counter is located in the controller (1144).

II.7.9.3.4 STORAGE (1132's)

An X-bank has room for 8 storage modules and a Y-bank has room for 4 storage modules. Each module contains 32K x 21 memory cells (bits). Hence, one X-bank is sufficient to cover all NORD-10/S' primary storage requirements of 256K words.

Internally in the bank the address range is always from 0 in 32K increments up to 256K. Hence, an address transformation is normally required at each port. (Refer to Section 1.4 regarding addressing.)

The 18 address bits required for a 256K address range are used as shown in Figure 1.3.

The *memory integrated circuits (MIC)* contain *16K x 1 bit*. Hence, two MIC's are needed per data bit on a 32K module. Since storage of 21 data bits are required each storage module contains 42 MIC's. The 5 extra bits are used by the control code attached to the 16 data bits. (Error checking and correction is described in Section 1.5.)

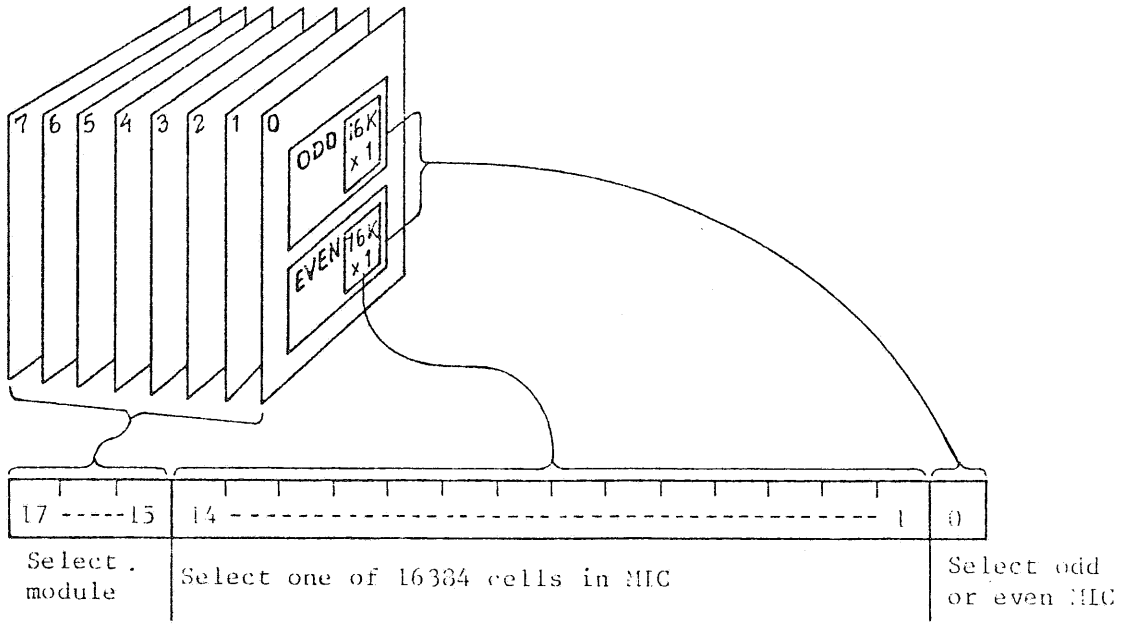


Figure II.7.11: *BMPM Internal Bank Address*

II.7.9.3.5 PORT (1142, 1143)

A port is the entrance for a channel to a bank. A bank holds from 1 to 4 ports (A - D). A port consists of receivers for addresses and transmitters and receivers for data and control lines.

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

A port consists of a data (1143) and an address (1142) module. A port carries out two major functions:

1. Select a bank for the source and convert the channel address to a local bank address (1142).

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

Each port defines the address range for a bank (1142). (Refer to Section 1.4.) Normally, the address range is the same as seen from the different ports, but for special purposes they will be different. One such example is the NORD-10/NORD-50 communication using BMPM as a common memory.

II.7.9.3.6 CHANNEL

In the memory system a Channel is physically a pair of cables. One cable carries address and request signals while the other carries *data* and *ready* signals. With the word channel we will normally mean a *16 bit data* channel. It is, however, possible to define a *32 bit* channel consisting of one address cable and two data cables. Since all ports are 16 bits wide, two ports are required in that case. A 32 bits channel is illustrated in Figure 1.1.

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

It is normally possible to connect several banks to one channel.

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

Note! The two banks in the same crate have their ports daisy chained internally. Figure 1.4 shows signals appearing on a channel.

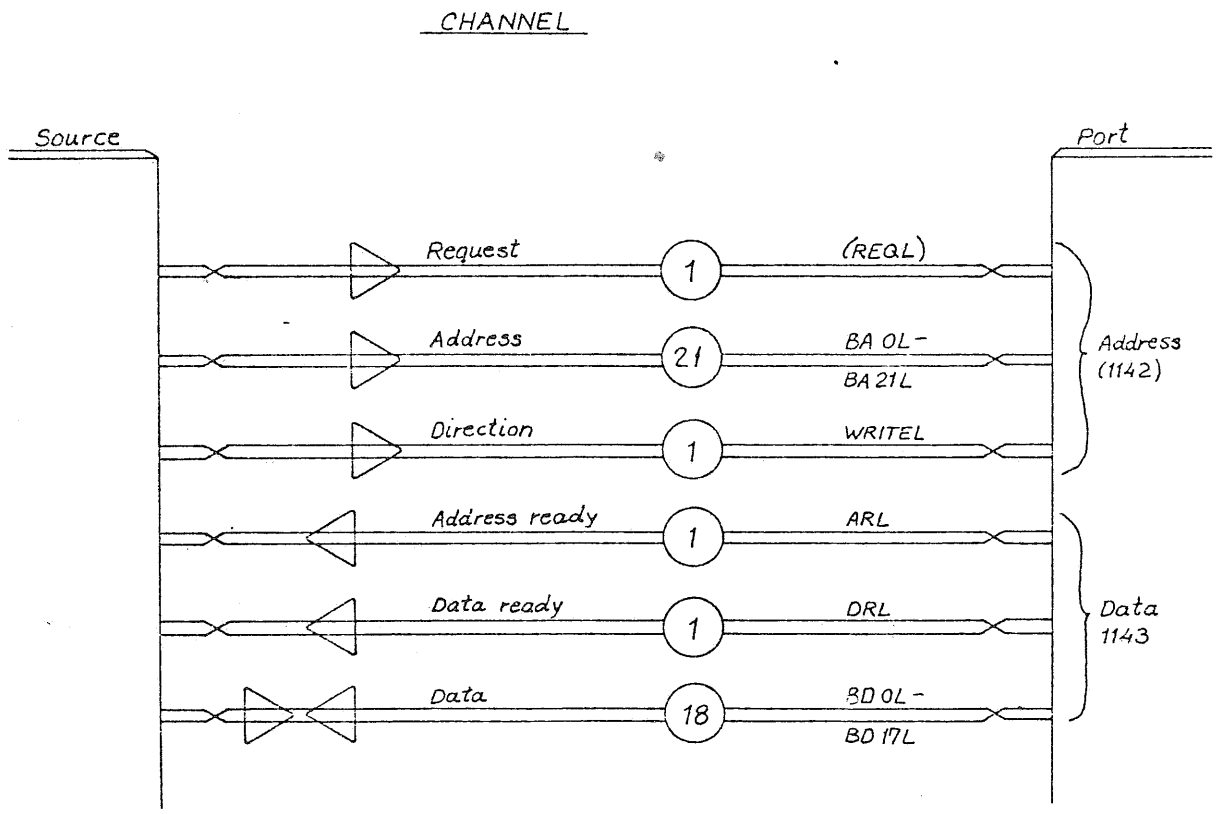


Figure II.7.12: *Channel Signals*

Note 1: A 32 bits channel consists of one address cable and two data cables.

Note 2: Up to 10 ports may be chained to one source. The cables must be terminated in the last port.

Note 3: Transmitters and receivers in the ports meet the RS-422 specifications. (Refer to Appendix E.)

II.7.9.3.7 SOURCE

A Source can be any kind of electronic unit which is capable of communicating with BMPM over a channel. Examples of sources can be: NORD-10, NORD-10/S, NORD-50 or DMA interfaces. The sources must conform to the channel specifications (refer also to Figure 1.1).

II.7.9.3.8 SERVICE CHANNEL

The service channel consists of an error log module (1145) located in the BMPM, an error log I/O interface module (1146) located in the NORD-10/S input/output system and a cable connecting those two modules together. BMPM will thus be regarded as an I/O device as seen from NORD-10/S.

The error log module (1145) is logically divided into two parts, one serves the X-bank and the other the Y-bank. Each part contains an error log memory. All errors (single or multiple) occurring in the bank and detected by the ports (1143's), will be recorded in this special memory. The error log memory is organized as a 512 x 1 bit memory where the address is a pointer to the failing memory integrated circuit (MIC).

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

The service channel can perform the following four functions:

1. (Action: 00), Refer to Figure 1.5.

   Scan the error log memory for a failing bit. The error log memory address where the failing bit is found (pointer to the failing MIC) is transferred to a one-word scan register located in the error log I/O interface (1146).

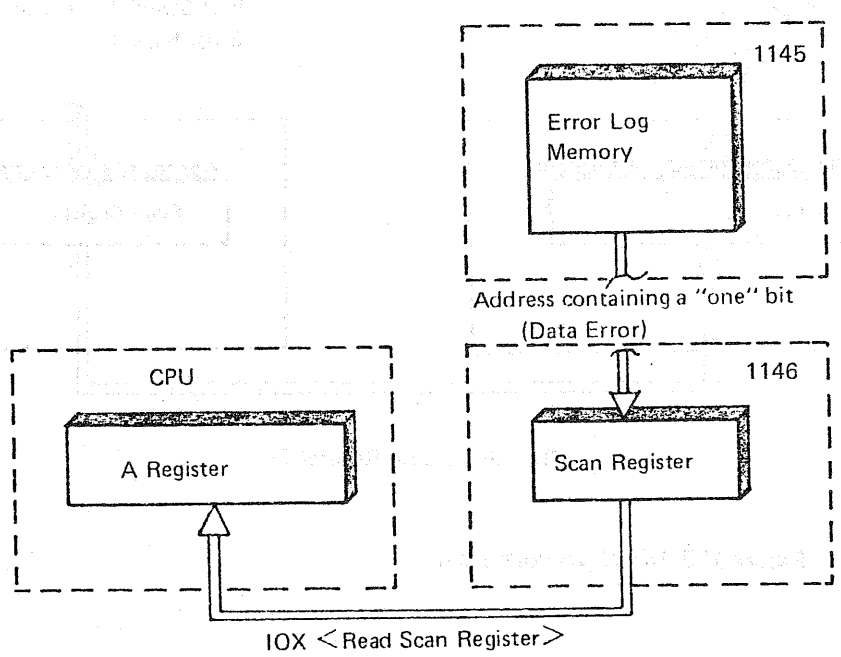


Figure II.7.13: *Scan Error Log*

2. (Action: 01), Refer to Figure 1.6.

   Scan the present ports for bank address range setting. This operation will also give the information whether the ports have detected an error or not.

Steps 1 and 2 are feed-back information from the BMPM and will be written into the scan register located in the error log I/O interface (1146). This will be transferred to the A register of the CPU by execution of an IOX <Read Scan Register> (IOX 750).

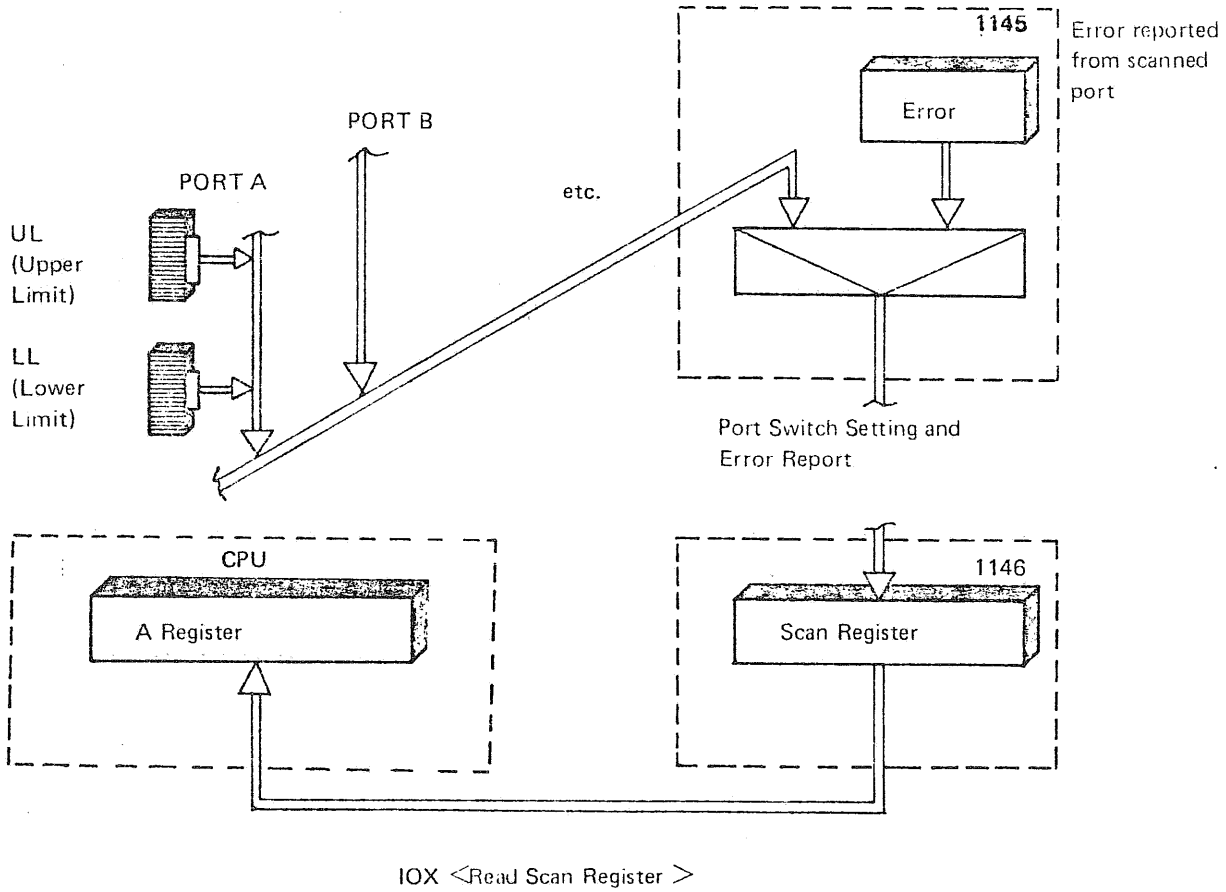


Figure II:7.14: *Scan Port Status*

3. (Action: 10), Refer to Figure 1.7.

   Set the content of the error correction control register (ECC) located in the data module (1143) of the ports.

4. (Action: 11), Refer to Figure 1.8.

   Execute test access. A read or write operation can be simulated without a channel connected.

Steps 3 and 4 are control information for the BMPM. The control information is transferred from the A register in the CPU to the command register located in the error log I/O interface (1146) by executing an IOX <Load Command Register> (IOX 751). From this register the control information is sent to BMPM via the error log module (1145).

Information regarding detailed description of programming specifications and word formats is given in Chapter 3.

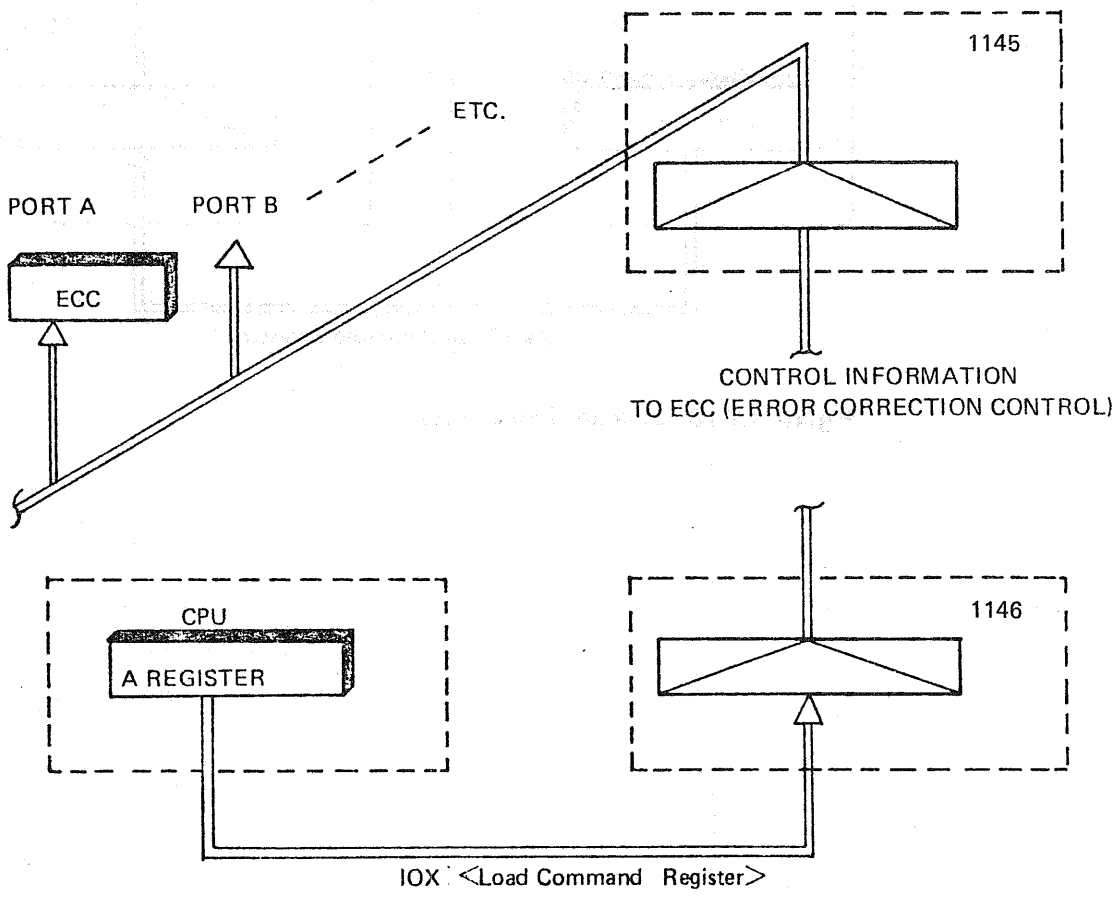


Figure II.7.15: *Write ECC Register*

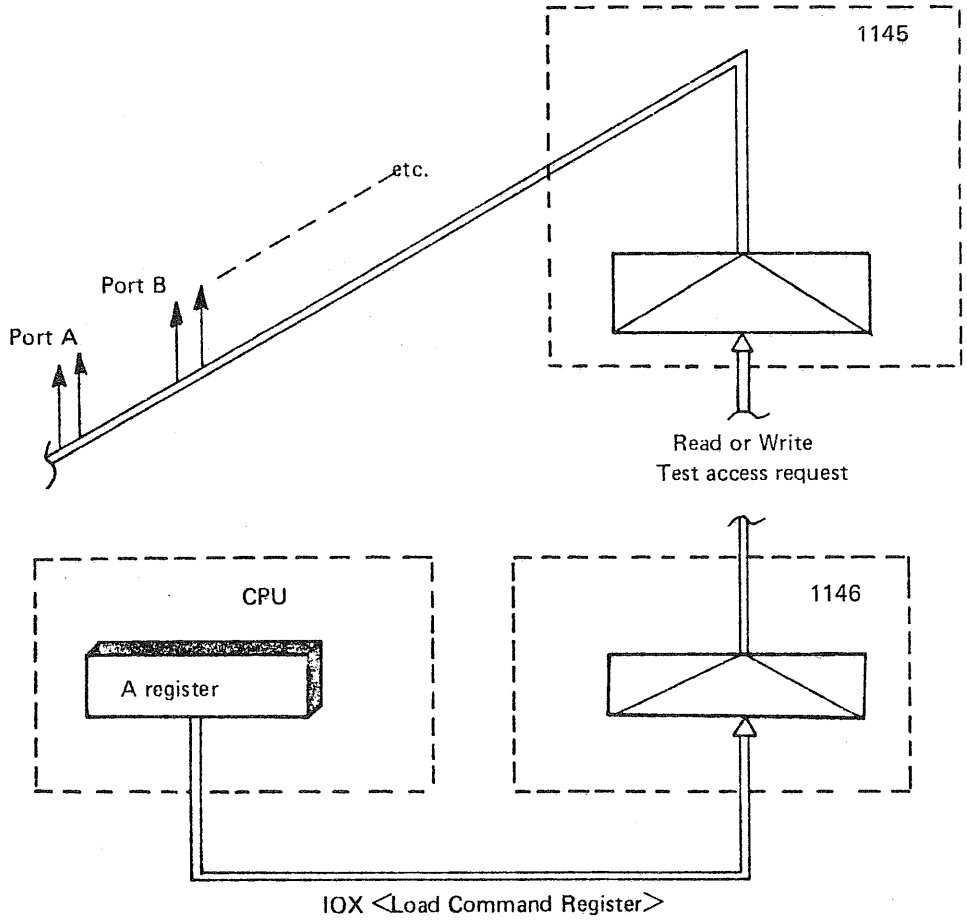


Figure II.7.16: ***Execute Test Access***

II.7.9.4 *Addressing*

II.7.9.4.1 ADDRESS CONVERSION

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

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

Straight decoding requires banks of equal size, so to permit banks of different sizes a more flexible method is required.

Each port has therefore a set of limit switches which define the address range the port will respond to.

One pair determines lower limit, LL, and one pair determines upper limit, UL. Each pair holds a two-digit octal number. Refer to the chart in Figure 1.9 for the corresponding address range.

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

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

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

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

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

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

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

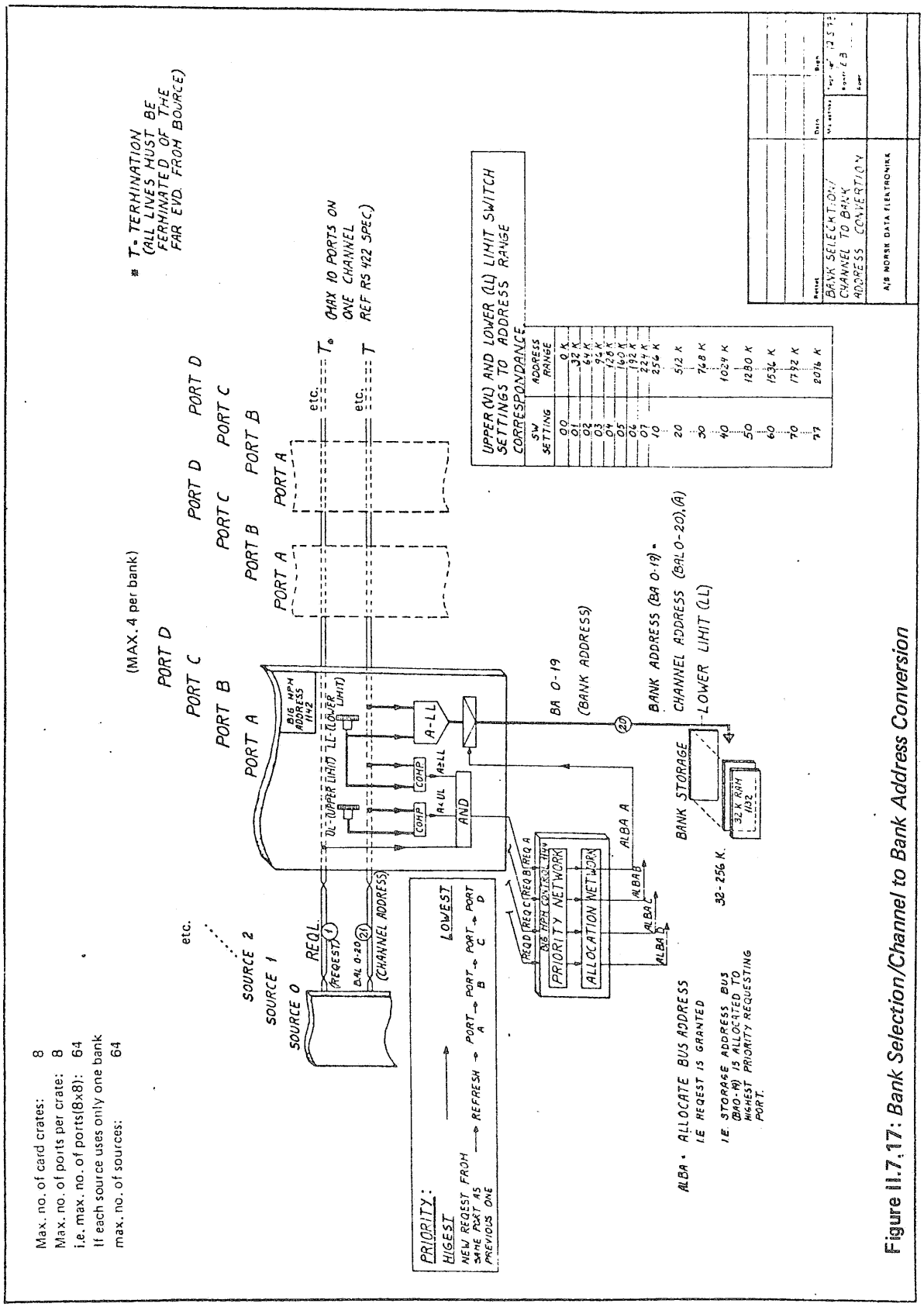


Figure II.7.17: *Bank Selection/Channel to Bank Address Conversion*

II.7.9.4.2 BANK SELECTION

For the following discussion refer to Figure 1.9.

When a source places an address on the bus together with a Request all the ports linked together by the channel will examine the channel address. The port (1142) that finds the channel address within its limits (LL ⩽ A < UL) will forward the request to the controller (1144).

Since the 4 ports associated with a given bank operate completely asynchronously with respect to each other, a priority network is required. The highest priority port will then be allocated to storage for one memory cycle. The bank address = (channel address — lower limit) will then be enabled onto the local bank address bus.

For further information regarding storage to port communication refer to Figure 1.10.

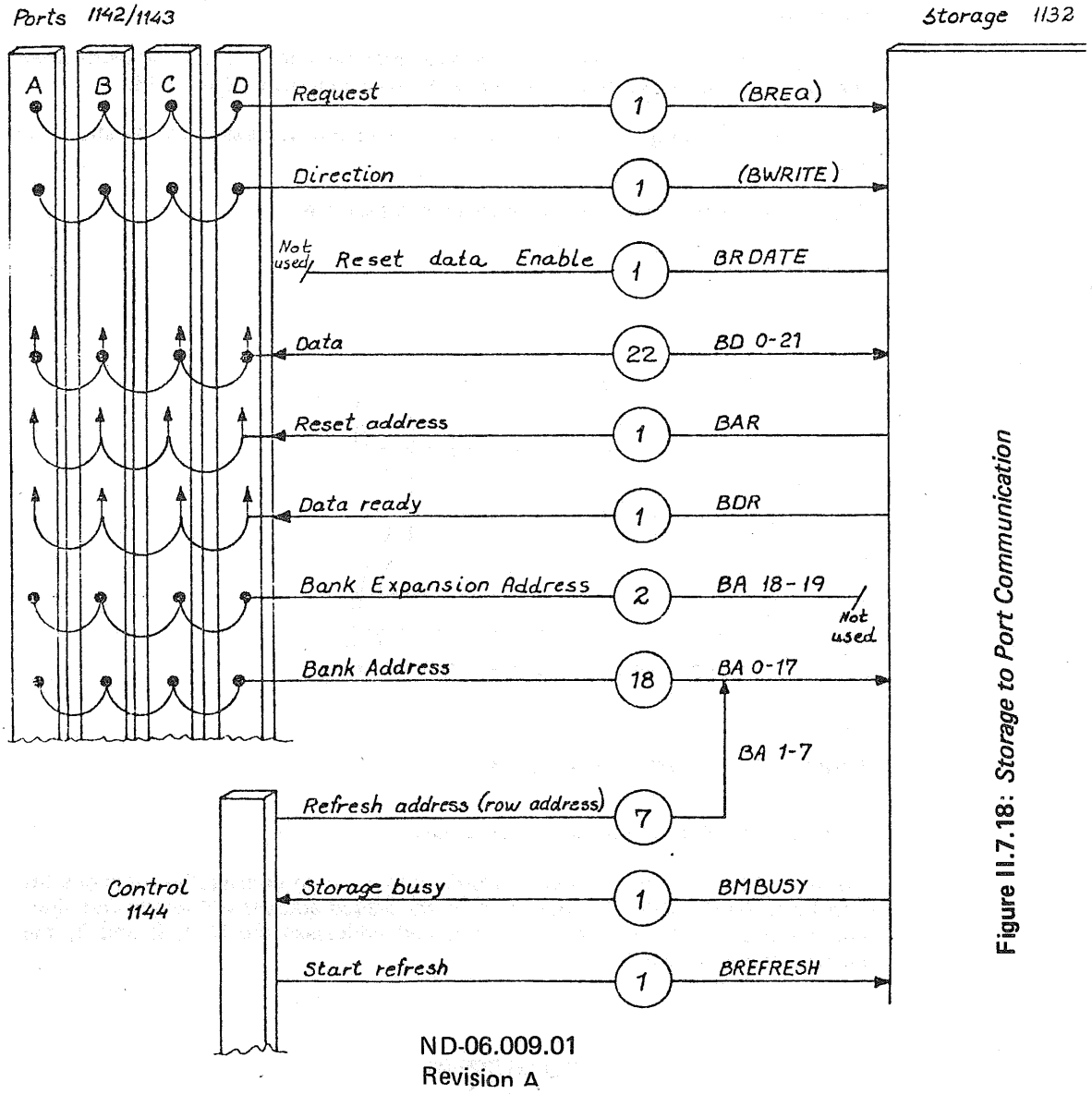


Figure II.7.18: *Storage to Port Communication*

II.7.9.4.3 INTERLEAVE (SHIFTED ADDRESS)

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

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

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

Two consecutive addresses (as seen from program) will therefore lie in different banks. The band width benefit expectation from this shifting is greatly exagerated when memory accesses are not pipelined.

However, interleaving is necessary when 16 bit and 32 bit channels are to communicate.

Two 16 bit words, at consecutive addresses, may be read as *one 32 bit* word when the two 16 bit words *reside in different banks.* (See Figures 1.12 and 1.13.)

The address shifting is performed by modifying the address cable as shown in Appendix C.

The concept may be illustrated in the following example.

Example:

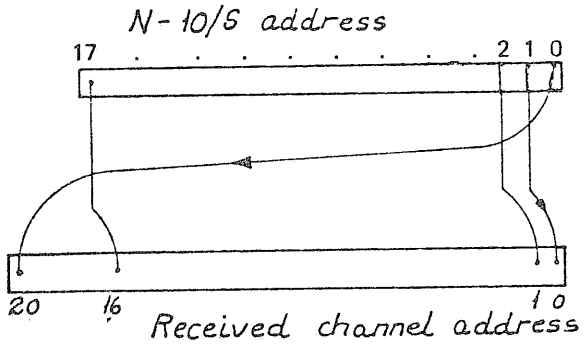


Figure II.7.19: ***Shifting of Address Bits***

Address bit number 20 is used to select a bank.

Assuming NORD-10/S is storing 4 words in memory in consequtive locations (as seen from NORD-10/S) the lowest bit on the issued address will toggle and alternating banks will be selected. If the issued addresses are 0, 1, 2 and 3, the received addresses would be:

Received Address:	Word No.:
0 0 0 0 0 0 0	(1)
4 0 0 0 0 0 0	(2)
0 0 0 0 0 0 1	(3)
4 0 0 0 0 0 1	(4)

etc.

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

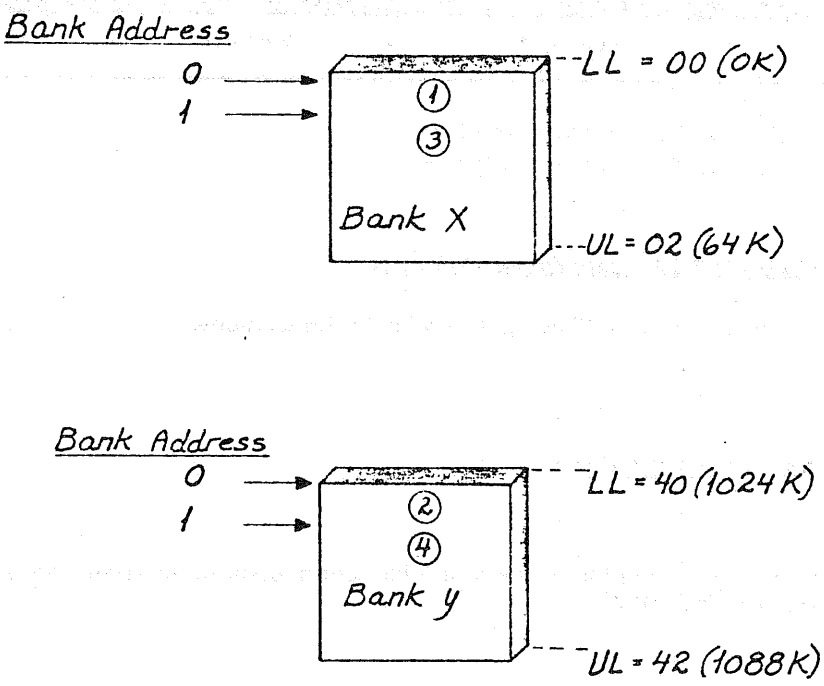


Figure II.7.20: *Banks as seen from NORD-10/S*

When NORD-50 reads those 4 NORD-10/S words, it regards the same two banks as one bank with double word length. The 4 16 bit words will then be read as two 32 bit words.

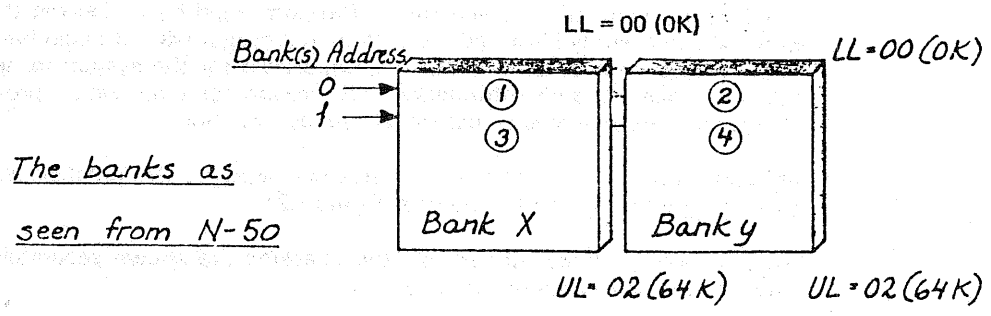


Figure II.7.21: *Banks as seen from NORD-50*

II.7.9.5 *Data*

II.7.9.5.1 FORMAT

As seen from the source (NORD-10/S), the standard BMPM data width is the 16 bits associated with 2 odd parity bits, one for lower, one for upper byte.

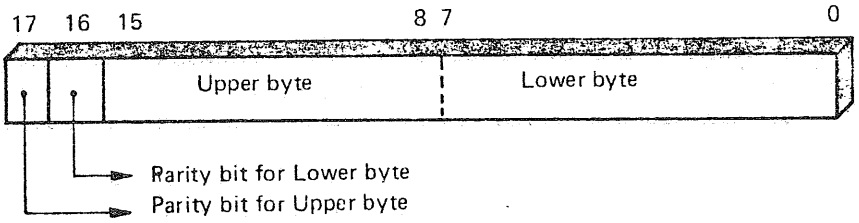


Figure II.7.22: ***Data Channel Format***

This makes the BMPM compatible with the former model.

II.7.9.5.2 DATA PROTECTION

Data is protected by means of a 5 bit control code. Error correction is therefore standard in BMPM.

II.7.9.5.3 GENERAL ABOUT ERROR DETECTION AND CORRECTION

In an error correcting memory 5 parity bits are stored in addition to the normal 16 data bits. These 5 bits are parities formed by the data bits in a unique way. Each port has its private error correction network in order to load the common memory resources as little as possible. The two source-generated parity bits are skipped and 5 new control bits are generated. The port could have checked the parity of write data but hardly informed the system in a proper way. It could have reported "Hey, I have seen an error", but it is impossible for the system to decide what action to take on such information. The reason for error would probably be a faulty cable, which would be detected by reading anyhow.

Although many people find error correction obscure, the practical treatment is quite straight forward and is shown in Figure 1.21.

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

The following 5 parities serve as control bits and are generated at each *write* access.

C0 = 0⊕1⊕2⊕3⊕4⊕6⊕8⊕10
C1 = 0⊕2⊕4⊕7⊕9⊕10⊕12⊕14
C2 = 1⊕2⊕5⊕7⊕11⊕12⊕15
C3 = 3⊕4⊕5⊕6⊕7⊕13⊕14⊕15
C4 = 8⊕9⊕10⊕11⊕12⊕13⊕14⊕15

They are stored together with the 16 data bits in memory.

When data is *read* the parity of the 21 bit word is checked. If no errors have been introduced all parities are *unchanged* which gives the error code 00000 = GOOD. The 5 bits in the error code are called syndrome bits. 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. For example, bit 2 is a part of C0, C1 and C2. If bit 2 has been changed (inverted) since generation C0, C1 and C2 will also be inverted. Hence, the error code 00111 is produced instead of the GOOD code 00000 which means no bit changed. Reference to Figure 1.15 will show that, in fact, 00111 is the error code for bit 2. Since error in bit 1 changes C0 and C2, 00101 is the error code for bit 1, etc.

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

Since 21 bits can produce only 21 single error codes the remaining codes must be caused by *errors in some combination of two or more bits.* Those errors cannot be corrected and are therefore fatal to the system.

A decoder is used to decide which of the 31 possible errors has occurred (see Figure 1.15).

In case it is a single bit error the correspondig bit is inverted by an EXCLUSIVE-OR gate.

In case a *multiple error* is detected a FATAL ERROR signal is generated. This signal *inverts the otherwise correct parity bit 16.*

Accordingly, multiple errors force a parity error which is detected by a normal parity check in the source. Single bit errors are (of course) not reported to the source. However, all errors are written in the error log (1145).

The theory for error correction is also covered in Section II.6 of the manual "NORD-10/S Functional Description (ND-06.009).

	S4	S3	S2	S1	S0	No Error	Single Data Error	Single Error Error	Multiple Errors
0	0	0	0	0	0	Good			
1	0	0	0	0	1			C0	
2	0	0	0	1	0			C1	
3	0	0	0	1	1		E0		
4	0	0	1	0	0			C2	
5	0	0	1	0	1		E1		
6	0	0	1	1	0				ME0
7	0	0	1	1	1		E2		
10	0	1	0	0	0			C3	
11	0	1	0	0	1		E3		
12	0	1	0	1	0				ME1
13	0	1	0	1	1		E4		
14	0	1	1	0	0		E5		
15	0	1	1	0	1		E6		
16	0	1	1	1	0		E7		
17	0	1	1	1	1				ME2
20	1	0	0	0	0			C4	
21	1	0	0	0	1		E8		
22	1	0	0	1	0		E9		
23	1	0	0	1	1		E10		
24	1	0	1	0	0		E11		
25	1	0	1	0	1				ME3
26	1	0	1	1	0		E12		
27	1	0	1	1	1				ME4
30	1	1	0	0	0		E13		
31	1	1	0	0	1				ME5
32	1	1	0	1	0		E14		
33	1	1	0	1	1				ME6
34	1	1	1	0	0		E15		
35	1	1	1	0	1				ME7
36	1	1	1	1	0				ME8
37	1	1	1	1	1				ME9

Figure II.7.23: *Syndrome Decoding*

II.7.9.5.3.1 Write

During a store operation the port (1143) will receive the data in the format as indicated in Figure 1.14.

The two parity bits are not processed in any manner by the port. Based on the 16 data bits a 5 bits control code is generated and sent to memory with the data. This is illustrated in Figure 1.16.

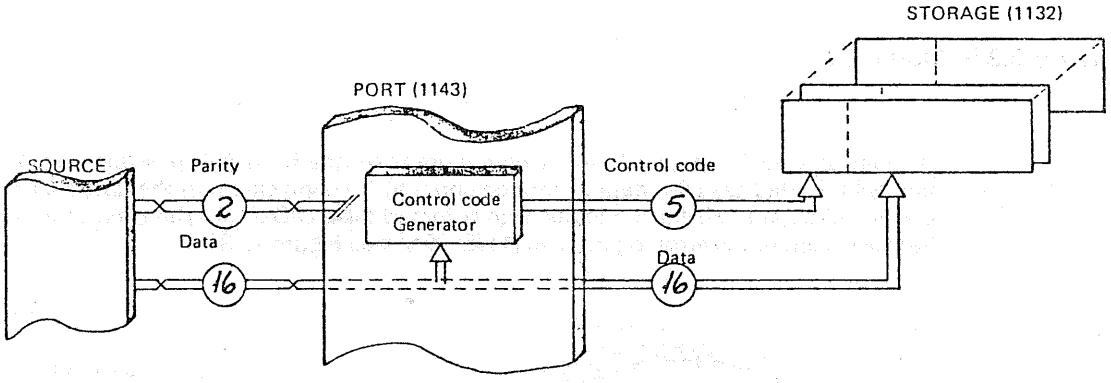


Figure II.7.24: *Data Flow on Write*

Note! Addressing and bank selection is not covered here.

II.7.9.5.3.2 Read

During a read operation the addressed data is received from the selected storage by the port (1143). Working on the 16 data bits, a new control code is generated and compared with the one stored with the data. If they are equal, the data is accepted as Good data.

The difference (exclusive OR) between the "new" and the "old" control code is called the syndrome. If the syndrome is equal to 0, the data is Good. As the control codes are checked, two parity bits are generated in the port (1143) and sent to the source with the data. See Figure 1.17.

II.7.9.5.3.3 Single Data Error

If a single data bit is failing, the 5 bits syndrome (exlusive OR of "new" and "old" control code) will be different from zero ($\neq 0$) where the syndrom (code) is a pointer to the failing bit. The failing bit is corrected (inverted) and the correct parity bits wil be generated and sent to the source with the correct data. The syndrome ($\neq 0$) will be sent to the error log module (1145) to indicate where and what type of error has occurred. See Figure 1.18.

II.7.9.5.3.4 Single Control Code Error

Using the same type of memory for storing the control code, a single bit can also be failing in the code. The syndrome (code) will also here be a pointer to which control bit that was failing. No data need be corrected for this type of error. The synfrome is sent to the error log module (1145).

II.7.9.5.3.5 Multiple Error

A multiple error is a type of error where more than one bit is failing within the 21 bits field. In this case the data cannot be corrected. To indicate that the data is not correct, the parity bit for the lower byte is forced false (BDL 16). The syndrome is here also sent to the error log module (1145). See also Figure 1.19.

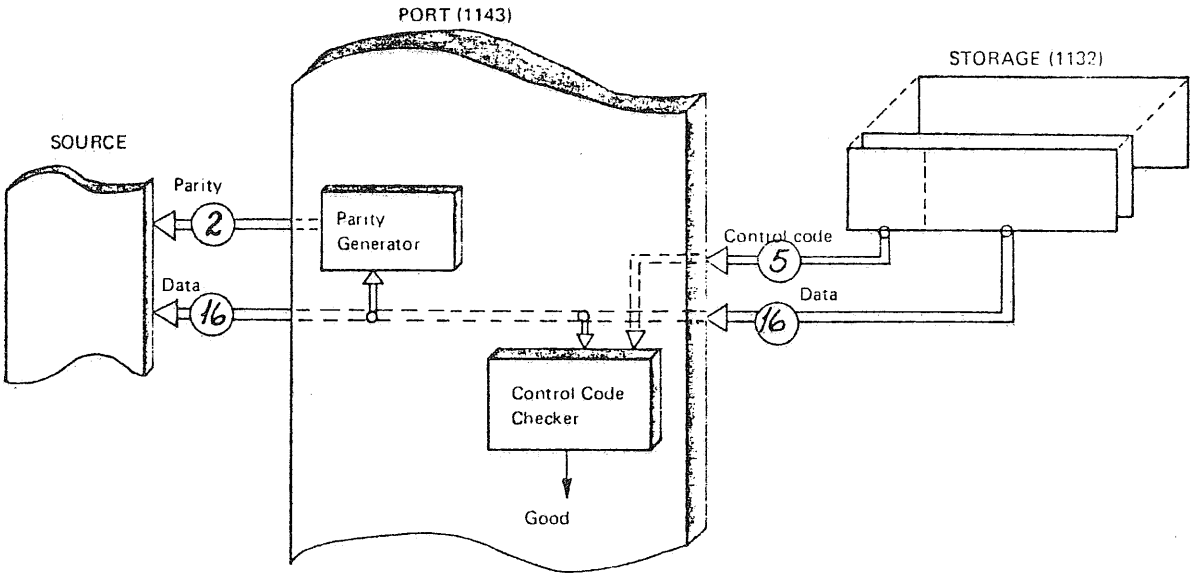


Figure II.7.25: *Reading Data (no error)*

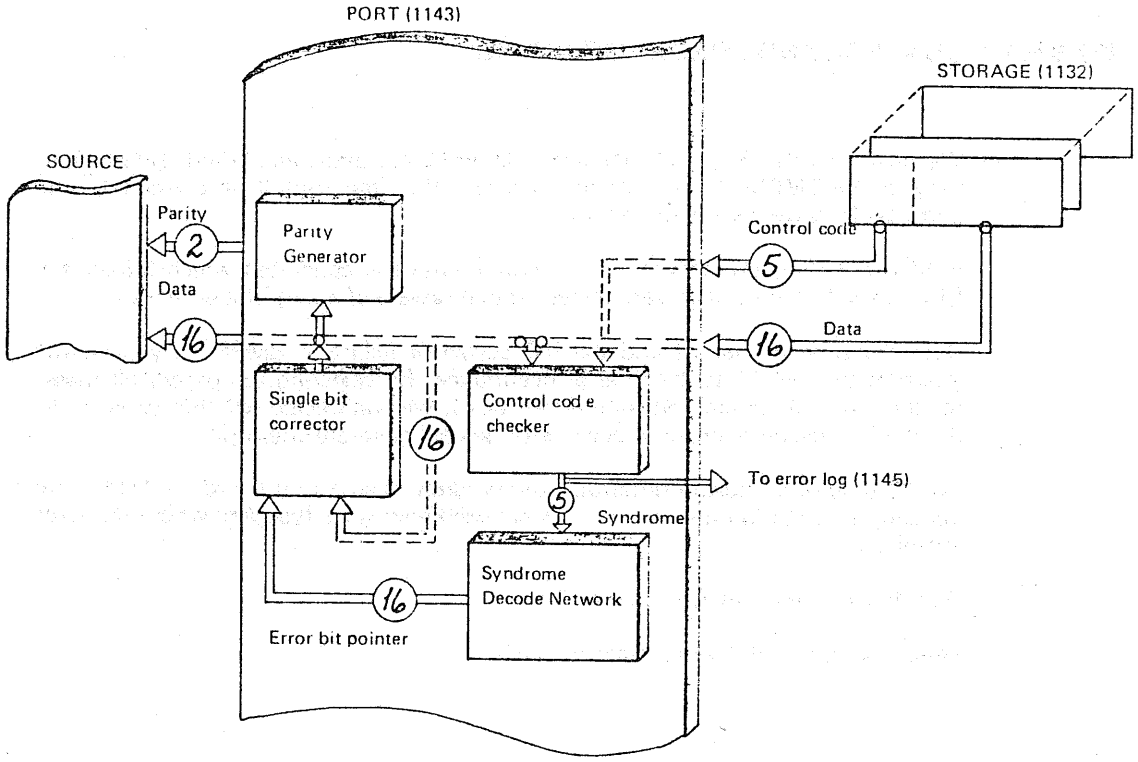
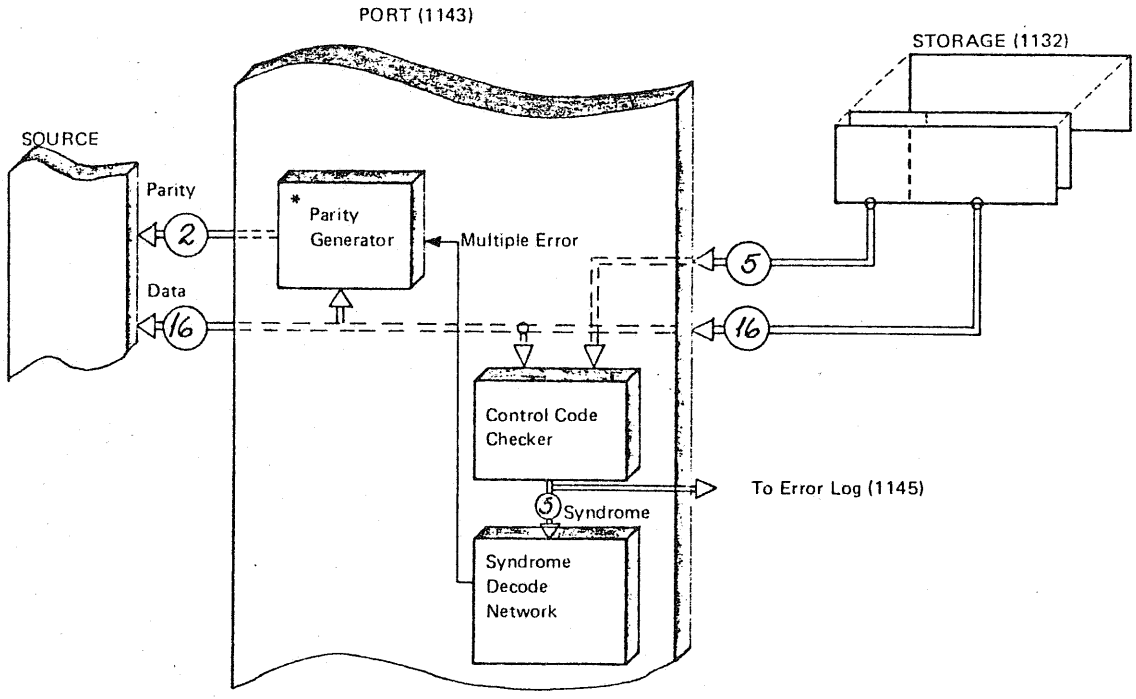


Figure II.7.26: ***Reading Data (Single Error)***



* when multiple error is detected parity bit for lower byte is forced false (BDL 16)

Figure II.7.27: ***Reading Data (Multiple Error)***

II.7.9.5.4 CONTROL AND ERROR REPORTING

We have already discussed that single bit errors are detected and corrected internally in the BMPM. If multiple errors occur, the data cannot be corrected, and parity bit for lower byte is forced false.

If NORD-10/S is the source, an internal interrupt is generated which forces the CPU to level 14. It is, however, of interest to be aware of a single bit error also.

Over the service channel and the I/O system a failure in BMPM may generate interrupt to level 13 at the time of occurrance. By selecting the proper bit mask for the error correction control register (ECC), we can decide whether interrupt is disabled, single bit errors or multiple errors should generate interrupt.

At the time of the failure (interrupt) the syndrom is sent to the error og (1145). By reading the error log memory the system will know what type and where the error occurred.

See detailed description in Chapter 3.

Refer also to Figure 1.20 for further details.

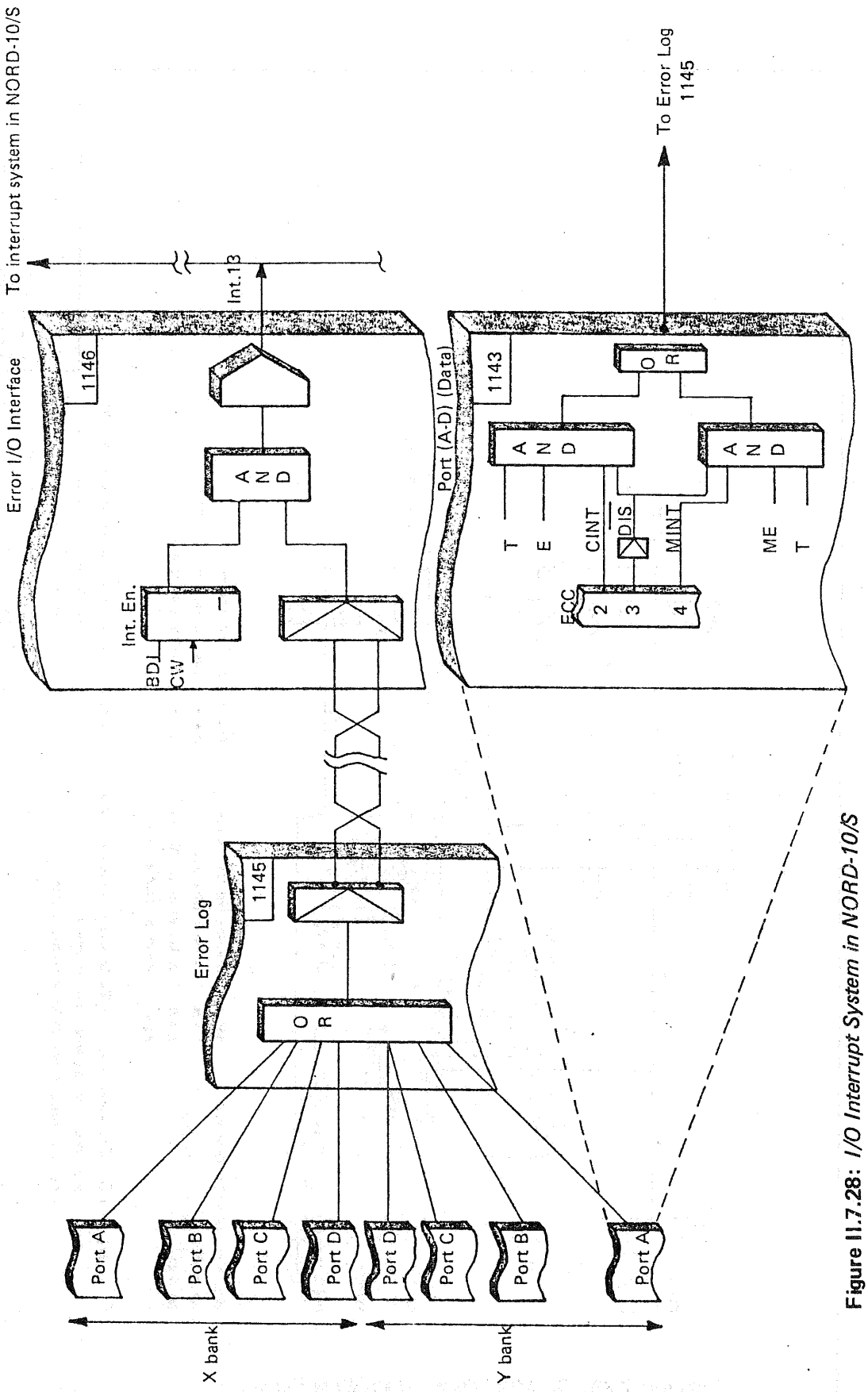


Figure II.7.28: *I/O Interrupt System in NORD-10/S*

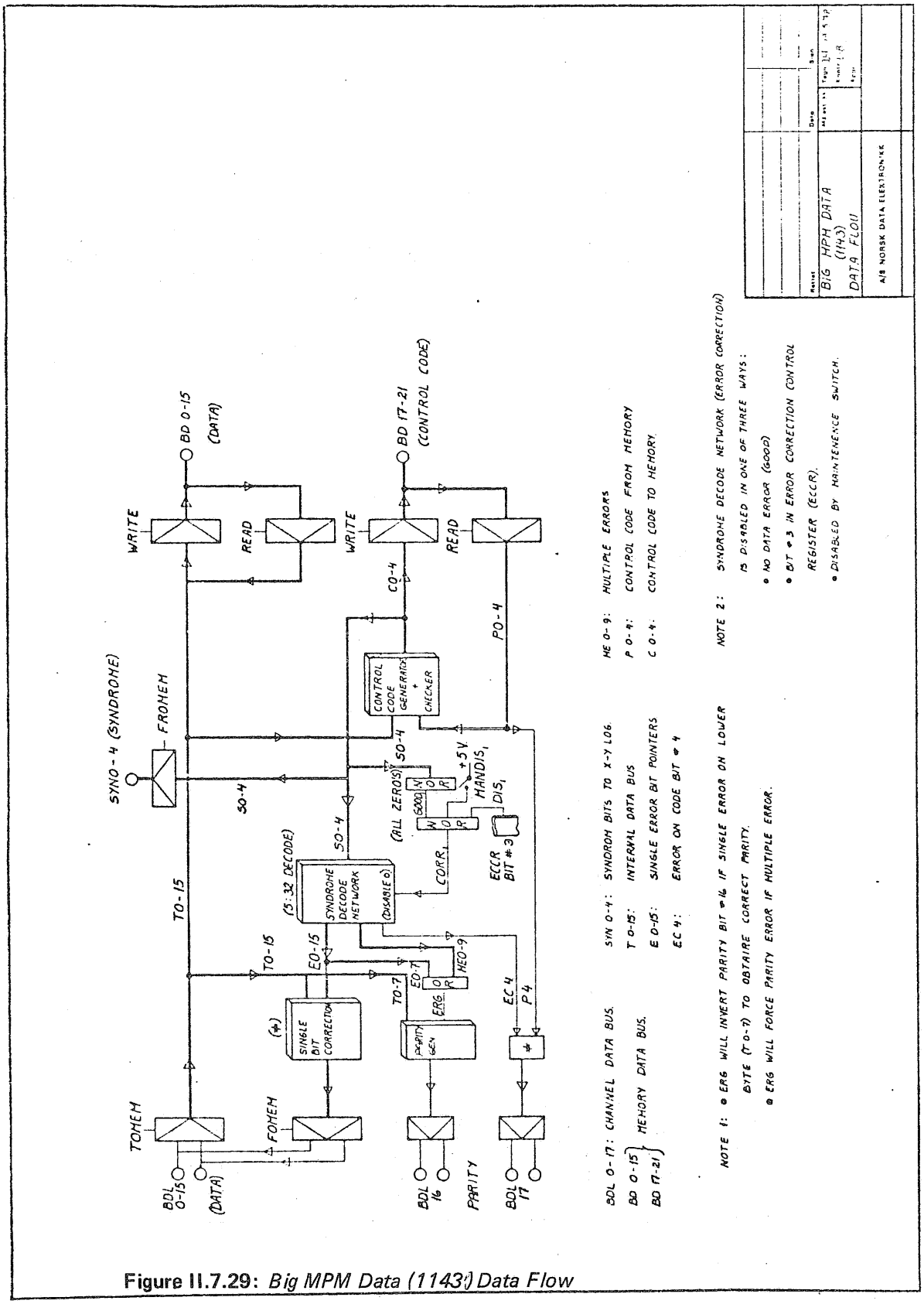


Figure II.7.29: *Big MPM Data (1143) Data Flow*

II.8 MEMORY MANAGEMENT SYSTEM

II.8.1 *GENERAL*

Programs ready for execution are stored on mass storage. When it is time for execution, the program will be transferred from mass storage to primary storage. If there isn't enough primary storage **available** a problem occurs as to which part of the program should be transferred to primary storage and which parts should remain on mass storage.

Another problem occurs when two programs using the same, or almost the same, address space attempt to be executed at the same time.

To solve these problems, a paging system is used. Several advantages, closely connected to the paging system, are gained by using a complete memory management system.

Here are the main points:

1. Virtual Storage

   For each programmer, a virtual storage of 64K 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 as to whether there is enough physical address space in storage when the program is to be placed, or whether other programs are using that part of storage or not.

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

   In NORD-10/S up to 256 K words of physical storage may be used, therefore, a 16 bits virtual address will be translated into an 18 bits physical address. Refer to Figure II.8.1.

2. Dynamic Allocation

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

3. Dynamic Relocation

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

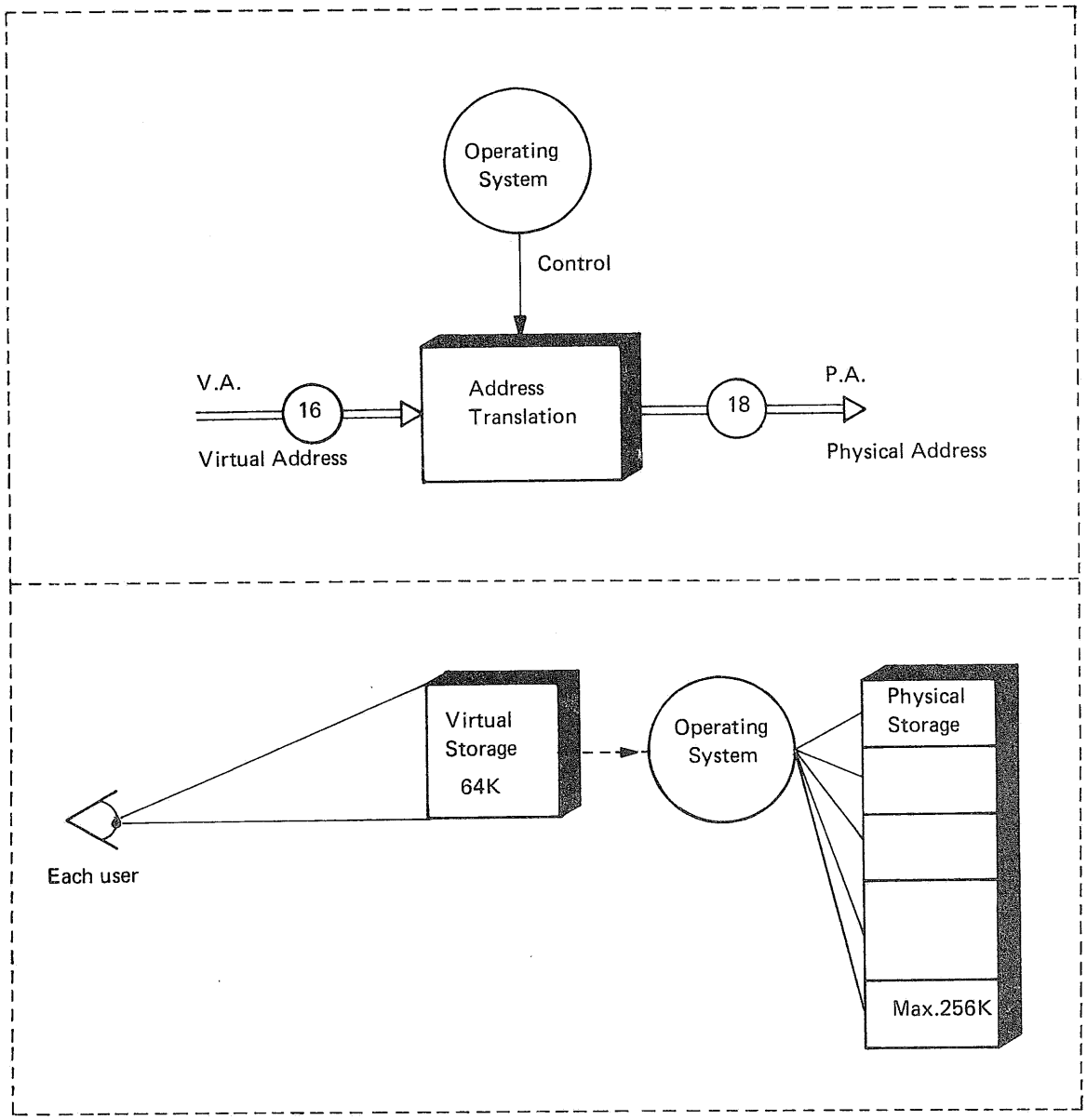


Figure II.8.1: *Virtual to Physical Address Translation*

4. Dynamic 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.

5. 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.

6. More Parallelism

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

Some drawbacks must also be accepted:

1. Increased Memory Access Time

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

2. Increased Execution Time

   Data transport to and from mass storage is rather slow compared to the speed of the processor. A longer execution time will therefore be an impact to a given program. To reduce this affect, more well-structured programs are desired.

3. 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.

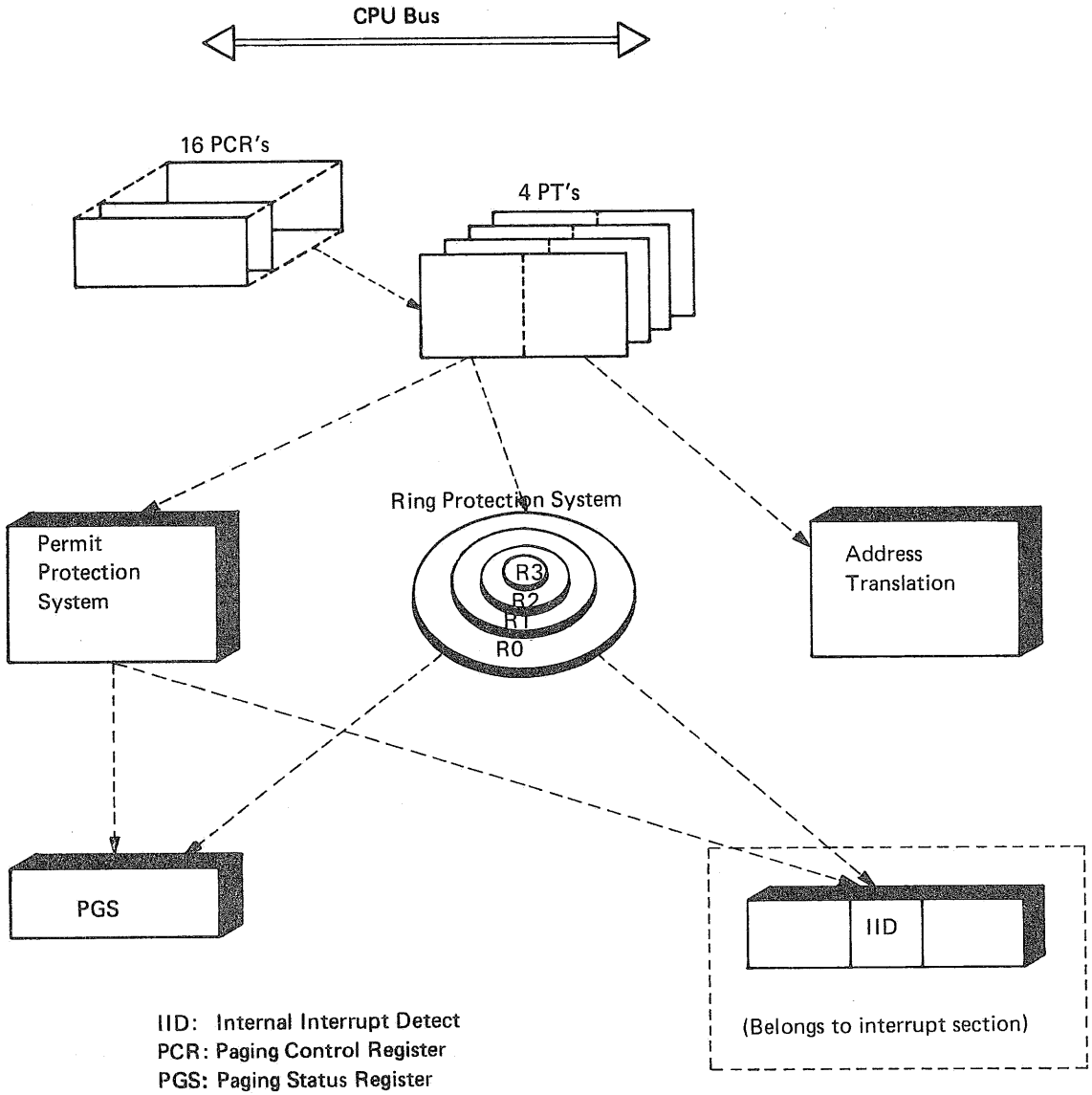
II.8.2 *REALIZATION*

The Memory Management implemented in NORD-10/S has two main purposes:

- Paging (Virtual Storage Implementation)
- Dynamic Memory Protection

The major building blocks in these systems are:

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



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

Figure II.8.2: *Memory Management Building Blocks*

II.8.3 *ADDRESS TRANSLATION*

Figure II.8.3 shows how the virtual to physical address translation is performed. To do this address translation, a separate high speed memory is used to maintain processor speed.

APT:	Alternative Page Table	$0 \leqslant APT \leqslant 3$
DIP:	Displacement within Page	$0 \leqslant DIP \leqslant 1023$
PL:	Program Level	$0 \leqslant PL \leqslant 15$
PM:	Permit Flags	
PPN:	Physical Page Number	$0 \leqslant PPN \leqslant 255$
PT:	Page Table	$0 \leqslant PT \leqslant 3$
PTM:	Page Table Mode (status bit 0)	
PTS:	Page Table Select flag	
R:	Ring	
VPN:	Virtual Page Number	$0 \leqslant VPN \leqslant 63$

NORD-10/S uses 1K (1024) words page. This implies that in order to map 64K words of virtual address space, a 64 (100_8) words Page Table (PT) is required.

For this discussion, disregard the fact that there are 4 PTs and assume that only one exists.

To address any location within a 1K address space (1 page), 10 address bits are required. These bits are the displacement with a page (DIP) and are transferred directly to the physical address bus (MR0-9). The most significant part of the virtual address (bits 10-15) are used as an address selecting one of 64 locations in PT. This address is referred to as Virtual Page Number (VPN). The 16 bits content of the PT is read. The upper 8 bits are used for protection, and are discussed later. The lower bits are transferred onto the physical address bus (MR10-17). These 8 bits are referred to as Physical Page Number (PPN). PPN will take a value between 0-255. That means PPN will select one out of 256 pages. Physical memory can thus be extended up to 256K words.

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

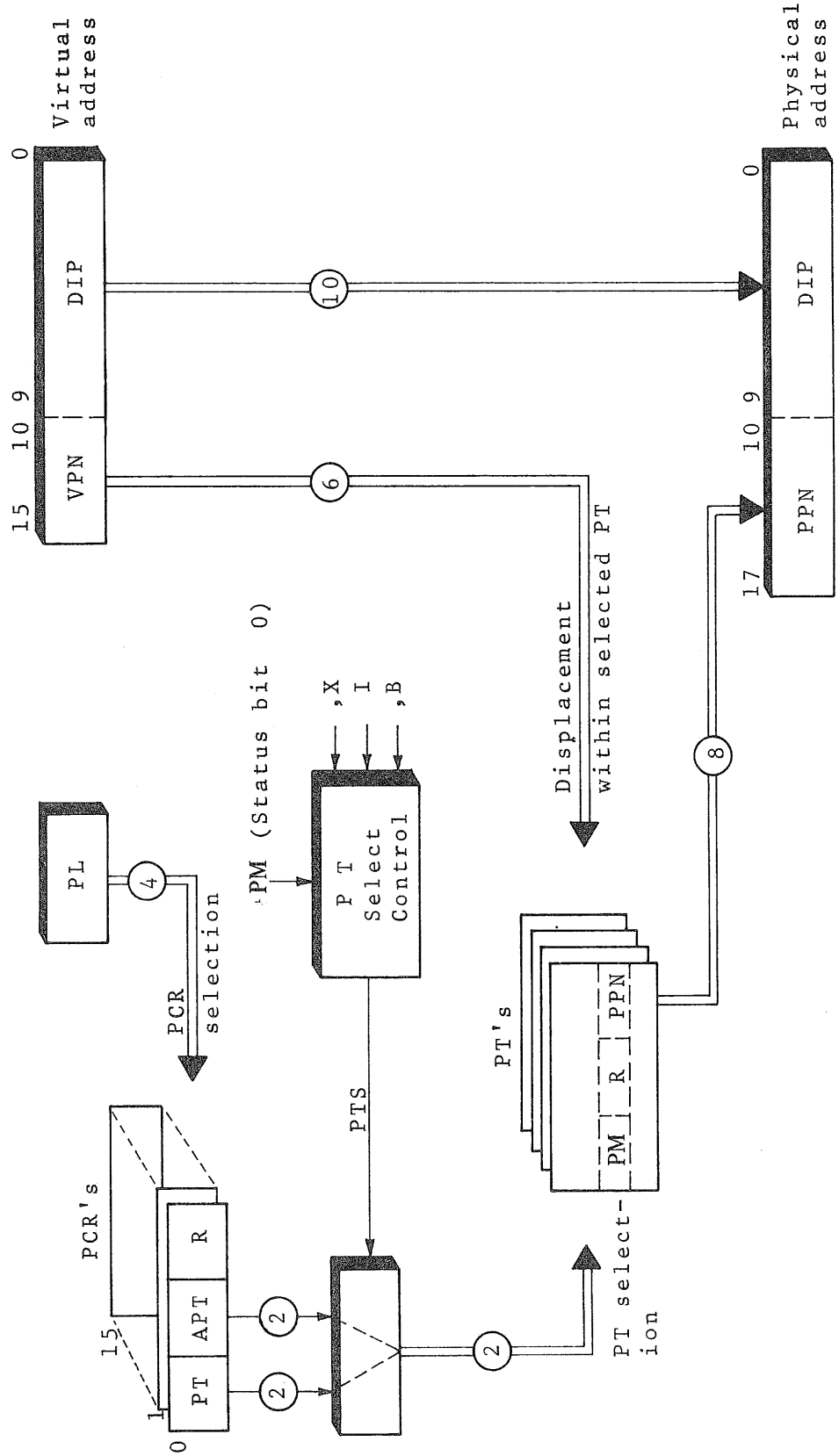


Figure II.8.3: *Virtual to Physical Address Mapping*

Example:

Assume that a user has a 3K program. The start address is 40000_8, i.e., the address space is $40000_8 - 45777_8$. Refer to Figure II.8.4.

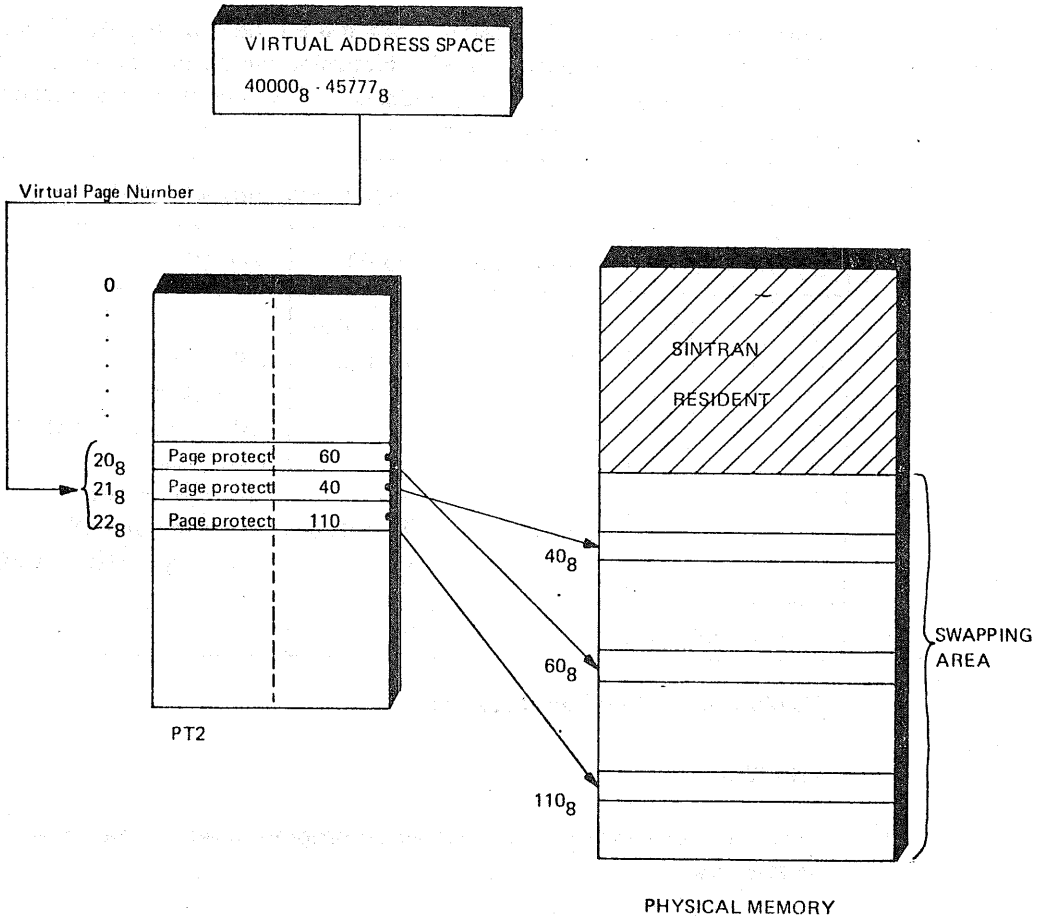


Figure II.8.4: *Virtual to Physical Address Conversion*

From Figure II.8.4, it can be seen that the logical or virtual address space of $40000_8 - 45000_8$ always will use the virtual or logical page numbers $20_8 - 22_8$. However, where in physical memory the 3 pages will be located, is controlled by the physical page number.

II.8.4 *PAGE TABLE SELECTION*

As depicted in Figure II.8.3, NORD-10/S employs 4 Page Tables. Which one is to be used is selected by the Paging Control register on the current program level.

This information is either taken from the PT field or the APT field. The PTS (Page Table Select) flag will determine which one is to be selected. Table II.8.1 shows the correspondance. The paging control registers are under control of the operating system.

Addressing Mode				Address Mapping with PM = 1	
,X	I	,B	Mnemonic	Via PT	Via APT
0	0	0		(P) + disp.	–
0	1	0	I	(P) + disp.	((P) + disp.)
0	0	1	,B	–	(B) + disp.
0	1	1	,B I	–	(B) + disp.; ((B) + disp.)
1	0	0	,X	–	(X) + disp.
1	0	1	,B ,X	–	(B) + (X) + disp.
1	1	0	I ,X	(P) + disp.	((P) + disp.) + (X)
1	1	1	,B I ,X	–	(B) + disp.; ((B) + disp.) + (X)
Instruction fetch				(P)	–

Table II.8.1: *Page Table Selection*

Example:

Assume a program is to be started on program level 1. That means, PCR 1 is selected.

Further, assume that the virtual address space for the program is 40000_8 - 45777_8 and 46000_8 - 53777_8 for the data, i.e., 3K.

Figure II.8.5 shows the page table selection if the following conditions are fulfilled.

1. Bit 0 in the status register (PM - page index module) should be one.
2. PCR 1 should contain 2 in the PT field and 3 in the APT field.
3. All instructions should be fetched using P relative addressing (always true, except for indirect jump).

All data should be accessed using B or X relative addressing (controlled by the programmer or eventually a compiler).

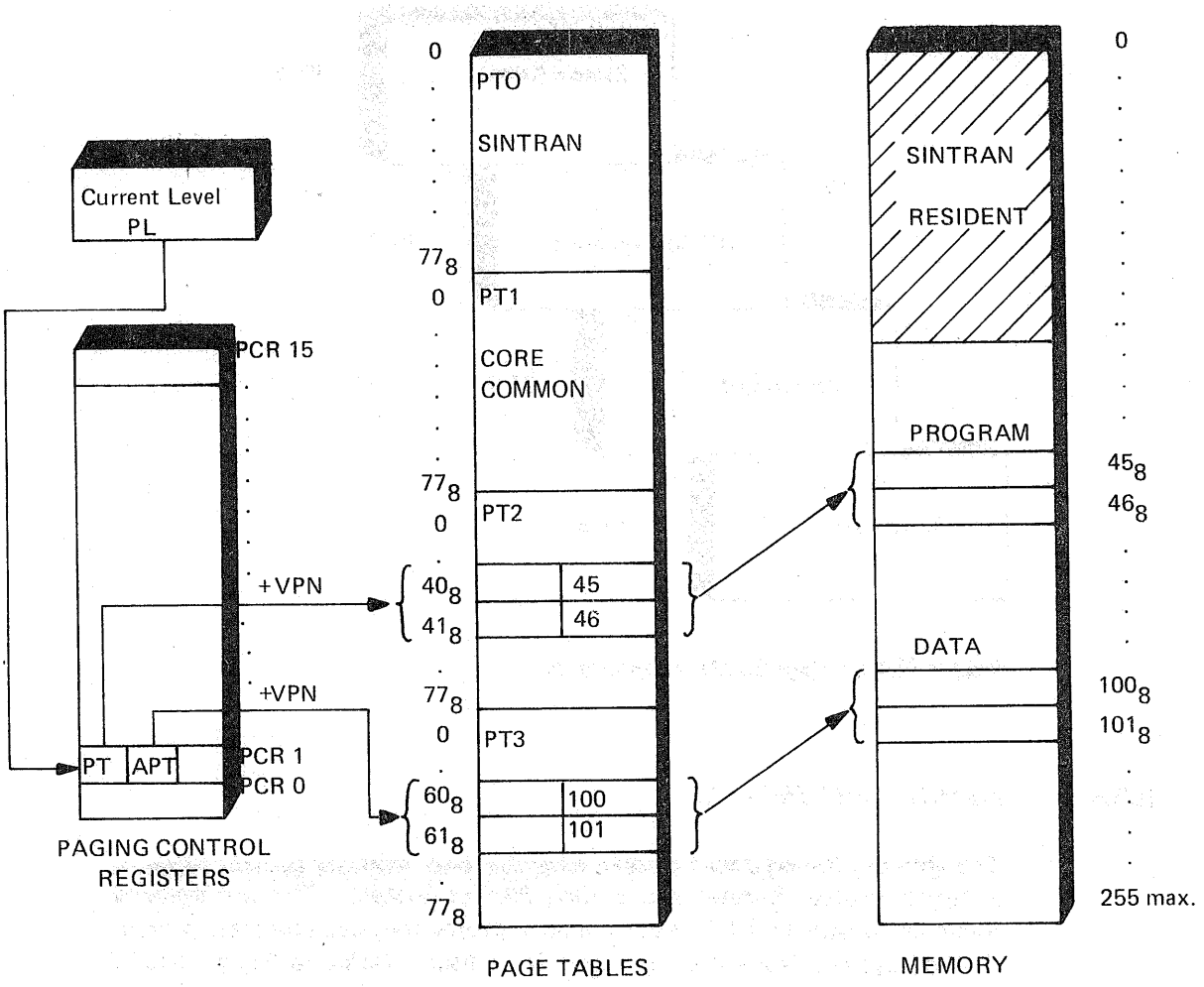


Figure II.8.5: *PCR and PT Usage*

Virtual addresses may overlap if a program uses two segments (mapped through PT and APT).

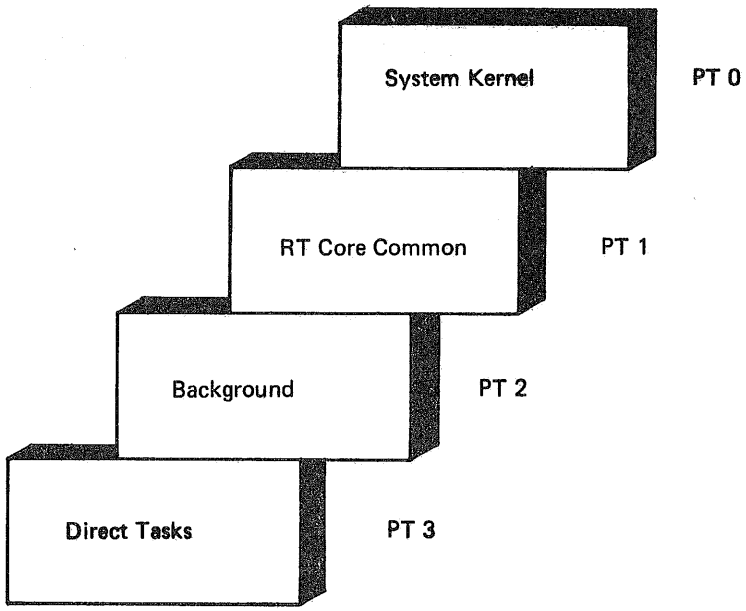


Figure II.8.6: *Page Table Assignments*

II.8.5 *MEMORY PROTECTION*

The Memory Management System employs two memory protect systems: a Permit Protect System and a Ring Protect System. The two systems together constitute a fail-proof memory protection, i.e., complete protection of system from user and user from user. Refer to Figure II.8.7.

The Memory Protect System works on 1K pages. If a memory access violates any of the protection systems, an interrupt to program level 14 will occur with the Internal Interrupt Code equal to 2 = MPV.

II.8.5.1 *Permit Protection System*

The Read, Write and Fetch Protect System is implemented by defining in bits PIT 13-15 how the page may be used. In hardware, this information is compared with the instruction being executed, i.e., if it is LOAD (Read), STORE (Write), INSTRUCTION FETCH or INDIRECT ADDRESS. Figure II.8.7 shows the exact format of the contents of a given PT address.

Bit 15 WPM – Write Permitted.

WPM = 0: It is impossible to write into locations in this page.
WPM = 1: Locations in this page may be written into if the Ring Protect System, RPS, allows.

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

Bit 14 RPM – Read Permitted.

RPM = 0: Locations in this page may not be read (they may be executed).
RPM = 1: Locations in this page may be read if the RPS allows.

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

Bit 13 FPM – Fetch Permitted.

FPM = 0: Locations in this page may not be executed as instructions.
FPM = 1: Locations in this page may be used as instructions.

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

Indirect addresses may be taken both from pages which have FPM = 1 and from 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 with Internal Interrupt Code, IIC, equal to Page Fault.

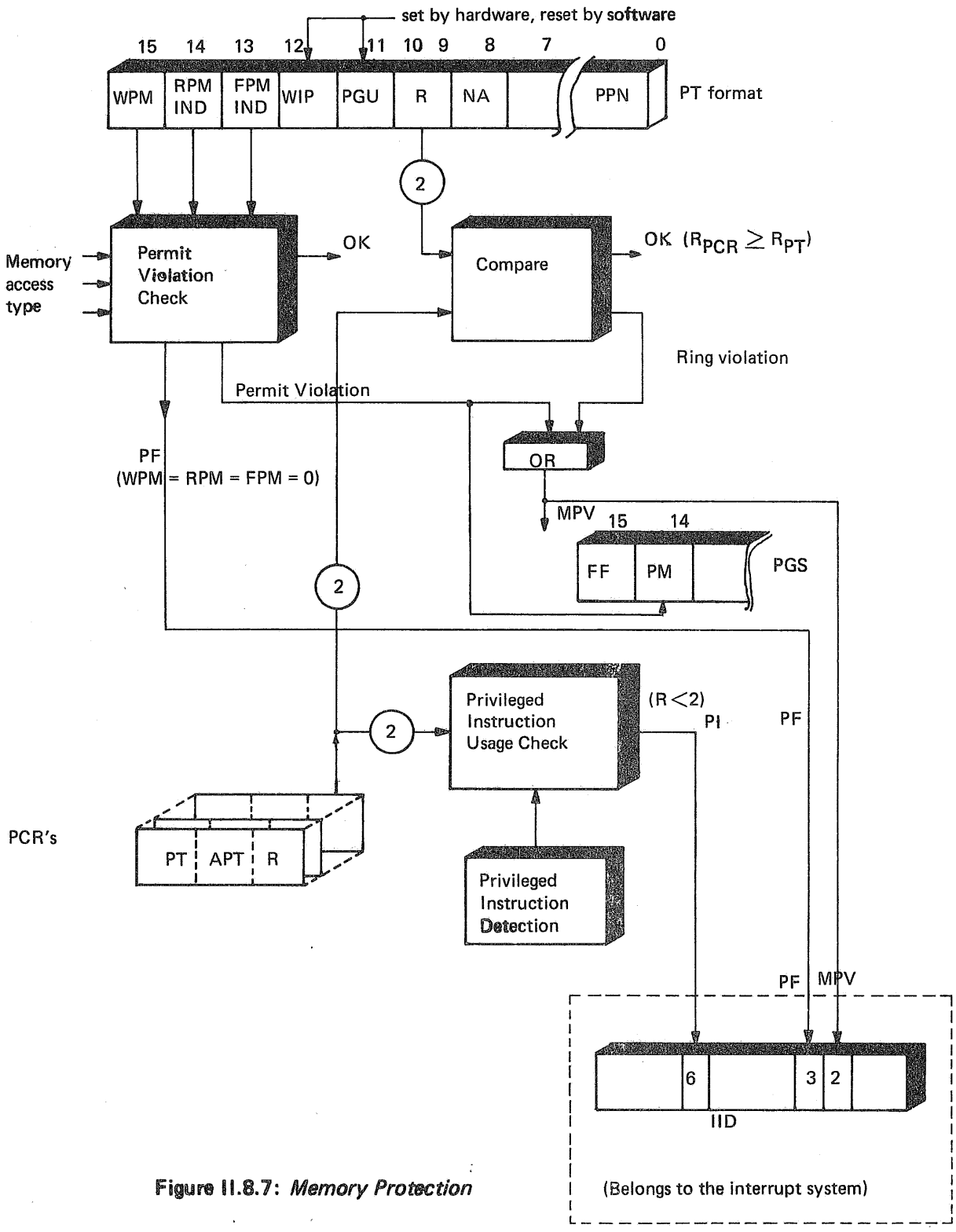


Figure II.8.7: ***Memory Protection***

Abbreviations in Figure II.8.7:

FF:	fetch fault
FPM:	fetch permitted
IID:	internal interrupt detect register
IND:	indirect address readout permitted
MPV:	memory protect violation
PCR:	paging control register
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

Example:

Assume that a user has a virtual address space, equal to 40000_8 - 60000_8. The Operating System is using the logical or virtual pages from 0 - 15_8. Refer to Figure II.8.8.

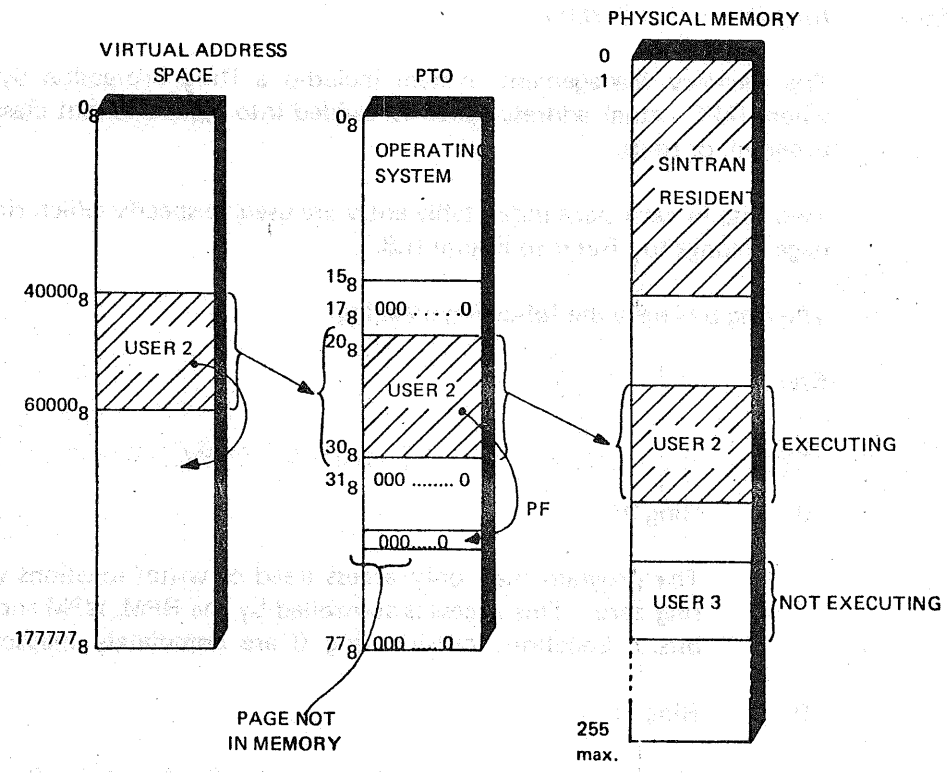


Figure II.8.8: *Virtual Address Space to Physical Memory*

In physical memory there are three users, user 2 is executing. In the PIT user 2 has his page index table. The rest of PIT has all locations equal to zero except for the part used by the Operating System.

The following description will show how users 1 and 3 are protected against user 2. (The Operating System is protected by the Ring Protect System.)

Assume that while user 2 is executing, he makes an error and jumps out of his predefined address space. The result will be that the logical page number he refers to has its contents all equal to zero. However, when bits 13 to 15 are all zero, it is interpreted as Page Not In Memory which in turn generates an internal interrupt with IIC equal to Page Fault, the Operating System has been altered and user 2 is thus prevented from doing this illegal access outside his predefined address space.

The above example describes how a user is protected from other users. If, in the above example, the illegal access had been such that he had addressed logical pages between 0-15, he would have found the PIT for the Operating System. In this case, the Operating System could have been destroyed (depending on WPM, RPM and FPM). To prevent this, the Ring Protect System is used.

II.8.5.2 *Ring Protection System*

The Memory Management System includes a Ring Protection System, where 64K virtual address space is divided into four different classes of program, or rings.

Two bits in each page index table entry are used to specify which ring the page belongs to. Refer to Figure II.8.7.

The ring bits have the following meaning:

Bits:

10 9

0 0 Ring 0:

The program may only access (read or write) locations within ring zero. This access is controlled by the RPM, WPM and FPM bits. Locations outside ring 0 are completely inaccessible.

0 1 Ring 1:

The program may access locations in ring 1 and ring 0. Access is controlled by the RPM, WPM and FPM bits.

1 0 Ring 2:

The program may access locations in ring 2, 1 and 0. The access is controlled by the RPM, WPM and FPM bits.

1 1 Ring 3:

The whole address space is accessible if not protected by the RPM, WPM and FPM bits.

An illegal ring access will cause a privileged instruction interrupt to program level 14, and the instruction which caused the interrupt will not be executed.

Figure II.8.9 illustrates the usage of the ring protection system.

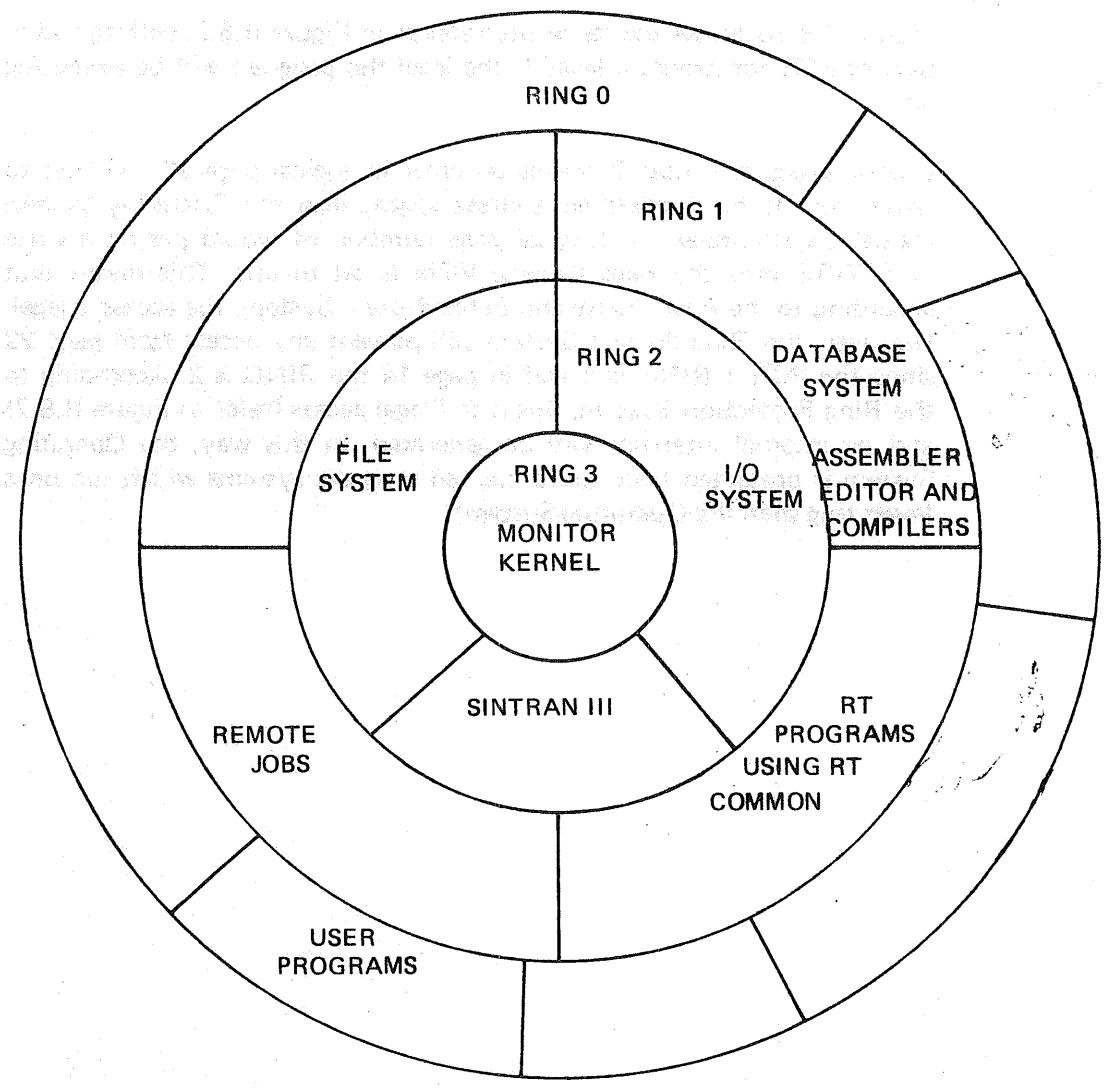


Figure II.8.9: *Ring Assignment*

Associated with the ring bits in a PT entry are the two ring bits in the current program level's Paging Control Register – PCR.

Before a program can start executing, the PCR on relevant program level is loaded by the Operating System with information about which PT, Alternative PT, and Ring is to be used. The program's PT must also be loaded by the Operating System prior to execution. When execution starts, the memory references will be compared with RING in PCR and in the addressed PT entry. The two RINGS should always be equal, otherwise, an internal interrupt will be generated (except for going from a higher to a lower ring).

Example:

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 II.8.10 shows the same situation as in Figure II.8.8, with the addition of PCR for program level 1, the level the program will be executing on.

During execution, user 2 makes an error in logical page 22 and tries to write outside his predefined address space, into the Operating System logical page number 14. Logical page number 14 would permit a write or STORE 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 Ring Protect System will prevent any access from page 22 since the PCR 1 RING is 1 and in page 14 the RING is 2. According to the Ring Protection System, this is an illegal access (refer to Figure II.8.7) and an internal interrupt will be generated. In this way, the Operating System is protected from users (i.e., all program systems which run on a lower ring than the Operating System).

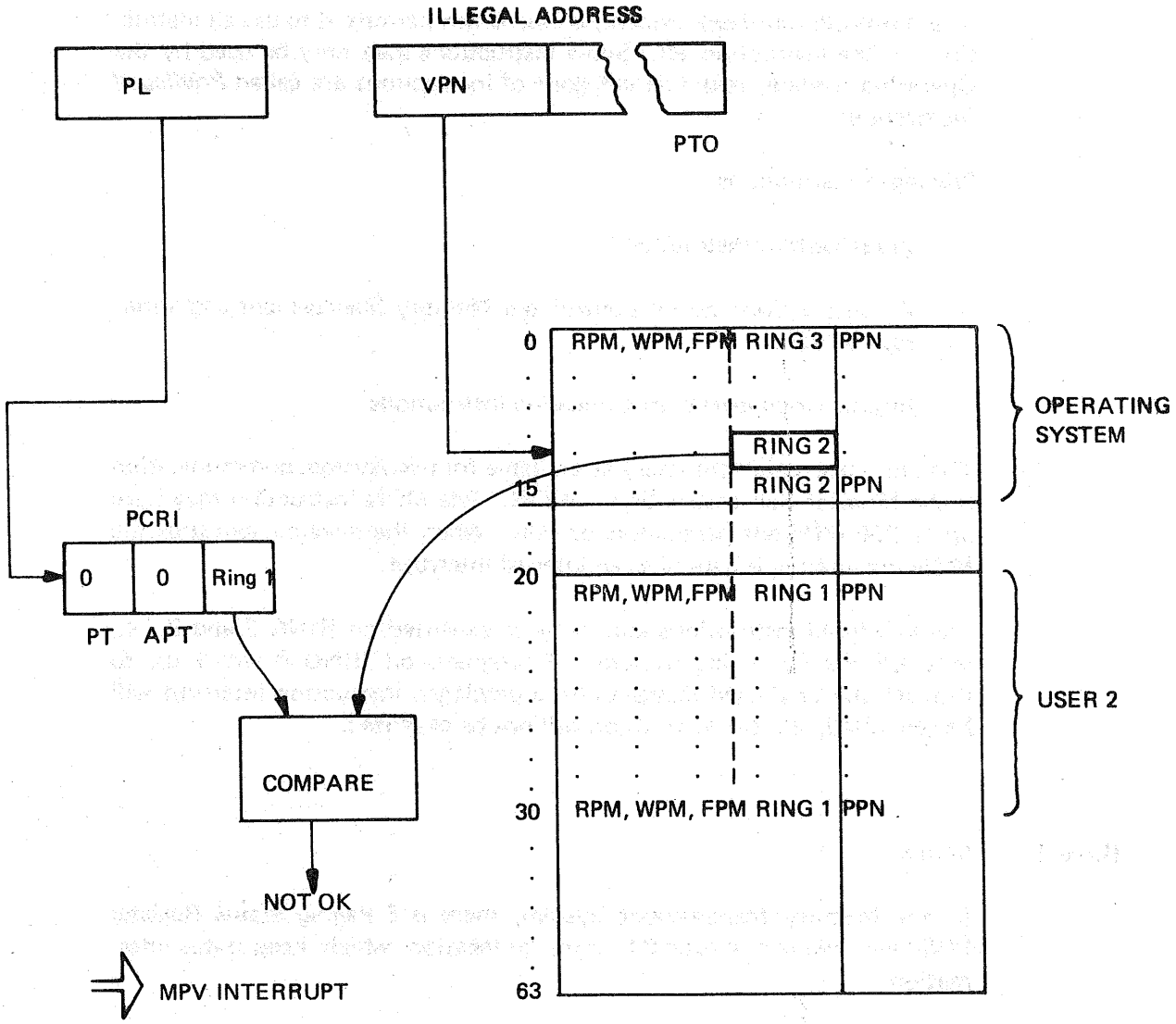


Figure II.8.10: *Page Table Usage, example*

II.8.6 *PRIVILEGED INSTRUCTIONS*

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

Privileged Instructions:

- Input/Output instructions
- All instructions which control the Memory Management and Interrupt System
- Interprogram level communication instructions

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

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

II.8.6.1 *Status*

In the Memory Management System, there is a Paging Status Register (PGS) and two bits in each PT entry or location which keep status information.

Bits 11 and 12 in each PT entry give information about the page whether it has been modified (written into – WIP) or referenced at all (page used – PGU).

When the Operating System has decided to replace a page, it uses the WIP bit to determine if it is required to write the page back to mass storage or not. If WIP = 1, it means that the page in memory has been written into and the copy which is maintained on mass storage, is different from the page in memory. The page in memory must then be transferred to mass storage.

If WIP = 0 it signifies that the copy on mass storage is identical to the page in memory. In this case, it is not required to transfer the page in memory to mass storage, and all overhead associated with a mass storage transfer has been nullified.

The setting of WIP and PGU is done in hardware. The Page Used bit (PGU) is used by the operating system to maintain a record of the access frequency of a page in a certain period of time. This is used in the decision making in the replacement algorithm.

II.8.6.2 *Replacement Algorithms*

There are different page replacement algorithms:

FIFO First In - First Out, where the pages are transferred to main memory in exactly the same sequence as they are requested until main memory is exhausted. Then the first page transferred to main memory is written back onto the drum again, and so on.

LRU Least Recently Used, where the page least recently used is the one page which will be written back onto the drum, whenever the main memory becomes exhausted.

The indication field must be reset periodically because the page with the fewest number of accesses will be the page that was most recently in core. It should not necessarily be overlaid.

WS Working Set. This is the set of information at a point of time t, being referenced by a program in virtual time interval (t $-\Delta$t, t). A page which is not referenced in this time interval will be written onto the drum when the memory is exhausted. The working set principle is related to the principle of locality. With locality it is meant that when a program is being executed it favours a subset of pages, and this subset of favoured pages slowly changes membership. Therefore, a working set is expected to contain the most useful pages.

II.8.6.3 *Paging Status Register*

Whenever the Memory Management System reports any errors (Page Fault, Memory Protect Violations), the Operating System is alerted through an internal interrupt with the Internal Interrupt Code equal to the error source. Next, the Operating System will read the Paging Status Register for further information. The paging status register is used for further specifications when a page fault or a memory protect violation occurs.

The instruction TRA PGS is used to read this register.

Errors lock the PGS register, TRA PGS unlocks it again. The bits in PSR have the following meaning:

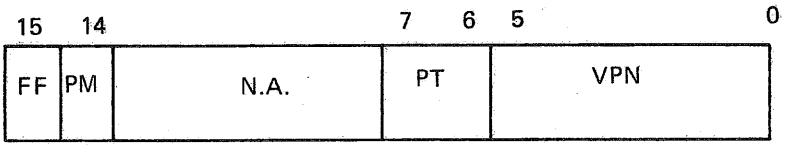


Figure II.8.11: *PSR Format*

Bit 15: Memory management interrupt occurred during an instruction fetch.

Bit 14: 1 - Permit violation interrupt (Read, Write, Fetch Protect System).

0 - Ring protect violation interrupt.

Permit violation has priority if both conditions occur.

Bit 7-6: Page table number

Bit 5-0: Virtual or logical page number.

Note that bits 0 - 7 are the eight least significant bits of the physical page table entry.

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

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

It is possible that executing a floating point instruction, four page faults may occur before the instruction may be started. (Fetch fault, a new fetch fault because of indirect addressing, data bycle fault and a new data cycle fault because the data was placed on a page boundary.) Therefore, a minimum of four pages in main memory is necessary in order to execute a general program requiring 64K virtual memory space.

II.8.7 *CONTROL OF MEMORY MANAGEMENT SYSTEM (MMS)*

II.8.7.1 *General*

The Memory Management System is controlled by the two instructions PON and POF.

PON Turn on Memory Management System (Paging on)

The instruction executed after the PON instruction will go through the address mapping mechanisms.

POF Turn off Memory Management System

The instruction will turn off the Memory Management System, and the next instruction will be taken from a physical address in the lower 64K, specified by the virtual address following the POF instruction.

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

II.8.7.2 *Control of Paging Control Registers – PCR's*

The setting of the PCR's is done by the operating system prior to the program execution. Only one PCR may be written into at the time by the instruction TRR PCR. Refer to Figure II.8.12.

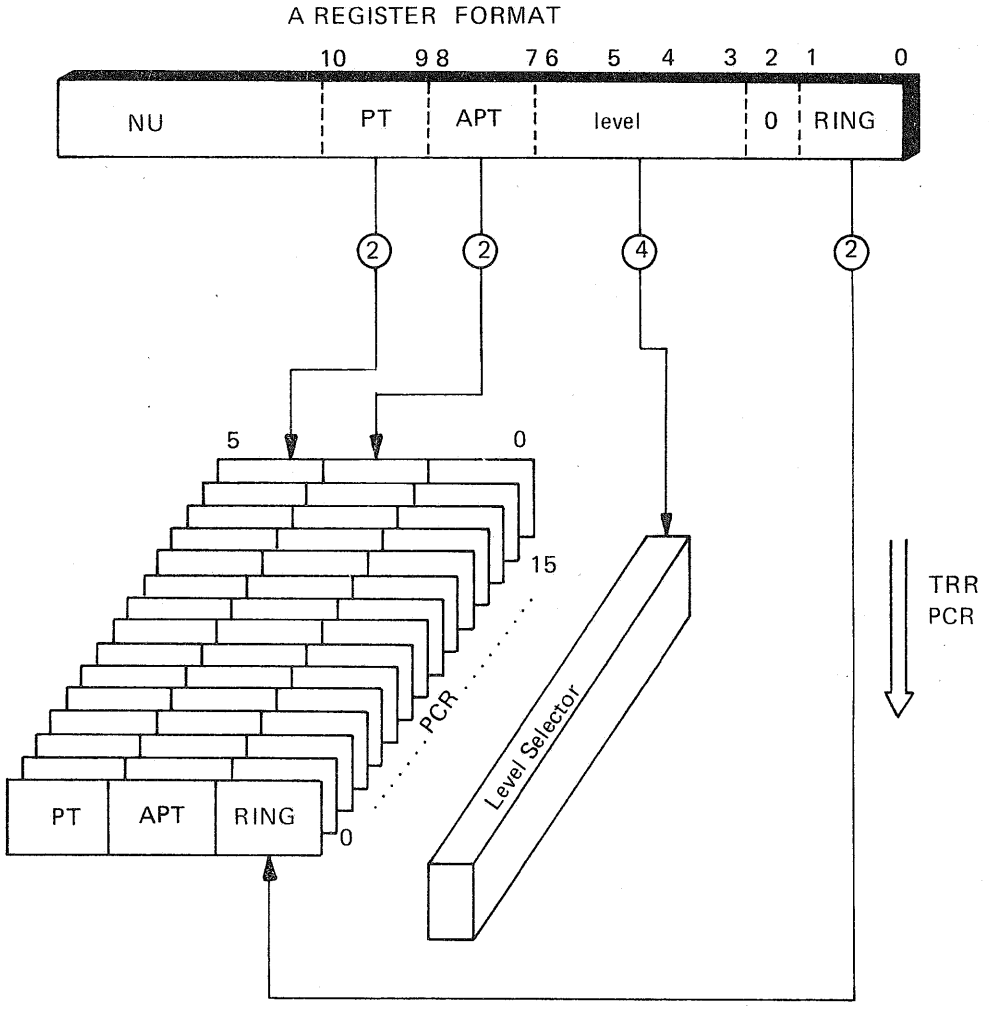


Figure II.8.12: *TRR PCR Flow Diagram*

II.8.7.3 *Control of Page Tables – PT's*

The operating system keeps track of the information to be written into the Page Table prior to program start. This information is taken from the Core-Map, a memory administration table belonging to the operating system.

There is no special instruction used for reading or writing the PT's.

Setting up certain conditions, the CPU can access the PT's as they were main memory in the virtual address space 177400_8 to 177777_8, i.e., the last 256_{10} locations of the 64K words virtual memory. In this mode, the 4 PT's are regarded as one high speed bipolar register file of 256 locations. Because of the logical location, the 4 PT's are referred to as Shadow Memory.

Two conditions can be set up by the CPU to reach the Shadow Memory.

– MMS active *and* Ring 3

or

– MMS off.

The address of 177400 is a bias address to the PT's, i.e., 177400 corresponds to PT address 0, etc.

Table II.8.2 shows the address range for the 4 PT's.

$177400_8 - 177477_8$: Page index table 0
$177500_8 - 177577_8$: Page index table 1
$177600_8 - 177677_8$: Page index table 2
$177700_8 - 177777_8$: Page index table 3

Table II.8.2: *PT's Address Range*

Figure II.8.13 will help illustrate.

Reading and writing the PT's are done by using any LOAD or STORE instructions.

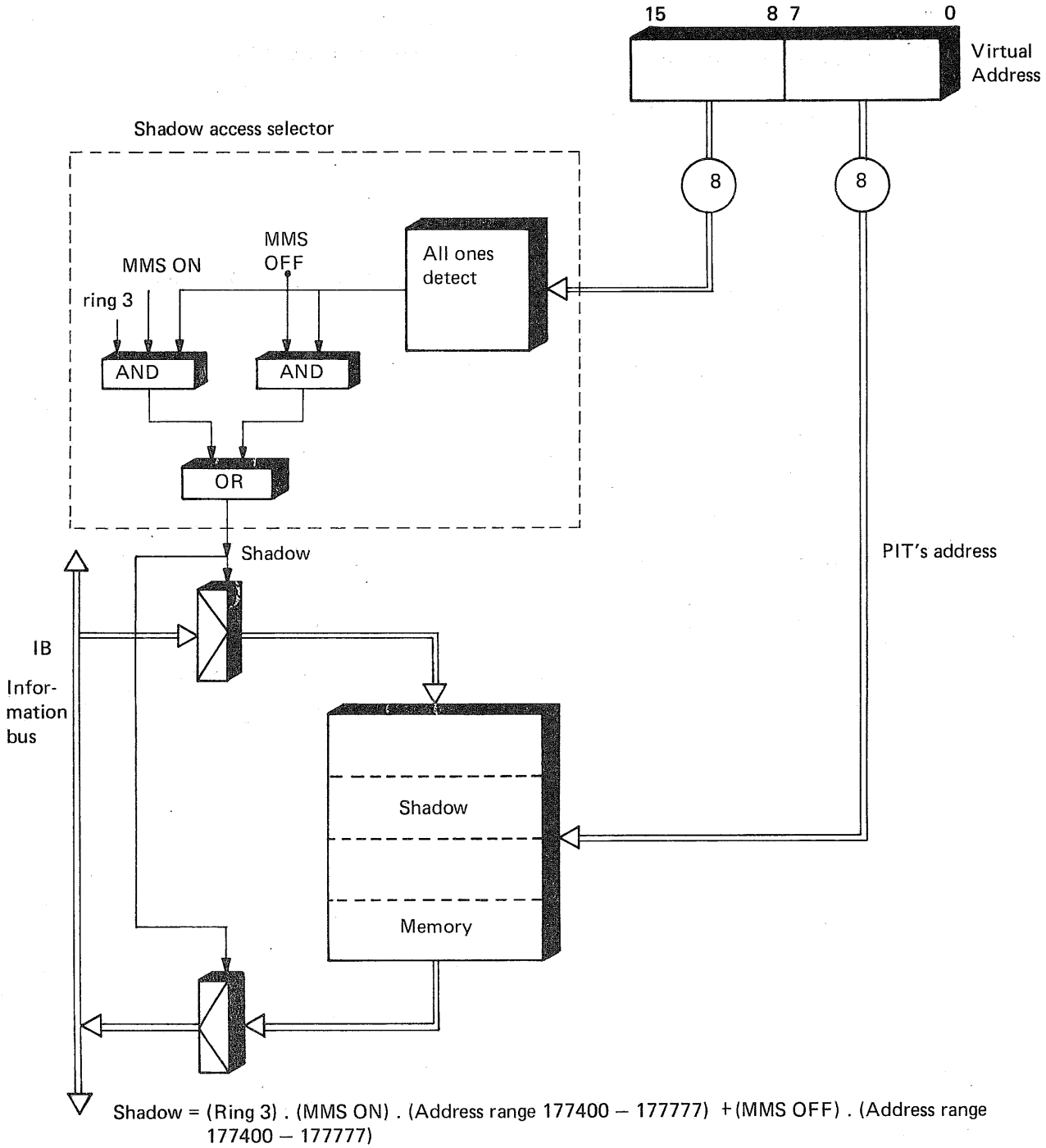


Figure II.8.13: *Shadow Memory Access*

II.8.7.4 *Timing*

Page table accesses are performed in parallel with cache memory look-up, and consequently there is no timing overhead associated with the Memory management system. However, if the cache memory option is not installed and the Memory management system is turned off, execution times are reduced by 0.1 μs for each memory reference. (LDA will use 1.7 μs instead of 1.9 μs.)

II.8.8 *PROBLEMS*

1. If Virtual Address = 36250_8, what PT address is used, and what must the contents of the PT address be if physical address is equal to 2250_8?

2. If it is permitted to execute and read data from a page which is being referenced, what must protect and ring bits be, when this is executed on ring 0?

3. Virtual Address = 61323_8.

   If physical address is equal to 121323:

   a) What physical page must now be used?
   b) What virtual page is being referenced?

4. If a program which is running on program level 3 executes a MON instruction:

   a) What is happening?
   b) What is PL, PK, PVL after the MON instruction is executed?

5. Program levels 3, 5 10_{10} and 14_{10} are enabled. Two different devices interrupt on program level 10_{10}. Describe what happens when a program on program level 3 executes a MON instruction and the monitor call routine is executed on program level 5. Assume that the MON instruction will be executed prior to the level 10_{10} interrupt.

6. PCR for level 0 has Ring = 10_2, APT and PT = 00_2. A program on level 0 has a virtual address = VA. Page Index Table Address = 33_8. The contents of Page Index Table 0 address 33_8 = 2053_8.

   a) What virtual addresses will point to Page Index Table address 33_8?
   b) What happens the first time Page Index Table address 33_8 is referenced?

7. A Page Fault coincides with a Power Fail. What is IIC?

II.9 CACHE MEMORY

II.9.1 *GENERAL*

Cache memory is also referred to as buffered memory or hidden memory. Cache memory is placed between the CPU and local memory as depicted in Figure II.3.1.

As earlier mentioned, cache memory reduces the average memory access time significantly.

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.

II.9.2 *CACHE MEMORY ARCHITECTURE*

The following key words are associated with the implementation of cache memory:

- Location
- Type
- Size
- Placement/replacement algorithm

II.9.2.1 *Location*

For best benefits of cache, the shortest possible access time is desireable. Cache is therefore located close to the CPU. Refer to Figure II.3.5.

II.9.2.2 *Type*

Simplicity reduces the cost and increases the reliability.

An associative cache memory is therefore chosen.

An associative memory is divided into two parts:

- Data
  and
- Directory

II.9.2.3 *Size*

The purpose of cache memory is to hold the most frequently used information.

The size is therefore mainly dependent of program loop sizes. In other words, the hit ratio one wishes to peruse 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 II.9.1 shows the cache size versus hit ratio.

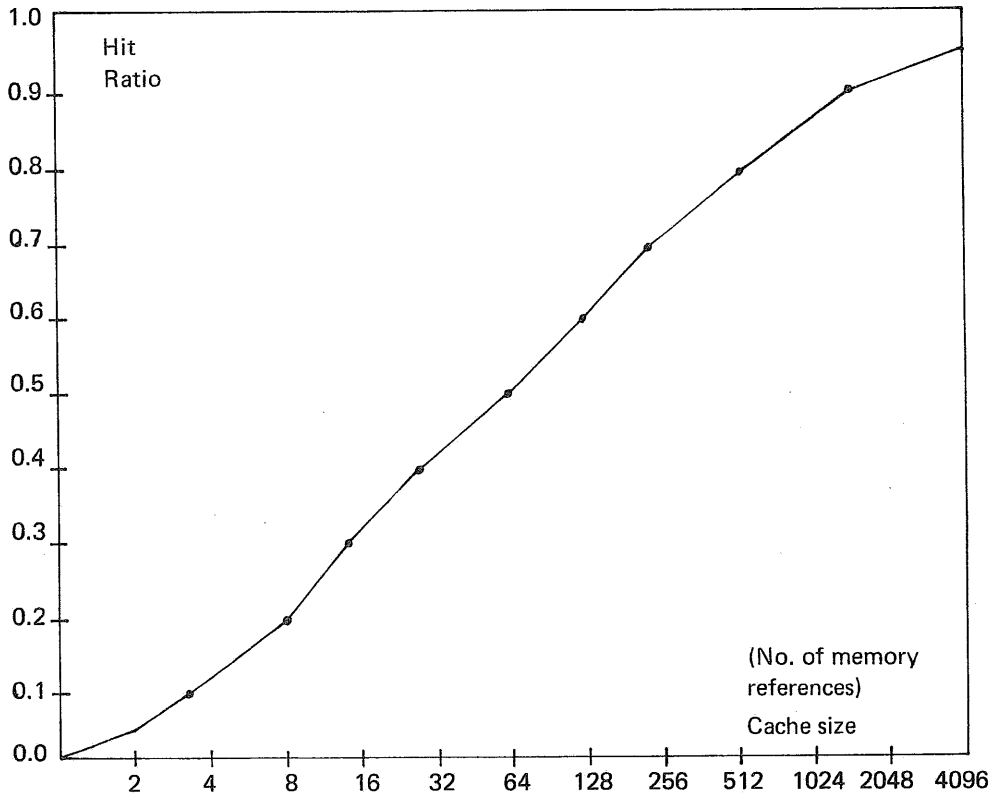


Figure II.9.1: *Hit Ratio versus Cache Size*

It should be noted that the curve will change dependent on how well structure a given program is and on program type.

Since NORD-10/S employs a paging system with page size of 1K word, it is pratical to use a cache memory of the same size.

II.9.2.4 *Placement/Replacement Algorithms*

Two main types of algorithms exist:

WT: Write through
CUW: Conflicting use writeback

CUW gives some higher performance, i.e., shorter average access times. On the other hand, WT:

- is easier to implement
- requires less hardware
- is more reliable due to the above mentioned and in case of power break (main memory is always kept updated)

NORD-10/S uses the WT (write through) algorithm.

The main features of this algorithm can be summarized in the following:

- A write operation goes to Cache Memory as well as Main Memory.

- During a read operation data is taken from Cache Memory if they are found there. Otherwise, they are taken from Main Memory and written into CPU as well as Cache Memory (for probable later use).

II.9.2.5 *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 II.9.1 where the absissa illustrates the number of memory references.

II.9.2.6 *Implementation*

The Cache Memory is organized as a 1K by 25 bits look-up table, as illustrated in Figure II.9.2. A word in Cache is identified with the main memory word of which it is a copy by means of its main memory physical address.

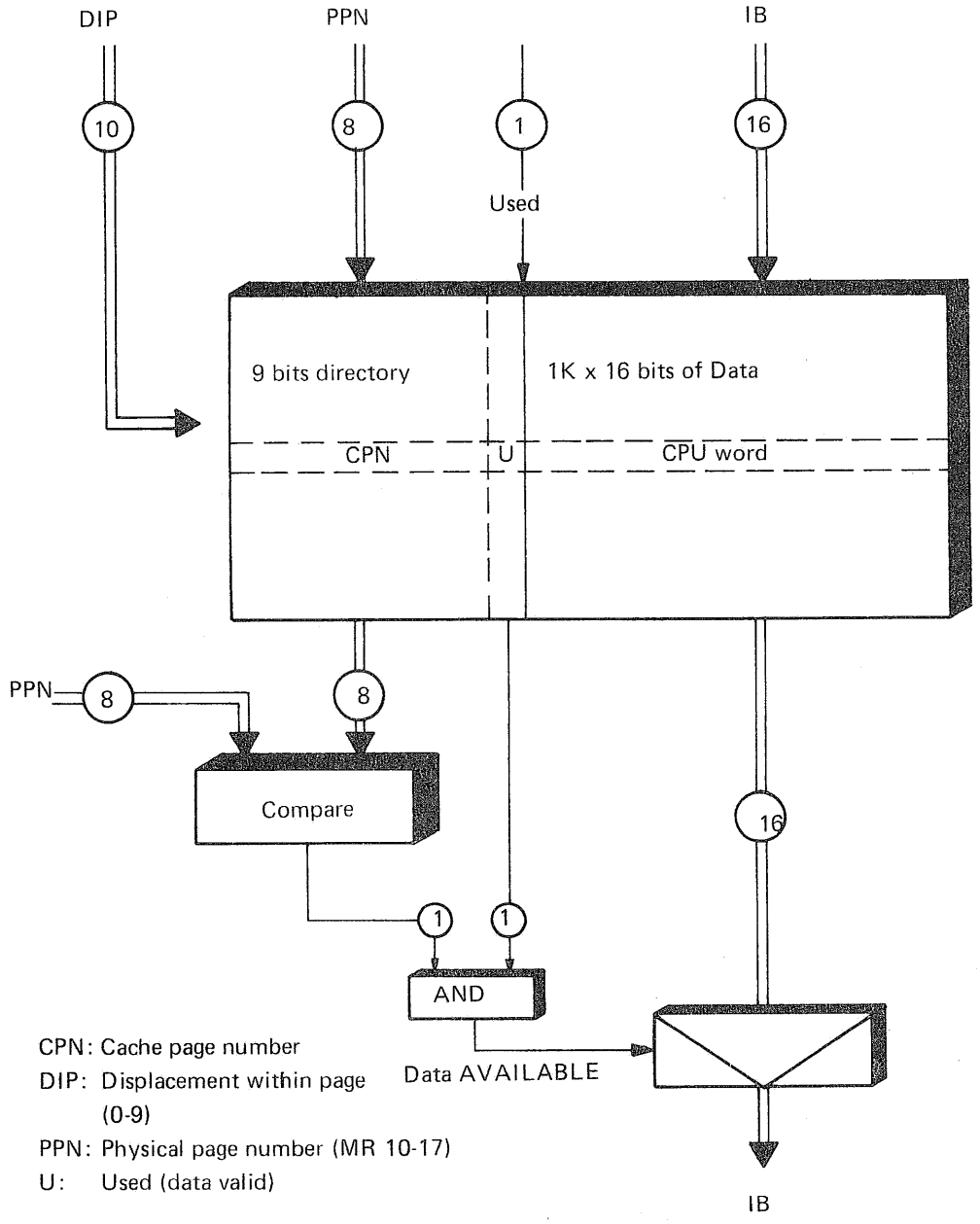


Figure II.9.2: *Cache Memory Organization*

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

Each word in the Cache Memory has the following format:

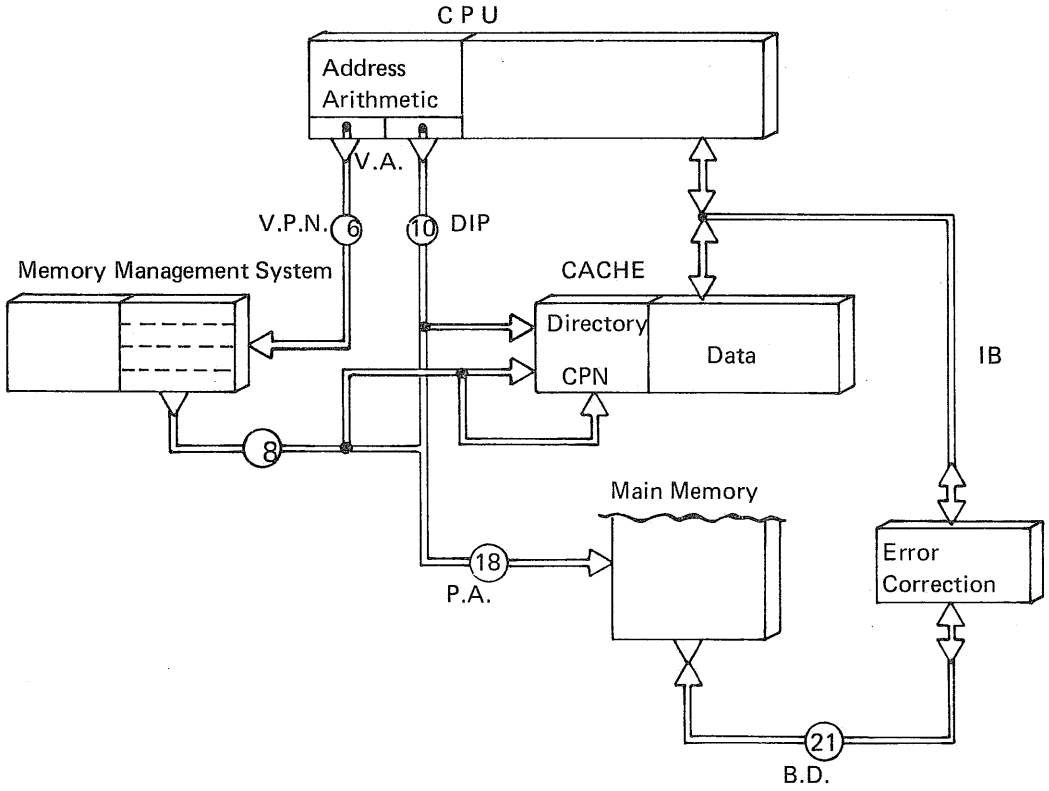
CPN, 8 BITS	U	DATA WORD, 16 BITS

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

U: Use bit. Indicates that the Cache word in this location is valid.

CPN: Cache Page Number. This is the physical page number of the main memory word of which the cache data word is a copy.

The interconnection between Cache Memory, Memory Management, CPU and Main Memory is depicted in Figure II.9.3.



BD: Bus Data (IB +control code)
CPN: Cache page number (MR 10-17)
DIP: Displacement within page (R 0-9)
IB: Information bus
PA: Physical address (MR 0-17)
PPN: Physical page number (MR 10-17)
VA: Virtual address (R 0-15)
VPN: Virtual page number (R 10-15)

Figure II.9.3: *Address and Data Flow*

II.9.3 *CACHE MEMORY ACCESSES*

II.9.3.1 *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 II.9.2.

The cache look-up is done in parallel with the page table look-up. PPN is then available at the same time as CPN and the two values are compared. If they match, sought data is found in Cache.

II.9.3.2 *Read Access*

During a read operation, information may or may not be found in Cache memory. If PPN=CPN and U (used bit) = 1, information is available from cache and the main memory reference is discontinued in the Memory Interface. The used bit indicates that the data in Cache is valid.

If PPN ≠ CPN or U = 0 the sought information is not present in Cache and the normal memory cycle is initiated.

At the same time as the data is presented for the CPU it is also written into the data part of Cache Memory where the sought information was not found. At the same time, PPN will be written into the Directory (CPN:=PPN) and setting the used bit equal to 1. When writing into cache the old information is simply overwritten.

II.9.3.3 *Write Access*

Requests from the CPU to write into memory are always forwarded to the main memory control. In parallel with the main memory access, a copy is written into the Cache Memory along with its corresponding PPN and setting U:=1. In this manner, the main memory will always contain only relevant and correct information. This is of special importance in case of power failure, and when several processors have access to a shared memory.

II.9.3.4 *Cache Inhibit Area*

The Cache Memory system contains two 8-bit registers which define a contiguous area in memory which will *not* be copied into Cache when accessed. The inhibited area includes all pages with:

Lower limit ⩽ PPN ⩽ Upper limit

The inhibit feature is intended for use on memory areas that are operated upon by high-frequency DMA transfers and/or parallel processors, to ensure that the CPU does not operate on stale 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 should always be accompanied by Cache initialization (see Section II.9.4.2).

"Master Clear" will cause all of main memory to be inhibited.

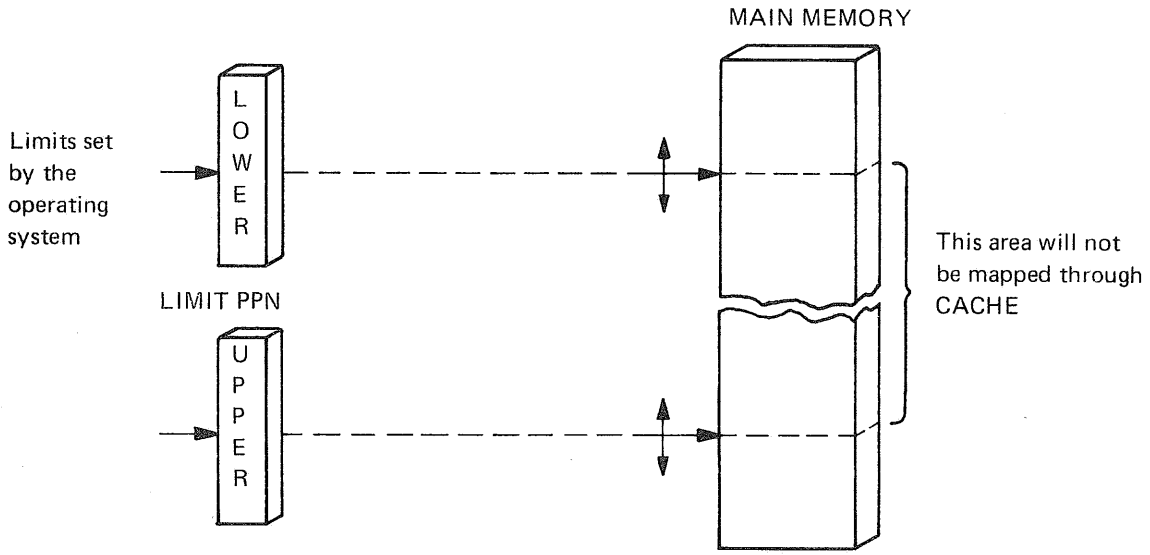


Figure II.9.4: *Cache Limits*

II.9.4 *CONTROL OF THE CACHE MEMORY*

The Operating System must perform two actions to control the Cache Memory:

- Setting of the Cache inhibit limit registers.
- Initialization of the Cache Memory after a DMA transfer outside the Cache inhibit area.

Note that the Cache control instructions have no effect on machines without the Cache Memory installed.

II.9.4.1 *Setting of Cache Inhibit Limits*

The Cache Inhibit Limits may be set by performing the instruction

TRR 12_8

with the following contents in the A register:

Upper limit (page no.)	Lower limit (page no.)

Note: An expansion of the cache inhibit area should always be followed by the "Clear Cache" instruction (see Section II.9.4.2).

II.9.4.2 *Cache Initialization*

Cache initialization is obtained by performing the "Clear Cache" instruction

TRR 10_8

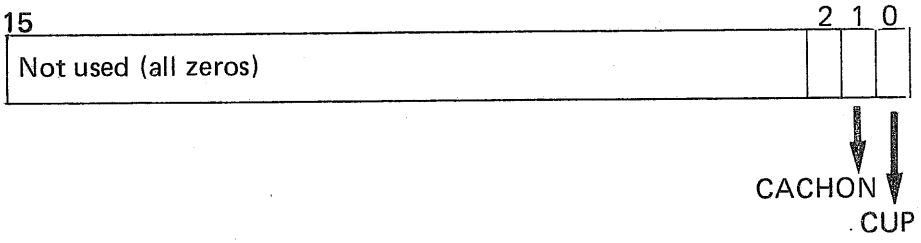
which will clear all U-bits in the Cache Memory. After the "Clear Cache" instructions, the Cache Memory will be disabled for 35 μs while the U-bits are being cleared.

II.9.4.3 *Cache Status Register*

This register is only used by diagnostic programs. It may be transferred to the A register by performing the

TRA 10_8

instruction, and it has the following format:



Bit 0: CPU cache up-date; is a "1" if the last memory reference (i.e., the instruction read-out for the TRA 10_8) caused writing into cache.

Bit 1: CACHON (Cache on); is a "1" on all machines with the cache memory installed, except during the 35 μs period following a "TRR 10_8" instruction or "Master Clear", while the U-bits are being set to zero.

II.9.5 *CACHE TIMING*

The Cache Memory access runs in parallel with the Memory Management Page Table look-up, on the same time base, so that the tests on CPN and U are completed at the same time as the physical memory address is ready. If the data word is available in cache, it will at this time be present at the CPU register inputs, giving a cache access time of 0.2 μs. This is the same as the Memory Management delay, so that the Cache Memory system will not incur any extra delay on memory accesses that must be forwarded to main memory. However, the 0.2 μs delay will now occur on references to main memory also when the Memory Management System is turned off.

Further details regarding timing and parallism is depicted in Figure II.9.5, II.9.6 and II.9.7.

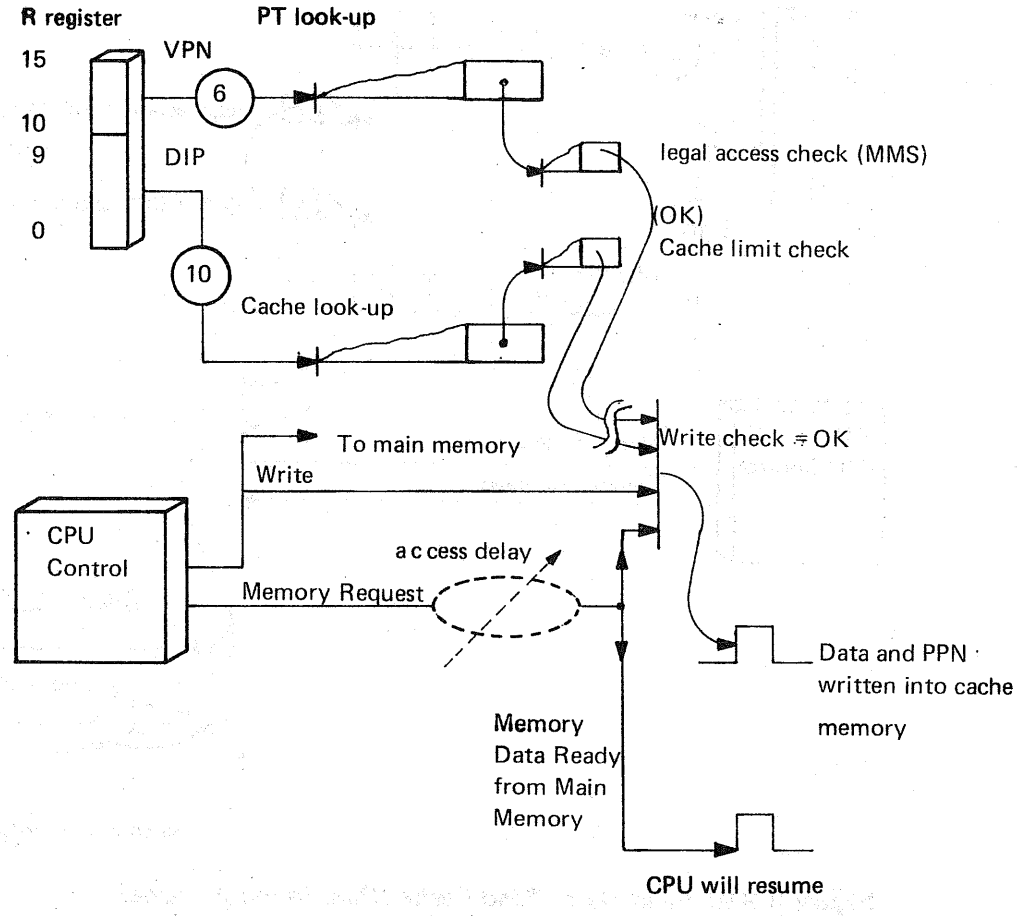


Figure II.9.5: ***Write Data, Write Cache***

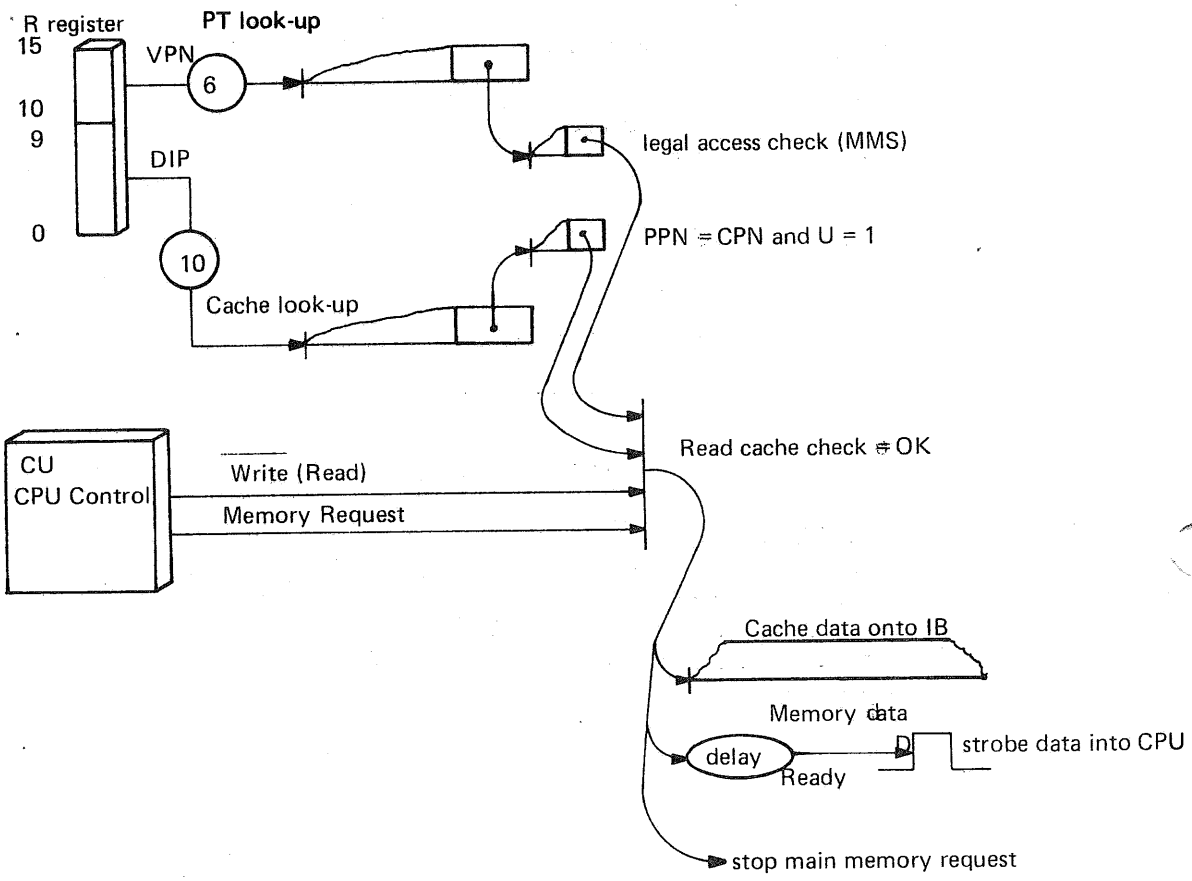


Figure II.9.6: *Read Data, Read Cache (Data found in cache)*

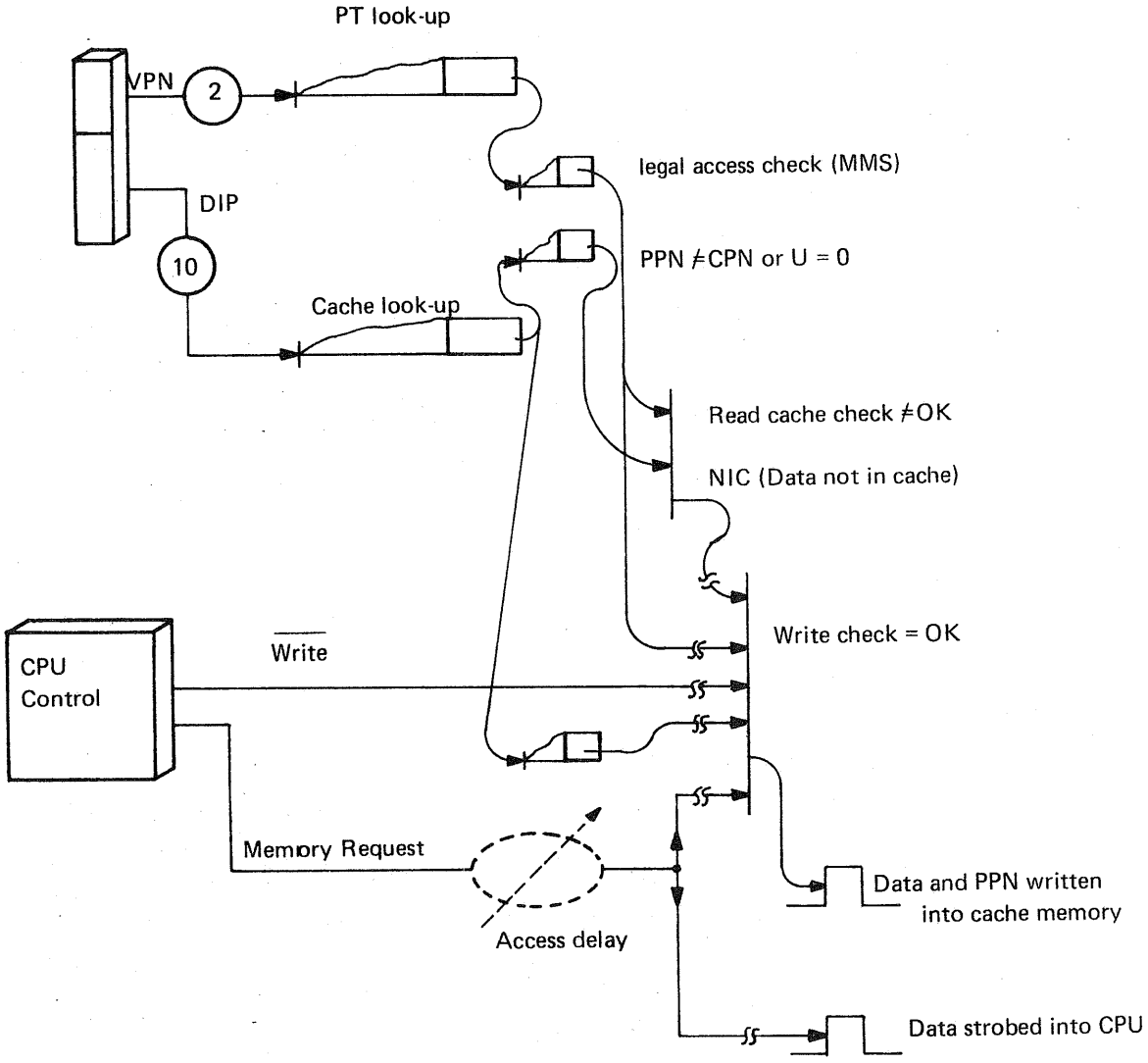


Figure II.9.7: *Read Data, Write Cache (Data not found in cache)*

II.10 POWER FAIL/AUTOMATIC RESTART

II.10.1 *GENERAL*

The Power Fail Unit is physically located in the Power Supply. The purpose of the Power Fail Unit is to detect the presence of the input voltage (230 VAC) and give an early warning to the CPU in case of power failure. This early warning is given through the Internal Interrupt System.

When notified that a power fail is in progress, the operating system will make the necessary steps towards a well defined stop point with its registers saved in memory. When the main power is restored, sensed by the Power Fail Unit, the operating system will go through a restart procedure enabling the executing programs to resume.

II.10.2 *POWER FAIL*

In the Power Fail Unit the input voltage (230 VAC) is full-wave rectified and applied to an RC network with a time constant of approximately 8 ms.

Each 10 ms (half period) the capacitor is charged to the peak voltage, and then allowed to discharge towards an adjustable reference or sense level set up to approximately half the peak value. The sense voltage is supplied by the stand-by battery. A differential amplifier compares the two voltages.

At normal operation, the capacitor is recharged again before reaching the sense level. The Power Fail Unit will then output a steady OK signal. If the input voltage drops below a certain level (approximately 200 VAC) or is missing completely, the capacitor will discharge below the sense level giving a false OK signal. Refer to Figure II.10.1.

A Power Fail Interrupt is generated and approximately 8 ms later a Memory Inhibit signal is enabled. In the meantime the CPU has saved all its registers.

The Memory Inhibit signal is used to prevent uncontrolled write operations to memory and disks. Refer also to Figure II.10.1.

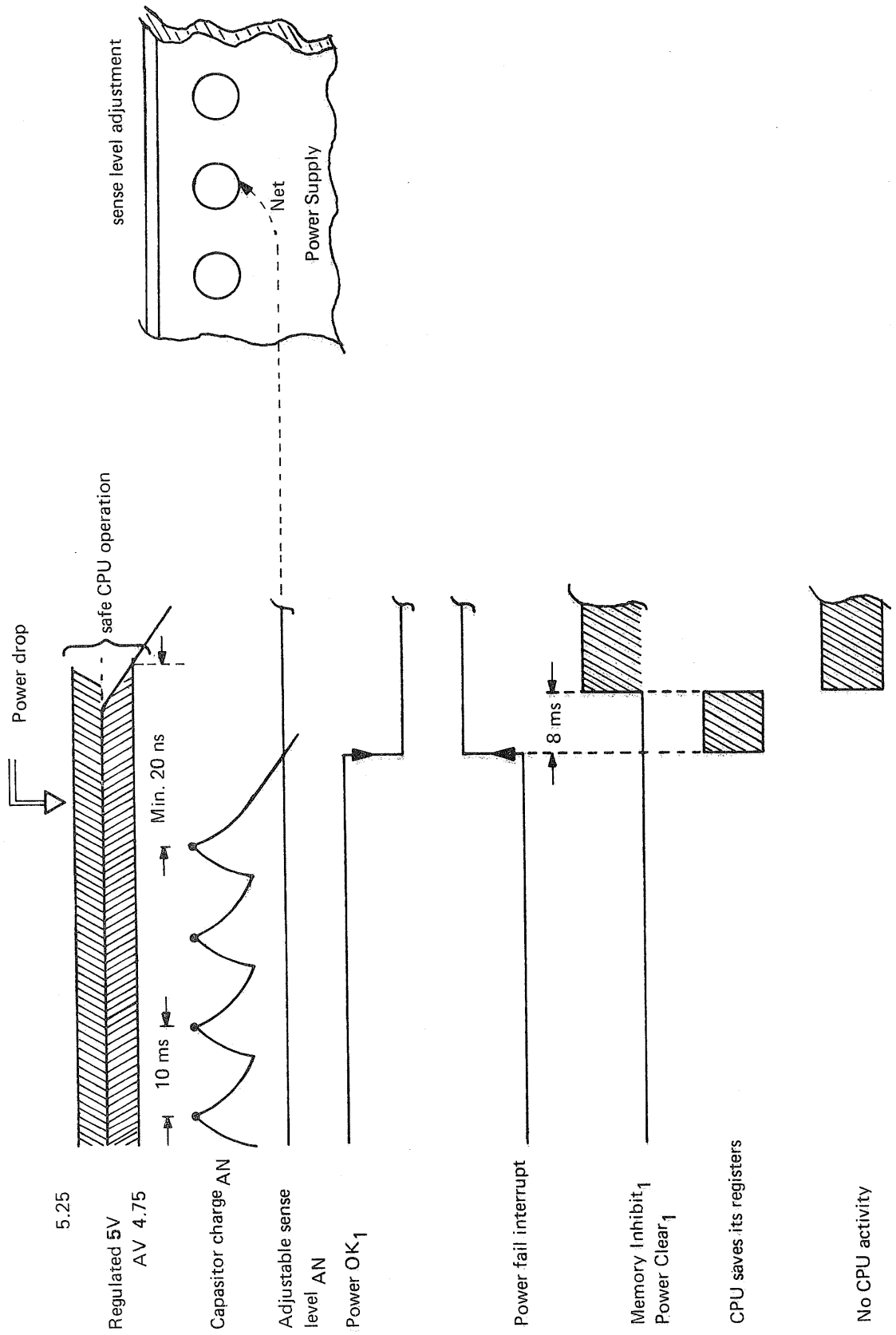


Figure II.10.1: *Power Fail Sequence*

II.10.3 *AUTOMATIC RESTART*

When power again is restored, the capacitor is recharged above the sense level, and after a time delay of approximately 0.7 seconds the OK signal is activated.

When the Power Clear signal disappears the CPU will enter the LOAD sequence. This task is initiated by the microprogram by reading a 16 bits switch register called the Automatic Load Descriptor (ALD register).

This register gives the adequate information regarding a load operation. Refer to Section I.4 for format and description of the ALD register. Refer also to Figure II.10.2.

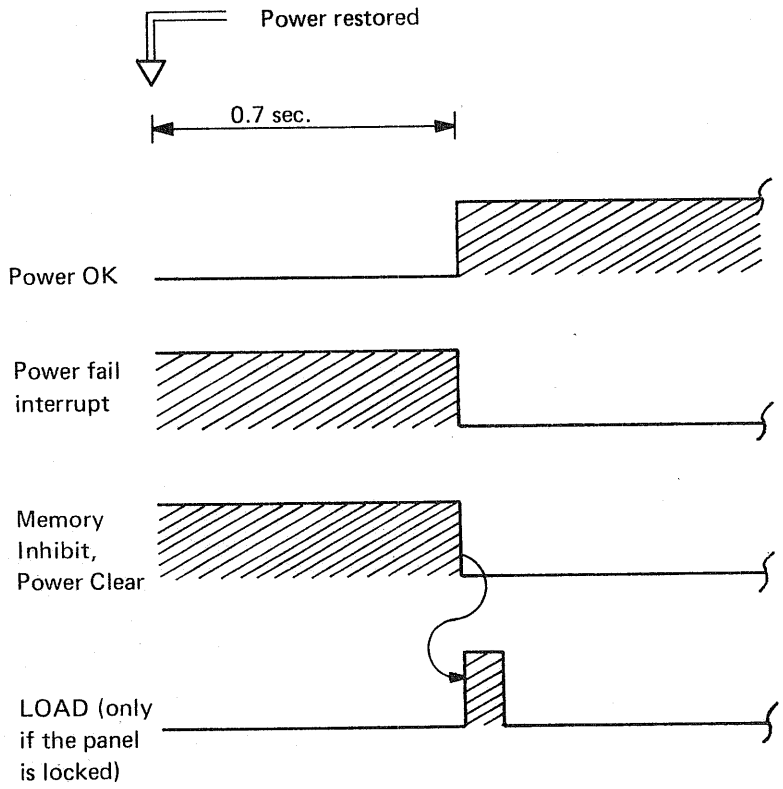


Figure II.10.2: *Automatic Restart*

DETAILED CONTENTS

+ + +

SECTION III

Section:		*Page:*
III.1	**System Control Programming**	III–1–1
III.1.1	System Control Instructions	III–1–1
III.1.2	Privileged Instructions	III–1–4
III.1.2.1	Register Block Instructions	III–1–5
III.1.2.2	Inter Level Register Instructions	III–1–7
III.1.3	Internal Registers	III–1–8
III.1.3.1	Instructions Working on the Internal Registers	III–1–9
III.2	**NORD-10/S Instructions**	III–2–1
III.2.1	Memory Reference Instructions	III–2–1
III.2.2	Register Block Instructions	III–2–3
III.2.3	Floating Conversion	III–2–3
III.2.4	Argument Instructions	III–2–4
III.2.5	Register Operations	III–2–4
III.2.6	Bit Instructions	III–2–6
III.2.7	Sequencing Instructions	III–2–7
III.2.8	Shift Instructions	III–2–8
III.2.9	Transfer Instructions	III–2–8
III.2.10	Execute Instruction	III–2–9
III.2.11	System Control Instructions	III–2–10
III.2.12	Input/Output Control	III–2–11
III.2.13	Interrupt Identification	III–2–11
III.2.14	NORD-10 Mnemonics and their Octal Values	III–2–12
III.3	**Programming Specification for I/O Devices**	III–3–1
III.3.1	NORD-10/S Teletype Programming Specifications	III–3–1
III.3.2	Specification of Tape Reader Interface	III–3–5
III.3.3	Specification of the Paper Tape Punch Interface	III–3–6
III.3.4	NORD-10/S Floppy Disk Programming Specification	III–3–7
III.3.4.1	Device Register Address	III–3–7

Section:		*Page:*
III.3.4.2	Instruction Formats and Descriptions	III–3–8
III.3.4.2.1	Read Data Buffer (IOX RDAT)	III–3–8
III.3.4.2.2	Write Data Buffer (IOX WDAT)	III–3–8
III.3.4.2.3	Read Status Register Number 1 (IOX RSR1)	III–3–8
III.3.4.2.4	Write Control Word (IOX WCWD)	III–3–9
III.3.4.2.5	Read Status Register Number 2 (IOX RSR2)	III–3–9
III.3.4.2.6	Write Drive Address/Write Difference (IOX WDAD)	III–3–10
III.3.4.2.7	Read Test Data (IOX RTST)	III–3–11
III.3.4.2.8	Write Sector/Write Test Byte (IOX WSCT)	III–3–12
III.3.5	Specifications of Line Printer Interface for CDC for NORD-10/S	III–3–13
III.3.6	Specification of Card Reader Buffer (Documentation) for NORD-10/S	III–3–14
III.3.7	NORD-10/S Disk Programming Specifications	III–3–16
III.3.8	NORD-10/S Drum Programming Specifications	III–3–19
III.3.9	Programming Specifications for Hewlett Packard Mag. Tape Controller	III–3–21
III.3.10	Program Specifications for Mag. Tape Formatter (Tandberg)	III–3–22
III.3.11	NORD-10/S Asynchronous Modem Programming Specifications	III–3–25
III.3.12	NORD-10/S Synchronous Modem Programming Specifications	III–3–32
III.4	**Specifications**	III–4–1
III.4.1	Processor	III–4–1
III.4.2	Memory	III–4–1
III.4.3	Interrupt System	III–4–1
III.4.4	I/O System	III–4–1
III.4.5	Physical	III–4–2

III.1 SYSTEM CONTROL PROGRAMMING

III.1.1 *SYSTEM CONTROL INSTRUCTIONS*

The following seven instructions are denotes 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

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

ION Interrupt System On

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 level exists, the ION instruction will be the last instruction executed at the old program level, and the P register at the old program level will point to the instruction after ION. The interrupt indicator on the operator's panel is lighted by the ION. The ION instruction is privileged.

IOF Interrupt System Off

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

IDENT Identify Vectored Interrupt
IDENT <program level number>

When a vectored interrupt occurs, the IDENT instruction is used to identify and service the actual Input/Output device causing the interrupt.

There are four IDENT instructions, one to identify and service Input/Output interrupts on each of the four levels 10, 11, 12 and 13. The particular level to serve is specified by the program level number.

The four instructions are:

IDENT	PL10	Identify Input/Output on level 10
IDENT	PL11	Identify Input/Output on level 11
IDENT	PL12	Identify Input/Output on level 12
IDENT	PL13	Identify Input/Output on level 13

The identification code of the Input/Output device is returned to bits 0 - 8 on 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.

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 serveced. When a device responds to an IDENT, it turns off its interrupt signal. The IDENT instruction is privileged.

PON Memory Management On

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

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

POF Memory Management Off

This instruction is a privileged instruction and may only be executed if the ring bits are 11_2 or 10_2 in PCR on the current level.

The instruction will turn off the memory mangement system, and the next instruction will be taken from a physical address (in lower 64K), specified by the virtual address following the POF instruction.

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

MON Monitor Call
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. MON instruction is *not* a privileged instruction.

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

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

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

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

III.1.2 *PRIVILEGED INSTRUCTIONS*

The instructions available only to programs running in system mode (ring 2 or 3) are termed privileged instructions, which are:

IOF	Turn off interrupt system
ION	Turn on interrupt system
POF	Turn off memory management system
PON	Turn on memory management system
WAIT	Give up priority, reset current PID bit
IOT	NORD-1 compatible Input/Output
IDENT	Identify interrupt
IOX	Input/Output
TRA	Transfer to A register
TRR	Transfer to internal register
MCL	Masked clear of register
MST	Masked set of register
LRB	Load register block
SRB	Store register block
IRW	Inter-register write
IRR	Inter-register read

IOF, ION, POF, PON and WAIT are already discussed. IOT will not be described here. IDENT and IOX are covered in Chapter I.3. The remaining instructions on the above list will be discussed in the following:

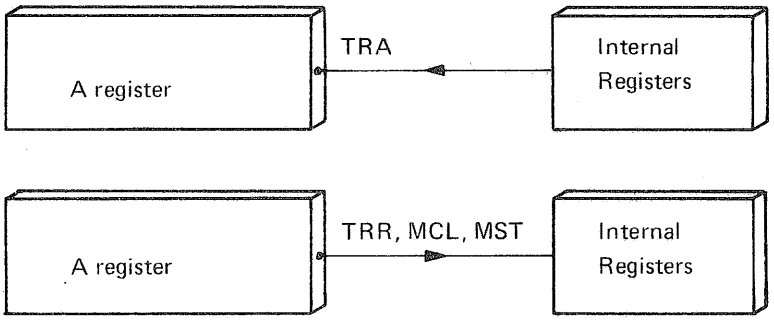


Figure III.1.1: *Internal Register Transfer Instructions*

TRA — Transfer to A register
TRA <register name>

The content of the specified internal register is transferred to the A register. The operator's panel and the paging system 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.

TRR — Transfer to register
TRR <register name>

The contents of the A register are copied into the register specified by <register name>. The TRR instruction is privileged.

MCL — Masked Clear
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 MCL instruction is privileged.

MST — Masked Set
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 MST instruction is privileged.

III.1.2.1 *Register Block Instructions*

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

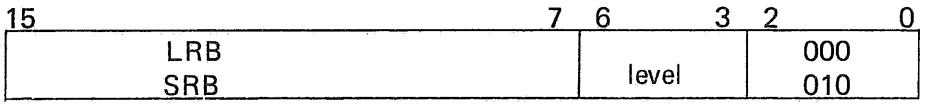
A register block always consists of the following registers in this sequence:

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

The addressing of these two instructions is as follows:

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

The specifications for the two instructions are as follows:



SRB Store Register Block

Format: SRB <$level_8$ * 10_8>

The instruction SRB <$level_8$ * 10_8> stores the contents of the register block on the program level specified in the level field of the instruction. The specified register block is stored in succeeding memory locations starting at the location specified by the contents of the X register.

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

Affected: (EL), +1 +2 +3 +4 +5 +6 +7
P · x T 3 D L STS B

LRB Load Register Block

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 LBR instruction is privileged.

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

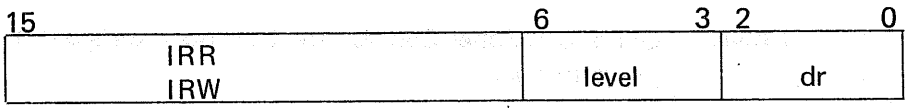
III.1.2.2 *Inter Level Register Instructions*

In the NORD-10/S there are 16 complete sets of registers and status indicators, one set for each level.

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

IRR Inter Register Read
IRW Inter Register Write

The format of this instruction is as follows:



Bits 0 - 2 specify the register to be read, using the same codes and mnemonics as are used for specifying destination register 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 outside program levels.

IRR Inter Register Read

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 register on the specified program level will be read into the A register bits 1 - 7, with bits 8 - 15 and bit 0 cleared.

IRW Inter Register Write

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. It is also possible to write into the registers on the current level. Then, if the P register is specified, the IRW instruction will be a dummy instruction. If bits 0 - 2 are zero, the A register bits 1 - 7 are written into the status register on the specified level.

III.1.3 *INTERNAL REGISTERS*

The following internal registers are implemented for internal control and status of the CPU. Some of the registers are μ-program accessed only. Others can be accessed by privileged instructions. None of these should be accessed by an ordinary customer's program.

Each register has its own code associated with the transfer direction. Internal registers can be accessed when the computer is in STOP mode. Refer also to Chapter I.4.

0	PANS	Operator's Panel Status, used by operator's panel micro-program only.
0	PANC	Operator's Panel Control, used by operator's panel micro-program only.
1	STS	Status Register, program level is contained in bits 8-11, bit 14 = PONI and bit 15 = IONI
2	OPR	Operator's Panel Switch Registe
2	LMP	Operator's Panel Lamp Register (will be overwritten unless U register is selected)
3	PGS	Paging Status Register
3	PCR	Paging Control Register
4	PVL	Previous Program Level (GETR PVL DP)
4	MISC	"Miscellaneous" register (used by micro-program to control IONI, PONI, MCALL and MOPC)
5	IIC	Internal Interrupt Code
5	IIE	Internal Interrupt Enable
6	PID	Priority Interrupt Detect
7	PIE	Priority Interrupt Enable
10	CSR	Cache Status Register, for maintenance only
10	CCLR	Cache Clear
11	ACTL	Active Level, decoded
12	ALD	Automatic Load Descriptor

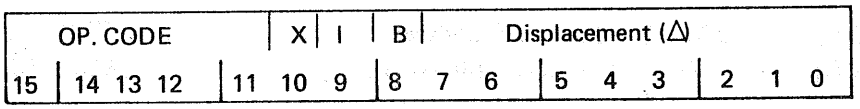
12	CILR	Cache Inhibit Limits Register
13	PES	Memory Error Status
13	CAR	Instruction Register, used by micro-program subroutine only
14	MPC	Microprogram counter (will show a constant)
14	IR	Instruction Register, used by the EXR instruction only
15	PEA	Memory Error Address
15	ECCR	Error Correction Control Register
16	IO	I/O transfer. Do not use.
17	–	Will show an arbitrary register. Do not use.

III.1.3.1 *Instructions Working on the Internal Registers*

Code	Name	TRA	TRR	MCL	MST	IRR/IRW	μ-program use only
0	PANS						x
0	PANC						x
1	STS	x	x	x	x	x	
2	OPR	x					
2	LMP		x				
3	PGS	x					
3	PCR		x				
4	PVL	x					
4	MISC						x
5	IIC	x					
5	IIE		x				
6	PID	x	x	x	x		
7	PIE	x	x	x	x		
10	CSR	x					
10	CCLR		x				
11	ACTL	x					
12	ALD	x					
12	CILR		x				
13	PES	x					
13	CAR						x
14	MPC						x
14	IR						x
15	PEA	x					
15	ECCR		x				
16	IO						x

III.2 NORD-10/S INSTRUCTIONS

III.2.1 *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)
,XI	003000	Post-indexing;	EL = (P + Δ) + X
,B	000400	Address relative to B;	EL = B + Δ
,X ,B	002400	Address relative to B and X;	EL = B + Δ + X
I,B	001400	Pre-indexing;	EL = (B + Δ)
,XI,B	003400	Pre- and Post-indexing;	EL = (B + Δ) + X

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	Memory increment and skip if zero;	(EL): = (EL) + 1

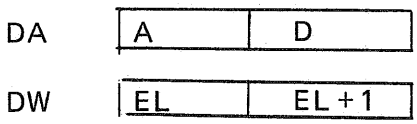
Load Instructions:

LDA	044000	Load A;	A: = (EL)
LDT	040000	Load T;	T: = (EL)
LDX	054000	Load X;	X: = (EL)

Arithmetical and Logical Instructions:

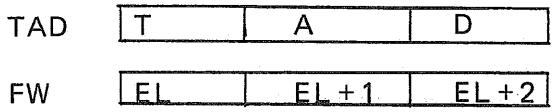
ADD	060000	Add to A (C, O and Q may also be affected);	A: = A + (EL)
SUB	064000	Subtract from A (C, O and Q may also be affected);	A: = A – (EL)
AND	070000	Logical AND to A;	A: = A ∧ (EL)
ORA	074000	Logical inclusive OR to A;	A: = A V (EL)
MPY	120000	Multiply integer (O and Q may also be affected);	A: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

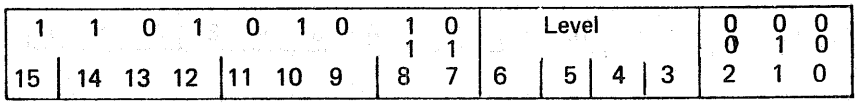
Byte Instructions:

Addressing:

EL=(T) + (X) / 2

Least significant bit of X = 1:		Right byte
Least significant bit of X = 0:		Left byte
SBYT	142600	Store byte
LBYT	142200	Load byte

III.2.2 *REGISTER BLOCK INSTRUCTIONS*



Addressing:

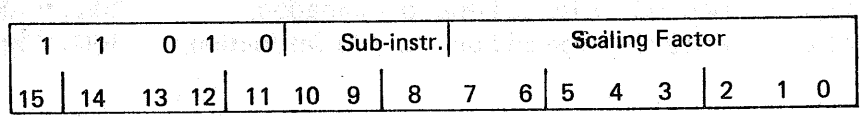
EL = (X) on current level

LRB	152600	Load register block P on specified level: = (EL)
		Load register block X on specified level: = (EL) + 1
		Load register block T on specified level: = (EL) + 2
		Load register block A on specified level: = (EL) + 3
		Load register block D on specified level: = (EL) + 4
		Load register block L on specified level: = (EL) + 5
		Load register block STS on specified level: = (EL) + 6
		Load register block B on specified level: = (EL) + 7
SRB	152402	Store register block

Specified Level:

0	000000	Level 0
01	000010	Level 1
.		.
.		.
017	000170	Level 15

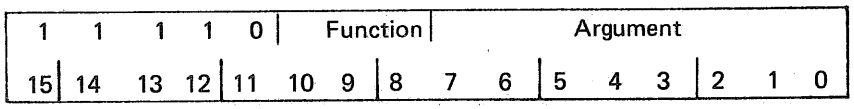
III.2.3 *FLOATING CONVERSION*



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

The range of scaling factor is –128 to 127 which gives converting range from 10^{-39} to 10^{39}.

III.2.4 *ARGUMENT INSTRUCTIONS*

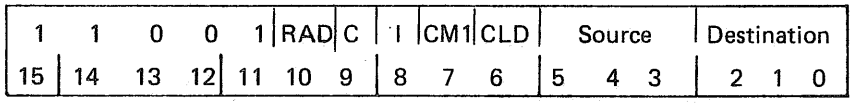


Function:

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

Argument is a signed number ranging from −128 to 127.

III.2.5 *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) $\wedge$ (sr)
REXO	145000	Logical exclusive OR;	(dr): = (dr) $\not\vee$ (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
RCDR	146200 = RADD CM1,	Register decrement

Specify Source Register (sr):

SD	000010	D register as source
SP	000020	P register as source
SB	000030	B register as source
SL	000040	L register as source
SA	000050	A register as source
ST	000060	T register as source
SX	000070	X register as source
	000000	Source value equals zero

Specify Destination Register (dr):

DD	000001	D register as destination
DP	000002	P register as destination
DB	000003	B register as destination
DL	000004	L register as destination
DA	000005	A register as destination
DT	000006	T register as destination
DX	000007	X register as destination

Extended Arithmetic Operations:

RMPY	141200	Multiply source with destination. Result in double accumulator.	AD: = (sr) x (dr)
RDIV	141600	Divide double accumulator with source register. Quotient in A, remainder in D.	A: = AD//(sr)
		(AD = A,* (sr) + D)	

III.2.6 *BIT INSTRUCTIONS*

1	1	1	1	1	Function			Bit no.			Destination		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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 and 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 complement;	K: = K ∧ $(B)_0$
BORC	177400	Logical OR with bit complement;	K: = K V $(B)_0$
BAND	177200	Logical AND to K;	K: = K ∧ (B)
BORA	177600	Logical OR to K;	K: = K V (B)

Specify Conditions:

ZRO	000000	Specified bit equals zero;	(B): = 0
ONE	000200	Specified bit equals one;	(B): = 1
BAC	000600	Specified bit equals K;	(B): = K
BCM	000400	Complement specified bit;	(B): = $(B)_0$

Specify Bit Number:

0	000000	Specifies bit in destination register;	B: = 0
010	000010		B: = 1
020	000020		B: = 2
.			
.			
0170	000170		B: = 15

For destination (D) mnemonics, see the previous section – III.2.5. D = 0 specifies STS register.

Specify Control Flip-flop:

SSTG	000010	specifes:	floating rounding;	B = TG
SSK	000020		one bit accumulator;	B = K
SSZ	000030		floating point overflow;	B = Z
SSQ	000040		dynamic overflow;	B = Q
SSO	000040		static overflow;	B = O
SSC	000060		carry;	B = C
SSM	000070		multi-shift link;	B = M

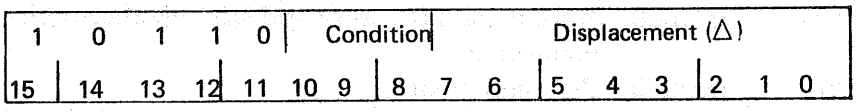
III.2.7 *SEQUENCING INSTRUCTIONS*

Unconditional Jump:

For instruction word format and effective address, see Section III.2.1.

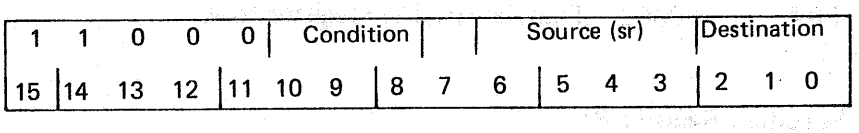
JMP	124000	Jump;	P = EL
JPL	134000	Jump to subroutine;	L = P; P = EL

Conditional Jump:



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

Skip Instructions:



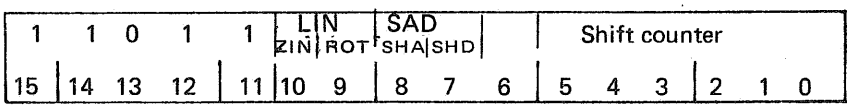
SKP	140000	Skip next location if specified condition is true;	P = P + 1

Specified Condition:

EQL	000000	Equal to
UEQ	0020000	Unequal to
GRE	001000	Signed greater or equal to
LST	003000	Signed less than
MLST	003400	Magnitude less than
MGRE	001400	Magnitude greater or equal to
IF	000000	May be used freely to obtain
O	000000	easy readability

For Source and Destination mnemonics, see Section III.2.5.

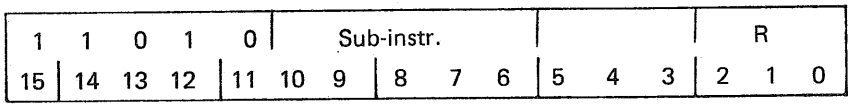
III.2.8 *SHIFT INSTRUCTIONS*



SHT	154000	Shift T register
SHD	154200	Shift D register
SHA	155500	Shift A register
SAD	154600	Shift A and D register 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		Shift right; gives negative shift counter.

III.2.9 *TRANSFER INSTRUCTIONS*

Level Independent Instructions:



TRA	150000	Transfer specified register to A.

Specified Register R:

STS	1	Status register
OPR	2	Operator's panel switch register
PSR	3	Paging status register
PVL	4	Previous level code register
IIC	5	Internal interrupt code register
PID	6	Priority interrupt detect register
PIE	7	Priority enable detect register
ALD	12	Automatic Load descriptor
PES	13	Parity error status register
PEA	15	Parity error address register

TRR	150100	Transfer A to.

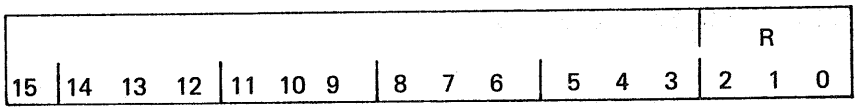
Specified Register R:

STS	01	Status register (bits 1-7)
LMP	02	Panel data display buffer register
PCR	03	Paging control register
IIE	05	Internal interrupt enable register
PID	06	Priority interrupt detect register
PIE	07	Priority enable detect register
MCL	150200	Masked clear of specified register
MST	150300	Masked set of specified register

Specified Register:

STS	000001	Status register (bits 1-7)
PID	000006	Priority interrupt detect register
PIE	000007	Priority interrupt enable register

III.2.10 *EXECUTE INSTRUCTION*

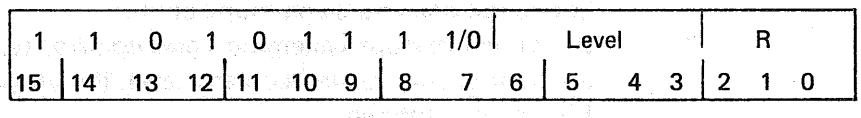


EXR	140600	Execute instruction found in specified register.

Specified Register R:

SD	000010	D register
SB	000030	B register
SL	000040	L register
SA	000050	A register
ST	000060	T register
SX	000070	X register

Inter-level Instructions:



IRR	1536	Inter-register Read A: = Specified register on specified level
IRW	153400	Inter-register Write Specified register on specified level: = A

Specified Register R:

	000000	Status register
DD	000001	D register
DP	000002	P register
DB	000003	B register
DL	000004	L register
DA	000005	A register
DT	000006	T register
DX	000007	X register

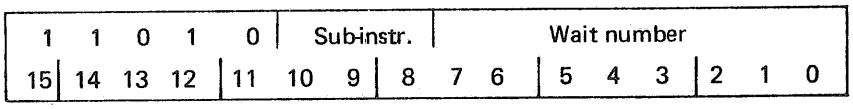
Specified Level:

00	000000	Level 0
010	000010	Level 1
.		
.		
0170	000170	Level 15

III.2.11 *SYSTEM CONTROL INSTRUCTIONS*

ION	150402	Turn on interrupt system
PON	150410	Turn on paging system
IOF	150401	Turn off interrupt system
POF	150404	Turn off paging system

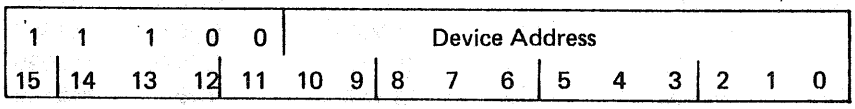
Halt Instructions:



WAIT	151000	1) When interrupt system off: halts the program and enters the operator's communication. 2) When interrupt system on: give up priority. If there are no interrupt requests on any level, the program on level zero is entered.

It is legal to specify a WAIT NUMBER $0 - 377_8$.

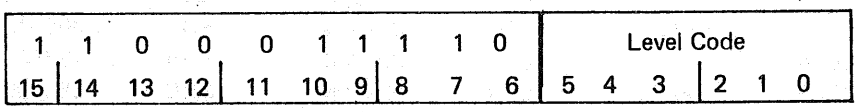
III.2.12 *INPUT/OUTPUT CONTROL*



IOX 164000 Transfer data to/from specified device.

(NORD-10 may also be delivered with a NORD-1 compatible I/O instruction IOT.)

III.2.13 *INTERRUPT IDENTIFICATION*



IDENT 143600 Transfer IDENT code of interrupting device with highest priority on the specified level to A register.

Level Code:

PL10	000004	Level 10
PL11	000011	Level 11
PL12	000022	Level 12
PL13	000043	Level 13

III.2.14 *NORD-10 MNEMONICS AND THEIR OCTAL VALUES*

AAA	172400	IOF	150401	RCLR	146100
AAB	172000	ION	150402	RDCR	146200
AAT	173000	IOT	160000	RDIV	141600
AAX	173400	IOX	164000	REXO	145000
ACT	000400	IRW	153400	RINC	146400
ADC	001000	JAF	131400	RMPY	141200
ADD	060000	JAN	130400	RORA	145400
AD1	000400	JAP	130000	ROT	001000
ALD	000012	JAZ	131000	RSUB	146600
AND	070000	JMP	124000	SA	000050
,B	000400	JNC	132400	SAA	170400
BAC	000600	JPC	132000	SAB	170000
BANC	177000	JPL	134000	SAD	154600
BAND	177200	JXN	133400	SAT	171000
BCM	000400	JXZ	133000	SAX	171400
BLDA	176600	LBYT	142200	SB	000030
BLDC	176400	LDA	044000	SBYT	142600
BORA	177600	LDD	024000	SD	000010
BORC	177400	LDF	034000	SHA	154400
BSET	174000	LDT	050000	SHD	154200
BSKP	175000	LDX	054000	SHR	000200
BSTA	176200	LIN	003000	SHT	154000
BSTC	176000	LMP	000002	SKA	001000
CCLR	000010	LRB	152600	SKP	140000
CILR	000012	LSS	002400	SL	000040
CLD	000100	LST	003000	SP	000020
CM1	000200	MCL	150200	SRB	152402
CM2	000600	MGRE	001400	SSC	000060
COPY	146100	MIN	040000	SSK	000020
CSR	000010	MIX3	143200	SSM	000070
DA	000005	MLST	003400	SSO	000050
DB	000003	MON	153000	SSQ	000040
DD	000001	MPY	120000	SSTG	000010
DL	000004	MST	150300	SSZ	000030
DNZ	152000	NLZ	151400	ST	000060
DP	000002	ONE	000200	STA	004000
DT	000006	OPR	000002	STD	020000
DX	000007	ORA	074000	STF	030000
ECCR	000015	PCR	000003	STS	000001
EQL	000000	PEA	000015	STT	010000
EXIT	146142	PES	000013	STX	014000
EXR	140600	PGS	000003	STZ	000000
FAD	100000	PID	000006	SUB	064000
FDV	114000	PIE	000007	SWAP	144000
FMU	110000	PIN	002000	SX	000070
FSB	104000	PL10	000004	TRA	150000
IRR	153600	PL11	000011	TRR	150100
GEQ	000400	PL12	000022	UEQ	004000
GRE	001000	PL13	000043	WAIT	151000
I	001000	POF	150404	,X	002000
IDENT	143600	PON	150410	ZIN	002000
IF	000000	PVL	000004	ZRO	000000
IIC	000005	RADD	146000		
IIE	000005	RAND	144400		

III.3 PROGRAMMING SPECIFICATION FOR I/O DEVICES

III.3.1 *NORD-10/S TELETYPE PROGRAMMING SPECIFICATIONS*

Teletype Addresses:

The codes below are relevant for the first Teletype (Teletype No. 0). The codes for the first eight Teletypes are found by adding 10_8.N to the codes given. N = Teletype number (0, 1, 2, . . , 7). For the next eight Teletypes the codes are found by adding $(1000 + 10(N - 10))_8$.N = Teletype no. $(10, 11, 12, ..., 17)_8$.

Input Channel (Interrupt Level 12):

Read Data Register

IOX 300

The number of data bits read into the A register is 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

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
Bit 8-14 Not used
Bit 15 Motor stopped

Note: The meaning of the error indicator bits are as follows.

Bit 5: Framing error means that the stop bit is missing.

Bit 6: Parity error means that a parity error has occurred while working in parity generation/checking mode.

Bit 7: Overrun means that at least one character is overwritten.

The meaning of bit 15, motor stopped, is that 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 transmit data back to received data, and nothing is transmittted to the peripheral device.

Bit 11-
Bit 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 Teletype is supposed to be running all the time. If this bit is a 1, the motor is given a stop signal, about 37 seconds after the last character was transmitted, and a start signal about 0.6 seconds before a new character is transmitted after a stop signal.

Output Channel:

Status Register

Bit 0 Ready for transfer interrupt enabled
Bit 1 Error interrupt enabled
Bit 2 Device active
Bit 3 Device ready for transfer
Bit 4 Inclusive OR of errors
Bit 5 Framing error
Bits 6-14 Not used
Bit 15 Motor stopped

Notes:

Bit 2: This status bit will be a 1 as long as the device is busy transmitting characters.

Bit 3: This bit indicates that the output data buffer is ready to receive a new character. This will be a 1 if bit 2 is 0, but may as well be a 1 when bit 2 is 1 due to the double buffer.

Bits 4-5: As there is only one error indicator for the output channel, these two bits have the same meaning. A 1 indicates that the current loop is broken, caused by either a broken line or no peripheral device connected.

Bit 15: The same bit as in Input Channel Status.

Control Register

Bit 0 Enable interrupt on device ready for transfer
Bit 1 Enable interrupt on errors
Bit 2 Activate device
Bits 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.

III.3.2 *SPECIFICATION OF TAPE READER INTERFACE*

Standard device number 0400 $(0400\text{-}0403)_8$
Number of device number 4
Standard interrupt level 12_{10}
Standard ident number 2

Control Word IOX Device Number + 3:

Bit 0 Enable interrupt on ready for transfer
Bit 1 Not used
Bit 2 Activate device (start reading next character on tape)
Bit 3 Test
Bit 4 Device clear
Bits 5-15 Not used

Status Word IOX Device Number + 2:

Bit 0 Interrupt enabled on ready for transfer
Bit 2 Read active
Bit 3 Reader ready for transfer (character read)
IOX device no. + 1 Not used
IOX device no. Read character. The same character may be read several times if desired.

TEST:

When bit 3 in the control word is set to 1, the interface may be tested without reader.

If bit 2 is also set to 1, the interface will give "ready for transfer" after a while.

If "ready for transfer" is constantly 1, the data register will increment for each time IOX device no. + 3 (control word write) is used, and then it is possible to test the data patch.

III.3.3 *SPECIFICATION OF THE PAPER TAPE PUNCH INTERFACE*

Standard device number 0410 $(0410\text{-}0413)_8$
Number of device number 4
Standard interrupt level 10_{10}
Standard ident no. 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 activate 0
Bit 3 Device ready

Write data word IOX Device no. + 1.
Write the 8 bits to be punched in a buffer register.

Read data IOX device number.
Only used under test.

It is not wise to write a character into the buffer if the punch is not ready.

TEST:

The interface may be tested without punch.

When bit 3 in the control word is one, the buffer register may be read back by IOX device number. If the interface is activated, it will become "ready for transfer" after a while.

III.3.4 *NORD-10/S FLOPPY DISK PROGRAMMING SPECIFICATION*

III.3.4.1 *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 Register 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}.

III.3.4.2 *Instruction Formats and Descriptions*

III.3.4.2.1 Read Data Buffer (IOX RDAT)

Reads one 16 bit word from the interface buffer.

Buffer address is automatically incremented after execution of the instruction.

III.3.4.2.2 Write Data Buffer (IOX WDAT)

Writes one 16 bit word to the interface buffer.

Buffer address is automatically incremented after execution of the instruction.

III.3.4.2.3 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.

Bits 9-11: Only used when formatting.

Bit 9 is active when buffer address bits 1 and 6 are active.
Bit 10 is active when buffer address bits 1 and 7 are active.
Bit 11 is active when buffer address bits 1 and 8 are active.

Note: Bits 4-7 are only significant after interrupt or when device busy is reset.

Bits
12-15: Not used.

III.3.4.2.4 Write Control Word (IOX WCWD)

Bit 0 Not used.
Bit 1 Enable interrupt
Bit 2 Not used
Bit 3 Test Mode (for description see IOX RTST and IOX WSCT)
Bit 4 Device Clear (NB: selected drive is deselected)
Bit 5 Clear interface buffer address
Bit 6 Enable time-out
Bit 7 Not used

The following bits are commands to the floppy disk drive and these are the only control bits that generate device busy and give interrupt. (NB: with the exception of bit 15, control reset.)

Bit 8 Format Track
Bit 9 Write Data
Bit 10 Write Deleted Data
Bit 11 Read ID
Bit 12 Read Data
Bit 13 Seek
Bit 14 Recalibrate
Bit 15 Control Reset

III.3.4.2.5 Read Status Register Number 2 (IOX RSR2)

Bit 0-7: Not used

Bit 8: Drive Not Ready

This bit is set if the addressed drive is not powered up, its door is open, the diskette is not properly installed or drive address is invalid.

Bit 9: Write Protect

This bit is set if a write operation is attempted on a write protected diskette.

Bit 10: Not used.

Bit 11: Sector Missing + No AM

This bit is set if the desired sector for a Read Data Write Data or Write Deleted Data cannot be located on the diskette.

In addition, this bit may indicate a non-locatable data field address mark, or a non-locatable ID field address mark.

Bit 12: CRC Error

Bit 13: Not used

Bit 14: Data Overrun

A data byte was lost in the communication between the NORD-10/S interface and the floppy disk system.

Bit 15: Not used.

III.3.4.2.6 Write Drive Address/Write Difference (IOX WDAD)

This is two instructions depending on bit 0 in the A register.

A) Bit 0 = 1: Write Drive Address

This instruction selects drive and format.

Bit 1-7: Not used.

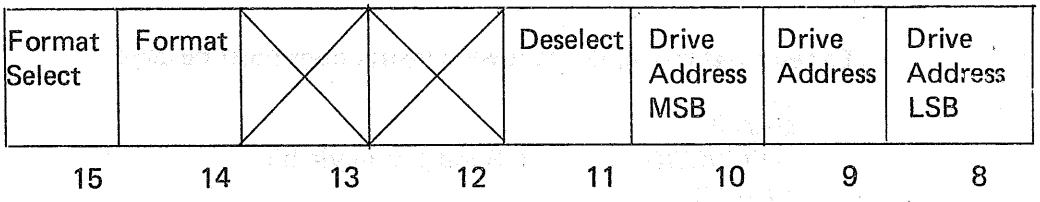
Bit 8-10: Drive Address (unit number) 0, 1 or 2

Bit 11: Deselect Drives

Bit 12-13: Not used.

Bit 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/secotr 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.

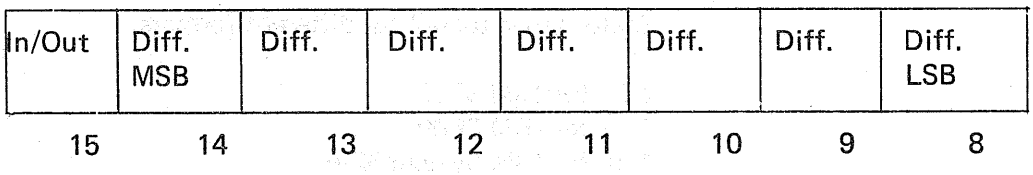
Bit 1-7: Not used.

Bit 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



III.3.4.2.7 Read Test Data (IOX RTST)

This instruction is used for simulation of a data transfer between the floppy disk system and NORD-10/S interface.

It does *not* transfer data from the NORD-10/S interface to the A register, but puts one 8 bit byte into the interface buffer, each time the instruction is executed. The bytes are packed to 16 bit words in the buffer and may later be read by IOX RDAT instructions.

The byte may be chosen by using the IOX WSCT instruction (see description of IOX WSCT).

IOX RTST is used for test purposes only and does not generate interrupt and busy signals.

The instruction is only active when the interface is set in test mode by the following instructions:

```
SAA 10
IOX WCWD
```

To reset test mode, the following instructions must be used:

```
SAA 0
IOX WCWD        % Reset test mode bit
SAA 20
IOX WCWD        % Device clear
```

III.3.4.2.8 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:

Bit 0-7: Not used.

Bit 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

B) In test mode:

Bit 0-7: Not used

Bit 8-15: Test byte

III.3.5 *SPECIFICATIONS OF LINE PRINTER INTERFACE FOR CDC FOR NORD-10/S*

Standard device number 0430 $(0430\text{-}0433)_8$.
Number of device number 4.
Standard interrupt level 10_{10}.
Standard ident number 3.

Write Control Word IOX Device Number + 3:

Bit 0	Enable interrrupt on ready for transfer
Bit 1	Enable interrupt on error
Bit 2	Activate device (print character now in buffer)
Bit 3	Test
Bit 4	Device and interface clear
Bit 5-15	Not used

Read Status Word IOX Device Number + 2:

Bit 0	Interrupt enabled on ready
Bit 1	Interrupt enabled on error
Bit 2	Not used
Bit 3	Ready for transfer
Bit 4	Error, bit 5 or 6 set
Bit 5	Line printer not ready
Bit 6	Load image request, used by a new type
Bit 7-10	Not used
Bit 11	Line ready. Mostly used for test of line printer.
Bit 12	Illegal character in buffer
Bit 12-15	Not used

Write Data Word IOX Device Number + 1:

Write a data word in the buffer register.

If CR is removed to speed up the line printer, CR will be an illegal character. An illegal character will be returned by the interface because after an "act", the "act" will be ignored and the interface will remain ready for transfer. So special attention should be necessary by software.

The "characters" 0140-0177 are used for line printer control.

0020-0027	VFU channels	Top of FORM (FF)
020	VFU channel 1	Bottom of FORM
021	VFU channel 2	
022	VFU channel 3	
030-037	Line counter control	
030	Suppress space (CR)	0 Line feed
031	Single space (LF)	1 Line feed
032	Double space	2 Line feed

Standard for CDC line printer is 2 bits (4 channels) VFU and 2 bits (up to 3 line feeds) line counter.

Read Data Word IOX DEVN:

Bit 3 is set in control word. The interface may be tested. You will read back the data transformed for the line printer. The control bit is read as bit 9. Bits 6, 7 and 8 are only read back. They are not transformed in any way.

Note: The interface is *not* NORD-1 compatible. The VFU channels are moved from 140-157 to 20-27 and line counter control is removed from 160-177 to 30-37. 140-177 are reserved for extension.

III.3.6 *SPECIFICATION OF CARD READER BUFFER (DOCUMENTATION) FOR NORD-10/S*

Standard device number 0420 $(0420\text{-}0423)_8$.
Number of device number 4.
Interrupt level 12_{10}.
Ident number 3.

Write Control Word IOX Device +3:

Bit 0	Enable interrupt on ready for transfer
Bit 1	Enable interrupt on error
Bit 2	Activate.): Feed one card. Clear end of card.
Bit 3	Test
Bit 4	Device clear. Clear interrupt flip-flop, overrun flip-flop. Continue feed flip-flop and set end of card.
Bit 5-8	Not used
Bit 9	Continuous feed): feed next card immediately.
Bit 10-15	Not used.

Read Status Word IOX Device + 2:

Bit 0	Interrupt enabled on ready for transfer
Bit 1	Interrupt enabled on error
Bit 2	Card reader active. Signal from card reader.
Bit 3	Ready for transfer): a column may be read. This bit is turned off by "read data" IOX DEV.
Bit 4	Bits 5-9 set, error
Bit 5	Hopper check error set from card reader.
Bit 6	Light or dark error from card reader.
Bit 7	Motion check error from card reader.
Bit 8	Overrun, one column was lost because it was not read before the next column was strobed into buffer. Cleared by MASTER CLEAR.
Bit 9	End of card.
Bit 10-15	Not used.

Be aware that the card reader sends hopper check while reading the last card.

Write Data IOX Device + 1:

Only used for testing the interface without card reader.

Read Data IOX Device:

Read last column.

TEST:

The interface may be tested without card reader. When bit 3 in the control word is set to one, the interface is in "test mode".

An IOX Device + 1 will simulate a column read by the interface, set ready for transfer and increment the data register.

An "end of card" is simulated by IOX Device + 1 and bit 5 is set to one.

Programming Example:

(Without Interrupt)

```
CRDEV  = 420
RD     = 0
RS     = 2
RC     = 3

)FILL
%
%                       Error bit set
%
CRERT,                  BSKP 110 DA ZRO; JMP CREOC
%                       Error status in A register
                        Exit
%
%
CRER2,
%                       Error to many or few columns read
                        Exit
CREOC,                  JXZ * +2; JMP CRER2
                        LDX (CBUF; EXIT AD1
)FILL
CBUF,                   0
CPUF + 120/0
```

III.3.7 *NORD-10/S DISK PROGRAMMING SPECIFICATIONS*

Disk Device Register Address:

The codes below are relevant for disk system I. Each disk system may consist of 4 disk units. For disk system II and 10_8 to the specified codes.

Read Core Address

```
IOX 500
```

Load Core Address

```
IOX 501
```

Read Sector Counter

```
IOX 502
```

Load Block Address

```
IOX 503
```

Read Status Register

IOX 504

Load Control Word

IOX 505

Seek Instruction

IOX 506

Load Word Count Register

IOX 507

The minimum number of words to be transferred is one sector, i.e., 200_8 words, the maximum number of words is one track, i.e., 25 sectors.

Read Block Address

This instruction is implemented for maintenance purposes only. By first loading a control word with bit 3 (test mode), the instruction

IOX 506

will return the previously loaded block address to the A register.

Control Word

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

Unit Select Code:

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

Device Operation Code:

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

To format a disk, the key for formatting has to be turned on, Write Transfer, and Write Format must be specified.

Status Word

Bit 0	Ready for transfer, interrupt enabled
Bit 1	Error interrupt enabled
Bit 2	Device active
Bit 3	Device ready for transfer
Bit 4	Inclusive OR of errors (status bits 5-11)
Bit 5	Write protect violate
Bit 6	Time out
Bit 7	Hardware error
Bit 8	Address mismatch
Bit 9	Parity error
Bit 10	Compare error
Bit 11	Missing clock error
Bit 12	Transfer complete
Bit 13	Transfer on
Bit 14	On cylinder
Bit 15	Bit 15 loaded by previous control word

Interrupt

The disk interrupt level is 11 and the ident number for the first disk system is 1.

III.3.8 *NORD-10/S DRUM PROGRAMMING SPECIFICATIONS*

Drum Device Register Addresses:

The codes below are relevant for drum unit 0. For several drums connected to one system, the codes for each drum system are found by adding 10_8.N to those specified. N + number of drums (0, 1, 2, 3, . . .).

Read Core Address

IOX 540

Load Core Address

IOX 541

Read Sector Counter

IOX 542

Load Block Address

IOX 543

Read Status Register

IOX 544

Load Control Register

IOX 545

Load Word Count Register

IOX 547

The minimum number of words to be transferred to one sector of 100 words, the maximum number of words is one track, i.e., 2 sectors.

Read Block Address

This instruction is implemented for maintenance purposes only. By first loading a control word with bit 3 (test mode), the instruction

IOX 546

will return the previously loaded block address to the A register.

Control Word

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

Device Operation Code:

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

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 Bit transfer error
Bit 9 Parity error
Bit 10 Compare error
Bit 11 DMA channel error
Bit 12 Transfer complete
Bit 13 Transfer on
Bit 14 Not assigned
Bit 15 Bit 15 loaded by previous control word

Interrupt

The drum interrupt level is 11 and the ident number for the first drum is 2.

III.3.9 *PROGRAMMING SPECIFICATIONS FOR HEWLETT PACKARD MAG. TAPE CONTROLLER*

Mag. tape device number 520-527.

	IOX
Read Core Address	520
Load Core Address	521
Read Status	524
Load Control	525
Read Bar (test)	526
Load Word Count	527
Load Bar (test)	523

Read Status:

Bit 0	Ready interrupt enabled (cleared by the interrupt)
Bit 1	Error interrupt enabled (cleared by the interrupt)
Bit 2	Device active
Bit 3	Device ready for transfer
Bit 4	Inclusive OR or error bit (6, 9, 10, 11 and 12) or if a reverse command is attempted when the unit is at load point.
Bit 5	Write enable ring present
Bit 6	LRC error
Bit 7	EOF detected
Bit 8	Load point (this status remains also after the first forward command after load point is detected)
Bit 9	EOT detected
Bit 10	Parity error
Bit 11	DMA error
Bit 12	Overflow in read
Bit 13	Density select 1 = 800 bpi, 0 = 556 or 200 bpi
Bit 14	Mag. tape unit ready (selected, on-line and not rewinding)
Bit 15	Bit 15 loaded by previous control card

Load Control:

Bit 0	Enable interrupt on device ready for transfer
Bit 1	Enable interrupt on errors
Bit 2	Activate device
Bit 3	Test mode
Bit 4	Device clear
Bit 5	Address bit 16
Bit 6	Address bit 17
Bit 7	Read odd number of character
Bit 8	Even parity (only to be used while writing/reading ASCII information on 7 tracks)
Bit 9	Unit select (up to 4 units)

Bit 10 Unit select (up to 4 units)
Bit 11-14 Device operation code
Bit 15

Device Operation Code:

Bit: 14	13	12	11		
0	0	0	0	Read one record	M0
0	0	0	1	Write one record	M1
0	0	1	0	Advance to EOF	M2
0	0	1	1	Reverse to EOF	M3
0	1	0	0	Write EOF	M4
0	1	0	1	Rewind	M5
0	1	1	0	Erase gap (4")	M6
0	1	1	1	Backspace one record	M7

Interrupt:

The MT interrupt level is 11 and the ident number for the first MT system is 3.

III.3.10 *PROGRAM SPECIFICATIONS FOR MAG. TAPE FORMATTER (TANDBERG)*

Mag. tape device number 520-527.

	IOX
Read Core Address	520
Load Core Address	521
Read Modus	522
Load Modus Word	523
Read Status	524
Load Control	525
In Test, Read Modus	526
Load Word Counter	527

Read Status:

Bit 0-3 Standard
Bit 4 Error inclusive OR of bits 5, 6, 7, 8, 9, 11, 12
Bit 5 Control or modus word error. Trying to write protected tape, reversing tape at load point, tape unit not on-line, etc. Action inhibited.

Bit 6	Bad data block. An error detected
Bit 7	End of file detected
Bit 8	The search character is detected
Bit 9	End of tape detected
Bit 10	Word counter not zero
Bit 11	DMA error
Bit 12	Overflow (in read)
Bit 13	Tape busy or formatter busy
Bit 14	Formatter busy
Bit 15	Interrupt when formatter ready

Read Modus Register:

Bit 0	Tape on-line
Bit 1	Write enable ring present
Bit 2	Tape standing on load point
Bit 3	CRC error/fatal error
Bit 4	LRC error/soft error
Bit 5-8	Number of VRC errors. If all is one, 15 or more.
Bit 9-15	Not used – may be one or zero.

Load Modus Word:

Bits 0-8 Action code or bits 0-7 search word.

As action code:

Bit 0	Density select (556 bpi)
Bit 1	Density select (220 bpi)
Bit 2	Reverse
Bit 3	Write
Bit 4	Space
Bit 5	File mark
Bit 6	High speed
Bit 7	Parity (only 7 track)
Bit 8	Threshold

Bits 2-6 are combined to give a specific action.

	2	3	4	5	6
Write one block	0	1	0	0	0
Write file mark	0	1	0	1	0
Erase gap	0	1	1	0	0
Read one block	0	0	0	0	0
Read one block rev.	1	0	0	0	0
Space forward one block	0	0	1	0	0
etc.					

Space reverse one file	1	0	1	1	1

The action is initiated by loading the correct control word.

Bit 0-7 Search word
Bit 12-14 Unit number
Bit 15 Odd number of characters to be read

Load Control Word:

Bit 0-6: Standard
Bit 7-10 Not used
Bit 11-12:

11	12	
0	0	Together with act the bits 0-8 in modus specify an action to be done
1	0	Strobe search word into formatter
0	1	Rewind
1	1	Rewind and unload

Bit 13-14 Should always be zero
Bit 15 Give interrupt when the formatter is ready (not waiting for the tape to be ready).

Operation:

First mode is loaded, then Control Word is loaded with "act" and bit 11 and 12 zero.

For search:

First search word is placed in modus (with unit number) and Control Word written.

Then a "position on block" action is executed.

Each time unit number is changed, a MASTER CLEAR should be given in advance.

III.3.11 *NORD-10/S ASYNCHRONOUS MODEM PROGRAMMING SPECIFICATIONS*

Asynchronous Modem Addresses:

The codes below are relevant for the first asynchronous modem (modem number 0). The codes for the first eight modems are found by adding 10_8.N to the codes given. N - modem number (0, 1, 2, ..., 7). For the next eight modems, the codes are found by adding $(1000 + 10(N - 10))_8$. N = modem number $(10, 11, 12, ..., 17)_8$.

Input Channel (Interrupt level 12):

Read Data Register

IOX 200

The number of data bits read into the A register is 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 202

Write Control Register

IOX 203

Output Channel (Interrupt level 10):

Connect Data Set to Line

IOX 204

This IOX instruction will connect the modem to the line in the same way as an external call, dependent on the input channel control register.

Write Data Register

IOX 205

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 206

Write Control Register

IOX 207

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 responding to level 10.

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
Bit 8	Carrier error
Bit 9	Carrier error enabled
Bit 10	Frequency status
Bit 11	Carrier missing
Bit 12	Connect missing
Bit 13	Data Set Ready missing
Bit 14	Not used
Bit 15	Half duplex

Notes: Additional explation to status bits.

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 generating/checking mode.

Bit 7: Overrun means that at least one character is overwritten while input is active.

Bit 8: Carrier error means that the line signal (carrier) is missing in a period where input control (and status) bit 9, carrier error enable, is set to 1.

Bit 9: See "Control Register", bit 9 for the meaning of this bit.

Bit 10: This is the status of the control signal frequency select.

Bit 11: Carrier missing gives the status of receive line signal detector, or carrier on the line.

0 indicates carrier present
1 indicates carrier missing

Bit 12: Connect missing gives the status of Connect Data Set to line.

0 indicates that connect is ON
1 indicates that connect is OFF

Bit 13: Data Set Ready missing gives the status of the Data Set Ready signal from the modem.

0 indicates that the modem is ready
1 indicates that the modem is not ready

Bit 15: 0 indicates that the terminal is operated in full duplex mode
1 indicates that the terminal is operated in half duplex mode

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 5-6	Not used
Bit 7	6 second time out disable
Bit 8	2 bits time out disable
Bit 9	Carrier error enable
Bit 10	Transmission frequency
Bit 11	Character length
Bit 12	Character length
Bit 13	Number of stop bits
Bit 14	Parity generation/checking
Bit 15	Half duplex

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. The modem will not be connected to the line if bit 1 in the input control and status register is 0.

Bit 3: Test mode will loop transmit data back to received data, if the other terminal is connected to the line, transmitted data will also be transferred to this terminal.

Bit 4: Gives a clear pulse to the buffer card. Disconnected modem from this line. After a DEVICE CLEAR, the buffer card must be initialized.

Bit 7: 6 second time out disable.

If no Carrier is received within 6 seconds after a Connect Data Set to line signal is given to the modem, a 6 second timing circuit will normally turn off the Connect signal. If a 1 is written into bit 7 of the input control register, the Connect signal can only be turned off by clearing the input activate device bit. Bit 7 is set to 0 by MASTER CLEAR and DEVICE CLEAR.

Bit 8: 2 bits time out disable.

This control signal has only meaning when the buffer is used in half duplex mode (control bit 15). In that case, it will cause the Request to Send signal to be turned off at the time of two bits after the last character is transmitted to the line. If a 1 is written into this control bit, the Request to Send signal can be turned off by writting 0 into either output channel control register bit 2 (which also turns off Connect Data Set to Line), or by giving a MASTER CLEAR or DEVICE CLEAR. In full duplex operation, Request to Send and Connect Data Set to Line are identical. Bit 8 is set to 0 by MASTER CLEAR and DEVICE CLEAR.

Bit 9: When set to 1, this signal will cause an error condition if input status bit 11 is 1. If IC means input channel control register, an IS means input channel status register, then

$$IS8 = IS11.IC9$$

and

$$IS4 = IS5 + IS6 + IS7 + IS8$$

Bit 9 is set to 0 by MASTER CLEAR and DEVICE CLEAR.

Note: This can give you a temporary interrupt, which can give an interrupt on level 14 is not handled in time.

Bit 10: Selects transmission frequency when used with full duplex modem.

0 selects frequency for called station
1 selects frequency for calling station

MASTER CLEAR and DEVICE CLEAR set the control bit to 0.

Bits
11-12: The content of these bits give the following character lengths, both for the input channel and the output channel:

Bit: 11 12

0	0	8 bits
1	0	7 bits
0	1	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: 0 selects full duplex operation. In this case, Request to Send is identical to Connect Data Set to Line.

1 selects half duplex operation. In this case, Request to Send is blocked by Carrier on the line.

MASTER CLEAR and DEVICE CLEAR set the control bit to 0.

Output Channel:

Status Register

Bit 0	Ready for transfer interrupt enabled
Bit 1	Not used
Bit 2	Device active
Bit 3	Device ready for transfer
Bit 4-9	Not used
Bit 10	Frequency status
Bit 11	Carrier missing
Bit 12	Request to Send missing
Bit 13	Ready for Sending missing
Bit 14	Not used
Bit 15	Half duplex

Notes:

Bit 2: This status bit will be 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
10-11: Same as for input channel.

Bit 12: Gives the status of the Request to Send control signal to the modem.

0 means that Request to Send is ON.
1 means that Request to Send is OFF.

Bit 13: Gives the status of the Ready for Sending status signal from the modem.

0 means Ready for Sending.
1 means *not* Ready for Sending.

Bit 15: Same as for input channel.

Control Register:

Bit 0 Enable interrupt on device ready for transfer
Bit 1 Not used
Bit 2 Activate device
Bit 3-15 Not used

Notes:

Bit 2: When operated in half duplex mode a 1 in bit 2 gives Request to Send if there is no Carrier on the line. If there is a Carrier on the line, Request to Send will go On when the Carrier goes OFF.

Control and Status Words:

Input:

Bit	Status	Control
0	RFT en	Enable RFT
1	ERR en	Enable ERR
2	Device act	Activate
3	Device RFT	Test
4	ERR OR	Device clear
5	Framing	
6	Parity	
7	Overrun	6 seconds time out disable
8	Carrier error	2 bits time out disable
9	Carrier error en	Carrier error enable
10	Frq. stat.	Frequency select
11	Carrier missing	Character length
12	Connect missing	Character length
13	DSR missing	Stop bits
14		Parity generation/check
15	Half duplex	Half duplex

Output:

Bit	Status	Control
0	RFT en	Enable RFT
1		
2	Dev. act.	Activate
3	Dev. RFT	
4		
5		
6		
7		
8		
9		
10	Frq. stat	
11	Carrier missing	
12	RQTS missing	
13	RFS missing	
14		
15	Half duplex	

III.3.12 *NORD-10/S SYNCHRONOUS MODEM PROGRAMMING SPECIFICATIONS*

Synchronous Modem Addresses:

The codes below are relevant for the first synchronous modem (Modem No. 0). The codes for the first eight modems are found by adding 10_8.N to the codes given. N = modem number (0, 1, 2, ..., 7). For the next eight modems the codes are found by adding $(1000 + 10\ (N - 10))_8$. N = modem number $(10, 11, 12, ..., 17)_8$.

Input Channel (Interrupt level 12):

Read Data Register

```
IOX 100
```

The number of data bits read into the A register is specified by bits 11 and 12 in the input channel control register. The received character is right justified (from bit 0 and upwards). Reset Device Ready for Transfer.

Write input and/or output SYN character register.

```
IOX 101
```

The eight least significant bits in A register are written into the SYN character registers specifed by A register bits 8 and 9.

A 1 in bit 8 prevents writing into the input SYN character register. A 1 in bit 9 prevents writing into the output SYN character register.

The program sequence

```
LDA (26          % Octal
IOX  101
```

makes the contents of both input and output SYN character registers 26_8.

The program sequence

```
LDA (777
IOX  101
```

makes all bits in the output SYN character register 1.

Read Status Register

IOX 102

Write Control Register

IOX 103

Output Channel (Interrupt level 10):

Write Data Register

IOX 105

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.

Reset device ready for transfer.

Read Status Register

IOX 106

Write Control Register

IOX 107

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 responding to level 10.

Serviced ident resets proper interrupt enabling flip-flops. The ident code for synchronous modem 0 is 4.

Input Channel:

Status Register

Bit 0 Ready for transfer interrupt enabled, reset
Bit 1 Error interrupt enabled
Bit 2 Device active
Bit 3 Device ready for transfer
Bit 4 Inclusive OR of errors

Bit 5 SYN character received
Bit 6 Parity error
Bit 7 Overrun
Bit 8 Carrier error
Bit 9 Carrier error enabled
Bit 10 Request to Send missing
Bit 11 Carrier missing
Bit 12 Connect missing
Bit 13 Data Set Ready missing
Bit 14 Ready for Sending missing
Bit 15 Half duplex

Notes: Additional explanation of status bits follows.

Bit 5: SYN character received is one of the receiver error bits but can be separately enabled/disabled. If SYN character error is enabled, bit 5 will be set to 1 each time a SYN character is received and be a 1 until disabled or the first character unequal to SYN is received.

Bit 6: Parity error means that a parity error has occurred while working in parity generating/checking mode.

Bit 7: Overrun means that at least one character is overwritten while input is active.

Bit 8: Carrier error means that the line signal (Carrier) is missing in a period where input control (and status) bit 9, Carrier error enable, is set to 1.

Bit 9: See section "Control Register" for the meaning of this bit.

Bit 10: Gives the status of the Request to Send signal.

0 means that Request to Send is ON.
1 means that Request to Send is OFF.

Request to Send is turned ON in full duplex mode by activating output, and in half duplex mode when output is activated and no carrier is present. Request to Send is turned OFF either programmable (writing 0 into output control register bit 2) or when connect data set to line is turned OFF.

Bit 11: Carrier missing gives the status of receive line signal detector or carrier on the line.

0 indicates Carrier present.
1 indicates Carrier missing.

Bit 12: Connect missing gives the status of Connect Data set to Line.

0 indicates that Connect is ON.
1 indicates that Connect is OFF.

The Connect signal is set by activating input or by a calling and is reset by a 6 second time out, MASTER CLEAR and DEVICE CLEAR.

Bit 13: Data Set Ready missing gives the status of the Data Set Ready signal from the modem.

0 indicates that the modem is ready.
1 indicates that the modem is not ready.

Bit 14: Gives the status of the Ready for Sending signal from the modem.

0 means Ready for Sending.
1 means NOT Ready for Sending.

Bit 15: 0 indicates that the terminal is operated in full duplex mode.
1 indicates that the terminal is operated in half duplex mode.

Control Register

Bit 0	Enable interrupt on device ready for transfer
Bit 1	Enable interrupt on errors
Bit 2	Activate device, Connect Data set to Line
Bit 3	Test mode
Bit 4	Device clear
Bit 5	Not used
Bit 6	Not used
Bit 7	6 seconds time out disable
Bit 8	SYN character error enable
Bit 9	Carrier error enable
Bit 10	Receiver reset
Bit 11	Character length
Bit 12	Character length
Bit 13	Odd/even parity
Bit 14	Parity generation/checking
Bit 15	Half duplex

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 also sets the Connect Data set to Line signal.

Bit 3: Test mode will loop transmit data back to received data at a rate of 19200 bits.

Bit 4: Gives a clear pulse to the buffer car. Disconnects modem from the line. After a DEVICE CLEAR, the buffer card must be initialized.

Bit 7: 6 seconds time out disable. If no Carrier is received or programmed Request to Send is given within 6 seconds after a Connect Data set to Line signal is given to the modem, a 6 seconds timing circuit will normally turn off the Connect signal. If a 1 is written into bit 7 of the input control register, the Connect signal can only be turned off by clearing the input activate device bit. Bit 7 is set to 0 by MASTER CLEAR and DEVICE CLEAR.

Bit 8: SYN character error enable gives a possibility to detect received SYN characters without bit pattern recognition in software. If enabled, a received SYN character gives error interrupt. If bit 8 is set to 0, received SYN characters will *not* give error interrupts. MASTER CLEAR and DEVICE CLEAR disable SYN character error.

Bit 9: When set to 1, this signal will cause an error condition if input status bit 11 is 1. If IC means input channel control register, an IS means input channel status register, then

$$IS8 = IS11 \cdot IC9$$

and

$$IS4 = IS5 + IS6 + IS7 + IS8$$

Bit 9 is set to 0 by MASTER CLEAR and DEVICE CLEAR.

Note: This can give you a temporary interrupt, which can give you an interrupt on level 14 if not handled in time.

Bit 10: A 1 in this bit gives a Receiver Reset pulse. This causes the receiver to be set in a SYN character searching mode. If Ready for transfer and/or any of the error conditions except carrier error are set, they will be reset.

In the search mode the serial received data bit stream is examined on a bit by bit basis until a SYN character is found. A SYN character is found, by definition, when the contents of the receiver SYN character register and their receiver shift register are identical. This character is then loaded into the receiver buffer register and the receiver is set into the character mode. In this mode each character received is loaded into the receiver buffer register, and at the same time device ready for transfer is given.

Bits
11-12: The contents of these bits give the following character length, both for the input channel and the output channel.

Bit: 11 12

0	0	8 bits
1	0	7 bits
0	1	6 bits
1	1	5 bits

If bit 14 is a 1, a parity bit is *added* to the number given in this table on the transmission side, and *checked and removed* a parity bit on the receiving side.

Bit 13: If the interface is in a parity generating/checking mode (bit 14), the

0 is used for odd parity, and
1 is used for even parity.

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 odd or even parity bit to the character on the ouput channel, and give an error indication if the received character has wrong parity.

Bit 15: 0 selects full duplex operation.
1 selects half duplex operation.

MASTER CLEAR and DEVICE CLEAR set the bit to 0.

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 SYN character transmitted
Bit 6-9 Not used
Bit 10-15 Identical to input channel status register

Notes:

Bit 2: This status bit will be 1 as long as the modem is ready to transmit 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: These bits are identical, and are set to 1 if the transmitted character is taken from the transmitter SYN character register. This is done when the transmitter buffer register is empty. If output error interrupt is enabled, this will cause an error interrupt on level 10.

Control Register

Bit 0 Enable interrupt on device ready for transfer
Bit 1 Enable interrupt on errors
Bit 2 Activate device, Request to Send
Bit 3-15 Not used

Note:

Bit 2: When operated in half duplex mode a 1 in bit 2 gives Request to Send if there is no Carrier on the line. If there is a Carrier on the line, Request to Send will go ON when the Carrier goes OFF. In full duplex mode, Request to Send is independent of Carrier.

Control and Status Word:

Input:

Bit	Status	Control
0	RFT en	Enable RFT
1	RFT en	Enable ERR
2	Device act	Activate, Connect
3	Device RFT	Test
4	ERR OR	Device clear
5	SYN received	
6	Parity	
7	Overrun	6 seconds time out disable
8	Carrier error	SYN err en
9	Carrier error en	Carrier error en
10	RQTS missing	Receiver reset
11	Carrier missing	Character length
12	Connect missing	Character length
13	DST missing	Odd/even parity
14	RFS missing	Parity generation/check
15	Half duplex	Half duplex

Output:

Bit	Status	Control
0	RFT en	Enable RFT
1	ERR en	Enable ERR
2	Device act	Activate, RQTS
3	Device RFT	
4	ERR OR	
5	SYN transmitted	
6		
7		
8		
9		
10	RQTS missing	
11	Carrier missing	
12	Connect missing	
13	DSR missing	
14	RFS missing	
15	Half duplex	

III.4 SPECIFICATIONS

III.4.1 *PROCESSOR*

Microprocessor cycle time:	260 ns
16 bit parallel processor, 32 bit parallel arithmetic during floating operation	
CACHE memory size:	1K/25 bits
Paging overhead with CACHE:	0%
Paging overhead without CACHE:	10%
Access time for the Page Index Table:	150 ns
Access time for the CACHE memory:	150 ns

III.4.2 *MEMORY*

Maximum virtual memory address space:	128 Kbytes
Maximum physical memory address space:	512 Kbytes
Access time for Local Memory:	380 ns
Cycle time for Local Memory:	400 ns
Access time for Multiport Memory:	700 ns
Cycle time for Multiport Memory:	450 ns
Parity:	2 bits – one per byte
Error Checking and Correcting Memory:	21 bits, single bit detection and correction (40% of all double bit errors detected)
Battery stand by power for memory:	Maximum 30 minutes

III.4.3 *INTERRUPT SYSTEM*

16 priority interrupt levels each with 8 registers	
Context block switching time:	1 μs
External Interrupt Identification time:	2 μs minimum 2.3 μs typical

III.4.4 *I/O SYSTEM*

Maximum DMA rate/channel to Multiport Memory:	1.6 Mbytes
Maximum transfer rate for Multiport Memory Channel:	2.8 Mbytes
Maximum DMA latency for highest priority devices: refresh + CPU + channel	2.5 μs

III.4.5 *PHYSICAL*

Dimensions:

Height:	1606 mm
Width:	582 mm
Depth:	730 mm
Power:	220V AC ± 10% 50 Hz ± 2 Hz 2.7 Amp @220V*
Cooling:	Forced cooling
Operating Ambient Temperature:	0 - 55° C
Operating Humidity:	10 - 90%, non-condensing

---

* Applies to a NORD-10 CPU with: Memory Management System, CACHE, Large disk interface, Bus Receiver, Bus Brancher and 128 Kbytes of MOS Memory.

APPENDIX A

LIST OF ABBREVIATIONS

!	START PROGRAM IN MAIN MEMORY
$	OCTAL LOAD
&	BINARY LOAD
( )	CONTENT OF .....
,B	B-RELATIVE ADDRESSING
,X	X-RELATIVE ADDRESSING
/	EXAMINE REGISTER OR MEMORY CEL
@	RESTART MOPC,CLEAR PIE
A	AMPER
A	ACCUMULATOR REGISTER
AC-REGISTER	ARITHMETIC OUTPUT HOLDING REGISTER
ACTL	ACTIVE LEVEL (DECODED)
ADR	ADDRESS
ALD	AUTOMATIC LOAD DESCRIPTOR
ALU	ARITHMETIC LOGIC UNIT
APT	ALTERNATIVE PAGE TABLE
ASCII	AMERICAN STANDARD CODE FOR INFORMATION EXCHANGE
ATS	ALTERNATIVE TABLE SELECT
B	BANK (SPECYFY 64 K BANK)
B	BASE REGISTER
BA	MEMORY ADDRESS BUS
BD	MEMORY DATA BUS
BYTE	BYTE (BYTE=8 BITS=1/2 CPU WORD)
C	CARRY INDICATOR
C	CAPASITANCE
CO-4	CONTROL CODE (5 BITS)
CAS	COLUMN ADDRESS STROBE
CAR	INSTRUCTION REGISTER
CCLR	CACHE CLEAR
CILR	CACHE INHIBIT LIMIT REGISTER
CP	CURRENT PROGRAM COUNTER
CPU	CENTRAL PROCESSING UNIT
CPN	CACHE PAGE NUMBER
CR	CARRIAGE RETURN
CSR	CACHE STATUS REGISTER
CUW	CONFLICTING USE WRITEBACK
CW	CONTROL WORD
D	D-REGISTER (DOUBLE LENGTH)
DIN	DATA IN
DIP	DISPLACEMENT WITHIN PAGE
DMA	DIRECT MEMORY ACCESS
E	EXTENTION
E0-15	SINGLE DATA ERRORS
ECCR	ERROR CORRECTION CONTROL REGISTER
EC	SINGLE CODE ERRORS
ED	EXTERNAL DEVICE
EP	ENTRY POINT
EPG	ENTRY POINT GENERATOR
EXM	EXAMINE

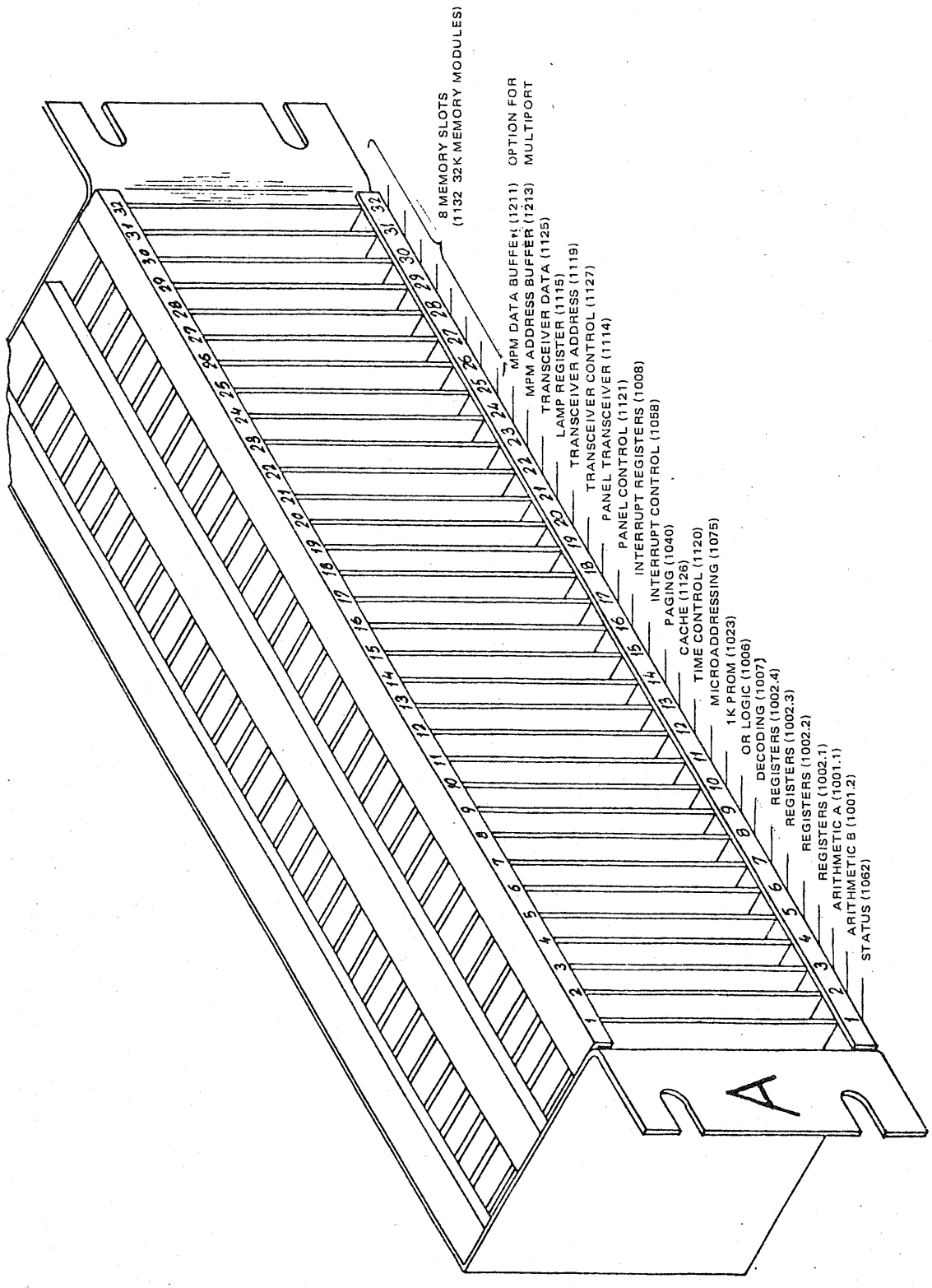
FETCH	FETCH NEXT INSTRUCTION
FF	FETCH FAULT
FIFO	FIRST IN-FIRST OUT
FPM	FETCH PERMITTED
H-REGISTER	DATA HOLDING REGISTER
I	INTERNAL REGISTER
I	INDIRECT ADDRESSING
I/O	INPUT/OUTPUT
IB	CPU DATA BUS
II	ILLEGAL INSTRUCTION
IIC	INTERNAL INTERRUPT CODE
IID	INTERNAL INTERRUPT DETECT
IIE	INTERNAL INTERRUPT ENABLE
IND	INDIRECT ADDRESSING
IONI	INTERRUPT SYSTEM ON INDICATOR
IOX	INPUT/OUTPUT TIMEOUT INTERRUPT
IOX	INPUT/OUTPUT INSTRUCTION
ION	INTERUPT SYSTEM ACTIVE
IOXE	I/O STROBE SIGNAL
IR	INSTRUCTION REGISTER
K	KILO(WORDS)=1024 WORWS
L	LINK REGISTER
LF	LINE FEED
LMP	OPERATOR'S PANEL LAMP REGISTER
LRU	LEAST RECENTLY USED
M	SHIFT LINK
M	MASS STORAGE
MA	MILLI AMPER (=1/1000 A)
MA	MAIN I/O ADDRESS BUS
MC	MONITOR CALL
MC	MASTER CLEAR
MD	MAIN I/O DATA BUS
ME	MULTIPLE ERRORS
MIR	MICRO INSTRUCTION REGISTER
MISC	MISCELLANEOUS REGISTER
MMS	MEMORY MANAGEMENT SYSTEM
MOPC	MICRO PROGRAMMED OPERATOR'S COMMUNICATION
MOR	MEMORY OUT OF RANGE
MOS	METAL OXIDE SEMICONDUCTOR
MPC	MICRO PROGRAM COUNTER
MPV	MEMORY PROTECT VIOLATION
MPX	MULTIPLEX
MPM	MULTIPORT MEMORY
MR	MEMORY ADDRESS BUS (18-BITS)
MR	CPU ADDRESS BUS
MTBF	MEAN TIME BETWEEN FAILURE
MW	MEGA-WORDS (=1 000 000 WORDS)
NA	NOT ASSIGNED
NS	NANOSECONDS (1/1 000 000 000 SECOND)
NU	NOT USED
O	STATIC OVERFLOW INDICATOR
O	OCTAL
OK	OK (NO ACCESS VIOLATION)
OPR	OPERATOR'S PANEL SWICH REGISTER

P	PROGRAM COUNTER
P0-3	PORT 0-3 (MULTIPORT MEMORY)
PA	PHYSICAL ADDRESS
PANS	OPERATOR PANEL STATUS REGISTER
PANC	OPERATOR PANEL CONTROL REGISTER
PCR	PAGING CONTROL REGISTER
PEA	MEMORY ERROR ADDRESS REGISTER
PES	MEMORY ERROR STATUS REGISTER
PF	PAGE FAULT
PGU	PAGE USED
PGS	PAGING STATUS REGISTER
PGU	PAGE USED
PI	PREVELEGED INSTRUCTION
PIE	PRIORITY INTERRUPT ENABLE
PID	PRIORITY INTERRUPT DETECT
PIK	PRIOPITY INTERRUPT CODE
PIO	PROGRAMMED INPUT/OUTPUT
PIT	PAGE TABLE
PIO	PROGRAMMED INPUT OUTPUT
PL	PROGRAM LEVEL INDICATOR
PL	PROGRAM LEVEL
PL	PROGRAM LEVEL (INDICATOR)
PL	PROGRAMLEVEL
PM	PAGE INDEX MODUS
PM	PERMIT FLAGS
PONI	PAGING ON INDICATOR
POW	POWER FAILURE INTERRUPT
POH	POWER ON HOURS
PSR	PAGING STATUS REGISTER
PTY	PARITY ERROR INTERRUPT
PTS	PAGE TABLE SELECT FLAG
PVL	PREVIOUS LEVEL INDICATOR
Q	DYNAMIC OVERFLOW INDICATOR
Q	TRANSISTOR
R	REGISTER (GENERAL)
R	RESTART
R-REGISTER	VIRTUAL ADDRESS REGISTER
R0-3	RING # 0-3
RADDR	READ ADDRESS SEQUENCE (IND ADDR)
RAM	RANDOM ACCESS MEMORY
RAS	ROW ADDRESS STROBE
RADDR	READ INDIRECT ADDRESS
RNI	READ NEXT INSTRUCTION SEQUENCE
ROM	READ ONLY MEMORY
ROP	READ OPERAND SEQUENCE
RPM	READ PERMITTED
RT	REAL TIME (PROGRAM)
SB	STANDBY
SP	SAVED PROGRAM COUNTER
STO	STORE OPERAND SEQUENCE
STS	STATUS REGISTER
STO	STORE OPERAND

T	TEMPORARY REGISTER
TG	FLOATING ROUNDING INDICATOR
TS	TABLE SELECT
TTY	TELETYPE
U	USED (VALID DATA)
V	VOLT
VA	VIRTUAL ADDRESS
WE	WRITE ENABLE
WIP	WRITTEN IN PAGE
WPM	WRITE PERMITTED
WS	WORKING SET
WT	WRITE THROUGH
X	POST INDEX REGISTER
Z	ERROR INDICATOR

APPENDIX B

NORD-10/S CARD ASSEMBLY



NORSK DATA A.S.
Lørenveien 57 - Postboks 163, Økern
OSLO 1

COMMENT AND EVALUATION SHEET

NORD-10/S – Functional Description
August 1977 ND-06.009.01

In order for this manual to develop to the point where it best suits your needs, we must have your comments, corrections, suggestions for additions, etc. Please write down your comments on this pre-addressed form and post it. Please be specific wherever possible.

FROM _____

_____

_____

– we make bits for the future

A/S NORSK DATA A, P.O. BOX 4, LINDERUD, 0212 OSLO 10 NORWAY PHONE 02 10 01 TELEX 18001