ND-110 Functional Description

ND-06.026.1 EN

ND-110 Functional Description

ND-06.026.1 EN

Preface

The product

The ND-110 Functional Description manual describes the architecture of the ND-110 computer series. The main building blocks and their functions are described.

The Reader

This manual is intended for all technical and maintenance personnel who wish to gain detailed information about the ND-110 computer.

The reader is assumed to have a general knowledge of digital techniques and computers. Some knowledge of ND-100 instruction set and assembly programming will also be helpful for parts of the manual.

The manual

This manual is intended to be read from beginning to end, since some sections of the manual assume knowledge of previous sections. The appendices are included chiefly for reference.

Related manuals

ND-06.029	ND-110 Instruction Set Describes the ND-110 from the programmer's point of view.
ND-06.017	ND-100 Bus Description Describes the bus signals used by ND-100 and ND-110 computers.
ND-60.096	MAC User Guide The assembly language manual for ND-100 and ND-110 computers.

Table of contents

List of figures

List of tables

CHAPTER 1 ND-110 ARCHITECTURE

CHAPTER 1 ND-110 ARCHITECTURE

ND-110 is a 16-bit general-purpose mini computer system in the ND-100 family of 16-bit computers from Norsk Data. It is general-purpose in the sense that it has both software and hardware available for most computer applications. The maximum address space is 16 Mwords (32 Mbytes). It is upwards compatible with NORD-10/S and ND-100 computers and runs the same operating system, SINTRAN III.

The ND-110 CPU is supplied in two versions that differ only in their performance.

- ND-110 Standard
- ND-110/CX

The ND-110 Standard has the same performance as the ND-100/CX. Dependent on application, the ND-110/CX CPU is from 1.5 to 3.5 times faster than ND-100/CX.

Both versions of the ND-110 have memory management, a new type of cache memory and the commercial extended instruction set as standard.

1.1 INSTRUCTION SET

Although the basic ND-110 word is 16 bits, the computer has a comprehensive instruction set which includes operations on:

- bits
- bytes
- single words (16 bits)
- double words (32 bits)
- triple words (48 bits)
- register blocks
- fixed or floating point numbers

Floating point instructions

The floating point instructions include add, subtract, multiply and divide. The standard 32-bit format has an accuracy of 23-bit (approximately 7 decimal digits). As an option, the ND-110 Standard and ND-110/CX may be equipped with a 48-bit floating point format which has 32-bit accuracy (approximately 10 decimal digits).

Commercial instructions

For efficient system control, specially tailored privileged instructions are included, such as loading and storage of the register blocks and inter-program level read/write operations. Other instructions perform binary-coded decimal (BCD) arithmetic. Together, these instructions comprise what is known as the commercial extended instruction set which is standard on all ND-110 CPUs (optional on ND-100).

New instructions for ND-110

In addition to all ND-100/CX instructions, ND-110 CPUs contain the following new instructions.

- TRA CS, TRR CS new instructions for reading and loading the control store. The old LWCS (load writeable control store) is still legal but performs no operation.
- TRR CILP Cache page inhibit of individual pages in cache.
- VERSN returns print version, microcode version, and installation number.

Writeable control store

To allow dynamic microprogramming, the microprogram control store is writeable (optional on ND-100). This allows programmers to load the control store with new microinstructions in order to extend the instruction set for special applications.

The ND-110 instruction set is described in the manual ND-110 Instruction Set, ND-06.029. Instructions that are new for the ND-110 CPU are also described in Chapter 8 (page 192).

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

The address arithmetic is implemented in hardware in the ND-110 whereas the ND-100 used microprogram. This gives the ND-110 an important speed advantage compared to its predecessor.

1.3 Bus structure

The main highway for addresses and data in the system is the ND-100 bus, a multiplexed address and data bus. All communication between ND-110 CPU card and the other cards in the system is provided by this bus.

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

1.4 The ND110 Computer

The ND-110 computer is delivered in a number of different configurations and performance. All versions are based on the ND-110 CPU, which is an improved version of the ND-100/CX. They execute the instruction set of the ND-100/CX with some extensions. Programs written for the ND-100 (all versions) and the NORD-10/S will run on the ND-110 without modifications.

Memory management, cache memory and the commercial extended instruction set are now standard on the ND-110.

1.4.1 ND-110 CONFIGURATION

Communication between ND-110 cards is achieved through an advanced high-speed bus called the ND-100 bus. The ND-100 bus is implemented as a printed backplane. The bus can be extended to any number of other ND-100 buses by a driver and a receiver card.

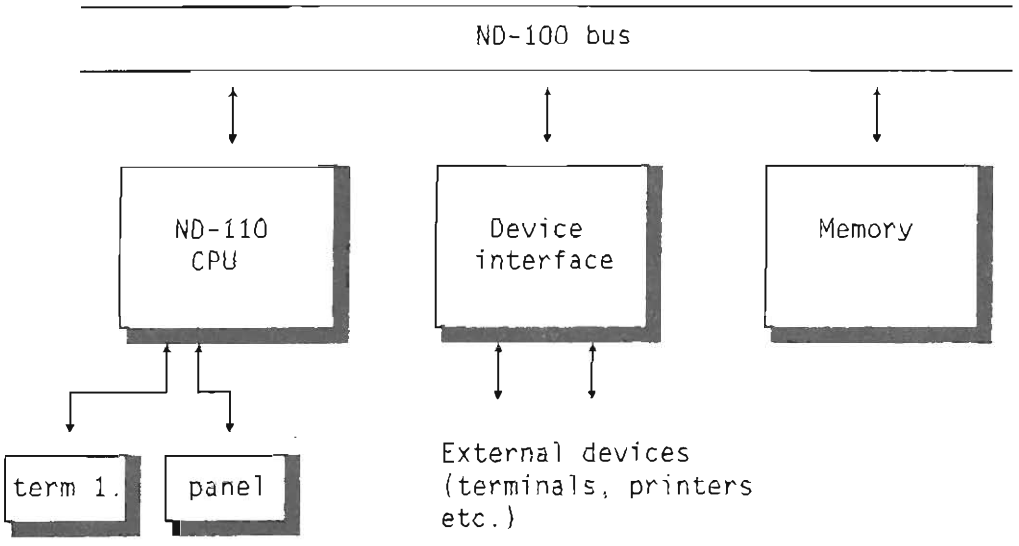


Figure 1. ND-100 bus Connection

1.4.2 CENTRAL PROCESSING UNIT

The Central Processing Unit (CPU) card contains:

- CPU
- real time clock
- terminal interface with switch selectable speeds, 50 - 9600 baud (bits per second)
- power fail and automatic restart
- memory management system
- cache memory
- panel interface

The central processor unit (CPU)

ND-110 CPU is a 16-bit parallel processor controlled by a microprogram. The following functions are implemented in microprogram:

- all instructions
- operator communication
- built in test routines
- bootstrap loaders
- program level change

Two performance versions

The ND-110 CPU is delivered in two versions; the ND-110 Standard which executes 0.32 Whetstone MIPS (the same as the ND-100/CX) and the faster ND-110/CX which executes 0.55 Whetstone MIPS.

Both versions of the ND-110 share the same (commercial extended) instruction set.

New cache technique

All versions of the ND-110 CPU include cache memory. A new cache technique is used, which integrates instruction decoding into the cache in a novel manner.

Program levels

The ND-110 has the same 16-level priority system as the ND-100. The 16 levels are usually referred to as program levels in this manual. Each level is assigned a complete set of working registers, and these registers are stored in the register file. A copy of the register set for the current level is located in a high-speed register set.

Memory Management

Memory management is now standard on all versions of the ND-110 CPU. Memory management provides:

- 64 Kword virtual address range for each user independent of physical memory capacity
- dynamic allocation/relocation of programs in memory
- memory protection

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

- paging system
- memory protection system

The paging system

"Pages" consist of blocks of 1 Kword (2048 bytes).

The paging system can work in two modes:

"Normal" This mode is compatible with the NORD-10/S paging system. This maps a 16-bit virtual address (describing a 64 Kword virtual memory) into a 19-bit physical address. In this mode the physical address space can be extended up to 512 Kword (1 Mbyte).

"Extended" This is now used by most programs written for ND-100 and ND-110 CPUs, The 16-bit virtual address is mapped into a 24-bit physical address. The CPU can then address 16 Mwords (32 Mbyte).

Sixteen page tables hold the physical page numbers assigned to active programs. These tables are located in high speed memory, reducing paging overhead to practically zero. The ND-110 can also be used with four page tables (normal mode only) for compatibility with NORD-10/S.

The memory protection system

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

- The page protection system
- The ring protection system

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

The ring protection system places each page and each program on one of four priority rings.

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

1.4.3 Memory System

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

- cache memory
- up to 16 Mwords local memory
- Memory channel to the multiport memory system

Cache memory

Cache memory is used to hold the most recent data and instructions to be processed. The presence of cache memory reduces average memory access time significantly.

Cache is implemented with high speed CMOS memory devices having an access time of less than 40 ns. The ND-110 CPUs feature a new extended caching system that integrates a microinstruction cache with the macro-instruction cache, effectively eliminating the need to decode instructions fetched from cache.

The two versions of the ND-110 differ in the size of their cache memory. The ND-110/CX has 4 Kword of cache whereas the ND-110 Standard has 1 Kword.

Local memory

Both versions of the ND-110 can address up to 16 Mword of local memory. Each word in main memory is stored with a 6-bit error correction code which makes it possible to:

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

Each memory card includes all necessary circuitry for error checking and correction on the card.

Multiport memory

In order for the ND-110 to access the multiport memory, a multiport memory transceiver is available.

If devices with high transfer rate are to be used, the multiport memory system should be employed to avoid cycle stealing from the CPU. ND-110 CPUs used as co-processors in ND-500 machines use multiport memory to communicate with the ND-500 CPU.

1.4.4 Input/Output System

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

Depending on the speed, a device could be connected to the ND-110 with:

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

Programmed Input/Output

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

Direct Memory Access

Direct memory access (DMA) is used to obtain high transfer rates to and from local memory. CPU activity and DMA transfers can occur simultaneously.

DMA shares the ND-100 bus with the CPU and has priority over the ND-110 CPU for bus access.

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

To avoid cycle stealing the DMA device can be connected to a separate port on the multiport memory system.

1.5 The ND cabinets

The ND-110 computer is delivered in variety of cabinets. The smallest versions (ND-110 Satellite) house a 7-position card crate, the ND-110 Compact models house a 12-position card crate, while the large cabinet versions can hold up to 21 cards.

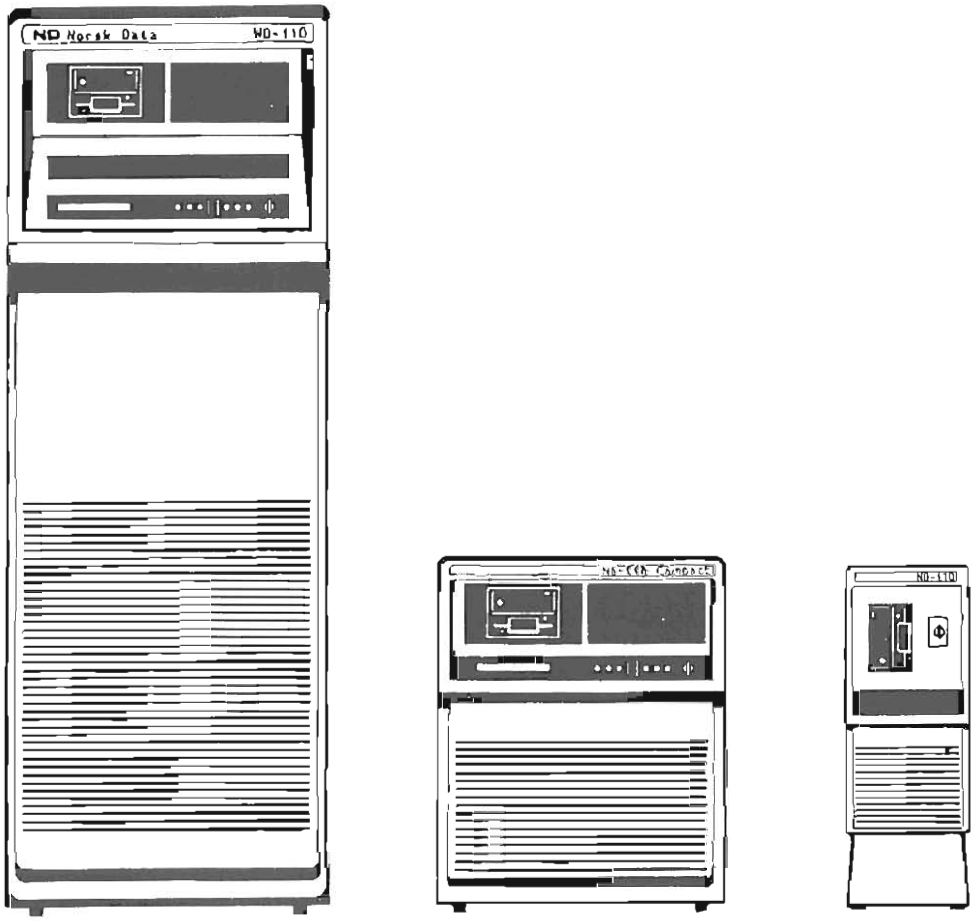


Figure 2. ND-110, ND-110 Compact and ND-110 Satellite cabinets

CHAPTER 2 CENTRAL PROCESSOR UNIT

CHAPTER 2 CENTRAL PROCESSOR UNIT

ND-110 is based upon a microprogrammed CPU architecture. A microprogrammed architecture means that:

- a large portion of system control is performed by the microprogram contained in the control store memory.
- each microinstruction contains bits to control each of the main elements in the system
- changes in the machine's instruction set are simple to make by rewriting the microprogram
- The microprogram resides in writeable memory (RAM) which can be modified by programs, allowing new versions to be installed without changing hardware.
- The hardware package-count is reduced, resulting in smaller computers.

The CPU fetches instructions from memory, then decodes and executes them. Each instruction consists of one or more microinstructions. These sequences perform the arithmetic, logic and control operations of the ND-110 CPU.

2.1 FUNDAMENTAL BUILDING BLOCKS

The ND-110 CPU card contains the following functional building blocks:

- RMAC address-arithmetic gate array
- BUFALU 16-bit ALU gate array
- RMIC microinstruction sequencer gate array
- Memory Management System
- Cache memory

- Control store
- Interrupt handler
- Trap handler
- Timing and real-time clock
- Operator control-panel interface
- Terminal no. 1 serial interface
- Register file
- CPU cycle controller
- ND-100 bus controller/interface

RMAC

The 16-bit virtual addresses are calculated here. They are sent to the memory management system (MMS) which converts the virtual addresses into physical addresses. If MMS is turned off, the logical addresses will be sent directly out to the memory system via the ND-100 bus.

BUFALU

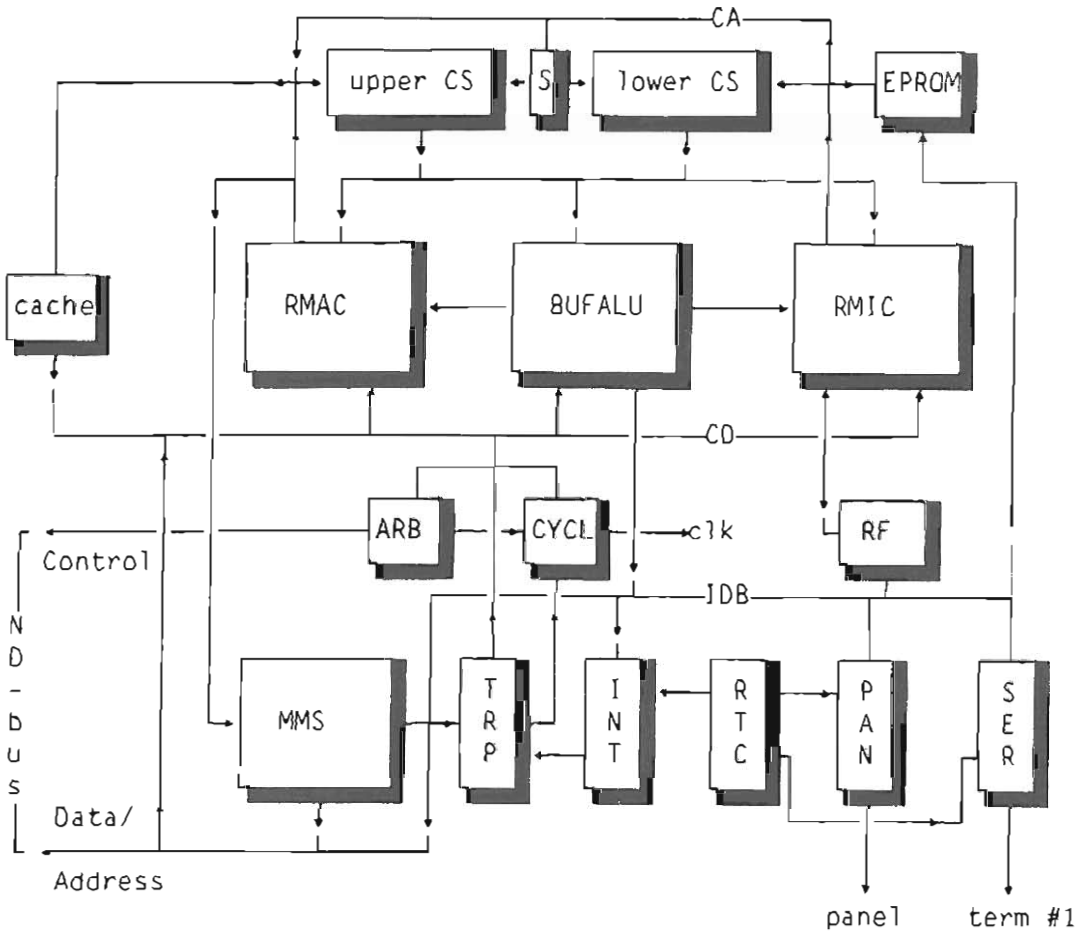
This is where the arithmetic and the logic functions are performed, i.e. the part in the processor that computes. It also contains the current register set.

RMIC

This is the control part of the CPU or microinstruction sequencer. Its job is to ensure that the CPU receives the microinstructions in the correct sequence.

Memory Management System

The memory management system converts the 16-bit virtual addresses from RMAC to physical memory addresses that are used on the ND-100 bus.



ARB : ND-100 bus arbitration controller
BUFALU : 16-bit ALU gate array
CA : cache address bus
CD : cache data bus
clk : internal clock signals
CS : control store (for microprogram)
CYCL : CPU cycle control
EPROM : control store firmware
IDB : internal data bus
INT : interrupt handler
MMS : memory management system
PAN : operator control-panel interface
RF : register file
RMAC : address-arithmetic gate array
RMIC : microinstructions sequencer array
RTC : timing and real-time clock
S : upper/lower cache bank select
SER : terminal no. 1 (console) serial interface
TRP : trap handler

Figure 3. ND-110 CPU functional blocks

Cache

On the ND-110/CX, cache is divided into four banks, each 1K (1024) words. Two banks are used for data words (16-bit), and the other two for instructions (16-bit + 64-bit micro-instruction). The ND-110 CPU uses only one cache bank for instructions and none for data.

Caching microinstructions in parallel with (macro)instructions is new for the ND-110 CPUs and is an important contribution to the speed advantage of the ND-110 family.

Interrupt handler

The interrupt system handles external interrupts by continually comparing the current priority level of the processor with the level of any interrupting devices. It identifies the device with the highest level above the processor's priority level, and generates a trap.

Trap handler

The trap handler is responsible for breaking into the execution sequence to react to any special condition requiring immediate treatment. Depending on the type of condition, the trap may interrupt the microprogram sequence or break in at the point when the CPU is about to fetch a new macroinstruction.

Timing and real-time clock

The ND-110 CPU derives all its timing signals from a central clock which is controlled by a quartz crystal.

Operator's Panel

A microprocessor controls the operator's panel and optional display.

Terminal no. 1

The terminal interface no. 1 communicates directly with IDB bus.

Register File

The working register sets for levels not currently running are stored here. The register file has two-way communication with the IDB, for saving and restoring the current register set.

The register file is also used as a scratch file by the microprogram. This part of the (extended) register file is only accessible from the microprogram.

CPU cycle control

The basic time unit of the ND-110 CPU is the nanocycle (26 ns). Microcycles, which contain of four or more nanocycles, are generated by a nano-controller. This is a finite-state machine which controls the sequence of events during the execution of a microinstruction.

ND-100 bus controller/interface

The ND-100 bus is controlled by an arbitration controller on the CPU card. All requests for the ND-100 bus and allocation of this bus are handled here.

2.2 INSTRUCTION SET

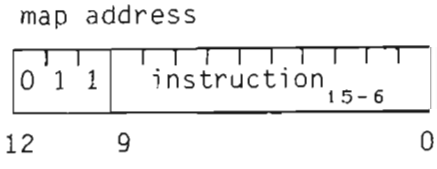
The ND-110 CPU uses a microprogrammed CPU architecture. The microprogram is executed from a 6K by 64-bit writeable control store. It is loaded from an EPROM at power on or when the MCL button is pressed, but not when MACL command is used from OPCOM.

The chief function of the microprogram is to perform the instruction set. Other functions implemented in microprogram include:

- operator's communication (OPCOM)
- built in test routines
- bootstrap loaders
- interrupt response
- saving/loading register set

New instructions have been included in the ND-110 instruction set, which allow the user to change the instruction set dynamically. These are described in more detail in chapter 8.

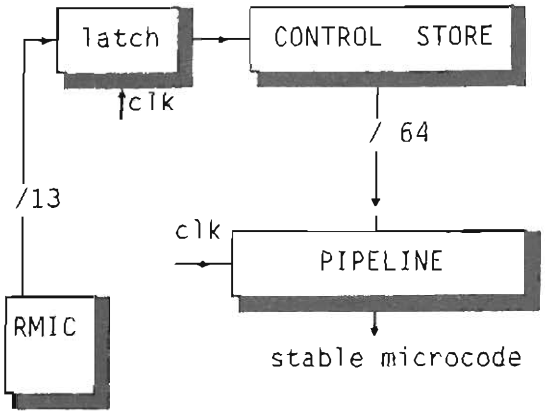
2.2.1 INSTRUCTION EXECUTION OVERVIEW



To find the microprogram entry point of an instruction, RMIC takes the upper 10 bits (bits 6-15) of the instruction itself shifts them 3 places to the right and combines them with the base address of an area in the control store called the map area. The data word at that address is the first microinstruction of the microprogram for this macroinstruction.

Several microinstructions may be needed to execute one machine instruction. RMIC controls the sequence of these microinstructions. Each microinstruction contains bits to control the various elements in the CPU. In addition it may contain a jump

address to a new microprogram sequence. RMIC contains a microprogram stack onto which the microprogram can "push" a return address.



The pipeline allows the next microinstruction fetch to occur in parallel with the execution of the current microinstruction. Parts of the current microinstruction are fed back into RMIC and are used to determine the next microprogram address. This address is clocked out of the microprogram sequencer at the same time as the current microprogram word is clocked into the pipeline register. The next microinstruction is being fetched while the current one is being executed.

Figure 4. Instruction pipeline

2.2.2 Instruction Fetch and Execution

The sequence of events during instruction fetch and execution is somewhat different for ND-110 compared to the ND-100.

Whereas the ND-100 pre-fetched the next instruction during the execution of the current one, the ND-110 fetches the new instruction on the last microcycle of each instruction. The ND-110 loses no speed advantage because of this, however. This is because instructions are normally fetched from cache memory, where they are stored partly decoded (See page 115 for details of the microcache).

Instruction Fetch

The machine instructions to be executed reside in memory. The CPU starts fetching the next macroinstruction during the last microinstruction sequence of the previous instruction.

The program-counter contents are sent out onto the CA and LA internal bus from RMAC, the address-arithmetic gate array. The memory management system translates the 16-bit virtual address to a 24-bit physical address, and the 16-bit instruction word is fetched.

INSTRUCTION DECODING

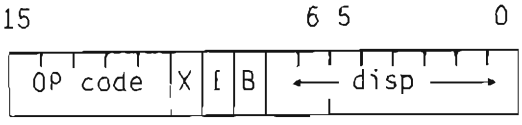


Figure 5. instruction format

Each instruction has a corresponding microprogram sequence in the microprogram control store. Execution of an instruction corresponds to running a microprogram sequence. Instructions must therefore be decoded to determine which microprogram sequence to run. This decoding is controlled by RMIC.

Bits 6 to 15 of the instruction specify which operation is to be carried out. Bits 8, 9 and 10 specify the addressing mode, when applicable. Bits 0 to 7 are normally used as a displacement to the operand address.

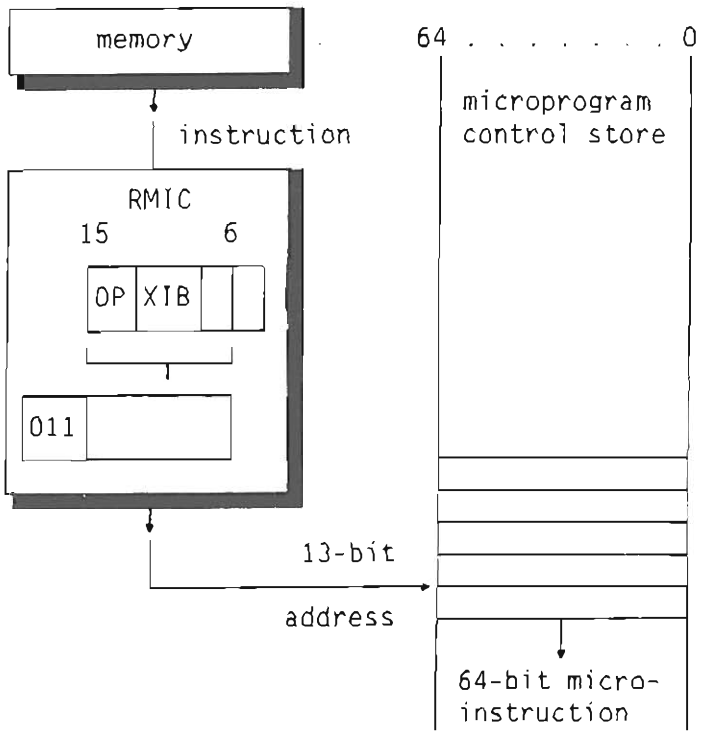


Figure 6. Instruction decoding

RMIC extracts the 10 most significant bits of the instruction (bits 6 to 15) shifts them six places to the right. This 10-bit address is the offset into the map area of the microprogram control store. The 3 most significant bits are set to point to the base address of the map area making a 13-bit control store address word.

The 64-bit microinstruction at this address is the first microinstruction of the macroinstruction. This microinstruction word may contain a jump to a new control store address if the microprogram sequence consists of more than one micro-instruction.

Instructions which use bits 0 – 7 as a displacement need four consecutive entries in the map area as bits 6 and 7 can take any value.

The output of the microprogram control store, together with the timing circuitry, controls the operation of the CPU. The microprogram sequencer manages the microprogram control store addresses and their sequence.

Instruction timing

The time an ND-110 CPU takes to complete the operations specified by one microinstruction is referred to as a microcycle, or the internal CPU cycle time. The ND-110 completes a microcycle in six or more nanocycles (one nanocycle = 26 ns). The faster ND-110/CX uses four nanocycles for its shortest microcycles.

The shortest (macro)instructions, when executed from cache, use one microcycle (compared to four on the ND-100 CPU).

Next microinstruction

When a microcycle is completed, the next microinstruction has already been read out from the microprogram control store. When the sequence of microinstructions is finished, a new fetch will be issued, and the CPU is ready for execution of a new macroinstruction (= machine instruction).

If the current macroinstruction was fetched from memory, RMIC uses bits 6 to 15 of the instruction word to generate an address within the map area of the control store. The data word at that address is the first micro-instruction of the macroinstruction.

Instructions fetched from cache "short circuit" this step. The first microinstr-uction is fetched from the microinstruction cache at the same time as the instruction is fetched from the instruction cache.

INSTRUCTION EXECUTION

An instruction to be executed will be fetched either from cache or memory. The sequence of operations following an instruction fetch depend on whether or not cache was used.

Fetch from cache

If the instruction is fetched from cache, the first microinstruction of that instruction will be fetched from microinstruction cache at the same time.

Fetch from memory

If the instruction was not available in cache, it will be fetched from memory. Bits 6-15 of the instruction (the op. code) will be shifted three places to the right in RMIC and added to the base address of the map area of the control store. The microinstruction at that address is the first of a sequence of microinstructions which must be executed to perform the function of the (macro) instruction.

Bits 0-10 of the instruction will be loaded into the instruction register (IR).

Microinstruction execution

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

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

Interruption of execution

The last microinstruction of each macroinstruction will test for an external interrupt before fetching the next macroinstruction. If an interrupt is active, a trap routine is entered. This trap routine will determine the source of the interrupt and take the appropriate action.

When changing from one program level to another as a result of an interrupt, the register set in BUFALU is saved in the register file. The working register set of the new level is then copied into BUFALU.

ALU operands

The ALU is controlled by the 64-bit microinstruction word which contains bit fields to select the A and B inputs to the ALU and the operation to be performed on them.

2.3 THE REGISTER FILE

There are 16 register sets available in the ND-110, one for each of the 16 program levels. Each of the register sets consists of 8 general programmable registers and 8 scratch registers which are accessible only from microprogram. Together these 256 registers are referred to as the register file.

In addition there are three sets of 256 registers which are referred to collectively as the extended register file (XRF). These registers are available only from the microprogram. The microprogram uses the extended register file as a scratch file.

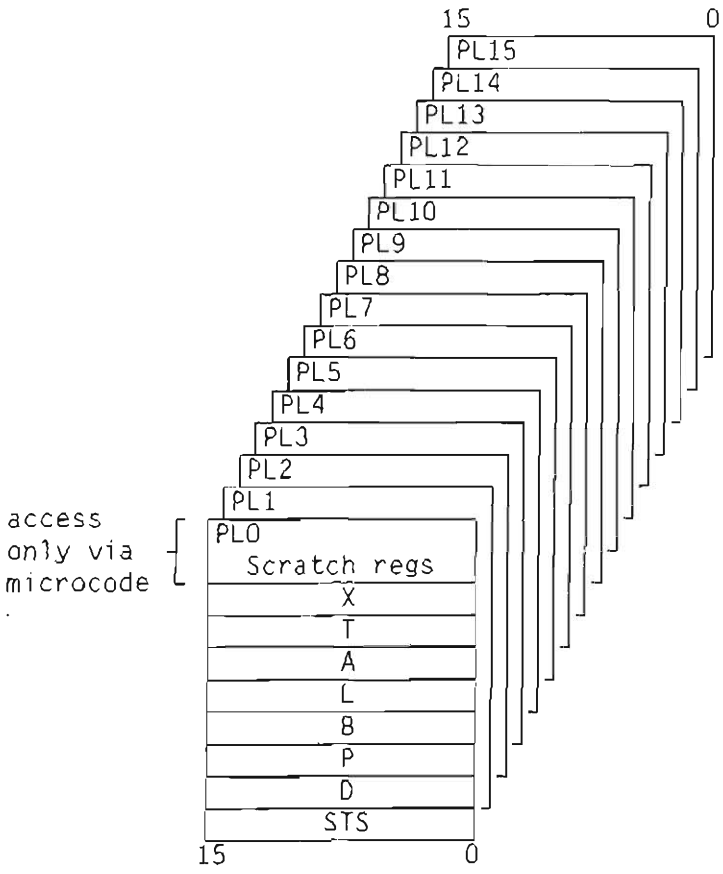


Figure 7. Register file structure

2.3.1 The 8 Working Registers

STS (status) register

This register holds the 8 status flags described on page 26.

D register

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

P register

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

B register

Base register or second index register. When used in connection with indirect addressing, it results in pre-indexing.

L register

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

A register

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

T register

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

X register

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

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

2.3.2 Status Flags

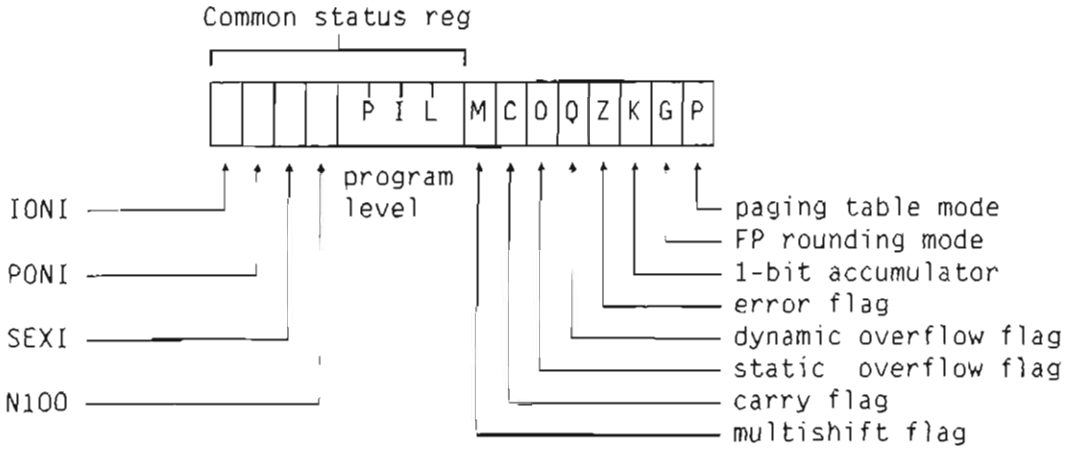


Figure 8. Status register (STS) bit assignment

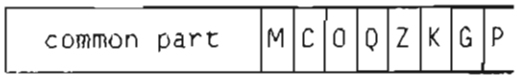


Figure 9. STS, program dependent

Eight flags are accessible by programs. These are:

M Multishift link flag. This is used in shift instructions as a one-bit extension of the register (A, D or T) to allow multiple word shifts.

C The carry flag is set or reset according to the result of arithmetic operations.

O Static overflow flag. The overflow flag remains set after an overflow condition, until it is reset by a program.

Q Dynamic overflow flag.

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

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

G Rounding flag for floating point operations.

P Page table mode. Enables use of the alternate page table.

These 8 flags are fully program controlled, either by means of the bit instructions or by the TRA or TRR instructions. Note, however, that TRR STS only writes to bits 0-7 of the status register, and not the whole register.

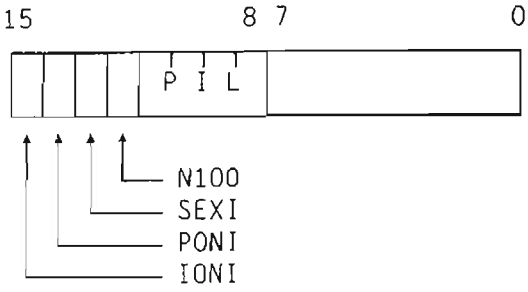


Figure 10. STS, machine dependent

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

IONI Interrupt system ON flag.

PONI Memory management ON flag.

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

N100 N100 flag to show that this is an ND-100 family CPU (ie not a NORD-10).

PIL Current program level.

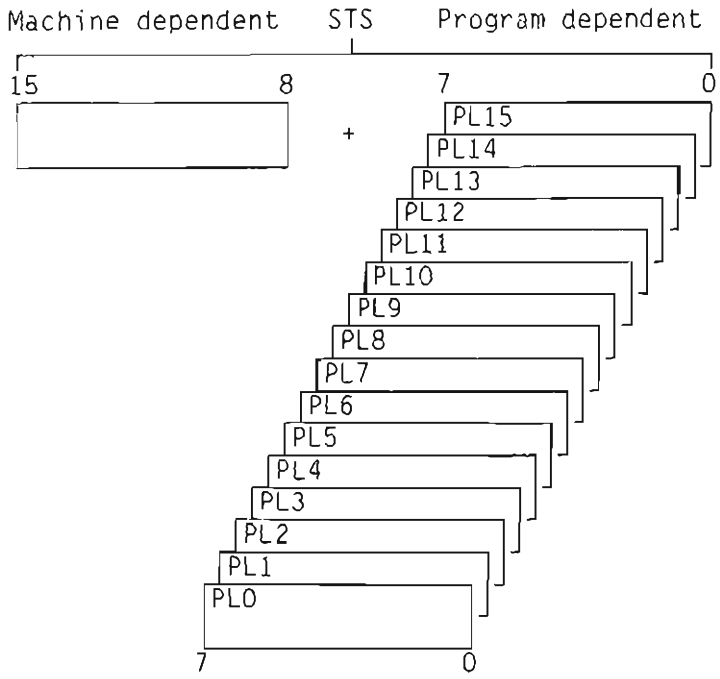


Figure 11. Status register

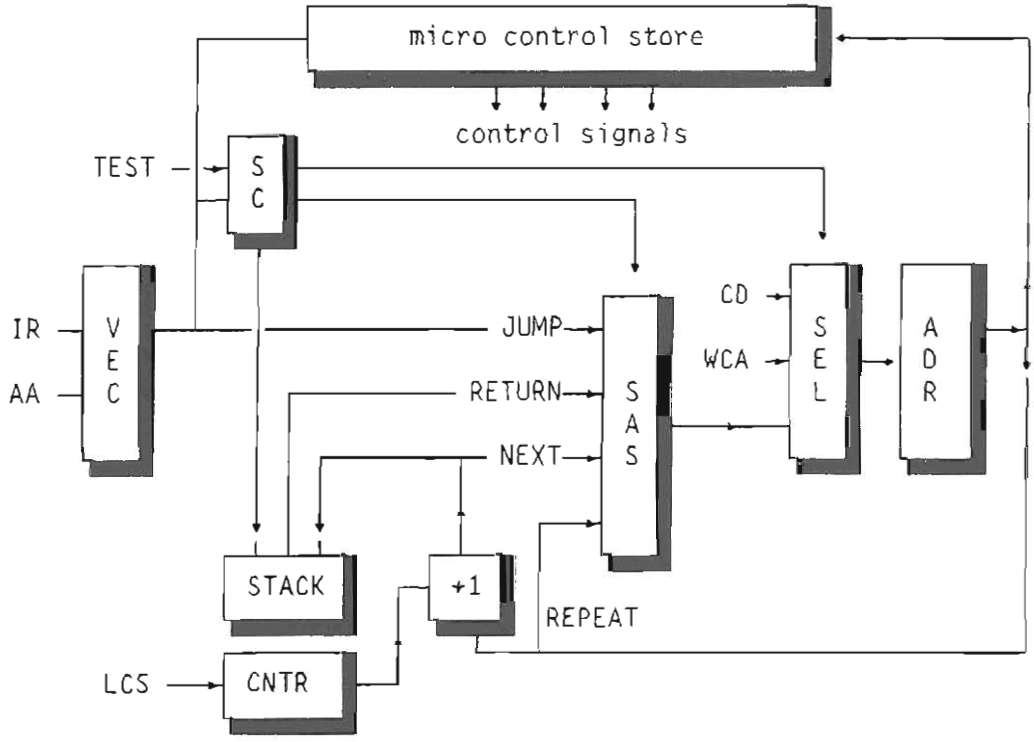
The figure above shows that each program level has its own version of STS bits 0-7 level, while STS bits 8-15 are common for all levels.

2.4 Microprogram sequencer RMIC

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

2.4.1 Sequencer operation

The purpose of the microprogram sequencer is to generate the address to the microprogram control store, making it possible to fetch and execute a microinstruction. The microprogram sequencer contains a microprogram address register (ADR) with multiplexed input, a push/pop stack and an incrementer. Control lines provide the information needed to select the source of the next microinstruction address.



AA : A operand address
ADR : microprogram address register
CD : cache data bus
CNTR : divide by two counter
IR : instruction register
LCS : load control store
SAS : sequencer address multiplexer
SC : sequence controller
SEL : microprogram address multiplexer
STACK : microprogram stack
TEST : test object (signal to used for a test condition)
VEC : vector address multiplexer
WCA : writeable control store address
+1 : next address incrementer

Figure 12. RMIC microprogram sequencer

It is possible to make branches in the microprogram, execute subroutines in the microprogram and carry out repetitive micro-instruction execution.

Sequencer control

RMIC determines the address of the next microinstruction with the help of two multiplexers (SAS and SEL in figure 12 on the previous page).

SAS selects from the address of current microinstruction (REPEAT), The address stored on the top of the microinstruction stack (RETURN), the output of the incrementer (NEXT) or to a direct address (JUMP).

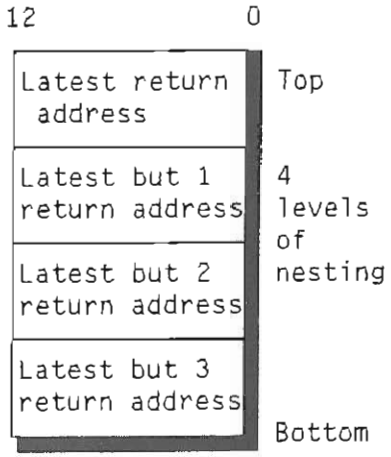


Figure 13. RMIC stack

The push/pop stack (LIFO) is used to store a return address when executing microsubroutines. It can be used for up to four return addresses. A set of control lines from the sequencer control, controls the push/pop stack and determines whether the function being performed is a jump to a subroutine (PUSH), or a return from a subroutine (POP). It is also possible to hold the stack information (HOLD) or to load the top word (LOAD) with the current microprogram address + 1 without affecting the rest of the stack.

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

The sequencer controller (SC) directs two multiplexers, SAS and SEL. In addition it controls the sequencer stack if a conditional sequence is not specified.

The first multiplexer, SAS, selects the source of the next microprogram address:

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

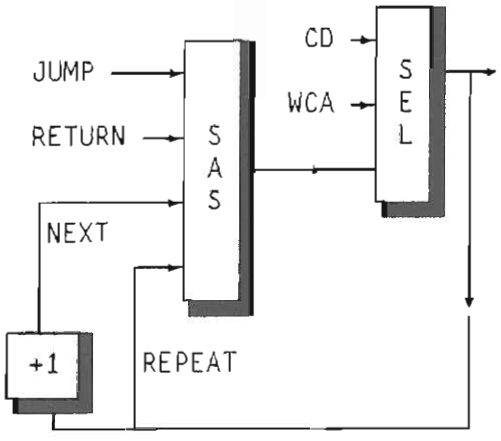


Figure 14. RMIC source address

A direct branch address comes from the branch address field in the microprogram.

If the incrementer is selected as the source address (NEXT), the sequencer will step to the next instruction of the microprogram.

The second address multiplexer, SEL, chooses between the output of SAS and an address which comes from the CD bus. The address from the CD bus is used for:

- trap vectors
- mapping (finding the first microprogram address of an instruction)

- directly addressing the writeable control store. In this case the address from CD is latched in WCA.

Control store words are 64 bits wide but must be written as four 16-bit words, which are stored temporarily in the register file. WCA holds the control store address, while the output of CNTR selects which 16-bit group is to be used.

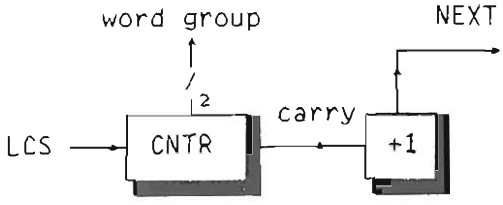


Figure 15. Loading control store

CNTR functions as a latch for TRR CS and TRA CS instructions. In this case CNTR contains the 2-bit select field for the 16-bit group specified in the instruction. When the control store is being loaded, CNTR steps through the 16-bit groups in sequence, propagating a carry to the microprogram address incrementer (+1).

2.4.2 Sequencing

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

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

The microinstruction format is given in appendix F. The Microprogramming Description manual for ND-100 (ND-06.018.1) may be referred to for general information. A new microprogramming manual for ND-110 is planned. Refer to your local sales office for availability.

2.4.3 Functional Flow

The execution of a macroinstruction will always start with the microinstruction found in the map area or a copy of this word fetched from cache. This microinstruction will normally contain a jump to the continuation of the microprogram sequence.

The microinstruction word supplies the 9 most significant bits (4 - 12) of a branch address, while the 4 least significant bits (0 - 3) are taken from the vector selector. Together they give a 13-bit address into the control store.

The input to the vector select is one of the following:

- The microinstruction word bits 0 - 3 In this case the full 13 bits come from the microinstruction word.
- IR (Instruction register) bits 0 - 3.
- AA (A address)

The source of the A address is determined by bits 50-52 (A OPER) of the microinstruction:

- The microinstruction word bits 12-15
- PIL register
- IR bits 3-5 or bits 3-6.
- The loop counter (LC)

Example of Instruction Fetch and Execution

The LDA ,8,X instruction is to be executed. Assuming that the instruction is fetched from memory and not from cache, the sequence of operations will be:

1. RMIC combines bits 6 to 15 of the instruction word (046400_8) with the base address (06000_8) of the map area of the control store.

2. The resulting 13-bit word (06464_8) is used as the control store address. The 64-bit word at that address is the first microinstruction of the LDA ,B,X instruction.

3. The first microinstruction adds the contents of the X and B registers and passes the result to RMAC on the RB bus. It also contains a jump to a new address (00170_8) in the microprogram where the LDA ,B,X operation is completed.

4. The microinstruction at 00170_8 performs a read from the 16-bit address pointed to by (X + B). RMAC adds this to the displacement (bits 0 to 7 of the instruction) and the resulting 16-bit logical address is converted to physical address by memory management (MMS). If the data is not present in cache, execution will wait for an ND-100 bus transaction.

5. When the data word is available, the microprogram jumps to the last microinstruction word of the LDA,B,X instruction.

6. The A register is loaded with the 16-bit data word now present on the DBR (internal register) and a new instruction fetch is started.

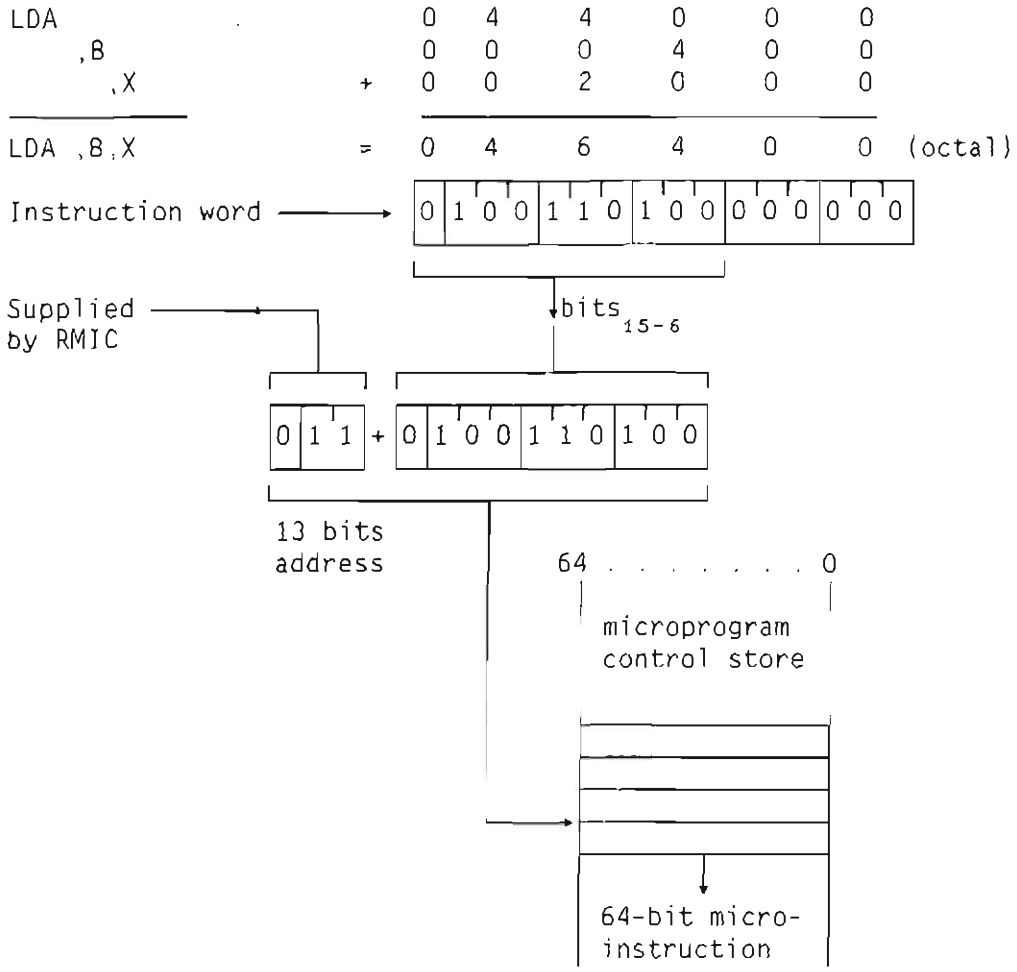


Figure 16. Decoding of the LDA ,B ,X instruction

If the instruction had been fetched from cache, the first microinstruction would have been fetched in parallel from the microcache. Points 1 and 2 above would be "short circuited" and execution would begin directly at point 3.

INTERRUPTION OF EXECUTION

Traps can break into the execution sequence in order to allow the CPU to attend to something more important than the current activity. The priority of the trap determines when it is allowed to interrupt execution. In decreasing order of priority they are:

- Internal interrupts (highest priority)
- External interrupts (including panel interrupts)
- Internal interrupts (remaining)

Internal interrupts of highest priority are conditions that are so important that they must break into the execution of the microprogram. They are listed in decreasing order of priority.

- Master clear/Power clear
- Page fault
- Protect violation
- Ring down
- Page used
- Written in page

External interrupts and the remaining internal traps are handled only when an instruction is to be fetched.

Panel interrupts have priority over other external interrupts.

The remaining internal interrupts are:

- Power fail
- Memory out of range
- Memory parity error
- IOX error
- Z error
- Monitor call

2.5 PIPELINE

The pipeline is not localized in one place, but is distributed within the CPU. It is neverthless convenient to regard it as a single register. Data is clocked into the pipeline register from the control store at the beginning of a microcycle. At the same

time the control store address for the next microinstruction is valid on the control store address bus. The ALU operation starts immediately.

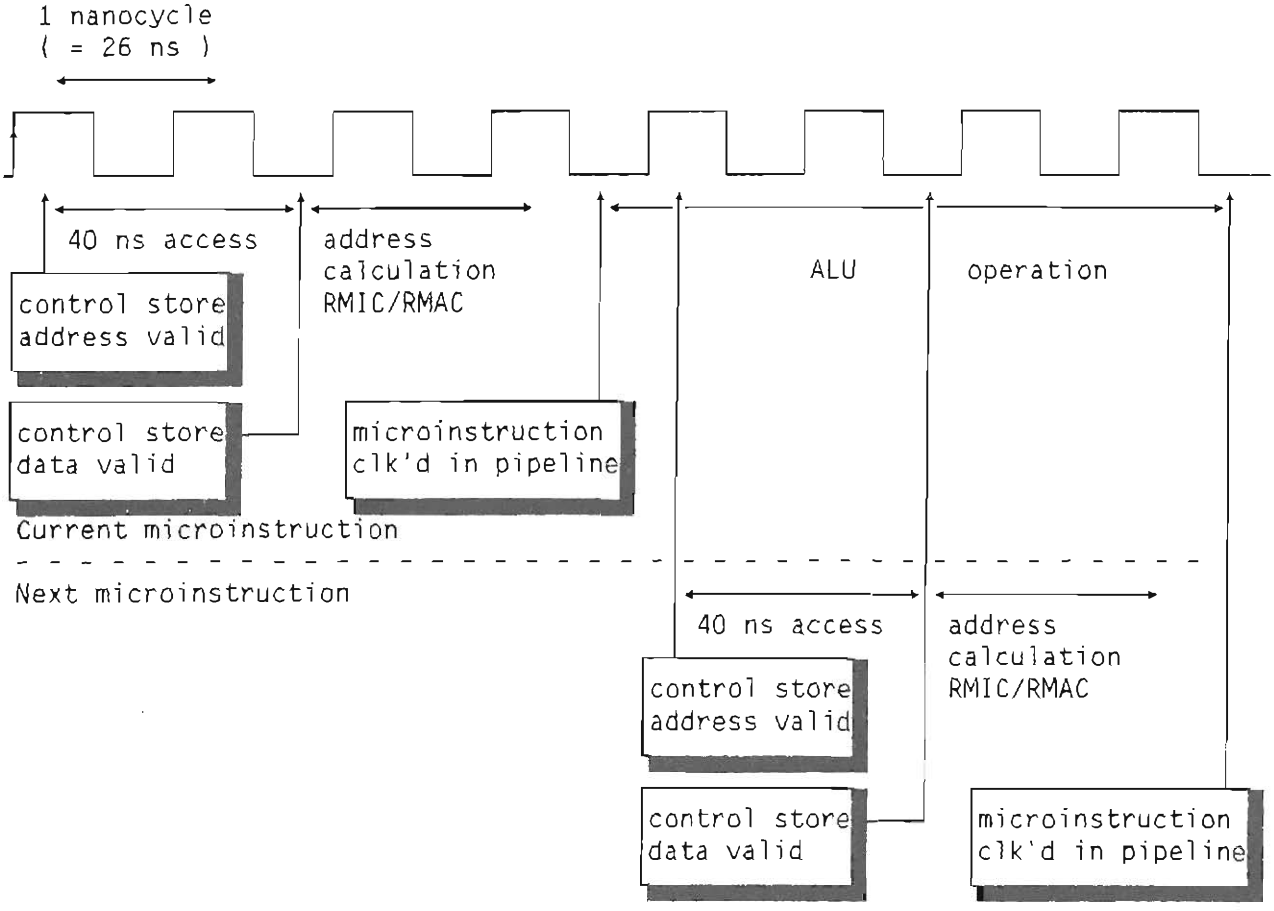


Figure 17. Execution pipeline

Forty nanoseconds after the start of the microcycle, the 64-bit data word is stable from the control store output. Parts of this microinstruction word are used by RMIC or RMAC in address calculations ready for the ALU operation in the coming microcycle.

At the end of the microcycle the new microinstruction is ready for execution and the control store has the next microinstruction valid at its output.

2.6 THE ARITHMETIC LOGIC UNIT

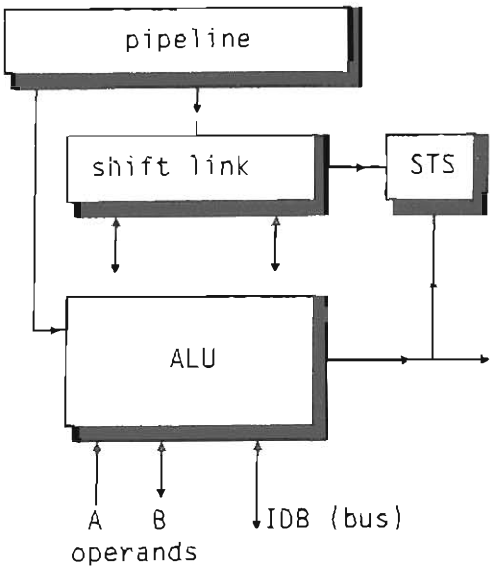


Figure 18. Arithmetic logic unit

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

The figure to the left shows the ALU and its connection to the system. The control lines from the pipeline register go in as instructions controlling the ALU operation, and as shift linkage control of the ALU shift operations with the right in/left out and right out/left in lines.

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

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

2.6.1 ALU OPERATION

Microprogramming
Readers who are not interested in microprogramming may prefer to skip this section and jump to the section on the interrupt system on page 40

The 16-bit wide ALU is contained within the BUFALU gate-array. In a microprogrammed processor system such as the ND-110, the functional blocks of the ALU are driven by sets of control lines corresponding to fields in the microinstruction. See Appendix (F) for details of the microinstruction format for the ND-110 CPU.

The ND-110 microinstruction format differs slightly from the format for the ND-100 CPU. In particular the IDBS (internal data bus source) field and the COMM (command code) fields have new values defined.

ALU primitive operations

Terminology
Although the ND-110 does not use the 2901 ALU bit slice, used in the older ND-100, the terms R, S, A, B and Q are used consistently with their use in the 2901.

The ALU has 8 primitive operations defined. These are the basic arithmetic and logic operations:

- R + S
- S - R
- R - S
- R OR S
- R AND S
- NOT (R) AND S
- R XOR S
- NOT (R XOR S)

R and S are specified by ALU source field (bits 55-57) of the microinstruction.

S may come from one of:

- Q (internal holding register)
- B-source
- A-source
- 0 (forced to zero)

R may come from one of:

- A-source
- D (internal data bus)
- 0 (forced to zero)

The A-source and B-source are specified by the A-oper (bits 50-52) and B-oper (bits 48-49) fields of the microinstruction.

A operand select

The A source can be specified by:

1. RA (bits 12-15 of the microinstruction)
2. PIL (program level)
3. IR (instruction register) bits 3-5

4. IR (instruction register) bits 3-6
5. LC (the loop counter)

B operand select

The B operand can be specified by:

1. RB (bits 0-3 of the microinstruction)
2. IR (instruction register) bits 0-2
3. IR (instruction register) bits 3-5
4. LC (the loop counter)

The ALU result (F) may be directed by the ALU-dest field (bits 61-63) to:

- Q (the holding register)
- B-source
- Y (ALU output = IDB bus)

In addition the ALU output (Y) may receive the A-operand directly.

The holding register (Q) can be used to keep results from ALU operations. This register can be shifted right or left. It may also be linked to the ALU result (F) for 32-bit shifts.

When changing from one level to another, the working registers (X, T, A, L, B, P, D, STS) in the current register set will be written into the current level registers in the register file. The working registers on the new level will be copied from the register file into the BUFALU register set.

The eight scratch registers contain temporary information managed by the microprogram, such as addresses during memory reference instructions, temporary results during floating point operations, etc. These scratch registers will not be saved under a level change.

EXAMPLES OF ALU OPERATION

Two brief examples illustrate the way the ALU is controlled by the microprogram.

Example 1

The LDA instruction:

The operand to be loaded into the A register is read from memory into the DBR (data register) by the first microinstruction.

The second microinstruction selects the DBR as the source for the IDB (internal data bus). The ALU function is PASSD. This specifies R to be D (i.e. the internal data bus IDB) and S to be 0 (zero). The operation performed is R XOR S (which passes R unchanged when S is zero). ALU destination is the B-operand, which is selected to come from the microinstruction itself, and has the value for the A register (= 5).

Example 2

The RADD SA DB instruction:

The A-operand is IR (instruction register) bits 3-5 which contains the value for the A register (= 5). The B-operand is IR bits 0-2 which contains the value for the B register (= 3). The ALU function is A + B and the ALU destination is the B-operand, which is the B register.

2.7 THE INTERRUPT SYSTEM

One CPU can handle many simultaneous processes, but only one process can be active at any time.

Polling

The simplest approach to such an asynchronous event handler is the method known as polling. The processor tests the status of each event, in sequence and, in effect, 'asks' if service is required.

Interrupts

Interrupt is an efficient way of servicing asynchronous requests. When the processor receives the interrupt request, it suspends the program it is currently executing, execute an interrupt service routine and then resume the execution of the suspended program.

Interrupt levels

The ND-110 uses a multilevel, vectored interrupt system. Interrupt requests can occur on different levels. Each level is assigned a priority; a lower number means lower priority. These "interrupt levels" are in fact identical with the ND-110 program levels.

Daisy-chain

As more than one device may generate an interrupt on the same level, there must be some means of deciding priority within a level. The method used by the ND-110 is the "daisy-chain". This means, in practice, that the position in the card frame decides priority. The closer to the CPU the higher the priority.

Nested interrupts

An interrupt service routine may be interrupted by an interrupt request from a higher level. The service routine for the higher priority request is executed, after which execution of the interrupted service routine is resumed.

Enabling/disabling interrupts

Interrupts may be enabled and disabled using the privileged instructions ION, and IOF. (The instruction PION and PIOF, which also affect the paging system, may also be used.)

Interrupt vectors

Each device that can use the interrupt system has an interrupt code assigned to it. This code may be used to direct execution to the appropriate interrupt service routine.

PROGRAM LEVELS

There are 16 program levels in the ND-110 and therefore, 16 sets of registers and status flags. Each set consists of A, D, T, L, X and B registers, program counter (P) and a status register (STS) with the status flags O, Q, Z, C, M, K, PTM and TG. There are also 8 registers that are only accessible from the microprogram.

Context switch

Changing program level is done by means of an interrupt. This may be external, internal (trap), or programmed. A program may relinquish priority by executing a WAIT instruction. Context switching from one program level to another is completely automatic and requires only 7.2 µs, including saving and unsaving all registers and flags.

Priority increases with level: program level

15 has the highest priority and program level 0 the lowest.

Table 1 below shows how the SINTRAN III operating system uses the 16 program levels.

All program levels may be activated by software. In addition, the levels 10, 11, 12 and 13 may be activated by 512 external I/O interrupts. The IDENT instruction is used to identify the interrupting device. The IDENT is described in greater detail on page 150.

Program level 15

This level is used for extremely fast interrupts and may only have one I/O interrupt source. It is not used by standard ND equipment or software, but is available for users who need an immediate access to the CPU.

Program level usage

Level	Usage (SINTRAN III/VSX vs.K & later)
15	Extremely fast user interrupts
14	Internal interrupts
13	Real-time clock
12	Input devices
11	Mass storage devices
10	Output devices
9	not used
8	not used
7	not used
6	not used
5	Xmsg
4	I/O monitor calls
3 [1]	segment administration
2 [1]	SINTRAN III monitor
1	Real-time and background
0	Idle loop

Note 1: Levels 2 & 3 have changed usage from previous versions of SINTRAN.

Table 1. Level Assignments

Program level 14

Program level 14 is used by the internal interrupt system, which monitors error conditions or traps in the CPU. Level 14 has 10 possible sources. Each source is represented by a bit in the interrupt control register (IIC) and interrupt mask register (IIE).

Program level 13

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

Program level 12

Character input devices such as terminals, tape readers, etc., are connected to level 12, and so is HDLC output.

Program level 11

Level 11 is used by mass storage devices such as disk, floppy disk, mag. tape, etc.

Program level 10

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

When an interrupt request is serviced, the CPU moves to a higher program level. The interrupt service routine ends with a WAIT instruction which relinquishes priority and the CPU moves to a lower level. If another interrupt request has occurred, with greater priority than the original program, then the new interrupt is serviced first. If no interrupt has occurred or the level of the new interrupt is lower, the original routine resumes.

The interrupt controller takes care of all interrupts on levels 10-15, including all internal interrupts. Interrupts on levels 0-9 are implemented by the microprogram.

The interrupt controller sends out a 5-bit vector specifying the interrupt source. These bits specify a branch address in the microprogram.

Operation

In the IIC register there are ten bits for level 14 and one bit each for levels 10-13, and level 15. Interrupts for levels 0-9 are accessed via the microprogram.

When an interrupt is received, the IIC register and the mask register (IIE) are combined using a logical AND operation. The priority encoder selects the highest priority level of the result. This is compared to the current level. If the interrupt level is higher and the interrupt system is on, the interrupt will be accepted.

An interrupt routine may be interrupted by a higher priority interrupt. Service to the lower priority routine is resumed automatically upon completion of the higher level.

2.7.1 The External Interrupt System

External interrupts are controlled by the two 16-bit registers:

PIE Program Interrupt Enable

PID Program Interrupt Detect

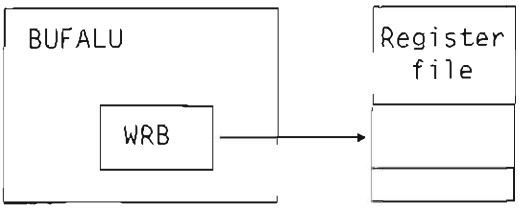
The PIE register is controlled by program only. The PID register may be controlled both by program and by hardware interrupts. At any instant, the program level is the highest program level which has its bits set (= 1) in both PIE and PID.

The actual mechanism is as follows:

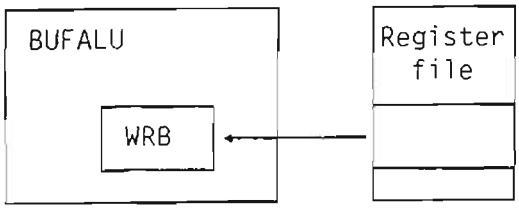
The 4-bit PIL field in the STS register contains the value of the current program level.

The PIL is compared with PK (the output of the priority encoder). PK contains the highest program level which has its corresponding bits set in both PIE and PID. Whenever PK is greater than PIL, an automatic change of context will take place. This is done by a microprogram sequence.

The level change can be illustrated as follows:



1. The interrupt system is temporarily blocked to prevent false interrupts.
2. The working registers (WRB) on the current level are saved in the register file.
3. The PIL (program level) register on the current level is copied into the PVL (previous program level) register.
4. The PK (new level priority code) register is copied into the PIL (program level) register. The level change takes place at this time.



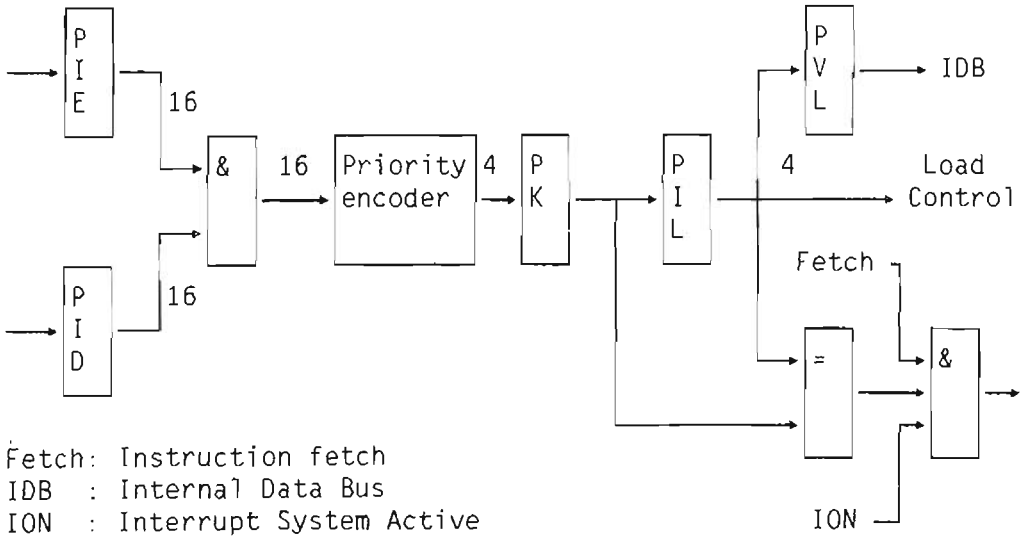
5. The register set for the new level is moved from the register file to the working registers. The paging control register (PCR) is loaded from the extended register file (XRF) at the same time.

6. The first instruction on the new level is then fetched.

This complete sequence requires 7.2 µs.

The PID register is a sixteen bit interrupt detect register used for both internal and external interrupts.

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



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

Figure 19. External Interrupt System

External Interrupt Identification

Since a vectored interrupt system is used, more than one device may use the same interrupt line. This means that we need to know which device generated the interrupt request. The vector or identification code is found using an IDENT instruction. The instruction has the following format:

IDENT <program level code>

In a ND-110 system there is a maximum of 2048 vectored interrupts. Each physical input/output unit will usually have its own unique interrupt response code and priority.

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

The standard way of using these levels is as follows:

Level 13: Real time clock

Level 12: Input devices

Level 11: Mass storage devices

Level 10: Output devices

When an IDENT instruction is executed, the bus controller hardware searches for the interrupting device.

The first device with an active interrupt on the current level will respond with its 9-bit identification code and remove its interrupt request. The CPU uses this code to calculate a vector to the driver routine for the interrupting device.

Using 9 bits allows 512 different vectors on each of the four external interrupt levels (10-13). This means that a maximum of 2048 vectors are possible.

If more than one device on the same level generates interrupts, the device interface located closest to the CPU has the highest priority. If there is more than one device

connected to the card, an internal priority mechanism on the card determines which is handled first.

Programming Example:

The following example shows how interrupts on level 13 might be handled. The first line of the routine uses the IDENT instruction to read the identification code of the interrupting device into the A register. The second instruction adds this value to the program counter (P register) to compute the address of the device handler. This causes a vectored jump to the device handler.

IDENT code zero means error

If the IDENT instruction returns a code of zero, it means that no device has sent an identification code. Consequently the instruction immediately following the RADD SA DP instruction, is a jump to an error handler.

The device handlers end with a jump to CONT which relinquishes priority. The next time level 13 is entered, execution continues with the instruction following WAIT, which is a jump to the start of the interrupt routine (LEV13).

Example of a level 13 interrupt handler

```
LEV13   IDENT    PL13     % Identify device on level 13
        RADD     SA DP    % Computed GO TO
                          % add A reg. to P reg (PC)
        JMP      ERR13    % error handler
        JMP      DRIV1    % Device handler 1
        JMP      DRIV2    % Device handler 2
          .
          .
          .
        JMP      DRIVN    % Device handler N
          .
          .
          .
CONT    WAIT              % relinquish priority
        JMP      LEV13    % loop to start of interrupt routine
```

2.7.2 Internal Interrupt System

Internal interrupts are generated by the trap handler. All internal interrupts are on level 14.

It is controlled from the two registers:

IIE: Internal Interrupt Enable

IID: Internal Interrupt Detect

The IID register is not program accesible.

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

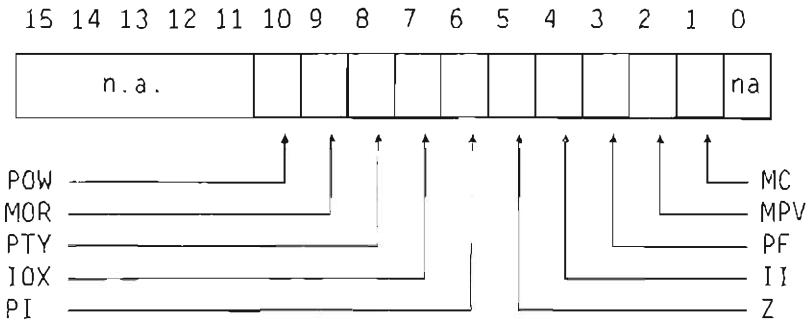
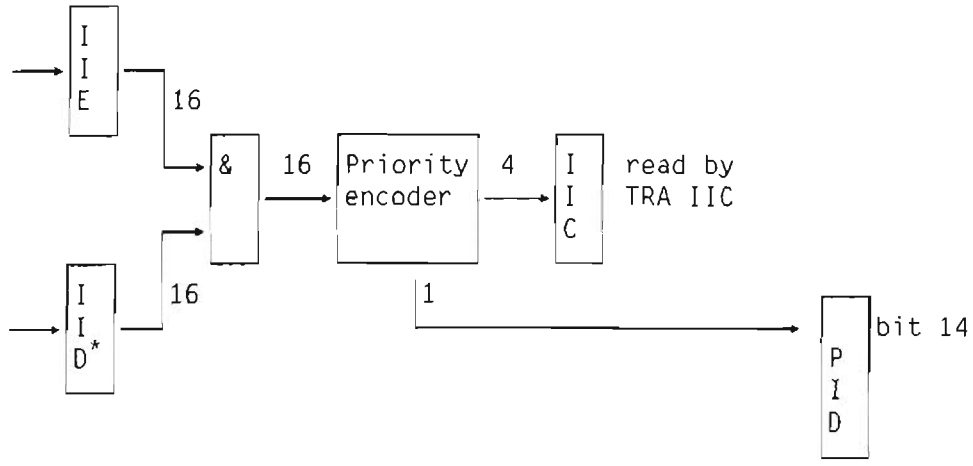


Figure 20. IIE register

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



Note (*) : The IID register is not program accessible.

Figure 21. Internal interrupt system

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

Condition	Code	Cause
NA	0_8	Not assigned
MC	1_8	Monitor call
MPV	2_8	Memory Protect Violation Page number is found in the paging status register
PF	3_8	Page fault Page not in memory.
II	4_8	Illegal instruction. Instruction not implemented.
Z	5_8	Error flag. The Z flag is set (= 1).
PI	6_8	Privileged instruction
IOX	7_8	IOX error. No answer from external device.
PTY	10_8	Memory parity error
MOR	11_8	Memory out of range Addressing nonexistent memory
POW	12_8	Power fail interrupt

Table 2. Internal interrupt codes

MPV, PF

Memory protect violation and page fault interrupt the microprogram, i.e. within a machine instruction.

PI, II

Privileged instruction and illegal instruction are detected and trapped by the microprogram.

IOX, MOR, MC
Z, PTY and POW

The remaining traps will not give an internal interrupt until the current machine instruction has been completed.

If PF or MPV occur during a fetch cycle, the P-register (program counter) is not incremented. In all other cases, P-register points to the next machine instruction.

Note
IOX, MOR and Z traps can occur up to two instructions after the event that caused them. This is because the instruction pipeline must be emptied first.

Power fail (POW) has the highest priority. There is no priority assigned among the other internal interrupts, as only one condition can arise at a time.

The PIE bit 14 must be set to enable internal interrupts as well as the appropriate bit in IIE Interrupt Enable.

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

Internal Hardware Status Interrupts

Monitor Call Interrupt

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

The MON instruction may have up to 255 different codes (the eight least significant bits of the MON instruction). The T register will contain this value, sign extended (bit-7 is sign).

Protect Violation Interrupt

Two types of protect violations are possible:

- Memory Protect Violation

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

- Ring Violation

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

Details about the cause of this interrupt are found in the paging status register.

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

Page Fault Interrupt

Generated if the program has attempted to reference a page that is not presently in memory. The paging status register will contain details of the page number, etc.

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

Illegal Instruction Interrupt

Caused by an attempt to execute an instruction that is not implemented.

Error flag interrupt

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

- Floating point divide by zero
- Attempt to EXR an EXR instruction
- DNZ overflow
- RDIV overflow
- Program setting of Z (BSET, MST or TRR)

Note: Level 14 interrupt routine must always reset the Z flag on the interrupting level, otherwise a new interrupt will occur when that level is reentered.

Privileged Instruction Interrupt

An attempt to execute a privileged instruction from ring 1 or 0 causes this interrupt. The complete list of privileged instructions are given in appendix D.

The paging system must be turned on (PON instruction) to receive a privileged instruction interrupt.

IOX Error Interrupt

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

Memory Parity Error Interrupt

A memory parity error has occurred. The PES and PEA registers contain information about the memory failure. The contents of these two registers are locked until the PEA register has been read.

The PES register contains the upper 8 bits of

the address, the error code and status information. The least significant 16 bits of the failing address may be read from the PEA register (TRA PEA). You must read the PES register first. Reading the PEA register unlocks both.

See page 129 in this manual and the ND-100 Hardware Maintenance Manual (ND-30.008.2) for details of how to interpret this information.

Memory Out of Range Interrupt

This interrupt occurs when the program attempts to access an address which does not exist in memory. The PES and PEA registers contain information about the access failure. You must read the PES register first. Reading the PEA register unlocks both.

The PES register contains further information as for memory parity error (see above). The least significant 16 bits of the referenced address can be read from the PEA register.

Power Fail Interrupt

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

Reading IIC

Executing the instruction TRA IIC will load the contents of IIC into the A register, bits 0-3. Bits 4-15 will be zero.

The instruction TRA IIC automatically resets IIC.

INTERNAL INTERRUPT IDENTIFICATION

An internal interrupt will force the CPU to level 14. The IIC (Internal Interrupt Code) register contains a vector indicating the source for the interrupt. The register is locked to prevent overwriting.

After executing a TRA IIC the IIC register is cleared and the A register contains the internal error code. These codes are listed in table 2. A branch to the internal interrupt handler can then be made.

EXAMPLE OF INTERNAL INTERRUPT ROUTINE

The following example shows how internal interrupts may be handled. The TRA IIC instruction reads the IIC register. The value is added to the program counter to form a jump address. The various internal interrupt routines must end by jumping to EXIT14 where a WAIT instruction relinquishes priority. The next time level 14 is entered, execution will continue to the JMP LEV14 instruction, which starts the interrupt handler routine again.

```
LEV14,    TRA       IIC      % Place IIC code in A reg.
                             % and reset error lock
          RADD      SA DP    % computed GO TO - add A reg.
                             % to P. reg. (program counter)
          JMP       ERROR    % 0, error not assigned
          JMP       MONCL    % 1, monitor call 7
          JMP       PROTN    % 2, protection violation
          JMP       PAGEF    % 3, page fault
          .
          .
          .
          JMP       POW      % 10, power failure
          .
          .
          .
EXIT14    WAIT
          JMP       LEV14
```

2.7.3 PROGRAM CONTROL OF THE INTERRUPT SYSTEM

When power is turned on, the power up sequence will reset PIE and the register block on program level zero will be used. Two instructions are used to control the interrupt system.

ION Interrupt system on

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

IOF Interrupt system off

The IOF instruction turns off the interrupt system, i.e. the mechanisms for a change in program levels are disabled. The computer will continue operation at the program level at which the IOF instruction was executed, i.e. the PIL register will remain unchanged. The interrupt indicator on the operator's display is reset by the IOF instruction. The IOF instruction is privileged.

PROGRAMMING THE INTERRUPT REGISTERS

PID and PIE may be read into the A register using the instructions

- TRA PID
- TRA PIE.

Three instructions are available to set these registers:

1. TRR PID (or PIE)
2. MST PID (or PIE)
3. MCL PID (or PIE)

TRR

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

MST

The MST, masked set, instruction will set the bits in the specified register to one where the corresponding bits in the A register are ones.

MCL

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

All program levels may be activated from program, by setting the appropriate bits in PIE and PID. These are called programmed interrupts.

EXAMPLES OF PROGRAMMED INTERRUPTS

Changing to a higher level

If interrupts to program level 9 have already been enabled (bit 9 in PIE is set), a call to that program level may be made from a lower program level by setting bit 9 in the PID register. This may be done with the MST instruction.

```
        SAA     0           % clear A reg.
        BSET    ONE 110 DA  % set bit 9 to 1 (11(8) = 9(10))
        MST     PID         % set PIO bit 9
NEXT,   .
        .
```

Changing to a lower level

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

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

Writing to IIE

The internal interrupt sources are enabled by setting the corresponding bits in the IIE register. This may be done with the TRR IIE instruction. The MCL and MST instructions are not available for the IIE register.

The IID register is not accessible from the program.

LEAVING THE INTERRUPTING LEVEL

When an interrupt routine has completed its work, it must relinquish priority using the WAIT instruction.

The WAIT will cause the CPU to exit the current program level, the corresponding bit in PID is reset, and the program level with the highest priority will be entered. This will then normally have a lower priority than the program level which executes the wait instruction.

If there are no interrupt requests on any program level when the WAIT instruction is executed, program level zero is entered. Level zero will normally perform an idle loop.

> Note
> The P register (program counter) of a level that has given up priority points to the instruction following the WAIT instruction. This instruction will often be a JMP to the start of the interrupt routine.

USE OF THE PVL REGISTER

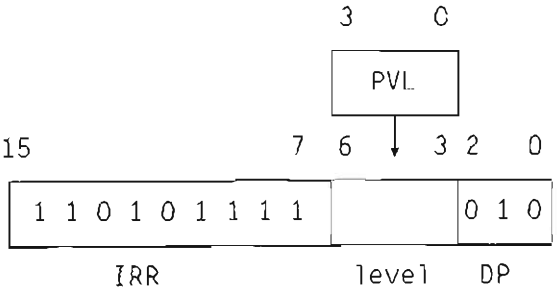


Figure 22. TRA PVL Instruction

When an internal interrupt occurs, it is often necessary to know the value of the P register (program counter) on the level that was active at the time of the interrupt.

This may be done with the TRA PVL instruction. This instruction will read the contents of the PVL register (4 bits) into the A register in bits 3-6. Bits 7-15 of the A register are loaded with the operation code for the IRR instruction (inter-register read). Bits 0-2 of the A register are set to DP (destination P register).

The A register will now hold the instruction:

IRR <previous level * 10_8> DP

By executing an EXR SA (execute A register) at this point, the contents of the A register will be executed as an instruction.

After the instruction has been executed, the program counter on the level which caused the interrupt will be found in the A register.

Note that there are some cases where the program counter has not been incremented, for example if a memory protect violation interrupt occurs. If this interrupt occurs during the fetch of an instruction, the program counter is not incremented, but if it occurs during the data cycle of an instruction, the program counter is incremented (see also the section on memory management system, page 87).

2.7.4 Initializing the Interrupt System

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

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

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

3. Initialising the P-registers (program counters), on the levels to be used. That is, the P-registers must contain the starting addresses of the programs to be executed on the respective levels.

4. If the Z (error) interrupt (IIE bit number 5) is enabled, the Z flag (bit 3) in the STS (status) register must be cleared for all levels being initialized.

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

   Performing a TRA instruction for IIC and PEA, will unlock all three registers.

6. The interrupt system must be turned ON.

Example:

The following example shows how the interrupt system can be initialised.

```
LDA         (76032    % 76032(8) = 0 111 110 000 011 010(2)
                      % levels 1, 3, 4, 10, 11, 12, 13 and 14
TRR         PIE       %
LDA         (3736     % 3736(8) = 0 000 011 111 011 110(2)
TRR         IIE       % Interrupt sources except Z flag
                    % Here the P register (program counter) for
                    % levels 1 and 3 are initialised. The other
                    % levels would be initialised similarly

LDA         (P1       % start address for level 1
IRW         10 DP     %

LDA         (P3       % start address for level 3
IRW         30 DP     %
.
.
.
TRA         IIC       % Unlock IIC
TRA         PEA       % Unlock PEA and PES
ION                   % Turn on interrupt system
JMP         START     % Go to main program
```

Note that 76032_8 is $0\ 111\ 110\ 000\ 011\ 010_2$

Bits 1, 3, 4, 10, 11, 12, 13 and 14 are set high (= 1) in the bit mask.

Similarly 3736_8 is $0\ 000\ 011\ 111\ 011\ 110_2$

Bits 1, 2, 3, 4, 6, 7, 8, 9 and 10 are set high (= 1).

2.8 MEMORY ADDRESSING

The ND-110 accesses memory as 16-bit words. There are four different types of memory access.

1. Instruction fetch. The word being fetched will be interpreted as an instruction.
2. Operand read. The word being fetched will be used as data.
3. Operand write. The word being written is data.
4. Indirect address fetch. The word being fetched will be treated as an address for the current operation.

The ND-110 uses relative addressing. This means that the address is specified relative to the contents of the program counter (P register), or relative to the contents of the B and/or X registers.

The following pages detail the various addressing modes available on the ND-110. These pages are preceded by a general description of the instruction format and the terminology used.

2.8.1 ADDRESS STRUCTURE

A large group of memory reference instructions share the same format:

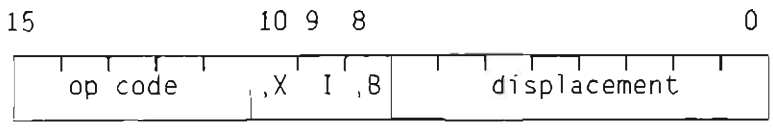


Figure 23. Memory reference instruction format

Bits 8 to 10 define the addressing mode and bits 0 to 7 the displacement. Together these two fields define the memory address.

The 8-bit displacement field is a 2's complement signed number (giving a displacement range of +127 to -128).

The five most significant bits, the op code, define the type of operation executed.

The eight possible combinations of ",X", "I" and ",B" define the following addressing modes:

- P relative addressing
- B relative addressing
- P indirect addressing
- B indirect addressing
- X relative addressing

- B indexed addressing
- P indirect indexed addressing
- B indirect indexed addressing

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

,X	I	,B	Mnemonic	Effective Address
0	0	0		(P) + disp
0	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 3. Addressing modes

EXECUTION TIMES

Indirect addressing increases the execution time of memory reference instructions. One extra microcycle is needed if the indirect address is found in cache; if it is not in cache the execution time is increased by one memory access.

When B relative indexed addressing (,B,X) is used, the instruction execution time is increased by one microcycle. This does **NOT** apply to B indirect indexed addressing (,X I,B).

PAGING

In the descriptions that follow the memory addresses used are 16-bit virtual addresses. You should remember that these are normally translated into 24-bit physical address by the memory management system. This translation process is described in detail in Chapter 3.

When memory management is ON, the translation may be done with the help of the standard page table (PT) or the alternate page table (APT). The rule is: P relative addressing uses the normal page table, B relative or indexed (,X) addressing modes use the alternate page table.

Indirect addressing results in two memory accesses. One for the indirect address and the second for the instruction operation itself. The memory management system regards these two memory accesses as separate operations and chooses PT or APT, according to the above rule, for each memory access.

The ND-110/CX normally uses cache for memory accesses. This has no effect on the way the memory address is formed. The only difference is that accesses from cache are faster. Section 5.4 (page 115) describes the ND-110 cache system in detail.

2.8.2 Addressing Modes

The following symbols are used below in the description of the addressing modes of the ND-110:

,X address relative to X register (post-indexed)

I indirect address

,B address relative to B register (pre-indexed)

d displacement (bits 0-7 of instruction) as a 2's complement value.

( ) contents of a register or memory location

ea effective address

n arbitrary address of a word in memory

K memory block base address pointer

* current value of the program counter

← points to

› loaded into

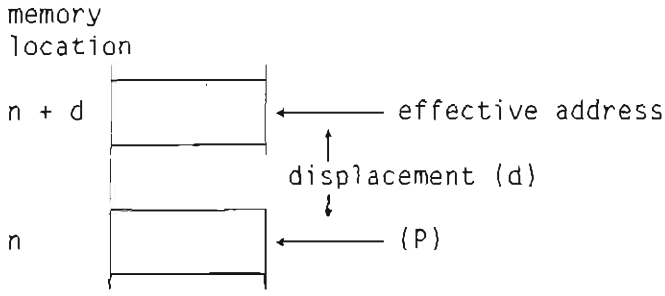
P RELATIVE ADDRESSING

,X=0
I=0
,B=0

Effective address = (P) + disp.

Description:

The effective memory address is calculated by adding the value of the displacement to the contents of the P register (program counter). If memory management is ON, the normal page table (PT) will be used.

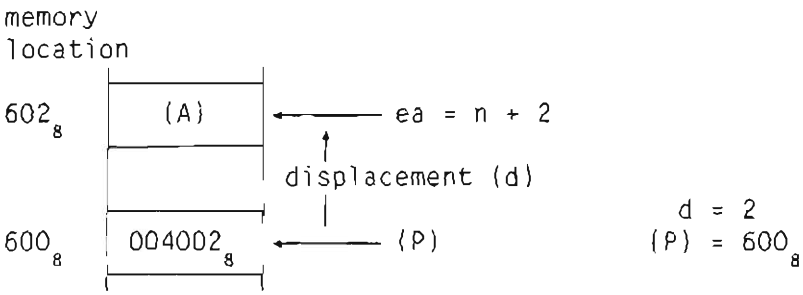


Note: d may have any value in the range -128 to 127.

Example:

STA *2 (instruction code 004002_8)

Store contents of A register in the memory location two words ahead of this instruction.



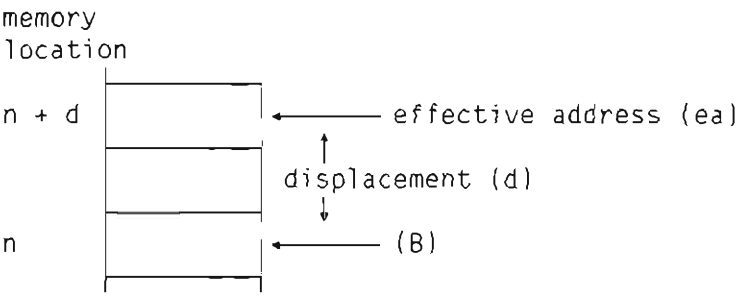
B RELATIVE ADDRESSING

,X=0
I=0
,B=1

Effective Address = (B) + displacement

Description:

The effective address is calculated by adding the value of the displacement to the contents of the B register.

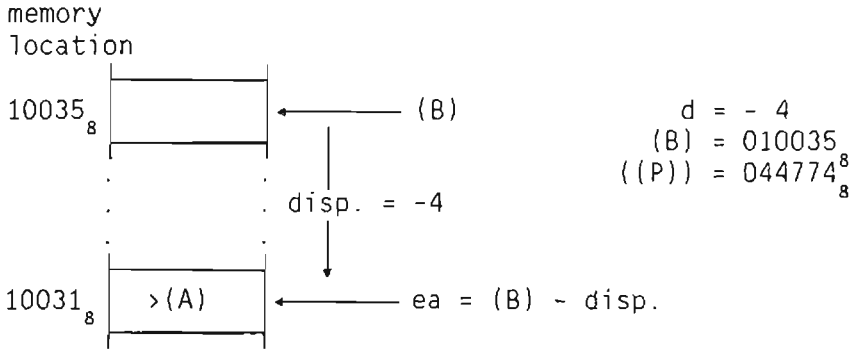


Note: d may have any value in the range -128 to 127.

Example:

LDA -4,B (instruction code 044774_8)

Load the contents of a memory location into the A register. The effective address location is the contents of the B register minus the value of the displacement (= 4).



P INDIRECT ADDRESSING

,X=0
I=1
,B=0

Effective Address = ((P) + disp)

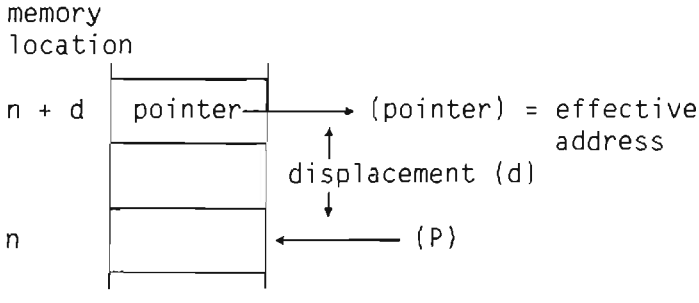
Description:

The contents of the P register (program counter) are added to the value of the displacement to find the indirect address (pointer).

If memory management is ON, the normal page table (PT) is used to convert the indirect address to a physical address.

The 16-bit word pointed to by the indirect address is the effective address for the operation.

If memory management is ON, the alternate page table (APT) converts the effective address to a physical address.

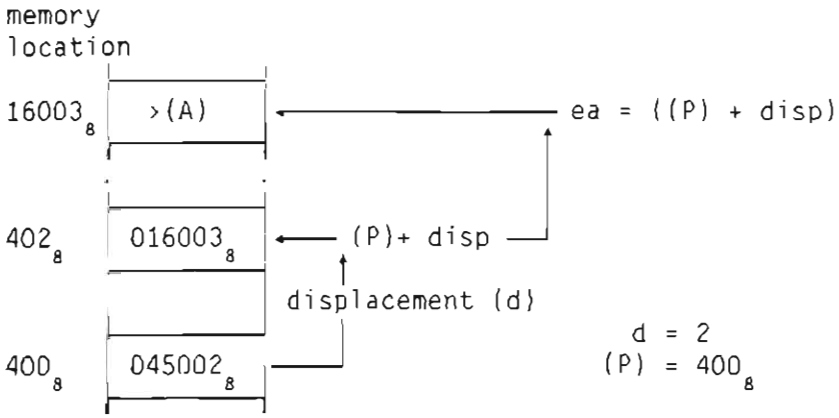


Note: d may have any value in the range -128 to 127.

Example:

LDA I *2 (instruction 045002_8)

Load the contents of the effective address into the A register. The effective address is the contents of the memory location two words (d = 2) ahead of the current instruction.



B INDIRECT ADDRESSING

,X=0
I=1
,B=1

Effective Address = ((B) + disp)

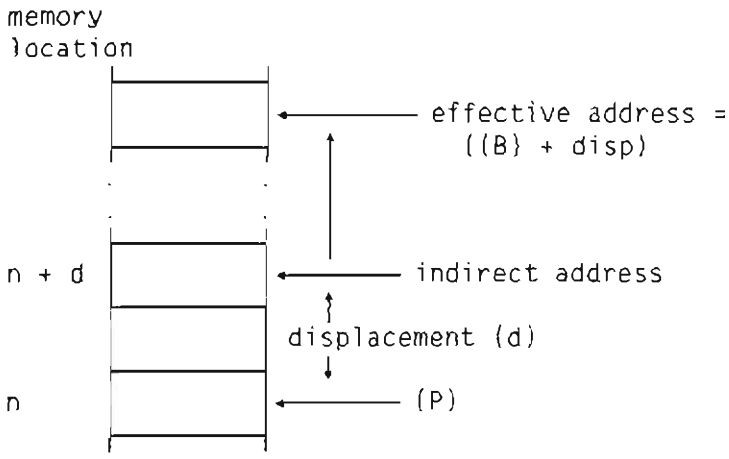
Description:

The contents of the B register are added to the value of the displacement to form the indirect address. The 16-bit word fetched from this location is the effective address for the operation.

If memory management is ON, the alternate page table (APT) will be used to convert both the indirect and effective addresses to physical addresses.

NOTE:

Indirect addressing adds one extra memory access to the execution time of the instruction.



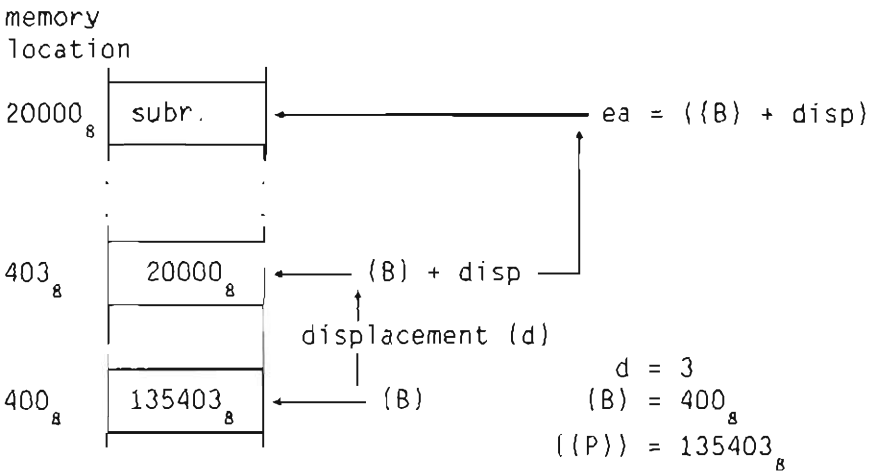
Note: d may have any value in the range -128 to 127.

Example:

JPL I 3,B (octal code for instruction 135403)

The contents of the B register plus the value of the displacement point to the memory location which contains the effective address.

The instruction saves the contents of the P register (program counter) in the L register and loads the P register with the effective address. This results in the next instruction (marked subr. in the diagram below) being fetched from the effective address.



X RELATIVE ADDRESSING

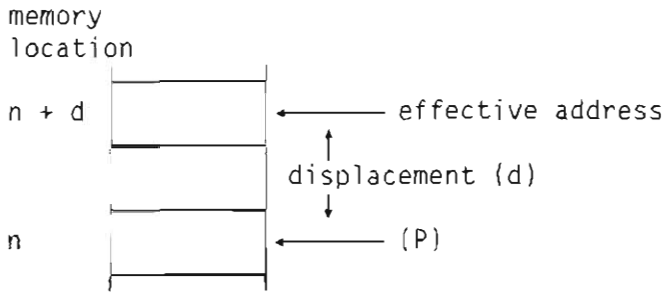
,X=1
I=0
,B=0

Effective address = (X) + disp

Description:

The effective address is calculated by adding the value of the displacement to the contents of the X register.

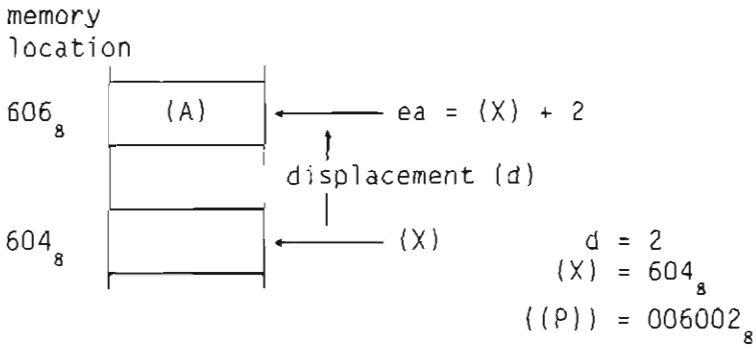
If memory management is being used, the alternate page table (APT) is used to convert the effective address to a physical address.



Note: d may have any value in the range -128 to 127.

Example: STA 2,X (instruction code 006002_8)

Store contents of X register in the memory location two words ahead of this instruction.



B INDEXED ADDRESSING

,X=1
I=0
,B=1

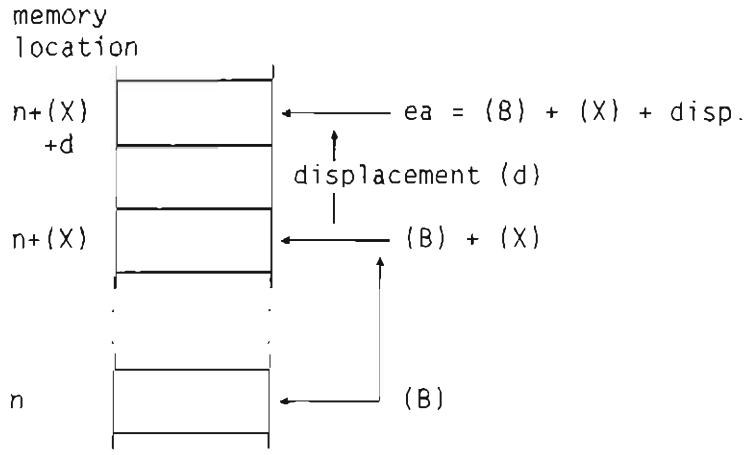
Effective address = (B) + (X) + disp

Description:

The effective address is calculated by adding the contents of the B register to the contents of the X register, and then adding the result to the value of the displacement.

If memory management is being used, the alternate page table (APT) will be used to convert the effective address to a physical addresses.

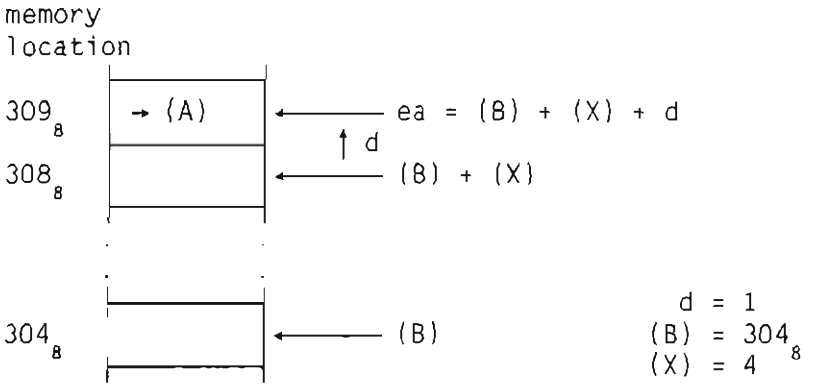
Note: This addressing mode adds one extra microcycle to the execution time of the instruction.



Note: d may have any value in the range -128 to 127.

Example: LDA 1,B ,X (instruction code 046401_8)

Load the contents of the memory location into the A register. The effective address is the contents of the B and X registers added together plus the displacement (= 1).



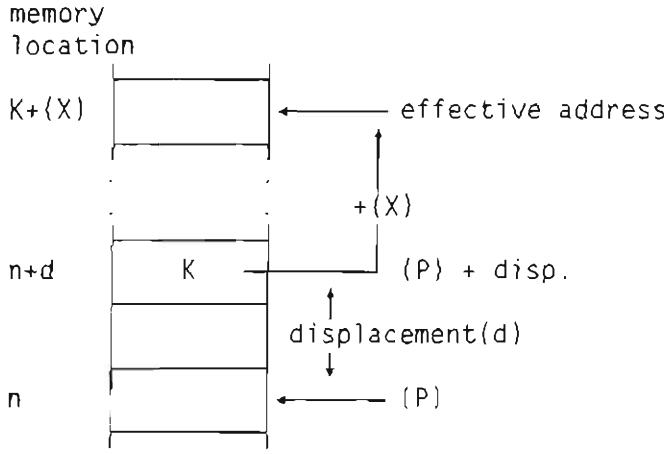
P INDIRECT INDEXED ADDRESSING

,X=1
I=1
,B=0

Effective address = ((P) + disp) + (X)

Description: The displacement value is added to the contents of the P register to determine an indirect address. The 16-bit word at this location is added to the contents of X (index) register to find the effective address. The indirect address can be used as a base pointer to a block of memory with (X) the index.

NOTE: Indirect addressing adds one extra memory access to the execution time of the instruction.



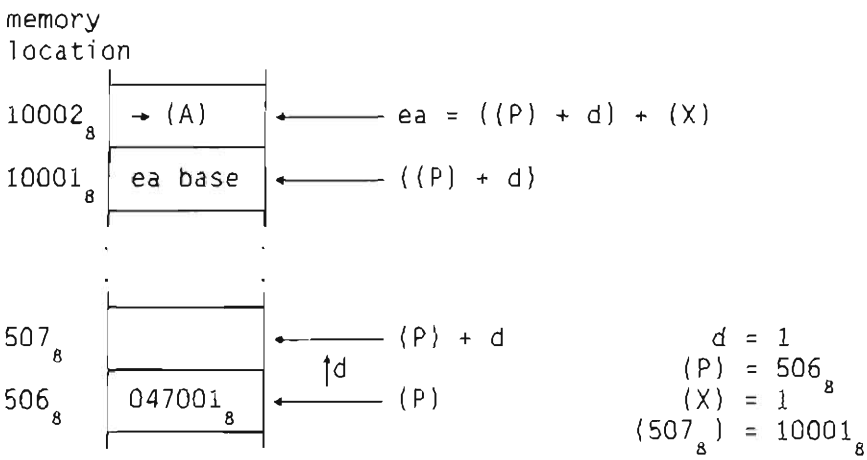
Note: d may have any value in the range -128 to 127.

Example:

LDA ,X I *1 (instruction code 047001_8)

The contents of the P register (program counter) are added to the value of the displacement (= 2) and the value fetched is used as the effective address.

The contents of the effective address are loaded into the A register.



B INDIRECT INDEXED ADDRESSING

,X=1
I=1
,B=1

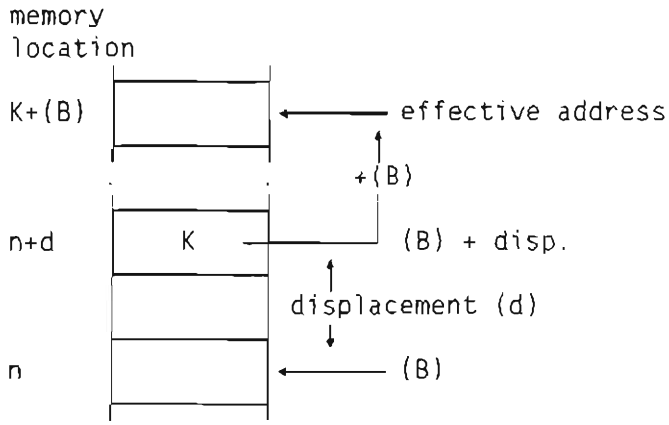
Effective address = ((B) + disp) + (X)

Description:

The value of the displacement is added to the contents of the B register to form an indirect address. The 16-bit word at this location is added to the contents of X (index) register to find the effective address. The indirect address can be used as a base pointer to a block of memory with (X) the index.

If memory management is being used, the alternate page table (APT) will be used to convert the effective address to a physical addresses.

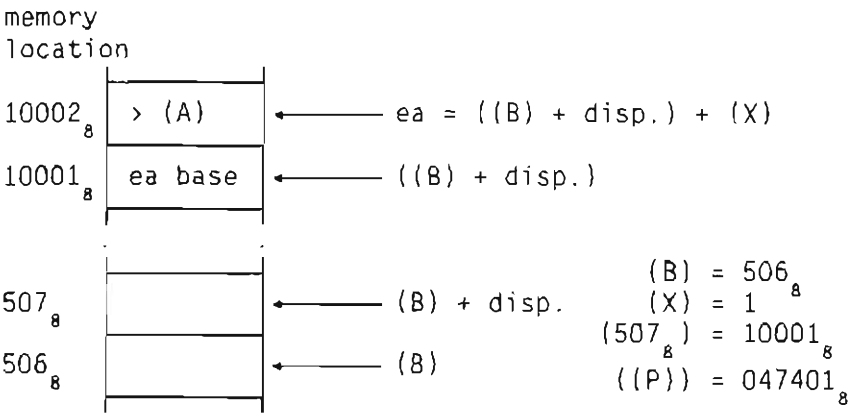
NOTE: Indirect addressing adds one extra memory access to the execution time of the instruction.



Note: d may have any value in the range -128 to 127.

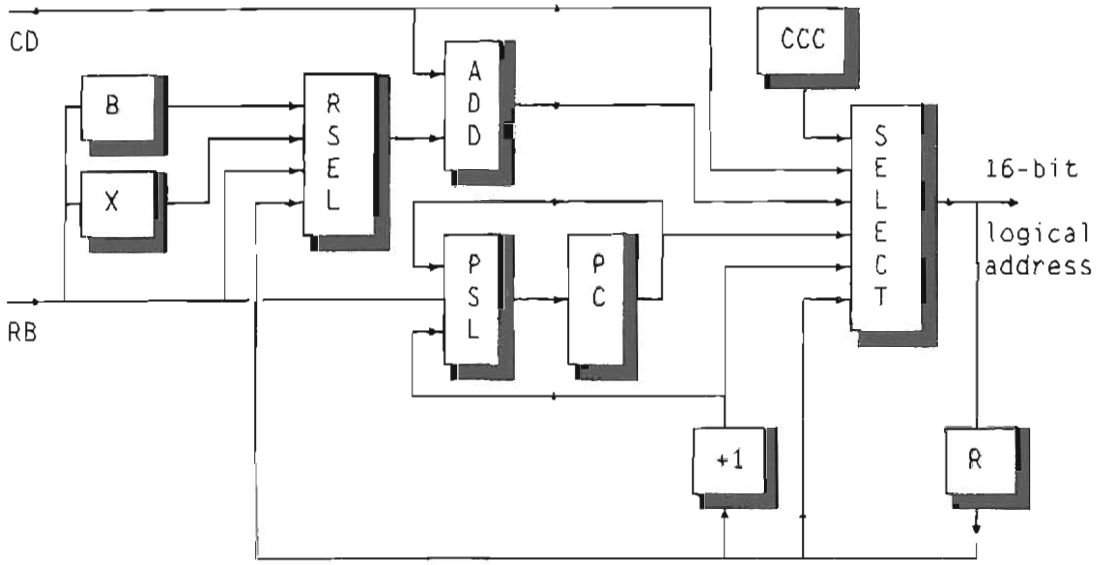
Example: LDA ,X I ,B *1 (instruction code 047401_8)

Load the contents of the effective address into the A register. The the contents of the B register plus the displacement (= 2) is the indirect address (507_8). The indirect address contains the base address (10001_8) to which is added the contents of the index register (X = 1) to form the effective address.



2.8.3 Principles of Address Arithmetic

The ND-110 CPU performs address arithmetic in the RMAC gate array. This function was performed by the microprogram in the ND-100. RMAC contains an adder and selection logic as well as copies of three important registers; B, X and P.



B : (copy of) B register
CCC : Clear cache count
CD : CD bus
PC : (copy of) P register (program counter)
R : the previous address
RB : bus from BUFALU
X : (copy of) X register

Figure 24. RMAC address-arithmetic gate array

RMAC operates one microcycle ahead of BUFALU in the pipeline. This means that addresses are ready for BUFALU to use without delay.

RMAC operation

RMAC calculates a memory address by adding a displacement to the base address. The displacement comes from the current instruction (see instruction format on page 59.). The base address comes either from the registers B, X and P or a memory location.

A multiplexer (RSEL) selects the base address from the four possible sources:

- B (a local copy of the B register in RMAC)
- X (a local copy of the X register in RMAC)
- R, a register in RMAC which contains the previous value of P (program counter).
- The RB bus from BUFALU. This source is used for B-relative indexed addressing (,B,X mode). BUFALU adds the X and B registers and sends then to RMAC via the RB bus.

The address adder (ADD) performs 2's complement addition on the operands:

- The register operand from RSEL.
- The CD bus. This can contain an 8-bit displacement from the current instruction or the result of a memory read cycle (indirect addressing).

The main multiplexer (SELECT) selects the address to be used from six possible sources.

- ADD, the address adder.
- PC, the program counter. (local copy of P register)
- R, a register in RMAC which contains the previous value of P (program counter).
- [+1], the incrementer. This contains (the previous value of P) + 1.
- CD, the internal bus. This contains the result of a memory read operation during the second RMAC cycle of an indirect indexed memory operation.
- CCC, cache clear count.

HOW THE EIGHT ADDRESSING MODES ARE HANDLED

The following table shows how the eight different addressing modes are handled. RMAC operates one μ-cycle ahead of the ALU.

B-relative indexed addressing uses the ALU to add X + B. This is done in the first ALU μ-cycle, that is, the second RMAC μ-cycle. The result from RMAC is ready in time for the second ALU μ-cycle, which uses the address for the memory access.

Addressing mode	Mnemonic	1st μ-cycle in RMAC		2nd μ-cycle in RMAC
P-relative		R + CD sign[1]		-
Ind P-rel	I	R + CD sign[1]		CD[2]
B-rel	,B	B + CD sign[1]		-
Ind B-rel	,B I	B + CD sign[1]		CD[2]
Indexed	,X	X + CD sign[1]		-
B-rel Indexed	,B,X	(X + B)[3]	⟶	RB[4] + CD sign[1]
Ind P-rel Indexed	I,X	R + CD sign[1]		X + CD[2]
Ind B-rel Indexed	,B I,X	B + CD sign[1]		X + CD[2]

Note: 1) The CD bus contains the sign extended displacement
2) The CD bus contains the result of the first memory read
3) X and B are added in BUFALU not in RMAC. This is done in the same μ-cycle as the RMAC operation.
4) RB contains X + B from BUFALU

Table 4. RMAC address operations

Sign extension is performed by setting bits 8 to 15 of the displacement equal to bit 7.

When a new instruction is to be fetched, the P register (program counter) is incremented by one before use. When a P relative read or write is performed, the pre-incremented copy of P (in R) is used.

Indirect addressing will always use two memory cycles. The first memory access fetches the address that is to be used by the second memory cycle (read or write).

B relative indexed (,B,X) addressing uses the ALU to add the contents of the B and X register in the first microcycle.

The actual time used for a memory cycle depends on whether cache is used. Memory read may use cache (if the address is in cache), but memory write operations always result in a ND-100 bus operation (write through cache). Cache memory operation is described in section 5.4 (page 115).

CHAPTER 3 MEMORY MANAGEMENT SYSTEM

CHAPTER 3 MEMORY MANAGEMENT SYSTEM

Memory management is needed to run the SINTRAN III/VSX (Virtual Storage) operating system. This is now standard on all ND-110 computers.

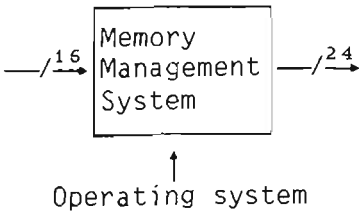
An ND-110 running SINTRAN III/VSX offers the following features:

- Two segments, each with 64 Kwords (128 Kbyte) virtual address range for each user, independent of physical memory capacity
- dynamic allocation/relocation of programs in memory
- memory protection
- paging mechanism

3.1 VIRTUAL ADDRESS SPACE

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

In order to implement virtual storage, an "intelligent" addressing translation mechanism must be employed. This mechanism is under the control of the operating system. Programs are written for a virtual machine with 64 Kword storage. The operating system (SINTRAN III) uses the memory management to translate virtual addresses into physical addresses.



In ND-110 systems up to 16 Mwords of physical memory may be used; a 16-bit virtual address will therefore be translated into a 24-bit physical address.

DYNAMIC ALLOCATION

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

Dynamic relocation

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

MEMORY PROTECTION

Memory protection is not attached to predefined memory areas, but to the program and will follow it as it is moved around.

No external fragmentation

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

More parallelism

Also as a result of the paging system, current parts of a given program only reside momentarily in primary storage. This gives room for more programs to be executed in parallel (multi-processing).

System overhead

Data transport to and from mass storage during paging is slow compared to the speed of the processor. The time used for this task increases system overheads and thereby reduces the CPU time available for user programs. ND-110 contains new instructions, used by SINTRAN III/VSX version K and later, which increase system efficiency.

3.2 Paging and Protection System

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

- Paging system
- Memory protection system

The implementation of paging is based on dividing physical memory into 1 Kword pages which, under operating system control, are assigned to active programs. Data and instruction pages may be allocated anywhere in memory without restriction.

Paging system

The paging system can work in three different modes:

- "normal" (four page tables)
- extended (four page tables)
- extended (sixteen page tables)

"Normal" mode, which is compatible with the NORD-10 paging system, uses four page tables (PTs) to map the 16-bit virtual address into a 19-bit physical address, extending the physical address space from 64 K to 512 Kwords (128 K to 1 Mbyte). Despite its name "normal" mode is now used only for compatibility with older programs.

"Extended" mode, which covers an address range of 16 Mwords, maps the 16-bit virtual address into a 24-bit physical address. Extended mode may use either four or sixteen page tables for the mapping process. All new programs for the ND-110 are written for 16 PT extended mode.

Memory protection system

The memory protection system may be divided into two subsystems:

- The page protection system
- The ring protection system

Page protection

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

Ring protection

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

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

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

CONNECTION TO CPU

Unlike the memory management system for the ND-100 CPU the ND-110 memory management system is entirely on the CPU card and all connections to the CPU are contained within the CPU card.

3.3 MEMORY MANAGEMENT ARCHITECTURE

Memory management consists of:

- 16 page tables
- 16 paging control registers
- Paging status register
- Page protection system
- Ring protection system
- Memory map table
- segment map table

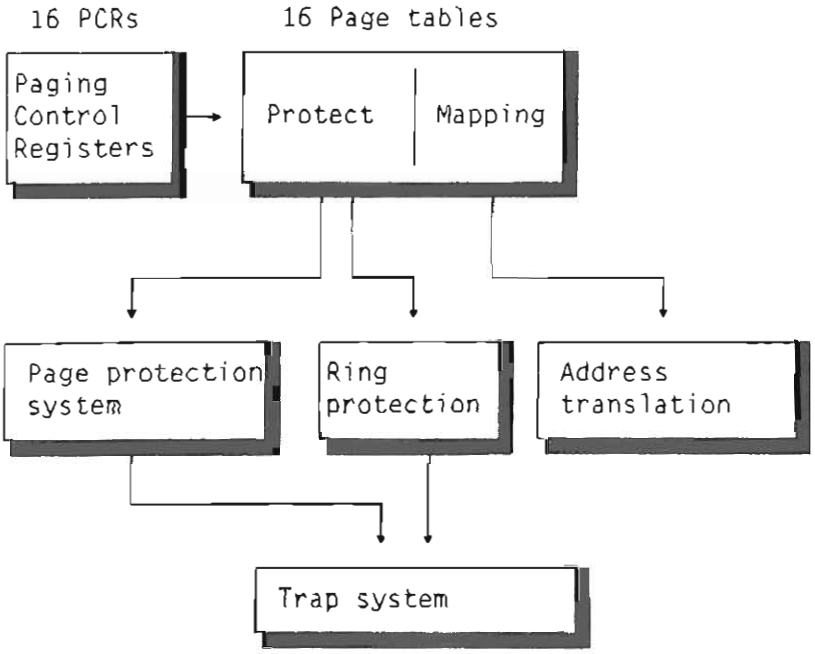


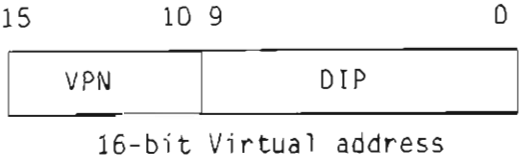
Figure 25. Memory Management Building Blocks

3.3.1 Virtual to Physical Address Mapping

The paging system maps the 16-bit virtual address from the address arithmetic into a physical address. The number of bits in the physical address depends on whether the "normal" (19-bit NORD-10/S compatible) or the "extended" (24-bit) addressing mode is used. In the following explanations, the extended mode, which is the mode most used by ND-110 systems, is used.

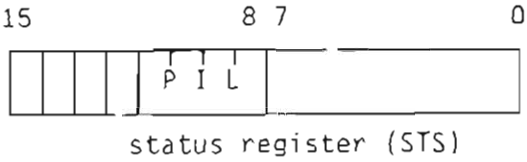
The paging system divides the memory into blocks of 1 K (1024) words. These blocks are referred to as pages. The page tables contain pointers to these pages.

The CPU sends a 16-bit virtual address to the memory management system. The memory management system translates this to a 24-bit physical memory address (in "normal" mode, 19-bit physical address).



16-bit Virtual address

To address any location within a 1 K address space, 10 address bits are required. These bits are referred to as the displacement within a page (DIP), and are transferred directly to the ND-100 bus.



The most significant part of the virtual address (bits 10-15) is used as an address selecting one of 64 locations in the page table. These six bits are referred to as the Virtual Page Number (VPN).

The program level (PIL) from the status register (STS) determines which of the 16 paging control registers (PCR) to use. Two fields in the PCR determine which page table is to be used for P-register relative addressing and which page table is to be used for other addressing modes. (See the section on addressing modes on page 62 and the table on page 84)

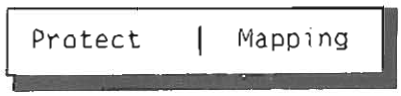


Figure 26. A page table entry

The virtual page number (VPN) addresses an entry in the selected PT. A page table entry may be thought of as being divided into two parts. The Mapping part contains the physical page number (PPN), and the protect part contains information about the access rights assigned to that page.

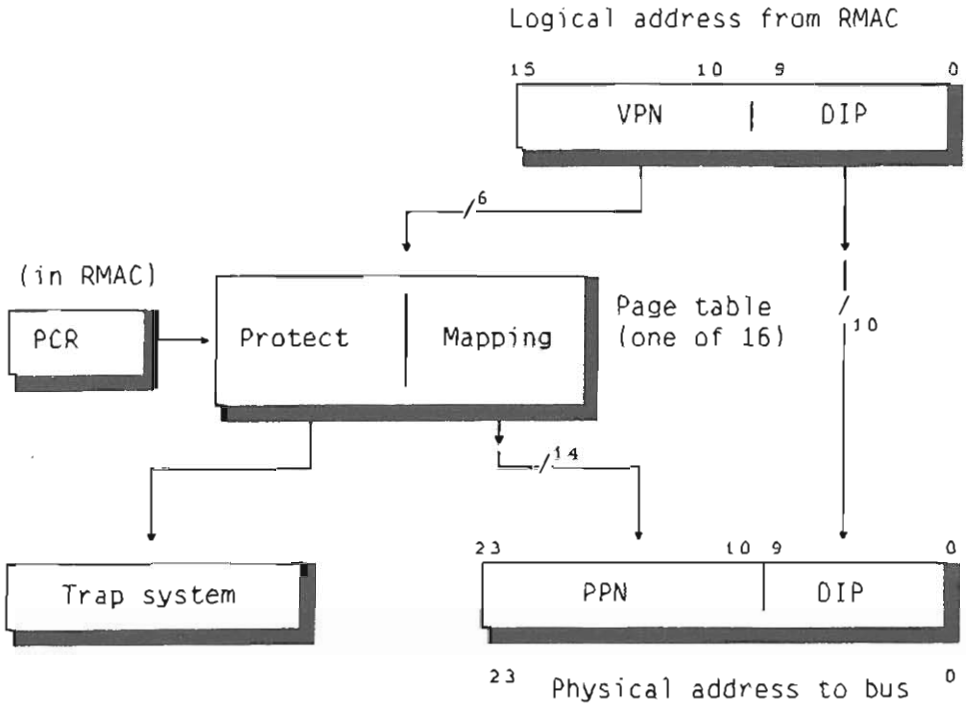


Figure 27. Virtual to physical address mapping

The fourteen-bit mapping field (9-bit for "normal" mode) in this page table entry is called physical page number (PPN). This is used as the upper 14 bits (bits $_{11-23}$) of the physical address. The lower ten bits are the displacement in page (DIP) which pass through unchanged.

The 14-bit PPN can have values in the range 0-16384 (0-511 for the 9 bit PPN in "normal" mode). It is thereby possible to access 16384 · 1 K (= 16 M) words physical memory in extended mode. In "normal" mode the corresponding addressing area is 512 Kwords.

The seven-bit protect field of the page table entry is used by the protection system and is described later (page 85).

Prior to program start, the operating system (SINTRAN III) must set the values of the protect and the mapping fields in the page table.

3.3.2 Page Table Selection

Programs running on an ND-110 may use up to two of the sixteen page tables at any time. If program counter (P) relative addressing is used, the standard page table (PT) is used. The alternate page table (APT) field is used for B relative and X relative memory references. Note that indirect addressing involves 2 memory references, where one may go via the PT and the other via the APT, or both via the APT.

The paging control register (PCR) contains fields which specify which page tables are assigned as PT and APT. The PCR register contains information for the currently active program level (specified by the PIL field in the STS register). PCR's for the other program levels are stored in the extended register file (XRF) and loaded automatically during a level change.

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

Table 5. Page table use and addressing mode

The main principle is that all P relative memory references are mapped via PT, and all other references via APT. This feature is used by processes which require access to two segments (two bank programs) with different virtual address spaces, giving the one process access to 128 Kword of virtual memory instead of 64 Kword. In one bank programs PT and APT will both point to the same page table, so that P relative accesses will use the same page table as other accesses.

3.3.3 Page Table Assignment

SINTRAN III - VSX version K assigns page tables according to the following table.

Page table	Mnemonic[1]	Usage
0_8	POF	SINTRAN start and restart
1_8	UPITN	users normal page table
2_8	UPITA	users alternate page table
3_8	FUPIT	remote file user page table
4_8	FPIT	file system
5_8	5PIT	Monitor call 60_8, ND-500 monitor
6_8	XPIT	Xmsg
7_8	DPIT	resident common data, RT descriptions
10_8	RPIT	Monitor calls, resident code
11_8	SPIT	SINTRAN, RT loader, DMAC
12_8	MPIT	Monitor PT, interrupts (level 14)
13_8	X5DPT	ND-500 name and standard domain
14_8		
15_8		
16_8		
17_8	DTPIT	user direct tasks

Note 1: Mnemonics refer to Sintran III documentation

Table 6. Page Table Assignments

The reader is referred to Sintran III release information ND-60.230 for further explanation of page table usage.

3.3.4 MEMORY PROTECTION SYSTEM

The memory management system employs two memory protection systems: a page protection system and a ring protection system. The two systems complement each other to provide extensive memory protection.

The memory protection system works on 1 Kword pages. If a memory access violates any of the protection systems, a trap to program level

14 will occur with the internal interrupt code equal to 2 (= MPV) (memory protect violation).

The two protection systems are independent, and that both the individual memory protection mode and the ring mode must be satisfied before an operation is allowed to proceed.

3.3.5 Layout of Page Tables

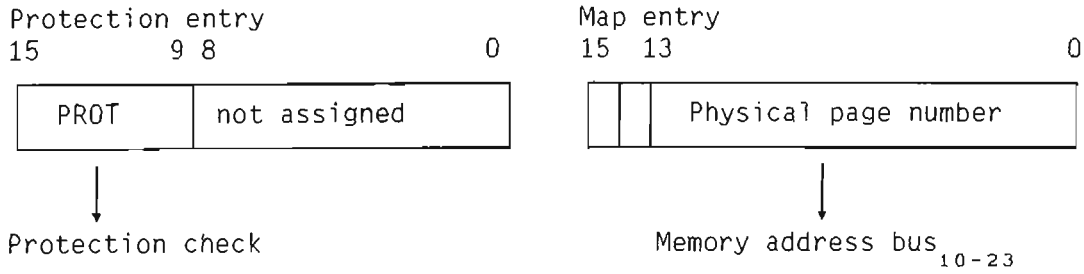


Figure 28. Layout of an entry in the page table (16 PT mode)

In the following, it is important to separate the view of the page tables as seen from program (as shadow memory) from how things work physically during the paging process. In this section paging is described from the hardware viewpoint.

The paging process is the same for normal, extended (4 PT) and extended (16 PT) modes. In section describing shadow memory, on page 97, the page tables are described from the programmer's viewpoint. There the three modes look quite different.

The paging process begins with lookup of an entry selected by the 6 bit VPN (see figure 27.). This entry is 32 bits long for both modes and consists of two parts, protect and map as shown in the figure above.

Each page has an associated entry in the page table which describes precisely what to do when the program uses that page.

The map part of a page table entry contains the 14-bit page address in physical memory where the page is stored. The address within a page (the 10 least significant bits of the address) is used unaltered. This is the

displacement in page (DIP) shown in figure 27. Together they make the 24 bits needed to address 16 Mwords (32 Mbyte).

When "normal" mode is used, the physical page number is only 9 bits. The 5 most significant bits of the physical address are set to zero. This allows access to the first 512 Kwords of memory (NORD-10/S mode).

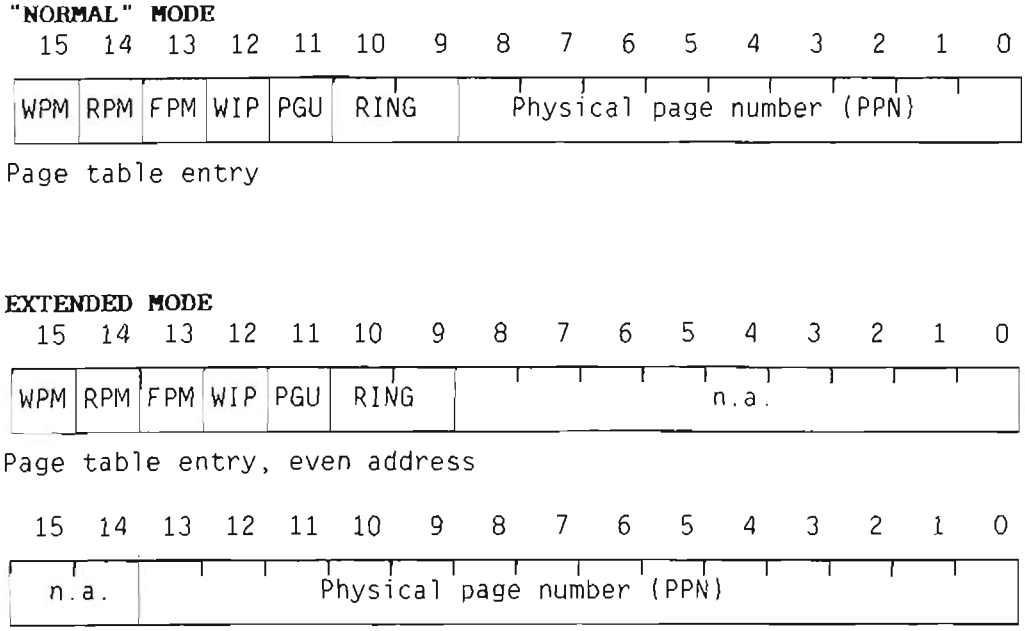


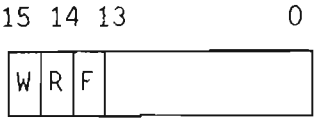
Figure 29. Page table entry

3.4 Page Protection System

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

- read access
- write access
- instruction fetch access

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



The read, write and fetch protect system is implemented by defining, in bits 13 - 15 of the even word of a table entry, how the page may be used. In hardware, this information is compared with the instruction being executed, i.e. if it is a read, write, indirect address operation or instruction fetch.

The three bits from a table entry have the following significance:

Bit 15: W

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

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

If an attempt is made to write into a write protected page, an trap to program level 14 will occur, and no data will be written.

Bit 14: R

R = 0 Locations in this page may not be read (but they may be executed if the ring bits allow it).

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

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

Bit 13: F

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

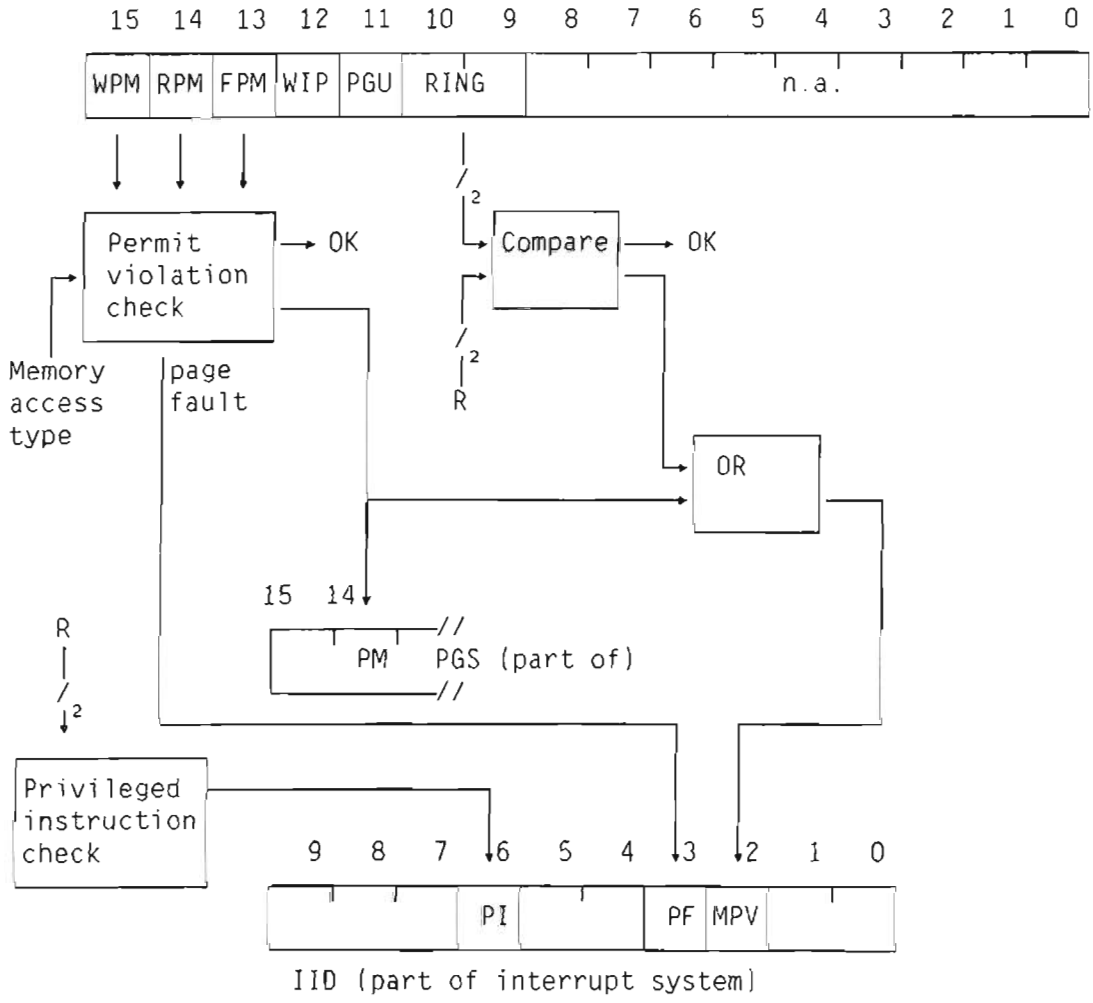
If an attempt is made to execute in fetch protected memory, a trap to program level 14 will occur and the execution is prevented.

NB!
Indirect addressing

Indirect addresses may be taken both from pages which have F = 1 and from pages which have R = 1.

All combinations of W, R and F are permitted.

However, the combination where W, R and F are all zero is interpreted as page not in memory and will generate an internal interrupt as a page fault.



IID : internal interrupt detect register
PM : permit violation
PI : privileged instruction violation
PF : page fault
PGS : paging status register
MPV : memory protect violation
R : ring number

Figure 30. Memory protection

3.5 RING PROTECTION SYSTEM

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

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

PCR bits 0 0 → Ring 0: Programs executed from this ring may not execute privileged instructions. They may only access locations in ring 0. Locations outside ring 0 are completely inaccessible.

PCR bits 0 1 → Ring 1: Programs executed from this ring may not execute privileged instructions. They may access locations in ring 1 and ring 0.

PCR bits 1 0 → Ring 2: All instructions are permitted on this ring. Programs executed from this ring may access locations in program 2, 1 and 0.

PCR bits 1 1 → Ring 3: All instructions are permitted and the whole address space, including the page tables, is accessible if not otherwise protected by the RPM, WPM and FPM bits.

The rings limit the privileges of a program and thereby its user.

Ring 3 programs have no limitations imposed on them by the ring system (but may still be limited by the paging system). Only trusted programs can be allowed to operate in ring 3. Sintran III monitor kernel operates in ring 3.

At the other level, ring 0 programs have access only to areas in ring 0. User programs are operated in ring 0. When they need to access areas outside their ring, they must use monitor calls to the operating system. This forces all such accesses through the operating system which thereby can maintain system integrity.

An illegal ring access or illegal use of privileged instructions will cause an internal interrupt on level 14, and the forbidden action will be avoided.

> **Note**
> This degrading only occurs when lower ring instruction codes are executed, but not when data is accessed.

If a program in ring 3 executes instructions assigned to rings 0, 1 or 2, its ring number is reduced accordingly. Such accesses are detected by hardware which automatically changes the ring number in the PCR register for the current program level.

Ring Assignment

The recommended way of using the rings is:

Ring 0: Timesharing users

Ring 1: Compilers, assemblers, data bases

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

Ring 3: Kernel of operating system

This may be visualised in the following manner.

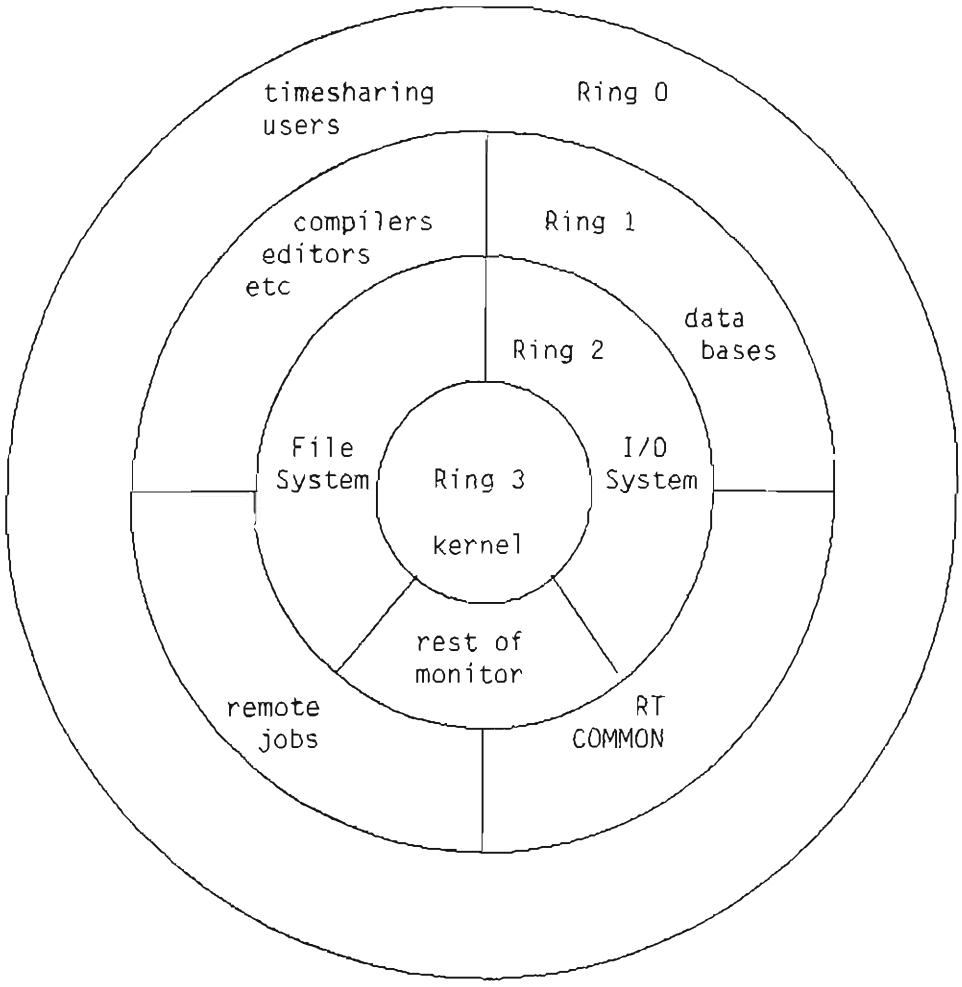


Figure 31. Ring Assignment

3.5.1 Privileged Instructions

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

Privileged Instructions:

- input/output instructions
- all instructions which control the memory management and interrupt system
- inter-program level communication instructions

Refer to the ND-110 Instruction Set ND-06.029 for further information.

The only instruction the user has available for user/system communication is the monitor call instruction MON. The MON instruction takes an unsigned eight bit number (0 - 256) as its parameter.

When the ND-110 executes the MON instruction, it generates an internal interrupt to level 14. The eight-bit parameter is sign extended to 16 bits and loaded into the T register of program level 14. The level 14 routine may use this value to branch to the appropriate code.

See the SINTRAN III Monitor Calls manual (ND-60.228) for details of the monitor calls defined in SINTRAN. Note that SINTRAN reserves 8 monitor calls (MON170 to MON177) for user defined routines.

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

3.6 Page Used and Written in Page

Entries in a page table are under program control only, except for the two bits PGU and WIP, which are also controlled automatically by the trap system.

Bit 12: WIP written in page

If this bit is set (= 1), the page has been written in, and should be written back to mass storage if the physical page is needed for another program. If it is zero, the page has not been modified and need not be written back. This bit is automatically set to one the first time a write occurs, and then remains set. It is cleared by program (whenever a new page is brought from mass storage).

Bit 11: PGU page used

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

3.7 Memory management control and status

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

PON

Turn on the management system (paging on).

The instructions that are executed after the PON instruction will go through the address mapping (paging) mechanism. The ring protection system will also be turned on by this instruction.

> Note
> Programs executing in ring 3 will access shadow memory even when paging is on. All other rings access main memory.

POF Turn off the management system (paging off).

The instruction will turn off the memory management system and the following instructions will be taken from a physical address in the lowest 64 K.

> Caution
>
> The machine will then be in an unrestricted mode without any hardware protection feature, i.e. all instructions are legal and all memory accessible. When paging is off, the ring protection system is also off. Shadow memory will be accessed instead of main memory.

THE SEX AND REX INSTRUCTIONS

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

SEX Set extended address mode

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

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

REX Reset extended address mode

The REX instruction will reset the extended address mode (24 bits) to normal address mode (19 bits). This implies that 512 Kwords of physical address space is now available. It also implies four page tables.

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

This mode is compatible with NORD-10/S.

NB! Changing the number of page tables changes the size of shadow memory. See page 97 for details of shadow memory addresses. After a change of mode, the page tables must be initialized before turning paging on again.

3.7.1 Paging Control Register

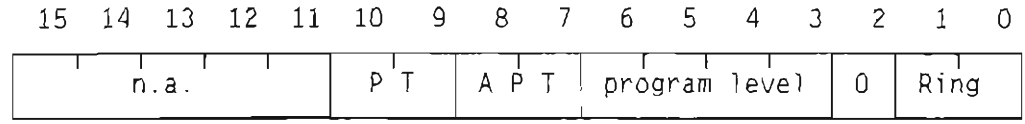
> **Note**
> The PCR registers are not cleared during power on initialisation. This must be done by program before executing a PON instruction.

The PCR (paging control register) is a register inside RMAC. There is a copy of PCR for each program level. These copies are stored in the extended register file (XRF). The PCR register is loaded from XRF at the same time as the register set is loaded from the register file.

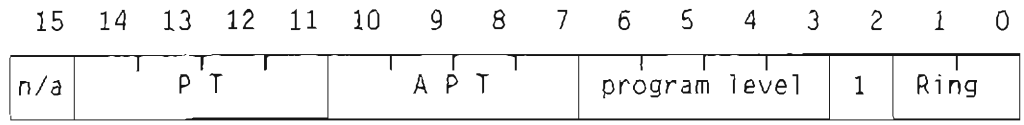
The instructions TRR PCR and TRA PGC allow the programmer to write to the PCR or read back its contents. These instructions can read and write to the (copy of the) PCR on any program level.

One PCR may be written into at a time, by the instruction TRR PCR.

This instruction uses the contents of the A register which must use one of the following formats.



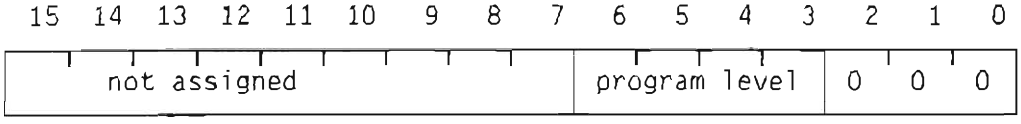
Four page table mode



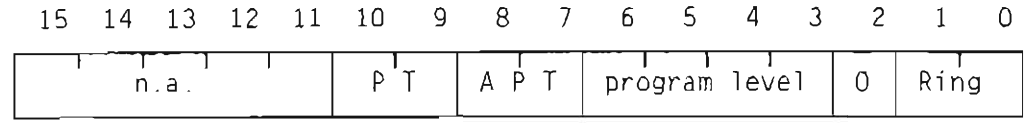
Sixteen page table mode

Figure 32. TRR PCR instruction, A register format

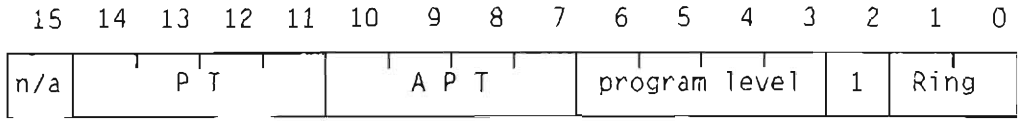
It may be desirable to read back the contents of the 16 PCRs. This is done with the TRA PGC instruction. The A register must be have the following format:



After execution the contents of the A register will be:



Four page table mode



Sixteen page table mode

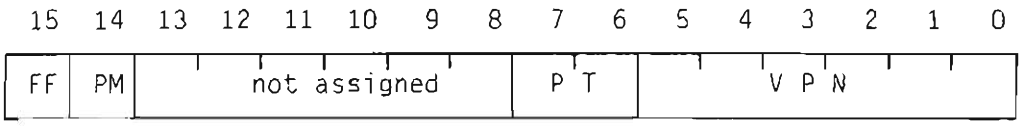
Figure 33. TRA PGC instruction, A register format

3.7.2 Paging Status Register

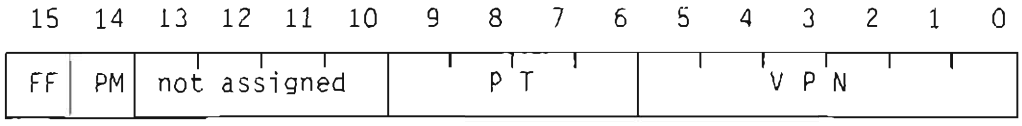
Whenever the memory management system reports errors (page fault, memory protection violations), the operating system is alerted through an internal interrupt with the interrupt code equal to the error source. The operating system then reads the paging status register for further information. The paging status register is used for further specifications when a page fault or a memory protection violation occurs.

The instruction TRA PGS is used to read this register. Errors lock the register, reading the register with TRA PGS unlocks it again.

The bits in PGS have the following significance:



Four page table mode



Sixteen page table mode

Figure 34. PGS Format

Bit 15: FF = fetch fault

Memory management interrupt occurred during an instruction fetch.

Bit 14: PM = permit violation

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

0 = ring protection violation interrupt

Permit violation has priority if both conditions occur.

PT:

Page table number. The page table that was in use when the violation occurred.

VPN:

Virtual page number. This points to the page table entry that caused the violation.

In "normal mode" it is the 8 least significant bits of the shadow address (see page 97).

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

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

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

3.8 Control of page tables

The operating system (SINTRAN III) manages the page tables. Some parts are fixed from system start time, and others are dynamically changed and updated. When a new program is started, the operating system examines an administration table to see which pages are free.

The map part of the page table is filled with

the physical page numbers (PPN) of the allocated pages. Pages are taken from other processes if necessary. The protect part of the page table entry will contain the access rights of the program.

3.8.1 Shadow Memory

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

To ease reading and writing of the page tables, they are accessed as memory. The highest memory locations in the 64 K virtual address space are reserved for page table access.

The memory requirements are as follows:

```
normal mode   (4  PTs):  1 x 64 x  4 =  256 words
extended mode (4  PTs):  2 x 64 x  4 =  512 words
extended mode (16 PTs):  2 x 64 x 16 = 2048 words
```

The addresses are:

	Normal Mode: 4 page tables	Extended Mode 4 page tables	Extended Mode: 16 page tables
0	177400_8 - 177477_8	177000_8 - 177177_8	174000_8 - 174177_8
1	177500_8 - 177577_8	177200_8 - 177377_8	174200_8 - 174377_8
2	177600_8 - 177677_8	177400_8 - 177577_8	174400_8 - 174577_8
3	177700_8 - 177777_8	177600_8 - 177777_8	174600_8 - 174777_8
4	n/a	n/a	175000_8 - 175177_8
5	n/a	n/a	175200_8 - 175377_8
6	n/a	n/a	175400_8 - 175577_8
7	n/a	n/a	175600_8 - 175777_8
8	n/a	n/a	176000_8 - 176177_8
9	n/a	n/a	176200_8 - 176377_8
10	n/a	n/a	176400_8 - 176577_8
11	n/a	n/a	176600_8 - 176777_8
12	n/a	n/a	177000_8 - 177177_8
13	n/a	n/a	177200_8 - 177377_8
14	n/a	n/a	177400_8 - 177577_8
15	n/a	n/a	177600_8 - 177777_8

Table 7. Page table address in shadow memory

This area is called shadow memory because it lies in the shadow of main memory and is inaccessible for users on rings 0, 1 and 2. When paging is off, shadow memory is accessible and the corresponding area of main memory is inaccessible. Programs running in ring 3, however, always access shadow memory.

3.8.2 Reading and Writing in Page Tables

Normal and extended modes use two different formats. A page table entry is a 16-bit word in "normal" mode. Extended mode uses a 32-bit page table entry. This 32-bit word is written as two consecutive 16-bit words in shadow memory.

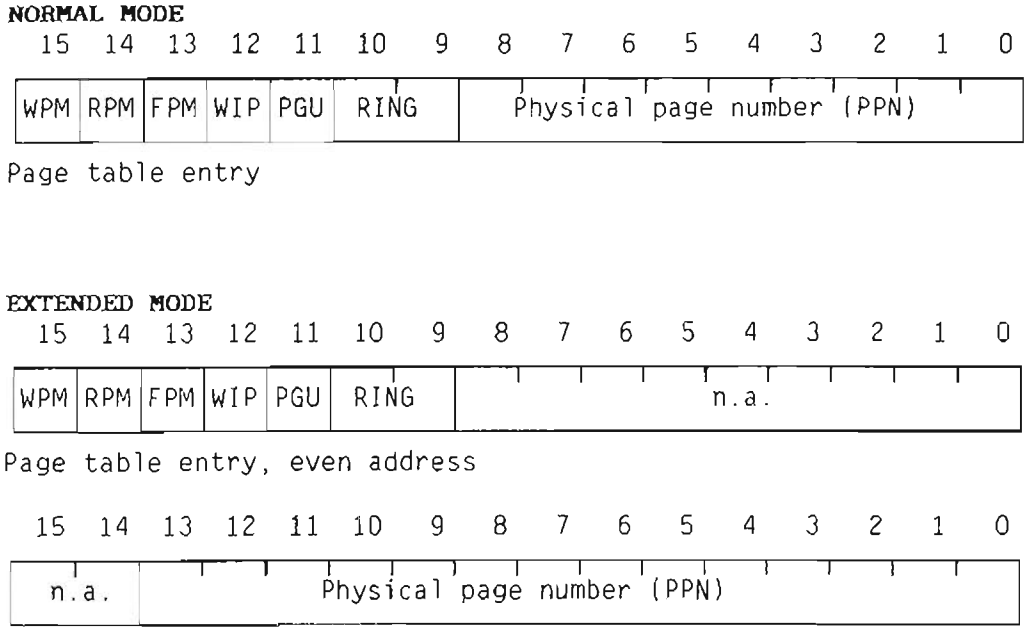


Figure 35. Page table entry

The reason for the unused bits in the page tables is that it shall be possible to read and write any contents in the tables without interpreting it as paging information. When paging is off, the page tables may be used as 2 Kword very fast random access memory. This memory may be used by test programs as it allows the CPU card to be tested with no functional memory available on the ND-100 bus.

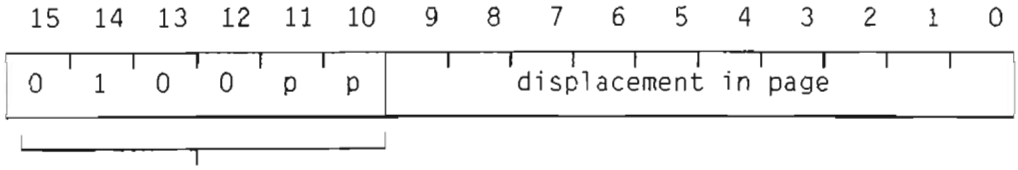
3.9 TIMING

Page table accesses are performed in parallel with cache memory lookup and consequently there is no timing overhead.

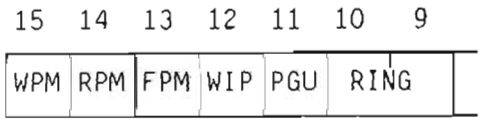
3.10 EXAMPLE OF PAGE TABLE USE

Example:

A user has a program occupying the 3K address area 40000_8 - 45777_8 : Addresses in this area will have the form:



pp may have the values 00_2, 01_2 or 10_2.



Page table entry, even address

The virtual page number may be 20_8, 21_8 or 22_8. User programs will be assigned page table 1 by SINTRAN (see page 85)

If we assume 16 page table extended mode, each page table entry consists of two consecutive words. The page table entries for page numbers 20_8, 21_8 and 22_8 are consequently found at addresses 40_8, 42_8 and 44_8 relative to the start of page table 1.

Seen as shadow memory this means the page table entries are:

174240_8 : Protect entry for virtual page 20_8
174241_8 : Mapping entry for virtual page 20_8

174242_8 : Protect entry for virtual page 21_8
174243_8 : Mapping entry for virtual page 21_8

174244_8 : Protect entry for virtual page 22_8
174245_8 : Mapping entry for virtual page 22_8

The three protect entries at addresses 40_8, 42_8 and 44_8 contain the bits WPM, RPM, FPM. SINTRAN sets these bits with the appropriate access rights.

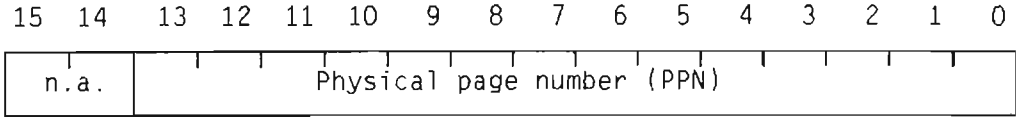
FPM (fetch permitted) must be set (= 1) as this is a program. In fact SINTRAN also sets WPM and RPM to 1 for user programs. WPM, RPM, FPM in the three words at 40_8, 42_8 and 44_8 will therefore be set to 1.

The written-in-page bit (WIP) will be set to 0 at the start and will not change unless at least one word is written to that page.

The PGU (page used) bit will initially be set to 0 by SINTRAN. It will be set to 1 as soon as an access (read write or fetch) is made to that page.

The RING field will contain the ring level assigned to the program. User programs are assigned to the lowest ring priority (= 0).

The odd address entries, 41_8, 43_8 and 45_8 contain the 14-bit physical page numbers.



Page table entry, odd address

The three physical pages pointed to by the page table entries need not be consecutive pages in physical memory. In fact, during normal operation, they will be scattered around in vacant areas of memory. This is transparent to the program however.

CHAPTER 4 ND-100 BUS SYSTEM

CHAPTER 4 ND-100 BUS SYSTEM

All system components and peripherals are connected to a high-speed parallel bus. The ND-110 uses exactly the same bus specification as the other members of the ND-100 family.

4.1 BUS CONTROL

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

- allocation of the ND-100 bus to a requesting bus user.
- supervising that the ND-100 bus is released by the allocated user within a certain time limit

The bus controller allocates the bus when:

- a memory-refresh cycle must be performed
- a DMA controller wants to transfer data
- the CPU wants to read or write data or fetch an instruction

If more than one source requests the bus at the same time, the bus control allocates in order of priority. A refresh request has the highest priority. DMA devices have higher priority than the CPU, except after a refresh operation, when the CPU is given highest priority. This guarantees the CPU access to the bus even during DMA operations.

4.2 PHYSICAL ARRANGEMENT OF THE ND-100 BUS

The ND-100 bus is implemented as a printed circuit backplane. Depending on the model, it may have 7, 12 or 21 card positions. The "C" connector is used for the ND-100 bus, leaving the "A" and "B" connectors free for other use (normally input/output).

Refer to Appendix B for ND-100 bus signal definitions. For a more detailed description of the bus signals and timing, refer to the ND-100 Bus Description manual, ND-06.017.02.

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

- 24-bit wide parallel multiplexed address/data bus
- control lines

The paired interrupt control lines, $\overline{\text{INCONTR}}$ - $\overline{\text{OUTCONTR}}$, $\overline{\text{INGRANT}}$ - $\overline{\text{OUTGRANT}}$, and $\overline{\text{INIDENT}}$ - $\overline{\text{OUTIDENT}}$ are connected $\overline{\text{OUTxxxx}}$ to $\overline{\text{INxxxx}}$ (daisy-chain). The card position code lines PA0 - PA3 are connected to +5V and 0V according to the following table.

PA0	PA1	PA2	PA3	Card position
0	0	0	0	rightmost[1]
0	0	0	1	rightmost - 1
0	0	1	0	rightmost - 2
		.		.
		.		.
binary sequence				

Note 1: Card position furthest from CPU

The PA0-3 signal lines were previously used by memory cards, but are now no longer used.

All other signal and power lines are independent of card position.

4.3 Organization of ND-110 card crate

ND-110 circuit cards are 366.8 mm high and 280 mm deep.

Every card has at least one 96 pin connector (the "C" connector) used for connection to the ND-100 bus. In addition, a card may have one or two additional 64 pin connectors ("A" and "B").

Connectors A and B are defined for each individual card that uses them. They are used

by I/O controllers for connections to external devices.

> **Note**
> Computers upgraded with ND-110 CPU card may use the vacant slot, previously occupied by the CPU, for newer types of memory cards. The installation description manual contains the part- and ECO-numbers of the cards which will function in slot 1.

The ND-110 CPU card is normally located in position number 2 in the card crate of a ND-100 machine or position number 3 for a ND-500 machine. This is the position used by the memory management card when the older ND-100 CPU card was used.

The first I/O card should be placed immediately to the right of the CPU card.

RULE:

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

If there are not enough card positions, a new card crate may be added. Connection between the crates is done with the help of a bus extender card.

4.4 Bus timing considerations

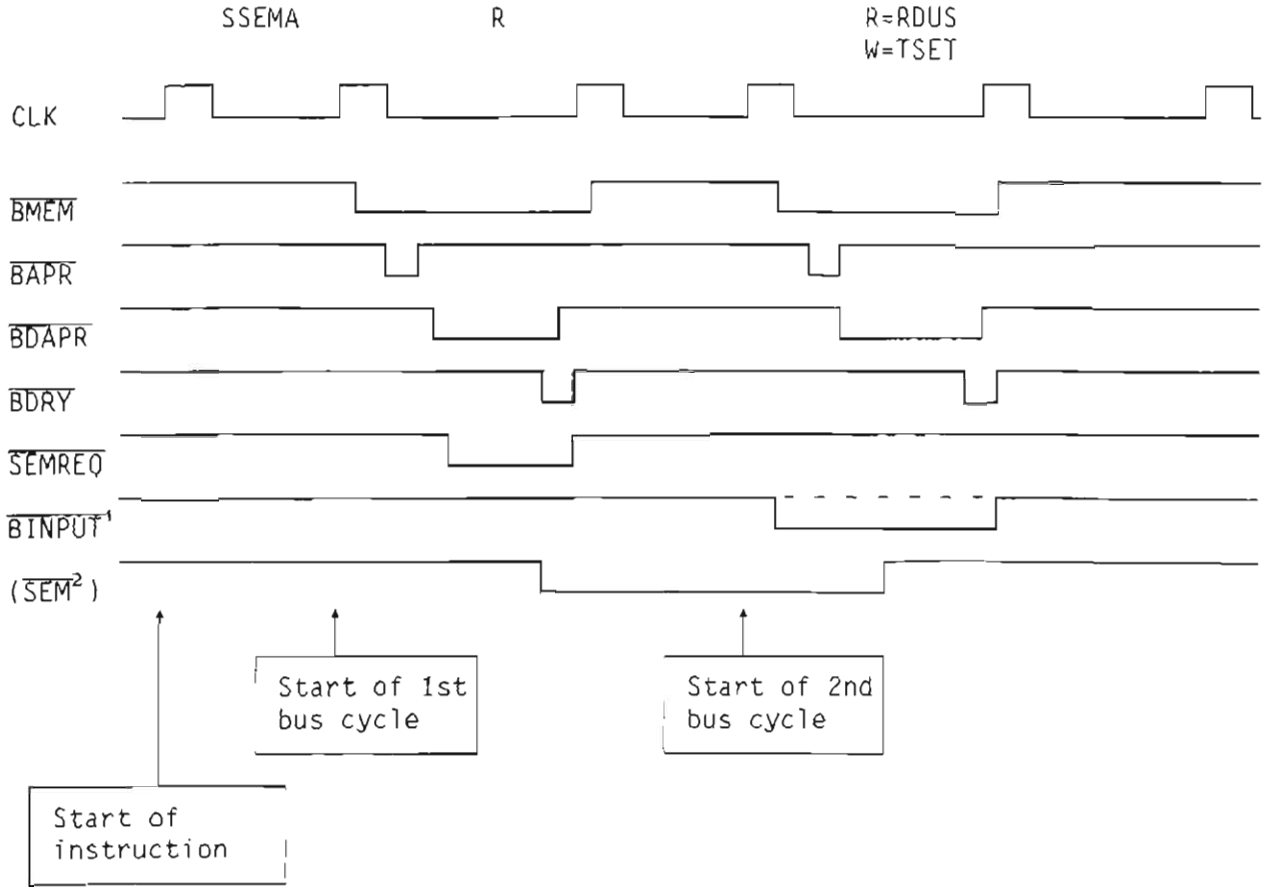
ND-110 uses the same multiplexed bus as the ND-100 family. A multiplexed bus uses the same signal lines for addresses and data. Address and data signals are present at different times during a bus cycle. That is, every cycle consists of two phases, one where an address is present and one where the data is present.

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

A CPU memory reference occupies the bus typically for 450 ns , and a DMA transfer for 550 ns . A special case is a bus cycle that uses the semaphore signal ($\overline{\text{SEMREQ}}$). This will lock the bus for two consecutive bus cycles.

CPU semaphore cycles

The instructions TSET and RDUS use the $\overline{\text{SEMREQ}}$ signal to lock the bus.



Note 1: $\overline{\text{BINPUT}}$ remains high (indicating a read) for RDUS.
2: Internal signal within the CPU indicates locked cycles.

Figure 36. CPU SEMREQ cycle timing diagram.

The TSET instruction might conceivably cause a protect violation on attempting the write cycle. Such a situation, if it were allowed to occur, would lock the bus. The micro-program traps such situations before starting a locked bus cycle.

An added feature in ND-110 is that the $\overline{\text{SEMREQ}}$ signal may be driven from a DMA device on the ND-100 bus (eg. multiport memory), and not only from the CPU itself.

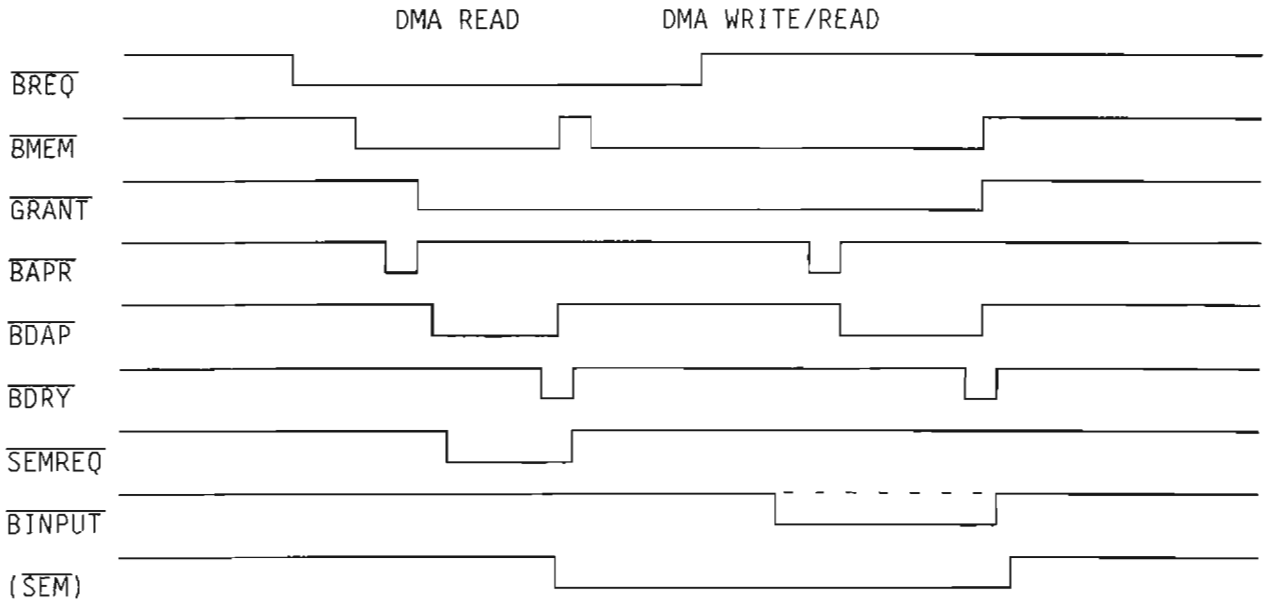


Figure 37. Bus SEMREQ cycle timing diagram.

The semaphore signal ($\overline{\text{SEMREQ}}$) will make sure that the bus arbiter will be locked for two consecutive bus cycles, thus ensuring that no other device modifies the memory location. Semaphore locked cycles always use a bus cycle; they never use cache.

CHAPTER 5 THE ND-110 STORAGE SYSTEM

CHAPTER 5 THE ND-110 STORAGE SYSTEM

Storage is one of the major building blocks in a computer system. It is used to hold programs (instructions), data, addresses and results.

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

- low access time
- low storage cost
- large capacity

These requirements are usually conflicting.

5.1 THE MEMORY HIERARCHY

In the ND-110, a hierarchical memory system is employed. It is convenient to consider the memory system as being divided into five levels.

- registers
- working register file
- fast cache memory
- primary memory
- secondary storage.

The fastest storage available to the programmer are the registers. The CPU can access these at full speed. They are severely limited in number.

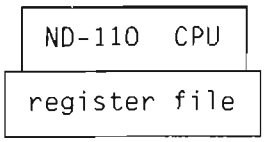
Next in availability is cache memory. The ND-110/CX uses cache for both data and instructions, but the slower ND-110 uses cache for instructions only. Cache memory is very fast but limited in size (4K for ND-110/CX, 1K for ND-110).

Primary memory can be expanded up to 16 Mbyte

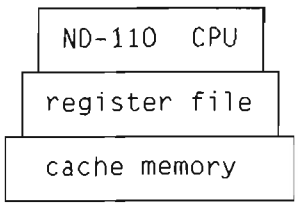
and is reasonably fast (450-550 ns). In normal operation, the most used data and instructions will be available from cache.

Secondary storage (disk, magtape) may be expanded to more than 2000 Mbyte.

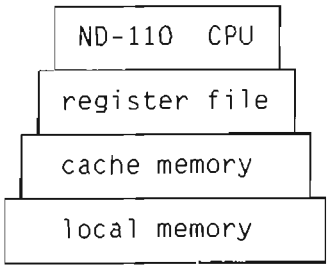
5.2 ND-110 MEMORY SYSTEM ORGANIZATION



The first level in the storage system is the register file, holding 128 programmable registers, 8 for each program level. The CPU has direct access to the register file via the the internal data bus (IDB).



Cache memory is also located on the ND-110 CPU board. It is a selective, high speed, CMOS memory of 4 Kwords, dynamically updated to hold the most recently used instruction (and data in the case of ND-110/CX). The cache memory reduces the average memory access time significantly.



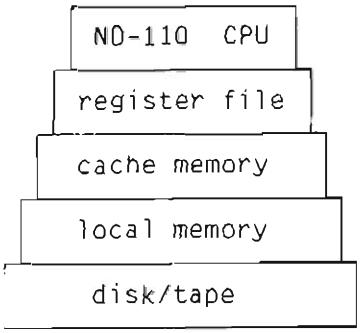
Local (main) memory is located in the same card crate as the CPU and may have any size from 32 Kwords (64K bytes) to 16 Mwords (32M bytes) in steps of 32 Kwords. Each card may contain a maximum of 1 Mword (2 Mbytes). Each word in local memory is stored with a 6-bit error-correction code which makes it possible to:

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

Multiport memory

On the same level as local memory, a multiport memory may be installed. Multiport memory is accessed through a multiport memory channel transceiver connected to one port in a separate card crate. The multiport memory may have several ports, allowing several sources to access the same physical memory area.

In principle there is no difference between local memory and a multiport (remote) memory, except that the access time is longer for the multiport memory. In addition, a multiport memory allows communication between several processors. Multiport memory is always used when the ND-110 CPU is part of a ND-500 computer.



The multiport memory channel is further described in Multiport Memory Channel Specifications (ND-10.006.).

The next level of the ND-110 storage system consists of mass storage devices such as disks and magnetic tapes. Some of the data to be processed is required seldom and does not have to be in memory at all times. This data can be stored on magnetic tape, disk packs or floppy disks in a data library.

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

All memory cards in the ND-100 system have asynchronous timing relative to the CPU. That is, several handshaking signals must be exchanged between the CPU and the memory system during the transfer. Memory cards with different access times can be mixed.

5.3 Cache

Cache is a very fast memory area where often used instructions and data are stored.

On the ND-110/CX, cache is divided into 4 banks, two for instructions and two for data. The ND-110 Standard uses one cache bank for all types of instructions and does not use cache for data.

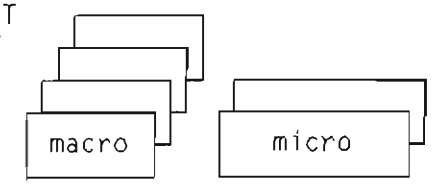
- Instruction, program level 1 (I1)
- Instruction, other levels (I2)
- Data, "standard" page table (PT)
- Data, alternate page table (APT)

The I1 cache bank is used for caching instructions which are fetched in program level 1 (user program level) while the I2 cache bank is used for instructions fetched in all other levels.

Data is likewise cached in the PT or APT bank according to which page table is used for addressing; standard or alternate.

5.4 CACHE ARCHITECTURE

The ND-110 CPU implements cache as an associative memory. Each cache bank is addressed using the lower 10 bits of the address bus from the MMS. This selects a word within any page in physical memory. Thus all pages of a particular type (Instruction: I1, I2; Data: PT, APT) share the same cache bank. The two instruction cache banks contain both a macroinstruction and microinstruction part.



The four cache banks

The advantage of using four cache banks instead of one, as used by the ND 100 CPU is that it avoids data and instructions competing for a cache entry, thereby increasing cache hit rate for certain types of tight loops. In addition program level 1 (normally user programs) are similarly separated from the other levels (subsystems, Sintran etc.); and operand accesses using normal page table (PT) are seperated from accesses using alternate page table (APT).

Any page in physical memory may be protected from updating cache by use of the cache-inhibit bit map. A page which is inhibited from update will still give cache hit for existing valid cache entries.

CACHE DATA ENTRIES

Cache entries can be logically divided into four parts:

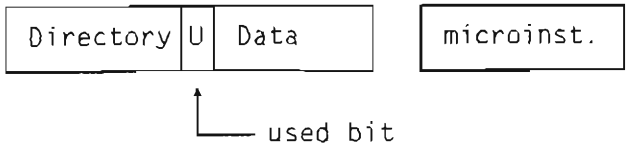


Figure 38. A cache entry

- Directory
- Cache-used bit
- Data word or instruction
- Microinstruction word (only for I1 and I2)

Directory

The directory entry is called the cache page number (CPN) and is the physical page number (14 bits) of the cache word.
A directory entry therefore contains the 14 most significant bits of the (24 bit) physical address of the memory location being cached.

Used bit

The cache-used bit is set (=> 1) if the cache location is currently in use and reset (=> 0) otherwise. Only cache entries which have their used bits set contain valid data.

Two separate cache-used bit-maps exist, only one of which selected at any time while the other is being cleared. Executing a cache clear instruction (TRR CCL) will exchange areas. Normally this will cause no extra overhead whereas ND-100 CPU disabled the cache for 60 µs while cache was being cleared.

The cache-used bit map is cleared automatically during those microcycles that do not reference memory. One bit is cleared for each such microcycle; the complete bit map needs 1024 microcycles that do not reference memory.

If a cache clear instruction is issued before the de-selected bank has been cleared, the CPU will halt until the bank has been cleared. This will take 104 ns times the number of bits left to clear (worst case ≈ 100 µs).

Data word or instruction

The data part of a cache entry contains the actual data or instruction being cached. When instructions are cached (cache banks I1 and I2 only), the first microprogram word of the instruction is also cached. This 64-bit microinstruction is stored in the microinstruction cache. This is actually physically part of the control store, but operates in parallel with the instruction cache and is not addressed separately.

CACHE MEMORY ORGANIZATION

Cache memory is divided into four banks of 1 page each. Two banks are for data and two banks for instructions. The instruction banks have corresponding microcache banks which are physically part of the writeable control store.

Cache used is implemented as a bit map. Each bit in the bit map represents a page in memory. There are two bit-maps in ND-110, only one of which is selected at any time. The CPU clears the other bit map automatically so that a clear cache instruction can normally swap bit maps without delay.

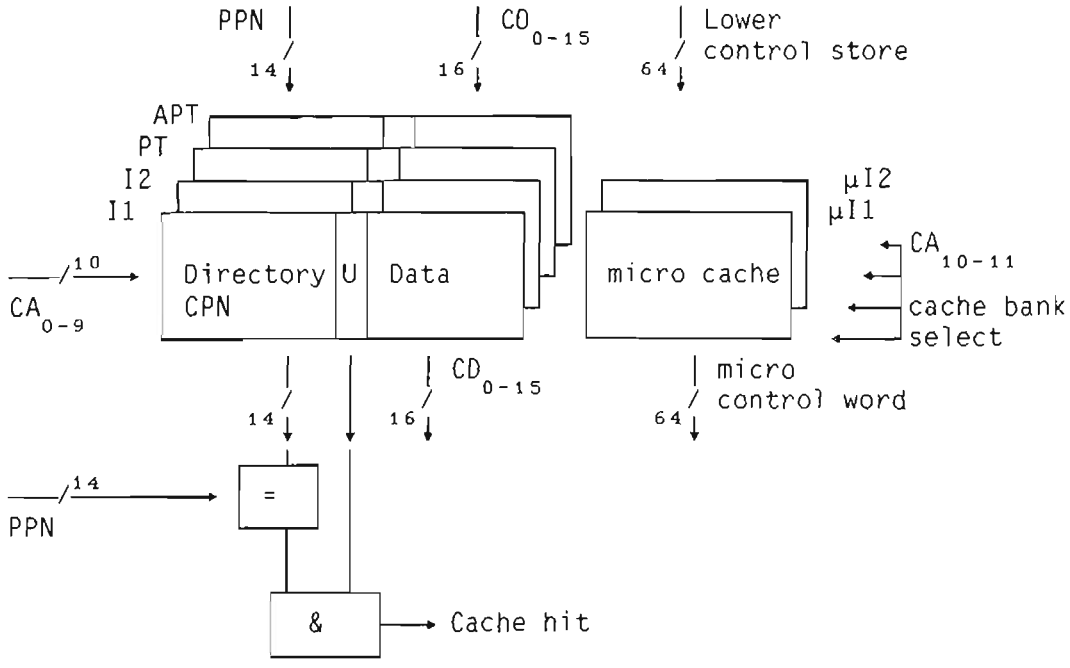
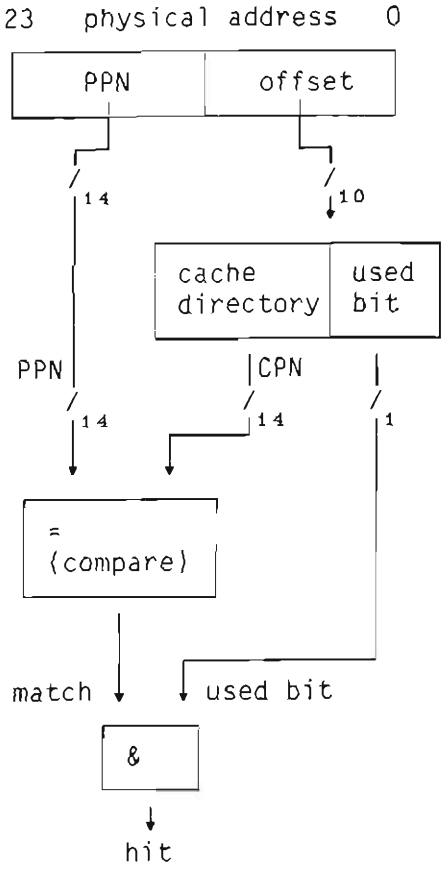


Figure 39. Cache organization

5.5 CACHE MEMORY ACCESS



Cache memory is accessed each time an instruction is to be fetched or a data operand is read or written. The cache bank is selected according to what type of memory access is involved. An instruction-fetch selects I1 (PIL = 1) or I2 (PIL ≠ 1). A data-read or write selects PT (normal page table) or APT (alternate page table).

The lower 10 bits of the physical memory address (offset) from the memory management system (MMS) are used to address within the selected bank. The contents of the cache directory at that address is the page number of the selected cache entry (cache page number, CPN). These two values are compared (block marked [=] in figure 39). If they match, the cache entry corresponds to the physical address. If, in addition, the used bit of that entry is set (= 1), then the cache entry contains valid data (cache hit).

The sequence of events during cache access varies according to whether an instruction is fetched, a data word read or a data word written. In the case of instruction-fetch and data-read, the action also depends on whether the instruction or data can be found in cache (cache hit) or if it must be read from memory (cache update). Data is always written to both cache and memory (write through cache algorithm).

Instructions are never written. Programs are normally loaded by DMA and cache cleared before execution.

Thus 4 different types of cache **read** access can be defined:

- Instruction-fetch Cache hit
- Instruction-fetch Cache miss
- Data-read Cache hit
- Data-read Cache miss

and one type of **write** access:

- Data-write through Cache

CACHE READ ACCESS

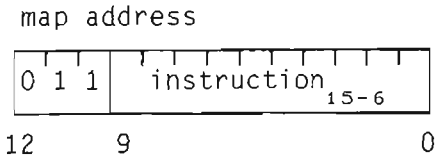
Instruction-fetch Cache hit

When an instruction is being fetched (last μ-cycle of previous instruction), the displacement (address within a page) is used to address cache memory. The current program level selects the bank (PIL = 1 selects bank I1, all others select I2).

The cache page number CPN is read out of the Directory and compared with the physical page number (PPN) from the memory management system (MMS). If these two are identical and the used bit is set (= 1), then a cache hit has occurred and the instruction cycle begins using the 64-bit word in the microcache, and the 16-bit instruction, which is copied into the GPR register in BUFALU.

Instruction-fetch Cache miss

If the cache page number and the page number from MMS are different or a match was found but the cache-used bit is not set, then an ND-100 bus transaction takes place. The 16-bit instruction is read from the bus, RMIC extracts bits 6 to 15 of the instruction and combines it with 06000_8 to create an address in the control store map area.



If the bit corresponding to the PPN in the cache page inhibit bit map is zero (not inhibited) the instruction is written to the cache data word and the cache-used bit is set to 1 to indicate that the cache entry is valid.

The 64-bit word read from the control store map area is the first micro code word of the instruction and is written to the microcache.

Data-read Cache hit

A data-read cache hit is similar to an instruction-fetch cache hit. The PT or APT bank is selected according to which page table is being used for the data-read. If the CPN in the Directory for the selected cache bank matches the page number from the MMS and the corresponding bit in the cache-used bit map is set, then the 16-bit data word from the cache is copied to the DBR register in BUFALU and the current instruction proceeds without a ND-100 bus transaction.

Data-read Cache miss

If the Directory CPN and the page address from MMS are different or a match was found but the cache-used bit is not set, then the word is read from the ND-100 bus and copied to the DBR register in BUFALU. If the bit corresponding to the PPN in the cache page-inhibit bit map has been cleared, the 16-bit word is written to the cache data word and the cache-used bit is set.

CACHE WRITE ACCESS

Data-write through Cache

A Data-write always updates both cache and memory. In addition the other data cache bank (PT or APT) and both instruction cache banks (I1 and I2) are searched for hit (CPN identical with MMS page address and cache-used bit set) and reset if a hit was found. This invalidates any old copies of the memory location in the other cache banks.

5.6 CACHE CONTROL AND STATUS

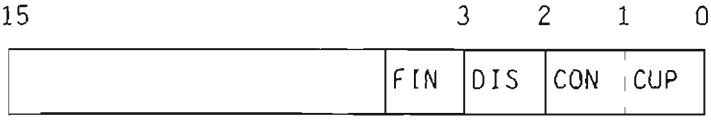
Cache functions are controlled with the help of a status register and a bit map. The bit map in ND-110 replaces and extends the upper inhibit and lower inhibit registers of the ND-100 CPU.

- Cache status register
- Cache-inhibit bit map

Cache status register

The cache status register (CSR) is loaded into the A register by the TRA CSR instruction.

The CSR has the following format:



Bit 0: CUP - The cache-updated bit is set (1) if cache was updated on the current memory request. This bit is valid from one memory request to the next.

Bit 1: CON - Cache on. On the ND-110 this bit is inverse of bit 3. It is set when cache is enabled using the cache ON/OFF switch on the ND-110 CPU card. Unlike the ND-100, it is not cleared during cache clear.

Bit 2: DIS - The manual-disable bit is set (1) when cache is disabled using the Cache ON/OFF switch on the ND-110 CPU card.

Bit 3: FIN - Cache clear finished. Indicates that the cache-used bit map, not currently enabled, has been cleared. If a cache clear instruction is issued while this bit is zero, the CPU will halt until the cache-used bit map is cleared.

Cache-inhibit bit map

Areas of memory may be defined as being inhibited from cache update. This is intended for memory areas that are involved in DMA transfers and for memory shared by other processors.

Caution
Inhibiting a cache page does not remove valid entries for that page. Use cache clear after inhibiting a page if the contents of that page are no longer valid.

Although ND-110 does not have the upper and lower limit registers of the ND-100 CPU, the instructions that set the upper and lower limit registers set bits in the bit map to perform the same function.

The upper and lower limits are set by the instructions:

```
LDA <lower limit>
TRR LCIL
```

and

```
LDA <upper limit>
TRR UCIL
```

In addition, ND-110 has a new instruction which allows single pages to be inhibited or freed.

```
LDA <limit word>
TRR CILP
```

The format of the limit word is:

= 0 : the page is set to inhibit
= 1 : the page is set to normal

5.7 Local Memory

5.7.1 Memory specifications

ND-110 memory cards are available with up to 8 Mbyte of error corrected memory. The table below gives the ND numbers for the cards along with their size and print number. Memory technology is changing rapidly. The table may not include the latest memory cards. Contact your ND sales office for current information.

ND sales number	Memory size	Print no.
ND-113	64 Kbyte	3036
ND-115	128 Kbyte	3036
ND-116	256 Kbyte	3034
ND-117	512 Kbyte	3034
ND-370	1 Mbyte	3042
ND-380	2 Mbyte	3042
ND-???	?4 Mbyte?	30??
ND-???	8 Mbyte	30??

Table 8. Available memory cards

Switch settings

All memory cards for the ND-100 and ND-110 CPUs may be configured by means of limit switches.

Lower Limit Switch

The lower limit (LL) switches set the lower memory address for the memory card. The value set corresponds to the two (or three) octal digits of the address.

Upper limit indicator

The upper limit (UL) address is displayed on the two (or three) digit indicator. The memory card calculates the UL value from the LL value and its own address range.

Example:

An ND-110 machine has two 512 K memory cards. The switch setting of the rightmost card (lowest addresses) will always be 0. The Upper limit indicator will show 040. This value is then set on the lower limit indicator of the second card. The upper limit indicator on the second card will now show 100.

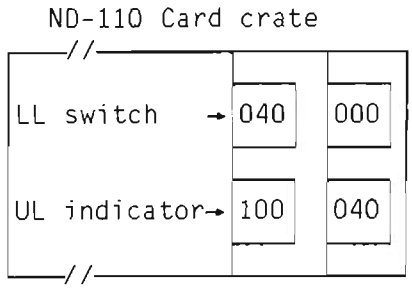


Figure 40. Memory switch settings

It is also possible to let the card determine both lower and upper memory address as a function of the slot position in the card crate. The lower limit switches should then both be set to 8. This is only meaningful for card with 64 K address space (64 Kwords).

Relative slot position [1]	Upper limit display	Address range
8	04	0K ≤ A ≤ 64K-1
7	10	64K ≤ A ≤ 128K-1
6	14	128K ≤ A ≤ 192K-1
5	20	192K ≤ A ≤ 256K-1
4	24	256K ≤ A ≤ 320K-1
3	30	320K ≤ A ≤ 384K-1
2	34	384K ≤ A ≤ 448K-1
1	40	448K ≤ A ≤ 512K-1

Note 1: The positions are increasing from left to right, where 8 is the rightmost slot position in the crate.

Table 9. Address Space for 64 K memory card

MEMORY ACCESS INDICATORS

The address space of the card is divided into four, and each part has a memory access indicator. When a memory access is performed on the card, the corresponding indicator lights.

ECC DISABLE SWITCH

The Error check and correction (ECC) system on the card may be disabled with the ECC disable switch. The red indicator, called **DISABLE**, lights when the ECC system is disabled.

5.7.2 ADDRESSING

The 16-bit virtual address is generated by RMAC. The memory management system converts this to a 24-bit physical address which is sent out onto the bus together with control signals. These control signals are described in the following section.

All memory cards connected to the ND-100 bus "listen" to the address on the bus. Each memory card has an address range (LL to UL-1). (see page 123 for switch setting details). Only the card containing the given address will answer.

MEMORY ACCESS

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

Local memory can be used by three different controllers:

- Refresh controller (on the CPU card)
- Direct Memory Access (DMA) controller(s)
- Central Processor (CPU)

These controllers operate independently of each other. A bus arbiter (on the CPU card) ensures that only one controller uses the bus at any time. If more than one controller

requests the bus at the same time, the arbiter allocates the bus in the order:

1. Refresh
2. CPU (if last bus cycle was a refresh)
3. DMA [1]
4. CPU (previous cycle not refresh)

Note 1: If there is more than one DMA controller, the distance from the CPU decides which controller has greatest priority (see page 157).

The CPU is however given priority over the DMA once each refresh cycle (every 13 µs). This ensures that the CPU can access the bus for fast interrupt handling. The ND-110 CPU will normally make the majority of its memory accesses from cache which do not involve the ND-100 bus. This means that the CPU can generally work undisturbed in the presence of heavy DMA activity on the ND-100 bus.

MEMORY ACCESS TIMING

A CPU memory read or write cycle is started by an internal signal requesting the bus. When the bus arbiter grants the request ($\overline{\text{BMEM}}$), the bus cycle can begin. The CPU sets $\overline{\text{BINPUT}}$ false (high) for a memory read cycle and true (low) for a memory write cycle. The 24-bit physical memory address is strobed onto the bus ($\overline{\text{BAPR}}$).

When valid data is available on the bus, the data source (memory card for read cycle; CPU for write) acknowledges with $\overline{\text{BDAP}}$. The memory card closes by the memory cycle by signaling with $\overline{\text{BDRY}}$ that data has been transfered.

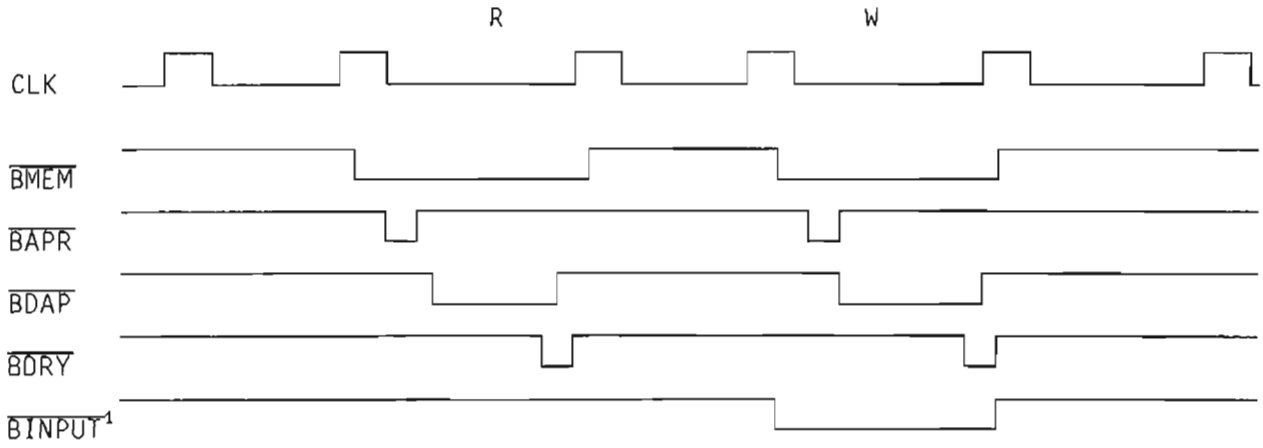


Figure 41. CPU read/write cycle timing diagram.

A DMA memory cycle is started by the DMA controller requesting the bus (BREQ). The bus arbiter sends a GRANT signal to the INGRANT/OUTGRANT daisy chain. The requesting DMA controller answers with BAPR. See page 157 for a detailed description of DMA transfers.

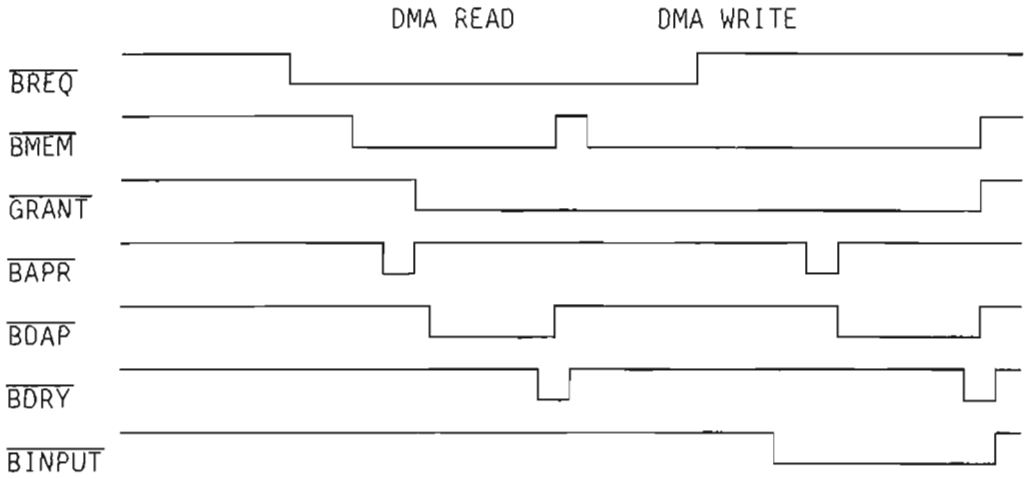


Figure 42. DMA read/write cycle timing diagram.

Locked bus cycles

The ND-110 CPU supports locked bus cycles using the SEMREQ signal. Locked bus cycles are two normal bus cycles that are indivisible. This means no other device can access the bus until the locked cycle has been completed. Locked bus cycles are used during the ROUS and TSET instructions. DMA devices (for example multiport memory) can also use locked bus cycles on ND-110 systems. See page 107 for more details about locked bus cycles.

5.7.3 Error check and correction (ECC)

The red indicator, called **ERROR**, lights when the ECC system detects a parity error. Use of the ECC disable switch or a Master Clear resets the indicator.

input A	input B	output A xor B
0	0	0
0	1	1
1	0	1
1	1	0

Table 10. Truth table for XOR

Each (16-bit) word in memory is stored together with a 6-bit ECC code (EC 0-5). This code is generated by hardware on the memory card. Each EC bit is the result of a logical exclusive-OR (XOR) operation on different subsets of the data bits.

When a memory read access is performed, the stored ECC code is compared with a newly generated ECC code. If the two ECC code are different, an error has occured. If only one bit (data or ECC code) is in error, that bit is corrected. If two or more bits are in error, the error is not correctable.

Error Correction Control Register (ECCR)

One register controls all the memory cards on the ND-100 bus.

The register is loaded with a TRR ECCR instruction. The microprogram performs an IOX type instruction to the appropriate address.

The format of the ECCR is as follows:

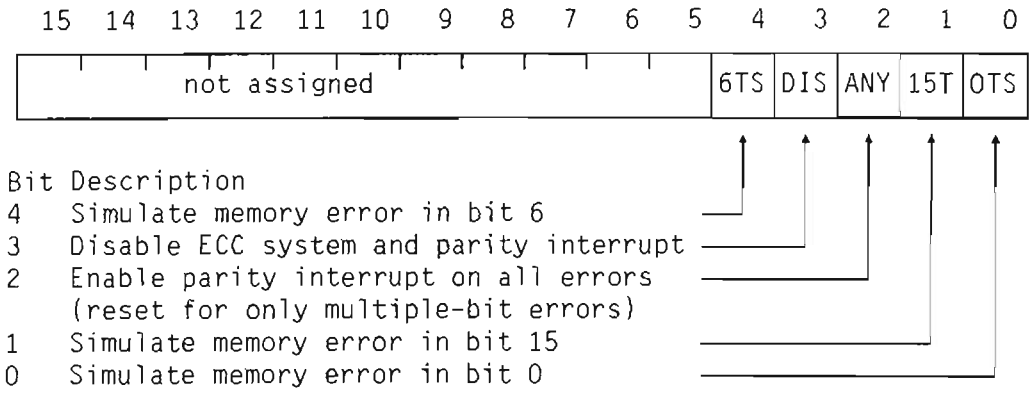


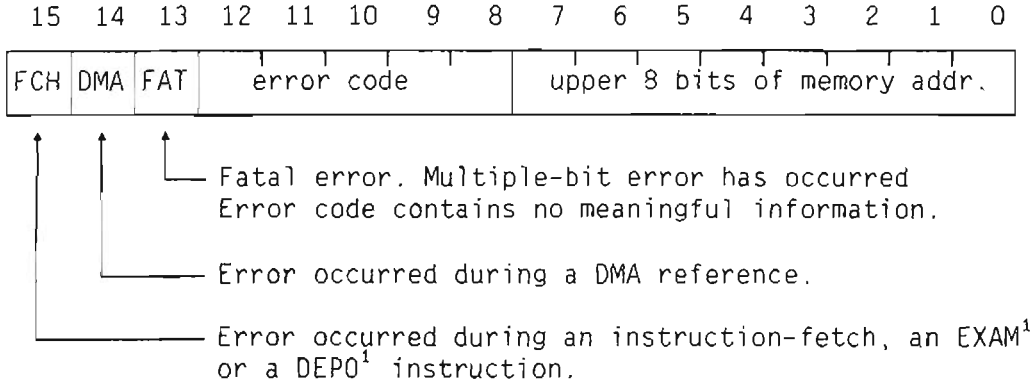
Figure 43. Error correction control register format

PARITY ERROR STATUS REGISTERS PEA, PES

These registers contain status information after an interrupt caused by a parity error or a memory out of range interrupt. The interrupt locks their contents. They are unlocked by reading the PEA register. This means that the PES register must always be read first.

PARITY ERROR STATUS (PES) REGISTER

The PES register is located inside the ND-110 CPU. It has the following format:



Note 1: EXAM and DEPO are OPCOM commands

Figure 44. PES register format

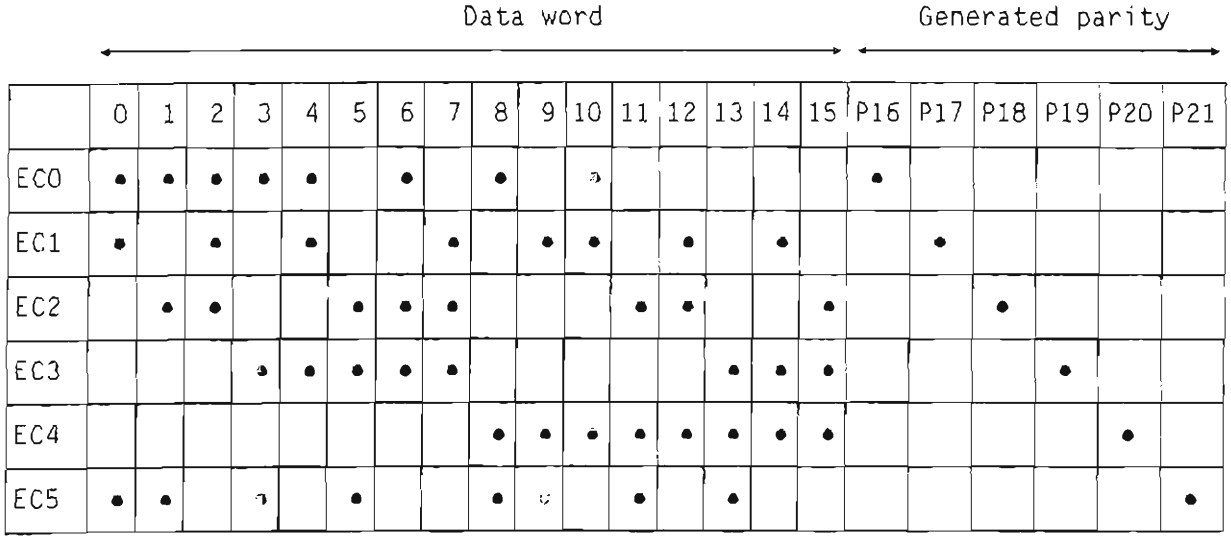
Bits 0-7 contain the 8 most significant bits of the last memory address issued on the ND-100 Bus. The PEA register contains the 16 least significant bits.

Error code EC 0-4	Failed bit (D=Data bit, P=Parity bit)	Error code EC 0-4	Failed bit (D=Data bit, P=Parity bit)
0	OK (no error)	20_8	P20
1_8	P16	21_8	D8
2_8	P17	22_8	D9
3_8	00	23_8	D10
4_8	P18	24_8	D11
5_8	D1	25_8	(not single)
6_8	(not single)	26_8	D12
7_8	D2	27_8	(not single)
10_8	P19	30_8	D13
11_8	D3	31_8	(not single)
12_8	(not single)	32_8	D14
13_8	D4	33_8	(not single)
14_8	D5	34_8	D15
15_8	D6	35_8	(not single)
16_8	D7	36_8	(not single)
17_8	(not single)	37_8	P21

Note 1: FATAL = 0 for all entries shown
Note 2: Errors shown as not single are multiple-bit errors. It is not possible to identify the failed bit for these errors.

Table 11. Coding of single-bit memory errors

Bits 8-12 contain the error code (EC bits 0-4) sent from the the interrupting memory card. The table below may be used to translate the error code.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	P16	P17	P18	P19	P20	P21
EC0	•	•	•	•	•		•		•		•						•					
EC1	•		•		•			•		•	•		•		•			•				
EC2		•	•			•	•	•				•	•			•			•			
EC3				•	•	•	•	•						•	•	•				•		
EC4									•	•	•	•	•	•	•	•					•	
EC5	•	•		•		•			•	•		•		•								•

The table shows which group of bits that are XOR'ed together during write and read. • means the data or parity bit is present in the EC group.

Table 12. Terms included in ECC Coding

During write the parity bits P16 - P21 are generated:

$$P_{n+16} = Dx_1 \text{ XOR } Dx_2 \ldots \text{XOR } Dx_9.$$

During read the ECC bits EC 0-5 are generated:

$$EC_n = (Dx_1 \text{ XOR } Dx_2 \ldots \text{XOR } Dx_9) \text{ XOR } P_{n+16}.$$

The error code in the PES register contains the ECC bits EC 0-4. EC 5 is zero not used for single bit errors.

Example: Suppose bit data bit D15 fails. Because the D15 is included in the equations for EC2, EC3 and EC4, these bits are set. The value of EC bits 0-4 becomes $011100_2 = 34_8$.

Parity Error Address (PEA) register

The PEA register is located inside the ND-110 CPU.

It contains the 16 least significant bits of the physical memory address present on the ND-100 bus at the time of the memory access which caused the interrupt.

The register may be read using the instruction TRA PEA.

CHAPTER 6 THE INPUT/OUTPUT SYSTEM

CHAPTER 6 THE INPUT/OUTPUT SYSTEM

The purpose of the input/output system (I/O system) is to ensure the physical communication between connected peripheral equipment and the ND-110 computer system.

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

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

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

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

Depending on the speed, a device could be connected to an ND-110:

Slow: with CPU controlled, Program Input/Output (PIO).

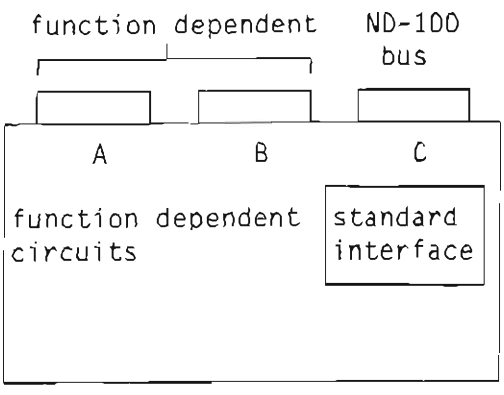
Fast: with Direct Memory Access (DMA)

6.1 ND-100 BUS IN THE I/O SYSTEM

The ND-110 computer system uses the ND-100 bus as a communication path between its functional blocks. This bus is unchanged from the ND-100 computer family and is described in detail in the ND-100 Bus Description manual ND-06.017.

New for ND-110, however, is that devices other than the CPU (e.g. multiport-memory) may now use the semaphore signal $\overline{\text{SEMREQ}}$ to reserve the bus for two consecutive bus cycles.

6.1.1 ORGANIZATION OF AN I/O DEVICE CONTROLLER CARD



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

The ND-100 dependent part includes bus handshake control logic This part is standardised for:

- all PIO device controllers
- all medium speed DMA controllers (e.g. mag. tape)

Figure 45. ND-100 standard I/O card

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

Examples of I/O cards

- four terminals and floppy disk controller
- eight terminal card
- one mag. tape controller (up to 4 units)

6.1.2 Allocation of the ND-100 Bus

One of the functions of the bus control, is to allocate the ND-100 bus to one of the possible requesting bus users (refer to chapter 4, page 105). That is, to:

- the CPU
- a DMA controller
- memory refresh cycle

These sources request the ND-100 bus asynchronously, and therefore a priority arbiter network is implemented in the bus control.

In order for the bus control to know who has initiated a request, each bus user is assigned a unique bus request signal.

CPU Bus Request

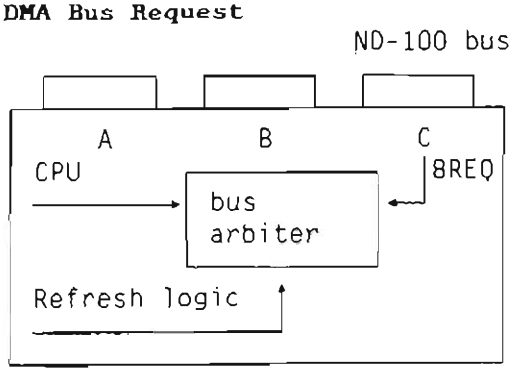
The CPU may allocate the ND-100 bus for one of six reasons:

- instruction fetch[1]
- operand read[1]
- indirect address read[1]
- operand store
- programmed access to the I/O system
- programmed access to external system control registers[2]

Note :

1. These causes a bus request only when the word is not in cache. See page 115 for more details of cache memory.
2. Control registers not located on the CPU card (for example, Error Correction Control Registers (TRR ECCR) on memory modules).

DMA Bus Request



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

Figure 46. Bus request sources

Memory Refresh Bus Request

The memory refresh cycle is started from the CPU card.

The ND-100 bus is allocated and released on a cycle basis, i.e. for every byte/word to be exchanged. The cycle may be subdivided into an address cycle and a data cycle (See chapter 4 for more details of the multiplexed bus).

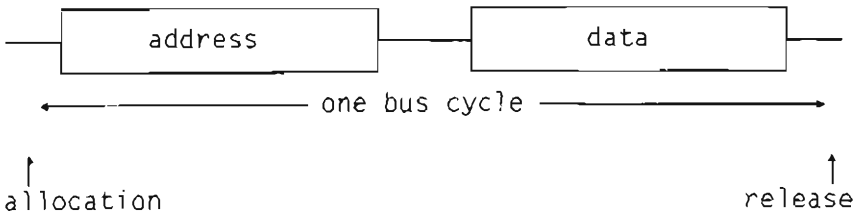


Figure 47. ND-100 Bus Cycle

The accessed device, i.e. the I/O system or memory system, releases the bus when ready (bus data ready - $\overline{\text{BDRY}}$) The time from allocation to release of the ND-100 bus shall not exceed 15 µs. This is monitored by the bus control.

If the bus is not released, it will cause a system hang-up. To prevent such a situation, the bus control will abort a bus cycle which exceeds 15µs. The faulty cycle is reported to the CPU as internal interrupt on level 14 (MOR, IOXERR).

6.2 Programmed Input/Output

External devices may be classified as:

1. Slow character/word oriented devices (e.g. terminals)

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

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

A PIO interface is always designed to handle slow byte/word oriented devices (tape reader, line printer, etc.), and is completely controlled by the CPU. In programmed data transfers, each word or byte is exchanged under program control.

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

The I/O device interface is controlled by means of registers on the interface card. The two instructions, IOX and IOXT are used to access these registers.

6.2.1 The Input/Output Instructions IOX and IOXT

In the ND-100 instruction set there are two instructions used for information exchange between the hardware device controllers and the CPU: the IOX and the IOXT instructions. These are privileged instructions.

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

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

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

I/O device controllers are assigned a group of registers addresses. Each I/O register has its special meaning for the particular interface. I/O device registers may be used for:

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

IOX and the IOXT instructions access an I/O device register by its address, referred to as 'device register address'. Data is always transferred via the A register.

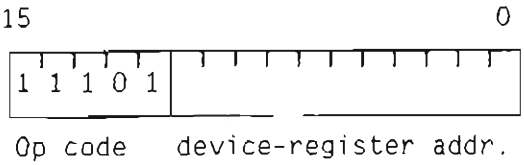


Figure 48. IOX Instruction Format

In the IOX instruction, the address of the I/O device register is specified in bits 0 - 10 of the instruction itself.

Usage:

IOX ‹device-register address›

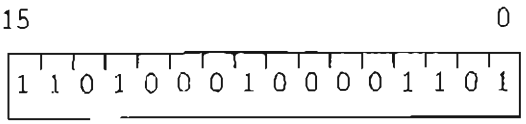


Figure 49. IOXT Instruction Format

In the IOXT instruction, the 16-bit device-register address is loaded into the T register prior to executing IOXT.

Usage:

LDT ‹device-register address›

IOXT

IOX TRANSFER DIRECTION

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

Direction encoding

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

The transfer direction is input if the device-register address is even.

That is:
bit 0 of the address decides the transfer direction.

Bit0 = 0 Input (from device to CPU)

Bit0 = 1 Output (from CPU to device)

This means that all I/O device registers which need to be loaded from the CPU A register (output transfer) are assigned an odd device register address. Similarly I/O device input registers are assigned even addresses.

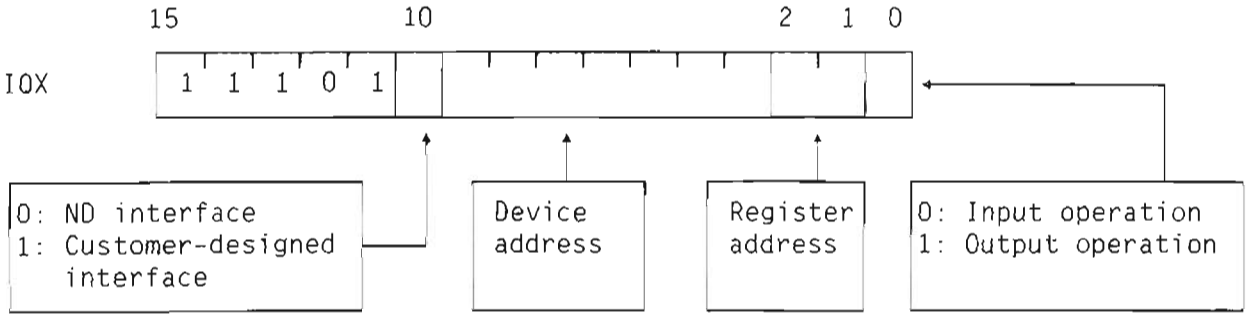


Figure 50. IOX Instruction decoding details

When the IOXT instruction is used, the T register contains the 16 bits device-register address. Bits 0, 1 - 2 and 10 of the address have the same interpretations as for the IOX instruction.

Calculation of the Device Register Address

The IOX Instruction Address Range

The IOX instruction can address a total of 2048 registers, i.e. addresses from 0-3777_8. However, the device-registers implemented on interfaces designed at Norsk Data use the address area 0 - 1777_8 only.

The remaining 1024 register addresses are available for customer-designed interfaces.

The IOXT instruction uses the 16-bit T register to hold the device-register address, and, in theory, can address 64 K register addresses (0-177777$_8$). Only some of these addresses are legal however.

The range covered by the IOX and IOXT instructions is illustrated below.

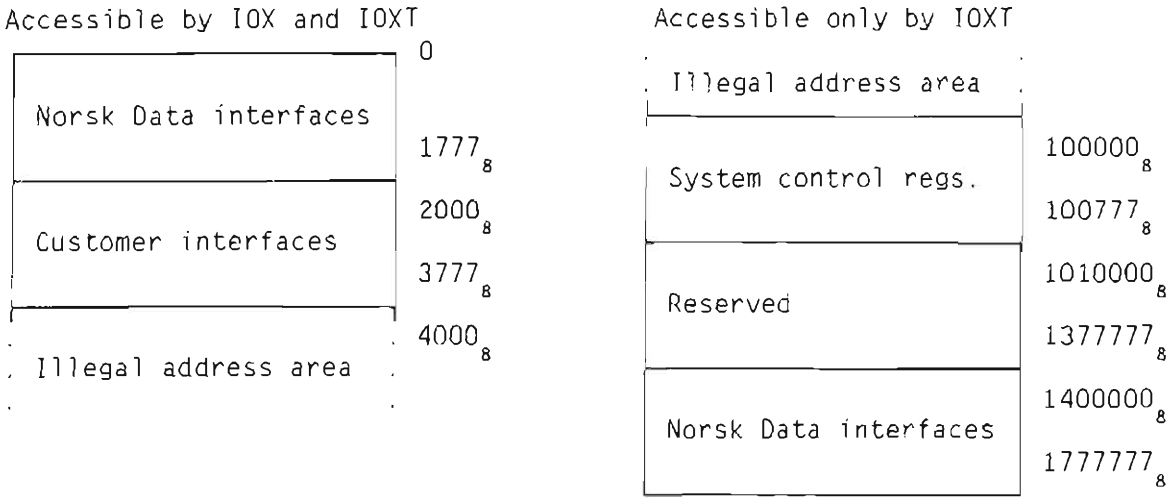


Figure 51. IOX and IOXT Address Range

The device-register address of the IOXT instruction overlaps in the IOX instruction's address range.

Illegal address area

The address range from 4000$_8$ to 77777$_8$ is illegal and an attempt to perform an IOXT instruction with an address in this range will cause an IOX interrupt (see section on internal interrupts, page 48).

Addresses from 100000$_8$ - 100777$_8$ are used to specify system control registers which have to be accessed via the ND-100 bus. An example is the Error Correction Control Register (ECCR), physically located on the memory modules.

ECCR is loaded by the TRR instruction. However, since ECCR is accessed via the ND-100 bus, the microprogram performs the equivalent of an IOXT instruction to address 100115$_8$.

Reserved for future needs

Addresses from 101000$_8$ - 137777$_8$ are reserved by Norsk Data for future needs.

Reserved for future I/O

Addresses from 140000_8 - 177777_8 are reserved by Norsk Data for future extension of the I/O device-register address range.

Since all present I/O device controllers designed at Norsk Data may be specified in the address area 0 - 1777_8, and may be specified by both the IOX and the IOXT instruction, the IOXT instruction is used in the programming examples and when referred to.

6.2.2 Specification of an I/O Device Register Address

Each I/O device controller is assigned a group of consecutive device-register addresses. The total number of registers assigned one I/O interface may be from 4 to 16, depending on the control functions needed on a device. The device-register address may therefore be divided into two parts:

- device number (base address of device)
- register number. Register address within the selected device

The IOX/IOXT device-register address is then formed by combining the two parts:

‹device-register address› = ‹device number› + ‹register number›

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

The numbers assigned to the various registers on an I/O interface are given in the specifications following each I/O interface. See appendix B (Programming Specifications for some I/O Devices) for more details.

Example:

The programming specification for terminal number 1 can be found on page 160. Eight register addresses are assigned to terminal 1 (they are described in detail on the above mentioned page). The lowest device-register address is to 300_8. This is the device address for terminal number 1.

Each peripheral type has a corresponding I/O interface and device number. As there may be more than one interface card for a particular

I/O type (e.g. terminal interface), there will be several device addresses available for each type of interface.

An edge switch on the I/O interface card is used to set the address of the particular card within the permitted device address area.

6.2.3 The Device Registers on I/O Interfaces

Each register implemented on an I/O interface is assigned a unique number in the interface. This is referred to as the register number.

The interpretation of the data word written to or read from the register is defined in the the programming specification for the interface.

The programming specifications for terminal number 1 and the real time clock may be found at the end of this chapter, starting on page 160. For other interfaces, the programming specifications will be found in the hardware description manual for the interface.

Example:

A PIO interface will have at least three registers for each channel:

- control register
- status register
- data register(s)

The control register is a 'write-only' register (IOX/IOXT output). Commands (start/stop transfer, mode of operation) from a device driver program to an I/O interface channel are given through this register.

The status register is a 'read-only' register (IOX/IOXT input). By reading the register, the status of an I/O interface channel (ready for transfer, busy, errors, etc.), may be investigated.

6.2.4 EXAMPLE OF A PROGRAMMED I/O ROUTINE

A programmed I/O (PIO) device may be driven either by an interrupt routine, or by a routine which polls (continuously senses) the status of the interface.

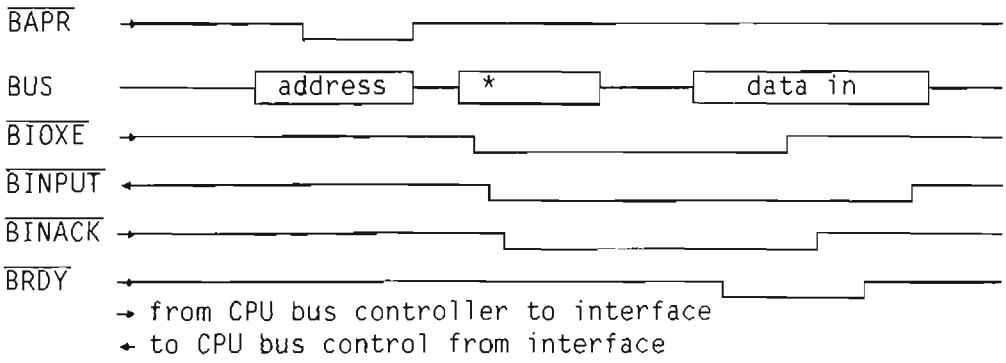
The following example shows how a polling routine reads a character from terminal 1. (See page 160 for programming specifications).

```
% The device number for terminal 1 is 300 .
                                          8
% First we must write a 1 to register 3, bit 2 to ensure that the
% device is active. This will also set the number of bits and
% parity The example uses 8-bit no parity.

START,      SAA 4           % A reg. bit 2 = 1
            IOX 303         % Load control word reg.

% Now we poll the read-status register 302  until bit 3 goes high
                                         8
% to indicate that a byte is available.

            IOX 302         % Read status reg.
            BSKP ONE 30 DA  % Is bit 3 = 1
                            % If yes, skip one location.
            JMP *-2         % If no, loop until true

% Data is available, so read one byte from data register 300  and
                                                           8
% save the value in X register.

            IOX 300         % Read data reg., char. in A reg.
            COPY SA DX      % Save character

% Now we shall echo the byte back to the terminal.
% First we must check if the device is ready to receive data

            IOX 306         % Read output status to check if ready
            BSKP ONE 30 DA  % If status bit 3 = 1
                            % skip one location
            JMP *-2         % otherwise loop until ready

% The device is ready, so copy byte into A register again and
% send it.

            COPY SX DA      % Unsave character
            IOX 305         % echo data

% Somewhere here in a real driver routine we would send the
% received data to another routine but as this is just an example
% we loop back to START.

            JMP START       % Repeat
```

6.3 ND-100 BUS SIGNALS DURING IOX INSTRUCTIONS

6.3.1 IOX INPUT

The figure below shows the control and data signals present on the ND-100 bus during the execution of an IOX input instruction.



Note (*) Data may be clocked out here for combined input/output cycles.

Figure 52. Control signals during an IOX input instruction

When an IOX instruction is executed, the 11-bit device-register address is sent out on the ND-100 bus, together with the control signal BAPR, bus address present. This signal tells all devices that a device address is present on the ND-100 bus.

Each interface compares the device-register address with its own address. The interface with the corresponding address is selected.

The controller sends the BIOXE, input/output enable signal, onto the bus signaling to the selected interface that data is may be placed onto the bus.

The selected device now sends a BINPUT signal to the bus controller, telling it that this is an input transaction. The bus controller, answers the BINPUT signal with BINACK, input acknowledge signal.

The 16-bit data word from the interface is now made available on the ND-100 bus. The interface informs the bus controller that the data is valid with the BDRY, data bus ready signal. When the CPU has read the data bus

into A register, it signals the interface by releasing the $\overline{\text{BINACK}}$ signal. The interface then releases the $\overline{\text{BRDY}}$ signal thereby ending the bus cycle.

6.3.2 IOX Output

The figure below shows the control and data signals on the ND-100 bus during the execution of an IOX output instruction.

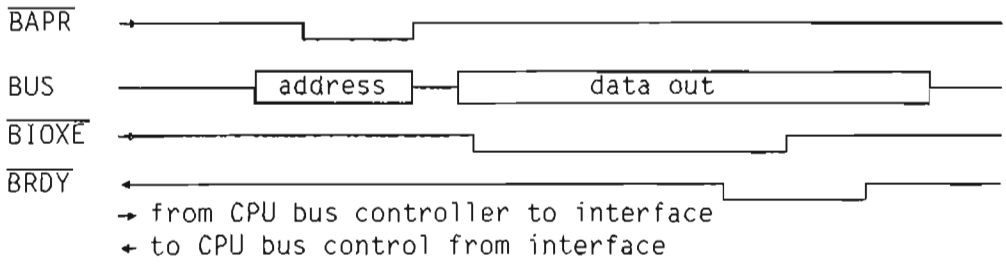


Figure 53. Control signals during an IOX output instruction

When an IOX instruction is executed, the 11-bit device-register address is sent out on the ND-100 bus, together with the control signal $\overline{\text{BAPR}}$, bus address present. This signal tells all devices that a device address is present on the ND-100 bus.

Each interface compares the device-register address with its own address. The interface with the corresponding address is selected.

The controller sends the $\overline{\text{BIOXE}}$, input/output enable signal, onto the bus signaling to the selected interface that data is may be placed onto the bus.

When the interface has read the data, the $\overline{\text{BDRY}}$, bus data ready control signal, is issued by the interface, terminating the bus cycle.

IOX Error

If $\overline{\text{BDRY}}$ is not received in the bus control within 15 µs after the start of an IOX input or output instruction, a timeout interrupt is generated. The cycle is terminated, and an internal interrupt, IOX ERROR, is sent to the interrupt system.

IOXT instructions

IOXT instructions follow the same sequence of control signals as IOX instructions. The only difference between an IOX and an IOXT instruction is that the IOXT instruction uses a full 16-bit device-register address. In the case of the IOX instruction the upper 5 bits of the address are always 0.

6.4 The I/O system and interrupt

When the operating system (SINTRAN III) is started, all I/O devices connected to the ND-100 bus will be initialised. Thereafter they operate asynchronously with respect to the CPU.

This means that the I/O controllers generate interrupts to signal to the CPU that a change of status has occurred.

Status changes that generate interrupts include:

- Error condition
- Output completion
- Input available

The status register of the interrupting device will contain flags (bits) which identify the exact cause of the interrupt.

6.4.1 INTERRUPT LEVELS

Interrupt levels 10-13 and 15 may be activated using signal lines available in the ND-100 bus. These lines go directly to the priority interrupt controller in the CPU. The CPU can read these signals from PID register.

For equipment produced by Norsk Data, the use of these lines has been standardised:

- All output interrupts use level 10
- All DMA controllers use level 11
- All input interrupts use level 12
- Real-time clocks and special devices such as HDLC input use level 13

Level 15 is not used by Norsk Data equipment, but is available for use by devices which need the fastest possible interrupt response.

6.4.2 DEVICE INTERRUPT IDENTIFICATION

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

The IDENT instruction performs a hardware search for the interrupting device. Only devices with active interrupts on the level specified in the IDENT instruction are included in the search. The device nearest the CPU on the daisy-chain and which has an active interrupt will respond with a 9-bit identification code.

The identification code is unique for each device and is used to generate a branch to the driver routine for that device. The driver will read the status register to find the reason for the interrupt and take appropriate action.

The interrupt sequence

1. An interrupt condition occurs in a device which latches the condition and drives the appropriate interrupt line (10, 11,

12 or 13).

2. If the CPU is operating on a level lower than the interrupt level, the CPU is forced to the interrupting level. If the CPU program level is higher than or equal to the interrupt level, the interrupt will remain pending until the CPU program level falls below the interrupt level.

3. The CPU issues an IDENT instruction. The identification code is read into the A register. At the same time the the IDENT instruction resets the interrupt condition on the interface.

4. Using the identification code to generate a branch address, the CPU will start executing the device driver routine.

5. The driver routine will normally begin by reading the status register of the device to find out the reason for the interrupt.

6. The driver routine will normally end with a WAIT instruction which gives up priority.

7. The CPU will restore the context of the interrupted program and resume execution.

On interfaces produced by Norsk Data, the edge switches which select the device address also select the appropriate identification code.

The Ident Instruction

The IDENT Instruction Format

IDENT ‹Program level code›

See appendix A for the values of the program level (PL) code.

Figure 54. The IDENT instruction

The IDENT instruction is a privileged machine instruction used in device interrupt identification. When executed, it searches for interfaces with interrupt condition set,

and reads the interface's identification code into the A register.

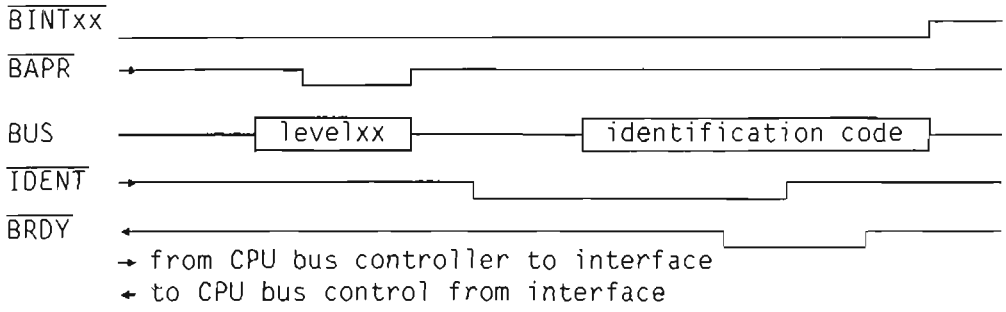
To maintain the interrupt priority the ident instruction searches only for interrupts on a specified level. The level to search is specified in the IDENT instruction.

Example:

The instruction IDENT PL12 searches for interfaces driving interrupt line 12 ($\overline{BINT12}$) only. An interrupt on level 10 or 11 will be ignored.

BUS SIGNALS DURING AN IDENT INSTRUCTION

The figure below shows the control and data signals on the ND-100 bus during an interrupt and IDENT instruction.



xx : Interrupt level (10, 11, 12 or 13)

Figure 55. Control signals during an IOX output instruction

The interrupt lines on levels 10 to 13 ($\overline{BINTxx}$) will set the appropriate bit in the priority interrupt detect register (PID register).

The CPU will change to the level of the interrupt and an IDENT instruction will be executed as part of the driver routine.

The six-bit program level code (see appendix A for values) is written onto the ND-100 bus and the $\overline{BAPR}$ (bus address present signal) is issued to tell all connected devices that a level code is present on the bus.

The CPU issues the IDENT signal and each device on the ND-100 bus accepts the $\overline{IDENT}$ signal from the device nearer the CPU and sends it to the next device in the daisy chain.

This search signal is daisy-chained from device to device until it is stopped by the interrupting device interface. The interrupt condition is cleared on this device, releasing the interrupt line.

The device sends the identification code to the CPU and signals that it is available with the $\overline{\text{BDRY}}$, bus data ready signal. This tells the CPU that data is ready on the ND-100 bus, and the identification code is read into the A register. The CPU acknowledges that it has read the data by releasing the $\overline{\text{IDENT}}$ signal, and the interrupt device terminates the bus transaction by releasing the $\overline{\text{BDRY}}$ signal.

It should now be obvious that there must never be any empty positions in the ND-100 bus between the CPU and I/O device controllers. An empty position would stop the search signal and never release interrupts on modules in higher slot positions than the empty one.

Among interfaces generating interrupt on the same level, the interface nearest the CPU has highest priority.

6.4.3 PROGRAM EXAMPLE OF INTERRUPT DRIVEN I/O

The example shows the use of interrupts in a driver routine for terminal number 1. It is assumed that there is a user program running on level 1 which will send and receive data via terminal 1. Data is placed in a variable (BUFF) ready for transmission and received data is made available in a corresponding variable RCVD.

In accordance with the standards used by Norsk Data devices, the input interrupt is on level 12 and the output interrupt on level 10. The device address of terminal number 1 may be found on page 160 and is 300_8.

Note the use of WAIT to relinquish priority, that is allow programs on lower levels to gain access to the CPU. Note also the JMP instruction after WAIT which brings execution back to the start of the interrupt routine again when the level is reentered.

```
% The following code must be executed to initialise the device
% setting bits 0, 1 and 2 (= 7 ) enables interrupts (for device
% ready and device error) and activates the device.
                                                    8
LEV1, SAA  7            % Set bit no. 0, 1 and 2 in A reg.
      IOX  CONTR        % Send to input control reg. (303 )
                                                        8

      SAA  7            % Set bit no. 0, 1 and 2 in A reg.
      IOX  CONTW        % Send to output control reg. (307 )
                                                         8
       .
       .                % initialising code for other devices would
       .                % follow here
      WAIT              % give up priority
       .
       .                % Code for the user program which will send
       .                % and receive data would come here
Level 10 (device output) routines are handled here
the data byte to be sent is available in the variable BUFF

LEV10, IDENT PL10 % Identify interrupt
       .
       .        % the code following the IDENT instruction
       .        % will route the terminal 1 interrupt to OUT1
       .
OUT1,  LDA  BUFF         % Get saved data
       IOX  WDATA        % send it to data output reg. (305 )
                                                           8
       IOX  STATUS       % Read output status reg. (306 )
                                                        8
       BSKP ZERO 40 DA % Check the error bit (no. 4)
       JMP  ERROR        % jump to an error handler
                         % (not given in example)

% if everything was OK, we continue here. We generate an internal
% interrupt to level 1 to say that output is done

       SAA  2            % set bit 1 (for level 1)
       MST  PID          % generate an internal interrupt

       WAIT              % Give up priority
       JMP  LEV10        % next time level 10 is entered execution
                         % continues here so we must jump to the
                         % beginning explicitly
```

...Continued on the next page.

Program example of interrupt driven I/O - continued.

```
% Level 12 (device output) interrupts are handled here
% the received data placed in a variable for use by some other
% program

LEV12, IDENT PL12       % Identify interrupt
       .
       .          % the code following the IDENT instruction
       .          % will route the terminal 1 interrupt to INP1
       .
INP1,  IOX  STATUS      % Read input status register (302 8)
       BSKP ZERO 40 DA % Check the error bit (no. 4)
       JMP  ERROR       % jump to an error handler
                      % (not given in example)

% we continue here if everything is OK

       IOX  RDATA       % Read data reg. (300 8)
       STA  RCVD        % Save data

% data has been received so we generate an interrupt to level 1
% where the user program that will use the data is running.

       SAA  2           % set bit 1 (for level 1)
       MST  PID         % generate an internal interrupt

       WAIT             % Give up priority
       JMP  LEV12
```

6.5 Direct Memory Access (DMA)

The most effective way for high-speed peripherals to transfer data to and from memory is the technique called direct memory access (DMA). This means that data is transferred without being read into a CPU register first. The CPU starts the DMA transfer, which then proceeds without CPU intervention.

The DMA interface, which controls the DMA transfer, must be initialised before it can be used. The initialisation routine must tell the DMA interface the memory address where the data block starts, the address of the external device and the number of words to transfer.

Parallel operation

Once started, the DMA transfer runs independently of the CPU. Both the DMA device and the CPU compete to gain access to the ND-100 bus. The bus arbiter, situated on the CPU

card, gives DMA devices priority over the CPU, but allows the CPU access at least once every memory refresh cycle.

As the ND-110/CX normally executes instructions and reads data from cache memory, the CPU performance will not be significantly decreased by DMA activity. The actual change in performance depends on how efficiently cache is being used. This again depends on the program that is being executed.

The ND-110 Standard uses cache memory only for instructions. DMA activity will affect its performance more than the ND-110/CX.

Total bandwidth of 1.8 Mword/s

More than one DMA controller may be active at the same time, sharing the available band width (1.8 Mword/second).

Typical DMA devices are:

- Disks
- Magnetic tapes
- Inter-computer links

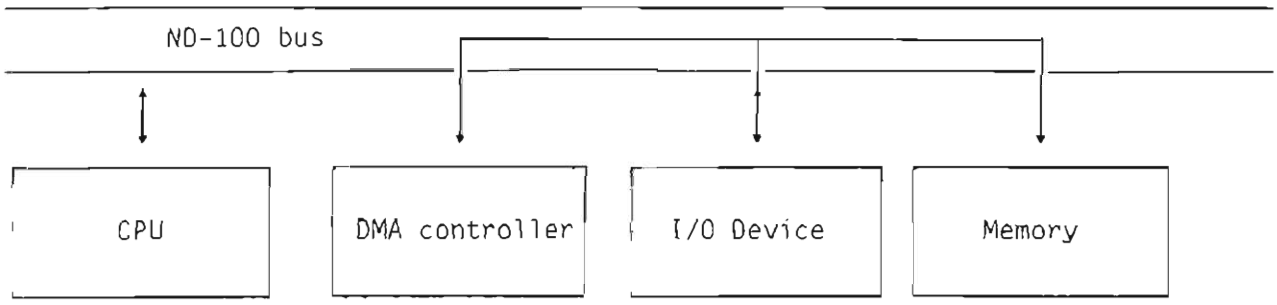


Figure 56. DMA data transfer

Data buffering (FIFO)

All DMA controllers for the ND-110 have at least 16 words buffering between device and memory. This means that data is written to and read from the buffer, instead of transferring directly to or from memory. The buffer is organised as a first in first out (FIFO) memory. If the DMA controller is unable to gain access to the ND-100 bus for short periods, data is accumulated in the FIFO buffer and transferred to memory as soon as the ND-100 bus is available. This form of buffering effectively prevents under-run on output and overrun on input.

6.5.1 DMA TRANSFER

A DMA transfer may be divided into 3 steps:

- Initialisation
- Transfer
- Termination and status check

INITIALISATION

The DMA controller must be initialised before a transfer can be started. For each DMA transfer, the device driver routine will define the following values:

- Start address in memory.
- Word count.
- Device dependent registers.

Start address

The memory address register (MAR) on the DMA controller contains the current memory address. This will be initialised to the first address in memory to be read (DMA output) or written to (DMA input). The address in the register is incremented automatically by the DMA controller during the transfer.

The memory address used by the DMA controller is a 24-bit physical address. The MAR must be loaded in two parts. The 8 most significant bits of the address are written first, followed by the 16 least significant bits.

Word count

The word count register is initialised with the number of words to be transferred.

Device dependent registers

The peripheral device used may require one or more registers to be loaded with control or data words.

For example, a disk controller has registers to specify the cylinder, surface and sector address.

TRANSFER

When the DMA controller has been initialised, transfer is started by writing a word to the control register. The word to be written will depend on the type of DMA controller and the type of transfer requested.

The current memory address register (MAR) is incremented for each word transferred. The word count register is decremented for each word.

TERMINATION

When the word counter is decremented to zero, the DMA transfer is complete. The status register on the DMA controller will indicate that it is ready for transfer again. If the interrupt system has been turned on (ION) and interrupts have been enabled on the controller, a level 11 interrupt request will be generated.

6.5.2 ND-100 BUS SIGNALS DURING A DMA TRANSFER

DMA Input

The figure below shows the control and data signals which are present on the ND-100 bus during a DMA input transfer. The DMA controller starts a bus cycle with by activating the $\overline{\text{BREQ}}$ bus line. The bus arbiter on the CPU card drives the $\overline{\text{BMEM}}$ line active to signal that a bus cycle will access memory.

An $\overline{\text{OUTGRANT}}$ signal is also generated by the bus arbiter. This is the response to $\overline{\text{BREQ}}$, indicating that the bus is available for a DMA cycle. The $\overline{\text{OUTGRANT}}$ signal from the CPU is connected to the $\overline{\text{INGRANT}}$ of the interface card nearest the CPU.

The signal is "daisy-chained" through each interface card until it reaches the interface card which issued $\overline{\text{BREQ}}$. This card holds its $\overline{\text{OUTGRANT}}$ signal inactive (high). In this way, if more than one controller has requested a DMA cycle, the DMA controller with the highest priority (nearest the CPU) wins control of the bus. This is the same technique used for interrupts, where the $\overline{\text{OUTIDENT}}$, $\overline{\text{INIOENT}}$ signals are used.

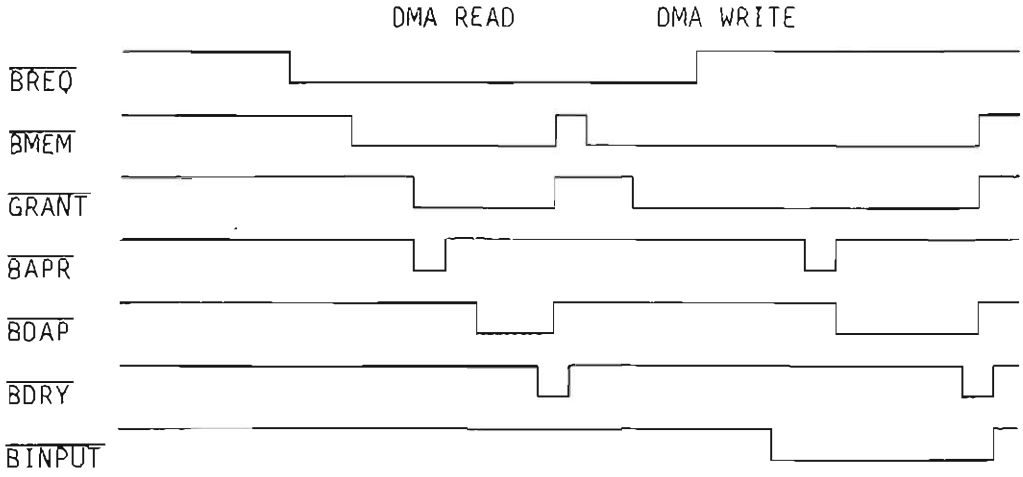


Figure 57. ND-100 Bus signals during a DMA transfer

The BINPUT line, is used by the requesting controller, to signal the direction of the memory operation. The BINPUT signal is driven active (low) to indicate that data is to be transferred from the DMA device to memory.

The DMA controller sends the 24-bit memory address onto the bus, and strobes the BAPR (address present) signal.

DMA input cycle

If the DMA transfer is an input cycle, the DMA controller places the data word on the bus and strobes the BAPR (data present) signal. The memory card that contains the addressed location replies by strobing BDRY (data ready) to acknowledge the transfer.

DMA output cycle

If the DMA transfer is an output cycle, the memory card that contains the addressed location is responsible for placing the data word on the bus and strobing BAPR. The DMA controller replies with BDRY

If the ND-110 computer does not have any memory at that address, no memory card will reply. The bus arbiter detects a hanging bus cycle (after 15µs) and generates a memory out of range (MOR) interrupt. PEA and PES registers contain the address that failed, bit 14 in PES is set (= 1) to show that it was a DMA cycle.

DMA semaphore cycle

The bus arbiter on the ND-110 supports DMA semaphore cycles. Semaphore cycles are bus-locked READ/WRITE cycles. Any DMA device may generate a semaphore cycle, but the feature

has been implemented chiefly for multiport memory. Semaphore bus cycles are described on page 109.

6.5.3 Programming a DMA controller

The following is an example of how a DMA transfer to a disk may be programmed. The DMA controller is in this case an integral part of the interface card for the disk.

This is assumed to have the following write registers:

MEMADR	Memory address register
WORDCNT	Word count register
BLCADR	Block (sector/cylinder) address register
CONTRW	Control word register

and (at least) one read register.

STATUS	Status register

```
INITIALISATION,

  LDA     RESET   % Command word to put the disk controller
                  % in a known state
  IOX     CONTRW  % Write to the control word register

  LDA     UMEMADR % Load most significant part (8 bits) of
                  % the memory start address
  IOX     MEMADR  % Write to the memory address register (MAR)

  LDA     LMEMADR % Load least significant part (16 bits)
                  % the memory start address
  IOX     MEMADR  % Write to the memory address register (MAR)

  LDA     WORDCNT % Load the number of words to be transferred
  IOX     WORDCNT % Write to word count register

  LDA     DISKADR % Load sector/cylinder address
  IOX     BLCADR  % Write to block address reg. (BAR)

  LDA     START  % Load start command word
  IOX     CONTRW % Write to control word register

  ...continued on next page
```

Programming example continued.

```
TRANSFER,
    .           % DMA transfer takes place parallel
    .           % with further CPU activity.
    .

TERMINATION,
    .           % Control reaches this point as a result
    .           % of an interrupt from the DMA controller
    .
  IOX    STATUS % Read status register..
    .           % and check that the transfer has been
    .           % completed without error.
```

6.5.4 I/O DEVICES ON THE CPU BOARD

The real-time clock and console terminal interface are located on the CPU board.

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

CONSOLE TERMINAL INTERFACE

The console terminal interface is on the CPU board. It occupies the I/O address address range 300_8-307_8.

I/O address	R/W	register
300_8	R	read data
301_8	-	not used
302_8	R	read status
303_8	W	read control
304_8	-	not used
305_8	W	write data
306_8	R	write status
307_8	W	write control

Table 13. Console interface registers

Read data (300_8)

The read-data register (address 300_8) contains the most recently received character.

Read-status (302_8)

The read-status register (address 302_8) contains the current status of the input channel. The bits are assigned as follows:

Bit 0 : Set (= 1) if interrupt on data available

Bit 1 : always zero

Bit 2 : always zero

Bit 3 : Set (= 1) if data is available[1]

Bit 4 : Set (= 1) if data is in error (one or more of bits 5-7 set).

Bit 5 : Set (= 1) if there was a framing error.

Bit 6 : Set (= 1) if there was a parity error.

Bit 7 : Set (= 1) if there was an overrun.

Bits 8-15 are always zero.

Note 1. Bit 3 is never set when the CPU is in OPCOM mode.

Read-control (303_8)

The read-control register (address 303_8) is used to set the input channel parameters. Bits 0, 11, 12 and 13 and 14 are used. Unused bits should be set to zero.

> Note
> The word length and parity settings also apply for the output channel.

Bit 0 : Set to 1 to enable interrupt when data is available.

Bits 11 & 12 : Bits 11 and 12 determine the word length. Parity, if used, adds 1 extra bit to the word length.

Bit 11	Bit 12	word length
1	1	5 bits
0	1	6 bits
1	0	7 bits
0	0	8 bits

Table 14. Terminal interface word length

Bit 13 : Set (=> 1) for one stop bit. Reset (=> 0) for two stop bits (1.5 for 5-bit word length)

Bit 14 : Set (=> 1) to make the interface check parity. The word length will be increased by one when parity is being used.

Write-data (305_8)

Data written to the write-data register (address 305_8) will be sent to the output channel.

Write-status (306_8)

The write-status register contains the current status of the output channel. The bits are assigned as follows:

Bit 0 : Set (= 1) indicates that the interface will generate an interrupt when it is ready for transfer.

Bit 3 : Set (= 1) indicates that the transmitter is ready for transfer (data may be written).

Bits 1-2 and 4-5 are not used (always zero).

Write-control (307_8):

The write-control register (address 307_8) is used to set the output channel parameters. Only bit 0 is used. Set the other bits to zero.

Bit 0 : Set (=> 1) to generate interrupts when the device is ready for transfer (a new data word may be written to the write-data register).

The word length and parity (if used) are the same as set in the read-control register.

THE REAL-TIME CLOCK

The real-time clock on the CPU board occupies device-register address range 10_8-13_8. Address 10_8 is not used (returns 0_8).

Clear real-time clock (11_8)

Writing to this address causes the next clock pulse to occur exactly 20 ms later. If this instruction is executed repeatedly, the counter will never be incremented, and no clock pulses will occur.

Read clock status (12_8)

Bit 0 : Set (= 1) means that the clock will generate an interrupt when next clock pulse arrives.

Bit 3 : Set (= 1) means that the clock is ready for transfer (that is a clock pulse has occurred). Bits 1-2 and 4-15 are always zero.

Set clock status (13_8)

Bit 0 : Set (=> 1) to enable interrupts when ready for transfer.

Bit 13 : Set (=> 1) to clear the ready for transfer bit in the clock status register.

6.5.5 PANEL PROCESSOR PROGRAMMING SPECIFICATION

The operator panel and optional display is controlled by a microprocessor. The panel processor is accesible from program by means of two internal registers:

- PANS, Panel Status (read only) The PANS register is also used to send data from the panel processor to the CPU.
- PANC, Panel Control (write only) The PANC register is buffered by a first-in/first-out (FIFO) queue. It is important to check if the FIFO is full before sending commands to the panel processor.

The instructions TRA PANS, and TRR PANC are privileged instructions.

The microprogram and the display

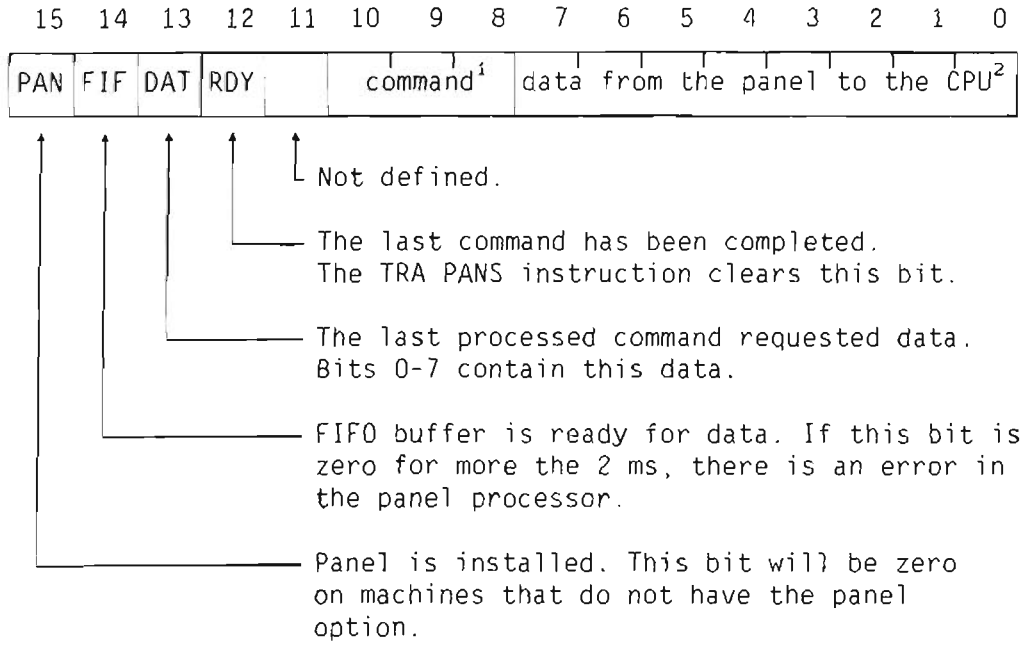
The microprogram sends commands and data to the panel processor every 20 ms during normal operation of the CPU. The FIFO buffer absorbs the commands and allows the panel processor to process the commands at its own pace.

Commands to the panel processor from a program come in addition to this steady stream from the microprogram, and it is the programmer's responsibility to check that the FIFO buffer is able to accept commands (PANS bit-14, see below), before using the TRR PANC instruction.

PANEL STATUS REGISTER (PANS)

The program can read the panel status register at any time using the privileged instruction TRA PANS. The result in the A register consists of two 8-bit fields. The upper half (bits 8-15) contains information about the status of the panel processor and its interface. The panel processor uses the lower half (bits 0-7) to send data to the CPU.

The format is as follows:



Note (1) : Bits 8-10 contain the last command processed.

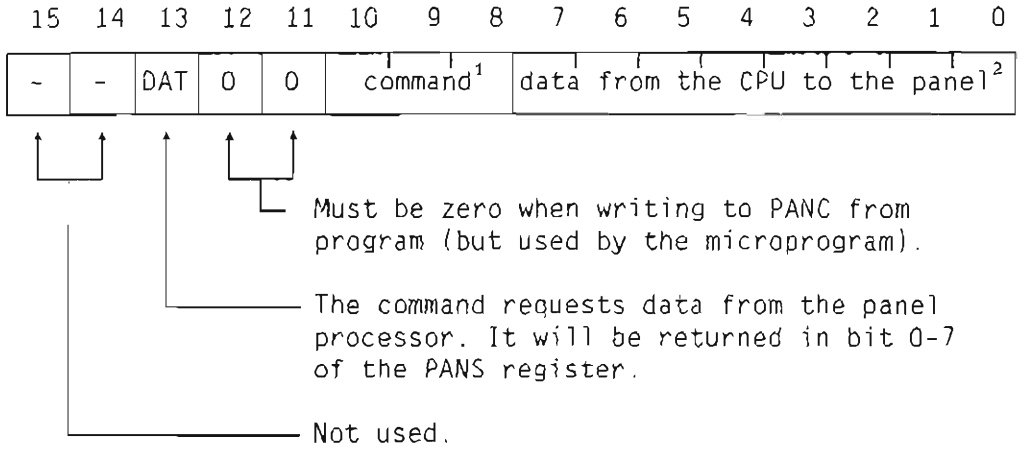
(2) : Bits 0-7 contain the data requested by the last processed command. If no data was requested, bits 0-7 will contain bits 0-7 of the command word written to PANC.

Figure 58. Panel status register (PANS)

PANEL CONTROL REGISTER (PANC)

Commands and data to the panel processor are written to the panel control register with the privileged instruction TRR PANC.

The A register must have the following format before executing the instruction:



Note (1) : Panel processor command. See below for legal values.

(2) : Data to the panel processor. Its interpretation depends on the command (see below).

Figure 59. Panel control register (PANC)

PANEL PROCESSOR COMMANDS

The table below gives the seven possible values of the command field (bits 8-10).

Command value	Interpretation
000	Illegal
001	Reserved
010	Message value (Write only)
011	Message control (Write only)
100	Clock Low Seconds (Read/Write)
101	Clock High Seconds (Read/Write)
110	Clock Low Days (Read/Write)
111	Clock High Days (Read/Write)

Table 15. Panel processor commands

The message and clock commands are descibed in detail in the following two sections.

PLACING A MESSAGE ON THE DISPLAY

Programs can send ASCII characters to panel display. These will be shown four characters at a time. A message of up to 40 characters can be displayed by commanding the panel processor to rotate the message.

The message control command interprets the data in bits 0-2 as follows:

000 : Stop rotating the message

001 : Return display to normal function

010 : Clear text buffer and Function display.

100 : Rotate the message in the text buffer, displaying four characters at a time.

110 : Clear the text and start rotation (command 010 plus command 100)

The display can also be returned to normal function from OPCOM with the F command (see page 185).

The message value field is interpreted as an ASCII character. It is placed at the end of the 40-character text buffer. When the buffer is full, further characters are ignore until the buffer is cleared.

UPDATING THE CALENDAR CLOCK

The clock calendar can be set using the clock commands.

The clock can also be adjusted using the SINTRAN commands @UPDAT or @CLADJ.

Note
Early versions of the ND-110 CPU (print no. 3090) did not have a separate cell for the clock, but took power from the backup power supply.

The calendar clock draws its backup power from a lithium cell on the CPU card. This cell will keep the clock running for many years. If the calendar clock needs setting after a power failure, it may mean that the lithium cell needs replacing.

CHAPTER 7 OPERATOR INTERACTION

CHAPTER 7 OPERATOR INTERACTION

The operator of the ND-110 computer can gain control over the ND-110 computer by using:

- the control panel
- the console (terminal number 1)

The control panel allows to operator to start and stop the ND-110 computer. Many versions of ND-110 have a key-switch which can be used to disable the control panel. To prevent unauthorised use it should normally be locked. In order to use the control panel the key-switch (normally on the right hand side of the panel) must be turned to the **ON** position.

7.1 CONTROL PANEL

The description below refers to the operator panel currently delivered with Norsk Data computers. Earlier models used a different panel. If your computer has been upgraded with an ND-110 CPU, you should refer to the documentation delivered with your computer.

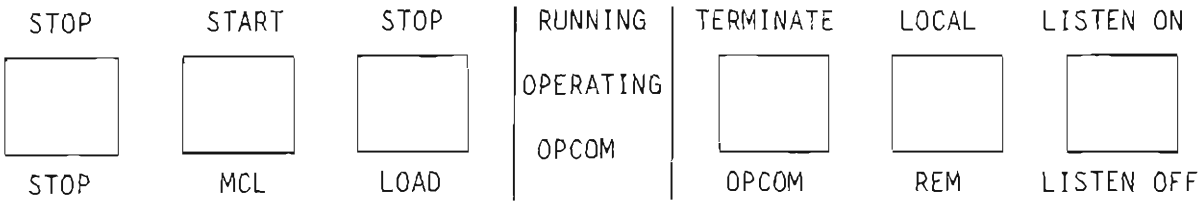
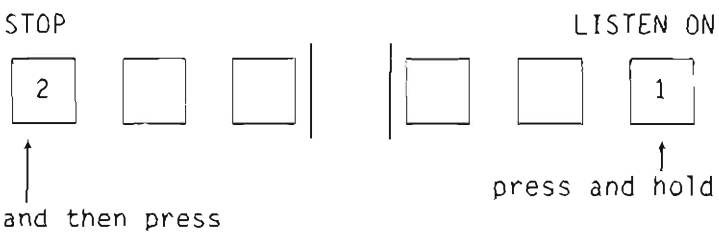


Figure 60. The Operator's panel

Setting the control panel in service mode

To set the panel must be put into service mode (also called advanced mode) press and hold the rightmost button depressed while pressing the leftmost button.



When the panel key is unlocked and the panel is in service mode, the panel push buttons are active and have the following effect:

MCL

The MCL (master clear) button is used to force the computer into a defined initial state. The CPU loads the microprogram into the control store from the EPROM where it is stored. The microprogram then traps to the master clear routine. This initialisation is also performed when the CPU goes through the power up sequence, and when the bus line called $\overline{\text{BMCL}}$ is activated.

Note

When the MACL command is used in OPCOM mode, the microprogram performs the same initialisation, but the control store is **not** loaded. Use the MCL button on the operator panel if you want to reload the microprogram.

The master clear routine turns off running indicator, the PIE register is cleared. The paging and interrupt systems are turned off. The paging system is set in "normal" mode (as if the REX instruction had been executed). The CPU self-test routine microprogram is executed. If no errors are found, the running indicator lamp is lit, and the terminal interface on the CPU board (terminal no. 1) is initialised to 7-bit plus even parity. Parity is not checked on input.

When the master clear routine is finished, the CPU will be in STOP mode.

STOP

The STOP button has the same effect as entering the STOP command when the CPU is in OPCOM mode. The CPU enters STOP mode and the OPCOM indicator will be lit. In STOP mode the CPU will respond to input from the console (terminal no. 1) as for OPCOM mode.

LOAD

The LOAD button has the same effect as entering the & (ampersand character) command when the CPU is in OPCOM mode. It will load the operating system (SINTRAN) from the mass storage device specified by the ALD edge-switch on the CPU board.

OPCOM

Pressing the OPCOM button puts the CPU in OPCOM mode. In this mode the console (terminal no. 1) communicates directly with the microprogram. When the CPU is in OPCOM mode, interrupts from the console are disabled. Data from the console goes directly to the microprogram. OPCOM mode is terminated by the pressing the escape (ESC) key on the console.

INDICATOR LIGHTS

The operator panel contains a number of text fields which are used to indicate the current CPU state. A text field becomes visible when the indicator lamp behind it is lit.

Running When the *running* field is lit, the computer is in its normal operating mode. This field is lit when SINTRAN is running.

Opcom When this field is lit, the console is in direct communication with the microprogram.

Operating The *operating* field is used on ND-500 computers to indicate activity on the ND-500 CPU. In normal operation this field flashes on and off. The operating field is not used on ND-110 computers.

DISPLAY PANEL

The layout shown below is the format used in current models. For computers upgraded to ND-110 you should consult the documentation which was supplied with the computer.

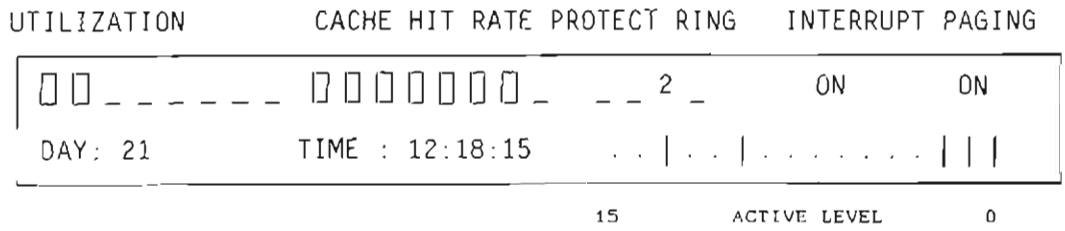


Figure 61. The display panel

UNDERSTANDING THE DISPLAY

Utilisation The eight ▯ utilisation indicators are displayed in progression to indicate how much time the ND-110 spends in program levels 1 to 15 (ie. not in the idle loop). Typically only a few indicators are displayed.

Hit

The eight (cache) hit ▯ indicators are displayed in progression to indicate how many memory accesses are found in cache. The ND-110 CPU operates fastest when most memory accesses are found in cache. Typically most of the indicators are displayed. The cache system is described on page 115.

Ring

The ND-110 CPU ring protection system has four levels (rings). When the ND-110 CPU is running under the SINTRAN operating system, the ring usage is:

0 Background RT programs (user programs)

1 RT programs allowed to access RTCOMMON

2 SINTRAN and RT programs using privileged instructions

3 SINTRAN segment administration

Paging must be **ON** for ring protection to function. Ring protection is described in more detail on page 90.

Paging

ON is displayed when paging is on. The field is blank when paging is off. During normal operation of SINTRAN, paging is **ON**.

Interrupt

ON is displayed when the interrupt system is on. The field is blank when the interrupt system is off. During normal operation of SINTRAN, the interrupt system is **ON**, but turned off for short intervals.

Active level

This display shows the most recent program (interrupt) levels used. The program levels are often active for too short a time to be visible if they were only displayed while they are active. The black | segments are displayed long enough to be visible to the human eye.

A description of how SINTRAN uses the various program levels may be found on page 42.

7.2 OPERATOR COMMUNICATION FROM THE CONSOLE (OPCOM)

The operator may communicate directly with the microprogram from the console (terminal no. 1). To do this the CPU must be in OPCOM or STOP mode. The *Opcom* field will be lit. OPCOM mode may be entered from SINTRAN by

entering the @OPCOM SINTRAN command or by pressing the OPCOM button on the display panel.

The operator can use these routines to:

- Load and start the operating system (SINTRAN)
- Perform backup and maintenance tasks
- Debug programs

> **Note**
> All values and addresses in the following description of OPCOM are assumed to be in **octal** except when explicitly stated otherwise.

Commands to the microprogram consist of one or more characters. All characters entered are significant. Spaces are not permitted within commands. The space character itself is interpreted as a command to ignore all previously entered characters. Most commands are interpreted immediately and need not (must not) be ended with carriage return. When a command requires a carriage return, it is shown (↵ sign).

The following commands may be used whenever the CPU is in OPCOM mode.

Command	Effect
/	Examine memory location or register
RD ↵	Dump registers
E ↵	Specify physical or virtual address for / command
F ↵	Specify the display format
*	Print the address of the last examined memory location
ESC	Terminates OPCOM mode. This command has no effect if the CPU is in STOP mode.
MACL ↵	master clear (but no microcode load)
STOP ↵	Puts CPU in STOP mode

Table 16. OPCOM commands

Many of the above commands may require parameters before the commands in order to work. See the description of the individual command for details. The following commands are legal only in STOP mode:

Command	Effect
!	Start program in main memory.
Z	Single step instruction.
$ or &	Bootstrap load.
.	Set breakpoint.
"	Manual instruction command.
#	Start microprogrammed memory test.

Table 17. Commands in STOP mode

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

7.3 LOAD COMMANDS ($ AND &)

The commands **$** and **&** both cause the CPU to load (and possibly execute) a program from a storage device. The device address is defined by an octal value entered immediately before the command. If the value is omitted, the load will be defined by the setting on the Automatic Load Descriptor (ALD) switch on the CPU card.

Note
An extended power failure is a power failure that lasted longer than the capacity of the standby battery, thereby resulting in memory contents being lost.

The following table shows how the ALD switch setting affects the load operation. The action taken is the same regardless of whether the load was due to an extended power failure, the load commands (**$** or **&**) or the LOAD button.

ALD I12[4]	Load action[1]
15 0	*Note 2*
14 1560	BPUN load from floppy (1560_8) and run[3]
13 20500	Bootstrap load from Winchester disk (500_8) and run[3]
12 21540	Bootstrap load from SMD disk (1540_8) and run[3]
11 400	BPUN load from paper tape (400_8) and run[3]
10 1600	BPUN load from HDLC (1600_8) and run[3]
9	Run[3] (no load)
8	Run[3] (no load)
7 100000	*Note 2*
6 101560	Binary load from 1560
5 120500	Mass storage from 500
4 121540	Mass storage from 1540
3 100400	Binary load from 400
2 101600	Binary load from 1600

Table 18. ALD switch settings

Note 1: The action will be taken if
a. **$** or **&** (without preceding value) has been typed
b. The LOAD button has been pressed
c. The power has been restored and the keyswitch is in the lock position, but the standby power has been lost (extended power failure).

Note 2: No load. The CPU is put in STOP mode.

Note 3: Run from address 20_8.

Note 4: Contents of internal register I12 reflects ALD setting

ALD switch settings 8 to 15 specify load and run, settings 2 to 7 specify load only. ALD settings 4, 5, 12 and 13 specify a bootstrap load from a disk. All other settings expect BPUN format. The start address is always the power fail restart address (20_8).

LOAD FROM AN OPERATOR SPECIFIED ADDRESS

The operator may specify the device address from the console. The device address is entered immediately before the **&** (or **$**). The default format is BPUN. SMD and Winchester disks use bootstrap format. To specify a bootstrap load set bit 13 of the device address to 1 (i.e. if the device address is 1550_8, enter **21550&**).

START PROGRAM (!)

Execution is started at the address entered immediately preceding the ! command. If no address is given, the current value of the program counter (P register is used).

Some important address when running SINTRAN are:

20_8 Power fail restart address

21_8 Warm start address

22_8 Cold start address

INTERNAL MEMORY TEST (#)

A memory test routine in the microprogram may be started with the # command. Memory is tested in banks (segments) of 64 Kword. The bank number is entered immediately before the # character.

A second # character is printed on the console if the test is successful. If an error is found, the test stops and ? is sent to the console. The registers then contain the following information:

T: Failing bits

P: Failing address

D: Error pattern

L: Test pattern

B: Start address

X: Stop address

7.4 PROGRAM DEBUGGING COMMANDS

Some OPCOM commands are chiefly intended for program debugging. These commands permit the user to alter memory locations and registers, set breakpoints, single step the CPU and execute a single instruction entered from the console.

SINGLE STEP EXECUTION

A single Z character will cause one instruction (or one interrupt level change) to be executed. If an value is entered before the Z character, that number of instructions will be executed.

Page faults, protect violations and interrupt level changes are executed correctly, but are counted as extra instructions. An extra overhead of approximately 3 µs is introduced between each instruction when the CPU single-steps instructions.

SET BREAKPOINT (.)

Enter the breakpoint address followed by the . (full stop/period) character. When the program reaches the entered address, execution stops and the . character is echoed to the console. An extra overhead of approximately 3 µs is introduced between each instruction when the CPU single-steps instructions.

If the specified address is never reached, execution continues until a character other than 0-7 or A-Y is typed.

EXECUTE ENTERED INSTRUCTION (")

This command starts continuous execution of the instruction specified as argument. The execution stops when a character other than 0-7 or A-Y is typed.

Example:

The paging-on instruction (PON) has the octal code 150410_8. The OPCOM command **150410"** turns on the paging system.

READ/WRITE I/O DEVICE (IO/)

Enter the device address followed immediately by **IO/**. The CPU executes an IOX instruction using the given device address.

The usual rules apply for direction (even address: input to CPU; odd address: Output from CPU).

Output data is taken from the OPR pseudo-register (See page 182). Input data is displayed on the console, but not stored anywhere. None of the working registers is affected.

PRINT CURRENT LOCATION (*)

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

EXAMINE MODE (E)

> Note
> The number used here is the actual page number. This is different from the ND-100 CPU where a code was used.

To specify the page table to used for subsequent OPCOM memory commands. The command is preceded by the page table number (0 to 17_8).

If no number is given, subsequent memory commands will apply to physical memory.

Example:

5E↵

If paging is on, future memory references will be made via page table 5.

EXAMINE MEMORY (/)

Enter the memory location address followed immediately by the / character. The contents of the location will be echoed to the console.

You then have the option of changing the contents of that location or viewing the contents of the next location.

Changing the contents in STOP mode

If you enter an octal value followed by carriage return, the memory location will be changed to that value. The contents of the next location will be displayed and, if you wish, you may change that location in the same way. Just entering carriage return

displays the contents of the next location without changing the previous one.

Changing the contents in RUN mode Changing the contents of memory locations while SINTRAN is running can produce unpredictable results! To guard against doing this inadvertently, you must follow the octal value by **DEP** (deposit).

Examples: In the following examples user input is underlined.

#100/137777	display contents of address 100_8
#100/137777 0↵	display contents of address 100_8 and change it to 0 (CPU is in STOP mode)
#100/137777 0DEP↵	the same as the second example but CPU is in RUN mode

If the paging system is used, you may specify virtual addressing by setting the page table to be used with the E command.

If you specify virtual addressing, page faults and protect violations are ignored. If physical addressing is used, the address may contain up to 24 bits (8 octal digits).

MEMORY DUMP (<)

To dump the contents of an area of memory to the console, enter the start address, the memory dump character <, the end address and terminate the command with carriage return. The contents of the memory addresses between the start address and the end address are printed on the console, with 8 addresses per line.

The dump is taken from the 64 Kword memory bank (segment) last addressed by a memory examine command /. A memory examine command should always be done before a memory dump. The dump may be stopped by pressing any key.

EXAMINE REGISTERS (R/)

The form of this command is the same as for the examine memory command /. The register name is written before the command instead of the memory address. The program level may be specified ($0-17_8$) before the register name. If the program level is omitted, program level zero is assumed.

The contents of the register is displayed after the / character. If you want to change the contents, enter the new value in the same way as for the examine memory command. If the CPU is in STOP mode, you must end the value with "DEP".

Working registers may be specified by either the form Ry where y is a number in the range 0 - 7 or the name of the register.

R0 - R7 correspond to : S, D, P, B, L, A, T, X respectively.

Internal registers are addressed as Iy where y is a number in the range $0 - 15_8$.

$I0 - I15_8$ correspond to the internal registers:

PANS (0), STS (1), OPR (2)*, PSR (3), PVL (4), JIC (5), PID (6), PIE (7), CSR (10_8), ACTL (11_8), ALD (12_8), PES (13_8), PCR (14_8), PEA (15_8).

Note (*) : OPR is a simulated panel switch register which can be written to from OPCOM. Programs can read the contents with the TRA OPR instruction.

Examples:

#A/126500	examine A register on level 0
#7P/140003	examine P register on level 7
#7R2/140003	examine R2 (= P) on level 7
#I7/030013	examine PIE register (internal register 7)
#OPR/00100	examine the OPR pseudoregister

DISPLAY PSEUDO-REGISTERS

In addition to the registers listed above, there are three other pseudo-registers that may be addressed. Addressing these pseudo-registers affects the (optional) display panel only.

ACT : The display shows active levels, clock and activity (normal display).

BUS : The display shows bus activity. A two digit code preceding the command specifies what data is displayed.
First digit:

0 = CPU data is displayed

1 = DMA data is displayed

2 = CPU address is displayed

3 = DMA address is displayed

Second digit:

0 = nothing is displayed

1 = Read access only is displayed

2 = Write access only is displayed

3 = Read and write are displayed

Example:

23BUS/ The display will show all data written from the CPU to memory. The function field of the display will show "ACWR".

OPR : The display shows the contents of the OPR register. This is a simulated panel switch register which can be written to from OPCOM. Programs can read the contents with the TRA OPR instruction.

U : The display shows the contents of a scratch register which can be written to by the TRR LMP instruction. This is used by the DISC-TEMA program to show the cylinder number during disc operations.

REGISTER DUMP (XX<YYRD)

The contents of the working registers on program levels xx to yy are displayed on the console. One register set is displayed per line. The registers are printed in the following order: STS, D, P, B, L, A, T, X.

To dump the register set on only one level xx must be equal to yy. If xx and yy are omitted (<RD), the registers on level 0 are dumped.

INTERNAL REGISTER DUMP (IRD)

The IRD command displays the 16 internal registers of the CPU. This command is only allowed when the CPU is in STOP mode. This is to avoid unintentional unlocking of PEA, PES and IIC when the CPU is running.

SCRATCH REGISTER DUMP (XX<YYRDE)

The RDE command dumps the contents of the 8 scratch registers (only microprogram accessible) on program levels xx to yy. One scratch register set is displayed per line. This command is intended for debugging microprograms only.

7.5 DISPLAY FORMAT (UUZZYXF)

This command will define the display format when the optional display unit is included in the system. uuzzyx are octal digits and define the chosen format. F, without argument, (or with argument equal to zero) will set the default display format, which is octal format. The fields of the argument have the following meaning:

x : Number representation code. x = 0 Displayed data is in octal representation. zz has no effect.

x = 1 Displayed data is in unary representation, i.e. 4 of the bits in the displayed data are used to light one out of 16 indicators. zz indicates which 4 bits to decode.

x = 2 Displayed data is in binary representation. zz has no effect.

y : "Stretch" code.

y = 0 No strectching

y = 1 Zeros are stretched.

y = 2 Ones are stretched.

y = 3 Zeros and ones are stretched.

zz: Lower start bit for unary display.

zz = $0\text{-}24_8$ Position of lowest bit position to be represented in unary representation.

uu: Display processor maintenance codes (4 bits)

uu = 1 Display year and month

uu = 2 Inhibit message

uu = 4 Initialize panel processor

uu = 10_8 Abort message

See page 167 for details about displaying messages on the panel.

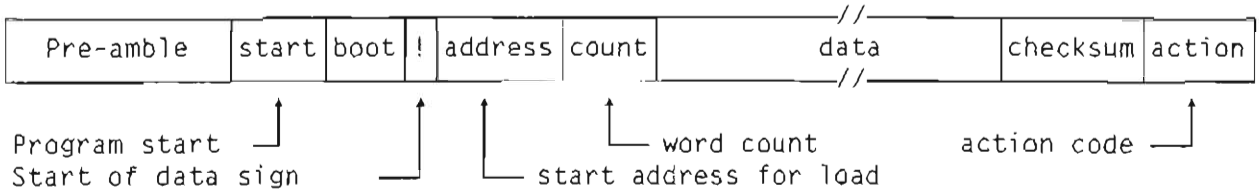
Example:

1421F↵

After this format specification, bits 14_8 - 17_8 (bits 12 - 15) will be shown in unary representation with afterglow on ones. (If the display shows an address, this is equivalent to pushing the DECODE ADDRESS button on NORD-10/S.)

7.6 BPUN LOAD FORMAT

The BPUN format is defined as follows:



Pre-amble : This may contain any characters except "!" (41_8). It's original use was a bootstrap loader for stand alone CPUs. This is no longer used. (It is, however, generated by the NRL command BPUN).

Start : A field containing an ASCII encoded octal start address for the program. It is terminated by a carriage return (15_8) and optionally a line feed (12_8).

Boot : A field containing an ASCII encoded octal value giving the start address of the above mentioned bootstrap loader and terminated by the "!" character (i.e. the next field). No longer used.

"!" : Start of data signal.

Address : Address where the binary load will start. Two bytes (most significant first).

Count : Number of words in the following data field. Two bytes (most significant first).

Data : A field consisting of ‹Count› (16 bit) words. Each word is recorded as two bytes, the most significant byte first.

Checksum : Arithmetic sum of all the words in the data field truncated to 16 bits (ie. modulo 2^{16}). recorded as two bytes, the most significant byte first.

Action : Two byte field. If the action field is zero, execution will start at the address specified in the start field. If the action field is no zero, the CPU will remain in OPCOM mode. The P register will contain the value read from the start field.

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

CHAPTER 8 NEW FEATURES IN ND-110

CHAPTER 8 NEW FEATURES IN ND-110

8.1 WHAT IS DIFFERENT FROM THE ND-100?

The main areas of change are:

- Physical size
- New technology
- Cache-memory strategy
- Address arithmetic
- Interrupt system
- Control store
- Control logic and timing

PHYSICAL SIZE

The ND-110/CX CPU now includes CPU, memory management system, cache memory and operator panel processor. The reduction in physical size is due to extensive use of gate arrays and PALs.

NEW TECHNOLOGY

The ND-110/CX CPU has been designed using gate-array technology. Gate arrays are "semi-custom" very large scale integrated (VLSI) circuits.

Semi-custom means that the gate-array manufacturer designs circuits with unconnected logic (gate) elements. The customer (in this case Norsk Data) specifies the function of the the circuit by telling the manufacturer how the elements are to be inter-connected.

The ND-110/CX CPU has three gate arrays:

- RMIC
- RMAC
- BUFALU

RMIC

The micro instruction controller, RMIC, contains the circuitry that previously consisted of three sequencer packages and about 30 other logic ICs.

BUFALU

This replaces the four bit-slice arithmetic and logic unit (ALU) circuits, some additional registers like the data bus register and the general purpose register, and the working register set.

RMAC - the microaddress controller

This is an implementation of ND-110 address arithmetic in hardware. In the ND-100 this function was performed by the microprogram.

New Cache-memory Strategy

The ND-110/CX CPU has a novel implementation of cache memory for microinstructions. The step known as mapping in the ND-100 is thereby avoided, because the first microinstruction word of a Macroinstruction is stored in cache memory.

Address Arithmetic

The address arithmetic in the ND-110/CX CPU is implemented in hardware by the RMAC gate array. This is an advantage compared to the ND-100, which performed address arithmetic in microprogram.

THE INTERRUPT SYSTEM

Changes in hardware architecture have created changes in the interrupt system. Unlike the ND-100 CPU, the ND-100/CX CPU handles synchronous interrupts as traps. This is similar to the ND-500. Asynchronous interrupts have not been affected.

THE CONTROL STORE

The control store uses, read/write memory (RAM). At power up it is initialised with the standard microprogram which is stored in two read only memory circuits (EPROM).

The standard microprogram can be modified using a special instruction implemented in the ND-110/CX CPU. The contents of the control store can also be read from program.

CONTROL LOGIC AND TIMING

Much of the control logic, timing circuits and bus interface logic have been designed using programable array logic (PAL).

The main oscillator is now a 39.3216 Mhz crystal oscillator. This oscillator is used for:

- the nano-sequencer
- the CPU clock
- sampling in the bus arbiter
- the real-time clock
- the serial console interface (the UART)

The ND-110 nano-sequencer is a four-bit state-machine. Its output is used throughout the CPU for timing and control.

NEW INSTRUCTIONS

TRA CS opcode 150017_8
Description:

Reads 16 control store bits into the A-register. The X-register contains the control store address.

TRR CS opcode 150117_8
Description:

Writes the A-register into 16 control store bits. The X-register contains the control store address.

For both the instructions TRA CS and TRR CS we have:

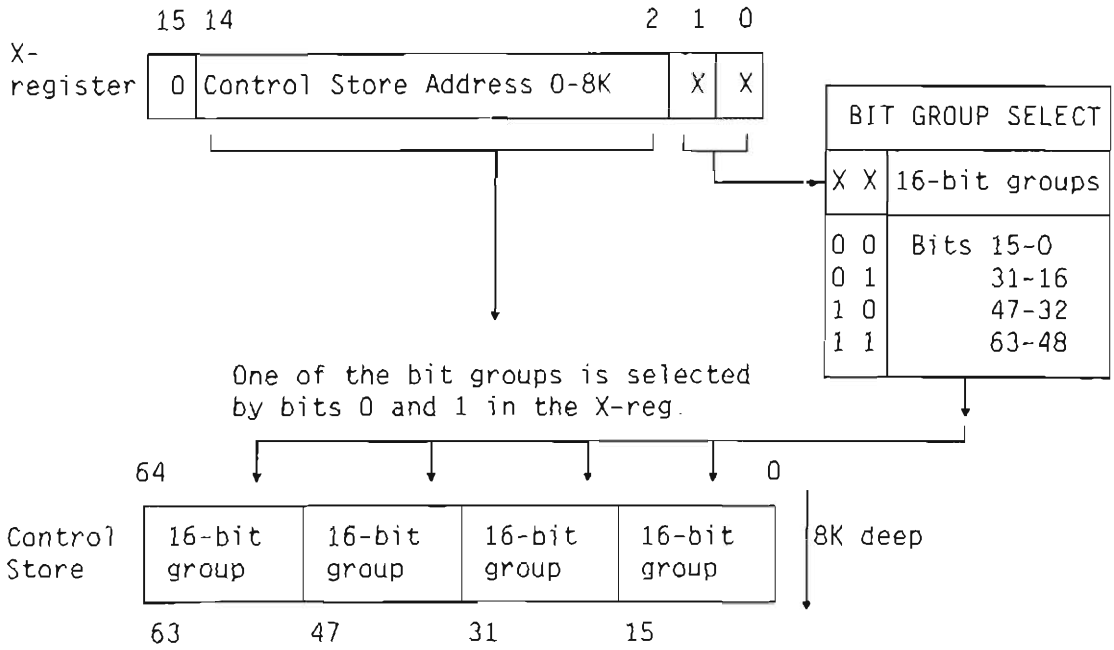


Figure 62. Control Store Bit Group Selection

TRR CILP opcode 150113_8
Description:

The cache inhibit page instruction allows the programmer to inhibit individual pages in cache.

The format of the A-register is shown in the following figure:

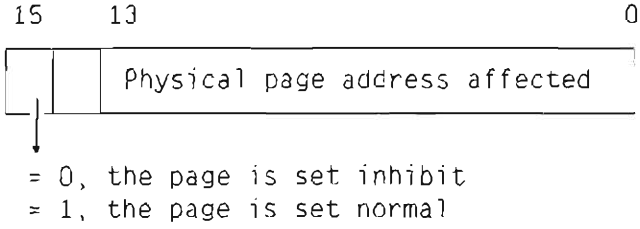


Figure 63. Cache Inhibit Page Instruction

VERSN opcode 140133_8
Description:

Reads version numbers of print and microprogram.

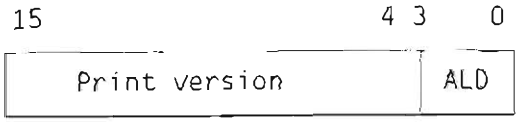


Figure 64. A-register after VERSN

1. Print version into bits 4-15 of the A-register. Bits 0-3 is the automatic load descriptor (ALD) switch setting.

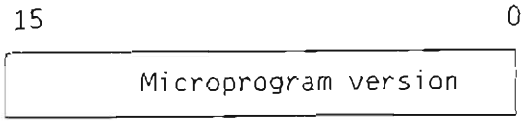


Figure 65. T-register after VERSN

2. Microprogram version into the T-register.

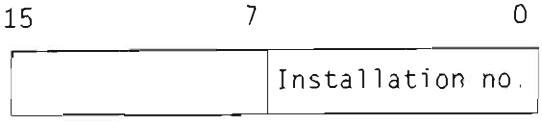


Figure 66. D-register after VERSN

3. One byte of the installation number (16 byte number) into bits 0-7 of the D-register. Bits 8-15 are not defined. Bits 8-11 of the A-register define which of the 16 bytes are read, and you have to

   load the A-register with these four bits before you execute the VERSN command.

   You must repeat the instruction 16 times to read the complete installation number.

SETPT opcode 140300_8
Description:

Set page tables.

This instruction can replace the following instructions:

```
SETPT:  JXZ     * 7
        LDDTX   20
        BSET    ZRO 130  DA
        LDBTX   10
        STD     ,B
        LDXTX   00
        JMP     *-6
```

Each time the loop is executed (until X becomes zero) two consecutive physical memory locations addressed by X are loaded into the A and D registers.

The word in A is the protect field of the page table, bit 11 (the PGU bit) is cleared to set the page table. The double word (in A and D) is then stored in two consecutive locations pointed to by the contents of the B register, the page table address.

* is the mnemonic for P relative addressing.

CLEPT opcode 140301_8
Description:

Clear page tables.

This instruction can replace the following instructions:

```
CLEPT:  JXZ     * 10
        LDBTX   10
        LDA     ,B
        JAZ     *3
        STATX   20
        STZ     ,B
        LDXTX   00
        JMP     *-7
```

Each time the loop is executed (until X becomes zero) the physical memory location addressed by X is loaded into the B register.

The B register contents provide the address of a page table entry, which is loaded into the A register.

If the page table entry is zero (unused) the loop is restarted.

If the page table entry is not zero (used) it is stored in a physical location addressed by X (8 locations away from its original entry) and the original page table entry cleared by placing zero in the location addressed by the B register.

The physical location addressed by X is then loaded into the X register itself and the loop restarted.

CLNREENT opcode 140302_8
Description:

Clear non re-entrant pages.

The contents of the memory address at A + 2 are read to find the page table to be cleared along with the SINTRAN RT bitmap (addressed by the X and T registers). The page table entries corresponding to those bits set in the RT bitmap are then cleared.

CHREENTPAGES opcode 140303_8
Description:

Change page tables.

The X register is used to address the current (R1) and previous (Rp) scratch registers .

If the R1 is zero, the re-entrant page has nothing to change so the loop is left, otherwise the contents of the memory location pointed to by the R1 + 2 are loaded into T.

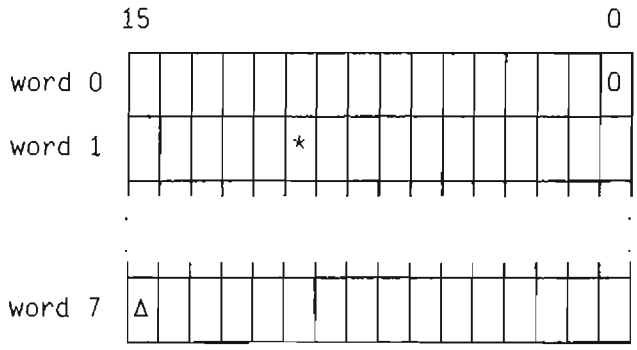
T then contains the protect table entry, if the page has not been written to (WIP bit 12 is zero) T and R1 are loaded with Rp. R1 (now containing Rp) is tested again for zero. If the page has been written to, the T register is loaded with the contents of the second scratch register (R2), pointed to by R1, and R2 becomes the address of Rp. X is loaded with R1 as the new pointer to the re-entrant pages and Rp is loaded into the D register pointed to by A.

CLEPU opcode 140304_8
Description:

Clear page tables and collect PGU information.

This instruction collects information on the PGU (page used) bit of a page table entry whilst executing CLEPT.

The instruction places PGU information in an eight word table called **the page map bank**. Each bit in the bank represents the status of a page's PGU bit as follows:



Δ denotes page 177_8 PGU bit
* denotes page 32_8 PGU bit
0 denotes page 0_8 PGU bit

The L register contains the address of the map entry.

WGLOB opcode 140500_8
Description:

Initialize global pointers.

(T) = bank number of segment table (STBNK)
(A) = start address within bank (STSRT)*
(D) = bank number of core map table (CMBNK)

* must be divisible by 8

RGLOB opcode 140501_8
Description:

Examine global pointers.

(T) ⟵ bank number of segment table (STBNK)
(A) ⟵ start address within bank (STSRT)
(D) ⟵ bank number of core map table (CMBNK)

INSPL opcode 140502_8
Description:

Insert page in page list.

```
R1 := (stbnk,B).7
X  =: (stbnk,B).7
R1 =: (cmbnk,X).0
 IF R1<>0 THEN
    Q := (cmbnk,R1).1
    X =: (cmbnk,R1).1
 ELSE
    Q := ( (B - stsrt) / 2 ) + 3
 ENDIF
Q := (cmbnk,X).1
T =: (cmbnk,X).3
```

REMPL opcode 140503_8
Description:

Remove page from page list.

```
R1 := (cmbnk,X).0
R2 := (cmbnk,X).1
 IF R2^3 = 0 THEN
    R1 =: (cmbnk,R2).0
 ELSE
    Q := (R2 * 2) + strst
    R1 =: (stbnk,Q).7
 ENDIF
 IF R1<>0 THEN
    R2 =: (cmbnk,R1).1
 ENDIF
0 =: (cmbnk,X).0
0 =: (cmbnk,X).1
```

CNREK opcode 140504_8
Description:

Clear non re-entrant pages.

```
Q := (stbnk,A).2
 IF A=0 THEN
    EXIT
 ENDIF
 R1 := ( (Q^1700) * 2) + 174000
 DO FOR R2=X TO X+10
    R4 := (T,R2).0
    IF R2 = X+10 THEN
       EXIT
    ENDIF
    DO  FOR lc=0 TO 17
        IF bit(lc,R4) = 1 THEN
           0 =: (R1).0
        ENDIF
        R1 := R1 + 2
    ENDDO
ENDDO
```

CLPT opcode 140505_8
Description: Clear segment from page tables.

```
WHILE X<>0 DO
  B := ( ((cmbnk,X).3) V 176000 ) * 2
  IF     A<0 THEN
         0 =: B.0
  ELSEIF A>0 THEN
         R3 := B.0
         IF R3<>0 THEN
            R3 =: (cmbnk,X).2
         ENDIF
  ELSE
         R3 := B.0
         IF R3<>0 THEN
            R3 =: (cmbnk,X).2
            0 =: B.0
         ENDIF
  ENDIF
  X := (cmbnk,X).0
  IF interrupt_pending THEN
         P := P-1
         EXIT
  ENDIF
ENDDO
EXIT
```

ENPT opcode 140506_8
Description: Enter segment in page tables.

```
WHILE X<>0 DO
      A  := ( (cmbnk,X).2 ) Λ 173777
      R3 := X/4
      B  := ( ( (cmbnk,X).3 ) V 176000 ) * 2
      A  =: B.0
      R3 =: B.1
      X  := (cmbnk,X).0
      IF interrupt_pending THEN
            P := P-1
            EXIT
      ENDIF
ENDDO
EXIT
```

INSPL opcode 140502_8
Description:

Insert page in page list.

```
R1 := (stbnk,B).7
X  =: (stbnk,B).7
R1 =: (cmbnk,X).0
 IF R1<>0 THEN
    Q := (cmbnk,R1).1
    X =: (cmbnk,R1).1
 ELSE
    Q := ( (B - stsrt) / 2 ) + 3
 ENDIF
Q := (cmbnk,X).1
T =: (cmbnk,X).3
```

REMPL opcode 140503_8
Description:

Remove page from page list.

```
R1 := (cmbnk,X).0
R2 := (cmbnk,X).1
 IF R2∧3 = 0 THEN
    R1 =: (cmbnk,R2).0
 ELSE
    Q := (R2 * 2) + strst
    R1 =: (stbnk,Q).7
 ENDIF
 IF R1<>0 THEN
    R2 =: (cmbnk,R1).1
 ENDIF
0 =: (cmbnk,X).0
0 =: (cmbnk,X).1
```

CNREK opcode 140504_8
Description:

Clear non re-entrant pages.

```
Q := (stbnk,A).2
 IF A=0 THEN
    EXIT
 ENDIF
 R1 := ( (Q∧1700) * 2) + 174000
 DO FOR R2=X TO X+10
    R4 := (T,R2).0
    IF R2 = X+10 THEN
       EXIT
    ENDIF
    DO  FOR lc=0 TO 17
        IF bit(lc,R4) = 1 THEN
           0 =: (R1).0
        ENDIF
        R1 := R1 + 2
    ENDDO
ENDDO
```

CLPT opcode 140505_8
Description:

Clear segment from page tables.

```
WHILE X<>0 DO
  B := ( ((cmbnk,X).3) V 176000 ) * 2
  IF     A<0 THEN
         0 =: B.0
  ELSEIF A>0 THEN
         R3 := B.0
         IF R3<>0 THEN
            R3 =: (cmbnk,X).2
         ENDIF
  ELSE
         R3 := B.0
         IF R3<>0 THEN
            R3 =: (cmbnk,X).2
            0 =: B.0
         ENDIF
  ENDIF
  X := (cmbnk,X).0
  IF interrupt_pending THEN
         P := P-1
         EXIT
  ENDIF
ENDDO
EXIT
```

ENPT opcode 140506_8
Description:

Enter segment in page tables.

```
WHILE X<>0 DO
      A  := ( (cmbnk,X).2 ) Λ 173777
      R3 := X/4
      B  := ( ( (cmbnk,X).3 ) V 176000 ) * 2
      A  =: B.0
      R3 =: B.1
      X  := (cmbnk,X).0
      IF interrupt_pending THEN
            P := P-1
            EXIT
      ENDIF
ENDDO
EXIT
```

REPT opcode 140507_8
Description:

Enter re-entrant segment in page tables.

```
WHILE X<>0 DO
      A  := ( (cmbnk,X).2 ) ∧ 073777
      R3 := X/4
      B  := ( ( (cmbnk,X).3 ) V 176000 ) * 2
      A  =: B.0
      R3 =: B.1
      X  := (cmbnk,X).0
      IF interrupt_pending THEN
            P := P-1
            EXIT
      ENDIF
ENDDO
EXIT
```

LBIT opcode 140510_8
Description:

Load single bit accumulator(K) with logical memory bit.

(X) points to the start of a bit array
(A) points to the bit within the array

SBITP opcode 140513_8
Description:

Store the single bit accumulator (K) in a physical memory bit.

(T) points to the bank number containing the bit array
(X) points to the start of a bit array
(A) points to the bit within the array

LBYTP opcode 140514_8
Description:

Load the A register with a byte from physical memory.

(D) points to the bank number containing the byte array
(T) points to the start of a byte array
(X) points to the actual byte within the array

SBYTP opcode 140515_8
Description:

Store a byte in physical memory.

(D) points to the bank number containing the byte array
(T) points to the start of a byte array
(X) points to the actual byte within the array

TSETP opcode 140516_8
Description:

Test and set physical memory word.

(T) points to the physical memory bank to be accessed
(X) points to the address within the bank
(A) is loaded with the word

The contents of the location addressed by T and X are simultaneously loaded into the A register as the location is written to with all 1s. No other memory access is allowed during this operation.
The old content of the memory address is always read from the memory and never from cache. The all 1s' data word is never written to cache.
This instruction can be used for processor synchronization.

RDUSP opcode 140517_8
Description:

Read a physical memory word without using cache.

(T) points to the physical memory bank to be accessed
(X) points to the address within the bank
(A) is loaded with the memory word

The old content of the memory address is always read from the memory and never from cache.

Note:The execution time of this instruction includes two bus-read cycles (The CPU uses semaphore cycles - see page 107)

LASB opcode $1407\Delta0_8$
Description:

Load the A register with the contents of the segment-table bank (STBNK).

(A) ⟵ (ea)

ea = (B) + Δ = STBNK entry

Δ 3-bit displacement added to B included in the instruction opcode.

SASB opcode $1407\Delta1_8$
Description:

Store the A register contents in the segment table bank (STBNK).

(ea) ⟵ (A)

ea = (B) + Δ = STBNK entry

Δ 3-bit displacement added to B included in the instruction opcode.

LACB opcode $1407\Delta2_8$
Description:

Load the A register from the core map-table bank (CMBNK).

(A) ⟵ (ea)

ea = (B) + Δ = CMBNK entry

Δ 3-bit displacement added to B included in the instruction opcode

SACB opcode $1407\Delta3_8$
Description:

Store the A register in the core map table bank (CMBNK).

(ea) ⟵ (A)

ea = (B) + Δ = CMBNK entry

Δ 3-bit displacement added to B included in the instruction opcode.

LXSB opcode $1407\Delta4_8$
Description:

Load the X register from the segment table bank (STBNK).

(X) ⟵ (ea)

ea = (B) + Δ = STBNK entry

Δ 3-bit displacement added to B included in the instruction opcode.

LXCB opcode $1407\Delta5_8$
Description:

Load the X register from the core table bank (STBNK).

(X) ⟵ (ea)

ea = (B) + Δ = CMBNK entry

Δ 3-bit displacement added to B included in the instruction opcode.

SZSB opcode $1407\Delta6_8$
Description:

Store zero in the segment-table bank (STBNK).

(ea) ⟵ 0

ea = (B) + Δ = STBNK entry

Δ 3-bit displacement added to B included in the instruction opcode.

SZCB opcode $1407\Delta7_8$
Description:

Store zero in the core map-table bank (CMBNK).

(ea) ⟵ 0

ea = (B) + Δ = CMBNK entry

Δ 3-bit displacement added to B included in the instruction opcode.

8.2 Microprogram Changes

The internal architecture in the ND-110/CX differs somewhat from the ND-100. The microprogram has been modified in places to reflect these differences.

Appendix F shows the microinstruction word format in detail. The changes can be summarized as follows:

- additions in the COMM field
- additions in the IDBS field
- the microprogram branch address is one bit wider (bit 20 functions as the extra bit).
- vectored branch field simplified
- clock timing field changed

APPENDIX A ND-110 MNEMONICS

APPENDIX A ND-110 MNEMONICS

A.1 ND-110 MNEMONICS IN ALPHABETIC ORDER

AAA	: 172400	AAB	: 172000	AAT	: 173000
AAX	: 173400	AD1	: 000400	ADC	: 001000
ADD	: 060000	ADDD	: 140120	ALD	: 000012
AND	: 070000	,B	: 000400	BAC	: 000600
BANC	: 177000	BAND	: 177200	BCM	: 000400
BLDA	: 176600	BLDC	: 176400	BORA	: 177600
BORC	: 177400	BSET	: 174000	BSKP	: 175000
BSTA	: 176200	BSTC	: 176000	CCLR	: 000010
CHREENT-PAGES			: 140303	CILP	: 000013
CLD	: 000100	CLEPT	: 140301	CLEPU	: 140304
CLNREENT	: 140302	CLPT	: 140505	CM1	: 000200
CM2	: 000600	CNREK	: 140504	COMD	: 140122
COPY	: 146100	CS	: 000017	CSR	: 000010
DA	: 000005	DB	: 000003	DD	: 000001
DEPO	: 150417	DL	: 000004	DNZ	: 152000
DP	: 000002	DT	: 000006	DX	: 000007
ECCR	: 000015	ELEAV	: 140137	ENPT	: 140506
ENTR	: 140135	EQL	: 000000	EXAM	: 150416
EXIT	: 146142	EXR	: 140600	FAD	: 100000
FDV	: 114000	FMU	: 110000	FSB	: 104000
GEQ	: 000400	GRE	: 001000	I	: 001000
IDENT	: 143600	IF	: 000000	IIC	: 000005
IIE	: 000005	INIT	: 140134	INSPL	: 140502
IOF	: 150401	ION	: 150402	IOX	: 164000
IOXT	: 150415	IRR	: 153600	IRW	: 153400
JAF	: 131400	JAN	: 130400	JAP	: 130000
JAZ	: 131000	JMP	: 124000	JNC	: 132400
JPC	: 132000	JPL	: 134000	JXN	: 133400
JXZ	: 133000	LACB	: 1407Δ2	LASB	: 1407Δ0
LBIT	: 140510	LBITP	: 140511	LBYT	: 142200
LBYTP	: 140514	LCIL	: 000011	LDA	: 044000
LDATX	: 143300	LDBTX	: 143303	LDD	: 024000
LDDTX	: 143302	LDF	: 034000	LDT	: 050000
LDX	: 054000	LDXTX	: 143301	LEAVE	: 140136
LIN	: 003000	LMP	: 000002	LRB	: 152600
LSS	: 002400	LST	: 003000	LWCS	: 143500
LXCB	: 1407Δ5	LXSB	: 1407Δ4	MCL	: 150200
MGRE	: 001400	MIN	: 040000	MIX3	: 143200
MLST	: 003400	MON	: 153000	MOVEW	: 143100
MPY	: 120000	MST	: 150300	NLZ	: 151400
ONE	: 000200	OPCOM	: 150400	OPR	: 000002
ORA	: 074000	PACK	: 140124	PCR	: 000003
PEA	: 000015	PES	: 000013	PGC	: 000014
PGS	: 000003	PID	: 000006	PIE	: 000007
PIOF	: 150405	PION	: 150412	PL10	: 000004
PL11	: 000011	PL12	: 000022	PL13	: 000043

POF	: 150404	PON	: 150410	PVL	: 000004
RADD	: 146000	RAND	: 144400	RCLR	: 146100
RDCR	: 146200	RDIV	: 141600	RDUS	: 140127
RDUSP	: 140517	REMPL	: 140503	REPT	: 140507
REX	: 150407	RGLOB	: 140501	REXO	: 145000
RINC	: 146400	RMPY	: 141200	RORA	: 145400
ROT	: 001000	RSUB	: 146600	SA	: 000050
SAA	: 170400	SAB	: 170000	SACB	: 1407Δ3
SAD	: 154600	SASB	: 1407Δ1	SAT	: 171000
SAX	: 171400	SB	: 000030	SBIT	: 140512
SBITP	: 140513	SBYT	: 142600	SBYTP	: 140515
SD	: 000010	SETPT	: 140300	SEX	: 150406
SHA	: 154400	SHD	: 154200	SHDE	: 140126
SHR	: 000200	SHT	: 154000	SKP	: 140000
SL	: 000040	SP	: 000020	SRB	: 152402
SSC	: 000060	SSK	: 000020	SSM	: 000070
SSO	: 000050	SSQ	: 000040	SSTG	: 000010
SSZ	: 000030	ST	: 000060	STA	: 004000
STATX	: 143304	STD	: 020000	STDTX	: 143306
STF	: 030000	STS	: 000001	STT	: 010000
STX	: 014000	STZ	: 000000	STZTX	: 143305
SUB	: 064000	SUBD	: 140121	SWAP	: 144000
SX	: 000070	SZCB	: 1407Δ7	SZSB	: 1407Δ6
TRA	: 150000	TRR	: 150100	TSET	: 140123
TSETP	: 140516	UCIL	: 000012	UEQ	: 002000
UPACK	: 140125	VERSN	: 140133	WAIT	: 151000
WGLOB	: 140500	,X	: 002000	ZIN	: 002000
ZRO	: 000000				

A.2 ND-110 MNEMONICS IN NUMERICAL ORDER

STZ	: 000000	IF	: 000000	EQL	: 000000
ZRO	: 000000	STS	: 000001	DD	: 000001
OPR	: 000002	DP	: 000002	LMP	: 000002
PGS	: 000003	DB	: 000003	PCR	: 000003
DL	: 000004	PL10	: 000004	PVL	: 000004
DA	: 000005	IIE	: 000005	IIC	: 000005
DT	: 000006	PID	: 000006	PIE	: 000007
DX	: 000007	CSR	: 000010	CCLR	: 000010
SSTG	: 000010	SD	: 000010	LCIL	: 000011
PL11	: 000011	UCIL	: 000012	ALD	: 000012
CILP	: 000013	PES	: 000013	PGC	: 000014
PEA	: 000015	ECCR	: 000015	CS	: 000017
SP	: 000020	SSK	: 000020	PL12	: 000022
SSZ	: 000030	SB	: 000030	SL	: 000040
SSQ	: 000040	PL13	: 000043	SA	: 000050
SSO	: 000050	SSC	: 000060	ST	: 000060
SSM	: 000070	SX	: 000070	CLD	: 000100
CM1	: 000200	SHR	: 000200	ONE	: 000200
,B	: 000400	GEQ	: 000400	BCM	: 000400
AD1	: 000400	CM2	: 000600	BAC	: 000600
ROT	: 001000	ADC	: 001000	GRE	: 001000
I	: 001000	MGRE	: 001400	,X	: 002000
UEQ	: 002000	ZIN	: 002000	LSS	: 002400
LIN	: 003000	LST	: 003000	MLST	: 003400
STA	: 004000	STT	: 010000	STX	: 014000
STD	: 020000	LDD	: 024000	STF	: 030000
LDF	: 034000	MIN	: 040000	LDA	: 044000
LDT	: 050000	LDX	: 054000	ADD	: 060000
SUB	: 064000	AND	: 070000	ORA	: 074000
FAD	: 100000	FSB	: 104000	FMU	: 110000
FDV	: 114000	MPY	: 120000	JMP	: 124000
JAP	: 130000	JAN	: 130400	JAZ	: 131000
JAF	: 131400	JPC	: 132000	JNC	: 132400
JXZ	: 133000	JXN	: 133400	JPL	: 134000
SKP	: 140000	ADDD	: 140120	SUBD	: 140121
COMD	: 140122	TSET	: 140123	PACK	: 140124
UPACK	: 140125	SHDE	: 140126	RDUS	: 140127
VERSN	: 140133	INIT	: 140134	ENTR	: 140135
LEAVE	: 140136	ELEAV	: 140137	SETPT	: 140300
CLEPT	: 140301	CLNREENT	: 140302		
CHREENT-PAGES			: 140303	CLEPU	: 140304
WGLOB	: 140500	RGLOB	: 140501	INSPL	: 140502
REMPL	: 140503	CNREK	: 140504	CLPT	: 140505
ENPT	: 140506	REPT	: 140507	LBIT	: 140510
LBITP	: 140511	SBIT	: 140512	SBITP	: 140513
LBYTP	: 140514	SBYTP	: 140515	TSETP	: 140516
RDUSP	: 140517	EXR	: 140600	LASB	: 1407Δ0
SASB	: 1407Δ1	LACB	: 1407Δ2	SACB	: 1407Δ3
LXSB	: 1407Δ4	LXCB	: 1407Δ5	SZSB	: 1407Δ6
SZCB	: 1407Δ7	RMPY	: 141200	RDIV	: 141600
LBYT	: 142200	SBYT	: 142600	MOVEW	: 143100
MIX3	: 143200	LDATX	: 143300	LDXTX	: 143301
LDDTX	: 143302	LDBTX	: 143303	STATX	: 143304
STZTX	: 143305	STDTX	: 143306	LWCS	: 143500

IDENT	: 143600	SWAP	: 144000	RAND	: 144400
REXO	: 145000	RORA	: 145400	RADD	: 146000
RCLR	: 146100	COPY	: 146100	EXIT	: 146142
RDCR	: 146200	RINC	: 146400	RSUB	: 146600
TRA	: 150000	TRR	: 150100	MCL	: 150200
MST	: 150300	OPCOM	: 150400	IOF	: 150401
ION	: 150402	POF	: 150404	PIOF	: 150405
SEX	: 150406	REX	: 150407	PON	: 150410
PION	: 150412	IOXT	: 150415	EXAM	: 150416
DEPO	: 150417	WAIT	: 151000	NLZ	: 151400
DNZ	: 152000	SRB	: 152402	LRB	: 152600
MON	: 153000	IRW	: 153400	IRR	: 153600
SHT	: 154000	SHD	: 154200	SHA	: 154400
SAD	: 154600	IOX	: 164000	SAB	: 170000
SAA	: 170400	SAT	: 171000	SAX	: 171400
AAB	: 172000	AAA	: 172400	AAT	: 173000
AAX	: 173400	BSET	: 174000	BSKP	: 175000
BSTC	: 176000	BSTA	: 176200	BLDC	: 176400
BLDA	: 176600	BANC	: 177000	BAND	: 177200
BORC	: 177400	BORA	: 177600		

APPENDIX B ND-BUS SIGNALS

B.1 ND-110 CPU C-CONNECTOR

Row a pin	Row a name	Row b pin	Row b name	Row c pin	Row c name
1	Gnd	1	Gnd	1	Gnd
2	+ 5V	2	+ 5V	2	+5V
3	BD 1	3	BD 16	3	BD 0
4	BD 3	4	BD 17	4	BD 2
5	BD 5	5	BD 18	5	BD 4
6	BD 7	6	BD 19	6	BD 6
7	BD 9	7	BD 20	7	BD 8
8	BD 11	8	BD 21	8	BD 10
9	BD 13	9	BD 22	9	BD 12
10	BD 15	10	BD 23	10	BD 14
11	Gnd	11	Gnd	11	Gnd
12	BREF	12	LOAD	12	BREQ
13	PA 1 [1]	13	RESTART	13	PA 0 [1]
14	PA 3 [1]	14	RUN	14	PA 2 [1]
15	BINT 10	15	CONTINUE	15	BINT 11
16	BINT 12	16	STOP	16	BINT 13
17	SEMRQ	17	BLANK	17	BINT 15
18	BINPUT	18	BPAERR	18	BDAP
19	BDRY	19	BINACK	19	BIOXE
20	BAPR	20	BMCL	20	BMEM
21	INCONTR [2]	21	BERROR	21	OUTCONTR [2]
22	INIDENT [2]	22	BCRQ	22	OUTIDENT [2]
23	INGRANT [2]	23	BMINH	23	OUTGRANT [2]
24	Gnd	24	Gnd	24	Gnd
25	+ 15V	25	+ 15V	25	+ 15V
26	Analogue Gnd	26	Analogue Gnd	26	Analogue Gnd
27	- 15V	27	- 15V	27	- 15V
28	+ 12V	28	+ 12V	28	+ 12V
29	Power Sense	29	Power Sense	29	Power Sense
30	+ 5V Standby	30	+ 5V Standby	30	+ 5V Standby
31	+ 5V	31	+ 5V	31	+ 5V
32	Gnd	32	Gnd	32	Gnd

Note 1: Position code

Note 2: These lines are connected as a daisy-chain.

Note 3: All bus signals are active low TTL level.
Low (logical 0) = 2.4 to 5.0 V
High (logical 1) = 0.0 to 0.5 V

Note 4: Refer to ND-100 Bus Description manual, ND-06.017.02, for more details.

Figure 67. ND-100 bus signals

B.2 ND-110 CPU B-CONNECTOR

Row a pin	Row a name	Row b pin	Row b name	Row c pin	Row c name
1	Gnd	1	GND	1	GND
2	+5V	2	+5V	2	+5V
3	LuA4	3	EBUS	3	LuA5
4	LuA6	4	INR6	4	LuA7
5	LuA8	5	INR5	5	LuA10
6	LuA11	6	INR0	6	LuA9
7	LuA3	7	INR2	7	LuA2
8	LuA12	8	INR3	8	LuA0
9	LuA1	9	PIL0	9	OSC
10	PPN24	10	PIL1	10	XCLK
11	COMM2	11	PIL3	11	COMM3
12	LCS	12	PIL2	12	COMM1
13	COMM0	13	INR1	13	COMM4
14	MIS0	14	SEL5MS	14	MIS1
15	IDB7	15	INR7	15	IDB6
16	IDB5	16	PPN25	16	IDB4
17	IDB0	17	INR4	17	IDB1
18	IDB2	18	MOR	18	IDB3
19	IDB12	19	TRAP	19	IDB13
20	IDB9	20	SELPT	20	IDB8
21	IDB10	21	LSHADOW	21	IDB11
22	IDB14	22	-	22	IDB15
23	LA6	23	LA18	23	LA7
24	LA4	24	LA19	24	LA5
25	LA2	25	LA20	25	LA3
26	LA0	26	LA17	26	LA1
27	LA14	27	LA16	27	LA15
28	LA12	28	LA22	28	LA13
29	MLA10	29	LA23	29	MLA11
30	LA8	30	LA21	30	LA9
31	+5V	31	+5V	31	+5V
32	GND	32	GND	32	GND

Note 1: All bus signals are active low TTL level.
Low (logical 0) ← 2.4 to 5.0 V
High (logical 1) ← 0.0 to 0.5 V

Figure 68. ND-110 Tracer signals

B.3 ND-110 CPU A-CONNECTOR

Row a pin	name	Row b pin	name	Row c pin	name
1	GND	1	GND	1	GND
2		2		2	
3		3	OSCCLc	3	
4		4	XTR2c	4	
5		5		5	
6	OC1c	6		6	OC0c
7	I1+c	7		7	TXDc
8	I2-c	8		8	RXDc
9	O1+c	9		9	RTSc
10	O2-c	10		10	
11		11		11	DSRc
12	DTRc	12		12	GND
13		13		13	
14		14		14	
15	SWLDc	15		15	SWLDc
16	GND	16		16	SWMCLc
17	GND	17		17	SWSTPc
18	GND	18		18	RUNc
19	GND	19		19	EAUTOc
20	GND	20		20	LOCKc
21	GND	21		21	CONSOL1c
22	GND	22		22	+5V
23	GND	23		23	XTR1c
24	GND	24		24	+5VSTBY
25	GND	25		25	DP1c
26	GND	26		26	DP2c
27	GND	27		27	DP3c
28	GND	28		28	DP4c
29	GND	29		29	DP5c
30	ON2c	30		30	CONSOL2c
31	+5V	31		31	+12V
32	GND	32	GND	32	GND

Note 1: All bus signals are active low TTL level.
Low (logical 0) ← 2.4 to 5.0 V
High (logical 1) ← 0.0 to 0.5 V

Figure 69. ND-110 I/O connector signals

APPENDIX C SWITCHES AND INDICATORS ON THE ND-110 CPU

C.1 SWITCH SETTINGS ON THE OLD CPU CARD (3090)

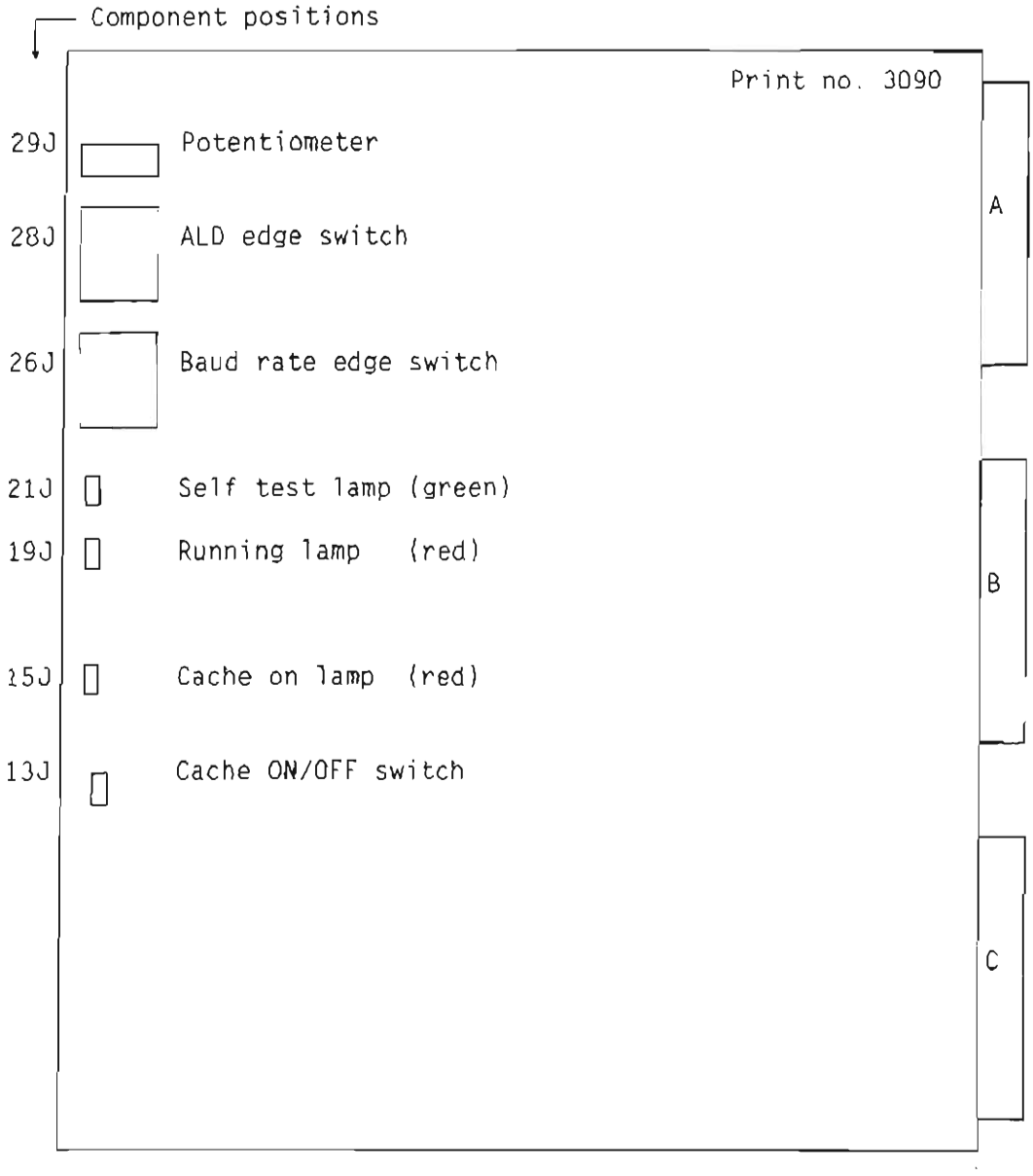


Figure 70. Switch settings ND-110, early version (3090)

C.2 SWITCH SETTINGS ON THE NEW CPU CARD (3095)

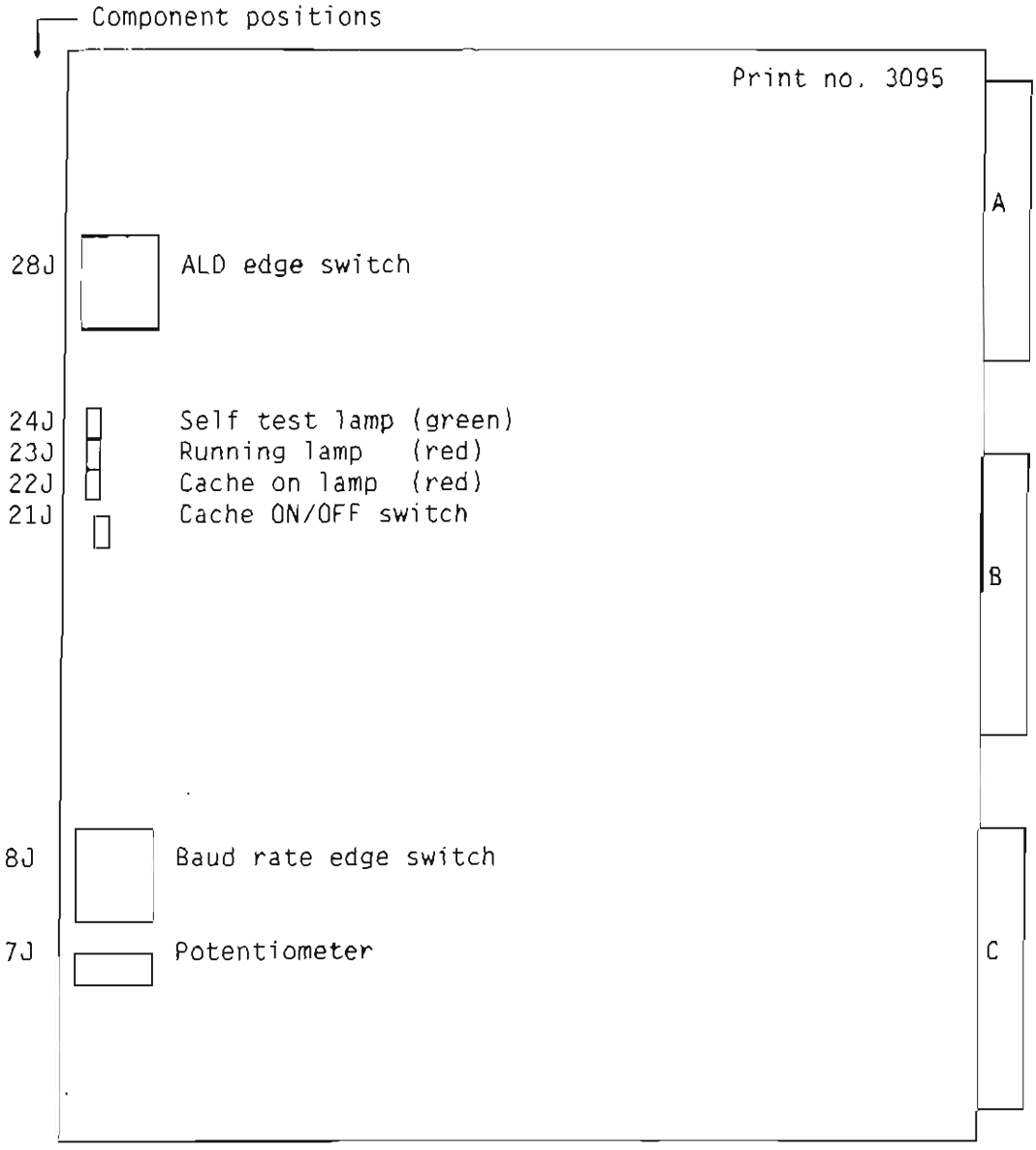


Figure 71. Switch settings ND-110, new version (3095)

C.3 SWITCH SETTINGS ON THE TERMINAL INTERFACE (3013)

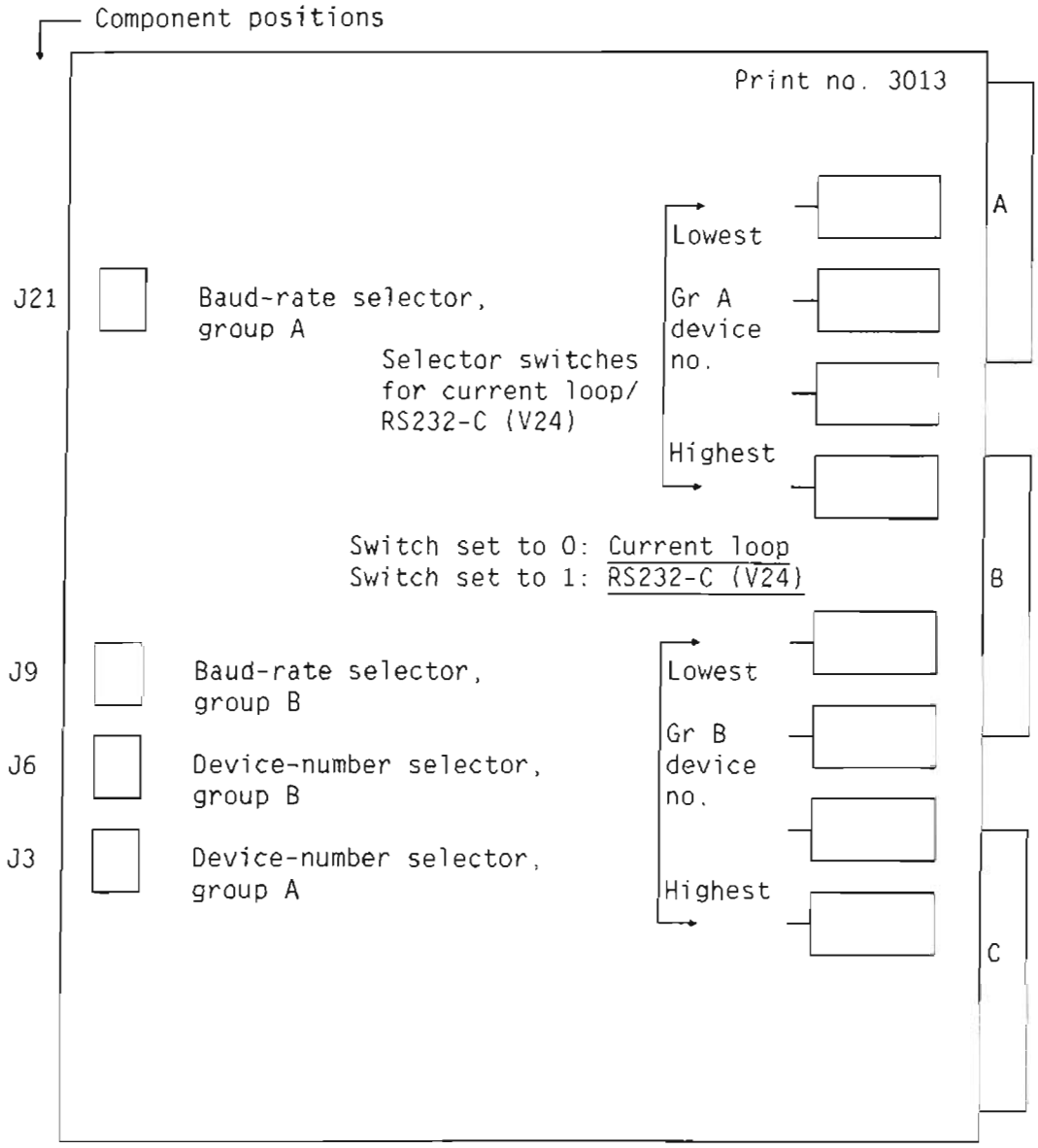


Figure 72. The 8-Terminal Interface (3013)

Baud rate switch settings

Switch setting	Baud rate	Switch setting	Baud rate
0	110	8	2400
1	150	9	600
2	300	10	200
3	2400	11	134.5
4	1200	12	75
5	1800	13	50
6	4800	14	Not used
7	9600	15	Not used

C.4 SWITCH SETTINGS ON THE TERMINAL INTERFACE (3107)

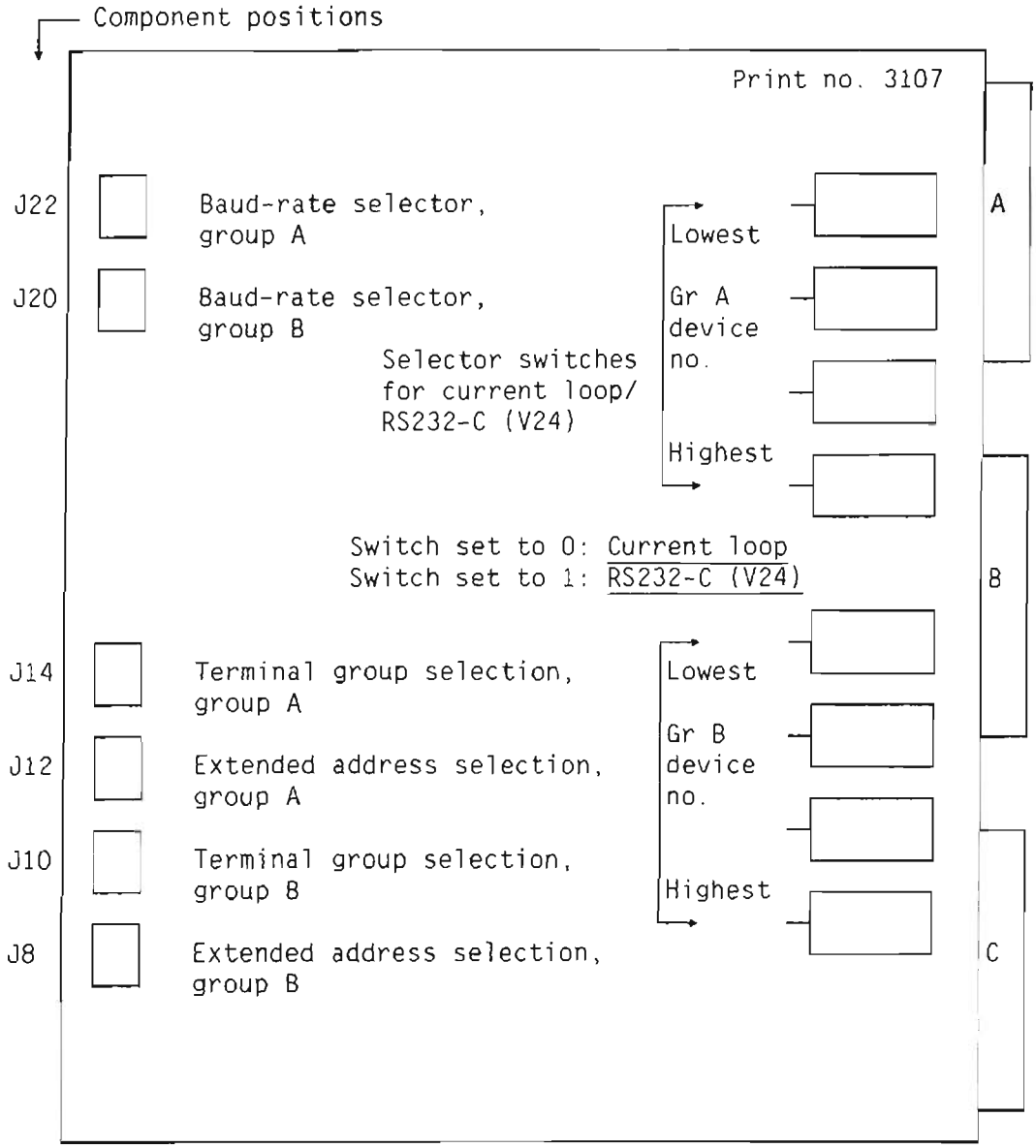


Figure 73. The 8-Terminal Interface (3107)

Baud rate switch settings

Switch setting	Baud rate	Switch setting	Baud rate
0	110	8	2400
1	150	9	600
2	300	10	200
3	2400	11	134.5
4	1200	12	75
5	1800	13	50
6	4800	14	Not used
7	9600	15	Not used

APPENDIX D PRIVILEGED INSTRUCTIONS

APPENDIX D PRIVILEGED INSTRUCTIONS

CHREENT-PAGES	change non reentrant pages	ce	140303
CLEPT	clear page tables	ce	140301
CLEPU	clear page tables,collect PGU info	ce	140304
CLNREENT	clear non reentrant	ce	140302
CLPT	clear segment from page tables	si	140505 *
CNREK	clear non-reentrant pages	si	140504 *
DEPO	memory deposit	px	150417
ENPT	enter segment into page tables	si	140506 *
EXAM	memory examine	px	150416
IDENT	identify interrupt	px	143600
INSPL	insert page in page list	si	140502 *
IOF	turn off interrupting system	px	150401
ION	turn on interrupting system	px	150402
IOX	input/output	px	164000
IOXT	input/output	px	150415
IRR	inter-register read	px	153600
IRW	inter-register write	px	153400
LACB	load A with core map table bank	si	1407Δ2 *
LASB	load A in segment table bank	si	1407Δ0 *
LBIT	load K flip-flop with logical memory bit	si	140510 *
LBITP	load K flip-flop with physical memory bit	si	140511 *
LBYTP	load byte from physical memory	si	140514 *
LDATX	load A with physical memory contents	si	143300
LDBTX	load B with physical memory contents	si	143303
LDDTX	load D with physical memory contents	si	143302
LDXTX	load X with physical memory contents	si	143301
LRB	load register block	px	152600
LWCS	load writeable control store	px	143500
LXCB	load X with core map table bank	si	1407Δ5 *
LXSB	load X with segment table bank	si	1407Δ4 *
MCL	masked clear of register	px	150200
MST	masked set of register	px	150300
OPCOM	set to OPCOM mode	px	150400
PIOF	turn paging and interrupt off	px	150405
PION	turn paging and interrupt on	px	150412
POF	turn memory management off	px	150404
PON	turn memory management on	px	150410
RDUSP	read a word without using cache	si	140517 *
REMPL	remove page from page list	si	140503 *
REPT	enter reentrant segment in page tables	si	140507 *
REX	reset extended address mode	px	150407
RGLOB	examine STBNK,STSRT;CMBNK	si	140501 *
SACB	store A in core map table bank	si	1407Δ3 *
SASB	store A in segment table bank	si	1407Δ1 *
SBIT	store K flip-flop in logical memory bit	si	140512 *
SBITP	store K flip-flop in physical memory bit	si	140513 *
SBYTP	store byte in physical memory	si	140515 *
SETPT	set page tables	ce	140300
SEX	set extended address mode	px	150406

SRB	store register block	px	152402
STATX	store in A physical memory contents	si	143304
STDTX	store in D physical memory contents	si	143306
STZTX	store in Z physical memory contents	si	143305
SZCB	store 0 in core map table bank	si	1407Δ7 *
SZSB	store 0 in segment table bank	si	1407Δ6 *
TRA	transfer internal register to A	px	150000 *
TRR	transfer internal register from B	px	150100 *
TSETP	physical test-and-set request	si	140516 *
WAIT	give up priority	px	151000
WGLOB	initialize global pointers	si	140500 *

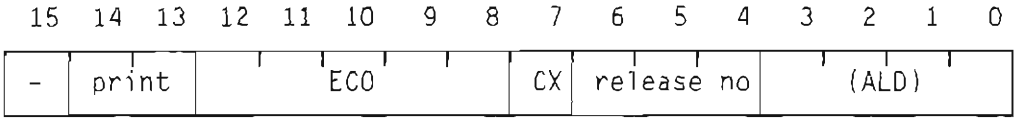
* new ND-110 instructions or instruction usage

APPENDIX E PRINT VERSION

APPENDIX E PRINT VERSION

The CPU card contains a 12-bit jumper area (component position E 35 on print number 3095; J 33 on print number 3090) The VERSN instruction may be used to fetch the settings of jumber field (See page 193 for a description of the VERSN instruction).

The A register has the following format after executing VERSN.



The bit fields have been assigned the following definitions:

E.1 PRINT NUMBER

Only two print numbers have been defined.

Bit 14	Bit 13	Print number
0	0	3090
0	1	3095
1	0	reserved
1	1	reserved

Table 19. Print number jumper settings

E.2 ENGINEERING CHANGE ORDER (ECO)

Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	ECO level
1	1	1	1	0	A
1	1	1	1	1	B
0	0	0	0	0	C
0	0	0	0	1	D
0	0	0	1	0	E
0	0	0	1	1	F
0	0	1	0	0	G

Continued on the next page...

Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	ECO level
0	0	1	0	1	H
0	0	1	1	0	J
0	0	1	1	1	K
0	1	0	0	0	L
0	1	0	0	1	M
0	1	0	1	0	N
0	1	0	1	1	P
0	1	1	0	0	Q
0	1	1	0	1	R
0	1	1	1	0	S
0	1	1	1	1	T
1	0	0	0	0	U
1	0	0	0	1	V
1	0	0	1	0	W
1	0	0	1	1	X
1	0	1	0	0	Y
1	0	1	0	1	Z
1	0	1	1	0	BA
1	0	1	1	1	BB
1	1	0	0	0	BC
1	1	0	0	1	BD
1	1	0	1	0	BE
1	1	0	1	1	BF
1	1	1	0	0	BG
1	1	1	0	1	BH

Table 20. ECO jumper settings

E.3 Speed version (CX)

Bit 7	CPU speed version[1]
0	ND-110/CX (fast)
1	ND-110 Standard

Note 1 : There are a few early CPU cards (print number 3090, release C) Where it is not possible to read this field.

Table 21. Speed version jumper settings

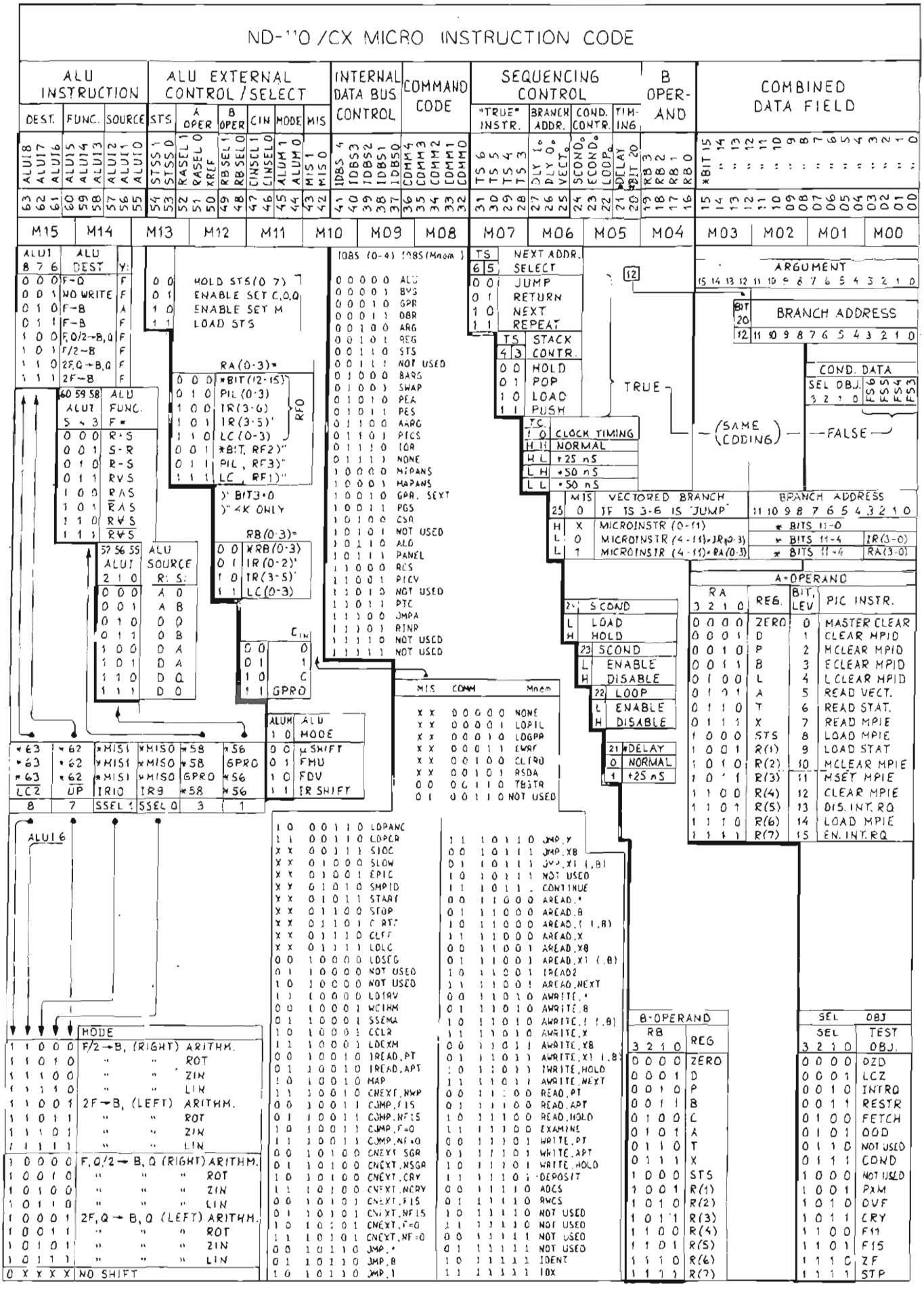
E.4 PRINT RELEASE VERSION

Bit 6	Bit 5	Bit 4	Print 3090 release vsn.	Print 3095 release vsn.
0	0	0	C	B
0	0	1	K	.
0	1	0	.	.
0	1	1	.	.
1	0	0	.	.
1	0	1	.	.
1	1	0	.	.
1	1	1	.	.

Table 22. Print release jumper settings

Only the above release versions have been defined at present.

APPENDIX F MICROCODE FORMAT


ND-110/CX MICRO INSTRUCTION CODE
ALU INSTRUCTION
ALU EXTERNAL CONTROL/SELECT
INTERNAL DATA BUS CONTROL
COMMAND CODE
SEQUENCING CONTROL
B OPERAND
COMBINED DATA FIELD
DEST.
FUNC.
SOURCE
STS
A OPER
B OPER
CIN
MODE
MIS
"TRUE" INSTR.
BRANCH ADDR.
COND. CONTR.
TIMING
M15
M14
M13
M12
M11
M10
M09
M08
M07
M06
M05
M04
M03
M02
M01
M00
ARGUMENT
BRANCH ADDRESS
COND. DATA
TRUE
FALSE
(SAME CODING)
A-OPERAND
B-OPERAND
MODE
NO SHIFT

APPENDIX G GLOSSARY

APPENDIX G GLOSSARY

This appendix contains explanations of some important terms and concepts used in this manual. The explanations given for the terms apply to their use in this manual and in Norsk Data products generally.

Bit The smallest unit of data in a digital computer. A bit may have the value 0 (zero) or 1 (one).

Bit map A register or area of memory that uses individual bits to represent logical values. Commonly used to record the state (used/not used) of areas of memory and memory devices (e.g. disks).

Byte One half of an ND-110 word. A group of eight bits treated as one unit.

CMOS Complementary Metal Oxide Semiconductor. A technology used for manufacturing integrated circuits. CMOS circuits consume less power than conventional (bipolar) circuits.

Commercial extended The extended version of the ND-100 instruction set that includes BCD operations and other extensions of the basic instruction set. This is now standard on all ND-110 computers.

Console The terminal used by the operator to communicate directly with the CPU. The console terminal is the only terminal that can communicate directly with the microprogram. (see also OPCOM).

Data Numbers, letters, symbols and the codes that are used to represent them, regarded as objects to be stored or processed by a computer.

Gate array A type of integrated circuit which is constructed as a regular array of standard circuits (logic gates). The interconnection of these gates is the final step in manufacture. The ND-110 CPU uses three gate arrays; RMIC, RMAC and BUFALU.

Information The interpretation given to data when understood in a specific context.

Thus, depending on the context, the 16-bit value 143304_8 may be understood as:

- The unsigned integer 50884_{10}
- The ASCII string "FD"
- The ND-110 instruction STATX

Instruction A code which can be interpreted by the CPU (or ALU) as a command to perform one or more logical or arithmetical operations. The ND-110 has two levels of instructions: macroinstructions and microinstructions. (see also macroinstruction and microinstruction)

Macroinstruction One of the set of operations which the ND-110 CPU can execute. Sometimes referred to as machine instruction or MAC instruction. (see also microinstruction).

Microprogram The set of very low level instructions that together define the functions of the CPU. (see also microinstruction)

Microinstruction One of the set of primitive operations which are used in the microprogram to define the characteristics of the CPU. Macroinstructions are defined by one or more microinstructions. (see also macroinstruction)

MOPC Acronym from Microprogram OPerator Communication. This is the part of the microprogram that communicates with the console (terminal no. 1) when the CPU is in the OPCOM mode.

ND-100 family The family of 16-bit general-purpose computers from Norsk Data consisting of the following machines:

- ND-100
- ND-100/CE
- ND-100/CX
- ND-110
- ND-110/CX

Operating System A program, or set of programs which provide the basic operating functions of a computer. The operating system used on Norsk Data computers is called SINTRAN.

OPCOM Acronym from OPerator COMmunication mode. In this mode the ND-110 CPU console (terminal no. 1) communicates directly with the microprogram. This mode is used for service and maintenence only.

Page A contiguous area of memory consisting of 1024 words (2K bytes). By extension, a page may also refer to 1024 words on a disk, tape or other storage medium.

Page index table A page table as seen from software. Table containing the physical page numbers and access information of pages of memory. Page index tables (PITs) are managed by the operating system (SINTRAN).

Page table An area of high speed memory, within the memory management part of the CPU, which contains the information needed for the memory management system to convert virtual (16 bit) address to physical (24 bit) address.

SINTRAN The operating system for all Norsk Data computers. The versions currently used are:

- SINTRAN III VSX For ND-110 computers
- SINTRAN III VSX-500 For ND-500 computers

Word The fundamental data unit of a computer. The ND-110 uses a word of sixteen bits.

Index

The information in this manual is subject to change without notice. Norsk Data A.S assumes no responsibility for any errors that may appear in this manual. Norsk Data A.S assumes no responsibility for the use or reliability of its software on equipment that is not furnished or supported by Norsk Data A.S. Copyright © 1987 by Norsk Data A.S.

UPDATING

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

New versions and revisions are announced in the ND Customer Support Information and can be ordered from the address below.

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

PRINTING RECORD	
PRINTING	NOTES
03/87	Version 1 EN

ND-110 Functional Description
Publ.No. ND-06.026.1 EN

RING BINDER OR PLASTIC COVER

The manual can be placed in a ring binder for greater protection and convenience of use. Ring binders may be ordered at a price of NKr. 45.- per binder.

The manual may also be placed in a plastic cover. This cover is more suitable for manuals of less than 100 pages than for larger manuals.

Please send your order, as well as all types of inquiries and requests for documentation to the local ND office, or (in Norway) to:

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

I would like to order

..... Ring Binders, B5, at NOK 35,- per binder

..... Ring Binders, A4, at NOK 45.- per binder

..... Plastic Covers, A4, at NOK 10.- per cover

Name: .....

Company: .....

Address: .....

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

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

Please let us know if you

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

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

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

Manual name: ND-110 Functional Description Manual number: ND-06.026.1 EN

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

_____

_____

_____

_____

Do you have suggestions for improving this manual ? _____

_____

_____

_____

_____

_____

Your name: _____ Date: _____

Company: _____ Position: _____

Address: _____

_____

What are you using this manual for ? _____

_____

NOTE!

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

Send to:

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

Norsk Data's answer will be found on reverse side

Answer from Norsk Data _____

Answered by _____ Date _____

Norsk Data A.S
Documentation Department
P.O. Box 25, Bogerud
0621 Oslo6, Norway