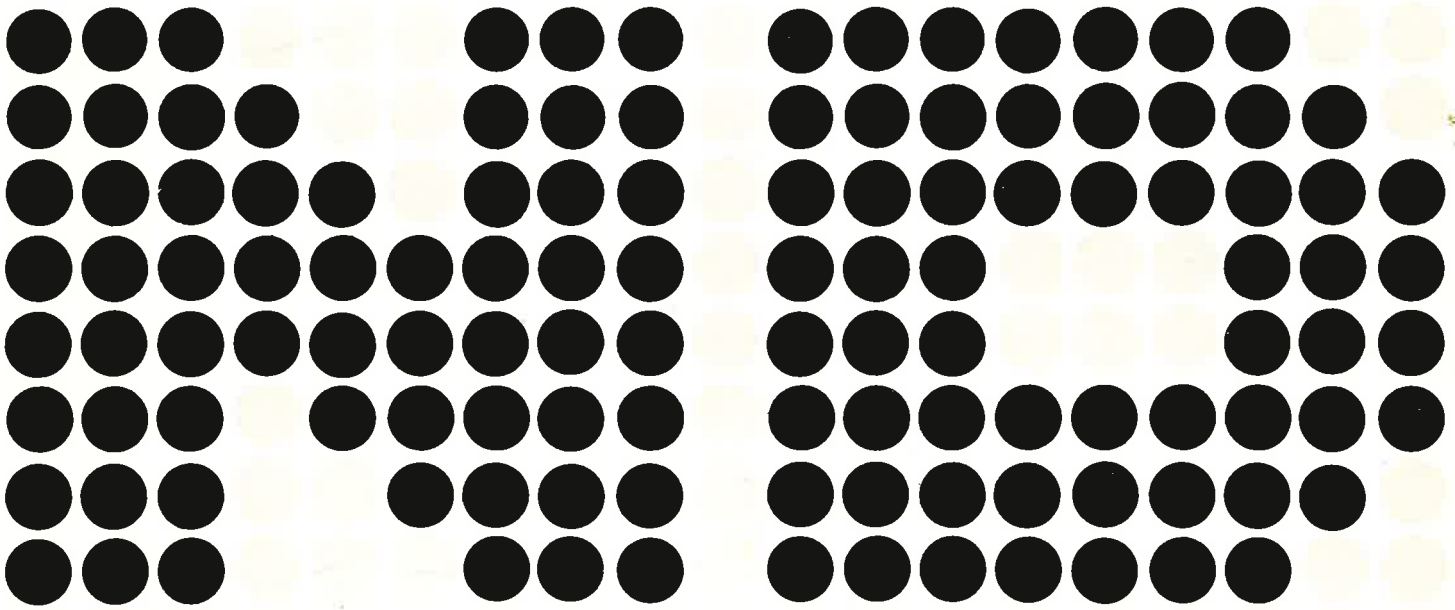
NORD PROCESS I/O

Software Guide

NORSK DATA A.S



NORD PROCESS I/O

Software Guide

REVISION RECORD

Revision	Notes
09/77	Original Printing
10/78	Revision A: Added Section 6.1 to 6.1.5 (new)

NORD Process I/O – Software Guide
Publication No. ND-60.093.01

NORSK DATA A.S.

Lørenveien 57, Postboks 163 Økern, Oslo 5, Norway

TABLE OF CONTENTS

+ + +

1 INTRODUCTION

1.1 *ANALOG AND DIGITAL I/O*

The process input/output is usually separated into two categories: the "analog" I/O, and the "digital" I/O. Of course, since our computer is a digital computer, all data given to it must be "digitalized" and represented by numbers. "Analog" I/O is applied to process values which, from the computers point of view, may vary more or less continuously and therefore must be represented by integers or floating point numbers. (Typical analog values may represent: pressures, temperatures, water levels, volumes, etc.)

The "digital" process values, however, consist of individual bits. These bits are normally grouped together so that one IOX instruction reads 8, 12 or 16 digital input values. (Digital process values normally represent the position of switches, alarm on/off conditions, etc.)

The procedure for reading analog input values is normally this: first, select value. This is done by giving its corresponding channel number to a *multiplexer*. The analog-to-digital conversion may now start and the value is found by reading the *AD converter*. There is always a certain delay from the time the channel has been selected until the converted value may be read. For solid-state multiplexers this delay usually is so short that an interrupt controlled input operation is not to be recommended. For relay multiplexers the delay is much longer.

The raw ADC value (the value read from the AD converter) must be further transformed to a normal integer or floating point number.

Analog output is usually performed in a way quite similiar to the analog input operation.

Normally, two possible methods of reading digital input exist:

1. Periodical scanning of the digital input status, while searching for changes-of-state.

2. Input is read by a driver which is triggered by interrupts.

   An interrupt is given when hardware discovers a difference between the input status and the content of a compare register ("match" register). The driver then identifies the change, and loads the match register with its new content.

Digital output is usually performed by setting a sequence of bits by means of one IOX instruction. Since it often occurs that only one bit is to be changed, the current digital output status should be tabulated so that the remaining bits can be set to the state they had before.

1.2 *SOME REMARKS ON SINTRAN III I/O*

If an I/O device for process control is interrupt operated by SINTRAN III, the communication with the user program will normally be accomplished by one of two alternatives, which are mutually exclusive:

1. The RT program may read/write by means of INBT/OUTBT (MON 1 and MON 2).

2. The RT program may be connected to an input interrupt by means of the monitor call CONCT (MON 106). The RT program will then be started every time an interrupt from the specified device has been given. The RT program must read its input data by means of IOX instructions or special monitor calls.

   It is not possible to connect RT programs to output interrupts.

Devices may also be operated by the IOSET monitor call (MON 141). A device dependent subroutine (SETDV subroutine), whose address is found in the SINTRAN III device data field, will then be called and may give a return value which will be given to the user program. IOSET always gives immediate return, and there can be no waiting for interrupt inside a SETDV subroutine. The formats of the monitor calls mentioned in this chapter are:

INBT:

```
LDT     LDEV     % T: = logical device number
MON     1        % INBT
JMP     ERROR    % Error return
                 % Data in A register
  .
  .
  .
```

OUTBT:

```
LDA     DATA     % A: = Output data
LDT     LDEV     % T: = logical device number
MON     2        % OUTBT
JMP     ERROR    % Error-return
  .              % Call executed
  .
  .
```

IOSET:

```
        LDA     (PAR        % 'A' points to parameter list
        MON     141         % IOSET
                .           % A register normally > = 0 if o.k.

PAR,    (LOGDV              % LOGDV = logical device number
        (0                  % 0 means input, 1 means output
        (PROG               % RT program occupying the device
        (CONTR              % control information
)FILL
```

CONCT:

```
        LDA     (PAR        % 'A' points to parameter list
        MON     106         % CONCT
                .
                .
PAR,    (PROG               % RT program "prog" will be connected
        (LOGDV              % LOGDV is the logical device number
)FILL
```

Before the call of INBT/OUTBT/IOSET the device must be reserved by either RESRV (MON 122) or PRSRV (MON 124). (Refer to the manual, SINTRAN III – User's Guide.)

For high-speed I/O it may sometimes be convenient for the user program to execute its own IOX instructions. Such programs must be loaded on a segment with protection ring 2 specified. This usually implies that all other segments which are linked to it, must also be loaded on ring 2. A program on ring 2 may ruin the entire software system by executing unmotivated, "dangerous" instructions (IOF, POF, etc.), which are all "legal" on ring 2. However, jump instructions cannot destroy the operating system since the paging system protects it (provided the RT program has not been loaded on page table 0, of course).

2 ND-820 AD-CONVERTER

2.1 *PROGRAMMING SPECIFICATIONS*

The ND-820 Analog to Digital converter has the following hardware specifications:

Resolution:	12 bits
Number of inputs:	8 differential
Input voltage:	± 10 V
Converted data:	2's complement
Input filter:	100 ms time constant
Conversion time for A/D:	25 μs
Total data acquisition time:	35 μs
Input voltage protection:	100 V (continuously)
Standard physical device no.:	$1440_8 - 1477_8$
Standard ident codes:	$101_8 - 110_8$

The AD converter is operated by four IOX instructions.

IOX	DEV	Read data to A register bit 0-11. Bit 11 is the most significant bit for an integer given in 2's complement.
IOX	DEV + 1	Set multiplexer channel.
IOX	DEV + 2	Read device status. A register bit: 0 – interrupt enabled 2 – busy 5 – EXTERNAL interrupt
IOX	DEV + 3	Write control word. A register bit: 0 – enable interrupt 2 – start conversion 4 – Master Clear 5 – enable EXTERNAL interrupt 6 – enable EXTERNAL start of conversion

Both bit 0 and bit 5 must be set if EXTERNAL interrupt is to be enabled.

Programming Example:

```
% Input without interrupt.
        .
        .
LDA         CHANO           % A: = channel number
IOX         DEV + 1
SAA         4
IOX         DEV + 3         % start conversion
SAX         –40             % set timer
IOX         DEV + 2         % read status
BSKP        ZRO  20  DA     % conversion finished?
JNC     *   –2              % No. count and jump
JXZ         TMOUT           % Jump if time-out
IOX         DEV             % read data (12 bits)
        .
        .
        .
```

2.2 *STANDARD SUBROUTINE SAIRD*

A standard subroutine SAIRD callable from FORTRAN, has been made. It reads analog input by executing its own IOX instructions, and its parameter list is equal to that of the monitor call AIRDW (see Section 2.3.1).

The calling sequence is:

FORTRAN: CALL SAIRD (INUM, ILDAR, IRSAR, IVAL)

MAC assembly:

```
        )9BEG
        )9EXT  SAIRD
               .
               .
               LDA      (PARLI
               JPL   I  (SAIRD
               .
               .
               .
PARLI,  INUM; ILDAR; IRSAR; IVAL
INUM,   5
ILDAR,  0; 1; 2; 3; 4
IRSAR,  0; 0; 0; 0; 0
IVAL,   0
)FILL
               .
               .
)9END
```

Execution time: 45 μs + 90 μs/channel. Extra overhead for FORTRAN call due to 8ENTR and 8LEAV.

2.3 *ND-820 UNDER SINTRAN III*

2.3.1 *The Monitor Call AIRDW*

A monitor call AIRDW (MON 37) has been implemented to read a number of channels. Since SINTRAN III will support up to 8 ND-820 AD converter cards, and each card has 8 channels, the channels are numbered 0-63 (0-7 for the first card, 8-15 for the second card, etc.). The format of the monitor call is:

```
                .
                .
        LDA     (PARL
        MON     37          % AIRDW
        LDA     IVAL
        AAA     -1
        JAF     ERR         % Jump if error
                .
                .
                .
PARL,   INUM
        ILDAR
        IRSAR
        IVAL
INUM,   5
ILDAR,  0; 1; 2; 3; 4
IRSAR,  0; 0; 0; 0; 0
IVAL,   0
```

Description of the parameters:

INUM: Number of channels to be read.

ILDAR: Array of channel numbers. Number of elements corresponds to INUM. Channel numbers have values in the octal interval [0, 77].

IRSAR: Array where the values will be stored by AIRDW. Bits 0-15 give the value in the two's complement. (The 12 input bits are sign extended.) The number of elements in IRSAR corresponds to INUM.

IVAL: Error indicator.

IVAL = 1: No errors
IVAL = 2: Operation incomplete
IVAL = 3: Error conditions in monitor call

Monitor call execution time: 1 ms + 0.3 ms/channel.

2.3.2 *EXTERNAL Activation*

The analog-to-digital converter may be activated by an EXTERNAL signal. To use this function the AD converter must be initialized by means of the monitor call IOSET.

Control information in the call of IOSET:

–1	Master Clear
0	Master Clear
1	Enable EXTERNAL interrupt
2	Enable EXTERNAL start of conversion, interrupt when conversion is finished.

In these modes (1 or 2) an AD converter card will have to be reserved for only one RT routine. It is the user's responsibility to do this.

Note that a call of AIRDW results in a setting of the control word that disables all EXTERNAL triggering.

2.3.2.1 EXTERNAL Interrupt

When initializing the AD converter with the control information equal to 1, the recommended way of treating the AD converter is: use the monitor call CONCT to connect an RT program to the logical device number (one device number for each AD converter card). When an interrupt is forced by the EXTERNAL signal, the connected RT program is activated, and this program may now access the AD converter by means of the monitor call AIRDW or the subroutine SAIRD.

2.3.2.2 EXTERNAL Start of Conversion, Interrupt when Finished

When initializing the AD converting with the control information equal to 2, the recommended way of handling the AD converter is: use the monitor call CONCT to connect an RT program to the wanted logical device number. Load the channel number for the actual card, using IOX DEV + 1. When the conversion, started by the EXTERNAL signal, is finished and an interrupt is given, the connected RT program is activated and this program may now read the converted data with an IOX DEV instruction. This RT program must be loaded on ring 2.

% EXTERNAL INTERRUPT

```
        LDA     (PARC
        MON     106         % CONCT.
        LDA     (PARR
        MON     122         % Reserve card
LOOP,   LDA     (PARI
        MON     141         % IOSET
        MON     135         % RTWT
        LDA     (PARA
        MON     37          % Read channels
        JMP     LOOP
```

% EXTERNAL START OF CONVERSION

```
        LDA     (PARC
        MON     106         % CONCT.
        LDA     (PARR
        MON     122         % Reserve card
        LDA     (PARI
        MON     141         % IOSET
LOOP,   LDA     CHANO
        IOX     DEV + 1     % Set channel number
        MON     135         % RTWT
        IOX     DEV         % Read data
        STA     DATA
        JMP     LOOP
```

3 NORATOM – NORCONTROL PROCESS EQUIPMENT

3.1 *N-NN-229 INTERRUPT CONTROL CARD*

N-NN-229 is a control module for up to 8 process interrupts to NORD-10. The eight interrupts are separated into two groups, A and B. Both groups operate on interrupt level 10, 11, 12 or 15. They need not necessarily operate on the same level. Each of the eight process interrupts has its own IDENT code in the interval 177700_8 - 177777_8. (The least significant digit corresponds to channel number 0-7 on the card. The next digit depends on the backwiring.)

In small systems, the process interrupts may come directly from the digital inputs. In greater systems it will be more reasonable to have one interrupt signal for each group of 16 or 32 inputs. In the latter case, the whole group has to be read to find out where change-of-state has occurred.

3.1.1 *Programming Specifications*

Three of the four standard physical device numbers are used:

IOX	DEV + 1	Write data
IOX	DEV + 2	Read status
IOX	DEV + 3	Write control word

IOX DEV + 3 enables or disables the different interrupt channels (0-7) of the module. The A register holds the channel number (0-7) in bit 0-2. Bit 3 = 1 enables the channel, bit 3 = 0 disables it. "Write control word" for one channel has no influence on the other channels.

IOX DEV + 2 reads a bit-mask of eight bits that tells which of the interrupt channels of the module are enabled. If bit = 1, the channel number corresponding to the bit number is enabled.

IOX DEV + 1 may generate programmed interrupts. Bit 0-2 in the A register gives one of the channels 0-7, and an interrupt is generated if the level has been enabled.

```
% In this example, channel number 0 is associated with digital input word,
% which may be read by an "IOX DIDEV' instruction.

% Initiate Driver System

INIT,   IOF
        IOX     DIDEV     % Read input
        STA     LASDI     % Save it for the driver
        LDA     CHANO
        BSET    ONE 30 DA
        IOX     DEV +3    % Enable interrupt
        ION
        .
        .
        .

CHANO,  0                 % Channel number

% Input Driver on Level 12

WT12,   WAIT
LEV12,  IDENT   PL12
        .
        .
        IOX     DIDEV     % Read input
        LDX     LASDI
        STA     LASDI
        REXO    SA DX
        JXZ     NOCHA     % Jump if no change found
        .
        .
        .

NOCHA,  LDA     CHANO
        BSET    ONE 30 DA
        IOX     DEV +3
        JMP     WT12
```

3.2 *DIGITAL INPUT UNDER SINTRAN III*

Initially, the user program decides which digital input cards give interrupt and which cards do not. A monitor call NCMON (MON 175) gives this information to SINTRAN III. (The interrupts are assumed to be generated by the NN-NN-229 card, or a module with similar programming specifications.)

The format of NCMON is:

```
        LDA     (PARL
        MON     175
        JAF     ERR
        .
        .
PARL,   LDEV
        FUNC

LDEV,   LOGDV           % Logical division of card
FUNC,   0               % '0' means 'not interrupt'
```

The first parameter is the logical device number of the card and the second is the interrupt flag. If unequal to 0, interrupt-controlled input is assumed.

The monitor call gives an A register value in return. A = 0 if no error has been found.

Digital input (16 bits) may be read by IOSET, if "control information" is unequal to −1. Interrupt is not waited for, no matter which mode has been set by NCMON. If IOSET is called with "control information" equal to −1, this means "device clear". If the device is in interrupt mode, the SINTRAN III input buffer will be cleared and interrupt will be disabled for the corresponding channel. "Mode of operation", set by NCMON is not affected. If device is not in interrupt mode, nothing is done.

Digital input may also be read by calling INBT. If the card is in interrupt mode, a change-of-state will be waited for. If not in interrupt mode, the 16-bit status will be read and "return" is immediately given.

% Digital Input by means of 'IOSET'

```
        LDA     (PARR
        MON     122         % Reserve
        .
        .
        LDA     (PARI
        MON     141         % IOSET
        .                   % 16-bit status in A register
        .
```

```
PARR,   (LOGDV
        (0
        (0
PARI,   (LOGDV
        (0
        (0
        (0
)FILL
```

% Digital input by means of 'INBT'

```
        LDA     (PARR
        MON     122         % Reserve
        .
        .
        LDT     (LOGDV
        MON     1           % INBT
        JMP     ERR
        .                   % 16-bit status in A register
        .
```

3.3 *DIGITAL OUTPUT*

3.3.1 *Programming Specifications*

A 16-bit digital output status is simply set by executing an IOX instruction with the proper, physical device number and the output status in the A register.

3.3.2 *Digital Output under SINTRAN III*

A 16-bit digital output status may be set by either IOSET or OUTBT. Immediate "return" from both.

Examples:

% Output by means of 'IOSET'

```
        LDA     (PARR
        MON     122         % Reserve
        .
        .
        LDA     (PARI
        MON     141         % IOSET
        JAN     ERR         % Error if A register negative
        .
        .
PARR,   (LOGDV
        (1
        (0
PARI,   (LOGDV
        (1
        (0
        OSTAT
OSTAT,  177401              % the 16-bit output status
)FILL
```

% Output by means of 'OUTBT'.

```
        LDA     (PARR
        MON     122         % Reserve
        .
        .
        LDT     (LOGDV
        LDA     OSTAT       % A: = Output status
        MON     2
        JMP     ERR         % Error return
        .
        .
```

3.4 *ANALOG INPUT*

Programming specifications:

IOX	DEV	Read input value (12 bits)
IOX	DEV + 1	Set multiplexer channel.

Conversion time: 40 μs

Bit 15 = 1 in the channel number gives a delay of 15 μs before conversion starts. Bit 15 = 0 gives a delay of 100 μs. Conversion is finished when bit 13 in the input data word is equal to zero.

Programming example:

```
        .
        .
        LDA     CHANO     % Channel number
        IOX     DEV + 1   % Set channel
        SAX     –200      % Set timer
        IOX     DEV       % Read data
        BSKP    ZRO 150 DA% conversion finished?
        JNC     * –2      % Number count
        JXC     TMOUT     % Jump if time-out
        .
        .
        .
```

Analog input may be read by monitor call NCAIN (MON 176, a user monitor call). It expects 3000_8 - 3003_8 as the physical device numbers of the AD converter but this may be changed by a simple patch. Called with multiplexer channel as parameter. Conversion time is specified by bit 15. Return with: twelve-bit analog input in the A register (or –1 in the A register, which means conversion time-out).

```
        .
        .
        LDA     (PARI
        MON     176
        JAN     ERR
        .
        .
PARI,   CHNO
CHNO,   0                 % Channel number
```

3.5 *ANALOG OUTPUT*

Output operation is performed simply by executing the proper IOX instruction with the output value in the A register.

Standard physical device numbers:

Analog output may also be set by the SINTRAN III monitor call NCAOU (MON 177, a user monitor call).

Each output channel has its own logical device number 400_8 - 477_8.

Parameters in call:

Logical device number for channel, and output value.

Return with:

```
A = 0                   if ok
A = –1                  if illegal parameters in call

        LDA     (PARO
        MON     177
        .
        .

PARO,   LDEV
        VALUE
LDEV,   420             % Logical device number
VALUE,  0               % Output value
```

4 DIGITAL I/O MONITOR CALLS DIW AND DOLW

4.1 *GENERAL REMARKS*

Two SINTRAN III monitor calls, DIW (MON 165) and DOLW (MON 166), have been made to operate on several digital registers in the same monitor call.

DIW reads the present digital status for the selected group of input registers. DOLW sets the specified digital output for the selected group of output registers. DIW and DOLW accept logical device numbers in the interval 400_8 - 417_8. These devices need not be reserved before call of DIW or DOLW. DIW and DOLW always give immediate return, since the I/O operations are not controlled by interrupts. Execution times: 1.3 + 0.4 ms/register for DIW, 1.5 ms + 0.5 ms/register for DOLW.

4.2 *DIW*

The format of the call of DIW is:

FORTRAN call: CALL DIW (INUM, ILDAR, IRSAR, IVAL)

MAC Assembly:

```
        LDA (PARI
        MON 165
        .
        .
PARLI,  INUM
        ILDAR
        IRSAR
        IVAL
INUM,   10
IVAL,   0
ILDAR,  400; 401; 402; 403; 410; 411; 412; 413
IRSAR,  0; 0; 0; 0; 0; 0; 0; 0
```

In this example, eight input registers are read:

Description of the parameters for DIW:

INUM: Number of input registers to be read. INUM corresponds to the number of elements in the arrays ILDAR and IRSAR.

ILDAR: Integer array of logical device number of the input registers to be read.

IRSAR: Integer array where the read data values will be stored.

IVAL: Error indicator. IVAL = 1 if okay. If unequal to 1, there is probably an error in the parameter list.

4.3 *DOLW*

DOLW operates on the specified bits within a selected group of digital output registers. Only the bits specified are changed, the rest of the digital output status is fetched from the device data fields. The digital output is inverted if the bit 5INVRT (bit 7) in the TYPRING word in a device data field is set.

The format of the call of DOLW is:

FORTRAN call: CALL DOLW (INUM, ILDAR, IVARR, IMARR, IVAL)

MAC Assembly:

```
        LDA (PARLI
        MON 166
        .
        .
PARLI,  INUM
        ILDAR
        IVARR
        IMARR
        IVAL
INUM,   5
IVAL,   0
ILDAR,  400; 401; 402; 414; 415
IVARR;  1777; 0; 177777; 376; 4
IMARR,  3777; 0; –1; 777; 4
```

In this example, DOLW operates on five digital output registers.

Description of the parameters for DOLW:

INUM: Number of output registers to be operated. INUM corresponds to the number of elements in the arrays ILDAR, IVARR and IMARR.

ILDAR: Integer array of logical device numbers of the output registers to be set.

IVARR: Integer array of digital output. The order of elements in IVARR corresponds to the order of elements in ILDAR and IMARR.

IMARR: Integer array of bit masks, indicating which bits will be affected by the DOLW call. Bit equal to "one" indicates that the digital output bit will be set according to the value given by IVARR. The remaining bits will be set according to values found in the corresponding device data fields, which are updated by the DOLW call. Please ensure that these data fields always hold the current digital output status by, for instance, always using DOLW.

IVAL: Error indicator. IVAL = 1 if okay. If unequal to 1, there is probably an error in the parameter list.

5 ND-810 DIGITAL INPUT

5.1 *PROGRAMMING SPECIFICATIONS*

The ND-810 digital input system gives a 12-bit input. Interrupts resulting from digital input change, may be programmed. (These interrupts are generated when the digital input status differs from the content of a match register, and interrupt has been enabled by proper setting of the control word.)

IOX	DEV	Read data (12 bits)
IOX	DEV + 1	Write 12 bits into the match register.
IOX	DEV + 2	Read device status. Bit 0 = 1 means "interrupt enabled". Bit 3 = 1 means "interrupt wanted".
IOX	DEV + 3	Load control word. Bit 0 = 1 means "enable interrupt"; Bit 2 = 1 means "turn off interrupt wanted".

"Interrupt wanted" is the interrupt given when match register content is unequal to the digital input. A delay is available from unmatch if detected to interrupt is given.

Standard device number:	Extension group.
Standard interrupt level:	12
Standard ident number:	$200\text{-}377_8$ (extension)

5.2 *HOW TO IMPLEMENT IT*

If digital input *scanning* (instead of waiting for interrupt) is wanted, the digital input status is simply read by an IOX DEV instruction. The control word should initially be set to 4, to avoid interrupts.

When interrupt-triggered input is wanted, and a driver system must be made, the recommended way of programming is as follows:

```
% INITIATION PROGRAM

        SAA     4
        IOX     DEV + 3     % No interrupts wanted
        .
        .
        IOX     DEV         % Read input
        IOX     DEV + 1     % Set match register
        STA     MATCH
        SAA     5
        IOX     DEV + 3     % Enable Interrupt
        .
        .

% DRIVER ON LEVEL 12

W12,    WAIT
LEV12,  IDENT   PL12        % Identify Interrupt
        .
        .
        .
        IOX     DEV         % Read new status
        IOX     DEV + 1     % Load match register with it
        LDT     MATCH
        STA     MATCH
        REXO    SA DT       % T:=Bit mask for changes
        SKP     IF DT UEQ 0
        JMP     PINDV       % No change found
        .
        .
        .
PINDV,  SAA     5
        IOX     DEV + 3     % Enable Interrupt
        JMP     WT12
```

5.3 *ND-810 UNDER SINTRAN III*

A driver system has been made in order to operate ND-810 under SINTRAN III. A 12-bit digital input is read by means of INBT (MON 1), or by IOSET. IOSET may also set the "mode of operation" for the input driver.

5.3.1 *The Use of IOSET*

The control information to IOSET is divided into two parts: bits 0-11, which is the DATA part, and bits 12-15 which is the FUNC part. DATA gives a mask that tells which bits the software system shall consider as significant, while FUNC gives the "mode of operation". However, control information = −1 gives "device clear" (as usual) and does not affect "mode of operation". If control information = 0 the digital input registers are read and returned to the user program, without modifications, and "mode of operation" is not affected.

Control Information:	Function:	Effect:
−1	–	Device buffer is cleared. If "mode of operation" is unequal to 0: match register is set equal to digital input register and interrupt is disabled.
0	–	Read 12-bit digital status without modifications. Immediate return.
10000_8 + DATA	1	Interrupt controlled input. INBT will return with a data byte every time a change-of-state occurs. If transition 0 → 1, the value is 60_8 + bit number. If transition 1 → 0, the value is 100_8 + bit number. The DATA part of the control information gives a bit mask for significant bits.
20000_8 + DATA	2	Same as for FUNC = 1, except for the fact that INBT only reads transitions 0 → 1.
30000_8 + DATA	3	Same as for FUNC = 1, except for the fact that INBT only reads transitions 1 → 0.

Control Information:	Function:	Effect:
40000_8 + DATA	4	INBT gives a 12-bit mask which gives all transitions 0 → 1 since the previous INBT call. This bit mask does not reflect the current status, since the opposite transition has no influence on it.
50000_8 + DATA	5	A call of INBT in this "mode of operation" gives the new 12-bit digital status, truncated by an AND operation with the DATA part of the "control information". Change-of-state is waited for.

5.3.2 *Programming Examples*

```
% INITIATION PROGRAM
          .
          .
          LDA     (PARR
          MON     122       % Reserve device
          LDA     (PARI1
          MON     141       % IOSET. Function = 1, all bits significant
          LDA     (PARI2
          MON     141       % IOSET. Clear, initiate match register.
          LDA     (PARR
          MON     123       % Release device.
          .
          .
          .
PARR,     (LOGDV
          (0
          (0
PARI1,    (LOGDV
          (0
          (0
          (17777
PARI2,    (LOGDV
          (0
          (0
          (-1
)FILL
```

```
% DIGITAL INPUT HANDLER
        .
        .
        LDA     (PARR
        MON     122         % Reserve device
        .
        .
        LDT     (LOGDV
        MON     1           % INBT. Wait for transition.
        JMP     ERROR
        .
        .
        .
        LDA     (PAR13
        MON     141         % IOSET. Read input (instantly)
        .
        .
        .

PAR13,  (LOGDV
        (0
        (0
        (0
)FILL
```

6 ND-811 DIGITAL OUTPUT

16-bit digital output may be set by means of the SINTRAN III monitor call IOSET. The function code IFUNC gives the new 16-bit status. The monitor call OUTBT may be applied to manipulate with individual bits. The byte in the A register indicates bit number and wanted transition.

Transition to "1":	Output byte = 60_8 + bit number.
Transition to "0":	Output byte = 100_8 + bit number.

Each ND-811 card of 16 bits has its own logical, SINTRAN III device number. A bit in the data fields indicates if output is to be inverted. This applies to both IOSET and OUTBT. Output operations on ND-811 are performed by loading the 16-bit status (or inverted status) into the A register and execute the proper IOX instruction.

6.1 *ND814 DIGITAL INPUT/ND815 DIGITAL OUTPUT*

6.1.1 *Short Hardware Description*

The 1158 DR11-C interface is a digital input/output module which does the same job in a NORD-10 system as the DR11-C does in a PDP-11 system.

In this case, two ND-1158's are used, one in input mode and the other in output mode. In the input mode, the card is callled ND814 and in the output mode it is called ND815. However, some of the parameters are improved on.

The 1158 is a 16-bit digital input/output module. Input or output mode is switch selectable. Interrupt level is 10 in output mode and 12 in input mode.

An external input signal "request" will generate the proper interrupt on a transition from a zero to a one logic level. The polarity of this signal is switch selectable. It is also possible to operate the interface without giving interrupts.

A data strobe is generated by the module. This has the meaning "New Data Ready" in output mode and "Data Transmitted" in input mode.

In addition to 16-bit input/output data transfer, two control outputs are supplied making it possible to control a peripheral device.

Data from output are held in a register clocked by the "Write Data" pulse. There is no register holding input data.

For a further hardware description see the hardware manual "1158 DR11-C Interface" – ND-12.015.01.

6.1.2 *Programming Specifications*

IOX DEV READ DATA (16 bits)

If the module is in input mode, the value on the input/-output terminals are copied into the A register.

If the module is output mode, a read-back of the "write data" occurs.

IOX DEV + 1 WRITE DATA (16 bits)

In input mode this is meaningless.

In output mode the 16 bits in the A register are copied into the data register on the module and will appear on the input/output terminals.

IOX DEV +2 READ STATUS

The status of the module is copied into the A register. Only bits 0, 3, 12, 13 and 14 are used. All other bits will become 0.

Bit 0: Interrupt enabled. A one in bit 3 (RFT) will result in an interrupt if this bit is one.

Bit 3: Ready for Transfer. A low to high transition has occurred on the REQUEST input. Cleared by "WRITE DATA" output and "READ DATA" in input.

Bit 12: Read back of control output CSR0 (control word bit 12). A high on terminal 57 will correspond to a 1.

Bit 13: Read back of control output CSR1 (control word bit 13). A high on terminal 59 will correspond to a 1.

Bit 14: Direct input of REQUEST line. A one on terminal 61 will give a 1 in this bit. Remember the selected polarity.

IOX DEV +3 WRITE CONTROL

Only bits 0, 4, 12, 13 and 14 are used. The other bits have no effect. The control word cannot be set to anything but 0 at the same time as device clear is executed.

Bit 0: Enable interrupt on request. Signal rising or falling edge switch selectable.

Bit 4: Device Clear. Clears all control and status bits, and also sets teh data output register to 0.

Bit 12: Control output CSR0. A one will give a high level on terminal 57.

Bit 13: Control output CSR1. A one will give a high level on terminal 59.

Bit 14: Set RFT flip-flop. If equal to 0, the RFT flip-flop is not affected.

HARDWARE DEVICE NUMBER/IDENT NUMBER/S-III LOGICAL NUMBER (Interrupt level 12 if input, level 10 if output).

HDEV INPUT	HDEV OUTPUT	IDENT	S-III DEV. ND
770- 773	774- 777	17	400
1000-1003	1004-1007	26	401
1010-1013	1014-1017	27	402
1020-1023	1024-1027	43	403
1040-1043	1044-1047	15	404
1050-1053	1054-1057	40	405
740- 743	744- 747	264	406
750- 753	754- 757	265	407
700- 703	704- 707	11	410
710- 713	714- 717	21	411

6.1.3 *ND814/ND815 under SINTRAN III*

A driver system has been made in order to operate ND814/ND815 under SINTRAN III. Interrupt controlled I/O of 16-bit digital data may be done by means of the standard interrupt handling and ring buffer system in SINTRAN III by use of INBT (MON 1) from assembly programs or INCH from FORTRAN programs. Similarily OUTBT (MON 2) and OUTCH are used for output.

Program controlled I/O (not using the interrupt system) may be done by IOSET. This applies both to the reading of 16-bit data and to the reading/-writing of the 2-bit control output on the interface.

IOSET is also used to set the "mode of operation" for the driver and interface.

Program controlled I/O of several digital registers at the same time may be done by using the monitor calls DIW (MON 165) and DOLW (MON 166) (see also Chapter 4).

6.1.4 *The Use of IOSET*

Control Information (octal):	Returned function value If not otherwise noted 0 = OK, -1 = Error in input parameter, –2 = wrong control code	Effect:
–1	–	Device clear is executed. Device buffer is cleared.
0-3	–	Write CSR0 - CSR1 (control outputs)
4	Value 0-3 of CRS0-CRS1	Read CSR0-CSR1.
5	–	Enable interrupt.
6	–	Disable interrupt
7	–	Set Ready-for-transfer-flip-flop
10	0 1	Read request line
11	0 1	Read Ready-for-transfer flip-flop.
12	–	Set "aristo-driver" in "cm" modus.
13	–	Set "Arito-driver" in "inch" modus
14	Value of input lines	Read input lines
Negative numbers except "–1"	–	Write bits 0-14 (15 bits) to output register/terminals. Not possible to set sign or all bits 0-14 to "1".

6.1.5 *Interfacing Aristogrid Digitizer Series 100*

The Aristogrid Digitiser has been interfaced through the ND814 digital input.

Due to the special handling required to serve this device, a special driver has been made.

Two ND814 digital input cards are required for each digitizing table. One for initiating a conversion of a XY coordinate pari and reading the result, and another for reading of 5 push-buttons on the cursor.

The "DELAG" modus word in the data field contains information for this special driver.

Bit 0 = "0" Read push-button
"1" give 50 μs "START CONVERSION" pulse
Collect 10 BCD digits
Convert to 1/10 millimeters
Convert to 1/100 inch

Bit 1 = "0" Result in inch
"1" Result in cm

SINTRAN III Logical Numbers for ARISTOGRID:

400, 402, 404 ..... XY coorindates of board 1, 2, 3
401, 403, 405 ..... Push-buttons of board 1, 2, 3.

Example:

Reading of XY coordinates

```
      PROGRAM RXY
      DOUBLE INTEGER   XY, DY
      DIMENSION        IX (2), IY (2)
      EQUIVALENCE      (DX, IX), (DY, IY)
      .
      .
      .
C
C     Clear device/Clear input buffer
C     IRW = 0
      IPROG = 0
      ICONT = –1

      IERR = IOSET (IXY, IRW, IPROG, ICONT)
      .
      .
      .
```

```
C
C    ICONT = 12B          Coordinate in i/100 cm.
C                         Treat data as integer
C                         Range – 32768   32767
C    Series 100                 –750      9000 (–7.5 cm – 90.00 cm)
C
C    ICONT = 13B          Coordinates in 1/1000 inch
C                         Treat data as first part of double integer
C                         Range 0 – 65534
C    Series 100                 0 – 36000 (0.00 inch – 36.000 inch)
C
C    Choose "cm"/"inch" modus
     ICONT = 12B
     IERR = IOSET (IXY, IRW, IPROG, ICONT)
     CALL LESXY (IXY, IX(2), IY(x), IERR)
C
C    IERR = 0      OK
C
C    IERR = 4      Cursor lifted
C
C    IERR = 5      Negative number in "inch" modus

     IF (ICM EQ.1) GOTO 200
     RX = DX/1000.0
     RY = DY/1000.0
     GOTO 300

     200 CONTINUE
     RX = IX(2)/100.0
     RY = IX(2)/100.0
     300 CONTINUE
```

MAC routine "LESXY":

```
     )9BEG
     )9ENT    LESXY
              IXY = 0
              IX = 1
              IY = 2
              IERR = 3

     LESXY,   SWAB DB SA
              COPY DX SA
              LDT I ,B IXY
              MON 1              % Input 1. coordinate
              JMP EX
              STA I ,B IX
              MON 1              % Input 2. coordinate
              JMP EX
              STA I ,B IY
              SAA 0
```

```
      EX1 STA I ,B IERR
          COPY DB SX
          EXIT
)9END
)LINE
```

7 CAMAC

CAMAC is a standarized way to connect peripheral equipment to a computer. It allows connection of a wide range of readily available equipment for process control, data aquisition and data handling to any computer with a CAMAC port.

A CAMAC system consists of one or more *crates* which communicate with the computer through a special *control* unit. Each crate is separated into 25 *stations* (control station and 24 normal stations) where various kinds of CAMAC modules may be placed. A maximum of 16 crates may be connected to, and addressed by, one NORD-10. The CAMAC crate controller CC-NORD-10 is a double width CAMAC module occupying the two right-most positions (control station plus one normal station) of a standard CAMAC crate.

A CAMAC module is a unit occupying one or more of the 25 CAMAC positions, and it is identified by one station number.

When a module wants computer program attention (interrupt), it generates a LAM ("look-at-me") request on a private line, L, to the control station.

Up to 23 L-signals may accordingly arrive to the control station. However, these 23 lines must be reduced to 16 to allow efficient handling in the computer. Some of the L-lines must, therefore, be tied together according to the user's choice. The resulting 16 lines are called GRADED LAM (GL) lines. The GRADED LAM register, which may be read by program, holds a bit mask which gives the current LAM requests. (Bit = 1 indicates LAM). A LAM may be accomplished by an interrupt, if this has been enabled. The different graded LAM lines have individual IDENT codes, bits 0-3 giving GRADED LAM code and bits 4-7 giving crate code, while bit 8 = 1 indicates CAMAC. A LAM MASK REGISTER is used to enable (bit = 1) or disable (bit = 0) any one of the 16 graded LAMs. Each time an IDENT code is read, the associated bit in the MASK register is cleared and must be set by program to enable a new interrupt.

A Control/Status register (COST register) holds some extra information about the mode of operation of the crate, and the current crate status. It holds information concerning which types of interrupt that are enabled, on which hardware interrupt level the LAM interrupts are transmitted, etc.

Control Register:

Bit 0 Dataway inhibit
Bit 1 Enable DMA transfer
Bit 2 Enable DMA block transfer
Bit 3 Enable LAM interrupt
Bit 4 Enable QX error interrupt (level 13)
Bit 5 Enable RT interrupt (level 13)
Bit 6 L10
Bit 7 L11
Bit 8 L12
} Set crate to interrupt level. Only one of these bits should be set.

The RT interrupt is an external high-priority interrupt (level 13), not necessarily generated by any CAMAC module. The QX error interrupt is also a level 13 interrupt. To distinguish them, the status register must be inspected. The common IDENT code gives the crate code by bits 4-7, while bits 0-3 are all zero.

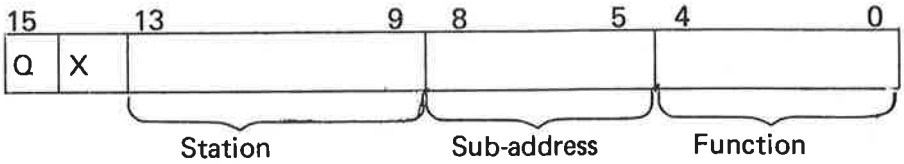
Further information about the Q and X status must be sought in ESONE specifications EUR 4100.

Status Register:

Bit 0 Dataway inhibited
Bit 1 DMA transfer enabled
Bit 2 DMA block transfer enabled
Bit 3 LAM interrupt enabled
Bit 4 Error interrupt enabled
Bit 5 RT interrupt enabled
Bit 6 L10
Bit 7 L11
Bit 8 L12
} Crate interrupt level
Bit 9 LAM demand
Bit 10 QX error occurred
Bit 11 RT interrupt occurred
Bit 12 Dataway is cleared (C)
Bit 13 Dataway is initialized (Z)
Bit 14 Dataway X response
Bit 15 Dataway Q response

The Q and X responses are given to the last dataway operation.

The CAMAC stations are controlled by the NAF register. NAF data looks like this:



If Q = 1 and/or X = 1, then Q and X responses from the station are automatically checked. Error interrupt will be generated if this has been enabled (see specifications for the control registers). For DATAWAY C (clear) and DATAWAY Z (initialize) no data is necessary in the NAF register. A data input/output operation must include an "EXECUTE DATAWAY CYCLE" function. The two signals Z and C will now be described.

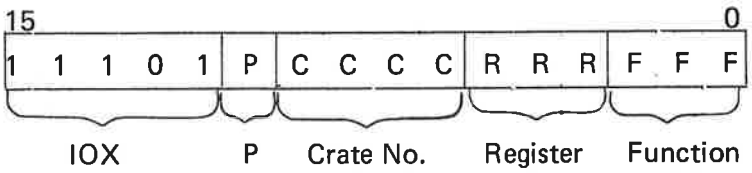
DATAWAY Z:

This signal generates a "CAMAC Initialize" cycle. NAF and DATA registers are cleared. COST takes the value 030001_8. MASK register remains unchanged. The Z cycle may be generated by program, MASTER CLEAR or "power up".

DATAWAY C:

This signal generates a "CAMAC clear" cycle. NAF and DATA registers are cleared. Bit 12 in COST is set to 1.

7.1 *IOX ADDRESSING FORMAT FOR CAMAC*



P = NORD-10 Peripheral Type

P = 0	Norsk Data peripherals
P = 1	CAMAC crates

C = CAMAC Crate Address 0-15

R = Crate Controller Register Address

R = 0	Dataway Z
R = 1	Dataway C
R = 2	Data Buffer
R = 3	NAF register
R = 4	Graded LAM pattern
R = 5	(Not used)
R = 6	Mask register
R = 7	Control/Status (COST)

F = Register Function

F = 0	All registers – Read
F = 1	All registers – Write
F = 3	Selected Bit Clear
F = 5	Selected Bit Set
F = 7	Execute Dataway Cycle

Register/Function	R	F
Dataway Z	0	7
Dataway C	1	7
Data Read	2	0
Data Write	2	1
Data Execute	2	7
NAF Read	3	0
NAF Write	3	1
NAF Execute	3	7
Graded LAM Read	4	0
Mask Read	6	0
Mask Write	6	1
Mask Bit Clear	6	3
Mask Bit Set	6	5
COST Read	7	0
COST Write (not bit 9)	7	1
COST Bit Clear (not bit 9)	7	3
COST Bit Set (not bit 9)	7	5

7.2 *PROGRAMMING EXAMPLES*

The letter 'C' in the IOX address indicates the 4-bit code for CAMAC crate 'C'.

```
% CAMAC INITIALIZE

        IOX     2C07        % Generate a 'Z' cycle

% CAMAC CLEAR

        IOX     2C17        % Generate CAMAC clear cycle

% READ STATUS

        IOX     2C70        % Read 'COST'

% SET CONTROL

        IOX     2C71        % Write 'COST'

% READ DATA

        LDA     NAFR        % A:=CAMAC read command
        IOX     2C37        % Write and execute NAF
        STA     BUFR        % Store data
```

```
% WRITE DATA

        LDA     DATA    % A:=Output data
        IOX     2C21    % To data register
        LDA     NAFW    % A:=Write command
        IOX     2C37    % Execute NAF
OR
        LDA     NAFW    % A:=Write command
        IOX     2C31    % Write to NAF
        LDA     DATA    % A:=Output data
        IOX     2C27    % Execute CAMAC cycle

% READ GRADED LAM STATUS

        IOX     2C40    % Read 'Graded LAM'
        JAZ     * – 1   % Repeat until LAM set

% MASK OPERATIONS

        IOX     2C60    % Read Graded LAM mask
        IOX     2C61    % Write GL mask
        IOX     2C63    % Bit clear to disable LAM
        IOX     2C65    % Bit set to enable LAM
```

7.3 *CAMAC UNDER SINTRAN III*

By means of the monitor call CONCT (MON 106), different RT programs may be connected to different LAM interrupts (numbered 1-16) on levels 10, 11, 12 or to the RT interrupt on level 13 (called LAM no. 0).

MON 146 INIT

CALL INIT (<flag>, <crate no.>, <level>)

Initializes the specified crate for the appropriate level, i.e., clears data away and masks and writes the COST register (enables RT, ERROR, and L for the actual level). All LAM interrupts enabled by MON 151 (LMASK) appear on the specified interrupt level.

MON 147 CAMAC

CALL CAMAC (<value>, <flag>, <crate no.>, <station>, <sub. address>, <function>)

Operates CAMAC (executes NAF).

If <function> is "clear", the contents of COST are returned in <value>. If function is "read", data is returned in <value>.

For "clear" and "read" functions <flag> must be ⩾ 0 on entry.

If <function> is "write", data must be in <value> and <flag> must be < 0 before the call.

Return: <flag> = 0, OK
<flag> = 24, NOT OK

If bit 15 in <crate no.> is set equal to one, <value> is treated as an integer instead of floating point.

MON 150 GL

CALL GL (<value>, <flag>, <crate no.>)

Read graded LAM status. The status is returned in <value>.

MON 151 LMASK

CALL LMASK (<value>, <flag>, <crate no.>)

Reads and writes MASK.

If <flag> ⩾ 0, the mask is returned in <value>. If <flag> < 0, the content of <value> is written to MASK.

Return: <flag> = 0, OK
<flag> = 24, NOT OK

MON 152 CONTR

CALL CONTR (<value>, <flag>, <crate no.>)

Reads and writes COST (control/status register).

If <flag> ⩾ 0, the COST is read and returned in <value>. If <flag> < 0, the contents of <value> are written to COST.

Return: <flag> = 0, OK
<flag> = 24, NOT OK

MON 153 IOXN

CALL IOXN (<value>, <flag>, <device no.>)

Issues direct IOX commands.

If <flag> ⩾ 0, an input transfer is executed and the information is returned in <value>. If <flag> < 0 an output transfer is executed and the output information must be in <value> before the call.

Return: <flag> = 0, OK
<flag> = 24, NOT OK

MON 154 ASSIG

CALL ASSIG (<logical no.>, <graded LAM>, <crate no.>)

Assigns a graded LAM in CAMAC ident table to a logical number in the logical number table. The graded LAMs are numbered 1-16. Note that LAM 0 (<graded LAM> = 0) is used for high priority interrupts on level 13.

The calls CAMAC, GL, LMASK and CONTR handle their <value> parameters as an integer if bit 15 in the <crate no.> parameter is set, otherwise, <value> is treated as a real number.

Note: For use with MAC, Standard FORTRAN, parameter communication is applied.

8 DIRECT TASKS

A "Direct Task" is a routine running on one of the free interrupt levels in SINTRAN III. A direct task may be started by an RT program (by the instruction MST PID, for instance) by activating the interrupt level where the direct task is placed. A direct task will run independent of the SINTRAN III system, and a direct task cannot use monitor calls, files or other facilities in SINTRAN III. It may, however, start RT programs.

Direct tasks must be loaded by the RT loader, and should be fixed in core by means of the monitor calls (or commands) FIX or FIXC.

Finally, the ENTSG monitor call (command) should be used to enter a direct task or a driver routine into the operating system. For further information about direct tasks, see the manual "SINTRAN III User's Guide".

An example, showing the load procedure for a direct task, is given below. The direct task, loaded from the file DIR-TASK:BRF is linked to a segment loaded from the file USER-PROGRAM:BRF which contains the RT program USERP.

```
@RT-L

REAL-TIME LOADER   76.04.29

*CLEAR-SEGMENT     36
*CLEAR-SEGMENT     37
*SET-PAGE-TABLE     3
*NEW-SEGMENT       36,,,,
*SET-LOAD ADDRESS 36, 100000
*LOAD  DIR-TASK:BRF,,
*WRITE-LOAD-ADDRSESS 36

L.ADR: 100000        U.ADR: 100271        C.LADR: 100272
*END-LOAD
*SET-PAGE-TABLE     3
*NEW-SEGMENT       37,,,,
*SET-LOAD ADDRESS 37,  102000
*LOAD USER-PROGRAM:BRF, 37, 36
*END-LOAD
*WRITE-PROGRAMS
OUTPUT FILE:

USERP   24567   37     36

*EXIT-LOADER

@FIXC   36   40
@ENTSG 36     0   8   100036B
@RT USERP
```

9 DDC PROGRAM PACKAGES

9.1 *MEAS II*

MEAS II is a program package for analog input, run by SINTRAN III monitor. There exists one version for solid-state multiplexer (V1) and one version for relay multiplexer (V2). Programming language is MAC.

MEAS II may handle up to 2048 process variables. Those variables which are scanned with the same frequency, are grouped together under one common RT description.

The user programs communicate with MEAS II by means of a set of interfacing subroutines, callable from FORTRAN. The main purpose of MEAS II is:

- to read analog signals from a process into the process computer (NORD-10/NORD-12) by using a multiplexer and an analog-to-digital converter (ADC)
- to handle calculated variables, i.e., process variables which are calculated by some program rather than being scanned by a multiplexer
- to perform measurement corrections on the signals scanned by the multiplexer, i.e., corrections for drift in the signal level amplifiers
- to perform measurement conversion, i.e., conversion from raw ADC value into internal machine representation
- to perform instrument limit check
- to perform measurement linearization
- to perform rate of change limit check
- to perform digital filtering
- to perform process variable limit check
- to perform measurement storing
- to generate messages about the alarm and return-to-normal conditions that are detected
- to give the user (operator, process engineer, application programmer) and the user-written programs the possibility to communicate with the system

MEAS II has a modular design, and some of these functions are optional.

The user specifies his process to an off-line program called DDC TABLE GENERATOR. (This program also generates tables for the two other program packages PROCSY and PROSO.) The tables generated by this program must be loaded together with MEAS II.

To run MEAS II, the SINTRAN III system must include some monitor routines (MON 170-173). In addition, 4 semaphores are used by MEAS II.

9.2 *PROCSY*

PROCSY is a modular program package for process control, run by SINTRAN III monitor. Programming language is MAC.

Process variables are gathered by the analog input program package MEAS II either periodically or at demand. PROCSY uses these variables as input to the control algorithms thereby generating a set of control variables that in turn are handled over to the analog output program package PROSO.

The user programs communicate with PROCSY by means of a set of interfacing subroutines, callable from FORTRAN. The main purpose of PROCSY is:

- to do proportional, integral and derivate (PID) control or variations of PID
- to do cascade control
- to do ratio control
- to do feed-forward control
- to do simple multi-variable control (cross-coupling between control loops, etc.)
- to do limit checking of set-point and deviation
- to generate messages about the alarm conditions that are detected (both alarm and return-to-normal messages)
- to link user written programs (application dependent programs) to the system through a set of interfacing routines.

The PROCSY tables are generated by the DDC TABLE GENERATOR, an off-line program which is run on a background terminal. The BRF file produced by it must be linked to the DDC program packages. PROCSY needs two SINTRAN III semaphores, and the same monitor routines as MEAS II.

9.3 *PROSO*

PROSO is a modular program package for analog output, run by SINTRAN III monitor and written in MAC. PROSO outputs process actuator commands either at demand or at various frequencies. The user programs communicate with PROSO by means of a set of interfacing subroutines, callable from FORTRAN. The mains purpose of PROSO is:

- to take input from controller programs. This input, which is the output of a controller algorithm (the control variable), has to be an absolute (position) value.
- to perform limit checking of the control variable. If out-of-range values are detected, the control variable is clamped and the violated limit value returned to the calling program to avoid controller wind-up.
- to generate messages about the alarm conditions detected (both alarm and return-to-normal messages)
- to convert the accepted control variable into a meaningful actuator command
- to send the actuator command to the process interface. The correcting value, which is part of the command, can either represent an absolute or incremental signal
- to link user-written programs (application dependent programs) to the system through a set of interfacing routines

The PROSO tables are generated by the DDC TABLE GENERATOR which is a completely off-line program. These tables must be loaded together with PROSO. PROSO requires four SINTRAN III semaphores, and the same monitor routines as MEAS II.

10 SOME I/O DEVICES FOR THE DDC OPERATOR

10.1 *THE PROCESS CONSOLE*

10.1.1 *Programming Specifications*

The process console is a free standing unit. The unit is provided with communication interface for full duplex, TTY compatible connection, and for connection for asynchronous modem, RS-232.

Standard Features:

- Alphanumeric display, 32 characters long
- Keyboard with up to 8 x 16 keys in a rectangular matrix. The number and arrangement of the keys are left to be decided by the customer. Specifications for the keyboard layout should be available in Norsk Data's production department 3 months prior to delivery.
- Each key may have text and individually programmable light.
- A lockable key switch will add the octal value 200 to any key number (force bit seven to one) to facilitate a selectable key lock function.
- Accoustic signal (600 Hz) with programmable duration.
- Test mode. Echo on all received characters.
- Transmission speeds: 110 or 1200 bauds.

The process console is programmed through a set of instructions. These instructions are ordinary characters which are decoded inside the process console. Available instructions:

SCMND	Set Command	Octal code 100
SADR	Set Address	Octal code 140

Command bits in SCMND instruction:

Bit 0	Blank display
Bit 1	Clear display buffer
Bit 2	Sound on
Bit 3	Test mode on

Note: Command bits are active until they are reset by the program. Characters to display should be given in 6-bit ASCII code.

The SADR instruction transfers the column address (the 4 significant bits of the 7-bit key address) to the console as bits 0 to 3 of the instruction. This instruction must always be followed by an 8-bit bit pattern for the corresponding 8 lamps in one column.

It is recommended to use one key for a lamp test function. The code of this key should be detected, and for a length of time determined by the program (1-5 seconds) all lamps should be lighted without disturbing the contents of the lamp map in core.

The input character given when pressing a push-button consists of the row number (0-7, where 0 denotes the lowest row) in bit position 0-2 and the column number ($0\text{-}17_8$, where 0 denotes the right-most column) in bit position 3-6. Thus, the top-most, left-most push-button in the matrix has input code = 177_8.

10.1.2 *Programming Examples*

```
SCMND  =   100
BLDIS  =     1
CLDIS  =     2
SOUND  =     4
SADR   =   140

% CLEAR DISPLAY, SET DISPLAY POSITION 0
      LDT    (CONS            % T:=Logical console DVN
      SAA    (SCMND + BLDIS + CLDIS
      MON    2
      JMP    ERROR
      SAA    SCMND
      MON    2                % OUTBT skip if successful
      JMP    ERROR
      .
      .
      .

% WRITE CHARACTER ONTO CURRENT DISPLAY POSITION
      LDT    (CONS
      SAA    SCMND
      MON    2
      JMP    ERROR
      LDA    CHAR             % A:= character
      AND    (77              % Truncate
      MON    2
      JMP    ERROR
      .
      .
      .
```

```
% PUT ON LAMP NO. 2 IN COLUMN NO. 7
% (IN THIS EXAMPLE ALL OTHER LAMPS IN COLUMN ARE
  SWITCHED OFF)
    LDT     (CONS
    SAA     SADR + 7        % SADR + COLUMN
    MON     2
    JMP     ERROR
    SAA     4               % Bit mask for lamp no. 2
    MON     2
    JMP     ERROR
    SAA     SCMND
    MON     2
    JMP     ERROR
    .
    .
    .

% SWITCH OFF LAMP 3 AND 5 IN COLUMN NO. 1.
% SWITCH ON ALL THE OTHER LAMPS IN THAT COLUMN
    LDT     (CONS
    SAA     SADR + 1
    MON     2
    JMP     ERROR
    LDA     (327
    MON     2
    JMP     ERROR
    SAA     SCMND
    MON     2
    JMP     ERROR

% SWITCH ON SOUND:
    LDT     (CONS
    SAA     SCMND + SOUND
    MON     2
    JMP     ERROR

% SWITCH OFF SOUND
    LDT     (CONS
    SAA     SCMND
    MON     2
    JMP     ERROR
    .
    .
```

If the console is handled by a stand-alone program, the MON 2 instruction may be replaced by a JPL I (OUTB instruction, where OUTB may look like this:

```
OUTB,   COPY  SA  DD
        LDX   100000      % Time counter
        IOX   DEV + 2     % Read status
        BSKP  ZRO 40 DA   % Skip if no errors
        EXIT
        BSKP  ONE 30 DA
        JNC   * - 4
        JXZ   * – 3       % Jump if time-out
        COPY  SD  DA
        IOX   DEV + 1     % Write data
        SAA   4
        IOX   DEV + 3     % Activate device
        EXIT  AD1
)FILL
```

10.1.3 *The Process Console under SINTRAN III*

For input, apply the driver TELIN. (TRGET should be used as its IOTRANS routine.) For output, the normal teletype output driver may be used without modifications. Since TELIN may be a library routine in SINTRAN III, the need for it should be mentioned in the comment field of the "SINTRAN III Order Form".

TELIN gives break for every character, and no echo is given. (This must be given by the user program, which interprets the input.)

A console output handler KONSO has been made, and may be delivered on special request. It consists of an NPL subprogram which may be called from FORTRAN and runs on application level. Characters may be written or read from the display, lamps or sound are put on/off, etc. A complete version occupies approximately 600_8 locations.

10.1.4 *KONSO, the Console Handler Program*

A general integer function KONSO is used for output to the console.

KONSO has 3 parameters:

I = KONSO (K, L, M)

K is console number (between 0 and BORDM, BORDM is a symbol defined in KONSO).

L is a function code (between 0 and FUMAX, FUMAX is a symbol defined in KONSO).

M depends on the function code L (M may be lamp number, character, display position, dummy).

I is the returned function value, where negative values indicate error:

I = –1: K (console number) out of range
I = –2: L (function code) out of range
I = –3: Lamp number out of range
I = –4: Display position out of range
I = –5: Output character out of range
I = –7: Any error from SINTRAN I/O (outbt error)

The implemented functions are:

L = 0, M = dummy

Update all lamps from the lamp table. Each lamp has a bit in the lamp table. If the bit is zero, the lamp is set to off. Initially, all lamps are off.

L = 1, M = lamp number ($0\text{-}127_{10}$)

Read the status of the lamp specified in M. If the lamp is off, the function value I = 0. If the lamp is on, I = 1. (This information is found in the lamp table.)

L = 2, M = lamp number ($0\text{-}127_{10}$)

Set the lamp specified in M to ON. The lamp table is updated (corresponding bit is set to "1").

L = 3, M = lamp number ($0\text{-}127_{10}$)

Set the lamp specified in M to OFF. The lamp table is updated (corresponding bit is set to "0").

L = 4, M = display position ($0\text{-}31_{10}$)

Set current display position to value specified in M.

If the current display position is decremented, the display is cleared and then the contents of the display table from the start to the new currrent position is copied to the display.

If the current position is incremented, the contents of the display table from the start to the new current position is copied to the display.

If the current position is incremented, the contents of the display table from the old to the new current position is copied to the display.

L = 5, M = character (40_8 - 137_8), $32_{10} = 95_{10}$)

Write the specified character to the current position of the display. The display table is updated, and the current position is incremented by one.

If the original current position was skip marked, the character will be written to the first not skip marked position. Space will be written to all skip marked positions. (Any other special character may be used instead of space.)

L = 6, M = display position (0-31)

Read the character in position M in the display table. The character will be returned as the function value I. If the specified position is skip marked, bit 8 of the returned character is "1".

Other functions may be implemented in KONSO. These may be:

Reserve and release
Transparent output (not affecting the tables)

These functions may be performed by the Standard SINTRAN calls, however, the SINTRAN logical device numbers must be used instead of the logical console number used in KONSO. Therefore, it is convenient to define a function in KONSO to find the SINTRAN logical device number for a specified console.

L = 7, M = dummy

The SINTRAN logical device number of the console specified in parameter K is returned as function value I.

L = 10_8, M = state

Sound on or off ("on" if M unequal to 0).

L = 11_8, M display position (0-31_{10})

Clear console from given position. Set given position.

10.2 *THE NORDCOM COLOUR TERMINAL*

10.2.1 *Main Characteristics*

The NORDCOM NCT is a semigraphic colour display terminal. Since it will be described in other manuals, only some main characteristics will be listed here:

- 48, 36, 24 or 18 character lines, depending on character size. (There is a possibility of "windowing".)
- 64 characters on each character line
- character length is always 8 dots, and the possible character sizes are 8 x 6, 8 x 8, 8 x 12 or 8 x 16
- The number of different symbols is 128 for 8 x 12 or 8 x 16 dot matrices, or 256 for 8 x 6 or 8 x 8 dot matrices.
- The size of the symbol and the symbol forms are program controlled.
- Each symbol written on the screen appears with one foreground colour and one background colour.
- Blink state may also be set.
- The 16 possible foreground colours, the 8 possible background colours and the 8 possible cursor colours are program selected from a set of 4096 different combinations (16 red intensity levels, 16 green levels, 16 blue levels; 16 x 16 x 16 = 4096).

- Screen positions may be given by absolute addresses or by incremental address steps.

- Reading from the NCT is not possible. An ASCII character set (8 x 8 dot matrices) and a standard colour set is loaded from a read only memory initially (at power-up time).

- Some hardware status is set to default values.

- A special NCT key-board and a joystick may be connected.

10.2.2 *The NCT-SP*

The NCT-Service Program is separated into two parts:

- one subsystem which makes it easy to update, generate and save different pictures (or picture parts), symbol and colour definitions.

- one BRF library containing most of the subsystem facilities to be called and used by other subsystems.

A reasonable way to use the NCT-SP is to generate the wanted symbols, colours, macros and main pictures by means of the subsystem, and then update, change and look at the results by calling the BRF library from programs written in BASIC, FORTRAN or NORD PL.

Since the NCT-SP is described in a separate manual, only a brief description is given here. Some of the most important subsystem commands are:

NCT-NAME <device name>

- tells the NCT-SP the file name of the NORDCOM COLOUR TERMINAL

LOOK-AT-COLOUR <FG/BF> <No.>

- displays the colours on the NCT

WRITE-COLOUR <FG/BG/CU> <No.> <input file>

- defines a colour by giving the number of red/green/blue parts the colour consists of

SAVE-COLOURS <output file>

- saves the present colour definitions on the specified output file

LOOK-AT-SYMBOL <buffer no.> <symbol no.>

- displays the symbols on the NCT

WRITE-SYMBOL <buffer no.> <symbol no.> <input file>

- symbols may be defined by giving their forms by matrices of x's and points

SAVE-SYMBOLS <buffer no.> <output file>

- saves the symbol definitions on the specified output file

CLEAR

- clears the screen and two internal program buffers holding picture data

SAVE-PICTURE <output file>

- saves all information given to the NCT since last CLEAR command on the specified output file

WRITE-PICTURE <output file>

- saves the content on the refresh image buffer on the specified output file

In addition, there are several macro commands. A macro is a picture part which may consist of both control and data characters. Macros may be defined, inspected and moved on the screen.

10.2.3 *Picture Editing with GPM*

An interesting tool for building of static NCT pictures, or picture parts, is the General Purpose Macroprocessor – GPM.

GPM works as subsystem in SINTRAN III, completely off-line. It reads characters from the specified input file and sends its output to the specified input file and sends its output to the specified output file. A call of macro NAME with actual parameters PAR1 and PAR2 looks like this:

↑NAME, PAR1, PAR2;

GPM has system macros for arithmetical operations, and may therefore easily transform NCT screen co-ordinates and modify characters (by adding increments to their ASCII code, for instance).

The main advantages of using GPM is:

- Control information may be referenced by name. (Example: control character for "blue foreground colour" may be given by the macro call ↑BLUE-FG;)

- A macro call may yield a sequence of semigraphic characters. The number and type of characters may be deducted, or computed, from the parameters in the macro call. (Example: a horizontal line segment on line number 5, from column 8 to 46 may be written ↑H-LINE, 5, 8, 46;).

- Special symbols may be called by name. (Example: a value symbol in position [9, 6] may be written ↑VALVE, 9, 6;)

- Some standard figures such as squares, triangles, etc. may be defined as macros. Note that the size and form of these figures may be altered by parameter values in the macro call. (Example: a square with the topmost, left-most corner in [7, 3], with hight as 5 and length as 10 may be written ↑SQUARE, 7, 3, 5, 10;)

- The user may define and name his own picture parts. In the definitions, he may freely switch between symbol buffers, FG and BG colours, etc. (Example: the user may define a picture part AMPLIFIER that consists of green wires, red resistors, blue condensers, etc. If he wants it with the topmost, left-most position in [5, 3], he writes ↑AMPLIFIER, 5, 3;)

Description of Use:

The user edits (with the QED editor, for instance) his picture data on a file PICTURE. This symbolic file may look similar to this:

```
↑ERASE;
↑BG-COLOUR, 0;
↑FG-COLOUR, 3;
↑SYMBOL-BUFFER, 0;
↑POS, 2, 2; THIS IS THE HEADING
↑SYMBOL-BUFFER, 1;
↑H-LINE, 5, 0, 63;
↑SQUARE, 10, 8, 5, 4;
↑OCTAGON, 20, 10, 2, 2, 2;
↑SILO, 20, 40, 5, 2;
↑FG-COLOUR, 1;
↑TRIANGLE, 7, 40, 5;
↑VALVE, 8, 12;
↑PICTURE-PART-1, 20, 30;
```

Then he recovers GPM. Imagine that the name of his macro definition file in MACRODEF. (There is the definitions of FG-COLOUR, ERASE, PICTURE-PART-1, etc.) The user writes on his terminal:

```
@GPM

OUTPUT FILE NAME:  NCT
INPUT FILE NAME:  MACRODEF
INPUT FILE NAME:  PICTURE
INPUT FILE NAME:
```

Now he should have his picture on the NCT screen, if the proper content of symbol buffers and colour buffers has been set by the NCT service program.

NORSK DATA A.S.
Lørenveien 57 - Postboks 163, Økern
OSLO 1

COMMENT AND EVALUATION SHEET

NORD Process I/O – Software Guide ND-60.093.01
September 1977

In order for this manual to develop to the point where it best suits your needs, we must have your comments, corrections, suggestions for additions, etc. Please write down your comments on this pre-addressed form and post it. Please be specific wherever possible.

FROM _____

_____

_____

DISCUSSION

* Which are the main markets?

* How big are these markets?
  — Today
  — In 1985

* How should/could ND approach these markets?
  — OEM contracts with turn-key vendors
  — Joint venture with specialized system houses
  — Different approach in each country?

* What software should ND access to and how?
  — Basic
  — Applications