A L I C E

A computer program for nuclear data acquisition

T.B. Skaali

Institute of Physics, University of Oslo
POB 1048, Blindern, Oslo 3, Norway

OUP--

Report 81-06

A L I C E

A computer program for nuclear data acquisition.

T.B. Skaali

Institute of Physics, University of Oslo
POB 1048, Blindern, Oslo 3, Norway

Alice thought to herself
"I don't see how he can ever finish,
if he doesn't begin".

Lewis Carroll,
Alice's Adventures in Wonderland.

Feb. 1981

CONTENTS

PART 1 : A L I C E U S E R S G U I D E

CONTENTS

PREFACE

This manual contains the Users Guide and the program documentation for the ALICE data acquisition system. The ALICE Users Guide, which is contained in part 1 of the manual, can be read independently of the program documentation in part 2.

The ALICE program is written in the interpretive language NODAL. Due to the inherent slow execution speed of interpreted code time consming tasks such as non-linear least squares peak fitting can not be implemented. On the other hand the special features of the NODAL language have made possible facilities in ALICE which hardly could have been realized by, for instance, a FORTRAN program.

The complete system can be divided in two parts, i) the ALICE program written in NODAL, and ii) a data acquisition package which logically represents an extension of the SINTRAN III operating system. The system is thus portable to other NORD-10/100 installations provided that the floating hardware is 48 bits.

1. INTRODUCTION

ALICE is a data acquisition program system developed for the NORD-10.S computer installation at the Oslo University Cyclotron Laboratory.

The basic task of ALICE is the accumulation of one-dimensional nuclear data spectra together with some on-line data reduction operations. Among the special features implemented in the system are

- HELP commands for on-line user support.
- A sequence of commands can be grouped together in a "command program" and executed as an entity. Command programs can be generated and stored onto file by means of a built-in command editor. Such a command program can be executed by a single command from the terminal.
- Symbolic parameters can be substituted for numerical command parameters. The value of a symbolic parameter will be evaluated at execution time by the command processor. The value of the symbolic parameters are set and changed by commands.
- Users can generate and execute their own command routines written in NODAL.
- The command processor can be set in a mode where command strings are interpreted as NODAL program statements.
- Terminal output from the program can also be written onto a log file.
- The data acquisition task is a completely independent task. Once a data acquisition is started, ALICE can be stopped and other programs can be executed. When restarted, ALICE resumes the execution in the same state as it was when it was stopped.

The Users Guide part of this manual gives a description of ALICE from a users point of view, with a particular emphasis on the command syntax and the command repetoire. The basic concepts of the program system are presented in section 2. The data acquisition task is described in section 3, and a short description of the DICO display system is given in section 4. Sections 5 and 6 contain descriptions of the command syntax and the various commands. Users who are only interested in the practical application of the program can skip sections 3 and 4.

The second part of the manual gives a more detailed description of the organization of the program system. The program files of ALICE are described in Appendix A. Documentation of the internal program structure and the program variables are given in Appendices B and C, respectively. Finally, Appendices D and E deal with the real time data acquisition functions and the SINTRAN III data acquisition package.

The ALICE program system is written in the interpretive language NODAL. The basic NODAL system has been expanded with a set of functions which controls the real time data acquistion task by means of monitor calls to the operating system. The program runs as an interactive RT-NODAL program under the operating system SINTRAN III. The NODAL language and the SINTRAN III - NODAL system are described in Refs. 1 and 2, respectively. SINTRAN III is documented in Ref. 3.

1.1 How to run ALICE

The ALICE program is executed as a RT program. Under SINTRAN III each terminal may be assigned an interactive RT NODAL, cf. Ref. 2. The interactive RT-NODALs are named

NOD<dev>

where <dev> stands for the terminal device number. Each RT-NODAL has its own working area which contains the program code and the variables. The remaining free part of the working area is used by the interpreter as a stack area.

In principle ALICE can be run from any terminal provided that the NODAL working area is 32K or more. (Program + variables 12K, maximum spectrum length 16K, stack area 4K.) Normally ALICE is started through a small NODAL program stored under the user RT. (Default user name for files accessed from RT-programs is RT). This program will check that the working area is large enough.

Procedure to start ALICE from terminal 34.

- Login under user RT
- @RT NOD34
- @LOGOUT
- NOD34 should now start by printing out a message. If not, check terminal number.
- >RUN ALICE
  The program starts by printing out some general information plus the character ":" (colon) which signals that a command can be entered.
- :commands
- : ....
- :ZZZZ
  This command terminates the program.

If ALICE has to be aborted, this can be done by logging in on another terminal under user RT or SYSTEM and giving the

command

$ABORT MOD34

After a forced abort or an abort due to a fatal program error, the data acquisition task must be reset. This is done by running the program

(ALICE)CLEARUP

from a RT MODAL. This program will also set the ALICE status to "stopped by ZZZZ".

The program (ALICE)CLEARUP also contains a section which defines the hardware addresses and writes them onto file, cf. program listing.

2. GENERAL DESCRIPTION OF A L I C E

Conceptually, the ALICE program system may be divided into the following parts:

i) A command processor which interpretes and executes commands entered from a terminal or read from file.

ii) Global subroutines and program variables for spectrum handling and display control.

iii) Command subroutines, which may either be part of the main program or file resident. The response time from file resident command routines is relatively slow due to the extra overhead from the file system.

iv) Associated real time (RT) programs for specific tasks. A typical example is the plotting program for the Calcomp plotter. The associated RT programs can be written in any language. Parameter and data transfer are done via files.

v) A set of data areas, where the data is stored as pulse height spectra (histograms) in one dimension. The data areas are identified by a number, the spectrum number. All data areas are mass storage resident, and only one area can reside at a time in the NODAL working area. A data area that is referenced by a command and which is not already in the working area is automatically fetched before the command is executed. If changes have been made on the workspace resident data area this area will be written back onto mass storage before a new one is fetched. From a user's point of view the data area handling is completely transparent.

Each data area consists of three items; a spectrum data array, a spectrum information array, and an energy calibration array. Hence each spectrum may for instance have its own energy calibration.

1) Spectrum data array.
NODAL single or double precision integer array. Single and double precision integers are represented by 16 and 32 bits, respectively. Hence the range of a single precision integer is limited to between -32768 and +32767 counts.

2) Spectrum information array.
NODAL integer array.
Contains spectrum parameters (length, precision, display parameters, etc.)

3) Energy calibration coefficients.
NODAL floating array.
Contains coefficients for the expression
E(keV) = DE(1)+DE(2)*X+DE(3)*X**2, where X denotes the channel number and DE(i) are the elements of the energy calibration array.

The length and precision of the spectrum arrays are set by command. At program start-up the initial data area configuration will be taken from the previous run.

3. THE DATA ACQUISITION TASK

The accumulation of data requires the use of the interrupt system of NORD-10 and allocation of memory space for data storage. This task is carried out by routines that have been added to the operating system SINTRAN III. These routines are activated by monitor calls or external interrupts, cf. Appendix E. The data acquisition is controlled from the NODAL level by means of the data acquisition functions that are described in Appendix D.

In order to understand the basic principles of the data acquisition task a few words must be said about the memory management system of the NORD-10 computer. The memory management system organizes the memory in "pages", each page is 1 K words = 2 K bytes. The system enables programs to execute in a 64 K word logical memory address space, even if the available physical memory is less than the size of the program. The operating system will move "pages" to and from the disc when required. On the other hand, the maximum size of a program is (usually) limited to 64 K words, even if NORD-10 can accommodate up to 256 K words of memory.

3.1 Software

The NODAL working area constitutes a contigous logical address space, but physically it may be scattered around in the memory. Furthermore, the working area is dynamic in the sense that program variables may be created and erased at will, and the variables can therefore not be assigned fixed addresses in the area. (NODAL organizes the variables in a list structure. When a variable is erased the corresponding element is removed from the list).

Data from the ADCs are transferred via interfaces in CAMAC to NORD-10 in Direct Memory Access (DMA) mode, cf. section 3.2. Data transfer in DMA mode must go to a fixed buffer area in physical memory. Hence the data acquisition task must copy the data from this buffer area to the different spectrum areas of ALICE. Another problem arises from the fact that the NODAL working area can only contain one spectrum area at a time (cf. section 2), and as explaned above, this spectrum area will not be assigned a fixed address space.

In brief, the data acquisition task has been implemented in the following way.

For each spectrum which is connected to an ADC, an "image" of that spectrum is created in a common data acquisition area. (Also called ADC data segment in Appendices D and E). This common area does not belong to the NODAL address space, and can therefore be made as big as 64 K words. The common area is locked in memory as long as the data acquisition task is active. The length of the area is fixed, and may be less or greater than the totalled length of the spectra defined by ALICE. Obviously, if the data acquisition area is less than the accumulated length of the spectrum arrays not all spectra can be connected to an ADC.

The length of the data acquisition area is printed out during the initialisation of ALICE. By means of monitor calls the content of an "image" spectrum can be copied into an ALICE spectrum array and vice versa whenever required.

This structure gives complete freedom for concurrant data acquisition and spectrum analysis. The spectra which are not connected to an ADC can be used for the analysis of previously recorded data. In fact, the data acquisition task is completely autonomous, and can run independently of ALICE provided that the data acquisition area is not affected. This feature is utilized by the PAUSE command, cf. section 6.

3.2 Hardware

The data acquisition hardware is connected to the computer via CAMAC, using the dedicated NORD-10 CAMAC Crate Controller.

The ADCs are interfaced by a specially developed input register, the ADC-CAMAC Interface. This single width CAMAC module is a general purpose unit for interfacing of Analog-Digital-Converters to CAMAC. The module uses the DMA and AFC facilities (link bus) of the NORD-10 Crate Controller. The module is documented in Ref. 4.

The maximum data rate is usually limited by the conversion time of the ADC. For fast ADCs the mean conversion time is typically around 10 microseonds. The time used for transferring a dataword from an ADC to the memory buffer in NORD-10 is 2-3 microseonds. However, an overhead of approximately 200 microseconds is added whenever the a buffer is [illegible] and the data stream is switched to another buffer.

The data acquisition through the ADC-CAMAC interfaces is controlled by the Inhibit signal on the CAMAC dataway. An external gate signal derived from the current integrator unit is connected to the Crate Controller Inhibit input. This signal synchronizes the start and stop of data acquisition in the crate.

As the data acquisition start/stop is also controlled by program commands, the following procedure must be used if a synchronous start/stop is required.

At start:

1)	Program command	START,<ADC #1>,<spectrum #1> START,<ADC #2>,<spectrum #2> .....
2)	Current integrator	ON

At stop:

3)	Current integrator	OFF
4)	Program command	STOP [,<ADC no>]

4. THE DICO DISPLAY SYSTEM

The DICO display system is a CAMAC based system which produces character and graphic information on a standard TV monitor. The system and the basic software have been developed at CERN, Ref. 5.

The system consists of two CAMAC modules. The DICO (DIsplay COntroller) module contains a microprocessor with a program memory and control logic. The DICO processor is programmed to accept simple display instructions from a host computer. From such instructions DICO will generate a sequence of display coordinates which are stored in the second CAMAC module; the video picture memory. A high resolution picture memory contains 768 by 576 picture elements along the x-axis and y-axis, respectively.

Under the operating system SINTRAN III the DICO systems are accessed as peripheral files. On the Cyclotron Laboratory installation the filenames are DICO-1 and DICO-2. For further documentation see Ref. 5.

The DICO system is fairly slow. The time used to display a 512 channel spectrum is 2-3 seconds. Erase can either be full screen or selective. In the latter mode the information to be erased is redisplayed with inverted intensity setting. Hence the system is less suited for "live" display.

5. THE COMMANDS OF ALICE

All actions by ALICE are initiated by commands. A command consists of a command name followed by a parameter list if required. Commands can be entered from a terminal or read from a command program on file

For terminal commands, i.e. commands executed immediately after being typed in from a terminal, the normal line editing facilities of NODAL can be used. (These are similiar to the edit functions of SIMTRAN). For example, pressing (CTRL)D will repeat the last command, (CTRL)A will delete (backspace) one character, and (CTRL)Q will cancel all that have been typed in.

Command execution can be aborted from the terminal by means of the "BREAK" character. This feature is useful for instance when one wants to terminate the execution of a command loop or a command program.

The sections 5.3 to 5.6 can be skipped if only a basic knowledge of the command syntax is wanted.

5.1 Command Syntax

When ALICE is ready to accept a command it prints ":" (colon) on the terminal. The typed-in command string, which may contain several commands delimited by ":", must be terminated by RETURN.

The name and parameters in a command string must be separated either by "," (comma) or space. For some commands certain parameters will take default values if they are omitted from the command.

Command names can be abbreviated by giving sufficient characters to distinguish them from other names. The minimum abbreviation is determined by the command processor, hence the introduction of new commands may change the permissible abbreviation of existing ones. The HELP command gives a list of the command names and their abbreviations.

A few examples of command strings are shown below. The first character ":" is printed by the command processor.

```
:DISPLAY,1                - Display spectrum 1.

:INTE 1:INTE 2            - Integrate spectrum 1,
                          - Integrate spectrum 2.
```

For repetative execution of a command sequence the sequence can be enclosed by a

LBEG,<n>: ..... :LEND pair.

The parameter <n> stands for the number of executions of the sequence. Command loops can not be nested. (See however section 5.3).

```
:ZERO,1:START,1,1:LBEG,5:LIST,1,512,540:LEND:STOP

          - Zero spectrum 1
          - Start data accumulation from ADC 1
          - Begin loop of 5 sequences
          - List content of channels 512 to 540
          - Terminate loop sequence
          - Stop data accumulation.
```

5.2 Display Oriented Commands

The display system plays a central role in the interaction between a user and the operations carried out by ALICE. The concept of the "current spectrum" establishes a link between the user and a working spectrum. The "current spectrum" is always the displayed spectrum, and if several spectra are displayed simultaneously, the first one as counted from the bottom of the screen. Display oriented commands such as setting of scale factor, markers, etc. use the "current spectrum" as default operand. Furthermore, some commands as LIST, ZERO, READ, WRITE etc. operate on this spectrum if the command is given without spectrum number argument, cf. Sect. 6.

5.3 Command Programs

The execution of tedious command sequences can be simplified by using a facility that ressembles the MODE command of SIMTRAN III.

A command sequence, hereafter called a command program, can be stored onto file and executed by means of the MODE command. A command program can consist of any number of lines, and it can contain almost all commands that are allowed in a normal terminal command. (The few exceptions are obvious; for example MODE is not permitted inside a command program). A command program can be edited; that is, lines can be changed, deleted and added, by means of a simple command editor. The use of this editor, which is itself entered by a command, M2ED, is self-explanatory.

A command program can be stored on any file of type :MOD. The default user name is RT. If the filename is omitted a default file is used.

A command program can contain a LBEG...LEND sequence, even if it is entered from inside a loop in the terminal command.

Assume that the following command sequence has been created by means of M2ED and written on the default command program file.

```
ZERO,2
START,1,2
LBEG 10
DISPLAY,2:INTEGRATE,2
LEND
STOP
```

This program can be executed by the command

```
MODE
```

The following terminal command will execute the same command program four times

```
LBEG,4:MODE:LEND
```

5.4 Symbolic Command Parameters

A symbolic argument can be substituted for any numerical parameter in a command string. Three symbolic arguments have been implemented, they are denoted A1, A2 and A3.

The value assigned to a symbolic argument is set and changed by the commands ARGSET and ARGCHANGE.

```
:ARGSET,1,3            % A1:=3
:ARGCHANGE,2,-1        % A2:=A2-1
:ARGCHANGE,A1,0.25     % A(A1):=A(A1)+0.25
```

As an example, after ARGSET,1,3 the command

```
LIST,A1,500,600
```

is equivalent to

```
LIST,3,500,600
```

Initially the value of the symbolic arguments are zero.

The symbolic command parameters are very useful in command programs. As a simple example, assume that the spectra on magtape files 5 - 34 are to be integrated from channel 420 to 480. A mode file for this job will look like

```
FFILE,MT1,35
ARGSET,1,5
LBEG,30
READ,MT1,A1,1
INTEGRATE,1,420,480
ARGCHANGE,1,1
LEND
```

5.5 Execution of NODAL Program Statements

The command processor of ALICE can be set in a mode where it interprets a command string as NODAL program statement(s). No syntax check is performed on the string prior to its execution, NODAL errors will, however, be reported in the normal way and control will be returned to the command processor. The command processor is set in NODAL mode by means of the command NODON. Return to normal command mode is done by NODOF.

A NODAL command statement may not contain the delimiter ";".

The NODAL command mode is useful for quick evaluation of arithmetic expressions. It is also extremely powerful for program debugging since all program variables can be accessed.

```
:NODON                          - Set in NODAL mode
:TYPE SIN(37.5*PIE/180)         - Evaluate sin(37.5)
:TYPE ! 1800/4.23               - Evaluate expression
:NODOF                          - Return to normal command mode
```

The same sequence can also be given in one command string. Note that the individual NODAL commands must then be delimited by ";" (semi-colon).

```
:NODON·TYPE SIN(37.5*PIE/180); TYPE ! 1800/4.23:NODOF
```

Care should be exercised when executing NODAL statements that sets the value of a variable, such as "Q" in SET Q=LOG(20). Firstly, new variables that will be created will occupy space in the working area, and secondly, if a variable of that name is already defined and used by the program the whole system may be destructively affected. Hence, if variables must be created the same rule which apply for ENTER commands must be followed, that is, the name of a variable must start with the letter "Q". No program variables have names starting with this letter. Such variables should be erased after use, i.e. by

```
ERASE Q1 Q2 QX
```

if these variables have been created.

Warning!
A NODAL command statement must not contain the END command.

5.6 User Defined NODAL Command Programs

This feature enables the user to implement her (his) own commands. In ALICE, the NODAL line numbers 90.01 to 94.99 are allocated for user defined command routines.

For each user defined command the associated NODAL program must be written off-line and stored on a file by means of the SAVE command. The program module is executed by means of the command

ENTER,<filename>,<parameter list>

Further details are given in Sect. 6.2.

It should be emphasized that a good knowledge of both the NODAL system and the internal structure of ALICE is essential for implementation of user defined command programs.

6. DESCRIPTION OF THE COMMANDS

In the preceding sections, several examples of commands have been given without further description. This section contains a documentation of all implemented commands.

For commands that can be given with more than one set of parameters, this option is indicated by writing the optional parameters inside square brackets [ ]. Missing parameters are replaced by default values that will be defined in each case in the following.

As an example the command LIST will list the content of a spectrum or part of a spectrum. The notation

LIST [,n [,i,j]]

stands for:

```
LIST       : "current spectrum" from Low to High marker
LIST,n     : spectrum <n> from first to last channel
LIST,n,i,j : spectrum <n> from channel <i> to channel <j>
```

Angle brackets < > surround symbols for which the user must substitute a specific value, i.e. either a number or a string. (In fact, ALICE evaluates a parameter such that for instance the expression "2*5" can be given instead of the number "10", but the use of this feature is not encouraged).

In most cases the meaning of the command parameters should be obvious. The channels of a spectrum are numbered from zero. Hence for a 2048 chs. spectrum the numbering goes from 0 (zero) to 2047. The syntax of a file name denoted by <filename> is that of the SINTRAN III file system. Note that the default user name for files accessed from RT program is user RT. Hence if a file belongs to another user the user name must be included in the <filename>, for instance (INGEBRETSEN)FWHM. The default file type is :NOD for command programs and :SYMB for READ/WRITE operations on directory files.

6.1 Service Commands

HELP [,<command name abbrev>]

HELP without parameter will print a description of all implemented commands in alphabetic order.

HELP,<command name abbrev> will list all commands that satisfies the abbreviation.

STATUS

List the current status of ALICE. The status include date and time, name of experiment, value of symbolic command parameters, reserved file units, spectrum configuration, and data acquisition status.

6.2 Command Processor Control

MODON

Set the command processor in MODAL mode. After this command has been given all command strings are executed as MODAL program statement(s). A command string may contain multiple statements delimited by semi-colons. However, no MODAL line numbering is permitted. Neither can a MODAL command string contain the character ":".

MODOF

Reset the command processor to normal command mode.

LBEG,<n>

Define beginning of repetetive command sequence. The following commands up to LEND will be executed <n> times. Command loops can not be nested. However, a command program that itself contains a loop can be called from inside a loop in a terminal command.

LEND

Terminate repetetive command sequence.

MØED

Enter MODE quick editor. By means of this editor a command program can be generated and edited. The use of MØED is selfexplanatory. The call of MØED is not permitted inside a command program.

MODE [,<filename>] [,<command print>]

Execute command program stored on file. If <filename> is omitted the default command program file is used. Otherwise, <filename> may be any file of type :MOD. If <command print> is non-zero each command will be printed before execution, if <command print> is zero or the parameter is omitted the printout will be suppressed.

A command program is stored as elements of a MODAL string array.

MODE is not permitted within a command program.

```
MODE           : execute command program on default file
MODE,,1        : execute same program with command printout
MODE,ANGLE     : execute command program on file (RT)ANGLE
MODE,ANGLE,1   : execute with printout of each command
```

ENTER,<filename> [,<parameter list>]

Execute user command stored on the file <filename>. The <parameter list> may contain up to 5 parameters, separated by comma or space.

The following rules apply for user defined MODAL command programs.

i) The program must be contained within the line numbers 90.01 - 94.99.

ii) The program module is entered as a subroutine in the first line of group 90, i.e. DO 90.

iii) The program module is entered with the parameters stored in the global variables ARG(10) to ARG(16). ARG(10) is equal to the number of parameters, including <filename>. ARG(11) to ARG(16) will contain the evaluated parameter values, and the string array S.ARG will contain the same parameters in string representation. If a parameter can not be evaluated as an expression, -9999 is stored in the corresponding ARG. Since <filename> is always the first parameter the value of ARG(11) will become -9999.

iv) Return to the command processor is done explicitly by RETURN or implicitly when the last line of group 90 has been executed. If a fatal error has been detected, for instance in the parameter list, the global error flag EF must be set to 1 before return.

v) The name of temporary variables must start with the character "Q". No ALICE variables have names starting with this character. It is strongly recommended that temporary variables are erased after use. The author of the program has the full responsibility for that ALICE program variables are not destructively affected.

vi) The program must not contain the NODAL command END.

vii) Execution of the program will be terminated and control returned to the command processor if a syntax error is detected.

```
ENTER,(INGEBR)FWHM,435,0.2187

     ARG(10) = 3
     ARG(11) = -9999        S.ARG(1) = "(INGEBR)FWHM"
     ARG(12) = 435          S.ARG(2) = "435"
     ARG(13) = 0.2187       S.ARG(3) = "0.2187"
```

PAUSE

Exit from ALICE in "pause" mode. All information is retained such that ALICE can be restarted in the same state as it was before the command. Reserved I/O devices and files are released, but they will be automatically re-reserved at restart. A started data acquisition will continue.

ZZZZ

Terminate the run and exit from ALICE. All resources will be released.

6.3 Declaration Commands

DEFINE,<spectrum no>,<length>,<precision>

Define spectrum <spectrum no> with length <length> and precision <precision>. The channels of the spectrum are numbered from 0 (zero) to <length>-1, and the precision parameter <p> is 1 and 2 for single and double integer precision, respectively. The value of <length> must be 1024, 2048, 4096 or 8192.

The total number of spectra and the maximum spectrum size are displayed by the STATUS command. A spectrum can not be redefined during data accumulation.

```
DEFINE,2,4096,2
```
Define spectrum 2 to be 4096 channels in double precision.

EXPNAME,<experiment name>

Define the name of the experiment. If defined, the string <experiment name> will be written onto all output files under the keyword !NAME=, cf. Ref. 6.

EXPNAME without parameter will clear the name.

```
EXPNAME,72Ge(alpha,p) - FI/POT/JR      - define name
EXPNAME                                - clear name
```

ARGSET,<argno>,<value>

Assign <value> to symbolic command argument A<argno>.

```
ARGSET,1,2              - A1:=2
```

ARGCHANGE,<argno>,<amount>

Change current value of A<argno> by <amount>.

```
ARGCHANGE,1,-1          - A1:=A1-1
```

6.4 Display Commands

These commands can be divided into three classes:
i) the DISPLAY command which selects the spectrum (or spectra) to be displayed
ii) commands that changes the layout on the screen, and
iii) display system allocation commands.

Note the non-standard syntax of the CHANNELS command.

DISPLAY [,<list>]

Select spectra for display, <list> stands for any sequence of spectrum numbers. The maximum length of <list> is the total number of spectra. If <list> is empty the current display is refreshed. Note that the first spectrum number in the <list> also defines the "current spectrum".

```
DISPLAY              : refresh current display
DISPLAY,2            : display spectrum 2
DISPLAY,1,2,3,4      : display spectra 1, 2, 3 and 4.
```

CHANNELS [,<ch1>,<ch2>] [,<spectrum no>]

Set channel limits of "current spectrum" or <spectrum no>. Default is "current spectrum". The command has no effect if <spectrum no> is not on display. Note that if more than 512 channels is displayed an averaging procedure is used. For instance, if 1024 channels are selected, only every second one is actually displayed with a value which is the average of two neighbouring channels. This averaging procedure may distort the spectrum somewhat, particularly in regions with poor statistics.

```
CHANNELS             : display complete "current spectrum"
CHANNELS,500,1500    : chs. 500 to 1500
CHANNELS,,1200       : set new high limit to ch. 1200
CHANNELS,400,,       : set new low  limit to ch. 400
CHANNELS,100,700,3   : chs. 100 to 700 of spectrum 3
```

MLOW,<channel>

Set Low marker in "current spectrum" in channel number <channel>. A marker is displayed as a vertical bar, and the two markers in a spectrum are referred to as the Low and High marker. By definition, the Low marker is always the one with the lowest channel number.

MHIGH,<channel>

Set High marker in "current spectrum" in channel number <channel>.

SCALE,<scale factor> [,<spectrum no>]

Set full scale factor for "current spectrum" or <spectrum no>, default is "current". The scaling is linear.

```
SCALE,500        : set full scale to 500
SCALE,2000,2     : set full scale to 2000 for spectrum 2
```

DSCHS

Display "current spectrum" in channel representation, i.e. channel content versus channel number. This is the default display mode.

DSKEV

Display "current spectrum" in energy representation, i.e. channel content versus KeV. Note that the parameters for CHANNEL and Marker commands are given in channel numbers irrespective of the display mode. The CALCHANNEL command will move the Low marker to the channel which corresponds to a specified energy.

DS-ON [,<display id>]

Connect display output to <display id>, <display id> may be DICO-1 or DICO-2. The default value of <display id> is defined in the program, and is used at start-up.

DS-OFF

Disconnect and release current <display id>. The display system may now be used by other programs.

6.5 Spectrum Handling

For the commands INTEGRATE, LIST and ZERO the spectrum number and the first (<ch1>) and last (<ch2>) channels affected by the operation are either given in the command or takes on default values from the number of the "current spectrum" and the channel position of the Low and High display markers, respectively.

The ARITHMETIC command always operates on complete spectra.

INTEGRATE [,<spectrum no> [,<ch1>,<ch2>]]

Integrate spectrum and print result on terminal.
 i) "current spectrum" from Low to High display marker
 ii) <spectrum no> from first to last channel
iii) <spectrum no> from channel <ch1> to channel <ch2>

After a background fit an INTEGRATE command will give both the total and background corrected counts.

```
INTEGR,1,200,300 - Integrate spectrum 1 between chs. 200
                   and 300, both inclusive.
INTEGR,2         - Integrate spectrum 2.
INTEGR           - Integrate "current spectrum" between
                   display markers.
```

LIST [,<spectrum no> [,<ch1>,<ch2>]]

List spectrum on terminal.
 i) "current spectrum" from Low to High display marker
 ii) <spectrum no> from first to last channel
iii) <spectrum no> from channel <ch1> to channel <ch2>

ZERO [,<spectrum no> [,<ch1>,<ch2>]]

Zero content of spectrum.
 i) "current spectrum" from Low to High display marker
 ii) <spectrum no> from first to last channel
iii) <spectrum no> from channel <ch1> to channel <ch2>

It is not possible to zero a spectrum that is being accumulated.

ARITHMETIC,<dest. spectrum>,<operand 1>,<op>,<operand 2>

Perform arithmetic operations on a spectrum or between two spectra, store result in <dest. spectrum>.

The following operations have been implemented

<op> = + (pluss)
<operand 1> and <operand 2> denote source spectra
<dest. spectrum>:=<operand 1> + <operand 2>

<op> = - (minus)
<operand 1> and <operand 2> denote source spectra
<dest. spectrum>:=<operand 1> - <operand 2>

<op> = * (multiply)
<operand 1> denotes source spectrum
<operand 2> denotes multiplication factor
<dest. spectrum>:=<operand 1> * <operand 2>

<op> = < (compress)
<operand 1> denotes source spectrum
<operand 2> denotes compression factor
<dest. spectrum>:=<operand 1> < <operand 2>

The length and precision of the destination spectrum corresponds to the maximum va[illegible] the source spectra. The command may therefore [illegible]ne the layout of the destination spectrum.

Due to memory space limitations the arithmetic operations are carried out by an associated RT program, which exchanges data with ALICE via files. Hence the execution time is of the order of 10 seconds.

```
ARITHMETIC,1,2,+,3       : spectr.1:=spectr.2+spectr.3
ARITHMETIC,1,1,-,2       : spectr.1:=spectr.1-spectr.2
ARITHMETIC,2,2,*,0.487   : spectr.2:=0.487*spectr.2
ARITHMETIC,3,2,<,2       : spectr.3:=spectr.2 compressed 2
```

6.6 Data Acquisition

The ADCs are numbered 1, 2, ... , cf. STATUS command. The necessary hardware initialisation are done by ALICE during start-up.

The start and stop of data acquisition is synchronized by a master gate signal from the current integrator control. The start-stop procedure is described in section 3.2.

START,<ADC no>,<spectrum no>

Start data accumulation from ADC to spectrum <spectrum no>. The initial content of the spectrum array is copied to the "image" in the data acquisition area (cf. section 3.1) before the accumulation is started. Each time the spectrum is referenced by a command the content of the "image" is copied to the spectrum array.

Note that it is not possible to ZERO a spectrum which is being accumulated.

STOP [,<ADC no>]

Stop data acquisition from all connected ADCs or from ADC <ADC no>, default is all.

6.7 Energy Calibration

Energy calibration can be established for each of the defined spectra. The energy is calculated from the expression

$$E(keV) = a0 + a1*X + a2*X**2$$

where X stands for the channel number.

The coefficients a0, a1 and a2 can either be determined from a fit to defined calibration points, or typed in directly. The coefficients are stored as part of the spectrum data area, cf. section 2.

Note that the CALINPUT command can have zero, one or two parameters.

CALINPUT,-1

Delete all defined calibration points. It is not possible to delete individual points.

CALINPUT,<channel>,<keV>

Define another calibration point. A maximum of 10 calibration points can be entered.

CALINPUT

List currently defined calibration points.

CALFIT,<spectrum no>

Determine the calibration coefficients from a least squares fit to the calibration points, and transfer the coefficients to spectrum <spectrum no>.

A command sequence for establishing an energy calibration from typed-in calibration points will typically be:

```
CALINPUT,-1              % delete previous points
CALINPUT,423.5,667       % point 1
CALINPUT,744,1172        % point 2
CALINPUT,846.3,1332      % point 3
 ..................
CALFIT,2                 % determine coefficients and
                         % transfer to spectrum 2
```

After this sequence more calibration points can be entered and the fit repeated.

CALCOEFF,<spectrum no>,<a0>,<a1>,<a2>

Input calibration coefficients directly to spectrum <spectrum no>.

CALPRINT,<spectrum no>

Print the calibration coefficients for spectrum <spectrum no>.

CALENERGY,<channel>

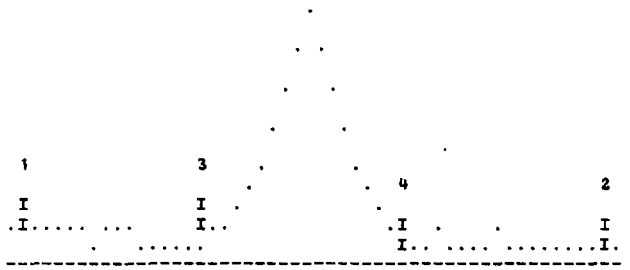
Calculate energy in keV for channel number <channel> in "current spectrum".

CALCHANNEL,<keV>

Calculate channel for energy <keV> in "current spectrum". If the channel is within the displayed part of the spectrum the channel is identified by the Low marker.

6.8 Background Fit

A linear background under peaks in the "current spectrum" can be determined from a least squares fit to the spectrum data.



In this figure the marker pairs 1-2 and 3-4 denote the outer and the inner limits of the fit, respectively. These limits are defined by means of the display markers, cf. description below. The fit is inclusive the marker channels, except in cases where an inner and an outer marker coincides, i.e. if 1=3 or 4=2.

The procedure for a background fit is as follows. Firstly, place display markers in outer limits and give the command BLIMITS. Then move the markers to the inner limits and give the command BFIT. The fitted background is displayed. Subsequent SCALE and CHANNELS commands will regenerate the background. It is erased by a DISPLAY command.

After a background fit an INTEGRATE command will give both the total and background corrected counts.

BLIMITS

Define outer limits for background fit as the current positions of the display markers. The length of the background are can not exceed 500 channels.

BFIT

Fit a linear background in "current spectrum". Prior to this command the outer limits must have been defined by the BLIMIT command. The chi-square of the fit is printed, and the background is displayed.

A fitted background is erased by the first DISPLAY command.

6.9 File Input/Output

Spectrum data are written onto file in the Nordic Standard Format for spectrum files, cf. Ref. 6. Although this standard pertains only to 9-track magnetic tape with recording density 800 or 1600 BPI, the standard is adopted as a general format for all file input/by ALICE.

Normally spectrum data are read/written from/to sequential files on magnetic tape or floppy disc. The total amount of data that can be stored on a floppy disc corresponds to approximately eight 8K spectra, the actual limit depends on the content of the spectra.

It is also possible to read/write from/to directory files, i.e. files on disc or floppy disc belonging to a specific user. In this case the file must either belong to the user RT or to a "friend" of RT. A directory file is identified by inserting the character * before the file name, see below. (A floppy disc may either be used as a directory or as a sequential device. In both cases the disc must be formatted before use, but as a file directory the directory name and user name must be created and file space given to the user.)

The actual device/file to be used is specified by the parameter <file id>, cf. the following description of the commands FFILE, READ and WRITE.

Magnetic tape	<file id> = MT1 and MT2
Floppy disc	<file id> = FL
Directory file	<file id> = *<file name>

The parameter <file name> stands for a full SINTRAN III file name.

Spectrum data can only be read/written from/to sequential files on magnetic tape or floppy disc after the first free file number has been declared by the FFILE command. Writing always takes place to the first free file, and the parameter is incremented by one after the operation. For reading, the file number must be explicitly given in the command. File numbers are counted from 0 (zero). The current values of the first free file numbers are shown by the STATUS command.

The magnetic tape and floppy disc units are automatically reserved by the first READ/WRITE command. Reserved devices are released when the program terminates, or by means of the FFILE command.

The recording density on magnetic tape is selected by the "1600" push button on the tape station.

FFILE,<file id>,<first free file>

Declare the value of the first free file on magnetic tape or floppy disc. If <first free file> = -1, the device will be released.

FFILE,MT1,0
Set the first free file on magnetic tape 1 to 0, i.e. to beginning of tape.

FFILE,FL,-1
Release floppy disc.

WRITE,<file id> [,<spectrum no>]

Write spectrum data onto file. If <spectrum no> is omitted the "current spectrum" is taken as source spectrum.

WRITE,MT2,3
Write spectrum 3 onto first free file on magnetic tape 2.

WRITE,FL
Write "current spectrum" onto first free file on floppy disc.

WRITE,*(RT)SPECDATA-1,2
Write spectrum 2 onto the directory file (RT)SPECDATA-1. The default file type is :SYMB.

READ,<file id>,<file no> [,<spectrum no>]

Read spectrum data from file to destination spectrum <spectrum no> or "current spectrum". The parameter <file no> is irrelevant for directory files.

Note!
This command may redefine the length and precision of the destination spectrum. This will happen if the parameters are less than those specified in the file header.

```
READ,FL,17
Read file 17 on floppy disc to "current spectrum".

READ,MT1,2,4
Read file 2 on magnetic tape 1 to spectrum 4.

READ,*(RT)SPECDATA-1,,1
Read data from directory file to spectrum 1.
```

6.10 Plot Commands

PLOT,<spectrum no>,<ch1>,<ch2>,<scale factor>,<chs per cm>

Plot spectrum on CALCOMP plotter. The full scale of the plotted spectrum is <scale factor>. The number of channels per cm is given by the parameter <chs per cm>, the maximum value is 40 (the plotter step is 0.25 mm).

The plotting is done by a RT program which is started from ALICE. It is possible to "queue" one plot, i.e. another PLOT command can be given whilst the previous one is being processed. Note however that no error message will be given if a non-processed plot is being overwritten.

```
PLOT,1,0,1023,5000,20
```
Plot spectrum 1 from channel 0 to 1023 with scale factor 5000 and 20 channels per cm.

TPLOT,<spectrum no>,<ch1>,<ch2> [,<full scale>]

Plot spectrum on terminal as a histogram. If <ch2> - <ch1> is more than 64 channels, the routine will plot every <n> channels, where n = 2, 3, 4,.... If the parameter <full scale> is omitted, the routine will calculate the linear scaling from the the maximum value found in the plotted channels. If <full scale> is negative, a logarithmic scaling over 3 decades will be used.

```
TPLOT,1,100,163        : linear full scale calc. from data
TPLOT,1,100,163,5000   : linear full scale = 5000
TPLOT,1,100,163,-3     : log scale 10**0 to 10**3
TPLOT,1,100,163,-5     : log scale 10**2 to 10**5
```

6.11 Log File

By default output from ALICE is printed on the terminal. Optionally some of the terminal output can also be written onto a logfile if a more permanent storage of the information is required.

The criteria used for logfile output are somewhat fortuitous, but in general only data and results produced by commands are copied. Typical examples are output from INTEGRATE, LIST and CAL- commands. Neither command strings nor syntax error messages are copied. (This limitation is partly imposed by the scarcity of disc file space).

The logfile has been created with a fixed maximum length. The message LOG FILE ERROR signals that the file is full, and must be rewound by means of the LOG-REWIND command.

LOG-REWIND

Rewind logfile. As the logfile is opened in "Append" mode, this rewind command must be used if previous information is to be deleted. LOG-REWIND can not be executed on an opened file.

LOG-ON

Open logfile. Data copied to the logfile will be added to the current content of the file. The name of the logfile is printed out.

LOG-OFF

Close logfile. The logfile is also closed by the commands PAUSE and ZZZZ.

REFERENCES

1. M.C. Crowley-Milling and G.C. Shering,
THE NODAL SYSTEM FOR THE SPS, CERN 78-07.

2. T.B. Skaali, The SINTRAN III NODAL system,
Report 80-17, Institute of Physics, University of Oslo.

3. SINTRAN III - User's Guide, Publ. no. ND-60.050.08

4. G. Midttun, An ADC-CAMAC interface, Internal Note.

5. P.S. Anderssen, The DICO Display Controllar,
CERN SPS/AOP/CO/Note/77-47.

6. Nordic Standard Format for Industry Compatible
Magnetic Tapes Containing Nuclear Experimental Data.
Edition: 16.01.1978.

APPENDIX A

THE A L I C E PROGRAM FILES

The program and data files are stored under the user (ALICE). As the program is executed as a RT program which logically "belongs" to the user (RT), the user (RT) has been declared as a "friend" of (ALICE) to allow writing onto the files during program execution. For a deeper understanding of these concepts the SINTRAN III Users Guide should be consulted, cf. Ref. 3.

A short description of the most important files is given below.

File : (ALICE)ALICE:NOD
Main program

File : (ALICE)SETUP:NOD
Initialization program executed in overlay. Defines command names and descriptions, global variables and parameters, and initializes the hardware.

Files : (ALICE)ARITH:NOD
(ALICE)BFIT:NOD
(ALICE)CALFIT:NOD
.....
File resident command routines. The filenames have been chosen to resemble the corresponding command word.

Files : (ALICE)SPECT-1:NOD
(ALICE)SPECT-2:NOD
(ALICE)SPECT-3:NOD
(ALICE)SPECT-4:NOD
Data area files; i.e. each file contains the spectrum data array + information array + energy calibration coefficients for one of the defined data areas.

Files : (RT)RTN1:NOD
(RT)DATA-RTN1:NOD
The file (RT)RTN1 contains the NODAL program code for the ARITHMETIC operation RT program. The program is executed as a file-driven RT-NODAL. The parameters are transferred via (RT)DATA-RTN1.

Files : (RT)RTN2:NOD
(RT)DATA-RTN2:NOD
The file (RT)RTN2 contains the file-driven RT-MODAL program for the Calcomp plotter. Data and plot parameters are transferred via (RT)DATA-RTN2.

File : (ALICE)PFILE:NOD
If ALICE is terminated by the PAUSE command all variables are written onto this file. When ALICE is restarted the variables will be reloaded. Note that it is the value of the variable PAUSE. which determines whether ALICE has been terminated by the ZZZZ or PAUSE command.

File : (ALICE)MFILE:NOD
Default file for MODE command programs.

File : (ALICE)CFILE:NOD
Contains the command description printed out by the HELP command.

File : (ALICE)LOG:SYMB
Log file used by the LOG command.

File : (ALICE)CLEARUP:NOD
Program to be executed after an ALICE abort. It releases all resources and resets the hardware. Furthermore it contains a section which can be executed for definition of hardware addresses. These addresses are written onto the file (ALICE)ADC-DEF:NOD.

File : (ALICE)ADC-DEF:NOD
Contains hardware addresses.

APPENDIX B

PROGRAM STRUCTURE OF A L I C E

ALICE is organized as workspace resident main program plus a set of file resident command routines. Furthermore space is allocated for execution of user defined file resident command routines. The file resident command routines are MODAL program units which use the group numbers listed below.

A Main program

MODAL group numbers 1 - 79 plus 95 - 99
Command processor
Global support routines
Display routines
Resident command routines

B File resident command routines

Loaded into MODAL group numbers 80 - 89

C User defined file resident command routines

Loaded into MODAL group numbers 90 - 94

Use of MODAL global task variables ARG(1) - ARG(16)

ARG(1)	Symbolic command parameter A1
ARG(2)	Symbolic command parameter A2
ARG(3)	Symbolic command parameter A3
ARG(6)	Scratch variable
ARG(7)	Scratch variable
ARG(8)	ODEV (device no) for terminal
ARG(10)	Number of parameters in a command
ARG(11)	Value of parameter 1 (if any)
ARG(12)	Value of parameter 2 (if any)
ARG(13)	Value of parameter 3 (if any)
ARG(14)	Value of parameter 4 (if any)

ARG(15) Value of parameter 5 (if any)
ARG(16) Value of parameter 6 (if any)

The definition of command name strings and global variables are done in the initialisation program SETUP which is executed in overlay during the start-up of ALICE. This program also assigns symbolic names to some of the groups of the main program. This technique makes the program substantially more readable, and is also very useful if the group number has to changed.

The initialisation program also loads the hardware addresses from file and initializes the data acquisition hardware. This address file is generated by the CLEARUP program, cf. Appendix A. For further details see the program listings.

In ALICE, all commands are executed as subroutines by means of the DO statement of NODAL. If errors are detected during the parameter check or execution of the subprogram, return to the main program must be done with the global error flag EF set to 1 (one).

The command processor decodes a command string and stores the parameters values in the global task variables ARG and the local string array S.ARG.

ARG(10) = number of parameters, minimum 0, maximum 6

If ARG(10) > 0,
ARG(11) to ARG(11+ARG(10)-1) will contain the evaluated parameter values, and
S.ARG(1) to S.ARG(1+ARG(10)-1) will contain the same parameters in string representation.

If a parameter can not be evaluated as a valid numerical expression the corresponding ARG will be set to -9999.

The global task variables ARG(1) to ARG(3) contain the current value of the symbolic command parameters A1, A2 and A3.

The implementation of a new command requires :

i) definition of command name, command execution string, and command decriptions in the SETUP program.
ii) Command subroutine, either included in the main program or file resident.

Example
The command SMOOTH,<spectrum>,<poly> is to be implemented as a file resident command. The code is stored on the file (ALICE)SMOOTH:NOD. As a file resident command routine it may use the line numbers 80-89, and it is called from the command interpreter by DO 80.

i) Include command definitions in SETUP program (cf. SETUP program listing for further details).

```
 ...
22.60 $SET CM.S(ARSIZE(CM.S)+1)="SMOOTH"
22.61 $SET CM.X(ARSIZE(CM.X)+1)="LOAD (ALICE)SMOOTH;DO 80"
22.62 $SET CM.D(ARSIZE(CM.D)+1)="Smooth spectrum"
22.62 $SET CM.D(ARSIZE(CM.D)+1)="SMOOTH,<spectrum>,<poly>"
 ...
```

ii) Excerpts of command routine

```
80.01 % SMOOTH
80.10 ... check parameters, GOTO 80.20 if OK
80.15 TYPE "error message"; SET EF=1; RETURN
80.20 ... execute
```

APPENDIX C

A L I C E PROGRAM VARIABLES

The most important program variables are listed below. Typical scratch variables are not shown, neither are temporary variables which are erased after use.

The type abbreviations have the following meaning :

```
F      = MODAL floating variable
DIM    = floating array
DIM-I  = integer array
S      = string variable
DIM-S  = string array
```

Name	Type	Description
CI.C	DIM-I	Command processor control CI.C(1) = processor mode, 1/2 : command/MODAL mode CI.C(2) = MODE printout control, 1 : print CI.C(3) = Break control, 1 : break CI.C(4) = Log file no if > 0
CM	F	Command execution control 1 : in terminal command execution 2 : in MODE command execution
CC	DIM-I	MODAL group line numbers CC(1) = group for terminal command execution CC(2) = group for MODE command execution
CL	F	Line number of current command string, used for execution of command programs, CL=1 for terminal command execution.
CP	DIM-I	Character pointer in current command string CP(1) = for terminal command CP(2) = for MODE command
LC	DIM-I	Command loop control (LBEG...LEND) LC(1,-) : used for terminal command loop LC(2,-) : used for MODE command loop LC(-,1) = 0/1 for outside/inside loop LC(-,2) = loop counter LC(-,3) = line number for start of loop LC(-,4) = char. pointer for start of loop

C.TYPE	F	Command type, -1 : syntax error detected 0 : end of current command string 1 : normal command 2 : NODAL command
CS	S	Contains command string read from terminal
CS.1	S	Contains current command line string, one line may contain several commands
CS.2	S	Contains current command, substring of CS.1
CS.M	DIM-S	Command program array. Each element contains one line of commands
CM.S	DIM-S	Command name array
CM.X	DIM-S	NODAL commands for execution of CM.S
CM.D	DIM-S	Command description, 2 lines for each command
S.COM	F	Sub-command type
CA	DIM-S	Name of symbolic command parameters CA(1)="A1" CA(2)="A2" CA(3)="A3"
S.ARG	DIM-S	Contains the parameters of current command in string representation
EF	F	Error flag returned by command routines 0 : command OK 1 : error, abort command interpretation
PAUSE.	F	Termination parameter written onto the file (ALICE)PFILE. PAUSE.=0; stopped by ZZZZ PAUSE.=1; stopped by PAUSE
DC	DIM-I	Data area information DC(1) = total number of defined spectra DC(2) = spectrum in working area DC(3) = spectrum to be fetched DC(4) = maximum spectrum size in words DC(5) = spectrum backwrite control DC(6) = current spectrum
DI.C	DIM-I	Spectrum information, workspace resident, Contains information for all spectra. DI.C(I,1) = precision for spectrum I DI.C(I,2) = no of last channel DI.C(I,3) = first channel on display DI.C(I,4) = last channel on display DI.C(I,5) = <scale factor>/25

DI.C(I,6) = low marker channel
DI.C(I,7) = high marker channel
DI.C(I,8) = 0/6 for channel/energy display
DI.C(I,9) = energy of channel DI.C(I,3)
DI.C(I,10)= energy of channel DI.C(I,4)

DA	DIM-I	Spectrum array, file resident DIM-I DA(1,X) : single precision DIM-I DA(2,X) : double precision
DI	DIM-I	Spectrum information, file resident DI(1) = precision (1 or 2) DI(2) = no of last channel DI(3) = first channel on display DI(4) = last channel on display DI(5) = <scale factor>/25 DI(6) = low marker channel DI(7) = high marker channel DI(8) = 0/6 for channel/energy display DI(9) = energy of channel DI(3) DI(10)= energy of channel DI(4)
DE	DIM	Energy calibration coefficients (keV units), file resident DE(1) = constant term DE(2) = linear term DE(3) = quadratic term The variables DA, DI and DE together constitute the spectrum data area. At any time the data field contained in the NODAL working area corresponds to the spectrum defined by DC(2). (Not DC(6))!
EXP.N	S	Name of experiment
CAL.N	F	Number of defined calibration points
CAL.C	DIM	Array for calibration point channel numbers
CAL.E	DIM	Array for calibration point energies
B.M	F	Background fit status 0 : undefined 1 : outer limits defined 2 : background fitted
BL	DIM-I	Background outer limit channels
BC	DIM	Background coefficients, BC(1) = constant term BC(2) = linear term
FN.S	DIM-S	SINTRAN file names for magnetic tape and floppy disc.

```
                FN.S(1) = MAG-TAPE-0
                FN.S(2) = FLOPPY-1
                FN.S(3) = MAG-TAPE-1

FN.A    DIM-S   ALICE <file id> names for
                magnetic tape and floppy disc.
                FN.A(1) = MT1
                FN.A(2) = FL
                FN.A(3) = MT2

UN      DIM-I   SINTRAN logical file numbers for
                magnetic tape and floppy disc.
                UN(1) = 32
                UN(2) = 512
                UN(3) = 33

UP      DIM-I   File read/write control .
                UP(1,-) : for magnetic tape 1
                UP(2,-) : for floppy disc
                UP(3,-) : for magnetic tape 2
                UP(-,1) = 0/1 for device released/reserved
                UP(-,2) = first free file
                UP(-,3) = current file

DICO.   F       Device number for connected DICO controller,
                DICO. = 0 if display disconnected

D.M     F       Display mode control,
                D.M = 1: generate display
                D.M = 0: erase    display

DN      F       Number of spectra on display

DS      DIM-I   Displayed spectrum numbers,
                DS(1) = first  spectrum
                DS(2) = second spectrum
                ..... = ...... ........

YM      F       Spectrum layout parameter

AX      DIM     Array for displayed subset of "current spectrum",
                maximum 512 elements

SX      F       Index step value in spectrum array DA
                for mapping of DA into AX

C       F       CAMAC crate number

NA      DIM-I   CAMAC module station numbers

NAL.A   DIM-I   CAMAC module control words,
                format ØXNNNNNAAAAØLLLL
                ØX    = expected ØX response
                NNNNN = station
                AAAA  = default subaddress, usually 0
                LLLL  = graded LAM -1
```

ADC.S	F	ADC system no
AC	DIM-I	Data acquisition control, for spectrum I : AC(I) = 0 : no data acquisition AC(I) = <ADC no> : data acquisition from ADC
SEG.N	F	ADC data segment no
D.FIL	S	Default name for display id
M.FIL	S	Name of default MODE file
L.FIL	S	Name of LOG file
C.FIL	S	Name of command description file
PS	S	String for output to terminal and log file
		The following variables designates NODAL group numbers for general subroutines.
BREAK.	F	Test for Break character, CI.C(3)=1 if Break
PNT.	F	Print string PS on terminal and log file
CSPEC.	F	Fetch spectrum to workspace, entered with DC(3) = fetch spectrum no
USPEC.	F	Copy current DI array to DI.C and set backwrite control DC(5):=1. Must be called if spectrum information is changed
WSPEC.	F	Force write-back of workspace spectrum to file, normally called from CSPEC.
CHPAR.	F	Decode spectrum command parameters of type LIST [,<spectrum no> [,<chn1>,<chn2>]] OK return with EF = 0, and ARG(11) = spectrum number ARG(12) = first channel ARG(13) = last channel Error return with EF = 1
ERR.P	F	Prints PARAMETER ERROR, sets EF:=1
DISPL.	F	Spectrum display subroutine. Sets up display from the value of DM and the content of the array DS

APPENDIX D

N O D A L DATA ACQUISITION FUNCTIONS

A summary of the NODAL data acquisition functions used in the ALICE program system is given below. These functions serve as interfaces between a user program and the data acquisition package which has been added to the SINTRAN III operating system.

The parameters of the functions relate to the SINTRAN III monitor calls of the data acquisition package. These monitor calls are described in Appendix E.

The data acquisition package can support several independent ADC systems. Furthermore the ADC data segment, which is locked in memory as a common data acqusition area, must satisfy the conditions imposed by the package.

```
ADCSEG - enter/clear ADC data segment in PT 3
CALL ADCSEG(<ADC system no>,<segment no>,<flag>)
     <ADC system no> : cf. data acqusition package
     <segment no>    : SINTRAN III segment no
     <flag>          : 1/0 for enter/clear segment
Uses MON 170
NODAL type 11 subroutine
```

```
ADCINI - initialise CAMAC ADC interface
CALL ADCINI(<ADC no>,<crate no>,<NAL>,<par2>)
     <ADC no>        : cf. data acqusition package
     <NAL>           : NAL for CAMAC ADC interface
                       NAL(0-3)   = GL-1 (0-17)
                       NAL(5-8)   = subaddress modifier
                       NAL(9-13)  = station
                       NAL(14)    = X-error
     <par2>          : 0XNAF for register if CDMA used
                       0 (zero) if CAMAC ADC-interface
Uses MON 171
NODAL type 11 subroutine
```

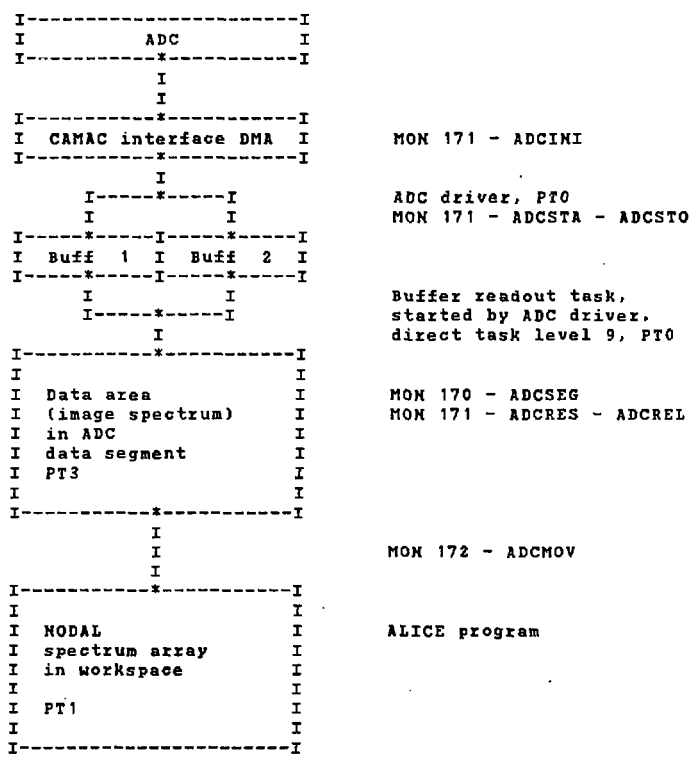
```
ADCRES - reserve data area in ADC data segment
SET Z=ADCRES(<ADC no>,<no of pages>)
      1 page = 1024 words
             = 1024 channels single precision
             =  512 channels double precision
Return value : logical start address of reserved data area
Uses MON 171
NODAL type 10 function
```

```
ADCREL - release reserved data area
SET Z=ADCREL(<ADC no>)
Return value : no of pages released
Uses MON 171
NODAL type 10 function


ADCSTA - start data acquisition from ADC to reserved area
SET Z=ADCSTA(<ADC no>,<precision>)
      <precision> = 1/2 for single/double
Return value : ADC status word
Uses MON 171
NODAL type 10 function


ADCSTO - stop data acquisition from ADC
SET Z=ADCSTO(<ADC no>)
Return value : ADC status word
               bit 0 = 1 : double precision
               bit 1 = 1 : overflow in single precision channel(s
               bit 4 = 1 : hardware error
Uses MON 171
NODAL type 10 function


ADCMOV - move data between ADC data area and NODAL array
CALL ADCMOV(<ADC no>,<NODAL array>,<flag>)
     <NODAL array>    : name of NODAL single/double prec. array
     <flag>           : 0 = move from ADC area to array
                      : 1 = move from array to ADC area
Uses MON 172
NODAL type 11 subroutine


Illegal parameter values or wrong calling sequence (for
instance calling ADCRES before ADCSEG) will cause a NODAL
error return.
```

APPENDIX E

THE SIMTRAN III DATA ACQUISITION PACKAGE

The SIMTRAN III data acquisition package contains drivers and monitor calls for single and multiparameter data acqusition. The basic operation modus is double buffering, e.g. data from an ADC is routed to one buffer whilst the other one is being read out to the ADC data area. Only those parts of the package which relate to single parameter data acquisition will be covered here.



The structure of the data acquisition package is shown schematically above. The indicated operations are all controlled from the ALICE program level by means of the *NODAL functions described* in Appendix D.

The standard usage by SINTRAN III of the four hardware page tables (PT) of NORD-10.S is as follows (each page table is 64 pages = 64 K) :

PT0 : SINTRAN III resident part + system segments
PT1 : RT programs
PT2 : Background user area
PT3 : Reserved for special applications

The ADC data segment is mapped into PT3, and it can therefore be made as big as 64 K. The segment is *accessed* from the buffer readout task and the monitor calls by means of the alternative page table (APT) instructions of NORD-10.S.

The following monitor calls are implemented as socalled user monitor calls, cf. Ref. 3. Return from the monitor calls is skip return if OK, error return is no-skip with an error number in A-reg.

```
MON 170 - enter/clear ADC data segment on PT3
          Data segment attributes :
          1) PT3, Ring 0, read/write permitted
          2) FIX'ed in memory
          3) Address space as defined by acquisition package
Called    T = ADC system no
          A = segment no + flag
              A(0-7) = segment no
              A(15)  = 1/0 for enter/clear
```

```
MON 171 - ADC interface control
Called    T = ADC no + mode
              T(0-7) = ADC no
              T(8-15)= mode (0-4)
Mode 0    Define CAMAC addresses and initialise
          A = NAL for ADC interface or CDMA
              QXNNNNNAAAAOLLLL
          D = crate no
          X = 0 if CAMAC ADC interface
          X = QXNAF for interface reg. if CDMA
          The use of CAMAC ADC interface is default.
          For ADCs connected via register-CDMA
          a flag-bit must be set in the status word of
          the data field, cf. program listing.
Mode 1    Release data area
          Return A = no of pages released
Mode 2    Reserve data area
          A = no of pages
          Return A = logical start address of data area
Mode 3    Stop data acqusition
          Return A = ADC status word
Mode 4    Start data acquisition
          A = 1/2 for single/double precision
          Return A = ADC status word


MON 172 - move between ADC data area and program array
          Data segment must have been entered by MON 170
          and ADC data area reserved by MON 171.
Called    T = ADC no + move flag
              T(0-7) = ADC no
              T(15)  = 0 : move from ADC area to prog. array
                     = 1 : move from prog. array to ADC area
          A = start address in program area
          X = word length
Return    A = number of words moved
```

COMMAND INDEX