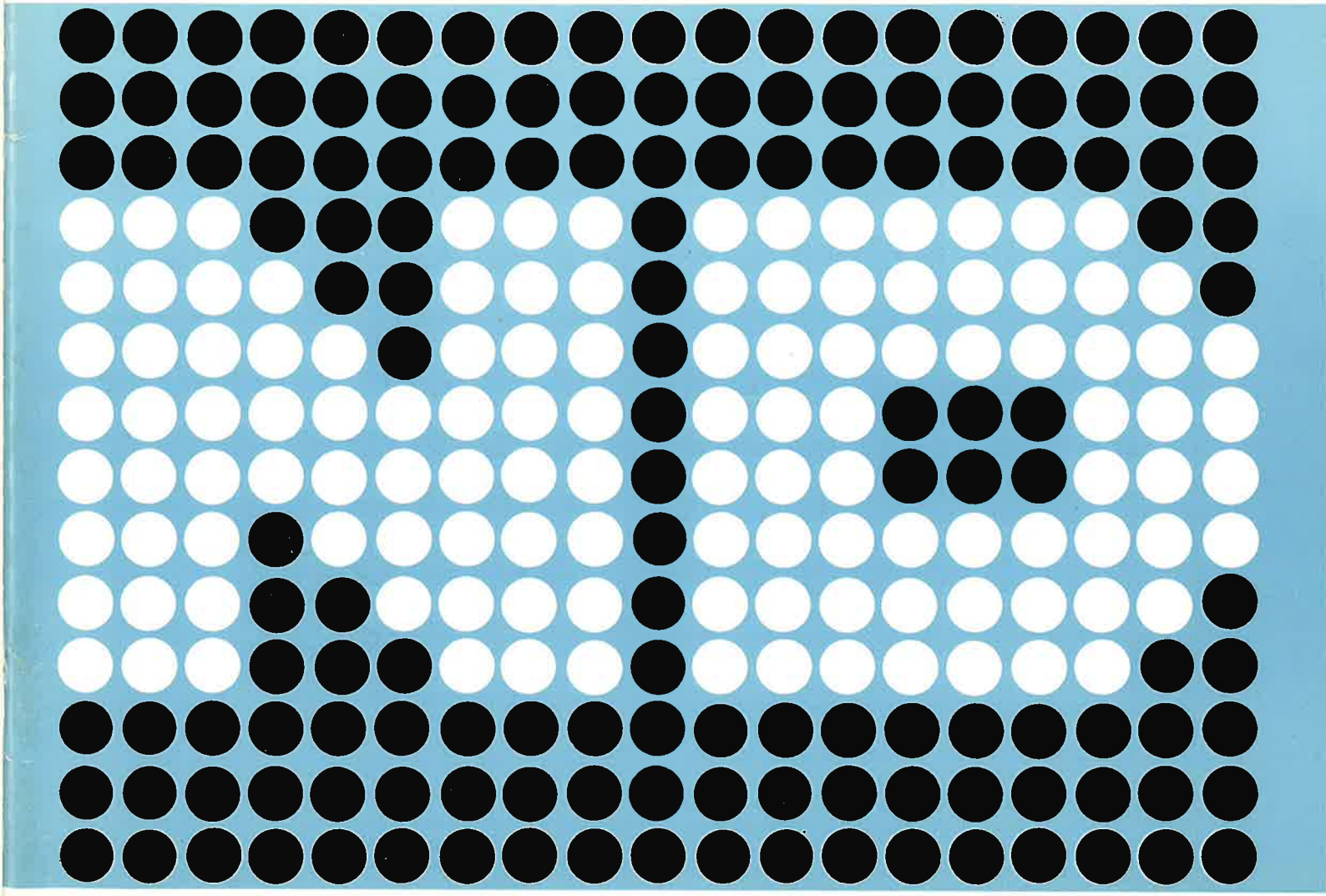
NO. 8

JUNE 1974

ND NEWS



A/S NORSK DATA-ELEKTRONIKK
ØKERNVEIEN 145 OSLO 5 NORWAY PHONE: 21 73 71

CONTENTS

ND NEWS

JUNE 1974 - No. 8

Published by
A/S Norsk Data-Elektronikk

Økernveien 145, Oslo 5, Norway
Telephone: (472) 21 73 71
Telegrams: Norskdata, Oslo
Telex: 18284 ND/N

Editor:
Jan Aske Børresen

Assistant to the Editor:
Inger Meidell-Haugland

The Multi-Computer Control System for the New CERN Synchrotron

BY GEORGE SHERING, CERN, LABORATOIRE II

This is a reprint of a paper which was presented at Nordata, Copenhagen, August 16th 1973 by Mr. G. Shering, who is a member of the Control Group of the SPS.

The European Organization for Nuclear Research (CERN)

CERN was founded in 1954 so that European physicists could have access to experimental equipment as good as, or better than, any existing elsewhere in the world. Twelve European countries contribute to CERN, namely Austria, Belgium, Denmark, Federal Republic of Germany, France, Greece, Italy, Netherlands, Norway, Sweden, Switzerland, and the United Kingdom.

From 1959 the 28 GeV proton synchrotron at CERN, followed by the Intersecting Storage Rings (ISR) in 1971, have provided excellent research facilities. In 1971, however, the member states authorized the construction of a «Super Proton Synchrotron» (SPS) so that the physicists in European universities would have continued access to top class research equipment right into the 1980's. This new accelerator will cost over a billion Swiss francs, it will deliver its first beams of protons in 1976, and will be completed by 1978.

The SPS accelerator consists essentially of a 2.2 km diameter ring around which protons are guided by some 1000 magnets whilst being accelerated by radio frequency electric fields. The accelerator hardware is situated underground in a tunnel bored by a special tunnelling machine. This minimizes environmental damage and will contain the radiation produced when the machine is running. Figure 1 shows a map of the site. At the bottom is the original CERN site containing the 28 GeV PS (200 m diameter) and the ISR (300 m diameter). This site is situated in Switzerland near Geneva. The perimeter of the SPS rings is 1/3 in Switzerland and 2/3 in France, the SPS laboratory buildings at the top of the figure being in France.

Protons will be injected from the PS at 10 GeV, accelerated to 400 GeV, then ejected for use in physics experiments. In 1976 they will be ejected along beam line TT60 to the existing experimental hall B1 at the bottom left of figure 1. In 1978 this will be followed by ejection along beam line TT20 to the North experimental area. The acceleration process is cyclic, with a repetition time from 4 to 8 seconds.

Although most of the accelerator hardware is contained in the subterranean tunnel the power supplies, control equipment and computers are housed in

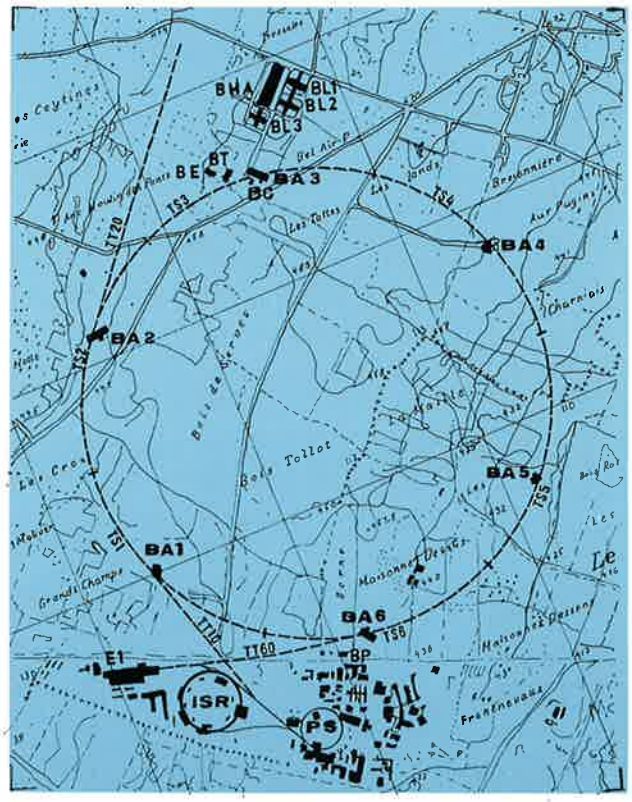


Figure 1. Map of CERN site with the SPS ring.

separate buildings, BA1 to BA6 distributed around the ring. The central control facilities are housed in a special building, B6, next to BA3.

The computer control system for the SPS will have to handle about 28 000 status indications, 14 000 binary commands, 1 400 analogue measurements and 1 000 analogue controls. Originally a single large central computer was envisaged, connected to the accelerator via small satellite computers. A schematic diagram of this system, as proposed in the pre -1971 design studies, is shown in figure 2. Further study, however, showed that this traditional hierarchical multi-computer system has some serious disadvantages for the SPS application. These are mainly due to the one-off nature of the project, as the development and commissioning phases up to 1978

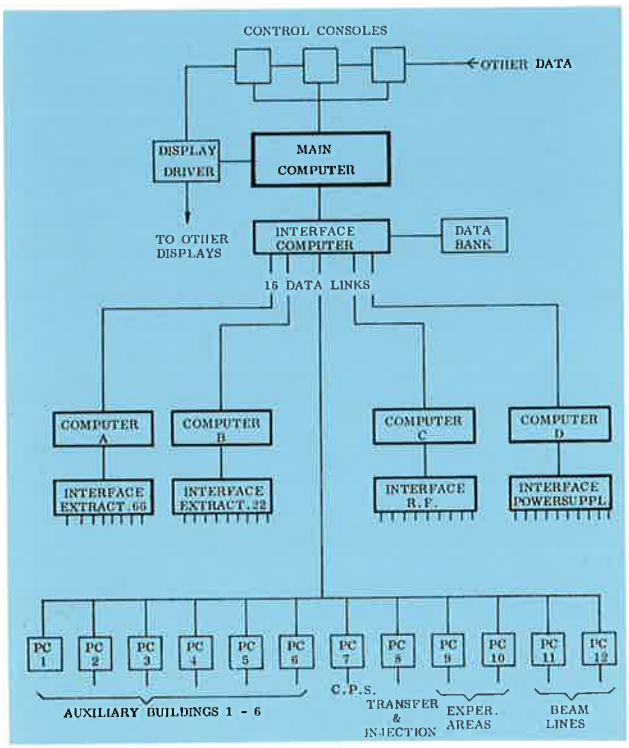


Figure 2. Schematic diagram of the system, as proposed in the pre-1971 design studies.

will put an even heavier load on the computer control system than will routine operation of the synchrotron later. A large central computer with very small satellites would be of little use until the whole system was working. Also the problems of reliability in the central computer become very serious.

For these reasons the alternative scheme shown in figure 3 was adapted. This scheme also proved to be cheaper. Each computer is identical being a NORSK-DATA NORD-10. The only vital over-all link in this scheme is the Message Switching System which links all 24 computers via 30 000 words/sec serial data-links. Even so each local computer is sufficiently powerful so that in any event full local control can be maintained without the use of extensive local hard-wired control panels.

The 16 computers outside the ring of figure 3 are situated close to the equipment being controlled, mainly in the six auxiliary buildings BA1 to BA6 distributed round the ring. Most of these computers have drums for local storage of programs and data. The six general purpose computers GP1 to GP6 handle the systems such as the vacuum and closed orbit correction which are distributed right round the ring, so these computers will be very similar. In some buildings there are concentrations of equipment, such as that for injection, extraction, main magnet power-supply, and RF. A dedicated computer is installed for control of these large sub-systems. These computers will be installed starting in November 1973 and used for hardware testing, commissioning and software development, long before the whole system is integrated, in 1976, for the first overall accelerator trials.

The central computers, those shown inside the ring of figure 3, are installed in the main control building. Overall accelerator control will be from three main consoles, each with its own computer, and one subsidiary console for personnel and administrative functions. Extensive use will be made of displays, touch-panels, and other means of human-computer interaction[1]. A prototype console and computer system is under construction.

A separate computer is shown for alarm analysis and alarms will be routed to this one computer, which will have to analyze them, decide which are the most significant and which are consequent, and only present the most significant to the operator. The display of this information will probably be on a special screen on each console, but it is intended that all other displays, including multi-colour and graphics, should be produced by a system using a standard T.V. raster. A separate computer will control this system, which has the advantage that any number of cheap T.V. monitor units can be added to it, to give displays whenever wanted.

One of the most important computers at the centre is the Library. This will have both fixed head drums and moving head drums. It will keep a library of programs and data for the console computers, and maintain a data-bank of all variables and relevant parameters. This will provide the information to set up the accelerator after a stop.

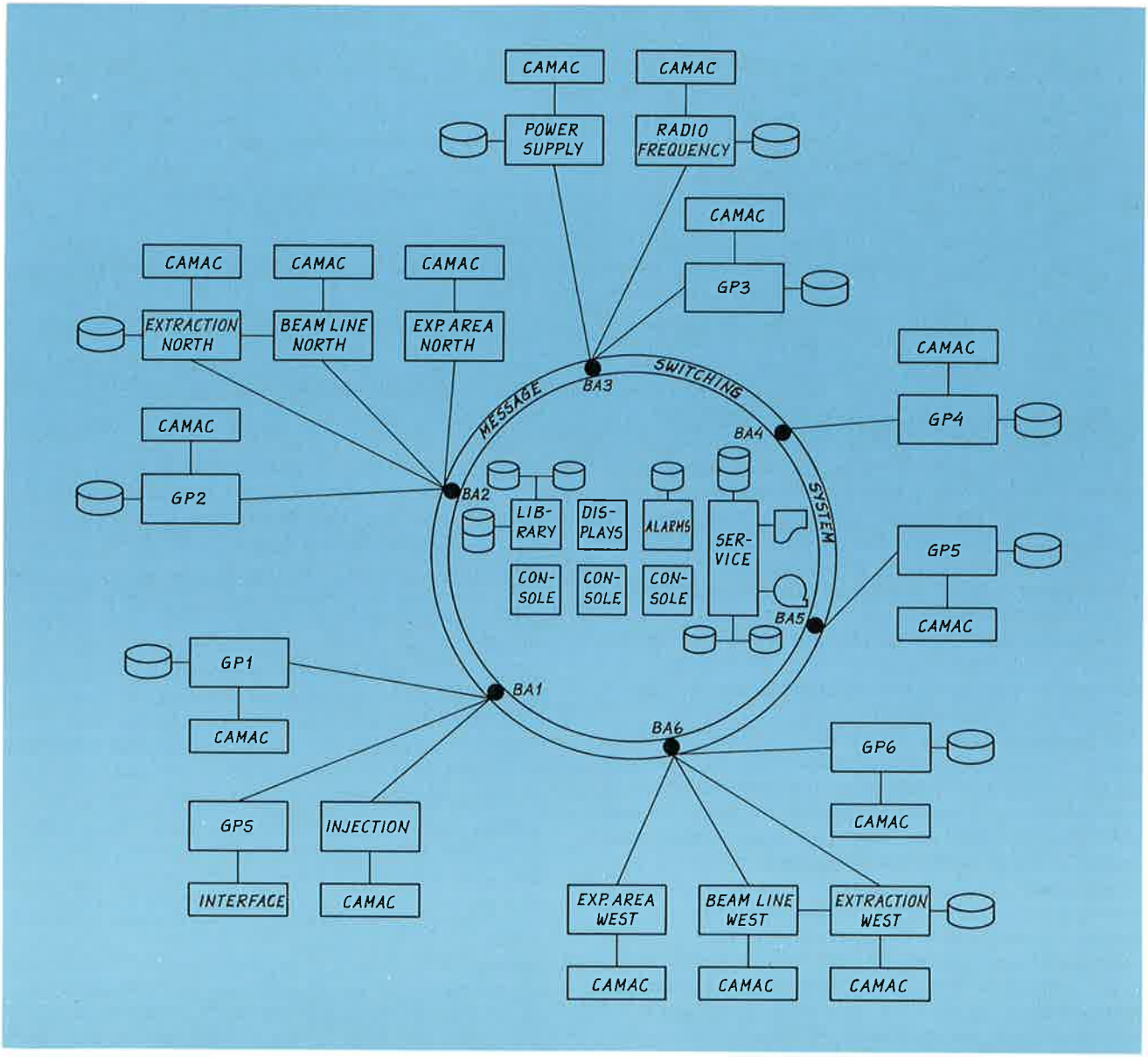
The largest computer in the whole system has been called the Service computer. It has 64K core and a wide range of peripherals, including magnetic tape,

[1] Talk given by F. Bech to NOCUS, October 1973.

line printer, cassette tape, card reader, etc. It will be used for program development and background process operations. In addition the Service computer will be able to stand-in for the Library or alarm computers, should these fail, merely by the interchange of a disk pack or two.

The interface with the accelerator equipment will be via CAMAC, as seen from figure 3. Although a special

Figure 3. Organization of the computers.



interface system designed only for the SPS would differ from CAMAC in several respects, the design effort would be considerable and CAMAC has the advantage of being internationally recognized and a great range of modules are already available. Another big advantage was that work on the interface system was started before even the choice of the NORD-10 was made. NORSK-DATA are supplying a crate controller for the NORD-10 which has been designed in conjunction with CERN[2]. This crate-controller approach was thought preferable to the branch-highway approach as only a few crates will be required for each computer, and the interrupt response can be better.

The CAMAC interface will be used directly for all special purpose work, but would be too expensive for all the many thousands of simple controls necessary. So a special sub-multiplexing system has been developed which is driven by CAMAC.

The Software for this NORD-10 system rests on two main pillars: A special high-level language interpreter, NODAL; and the real-time operating system SINTRAN II. SINTRAN II is supplied by NORSK-DATA and provides management of interrupts and I/O devices, together with facilities for scheduling programs. The fact that it exists in both core-only and mass-memory forms was a big suitability for this application. NODAL is a special CERN development created with the help of Bo Lewendal of NORSK-DATA under sub-contract. It will, however, be available to all NORD-10 users. Its main purpose is to provide the easy interactive programming so necessary in a one-off research application, and to ease the inter-computer communication which is vital to this multi-computer project.

The Software Layout in an individual NORD-10 is shown in figure 4. Commands or program material enters on the left, either from the local terminal or the data-links. Most computers have both Infoton VDU's as well as Teletypes. The input is then processed through the NODAL interpreter, into local machine code routines, then out to the accelerator through CAMAC. NODAL is re-entrant so it can handle several levels of real-time programs or commands under the control of SINTRAN. Typically three levels of NODAL will be provided: level 1 for on-line interactive use or for long term programs; level 5 for programs from the data-links, or locally stored programs, which will execute to completion by interrupting the level 1 program; level 9 for very short high-priority commands down the data-links.

Conversational on-line programming is the first feature of an interpreter such as NODAL. Since there are no compile and load steps a program can be written into the computer and executed immediately. NODAL stores the program in source form and has editing and test facilites that enables the program to be checked and modified on the spot. The line and group structure of NODAL enables parts of programs to be executed separately, then the values of particular variables printed out. Even data from the process can be interrogated directly from the keyboard. There is also a trace made which displays each line before executing it.

Such facilites are also ideal for checking hardware during the build-up of the accelerator equipment. For instance the command:

```
SRT CAMAC (1,12,0,16) = 312
```

can be given directly from the keyboard to set an interface CAMAC register.

```
FOR N = 1,8; FOR A = 0,15; TYPE N,A CAMAC (1,N,A,0)
```

will cause a bank of CAMAC registers to be printed out. A simple program to check if any zero-drifts become greater than 2 might be:

```
1.1 FOR N = 1,8; FOR A = 0,15; DO 2
1.2 END
2.1 IF (2 — ABS (CAMAC (1,N,A,0))) 2.2 S RETURN
2.2 TYPE! «OUT OF TOLERANCE» N,A, CAMAC (1,N,A,0)!
```

This program can be saved on a SINTRAN file by the command:

```
SAVE 65
```

then executed, say every 2 minutes by the SINTRAN command

```
* INTV TASK 65 2 3
```

Calling machine code routines is the primary feature of NODAL for on-line control work. These routines can take the form of normal call routines, e.g. CALL FFT (A, —1) or of read/write functions such as:

```
SET INTPHS = 12; TYPE BCT (5,7)
```

Thus NODAL programs at the accelerator level will be mainly a series of calls to «system variables» which are implemented as quite complex subroutines, after working in conjunction with a data-table

[2] The CERN SPS multi-computer and CAMAC real-time control system.

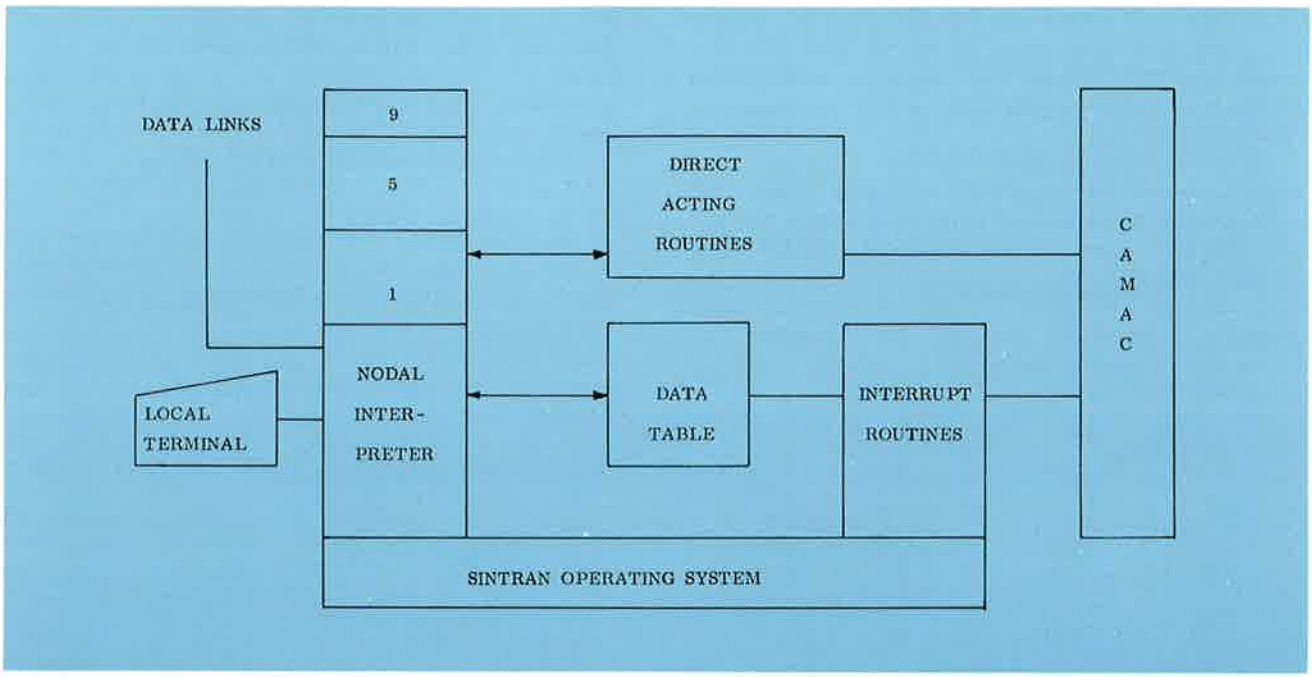


Figure 4. Software Layout in an individual NORD-10.

as shown in figure 4. Although the interpretation process is somewhat slow, a few milliseconds per statement, the use of powerful machine code subroutines enables a lot of work to be done in a short time. Fortunately the machine code routines which are mostly related to the hardware, or to special mathematical operations such as the Fast Fourier Transform, and so can be prepared in advance. The large number of applications programs using these routines, however, may not be specified, perhaps even the need for them not even realized, until late in the construction period or even during commissioning. Here the power and simplicity of NODAL will be invaluable as these programs can be written on-line by the machine-physicists rather than by specialist programmers.

Communication between computers is the other important role of NODAL. A NODAL program consists of a series of text and data blocks which are interpreted by the resident interpreter program. These blocks are arranged to be independent entities, relocatable within one computer and independent of the particular computer. The format of such blocks, or entries, is shown below.

LINE ENTRY

SIZE
TYPE = 1
LINE NO
BYTE COUNT
TEXT

NAME ENTRY

SIZE
TYPE
DESCRIPTORS
BODY OF ENTRY

All entries start with a word which gives the size of the entry, then one which gives the type of entry. Entries are then divided into two main classes, line entries and named entries. Line entries represent one line of NODAL, e.g.

```
1.1 SET INTPHS = 12; SET A = 1
```

Named entries can represent variables, arrays, or even groups of line entries which make up a NODAL subroutine. Integer arrays, read-only arrays, and variables are provided.

Such entries are easily transferred between computers to provide the required inter-computer dialogue. For example, typing in a console computer

```
1.1 SET INTPHS = 12
EXECUTE (8) 1.1
```

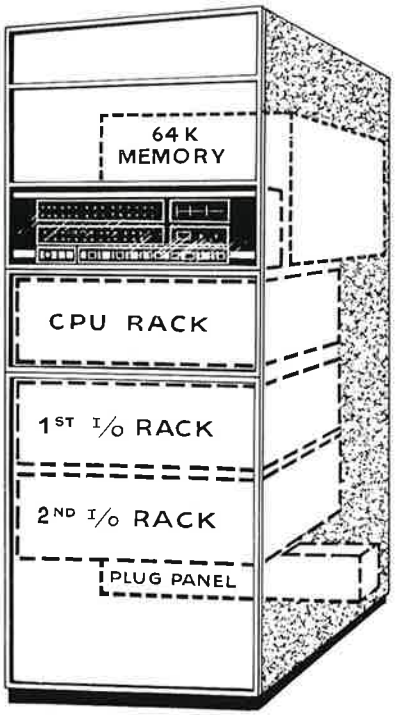
will cause the line entry 1.1 created in the console computer to be sent to computer 8 for execution. Similarly to acquire and print out some power supply currents from the injection computer one could use the program:

```
1.1 EXECUTE (INJCMP) 2; WAIT INJCMP
1.2 FOR E = 1,20; TYPE «POWER SUPPLY»
    I «CURRENT» A (I)!
1.3 END
2.1 DIM A (20)
2.2 FOR E = 1,20; SET A (I) = PSINJ (I, 1)
2.3 REMIT A
```

These EXECUTE, REMIT, and WAIT commands together with the relocatable NODAL entries thus provide the required communication.

INSIDE NORD-10

BY CAND. REAL. JAN ASKE BØRRESEN,
A/S NORSK DATA-ELEKTRONIKK



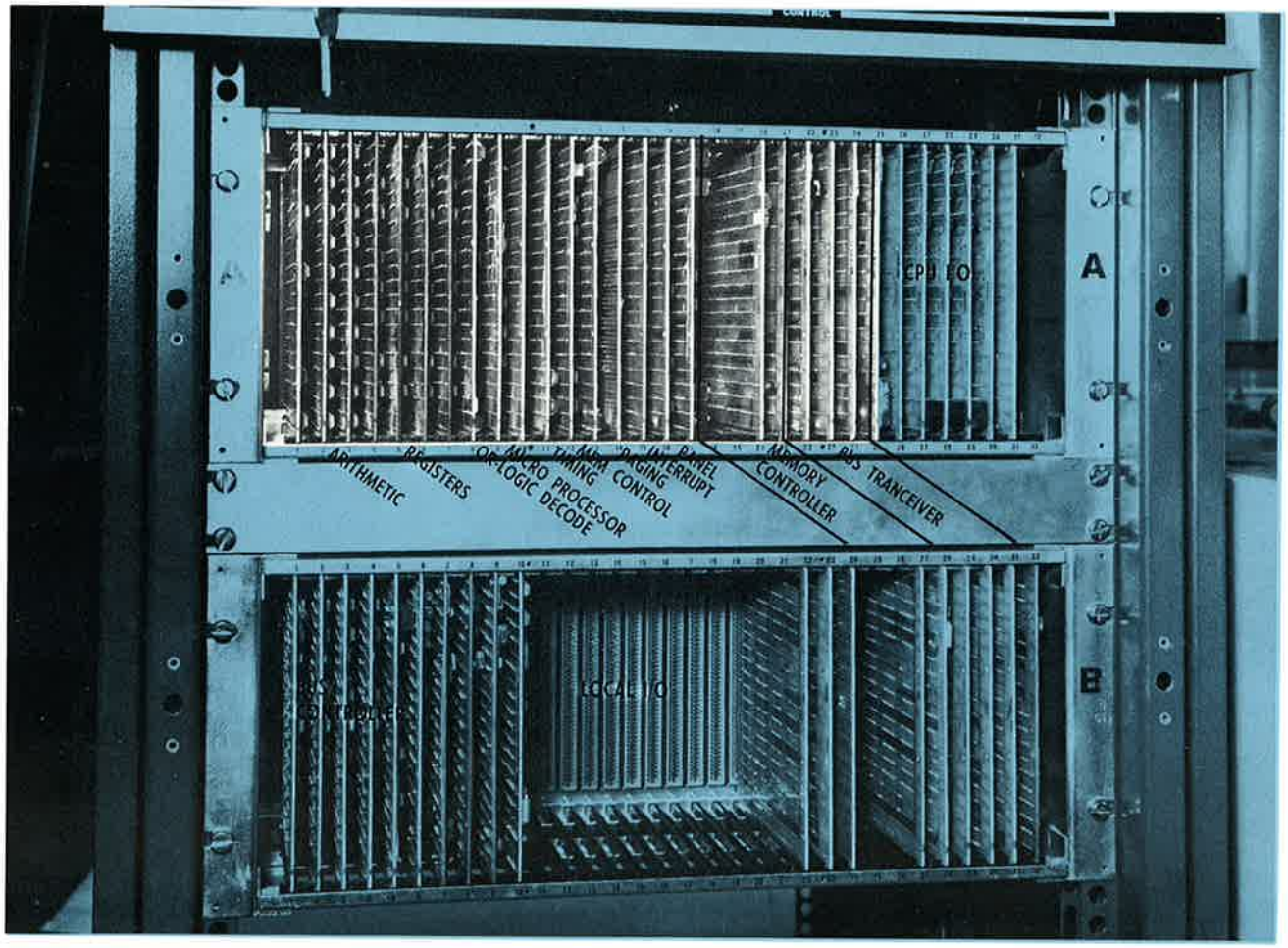
STANDARD NORD-10 cabinet with the different modules stippled.

Introduction

NORD-10 is a medium-sized general purpose computer designed for multi-lingual time-sharing applications and for real-time multiprogram systems. A key word in the design has been modularity, and the computer is easily expandable in the field due to its flexible structure.

The central processing unit has a Micro-Processor and instructions, operator communication, «bootstrap loaders», and hardware test programs are implemented in a 1K Read-Only Memory. The Micro-

The picture, which is taken from the front, shows the CPU rack and one I/O rack.



Processor also allows for customer specified instructions to be built in. NORD-10 has a memory management system with hardware paging extending the memory size from 64K to 256K 16-bit words and two independent working protecting systems, one acting on each page and one on the mode of instructions. The interrupt system has 16 program levels in hardware, each with its own set of general registers. The context switching from one level to another is extremely fast. Interrupting external devices are identified uniquely by a vectored interrupt. The I/O Bus is common for all devices, either program controlled or on direct memory access. The bus system is divided into groups to prevent a device from jamming the whole bus in case of malfunction. In the following NORD-10 will be presented as seen from the inside, trying to elucidate the different construction principles mentioned above. The drawing shows a standard NORD-10 cabinet where the mounting of the different modules to be described are sketched. NORD-10 is built up of standard 19″ rack with 32 positions. Access to all CPU cards is from the front. In the rear the core memory is housed and in the bottom is the plug panel for connecting all external devices.

The Central Processing Unit

The CPU consists of totally 24 cards as seen on the picture. The last 8 positions in the rack are to be used for input/output devices operated by program such as console Teletype, paper tape reader, paper tape punch, card reader, line printer, display, operator's panel, real-time clock.

Connection between the first and the second rack is a general I/O Bus which will be described later. The bus transceiver occupies the last three positions of the 24 CPU cards. The memory control for 32K standard configuration requires two rack positions in the CPU rack.

NORD-10 has 160 registers of which 128 are available to programs, 8 on each of the 16 program levels. 6 of these registers are general registers, one is a program counter and another contains status information.

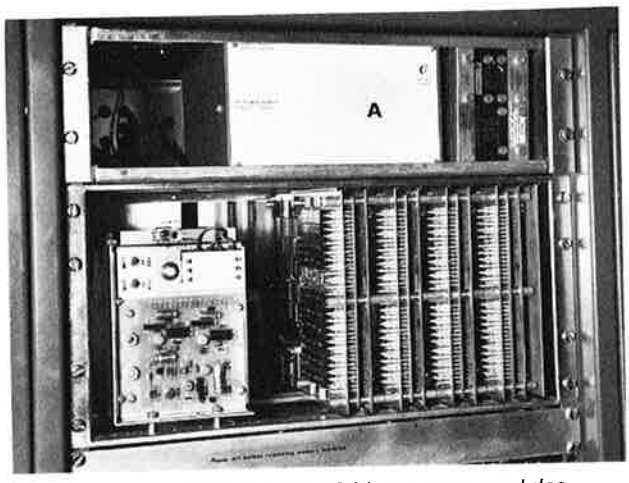
Floating point operations are standard. NORD-10's instructions can operate on 5 different data formats. These are:

- single bit
- 8-bit byte
- 16-bit words
- 32-bit double words
- 48-bit floating point words.

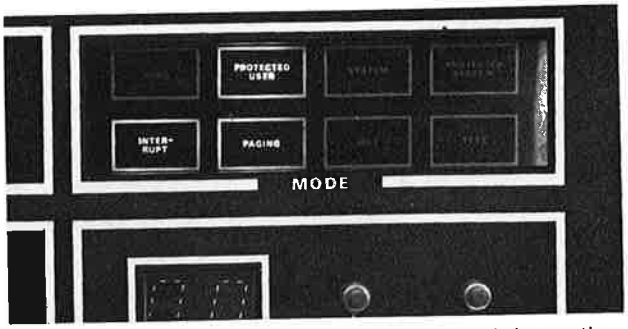
NORD-10 can address 64K of 16-bit words, and there are no restrictions as for the lengths of instructions or data fields within these 64K.

The Memory

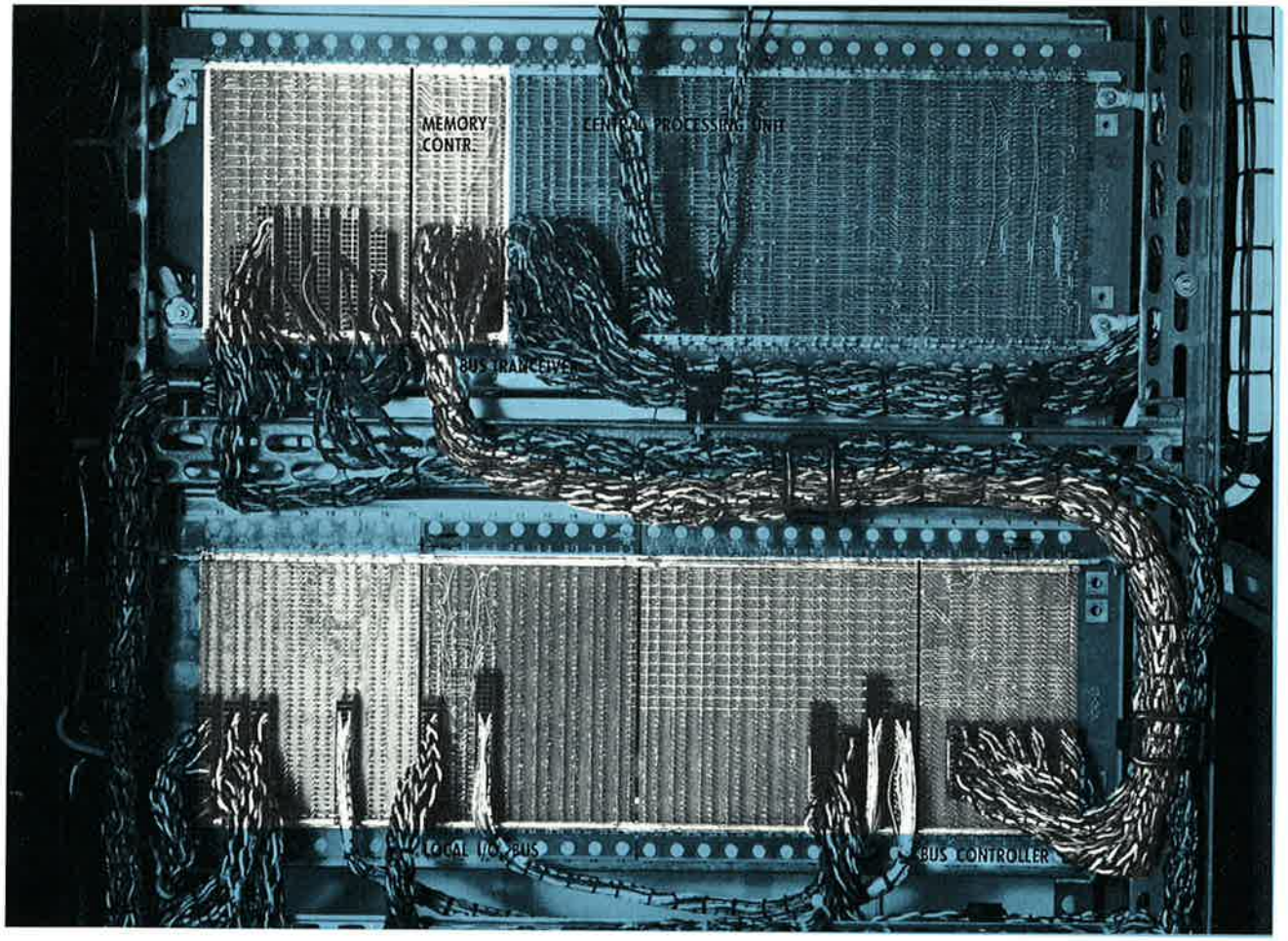
The memory system is built up of 8K 16-bit modules housed in a special memory rack as shown in the picture. One 19″ rack can take up to eight 8K modules, the power supplies are then mounted above the rack. The unit shown on the picture is a 32K



The memory rack with 4 8K 16-bit memory modules.



Section of the front panel showing the mode of the particular program now running.



The CPU rack and the first I/O rack as seen from the rear, and with the different parts of the Bus system outlined.

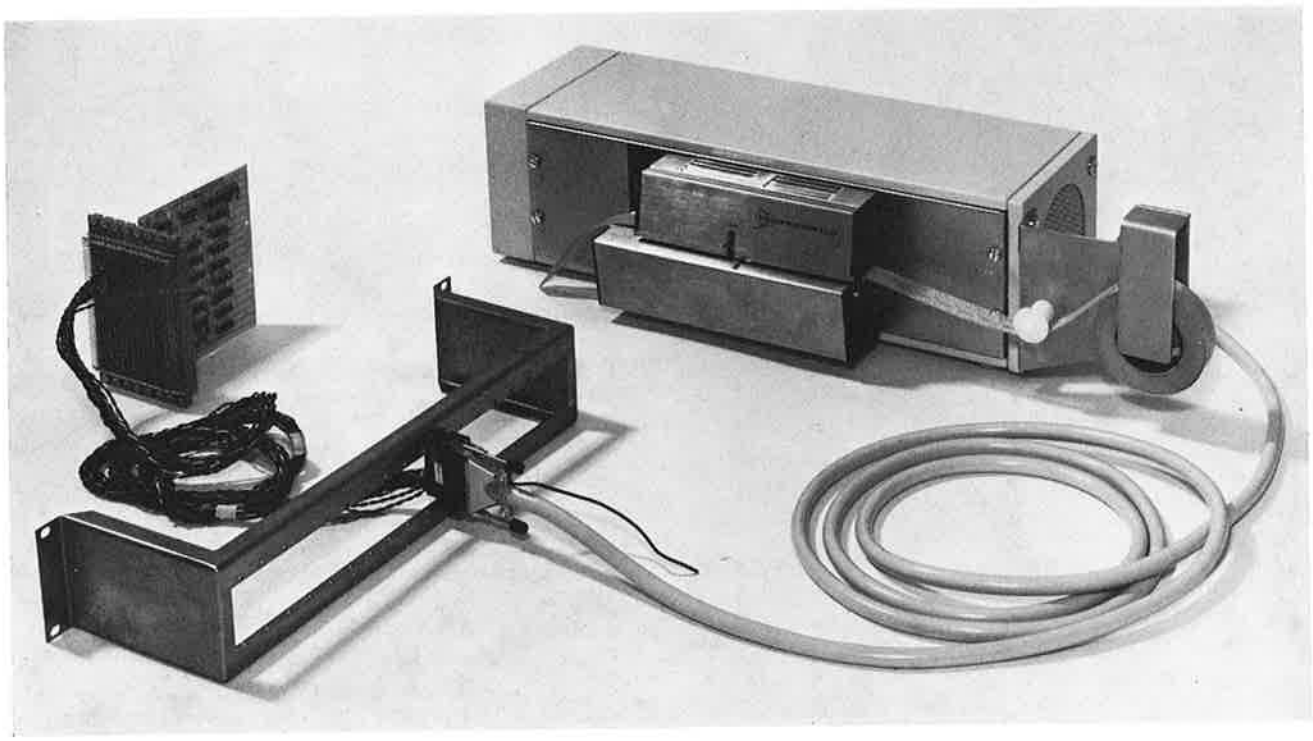
version, all complete with power supply and fans.
By use of a paging system, it is possible to extend NORD-10's physical address space beyond 64K up to a maximum of 256K of 16-bit words. The paging system translates a 16-bit virtual address into an 18-bit physical address.
The hardware paging system makes it possible for one user to write programs up to 64K (virtual memory), and only parts of the program are present in physical memory at any time (dynamic memory allocation). The paging system divides memory into 1K pages. The 4 page index tables are found in a 256 word extremely fast memory block. The calculation of a physical address results in no appreciable delay in the effective memory cycle time.
NORD-10 has two independent protection systems. Each individual page can be protected against being read from, written into (type data or type instructions), or against reading of instructions. In addition, there is a system which divides the pages into four different categories, here called rings. The rings have a priority from 0 to 3. A program on a

lower ring is never allowed to access the pages on a higher ring. Programs which run on rings 2 and 3 can use the whole NORD-10's instruction set, while programs on rings 0 and 1 only have a limited instruction set available. The different rings are displayed on the operator's panel as shown in the picture. For example, ring 0 (USER) may hold a user program, while compilers and assemblers run on ring 1 (PROTECTED USER). The bulk of the operating system can run on ring 2 (SYSTEM), and the kernel of the operating system on ring 3 (PROTECTED SYSTEM). If one attempts to execute privileged instructions on ring 0 or 1, or attempts to access a protected page, a hardware status interrupt will automatically be generated on program level 14 indicating the error.

The picture shows how a paper tape reader is interfaced. The device control logic plug into the bus from the front. The physical connection to the device is done through an internal cable and the external cable meeting at a plug board.

I/O System and Bus Architecture

NORD-10 is equipped with a common bus system for all external devices. The bus system is divided into groups, and a great deal of effort has been made to ensure that no device will be able to jam the bus system in the case of malfunction. The system is separated from the CPU and memory by a set of electronic switches called Bus Transceiver. Each group has its own controller which in addition to functioning as an electronic switch for the bus system, can also change priority for the whole group. The picture shows the CPU rack and the first local I/O group as seen from the rear. All interconnections between the cards are done with multilayer printed circuit backwiring boards, and all I/O interfaces have the same standard form. The system can therefore be extended or reconfigured by plugging in new or shifting around the existing interface cards.

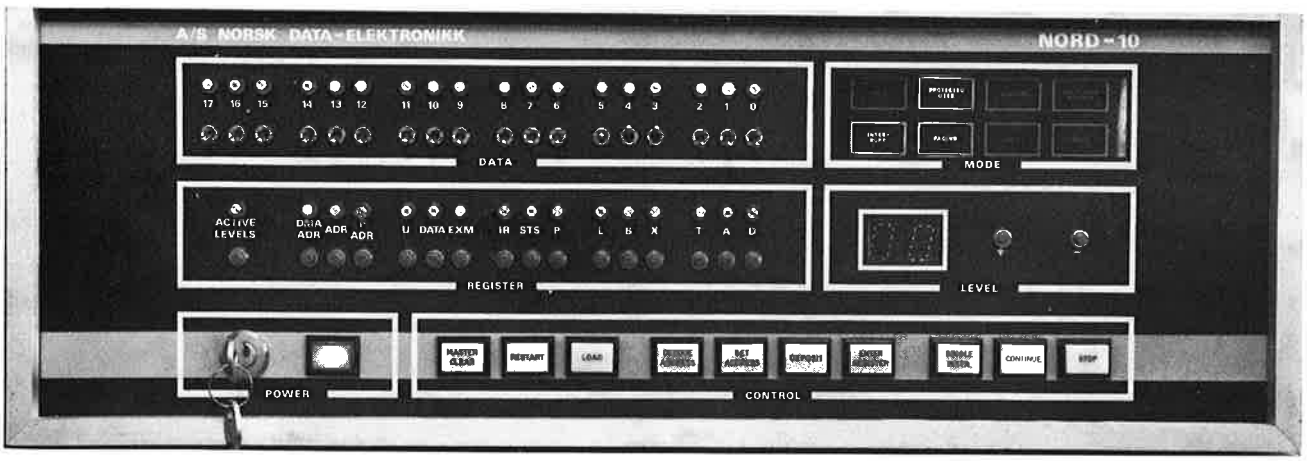


The position of the device interface in the card rack determines the interrupt priority of the device. If several devices within one rack are connected to the same interrupt level, the device closest to the Bus Controller has the highest priority within that level.
The address, data, and control information are sent to all devices via 50 bus lines. In a programmed I/O data transfer, for example, the user addresses the I/O device with its special address. The device answers, and data can be sent or received. In DMA transfers the device sends a «REQUEST». The CPU answers with a «GRANT» signal, and this signal is sent from device to device until it comes to the device which initiated the «REQUEST», and transfer to the memory can then take place. When two or more devices request a DMA transfer simultaneously, the device situated physically closest to the CPU has the highest priority. One memory cycle later the next DMA along the chain will be allowed to send data, and so on, until a higher priority device again sends a «REQUEST». This means that many DMA devices can use the same bus system at the full data transfer rate. It is not necessary to establish a «master-slave» connection. The transfer rate is one 16-bit word/850 nsec, or 1.1 M words/sec.
The printed backplane of the I/O bus is modular in groups of 8 interface slots. The picture shows how a tape reader is connected to the system. The control card is placed in the appropriate slot according to the system configuration. The cables and plugs both for interconnecting the I/O units to the Bus system and the cables and plugs for physically connecting the I/O devices to the NORD-10 have been standardized.
Interfaces for mass storages as disk, drum, magtape, etc., are built with one interface card to be plugged at the appropriate place in the Bus system, the remaining control cards (6—7) are placed in one of the backplane modules as shown in the lower rack of the picture of the CPU and I/O racks.

The Interrupt System

NORD-10 has a multiprogram system with 16 priority program levels. Each program level has its own set of registers, including a program counter and a status word. The levels running can be shown on the front panel by pressing the button ACTIVE LEVELS. Levels 0 through 9 are used for Programs. Internal hardware status interrupts are assigned to level 14, whilst level 15 is reserved for extremely fast user interrupts.
The picture shows that programs are running on levels 1, 4, and 8. Hardware devices on levels 10, 12, and 13 are used.
Levels 10, 11, 12, and 13 are reserved for external devices. Each device has its own unique identification vector. In all 2048 such vectors are available. The «IDENT» instruction determines which device is

The front panel of NORD-10 at the moment showing which of the 16 program levels are active.



giving an interrupt. The identification of an interrupt takes 1.8 μsec includnig the «enabling-disabling» of the registers.

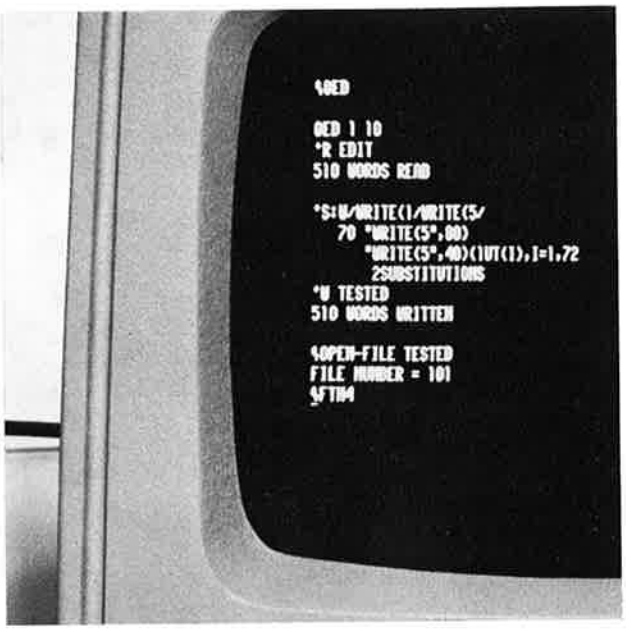
System Software

NORD-10 is delivered with a powerful time-sharing system, NORD-TSS, and a real-time multiprogramming operating system, SINTRAN III. The minimum configuration for SINTRAN III includes a standard NORD-10 with 8K of core. The monitor schedules the real-time programs and supervises the execution. The real-time programs can be coded in assembly or FORTRAN. With NORD-TSS all users can simultaneously run any of the systems FORTRAN IV, BASIC, MAC Assembler, NODAL, NORD-PL, or QED. The picture shows an example of the use of a time-sharing terminal where a program is being edited under the QED Editor. In the time-sharing system as well as in the NORD File System the files are called by name. The command

QED	causes the QED program to be loaded for this particular terminal.
QED 1.10	answer from the system that QED is ready for input.
*R EDIT	this command means read from the file called EDIT.
510 WORDS READ	answer from the system that the file has been read into the QED buffer and it consists of altogether 510 words.

All FORTRAN WRITE statement can, for example, be changed from output on line printer (logical device No. 5) to output on this particular terminal (logical device No. 1). The Substitute «S» is used:

*S:W/WRITE(1/ WRITE(5/	causing the contents between the last two break signs (/) to be substituted by the contents between the first and the second break signs (/).



The picture shows how to edit a program on a time-sharing terminal using QED.

70«WRITE(5»,80)
«WRITE(5»,40)(1UT(I), = 1,72)

:W is an optional command which causes the line to be printed where the text string to be changed is found. When no other parameter is specified, QED scans the whole text starting from the current location.

2 SUBSTITUTIONS	2 substitutions are made.
*W TESTED	The command W causes the contents of the QED buffer to be written on a file named TESTED.
510 WORDS WRITTEN	The file transfer completed.
α OPEN-FILE TESTED FILE NUMBER = 101	The command causes the file TESTED to be opened for use by this terminal under the time-sharing system.

α FTN4

This command causes the FORTRAN IV compiler to be loaded and the program on the file 101 can be compiled and run.

Altogether the software and hardware provide the user with a powerful tool for solving his problems now, and in the years to come.

CAMAC for NORD, the international interface standard in use for NORD-1 and NORD-10

PER JØRG JOHNSEN AND ROLF NORDHAGEN, PHYSICS INSTITUTE, UNIVERSITY OF OSLO

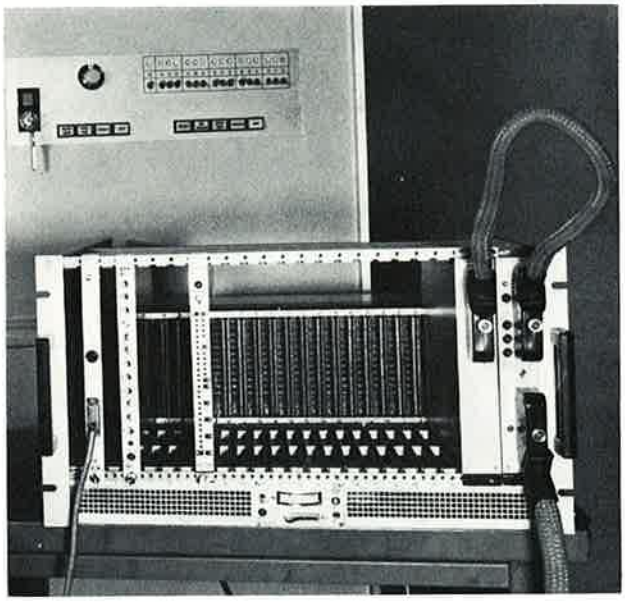
Introducing CAMAC

With the introduction of minicomputers in the laboratory, it was soon realised that a considerable technical effort would be needed in order to interface all kinds of equipment to the computers. Particularly in the large nuclear physics laboratories, where electronic registration and control units were used in great number in various types of experimental setups, the need for a standard interfacing scheme to ease the technical and programming problems became apparent. In the English laboratory Harwell work on an interface standard was started in 1964, and as any successful standard would need international approval, cooperation with the European Standard of Nuclear Electronics Committee (ESONE-committee) was obtained. Within 1966 a working group representing 10 European laboratories was established, the name CAMAC was chosen for the project, and a preliminary proposal published in January 1966. Since then a number of supplementary standards have been approved, cooperation and approval by the instrumentation standards committee of the National Bureau of Standards in U.S.A. have been obtained, and within 1971 36 laboratories in 15 different countries were involved in the work. Up to late 1971 the total work was estimated to represent an effort of nearly 54 man-year, and at total cost of roughly 4 mill. $. The adoption of the CAMAC-standard by the instrumentation industry has been rapid and extensive, by latest count CAMAC equipment is manufactured or sold by 64 companies in the field of electronic instrumentation and computer equipment.

CAMAC then is an international standard for interfacing external devices to a processor. The name itself is an anti-acronym, it has no particular meaning but is pleasing to the tongue in most major languages. Its only symbolic content is that the name is symmetric. Actually, considerable effort has been made to assure that the name is not associated with any commercial interests, or any other particular places or names.

The CAMAC standard

CAMAC consists of both a mechanical, a logical, and an electrical standard. The main component is the definition of a «dataway» for exchange of information between connected devices. The mechanical implementation consists of a crate with carefully defined dimensions (fig. 1). The crate has room for 25 modular units, and the construction takes into account modern manufacturing techniques, as well as the high degree of compactness permitted by the use of integrated circuits. The dataway is brought out to each module through an 86 pin connector at the rear of the crate. 25 connectors each define a station on the dataway. 24 of these are normal stations, 1 station at extreme right in the crate is reserved for a control station. A modular unit can occupy one or more stations, or more than one unit of the crate front. The standard also contains the specifications of power supplies, which can be integral to the crate or separate.



A CAMAC-crate connected to the NORD-1 computer. A crate-controller module in stations 24 and 25 to the extreme right. The dataway connectors visible at the rear of the crate.

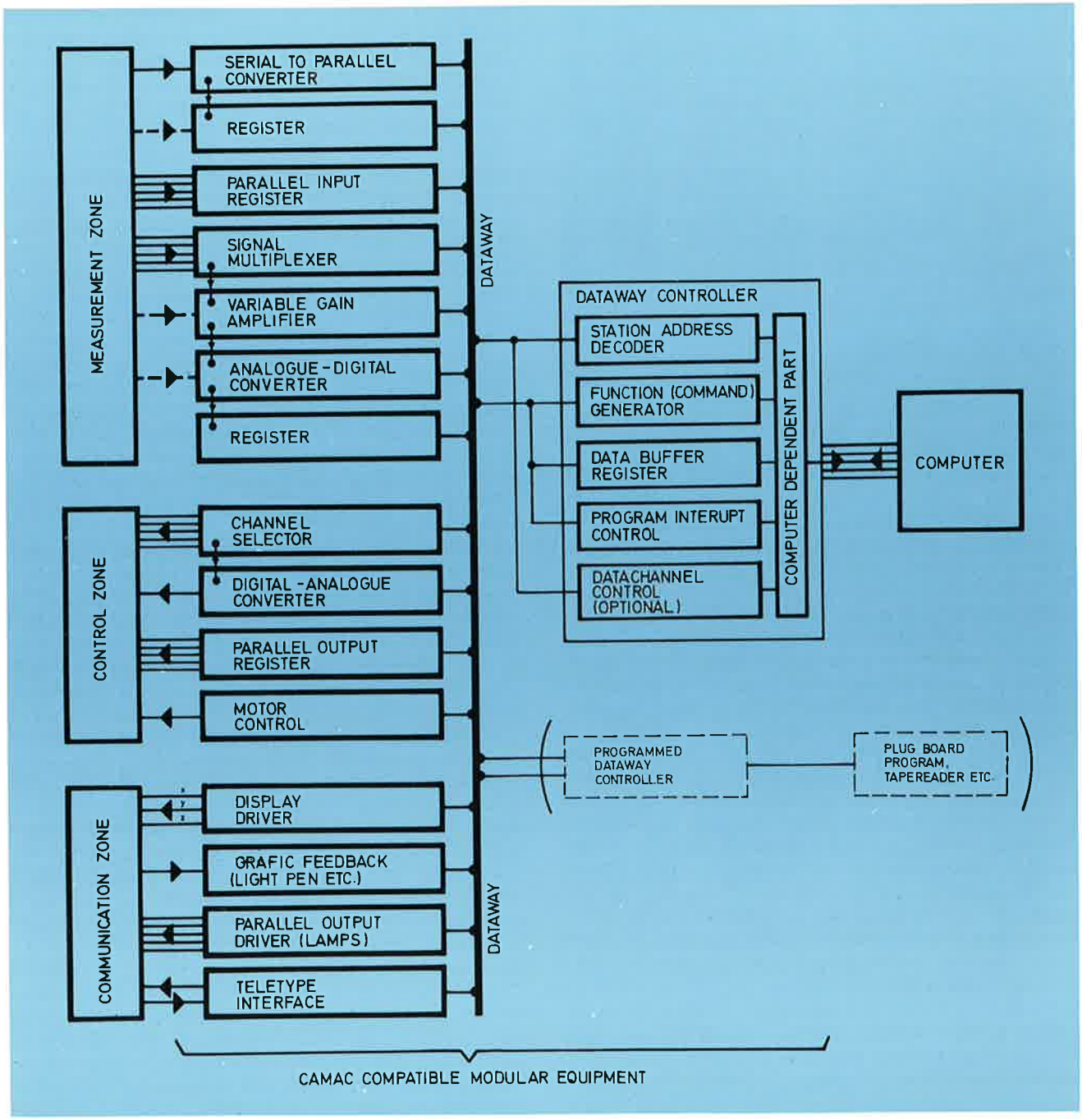
CAMAC is logically constructed to operate on 24 bit words. The dataway itself contains 24 separate read and write lines, 24 control lines and 14 power supply lines which are all brought out to each station. All lines are common except two, a station line (N) and an interrupt line (L) which connects each station

by separate (private) lines to the control station. The dataway connections are given in table 1.
Each station module communicates with the crate controller which is a special module using stations 24 and 25. The communication scheme between a normal module and the controller modue is defined by the standard, the controller generating a station number (N), register subaddress within the module (A), and module function (F) in response to control information generated either by a processor (computer) outside the crate, or by local information programmed into the controller by hardware (switch operated crate controller) or fed in e.g. by a papertape reader. A crate, although oriented towards compute operation, may thus be made to operate as a stand alone control system. The controller can be called by interrupts from each station, and transmit signals (Q) to the process control thus initiating appropriate action on the module. Moreover, a high degree of standardisation has been laid down in defining the module functions, which cover most normal input/ output operations, as register read and write, increment, disable, enable, test status etc.
The crate controller thus provides for a very flexible communication with any station module. The controller itself may be interfaced to a computer, and the specifications of this interface is given by the particular computer used. Thus the division between the CAMAC standard and a computer input/output structure is in the controller. For single crate systems special controllers are made to fit particular computers, and as long as such controllers are available, any CAMAC-module will fit the system.
The original standard is published in the report EUR 4100, covering the crate and dataway specification. An extension to multiple systems is given in a report EUR 4600. Here, a «branch highway» is defined, consisting of 7 crates connected («daisy-chained») by a 66 line highway to a branch driver. On the branch highway signals are defined to provide for a handshake type of operation, making the communication independent of module types and cabling lengths.
Furthermore, each crate contains a crate controller defined by the standard, and connecting the crate to the highway. The branch driver will provide the necessary interface to the computer. Further standards cover CAMAC-analogue signals in a report EUR 5100.

The use of CAMAC

Although the large laboratories in nuclear physics and atomic energy have been in the forefront developing CAMAC, the design is aimed at covering as wide a field of laboratory and peripheral systems as possible. Both in the large laboratory with its massive use of electronic instrumentation, requiring complicated and time-consuming work to interconnect, as well as in small, one-user systems requiring a minimum af flexibility and usage in different situations, the use of CAMAC gives substantial benefits.
As CAMAC is a standard, computer-independent system, changes and additions to existing equipment can be made independently of the connected computer. Thus obsolence of any system is prevented. Moreover, essential freedom exists from the computer manufacturers' own line of peripherals and input/output philosophy. The user is given a wide choise of suppliers of input/output-units for his particular application. Also the computer manufacturer does not need to provide complicated hardware for all types of equipment, but can concentrate on the computer proper.
Stock-keeping, service, exchange of information and experience, all is considerably simplified by a standard system. The perhaps most exciting aspect of CAMAC is the possibility of developing a computer independent CAMAC language. Several working groups within the ESONE collaboration are presently engaged in formulating the specifications of such a language.
The design of equipment for connecting to computer or other processor controlled systems employing CAMAC is considerably simplified, as logic and power is available in the crate. Any design will have a wide range of potential applications, and for manufacturers of instrumentation- og dataequipment the market for standard systems is very much larger than for specialized solutions. Thus quantity production can be planned. The manufacturer can afford to aim at higher reliability and simplified maintenance, and lower cost may be possible.
CAMAC usage is illustrated in fig. 2. In a complete system we can identify a measurement zone, consisting of instrumentation giving analogue signal outputs, timing events, coded or uncoded position information etc. These inputs are serviced by analogue-to-digital converters, multiplexers, inputregisters,

Block diagram illustrating CAMAC-usage in a measurement zone, a control zone and a communication zone.



counters etc. We further identify a control zone, providing outputs to process or experiment controls, stepping motors, encoders, switches, analogue-signal levels etc. Digital-to-analogue converters, various kinds of output-registers and selectors service this zone. Finally, a communication zone consists of devices for interaction between operator and experiment or process. In this zone we find teletypes, displays, plotters etc., serviced by various kinds of drivers or input interfaces.
As mentioned CAMAC oriented modules and equipment are available from a substantial number of commercial sources. A larger number of readily available units can be obtained at reasonable prices to cover most needs in measuring, control, and communication as specified above. Thus not only the larger laboratories, but also smaller users in the fields of process control and laboratory data-processing should take a close look at the possibilities offered by a CAMAC system. Already CAMAC is in use internationally in such diverse fields as industrial process control, medical electronics, broadcasting, communications and general automation, in addition to widespread use both in the very large and the smaller nuclear laboratories.
Admittedly, initial purchase price for a CAMAC crate and controller may be high, in Norway 4—5000 $, and for highly specialized, single-task systems this may be too large an initial expenditure. However, as soon as more tasks and reasonable flexible configurations are planned, the freedom to change and evolve the system independently of the computer hardware make CAMAC an ideal choice for an interface system.

CAMAC in the NORD-computers

At the Physics Institute, University of Oslo, work on a CAMAC interface for the NORD-1 computer was started late in 1971. The intentions are to develop a general computer based data-laboratory for the various physics research groups at the institute, such as nuclear physics, biophysics, solid state and materials research to name a few. From the institute's central NORD-1, branch highways with one or more crates will be connected to each experimental lab. Also, through this work we hoped to raise an interest among local research and manufacturing groups in the fields of instrumentation and dataprocessing.

The work at A/S Norsk Data-Elektronikk on the CERN-contract, which specify a CAMAC-system for the computer control of the next, large European high energy accelerator (in Geneva, on the Swiss-French-border), provided a strong incentive for the local CAMAC efforts. In collaboration with ND, our group at the Physics Institute has finished the work on a prototype NORD-1 CAMAC driver, and is currently involved in the joint effort by CERN and ND to develop a NORD-10 based system.
The NORD-1 CAMAC driver:
Compared to the EUR 4600 highway standard the following changes have been introduced. As only very large experimental areas need the full set of 7 crates in a normal branch highway, the number in a branch is reduced to 4. Instead, the NORD-1 driver can service up to 4 branches. Thus one of these units can serve 4 experiments or processes with up to 4 crates in each. The standard crate controller can be used, supplemented by termination units in the last crate, and a special driver module in distant crates (more than 20 meters away from the driver). The driver normally operates on the NORD-1 computer I/O bus, that is by accumulator transfers. Provisions are made for data channel operations, but not yet implemented.
In the prototype system, the CAMAC driver is a group 2 device, working on interrupt level 12. Two device numbers are in use, one (mnemonic DAMAC) for transferring data and status information, as well as load and execute (by decoding ACT, SKA, and PIN, the three IOT bits in the standard NORD-10 I/O instruction), and one (mnemonic CAMAC) for the skipping and preparing of interrupts, Q-response, system flags etc.
The transfer of 24 bits data or command word to and from the crates is accomplished in a two step procedure. A Main Data register in the driver is written or read by «low end» instructions, transferring 16 bits, and «high end» instructions, transferring 8 bits. Similarly, commands are executed by an instruction transferring first a crate and branch address from the accumulator (A-register) to a command register. Next the same instruction is used to transfer an accumulator containing N, A, and F, station number, module subaddress and module function, respectively. A non-zero station number will initiate a CAMAC-cycle on the highway.

Reading the status register as the result of an interrupt (in a system with more than one branch and crate) will transfer to the accumulator the address of the branch and crate responsible for the interrupt. As the same bit positions are used for loading the branch and crate address in the non-executing part of the command-register, a command to read the station numbers of the actual modules giving rise to interrupts can be issued immediately. A simple service routine may look like this:

```
IOT STAT DAMAC      % READ STATUS REGISTER
JMP ERROR           % ERROR EXIT IS PROVIDED
RAND (77            % MASK OUT BITS 9—13
IOT ACT DAMAC       % BRANCH AND CRATES
                      TO COMMAND-REG.
LDA RLAM            % READ LOOK-AT-ME
IOT ACT DAMAC       % EXECUTE
etc.
```

The address RLAM will contain the standard CAMAC NAF-bit-pattern to read the station interrupt from the addressed crate. Note that at the end of the executed CAMAC-cycle the station interrupt bits will be available in the CAMAC Main Data register, and must be read by the «low» and the «high end» commands.
This example exhibits another feature in the use of CAMAC. Clearly, the necessary software procedure for data transfer between the computer and the CAMAC system is considerably more cumbersome than the direct use of special interfaces for each device. However, this is more than compensated for by the possibility of coding standard dialogues with all devices.

NORD-10 and CAMAC

The NORD system described above must be regarded as a forerunner of a more general system which will be based on ND's new computer, the NORD-10. First of all the fast interrupt system of NORD-10 makes this computer particularly suitable for real-time applications (one of the reasons for CERN choosing NORD-10 for the accelerator system). Also, the input/output structure of NORD-10 is very well suited to service a CAMAC-system, being in itself built along a design with a bus structure, programmed access to a large number of device registers, and a standardisation of device dialogue procedures. This similarity is obviously due to the general applicability of these types of ideas to interfacing schemes.
The NORD-10 CAMAC will be based on the experience gained at CERN with implementing a larger number of CAMAC-systems in the last years. It will take particular advantage of the speed and ease of the NORD-10 direct memory access, and of the efficient CPU with a full set of registers for each of the 16 interrupt levels. The designed system is expected to be one of the most powerful CAMAC-systems for any computer at present.
Instead of constructing a standard branch highway, on the NORD-10 CAMAC driver the computers own I/O bus is extended and will permit a chain of up to 16 crates. In this way each of the CAMAC-crates may be considered as a standard peripheral device, and any register in the crate controller can be read directly (by the NORD-10 IOX-instruction) in one cycle without intermediate buffering. Equally important, the time taken from any specific module issues a «LAM»-request (look-at-me) until the CPU obtains a unique identification number, will be kept low. Using the identification number as a service routine pointer, the interrupt-to-service time will normally be less than 8 μs.
The NORD-10 IOX instruction will transfer any external register, identified by the 11 bit address part, directly to the accumulator. Among the 11 bits, one bit indicates CAMAC input/output (as opposed to standard peripherals) and 4 bits identify the crate. The remaining bits indicate a register address (3 bits) and an operation code (3 bits with operations as read, write, bit clear, bit set). If the register for the N, A, and F information (station number, module register address, and function) is addressed, the accumulator content will be taken as a CAMAC-command. In this case, a CAMAC-cycle is executed on the dataway within the IOX-cycle. Furthermore, if the function code initiates an input from a station module, the input data will be present in the accumulator at the end of the same cycle.
As a CAMAC-module is treated (almost) as a standard peripheral device, this restricts the wordlength to 16 bits also within the crate. This is not considered

Table 1. CONTACT ALLOCATION AT A NORMAL STATION

Bus-line	Free Bus-line	P2	B	Busy	Bus-line
Bus-line	Free Bus-line	P3	F16	Function	Bus-line
Individual patch contact		P4	F8	Function	Bus-line
Individual patch contact		P5	F4	Function	Bus-line
Individual patch contact		X	F2	Function	Bus-line
Bus-line	Command Accepted	I	F1	Function	Bus-line
Bus-line	Inhibit	C	A8	Sub-address	Bus-line
Bus-line	Clear	N	A4	Sub-address	Bus-line
Individual line	Station Number	L	A2	Sub-address	Bus-line
Individual line	Look-at-Me	S1	A1	Sub-address	Bus-line
Bus-line	Strobe 1	S2	Z	Initialize	Bus-line
Bus-line	Strobe 2	P1	Q	Response	Bus-line
		W24	W23		
		W22	W21		
		W20	W19		
		W18	W17		
24 Write Bus-lines		16	W15		
		W14	W13		
W1 = least significant bit		W12	W11		
W24 = most significant bit		W10	W9		
		W8	W7		
		W6	W5		
		W4	W3		
		W2	W1		
		R24	R23		
		R22	R21		
		R20	R19		
		R18	R17		
24 Read Bus-lines		R16	R15		
		R14	R13		
R- = least significant bit		R12	R11		
R24 = most significant bit		R10	R9		
		R8	R7		
		R6	R5		
		R4	R3		
		R2	R1		
Power Bus-lines	—12V d.c.	—12	—24	—24V d.c.	Power Bus-lines
	+200V d.c.	+200	—6	—6V d.c.	
	117V a.c. Live	ACL	ACN	117V a.c. Neutral	
	Reserved	Y1	E	Clean Earth	
	+12V d.c.	+12	+24	+24V d.c.	
	Reserved	Y2	+6	+6V d.c.	
	0V (Power Return)	0	0	0V (Power Return)	

a serious limitation since 24 bit modules are rare. In the cases where more bits are actually needed, the use of two separate commands will be more convenient, at the same time permitting a precision of 32 bits.
Other features will include separating off the direct-memory access (DMA) from the normal crate-controller, using instead an optional extra module in those cases where DMA is needed. Another feature will be the possibility of executing data-channel transfers in two modes. In a BLOCK MODE, the dataway is synchronized with the NORD-10 central processor to give a max. transfer rate of 0.9 M words/s. In an INTERLEAVE mode randomly occurring single-word data may be transferred with a max. rate of about 0.5 M words/s.
Finally then, we note that a simplified version of the NORD-10 will be offered by ND at a reasonable price. Such a machine, coupled with a CAMAC-interface system, should provide the smaller laboratories with a very flixible tool for doing laboratory automation and on-line experimenting. In particular, this will make it possible to give the experimenter essential freedom to interface a large number of instruments and equipment types on his on-line computer, without getting involved in complicated interface specifications and purchasing procedures. One may also hope for the emergence of general software tools applicable to a large number of installations thereby achieving the great benefits of collaborations in software development and exhange of programs. As is well known, in the future hardware prices are expected to decrease for comparable capabilities. Simultaneously, the expense of software development will probably increase. To improve on this situation, standardisation of schemes for handling external devices with computers will be of considerable benefit.

Literature

Available from the ESONE-committee:
Dr. W. Becker
JRC EURATOM
I-2100 Ispra (Va)
Italia.

1) EUR 4100 (e): A modular instrumentation system for datahandling. Description and Specification.
2) EUR 4600 (e): Organisation of multi-crate systems. Specification of the Branch Highway and CAMAC Crate Controller Type A.
Available from CERN:
Mr. J. Halon
CERN-NP
1211 Geneve 23
Sveits
3) F. Iselin et. al., Introduction to CAMAC, CERN-NP CAMAC note 25-00.
A journal is available at a price of approx. N. kr. 40,— per year from:
Commision des Communantés Europeennes
29, rue Aldringen
Luxembourg
4) Camac Bulltetin, ed. group H. Meyer et. al.
On Software Development:
5) Proposal for a CAMAC-language, CAMAC Bulletin supplement for issue no. 5, November 1972.
From A/S Norsk Data-Elektronikk.
6) Description of a CAMAC Branch Driver for NORD-1.

One-Line Evaluation of Bubble Chamber Data in Bergen

BY EGIL FETT, THE UNIVERSITY OF BERGEN

If you look through publications on experimental high-energy physics within the sixties, you will notice the dominating role bubble chambers played as particle detectors during this period. This is due to the enormous amount of information contained in bubble chamber photographs compared to other techniques. However, this amount of information (much of it is irrelevant to a specific problem) also causes a new problem:

«How to sort out the interesting information for subsequent data-analysis.»

For this purpose many devices have been invented to do automatic or semi-automatic «scanning» and data-collection. Common to all these methods is on extensive use of computers, large ones as well as small ones.

Let us first recall how bubble chambers work. Essentially a bubble chamber is a vessel containing a liquid, which at the time when a pulse of particles is let into it, is superheated such that the liquid is about to boil. Several cameras can look into the chamber from different angles and a strong magnetic field is superimposed. The first bubbles then appear around the ions produced by charged particles passing through the liquid. When these bubbles have grown to a visible size and before other bubbles have become visible, all cameras take photographs simultaneously. The pictures then show tracks of bubbles along the paths of the charged particles. Since there are several views (pictures) taken of the same event, a geometrical reconstruction in three dimensions may be done. The particles are deflected by the magnetic field, hence the curvature of the tracks may be used to determine both charge and momentum of the particles.

The steps in the data-analysis are roughly as follows:

a) Scanning, i.e. looking at the film and finding interesting interactions.
b) Measuring points along the tracks (on films).
c) Geometrical reconstruction of the points and fitting curves through these points, thus giving the curvature and direction of the tracks.
d) Combining the different tracks of the measured picture by imposing kinematical equations of constraint (including mass hypotheses) to identify the chain of interactions that has occurred.
e) Combining similar chains of interactions and doing statistical studies.

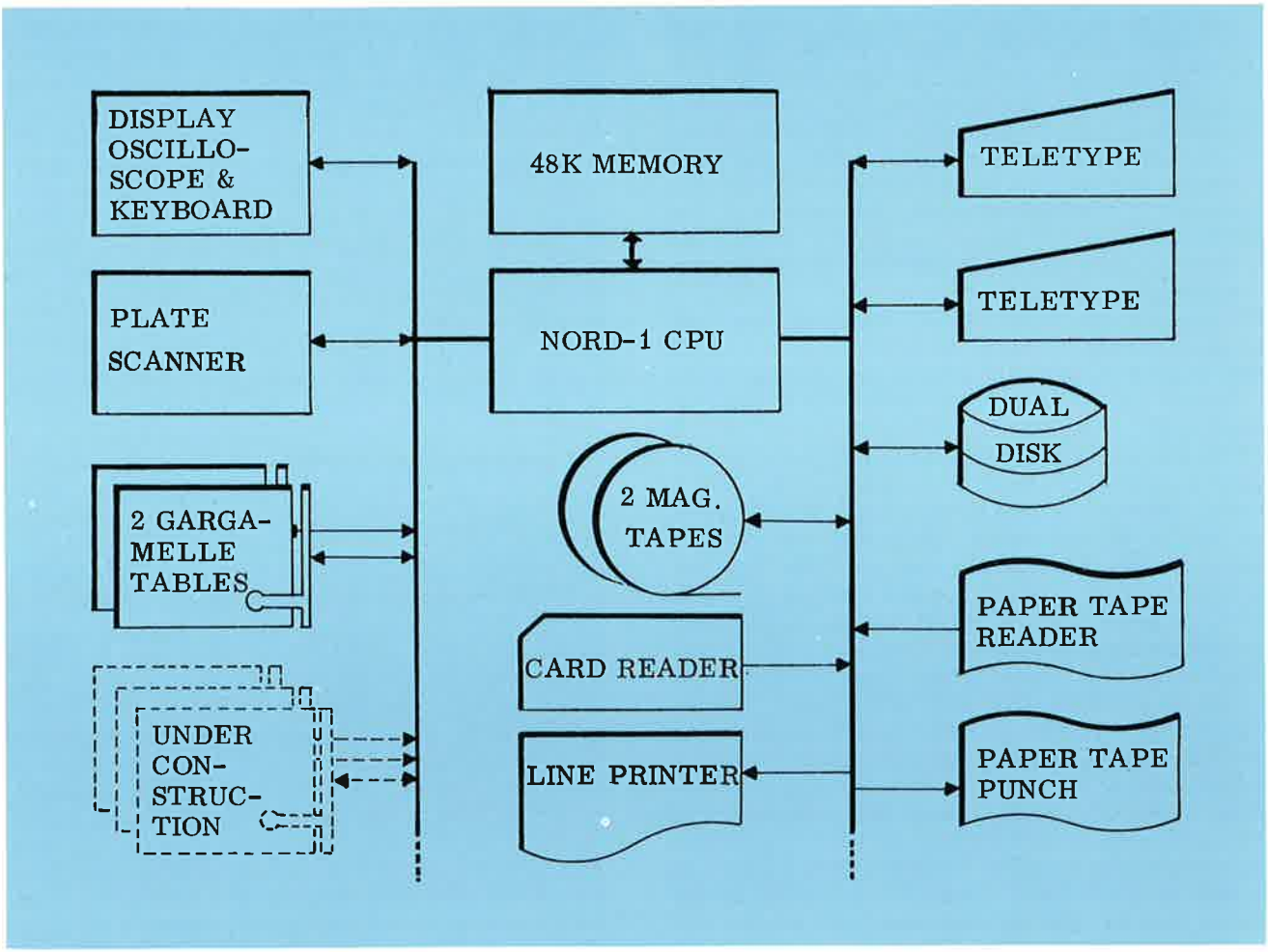
The last two steps are performed by large computers. Input and output are normally well edited magnetic tapes. The first steps, however, are more complex, involving human-operated measuring apparatus and sometimes more than one computer. Although the final geometrical curve fitting is normally done on large computers, use of smaller special-purpose computers is often made to guide and speed up the measuring procedure. We even feel the need of having large computers directly connected to the measuring device, thus being able to do full geometrical reconstruction on-line.

At the Institute of Physics, University of Bergen we started bubble chamber data analysis in 1962 using an IBM 1620 as a «large» computer. The measurements came out on punched cards and a very primitive geometrical reconstruction programme was executed on that computer. In 1966 we had to go to Copenhagen in order to find a «large» computer (an IBM 7090) for our new programmes. In 1967 we had become «big» in Bergen, having an IBM 360/50H at the University, which has now been replaced by a Univac 1110

During the same period bubble chambers in CERN (from where we get our films) have changed an got bigger as well. Last year we started to work with a new chamber called Gargamelle. According to the French literature on the subject, she ought to be a giant, and in some ways she really is. For example, no single camera can see the whole volume of the chamber, so we may see one part of a track in one view and the rest of the track in another view. This makes the first step (finding and identifying interesting events) almost impossible without a dialogue between the operator and a computer. For this purpose we have got new measurement equipment with a NORD-1 on-line. The NORD-1 was bought in 1970 for two special on-line purposes: The guiding of an automatic plate-scanner used for studies in low-energy nuclear physics, and the guiding of five Gargamelle measurement tables. (Two of these measurement tables are now in operation.)

The configuration of our NORD-1 machine is shown in the figure.

It is thus seen that our present «small» on-line computer is considerably larger than our «large» computer 10 years ago, and we may ask ourselves if it possible to do some geometry and kinematical fitting



on-line. However, we must realize that the effective core at our disposal for each event is considerably less, since we have to share the machine between five measuring tables and the plate scanner, and this requires a considerable system-overhead.
Let us now look at the system as it runs today. The bubble chamber pictures are projected down on to a table that acts as a screen. The projectors are fixed in the laboratory (in the ceiling) whereas the operator and the table are placed on a wagon on rails, such that the operator can move relative to the picture to study details that otherwise would be too far away. A coordinatograph is fixed on the floor with its effective range covering the whole picture.
The I/O-units at disposal for **each** measurement table are

a) A button to transmit 3 coordinates (x, y, 0) from the coordinatograph to the computer, 0 is the angle of a rotatable arrow through the reference point of the coordinatograph. Thus the position of a point on a track and the direction of the track at that point may be measured in one operation.

b) A normal keyboard to get necessary alpha-nu-

meric information into the computer. Some of the special characters are used as «functional keys» to communicate with the programme, e.g. saying that the last measurement was wrong, forget it.

c) Messages from the computer to the operator are written on a character display. The operator may also ask for certain results to be written on the display. However, the bulk result output is written on a lineprinter and on magnetic tape.

d) It is very annoying to do measurements without any response from the computer. It is also very tiring to read too many messages on the display. Each table has a loudspeaker giving tones of 4 different frequencies. The loudspeaker acts as a programmable output device giving a short and gentle «pip» after each input that the computer has accepted and a less gentle sound if an error condition has occurred. Details may then be found on the display.

Normally results are written on disk, and an off-line programme sorts and copies results from disk to magnetic tape once per day. The magnetic tapes are used as input to the large UNIVAC 1110 for further analysis.

There are three groups of programmes in the computer simultaneously, each having a set of interrupt levels associated with it. These are:

The system, which handles memory protect, interrupts, all I/O and some other features.
The Plate scanner programme.
The Gargamelle programme.

Apart from I/O (which is considered as a part of the system) the Gargamelle programme consists of two subgroups «fast processing» at interrupt level 4 and slow processing of interrupt level 1. I/O is done at interrupt level 9.

Fast processing takes care of input data and puts them into a data-structure, so that they are not lost. Appropriate markers are set to signal which type of data has come and where to find them. If the data structure is full, a «garbage collection» hos to be done quickly in order to find space for the new data. Thus fast processing mainly involves manipulation with a dynamic data structure rather than complicated arithmetic.

Slow processing routines take care of most of the calculation that can be interrupted. The routines naturally work on the same data structure as fast processing. For slow processing, an overlay structure of the programmes is used, having a complete set of the routines on disk and only those actually used in core at a specific time. As it works now points are reconstructed in three dimensions but no curves are fitted. The reconstructed points are mainly vertices of tracks, and they are used to check and guide the operator in finding and measuring the right topology. A point match routine is able to sort out the correct space points from random measurements of these points on the films, and furthermore to request any measurements which may be missing.

Logically the Gargamelle programme is divided into 3 parts:

The routines handling the data structure
The steering programme routines
Several processors performing special tasks.

The data structure is dynamic and common to all measurement tables. This considerably reduces necessary core space, but requires some extra CPU-time for routines administrating the data structure. This data structure corresponds to the HYDRA-structure developed at CERN which is accepted by most labs as standard, thus facilitating intercommunication between these labs.

The steering routines normally do nothing except a few tests and a set of calls to processors. However, the selection of processors determines the whole flow, and by keeping these routines small, it is very simple, even for a physicist to change the whole philosophy of dataevaluation just by changing the steering routine.

The processors do the different operations and calculations; some of them are among the fast processing routines, others are among the slow ones.
We have mentioned that some processors are needed purely for adminstration of the data structure. These are not called by the steering but by other processors working on the data structure. As examples of processors called by the steering, we mention:

a) Calculate transformation coefficients from coordinates of coordinatographs to coordinates of a reference plane based on measurement of fiducial marks with know positions in the reference plane.

b) Calculate a light-ray in the chamber based on the measurement of a single point in one view.

c) Check if two such light-rays from different views intersect in the chamber and if so, calculate the coordinates (x, y, z) of the intersection point.
d) Given a point in space calculate apparent positions of this point in a given view.
e) Do point-match, i.e. make a table of indices showing which measurements come from which points in space and how many such points we have.
f) Finding out if any measurements are missing and asking for more measurements.
g) Asking for measurements along tracks and preparing output data as input to the next programme.

These are only some examples of logical processors. Clearly each of them has several subroutines, and some of these are common to several processors. This is the reason that the technical division in processors it not always identical to that mentioned here, but the general philosophy is the same.
Some further remarks on the steerings should be made. Two main philosophies have been proposed:

a) To give the initiative to the computer, giving messages to the operator what to do step by step.
b) To give the initiative to the operator, to measure what he thinks is essential in every picture and ask the computer to do specific calculations and check for consistency and completeness.

In the first case the steering routine follows a fixed sequence of tests and calls. In the latter case the call is made by a jump depending on the type of input data.
In Bergen we have succeeded in combining the two philosophies having a) as a rule but allowing to use b) in a certain period at the beginning of each event.
The aim of this article has been to show wat our problems are and how they arise from experiments in high-energy physics. Further to tell which hardware components we have at our disposal to solve these problems and finally to explain the main philosophy of the organization of programmes and data in the NORD-1. No attempt has been made to explain the special mathematical formalism used to get the desired results. Instead we may conclude by stating that the general system (with some modifications) may have interest far beyond the domain of high-energy physics. Institutes of geography and of public area planning have already shown interest in our apparatus. And many more applications may exist or be proposed.

A Note on GPM with a practical example

BY TOROLF PAULSEN AND STIG NYBERG, A/S NORSK DATA-ELEKTRONIKK

In the Computer Journal, October 1965, C. Strachey described a macrogenerator called GPM (General Purpose Macrogenerator). GPM was originally planned to help write a compiler for the language CPL. The idea was to write the whole compiler as a set of macro calls.
In this way, one got a machine-independent compiler. By redefining the macros, a compiler for another machine could be produced, and by rewriting GPM, one could generate the compiler on another machine than the target machine.
GPM is referenced in most of the literature dealing with macro processors. We refer to two of these books:

Higman, Bryan: A Comparative Study of Programming Languages. MacDonald, London, 1967, pp. 55—64.
Wegner, Peter: Programming Languages, Information Structures, and Machine Organization. McGraw-Hill Book Co., New York, 1968.

Input to GPM is a character string. If there are no macro calls in the string, the input is simply copied to the output. When a macro call occurs, the macro is evaluated, and the result is copied to the output.
A macro call consists of a macro name and a list of the actual arguments, each separated by a comma. The name is preceded by an ↑ and the last argument is terminated by;, for instance,
↑NAME, ARG1, ARG2; ↑NOARGS;
Six characters have special meaning in GPM:

↑ Precedes macro calls
; Ends macro calls
, Separates arguments in a macro call
\ Means argument No. Followed by one character in the set 0-9, A-Z
< Start quote. This is a string parenthesis
> End quote. <and> must always occur in pairs. An unmatched >means stop.

Before a macro can be evaluated, it must have been defined. This is done by calling the macro DEF, which is one of the ten macros built into GPM. DEF should be called with two arguments, the name and the value of the defined macro:
↑DEF, NAME, VALUE;
↑NAME; will then yield VALUE.
The value of the macro may contain \ 1, \2, . . . etc. When the macro is evaluated, \1 is replaced by a direct copy of the first argument o fthe call, \ 2 by the second, and so on.

↑DEF,A,<A \ 1A>;	The value of A is A / 1A
↑A,C;	Yields ACA
↑A,ACA;	Yields AACAA
↑A,↑A,C;;	Yields AACAA.

Enclosing any string in quotes has the effect of preventing evaluation of any macro calls inside. One layer of string quotes is removed. By using string quotes it is possible to output any character except an unmatched string quote.

How GPM Operates

The input string is scanned from left to right and copied to the output until a macro call is encountered. The macro call is evaluated as follows:

a) The macro name and the arguments are evaluated in sequence from left to right. This process involves in its turn evaluating any macro calls which occur so that the whole process of evaluating is a recursive one.
b) When the argument list is complete (the terminating semicolon is reached), the current list of definitions is scanned in reverse order (from the most recent addition towards the oldest entry) in search of the name of the macro now being evaluated. The scanning stops at the first entry with the correct name, so that the most recent definition is used.
c) The symbol string corresponding to the macro name (the macro's value) is now scanned in the same way as the original input string, except that occurences of \ 1, \ 2, . . etc. are replaced by exact copies of the corresponding actual argument. \0 means the macro name. If an argument asked for is not supplied, NIL is taken as actual argument.
d) On reaching the end of the defining string the argument list (macro name and actual arguments) is lost. Any defininitions added to the definition list in the course of the macro evaluation are lost (temporary definition).
e) Scanning of the input string is resumed.

In this version of GPM there are ten macros built into GPM itself. These are:

DEF	Used to define other macros. It takes two arguments, the name of the new macro, and its value (the value is often enclosed in string quotes).

VAL — Gives the value of a macro. It takes one argument, the name of the macro.

UPDATE — Used to update macro definitions. Works in the same way as DEF. The new value must not be longer than the old value.

BAR — Performs binary arithmetic. Takes three arguments. The first must be +, —, *, /, or R, which mean add, subtract, multiply, (integer) divide, and remainder (after integer division). The second and third arguments are two binary numbers.

DECBIN — Four macros to perform conversion from decimal to binary,

BINDEC — binary to decimal,

OCTBIN — octal to binary, and

BINOCT — binary to octal.

HD — Gives first character of first argument.
↑HD, ABC; yields A.

TL — Gives the rest of the first argument
↑TL, ABC; yields BC.

Other Examples
The macro SUC is defined by:

```
↑DEF,SUC,<↑1,2,3,4,5,6,7,8,9,10,↑DEF,1,<\>\
1;;\>;
```

The effect of ↑SUC,3; is to make a temporary definition of a macro with name 1 and value\3, and then to call it with arguments 2, 3, ..., 10. The result is 4. The conditional macro COND is defined by:

```
↑DEF,COND,<↑\1,↑DEF,\1,NOT FIRST;↑DEF,A,
FIRST;;>;
↑COND,A;    yields FIRST
↑COND,B;    yields NOT FIRST
```

This is obtained by making use of the fact that only the later of two definitions of the same macro is used. If COND is called with A, then A is temporarily defined twice, and the last definition is used.

If COND is called with B, then B is defined as NOT FIRST, and A is defined as FIRST. This yields NOT FIRST, since B is called.

The macro SUM is defined by:

```
↑DEF,SUM,<↑BINDEC,↑BAR,+,↑DECBIN,\1;,
↑DECBIN,\2;;;>;
```

SUM adds two decimal numbers. ↑SUM, 123,45; yields 168.

```
↑DECBIN,\1;    Converts 123 from decimal to binary
↑DECBIN,\2;    Converts  45 from decimal to binary
↑BAR,+,. . . .; Adds the two binary numbers
↑BINDEC,. . . .; Converts the result from binary
                 to decimal.
```

The recursive macro FAC is defined by:

```
↑DEF,FAC,<↑\1,↑DEF,\1,<↑PROD,>\1
<,↑FAC,↑DIF,>\1<,1;;;>;↑DEF,0,1;;>;
```

↑FAC, 0; yields 1, ↑FAC, 2; yields 2,
↑FAC, 4; yields 24, and so on.

The macro PROD yields the product of two decimal numbers, and DIF the difference between two such numbers. PROD and DIF are very similar to SUM described above.

↑FAC, 3; works in the following way:

The macro 3 is called. Then 3 is defined as the product of 3 and ↑FAC, 2; and 0 is defined as 1. After this, FAC is called again with 2 as argument. This goes on until FAC is called with 0. This yields 1, since 0 is defined (temporarily!) twice, and the most recent difinition is taken.

A slightly enlarged version of GPM is implemented on NORD-1. This takes approximately 1300 words (decimal), and in addition a stack is necessary. The upper limit of this stack may be chosen freely.

GPM is a very nice program to play with. It is quite powerful, in spite of its small size (conditional expressions, recursive definitions, etc.). Another nice thing is that it is independent of other processors. It can be used to generate FORTRAN programs, as well as assembly programs, or whatever you like.

However, programs written in GPM tend to be very obscure, and are not easy to read.

A GPM Application

NORDCOM is a colour display system for graphic/alphanumeric pictures. The software part consists of a «Data Transform» part which reads the picture data and converts them into binary form, and a «Picture Generator» part which generates the picture into an external memory from which it is automatically refreshed on a screen. The screen contains 384 x 256 points. Characters are given by 6 x 9 dot matrices, which include the space necessary to distinguish them from adjacent characters. Special symbols consisting of one or more 6 x 9 dot matrices may also be displayed. These may be symbols for transistor, valve, switch, pump, or whatever the customer may require for his pictures.

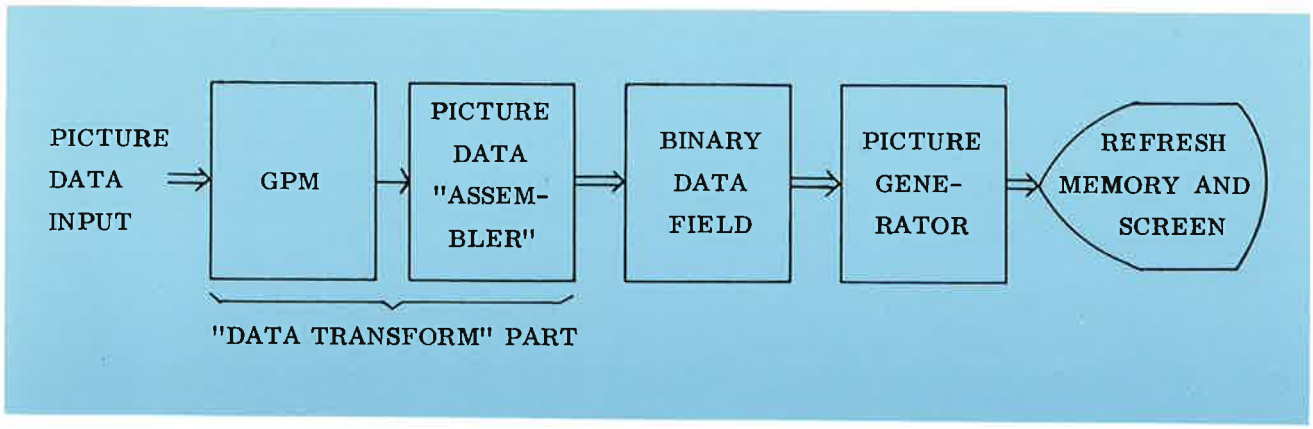
The «Data Transform» part includes a simple 2-pass picture data «assembler». This «assembler» accepts the following data types:

1. Line segments. They are given by their pair of end points (X_1, Y_1), (X_2, Y_2) and colour this way:
   *X_1 Y_1 X_2 Y_2 colour
   The asterisk indicates a line segment.
2. Character strings, given this way:
   α Line, column, colour, text
   The position (line, column) refers to the first character in the text.
3. Special symbols. Since these symbols are not found on the keyboard of an ordinary card punch, they have to be given by a group of text strings where each character corresponds to a 6 x 9 dot matrix on the screen. If, for instance, a transistor sybmol has an A as its corresponding character, the transistor is given by this string:
   However, since special symbols may cover more
   # line, column, colour, A
   than one character line, such symbols must be given by a group of such strings.

The picture data «assembler» also accepts certain control characters telling picture type, identification number, end-of-input, etc.

A GPM macroprocessor in front of this picture «assembler» will, if supplied by proper system macros, offer a number of facilities:

a) Special symbols may be called by name, rather than by the corresponding character strings, even if the symbol covers more than one character line. Example:
   A symbol called TRAFO is called in this way:
   ↑TRAFO, line, column, colour;
b) Control characters will be more readable. End-of-input to the picture assembler may, for instance, be given by the macro call:
   ↑END;
c) Character addressing may also be used for line segments, since GPM is able to compute the corresponding coordinates. This is useful when line segments mainly are connectors between special symbols, or rough block diagrams are drawn. A horizontal connector may be given by the macro call:
   ↑HC, line, column$_1$, column$_2$, colour;
   Example: ↑HC, 17, 47, 51, 0; is transformed by GPM to: *287 97 311 97 0.

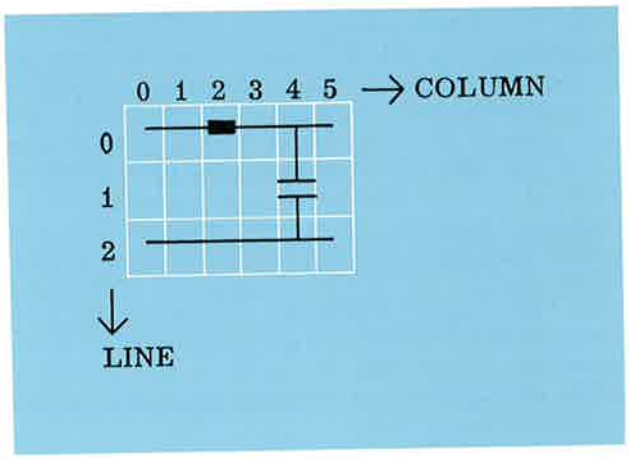


d) The user may define his own picture parts, and call them by name. In order to ease such definitions some system macros are given to make the picture parts relocatable.
Example: Given this simple RC circuit, drawn at the topmost, leftmost corner of the screen:
The user wants to have this circuit drawn at different positions on a number of pictures. He therefore wants a name for this picture part. He chooses the name LOWPASS, and writes the following macro definition:

```
↑DEF, LOWPASS,
↑RH, 0, 0, 5, 1;
↑RH, 2, 0, 5, 1;
↑RV, 0, 2, 4, 1;
↑RS, RESISTOR, 0, 2, 3;
↑RS, CONDENSER, 1, 4, 2;
;
```

RH, RV, and RS are NORDCOM system macro names that denote relocatable horizontal, relocatable vertical, and relocatable symbol respectively. The call ↑LOWPASS, line, column; yields the data for the RC circuit with the given displacement.
Such defined picture parts may also contain elements that will remain static, independent of displacement given in the call.

e) GPM offers the user a certain flexibility in forming his own system, since he may include additional NORDCOM system macros.



To implement the NORDCOM «Data Transform» program, the following must be done:
The special symbols used by the picture generator must be given names by means of some NORDCOM system macros. This simple job will be done once and for all.
The user may then include his own NORDCOM system macros, and picture part macros. These should be read by GPM before the picture data is read.

– we want bits of the future

A/S NORSK DATA-ELEKTRONIKK ØKERNVEIEN 145 OSLO 5 NORWAY PHONE: 217371 TELEX: 18284

Otto Falch - Oslo