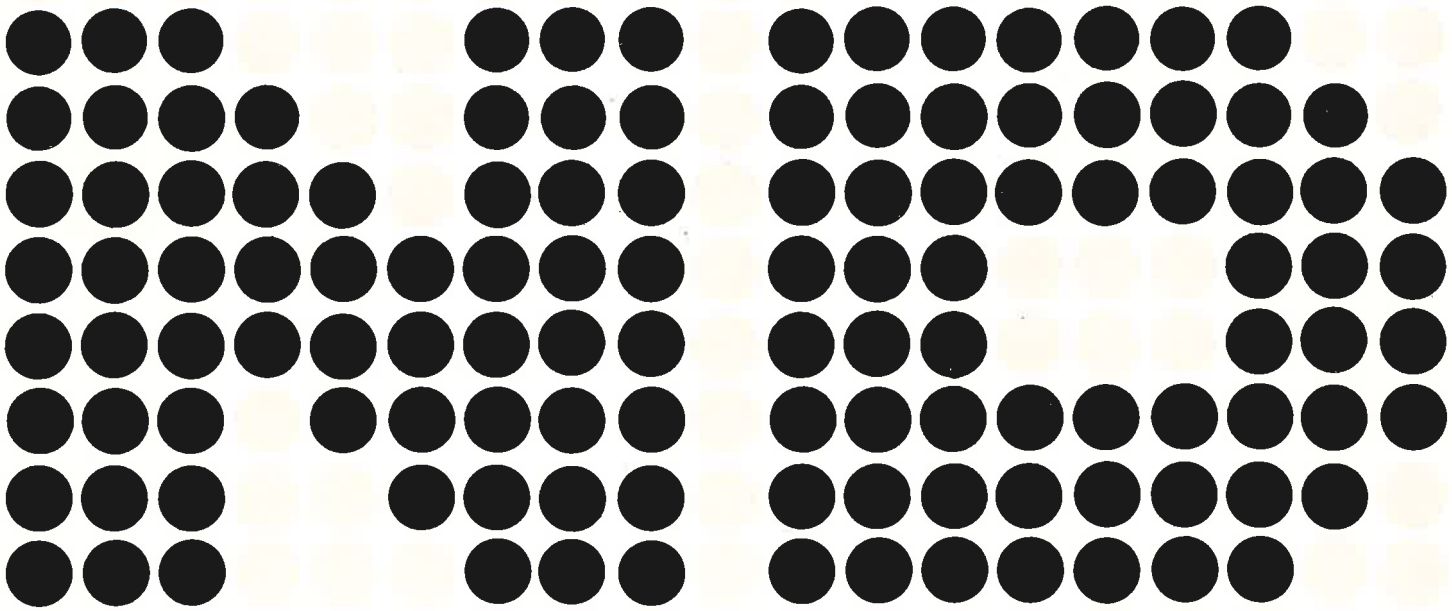
The NORD Screen Handling System

NORSK DATA A.S



The NORD Screen Handling System

REVISION RECORD

Revision	Notes
01/78	ORIGINAL PRINTING
04/79	Second Printing – replaces original printing

NORD SCREEN HANDLING SYSTEM
Publication No. ND-60.088.02

NORSK DATA A.S
Postboks 4, Lindeberg gård
Oslo 10, Norway

PREFACE

This manual explains how formatted screen pictures are created and how they are used in conjunction with application programs that require the use of video display terminals. As a systems analyst or experienced programmer who is acquainted with the idiosyncrasies of video display units, you should feel comfortable with the subject matter. This does not mean, however, that someone with little or no VDU programming experience will find thie manual to be all that difficult. On the contrary, most of the manual is presented in a straight-forward, easy to understand fashion. Given an adequate knowledge of FORTRAN, a familiarity with the SINTRAN III Operating System and a reasonable amount of time to experiment on the terminal, a "first time" programmer should be able to assimilate the Screen Handling System rather quickly.

The manual itself is comprised of five chapters and several indexes. Chapter One introduces the system by highlighting specific design features. Chapter Two tells you how to define a screen picture whereas the remainder of the manual addresses itself to the application of pictures vis-a-vis a user program. Also included is an index in which a COBOL program example is given. It is recommended that at the first reading you complete the entire manual. Later, you will see that whenever you wish the correct format of particular parameter or Call statement the manual will serve you well as a reference guide.

Finally, after you have acquired some practical experience in the use of the system, we would appreciate your suggestions for improving future editions of this manual. The form at the back of the manual can be used for this purpose. In the meantime we wish you "lykke til" (or "good luck" from Oslo) with the NORD Screen Handling System.

TABLE OF CONTENTS

\+ + +

Section: *Page:*

1 INTRODUCTION

If you have the responsibility for implementing a display-oriented application, the NORD Screen Handling System (NSHS) can make your job much easier. How? By eliminating the many hours it would take you to design and program a similar control or 'front-end' module. As a complete and comprehensive I/O system, NSHS can create files of formatted screen pictures, incorporate these pictures into application programs and enable you to manipulate data and pictures on a video display unit. It can accommodate up to eight different types of terminals simultaneously. All or any part of a picture may be edited extensively and can be made to accept or reject specific values. All or any part of a picture may be displayed differently, that is, underlined, blinking, inverse video, etc. What is more, NSHS can either be used with a background program or be integrated into a real-time system which uses many terminals. NSHS is also an integral part of the NORD Data Entry System. What all this means of course, is that you now have, completely operational, a very powerful and sophisticated software product that can serve you as an essential element in your overall system's approach to designing a VDU application.

A closer examination of the system reveals that NSHS is comprised of two functionally distinct subsystems:

— NORD Screen Defintion System (NSD)

— NORD Screen Library System (NSL)

NSD is the interactive utility program that allows you to create and modify pictures at a display terminal. It is this subsystem that is used in connection with NORD DATA ENTRY System.

NSL is the run-time subroutine library that permits application programs to use the picture formats created in NSD. There are two versions of NSL: one for real-time programs, the other for background programs. Pictures and call formats for both versions are completely compatible.

1.1 *GENERAL TERMS*

Every software system contains its own vocabulary: words and phrases used repeatedly, selected to convey quickly and clearly the system designer's original ideas. NSHS is no exception; for example:

Leading Text

. . . . . an informative or explanatory text that is defined when a picture is created. It is a permanent part of a picture and can not be modified by an application program at *run-time.*

Field

. . . . . a number of positions in a picture reserved for the input/output of data. There can be several types of fields, all of which are defined when a picture is created. Fields can be used either to display output from a user's program or to display characters read in from the keyboard, or both.

Field Characteristics

. . . . . such as how a field is to be handled, its maximum length in characters, its format, its edit and control paramerers, are all defined in NSD.

Picture

. . . . . the combination of leading texts and fields, defined in NSD. A picture has an 8 character name which must be unique within one picture file or within any set of pictures used "simultaneously" in one application. A picture cannot contain more than 255 fields.

1.2 *OPERATING PRINCIPLES*

1.2.1 *Picture Files*

A picture once defined is called the "source picture" and is stored in a contiguous SINTRAN file with the default file-type designation (:PIC). Up to 49 pictures identified by their individual picture names may be stored on this file. Before a picture can be used by an application program, it must be compiled into a run-time format. A compiled picture, called the "object picture", is stored on another contiguous SINTRAN file with the default file-type designation (:OBJ). This file may also contain up to 49 pictures. For a more in-depth discussion about file handling *see the NORD File System Manual.*

1.2.2 *Use of Pictures*

Picture Buffers

Each application program using the screen handling facilities must define three data areas adhering to fixed formats.

These data areas are:

1) the terminal array.

2) the private picture buffer. Both the terminal array and the private picture buffer are contained in a FORTRAN labelled common area called PRIVATE.

3) the public picture buffer contained in a FORTRAN labelled common area called PUBLIC.

The terminal array, normally given the name ITERM, contains information related to the type of VDU terminal being used. The private picture buffer, normally given the name IPRIV, contains the object pictures currently being used by a specific application program. The public picture buffer, normally called IPUBL, also contains current object pictures. It has the same layout as the private buffer, but can be accessed by several application programs. By setting a flag in the ITERM data area the application program can switch between the two picture buffers *(See Figure 1).*

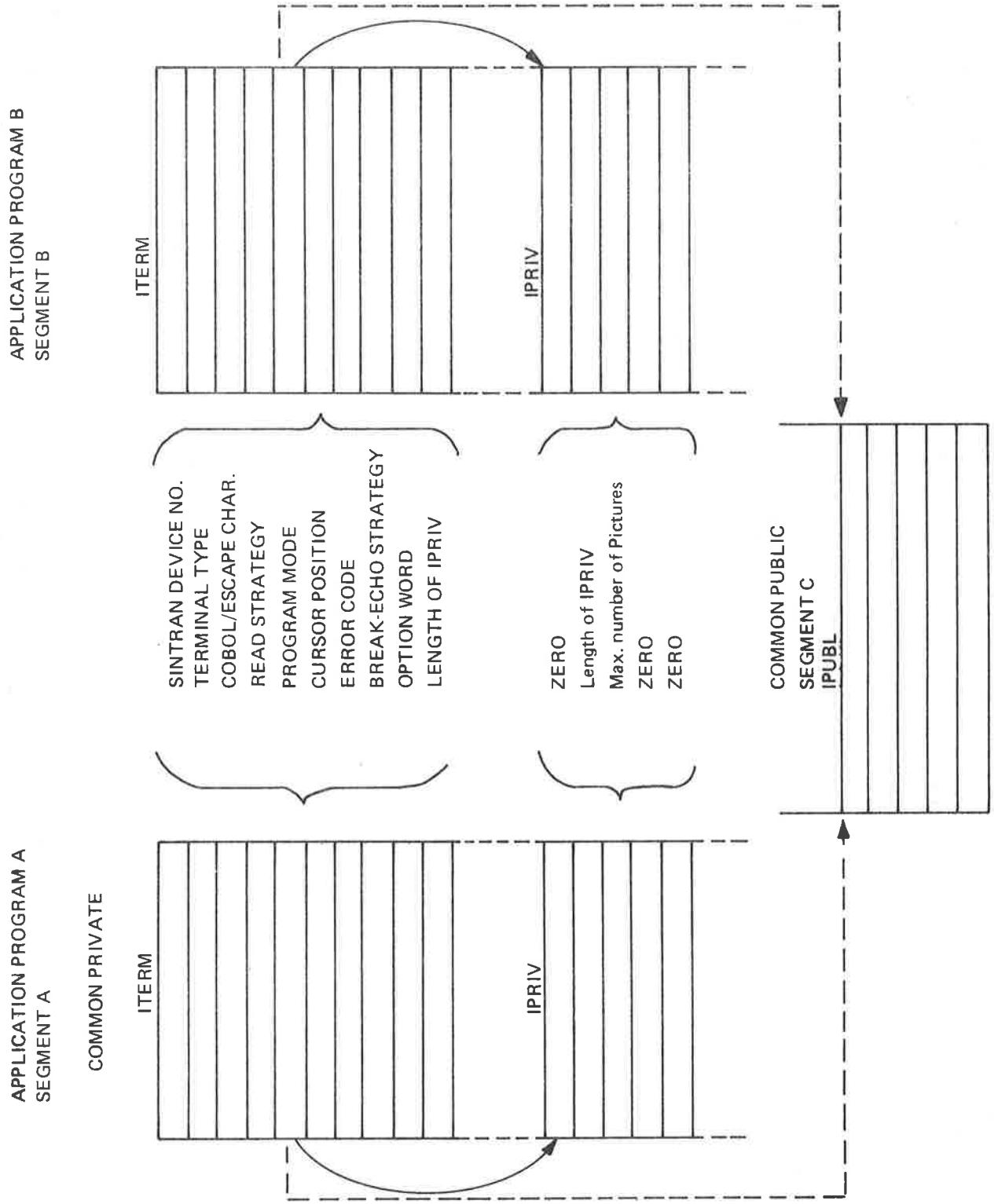


Figure 1: PICTURE BUFFERS USED IN NSHS

Through a subroutine call (GTPIC) the application program can request specific pictures from the object file to be loaded into the picture buffer. *(See Figure 2).*

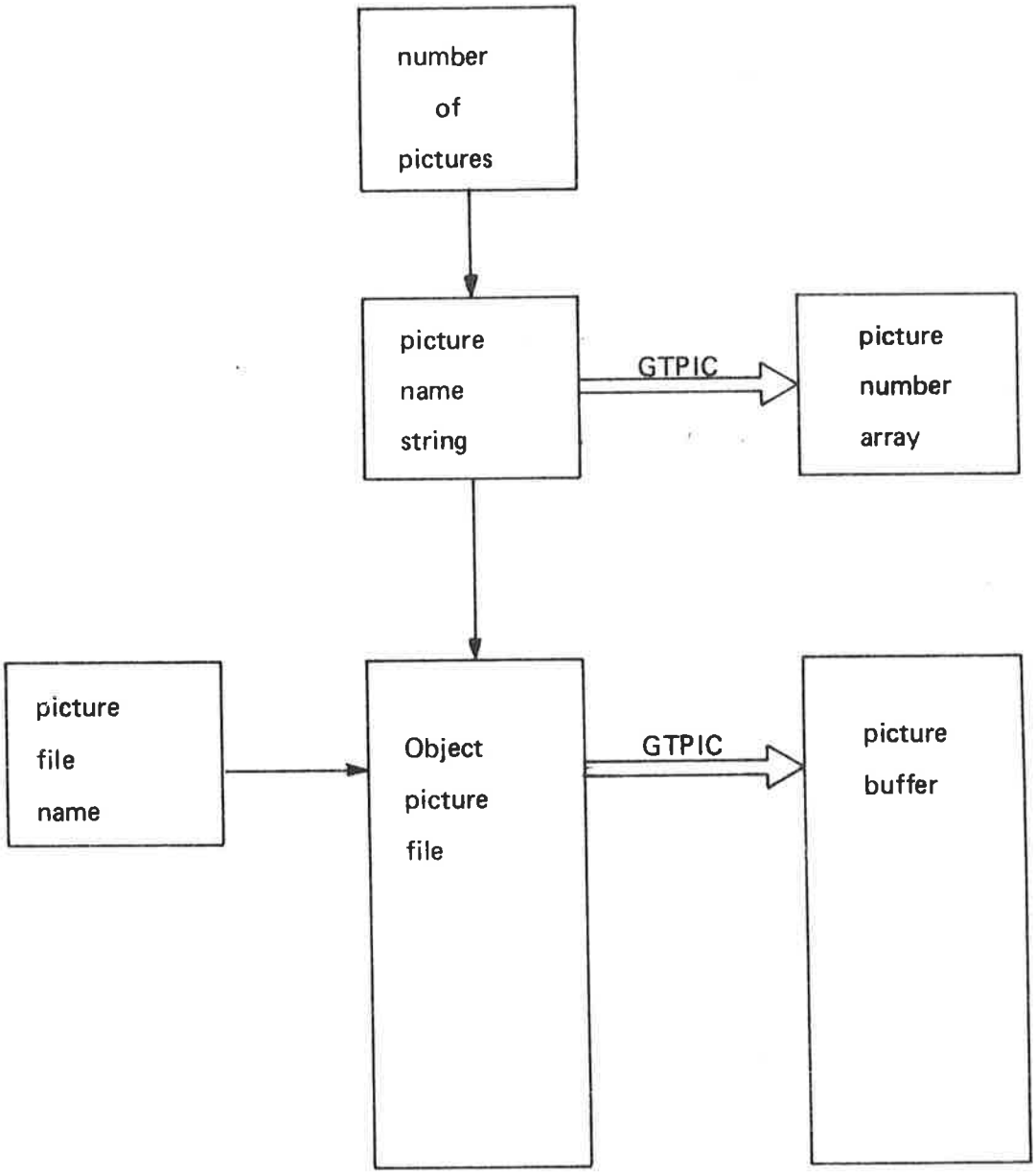


Figure 2: GTPIC SUBROUTINE - loads picture buffer and picture number array

Field Access

All fields in a picture are implicitly assigned absolute field numbers in ascending order starting from line 1, position 1 and ending at the last position of the last line.

Before an application program can access one or more fields it must first create an array of field indicators. This array, called the Field Indicator Array, may for example contain the absolute field numbers desired *(See Section 4.2.6).*

The Field Indicator Array is then passed to a NSL subroutine which converts the absolute numbers to field numbers. These field numbers are placed in another array called the Field Number Array, and it is in this form that the field numbers are stored in the application program *(See Figure 3).*

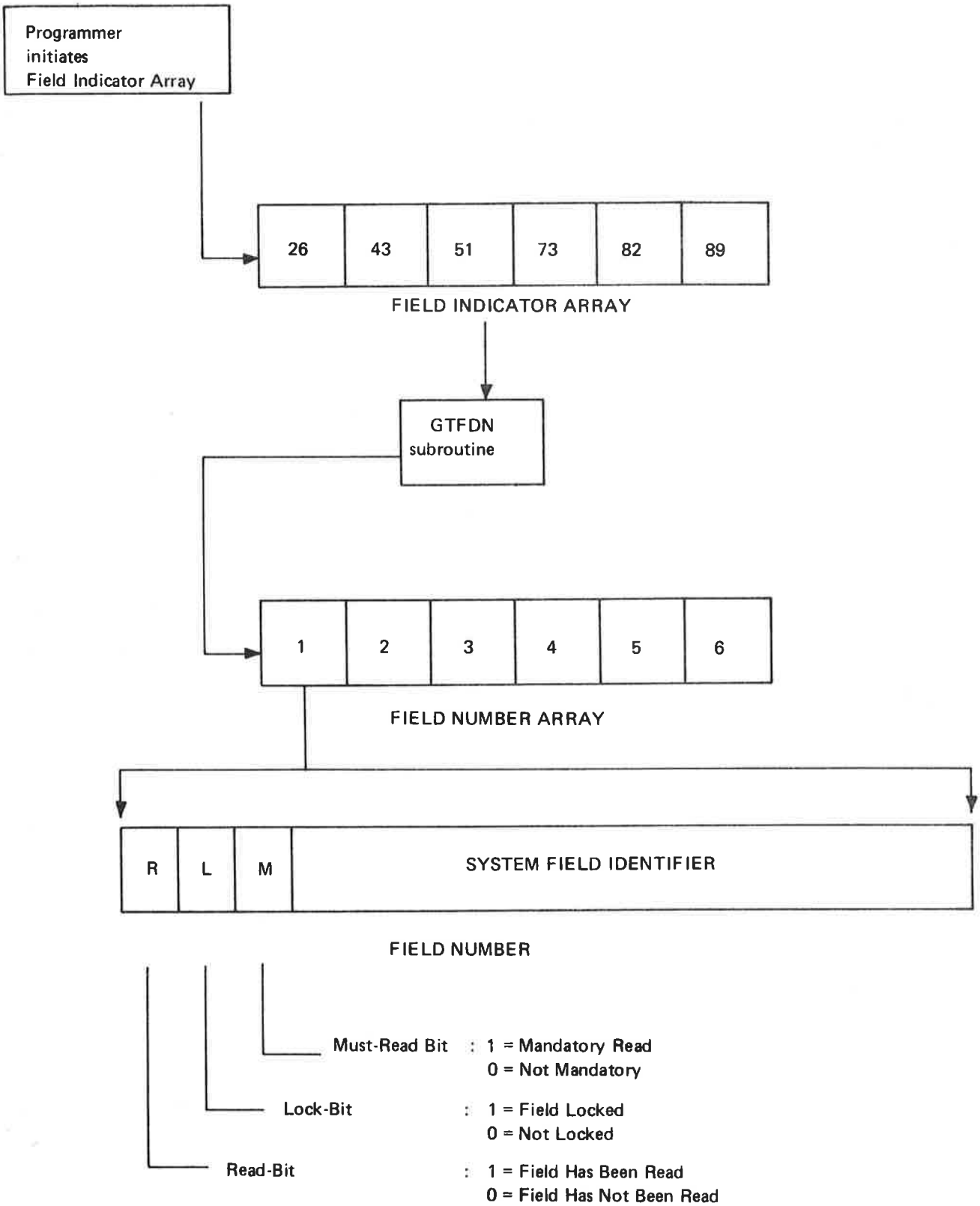


Figure 3: GTFDN SUBROUTINE - transforms absolute screen positions to field numbers

The Field Number Array may then be used as a parameter in calls for input or output of data to or from the display terminal *(See Figure 4)*. The Field Number Array need not specify all fields actually defined in the picture. When reading data from the terminal, it is possible, for the application program to specify how many fields are to be read and at which field the reading is to begin.

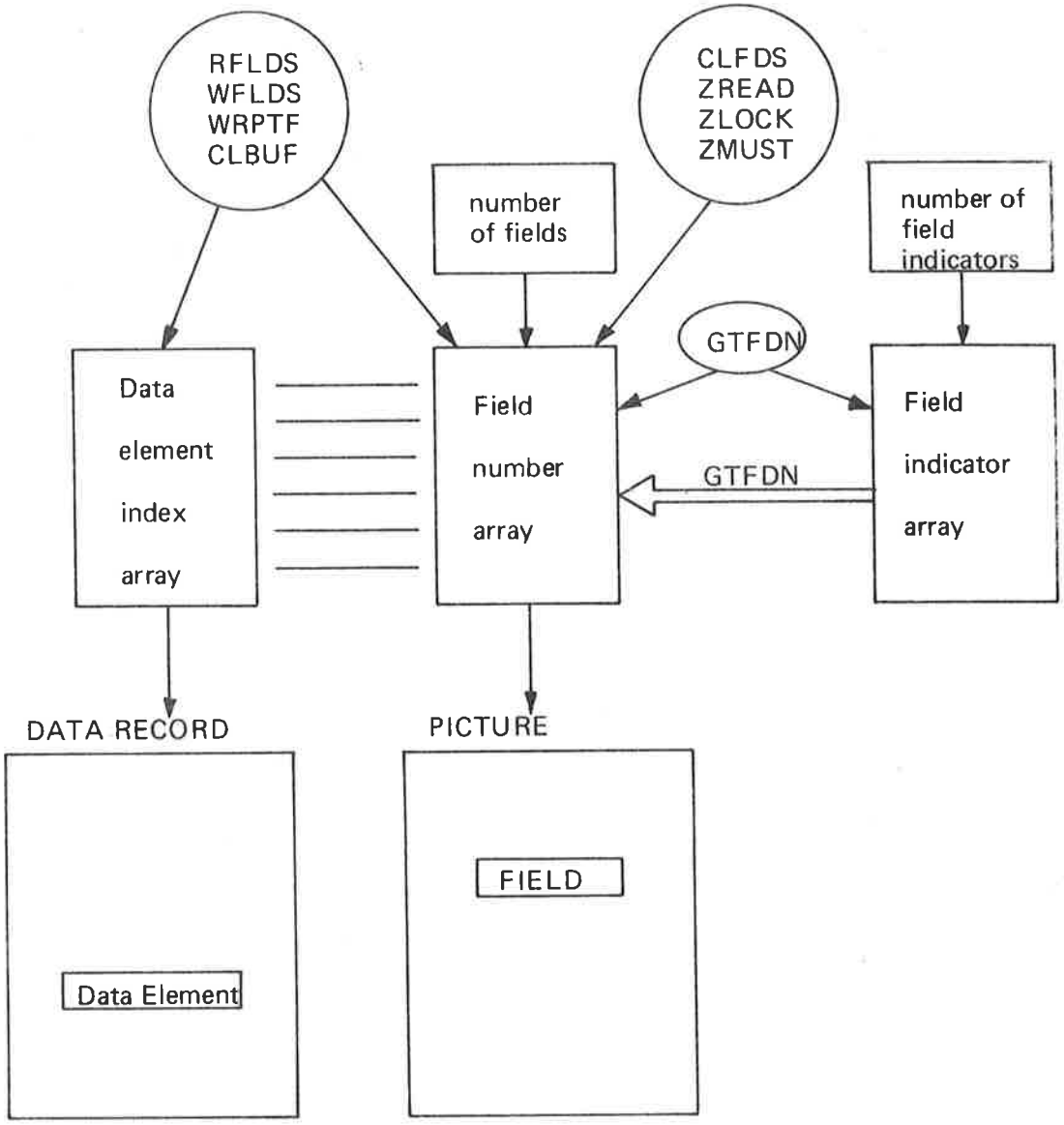


Figure 4: References to tables in NSHS calls

By creating several Field Number Arrays for the same picture, the application program can in effect use the same picture in a variety of ways. The same result can also be obtained by using a single field number array, together with dynamic use of the locking/unlocking facility.

Data Access

In NSL each picture corresponds to a data record, the data record being an integer array. Each field in the picture corresponds to a data element in the record. In writing or reading data to or from the data record, a Data Element Index Array is used. This array can be created by the application program such that the entries in it correspond directly to the entries in the Field Number Array, or, if the first data element entry is given a value of zero NSL will generate a default Data Element Index Array *(See Figure 5).*

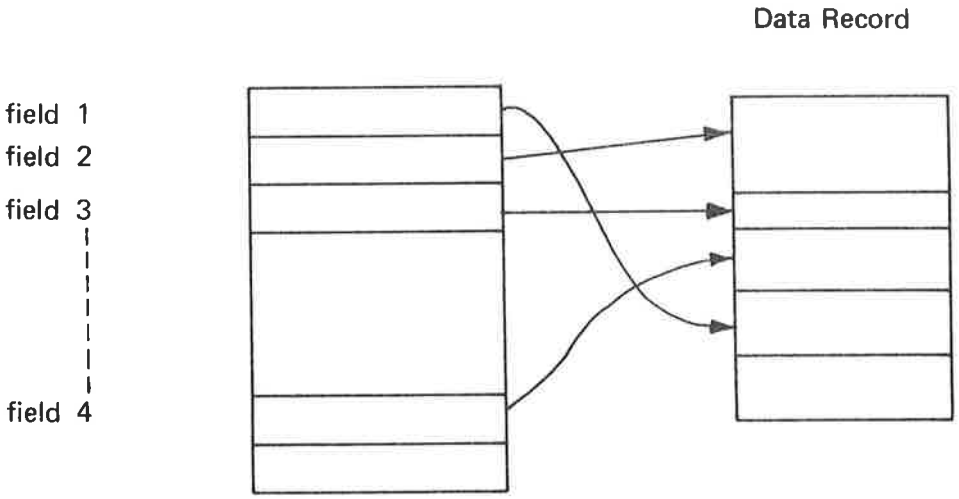


Figure 5: DATA ELEMENT INDEX ARRAY

2 NORD SCREEN DEFINITION SYSTEM (NSD)

2.1 *GENERAL*

The interactive utility program, NSD, is activated by the following SINTRAN command:

@ SCREEN-DEFINITION

Next, a heading page appears and you are asked: Terminal Type?

After this has been answered, the message READY is shown. This message will also appear at the completion of all operator commands. You can then set the program in command-receiving mode by typing a space or carriage return. The text SPM * will appear, signalling that a command may be entered. SPM stands for Screen Picture Maintenance!

The available commands listed below are described on the following pages. All commands may be abbreviated.

AVAILABLE COMMANDS

General Commands

HELP
EXIT
SET-TERIMINAL-TYPE
SET-DISPLAY-MODE
SET-VERIFY
RESET-VERIFY
SET-MUST-READ
RESET-MUST-READ

Picture Commands

CREATE-PICTURE
MODIFY-PICTURE
DELETE-PICTURE
SCRATCH-PICTURE-FILE
COMPILE-FILE
DEFINE-PICTURE-SIZE
RESCUE-PICTURE
DUMP-PICTURE-FILE
DESCRIBE-PICTURE
DESCRIBE-FILE
CLOSE-FILES

2.2 *GENERAL COMMANDS*

2.2.1 *HELP*

A list of all available commands *is written on the terminal.*

2.2.2 *EXIT*

This command stops the picture definition module and *returns* control to the operating system.

2.2.3 *SET-TERMINAL-TYPE*

A text, listing types of terminals available is shown on the terminal, so that the operator may select the correct value. This command is automatically issued at the start of the Screen Definition System.

2.2.4 *SET-DISPLAY-MODE*

Display mode handling is normally set to 1. One space on the VDU screen will then be reserved for the display mode character (i.e., blink, underline, etc.) and a space will be reserved both before and after each defined field. Pictures defined with this display mode are guaranteed to work on all VDU display types (VISTA, TDV 2000/2100, INFOTON 200/400, etc.)

If the display mode is set to 0, no spaces will be reserved and the pictures will not work properly on VDU terminals that require a screen position for display mode characters.

If the display mode is set to 2, no spaces will be reserved and at run time, for those terminals requiring a position on the VDU screen, it will be impossible to use anything other than normal display mode. Note that display mode 2 applies only to fields. It is possible, using display mode 2, to have leading texts in inverse video, underlined, etc.

2.2.5 *SET-VERIFY*

It is possible to have individual verification of fields. If this command is given you must indicate whether or not the field is to be verified. This command is mainly used for pictures defined in the NORD DATA ENTRY system. Unless specifically desired, there will be no run-time effects when such pictures are used from application programs.

2.2.6 *RESET-VERIFY*

This command reverses the action taken in SET-VERIFY.

2.2.7 *SET-MUST-READ*

It is possible to have must-read on individual fields. If this command is given you must indicate whether or not the field is to be a mandatory read. This command is mainly used for pictures defined in the NORD DATA ENTRY system. Unless specifically desired, there will be no run-time effects when such pictures are used from application programs.

2.2.8 *RESET-MUST-READ*

This command reverses the action taken in SET-MUST-READ.

2.3 *PICTURE COMMANDS*

If no files have been opened before entering one of the picture commands, the system will ask for source and object files. Both file name and type must be given. If the files which have been opened are not source/object files, you will be asked if you wish to continue. If "Y" and carriage return is given, these files will then be defined as source/object files. Any other character will give a return to command input mode.

2.3.1 *CREATE-PICTURE*

After this command has been given, the display will be cleared and formatted and the cursor set in the "home" position (top left). You can then define a picture by entering leading texts, headings, and field descriptions. Leading texts, preceded by display mode characters, or positioning spaces are entered directly. Fields are defined by typing a control E. There are many editing facilities which will be described later in this manual.

2.3.2 *MODIFY-PICTURE*

After this command has been given, you will be asked to give the name of the picture to be modified. If found, the picture will then be displayed on the screen, and the cursor returned to the home position. You can then modify the picture *using the same editing and control facilities as in Create-Picture.*

2.3.3 *DELETE-PICTURE*

This command outputs a list of defined pictures from the source and object pictures files and can also be used to delete pictures. You are asked for the name of the picture to be deleted, and you can then either give the name or just type carriage return which will give a return to command input mode.

2.3.4 *SCRATCH-PICTURE-FILE*

This command destroys source and object picture files, thus making them unacceptable for further use. You can choose which file is to be scratched, that is, the source or the object file, or both. Before it completes this task, the system will ask for a final confirmation on your initial decision thereby giving you a chance to change your mind.

2.3.5 *COMPILE-FILE*

This command compiles all the pictures in the source file to the object file. Old object pictures will be replaced, new pictures will be added. Normally, this command is not necessary since pictures are compiled automatically at the termination of the CREATE-PICTURE and MODIFY-PICTURE commands, but it could be very useful if by some accident the object picture file had been destroyed. The only way to compile a single modified picture is to use MODIFY-PICTURE, typing control W immediately afterwards.

2.3.6 *DEFINE-PICTURE-SIZE*

This command specifies the number of lines on the display (within the range of 1—31) and the number of characters per line (within the range of 1—127).

This means that a picture can have a maximum of 31 lines containing 127 characters each. But most display terminals have only 24 lines with 80 characters. (The Tandberg TDV 2100 has 25 lines, maximum line length being 80 characters). Therefore, pictures with more than 24 lines can be written to display terminals but the additional lines will not be shown immediately. (Note that NSHS never uses column 80 of a VDU and the last line on a VDU is used for system purposes which means that the default picture size is 23 lines of 79 characters).

If the number of lines in the picture exceeds the line-capacity of the terminal, the picture is automatically divided into two pages. By continuing to move the cursor beyond the first page, the second page will be automatically displayed, starting at line 1. When the cursor is moved back to the line-capacity, the pages are switched correspondingly.

If the line contains more than 79 characters, the line is split into two lines. Pictures defined with more than 79 characters per line can only be used in the Write Picture To File (WPRTF) command.

2.3.7 *RESCUE-PICTURE*

Having finished with the creation or modification of a source picture and after typing control W, should the source file become full, the picture data remains in core. The source file however, can be expanded and this command, Rescue-Picture, enables you to write the picture back to the original source file, or if desired, to another source file.

2.3.8 *DUMP-PICTURE-FILE*

This command writes out to a specified file a description of how the blocks on the source and object picture files are used; in addition, it shows which pictures are actually stored on the files. It also points out the number of blocks on a picture file. It asks the operator to confirm that this is correct (no input except "CR"), or to give a new value.

2.3.9 *DESCRIBE-PICTURE*

This command generates a description of a picture which is then written to a user selected file. This description contains the picture layout, the size of the object picture, the number of fields defined, and a description of the defined fields.

2.3.10 *DESCRIBE-FILE*

This command is equivalent to DESCRIBE-PICTURE, except that in this instance, all the pictures in the source file are described.

2.3.11 *CLOSE-FILES*

This command closes the source and object files.

2.4 *PICTURE CREATION AND MODIFICATION*

2.4.1 *Editing Facilities*

The editing facilities used in picture creation and modification operate on one line at a time, allowing characters to be copied from the line above, from the old line or from the keyboard. The following control characters are used:

Control A:	(←)	remove previous character or field
Control C:	(→)	copy one character from old line
Control D:	(↓)	copy remainder of old line from current position up to and including the last character in the old line
Control P:		copy one character from previous line
Control Q:		delete characters on this line and return cursor to *position 1* on the same line
Control R:	(↑)	copy previous line up to and including last character

Normally line editing is terminated by a carriage return (CR). When "in" a line, back cursor (←) is equivalent to control A; forward cursor (→) is equivalent to control C. However, when beyond position 1, cursor up (↑) and cursor down (↓) while equivalent to control R and control D respectively, will *also terminate the line.*

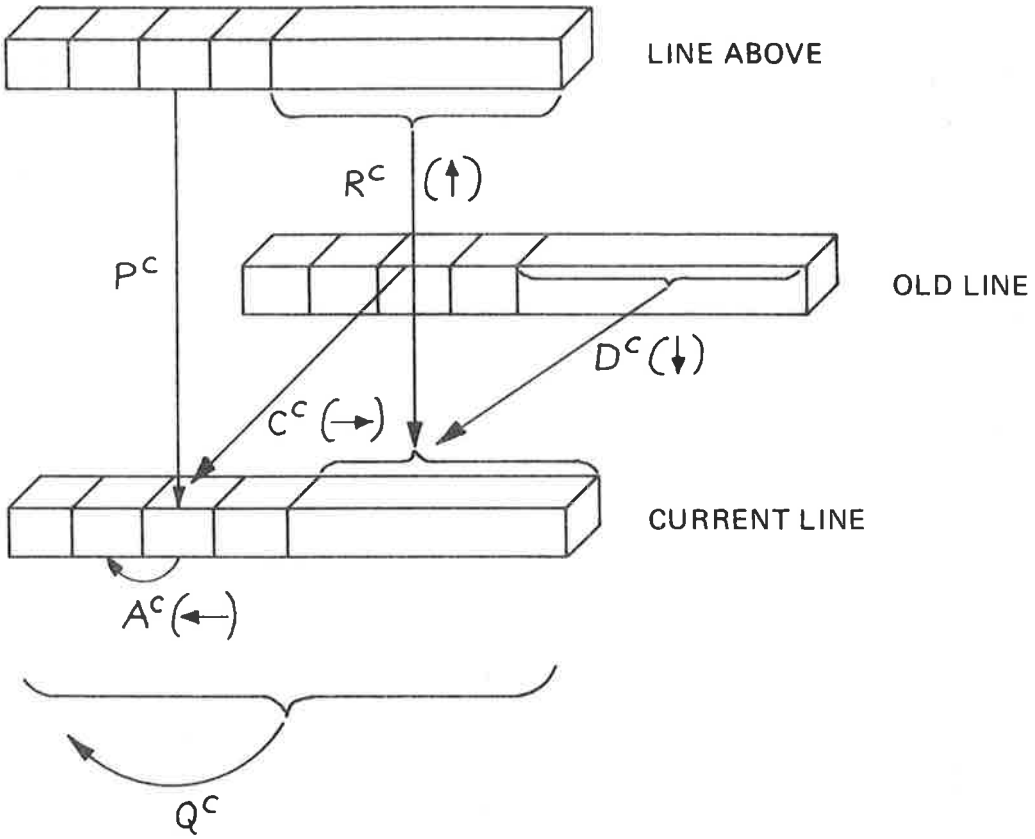
When editing a line, if control A, C, P or ← or → is typed, a field is handled as one character. For example, control A typed immediately after a 10 character's field causes the 10 characters to be replaced by spaces and the cursor to be moved backwards 10 positons. The editing of a line should always be terminated by a CR so that the cursor can be moved to the first position of the next line. The cursor can then be moved to the first position of any line by using the keys up (↑), down (↓), or home (↖). Note that CR always results in the deletion of all characters on a line to the right of and including the cursor position.

Furthermore, a line can be removed or inserted using:

Control J:	inserts one line between the current line and the line immediately above. The current line and all lines below get pushed down a line, and the last line is erased.
Control G:	removes the current line. All lines below will be raised a line, and the last line will become an empty line.

These functions work only when the cursor is at the beginning of a line; once executed they are IRREVERSIBLE.

The previous field, or any field already defined on the line to the left of the cursor can be repeated on the same line by typing home (↖) *(See Figure 6).*



J^C = INSERT NEW LINE

G^C = REMOVE CURRENT LINE

↖ = COPY PREVIOUS FIELD

WHEN CURSOR IN COLUMN 1.

MOVE CURSOR

Figure 6: EDITING CHARACTERS

2.4.2 *Termination of Picture Editing*

Newly created pictures must be given a name. A modified picture can be written back to the original picture it came from, to another picture (thereby replacing it), or to a new picture. Control W indicates that the user is satisfied with the screen picture and wishes to store it. After Control W is typed, the picture is automatically compiled to the object file. When control W is typed, a list file is requested. Should compile errors arise they will be written out on this file. Control W only works at the beginning of a line.

If a modified picture has become too large for its current file, an error message is issued. In this case, the RESCUE-PICTURE command may be used to save the picture.

Control F indicates that you are not interested in storing the picture currently on the screen. Note that control F gives "abort" at any time when outside field definition.

Error Messages

The various error messages are shown in Appendix C.

2.4.3. *Definition of Display Modes*

If the VDU terminal permits, leading texts can be displayed with blink, low intensity, etc. The definition of Display Mode is VDU terminal type independent, and is achieved by typing in one of the following control characters.

Blink	Control B
Low intensity	Control L
Normal	Control N
Invisible ("off")	Control O
Underline	Control U
Inverse video	Control V

A display mode is terminated by giving Normal; by giving a different display mode character; by terminating the line or by defining a field.

2.4.4 *Definition of Fields*

When creating or modifying a source picture, you indicate that you wish to define a field by typing Control E. The system then asks:

Question:	*Possible Answer:*
Storage code	(1–5)
Edit code	(0–9, A–S)
Fill code	(0–1)
Sign supress code	(0–1)
Number of significant characters in the field	(actual number of characters)

If the answers are acceptable and, depending on the particular edit code, the system then writes out the field with *all 9's* or *all A's* or *all X's.* It then asks

Question:	*Possible Answer:*
Field control functions	(0–10)

The next few sections explain these codes and control functions.

2.4.4.1 Storage Code

Storage code defines the internal representation of the characters that will be read in.

Storage code:

1. Single integer, (maximum 5 digits—32768 ⩽ value ⩽ 32767.)
2. Double integer, (maximum 10 digits—2 147 483 648 ⩽ value ⩽ 2 147 483 647.)
3. Byte, (maximum 79 characters.) *
4. "FORTRAN IV", that is, each 16 bit word contains a 4 digit integer corresponding to 4 decimal digits. Maximum 79 digits.*
5. Binary coded decimal, that is, each 16 bit word represents 4 decimal digits, each digit a binary value. Maximum 79 digits.*

*77 if display mode handling is 1

2.4.4.2 Edit Code

Edit codes define how fields are to be presented on the video screen. There are 29 edit codes available. The formats can be altered, within certain limits, to suit a user's needs. *(Refer to Appendix A).*

0—9 for fields which hold digits in 0—9 positions to the right of the decimal/comma and are edited with a decimal point between each three digits. Allowed characters on input are:

the digits 0—9
comma (,)
full stop (.)
minus (—)

Minus sign may be the first or the last character typed, a (.) or (,) define the position of the decimal/comma

A—J DIGIT FIELDS ONLY: 0—9, AND (—) LEGAL

A right justified digits; no editing characters

B edited as Norwegian bank account

C edited with decimal point between each two digits (used for clock/date)

D, E } right justified, set to the same as A

F left justified digits; no editing characters

G—J left justified digits; formats set to the same as F

K—S STRING FIELDS MUST HAVE STORAGE CODE 3

K numeric string, left justified

L alphabetic string, left justified

M alphanumeric string, left justified

N

O same as K, L, M but with edit characters

P

Q

R same as K, L, M but right justified

S

2.4.4.3 Fill Code

Fill code defines how "unused" character positions are filled out on the VDU screen.

Fill code:

0 spaces
1 zeroes

2.4.4.4 Sign Suppress Code

If a digit field will always contain positive values, then the sign position to the right of the field can be discarded.

Sign suppress code:

0 field has sign position
1 no sign position

2.4.4.5 Number of Significant Characters

This is the maximum number of digits that can be typed into a field. Editing characters will often cause this number to differ from the total number of character positions required by the field.

2.4.5 *Field Control Functions*

The control functions must be defined at the time a picture is created. They apply only to the check-record function and to the reading of fields.

Control code:

0 no control

1 default value

2 legal values

3 illegal values

4 legal range

5 illegal range

6 add field/accumulation field

7 equivalent field

8 system defined control algorithm

9 user defined control algorithm

10 combined control

2.4.5.1 Default Value

If the operator passes the field using a cursor control character, or gives the field a terminating character (e.g. CR) without typing in any other character, the field will automatically get the default value. The default value is defined when a picture is created.

2.4.5.2 Legal Values

The values read into the field will be checked against those in the legal values table. If no match is found, a message will be written out, and the cursor will be repositioned to the start of the field.

The number of legal values and the values themselves are defined when a picture is created.

2.4.5.3 Illegal Values

Values will be accepted only if they differ from those in the illegal values table.

The number of illegal values and the values themselves are defined when a picture is created.

2.4.5.4 Legal Range

The only values accepted are those that lie between or are equal to the given limits.

The legal range is defined when a picture is created.

2.4.5.5 Illegal Range

Values must lie outside the defined range, including the upper and lower limits, if they are to be accepted.

The range is defined when a picture is created.

2.4.5.6 Add Field

The value read into this field will be added to another field, namely the Accumulation field.

2.4.5.7 Accumulation Field

Values read into other specified fields are accumulated in this field. It is not possible to read values directly into an Accumulation field. The "start value" of an Accumulation field will be zero or the value represented by its corresponding data element at the time the read fields call is entered (if the field read bit is set).

Accumulation fields cannot be read into on an RFLDS (read fields) call, but they can be written into by a WFLDS (write fields) call. An Accumulation field will accept values from other Accumulation fields. The values of ordinary add fields are added to the Accumulation field directly as they are entered, and the new value of the Accumulation field is displayed. The value of one Accumulation field is not added to another Accumulation field until the RFLDS call is terminated.

All combinations of field types are allowed provided that the number of significant digits in the Accumulation field is sufficient.

2.4.5.8 Defining Add and Accumlation Fields

Control code 6 is used to define both Accumulation and Add fields. The system discerns the type of field in two steps. First, you are asked "Field number of accumulating field or 0 if none". Answer 0 if you are defining an Accumulation field. Answer with an absolute field number if you are defining an Add field. The values in the field that you are presently defining will then be accumulated in this absolute field location.

Next, you are asked " Number of fields to be accumulated or 0 if none". Answer 0 if an Add field; otherwise give the total amount of fields to be accumlated, and then give the absolute field number for these Add fields.

If this procedure is done incorrectly, error messages will not be given until the picture is compiled.

2.4.5.9 System Defined Program Control

Algorithm control functions in general use such as the checking of bank account and person numbers can be incorporated into the picture handling system. These control functions are explained in Appendix D.

2.4.5.10 User Defined Program Control

It is impossible to include all user requirements in a general system. If your application has a specific need for the control of fields as they are read in, you can add your own control algorithm. This should be written as a FORTRAN subroutine with no data statements (if it is to be used reentrantly). It will be called automatically by RFLDS both before a field is entered and after it has been read. The subroutine will be called as follows:

UCONT
(user-control-number, picture-number, field-information-array, number-of-fields, field-number-array, data-record, data-element-index-array, index-to-this-field-number, field-read-bit-array, status)

All these parameters are integer variables or arrays and are described later in this manual. *User-Control-Number* will be the control type number which you define, and it is then up to you to carry out the necessary control function in your program.

An example might be: "multiply the contents of the previous two fields and write it out in this field". If the control accepts the value given, the status returned should be 0, and RFLDS will then write out the field using the value placed in the data element by UCONT. If the value is not accepted but you still wish to proceed to the next field, status should be +1. If a new value for the current field is to be given, the status should be −1.

Note that a dummy routine called UCONT is defined in NSL and always returns status +1; therefore a user defined UCONT should be loaded before NSL.

Status is also used to indicate whether UCONT is being called before the field has been read (−1) or after the field has been read (+1). *(See Appendix F for a more detailed explanation).*

2.4.5.11 Combined Control

This feature enables you to define the checking of field values based upon combinations of all the other individual control functions. Combinations are formed by establishing logical "AND"; "OR"; "AND/OR" relationships between the individual functions and groups of functions.

To accomplish this, the control functions are assigned to four groups:

1. Default Value
2. Legal - Illegal Value
   Legal - Illegal Range
   System Defined Control
3. Add Fields
4. User Control Algorithm

Obviously the use of the Combined Control feature does not mean that you must use all groups; however, you must follow the prescribed sequence when you do use it. In other words, Group Four — User Control Algorithm must be defined *after* Group Three — Add Fields. The groups themselves can only be linked with "AND". Within a group, the functions can be combined in any order. Within Group Two, it is recommended that functions be combined with either "AND" or "OR", not both.

Perhaps the best way to demonstrate the use of the Combined Function is by example. Suppose you wish to define a two digit field with the following control criteria:

1. Values between 10 and 20 not accepted
2. Three digit numbers not accepted
3. Negative values not accepted

Symbolically translated, input value X must be such that

$$0 \leqslant X \leqslant 10$$
$$20 \leqslant X < 100$$

(Legal range 0—10) .OR. (Legal range 20—99)

The Combined Control function does impose a few limitations:

* a maximum of 40 funtions may be combined
* for User Control Algorithm
  Equivalent fields
  Default fields

  only one control of each type may be defined.
* no Accumulation Fields may be defined
* User Control Function must always be defined last.

3 NORD SCREEN LIBRARY SYSTEM (NSL)

3.1 *GENERAL*

NSL can be used with a background program by one person or alternatively in a real-time system which uses many terminals. In both instances buffers together with their initial data must be defined in labelled common areas. In real-time applications the location of the buffers must be fixed.

3.2 *PICTURE HANDLING*

An application program can have access to two picture buffers. They are called the PRIVATE buffer and the PUBLIC buffer and are to be found in the labelled common blocks defined as follows:

COMMON/PRIVATE/ITERM (128), IPRIV (1920)
COMMON/PUBLIC/NROOTSEG, IPUBL (2047)

The lengths 1920 and 2047 are used solely for the example. When running a multi-terminal application under the system it will normally be wise to choose the lengths of IPRIV and IPUBL so that the labelled common blocks are exact multiples of 1024. Even if IPRIV is not used, it must be defined to at least 30 words. Similarly, IPUBL must be defined to at least 5 words. For COBOL applications a FORTRAN block statement must be defined.

3.3 *THE LOCATION OF THE COMMON AREAS*

Common areas must start at the same location for all programs or segments. The PRIVATE area can vary in size from a minimum of a 158 words (no private pictures) up to whatever is considered applicable. The PUBLIC area will normally be defined as the same size for all applications, since it will be on either a shared segment or a reentrant segment.

3.4 *THE INITIAL CONTENTS OF THE COMMON AREAS*

The first 128 words of the "PRIVATE" area are called the "terminal buffer" and must be initiated correctly. The contents are as follows:

Word:	*Contents:*
1	SINTRAN device number for this terminal
2	Terminal type and picture displacement
3	COBOL FLAG/escape character
4	Read strategy
5	PAD character/program mode
6	Cursor position
7	Error code
8	Break and echo strategy
9	Option word
10	Length of private picture area

The remaining 118 words are used as a scratch area by NSL.

The length of the private picture area (word 10) is the total length of IPRIV. This should be set to zero if no PRIVATE pictures are to be used. Even then IPRIV itself must be at least 30 words long.

3.4.1 *Terminal Type*

The first 3 bits of the word are used for Terminal Type. At present it can be:

0 TDV 2000 (Tandberg)

1 VISTA

2 Reserved for future use

3 TDV 2100 (Tandberg)

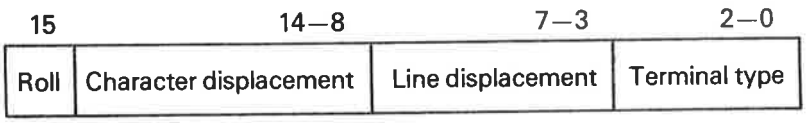
4 INFOTON 200

5 INFOTON 400

6 STANSAAB

7 Not used

Bits 3-7 indicate the picture displacement in lines (0-30). Bits 8-14 indicate the picture displacement in character positions from the left (0-78). Normally both of these will be zero. Lastly, the most significant bit is used to indicate whether the terminal is in ROLL mode (=1) or PAGE mode (=0). Thus, the second word in the terminal buffer looks like this.



For each terminal type NSL maintains a number of tables. New terminal types can be implemented; if this is desirable, contact your local Norsk Data office.

3.4.2 *COBOL FLAG/Escape Character*

Escape character defines the character which, when typed, always causes the execution of an immediate return from a "read fields" call back to the main application program. It must be right justified. The most significant bit indicates that the application program is written in COBOL: (1) COBOL (0) not COBOL.

3.4.3 *Read Strategy*

Read strategy defines how individual field reading is terminated and how, from a "read-fields" call, return to the main program is initiated.

Value:	*Meaning:*
0	Fields must be individually terminated. Return on typing control S-Y.
1	Same as 0, but with automatic return after reading, or by-passing the last field to be read, or after leaving the first field to be read using ↑ ←.
2	Fields automatically terminated on typing the last character; return the same as in 0.
3	Same as 2 but return the same as in 1.
4-7	Same as 0-3, but re-edited fields are written out with blink.
8-11	Same as 0-3, but re-edited fields are written out with underline.
12-15	Same as 0-3, but re-edited fields are written out with low intensity.
16-19	Same as 0-3, but re-edited fields are written out with inverse video.
20-23	Same as 0-3, but re-edited fields are written out "invisible".

The last two groups refer to TDV 2100 terminals.

3.4.4 *PAD Character/Program Mode*

The right byte of this word contains Program Mode character and indicates whether the application is using a PRIVATE picture (0) or a PUBLIC picture (1).

The left byte of the word contains the PAD character. The PAD character is used to fill the eventual unused byte in a data element of storage type Byte. This can be set by the user.

3.4.5 *Cursor Position*

Cursor Position keeps track of where the cursor is at any time and is initiated and updated automatically by the system.

3.4.6 *Error Code*

Error code is set to zero by the system when you call a subroutine; if an error is detected, an appropriate error code may be set.

3.4.7 *Break and Echo Strategy*

Break and echo strategy decides how and when characters typed in on a RFLDS call are to be handled and echoed.

There are two strategies from which to choose:

0 break on every character, and immediate echo

or

1 the special break and echo strategy designed for NSL, which uses a feature in the SINTRAN III teletype driver routine. Basically, when break occurs, this allows the driver to echo several characters which are legal for the field type (up to the maximum allowed for the field) or when a control or incorrect character is typed.
More detailed information can be found in Appendix I

Bit 3,4,15 handles various features in connection with terminal output.

Bit 3 = 0 normal

= 1 The monitor call using 8 bytes output (8OUT) is used instead of 1 byte output (OUTBT). *(See SINTRAN III ref. manual.)* If your version of SINTRAN has this feature, it will give better performance than OUTBT *(See Appendix I).*

Bit 4 = 0 normal

= 1 The monitor call OUTSTRING is used instead of OUTBT.

Bit 15 = 0 normal

= 1 The write-picture-to-display routine will only print out heading text. (Dots to indicate fields are not printed). This may save a lot of time on output. If used, the screen should be cleared before the WRPTD call.

3.4.8 *Option Word*

Some bits in this word are used to set NSL in different modes and can be used from an application program:

Bit 0 = 0 normal

= 1 ignore control

It is possible to force in illegal values, if desired. When this bit is set to 1, the control will carry on as normal, with error message and replacing of cursor to the begining of the field, but when typing → (cursor right) the illegal value will be stored in the record in the ordinary way.

3.5 *THE PRIVATE PICTURE AREA*

If PRIVATE pictures are to be used, the array IPRIV should be initiated as follows:

Word:	*Contents:*
1	0 - zero
2	length of IPRIV - must equal ITERM (10)
3	maximum number of picture descriptions which are to be read to IPRIV
4	0 - zero (used for number of pictures now in IPRIV)
5	0 - zero (used for number of words still available for new pictures).

3.6 *PUBLIC PICTURES*

The PUBLIC picture array can be initiated and used in the same way as the PRIVATE picture area. However, it is designed to be on reentrant, or read-only segment, and the IPUBL array will normally be pre-loaded.

It is a good idea to load it on a reentrant shared segment which also contains the system subroutines, other reentrant programs in common use, and the FORTRAN reentrant run-time library routines. In this case the PUBLIC area cannot be changed. The variable NROOTSEG must be set to zero, and IPUBL (1) must be set to −1. Instructions to load the PUBLIC area are given in section 3.6.2.

3.6.1 *Reading Pictures*

The Public area can of course be used in background applications; there are no restrictions on the reading of pictures to the PUBLIC area. However, they can not be removed. When the PUBLIC area is being shared, and IPUBL (1) equals zero, pictures can be read to the area. However, it is then necessary to use a semaphore to prevent two processes from reading in pictures at the same time. The SINTRAN device number for this semaphore should be placed in the left-hand byte of NROOTSEG. If this byte is zero, it will be assumed that the use of a semaphore is not desired.

3.6.2 *Loading the Public Area*

If you wish to have pictures pre-loaded to a read only segment in a real time application, you could:

1. Decide

   a) the size of the public picture buffer

   b) which pictures are to be in it

   c) the physical address at which it is to start (normally a page boundary)

2. Write a program to read in these pictures

3. Load the program in background, and set the upper limit *(See the Nord Relocatable Loader manual)* so that the Public common area starts at the desired address.

4. Run the program

   set IPUBL (1) = −1

5. Return to the loader and take a binary dump of the public picture buffer (setting IPUBL (1) to −1 flags the buffer as read-only).

6. The Real Time Loader can now be used to load a segment from this binary dump.

Note that in the background mode it is possible to achieve many of the advantages normally associated with real time applications, such as: reentrancy, shared code, shared pictures. Applications should be reentrant, and the real time libraries for both NSL and the NORD FORTRAN system should be used. The application is then run so that all pictures are read into the public or private buffer. After this, return to the loader; take a binary dump of the whole application, including picture data areas. Use this dump to define a reentrant subsystem *(refer also to the manual "SINTRAN III User's Guide", Section 6.9.2 and Appendix A).*

3.7 *SUMMING UP*

The data area strategy described in Section 3 should enable a designer to choose an arrangement suitable for his application. For example:

Many terminals - few pictures: use the PUBLIC area

Many terminals - many pictures - a few often used: Put pictures most commonly used in PUBLIC area; let others be read to PRIVATE area when needed.

For each terminal it is possible to choose any or no escape character, as well as a read strategy. In fact you could allow the terminal operator to choose what is best. *(See the SET-PARAMETERS command in the NORD DATA ENTRY Editor).*

4 PROGRAMMING WITH NSHS

4.1 *GENERAL*

To program with NSHS you must use specific call statements and parameters in order to activate a set of FORTRAN subroutines. In this chapter you will find a description of both the parameters and the subroutines themselves. You will notice that some of the subroutines have different entry points and parameters when they are called from a COBOL program. Here is a list giving a brief functional description of the all the calls:

GTPIC: get pictures; reads a number of pictures into the picture buffer and translates a number of picture names to picture numbers which can later be used to refer to pictures.

GTPICC: COBOL entry point

RMPIC: remove picture(s); removes one or more picture descriptions from the PRIVATE picture area.

WRPTD: write picture to display; writes a picture to a visual display unit with all I/O fields blanked out.

GTFDN: get field numbers; translates a list of field number indicators to field numbers which are then used to refer to the fields.

WFLDS: write fields; writes fields to visual display unit.

WRPTF: write picture to file; writes a picture with leading texts and fields to a file.

RFLDS: read fields; reads a number of fields.

CFLDS: clear fields; clears a number of fields on the visual display unit writing full stops for each possible character position.

CLBUF: clear buffer; all non-locked character fields are filled with spaces, others with zero (binary).

ZREAD: set read; sets bit 15 of the given field numbers to 1.

RREAD: remove read; sets bit 15 of the given field numbers to 0.

ZLOCK: set lock; sets bit 14 of the given field numbers to 1.

RLOCK: remove lock; sets bit 14 of the given field numbers to 0.

ZMUST: set must read; sets bit 13 of the given field numbers to 2.

RMUST: remove must read; sets bit 13 of the given field to 0.

ZVERI: set verify on all fields described; picture sets the lock bit on all fields which are not to be verified.

RVERI: reset verify; resets the verify modus.

ZMALL: set must all; set must on all fields described.

RMALL: reset must all; resets the must all modus.

CLSCR: clear screen; clears whole or part of the video screen.

WMSGE: write message; writes a given text to the last line of the screen.

WMSGEC: COBOL entry point

CLMSG: clear message on screen.

ZCURS: sets the cursor to the start of the wanted field.

ZBELL: ring bell on terminal.

The following sections explain in detail both the parameters and the subroutines.

4.2 *THE PARAMETERS*

4.2.1 *Picture-File-Name*

This is the name of a picture file; it should be given in the normal way with name and type. The *parameter must be specified as character string* when input from FORTRAN. From COBOL it must be an alpha-numeric variable in DISPLAY format defined on the 01 or 77 levels.

4.2.2 *Number-of-Pictures*

This is an integer variable with values between 1 and 49 inclusive.

4.2.3 *Picture-Name-String*

This is a parameter string specifying the names of the pictures to be used. Each picture name must fill 8 characters; names containing less than 8 letters must be padded out with spaces. The parameter must be specified as a character string when input from FORTRAN. From COBOL it must be an alpha-numeric variable in DISPLAY format defined on the 01 or 77 levels.

4.2.4 *Picture-Number-Array*

This is an integer array containing the picture numbers provided by the system. Each picture number identifies a particular picture.

4.2.5 *Flag*

This is an integer variable which can have two values:

0: write display mode characters

1: do not write display mode characters

4.2.6 *Field Indicator Array*

This is an array that defines the fields within a picture. The fields in the array are only those that will be used.

Field indicators can be:

Line number * 256	all the fields on a line
Line number * 256 + position on line	a single field on the specified line
1–255	absolute field number
0	all the fields in a picture

The field indicators in an array must be given such that the fields referred to are in ascending order of field number.

4.2.7 *Field Numbers*

These are supplied by the system as a translation of a given field indicator array for a given picture. They are used as identifiers in connection with the writing or reading of fields. Field numbers contain information as to whether or not the field is locked; whether or not it has been read into after a read command; and whether or not input to a field is mandatory. A field number fills one computer word.

4.2.8 *Field Number Array*

This is an array of field numbers generated from a Field Indicator Array.

4.2.9 *Number of Fields*

This is an integer variable indicating the number of fields in the Field Number Array.

4.2.10 *Record*

A record is an array of computer words containing the data element values associated with a given field number array. When writing to the screen, the record will contain the internal representation of the fields to be output; when reading from the screen the record will receive the values read in.

4.2.11 *Data Element Index Array*

This is an array containing the indexes to the first word of each data element in the record array. The indexes must be in a one to one correspondence with the field numbers in the Field Number Array.

If the first index in a data element index array is zero, the system will automatically generate a data element index array assuming that the data items are to be placed in the record in an order corresponding to the field number array. Indexes must be positive integers, greater then zero and less than or equal to the value in INDMX which is set to 2048. It can be patched if so desired.

4.2.12 *Code*

This is an integer variable

4.2.13 *Terminating Character*

The terminating character indicates how the read call was terminated. It can have the following values:

−1:	user defined escape character
0:	carriage-return (terminating last field)
1:	user defined field terminating character (terminating last field)
7-14:	control L, or control S-Y typed
>99	terminated by a user control function

For read strategy 1 and equivalent:

2 : ← }
5 : ↑ } for left bypassing of first field

3 : → }
4 : ↓ } for right bypassing of last field

Note that cursor control characters and control L or S-Y can only terminate reading when the cursor is "outside" a field. Also, if one of these control characters "collides" with a cursor control character, (control Y and Z on VISTA, and control X on TDV 2100), their function as a terminating character is invalidated.

Depending on the current break strategy, control V may be a special character vis-a-vis SINTRAN. This means in practice that the following can be used as terminating characters for a RFLDS call.

TDV 2000:	$L^C, S^C, T^C, U^C, W^C, X^C, Y^C$.
VISTA:	L^C, S^C, T^C, U^C, W^C.
TDV 2100:	$L^C, S^C, T^C, U^C, W^C, Y^C$.

4.2.14 *First-Line, Last Line*

These are integer variables giving the line numbers for the first and last line to be cleared in a clear screen call.

4.2.15 *Start/End-Position*

This is an integer variable giving the first and last position in a line to be cleared in a clear screen call.

4.2.16 *Text*

This is a character variable containing up to 79 characters. It must be a character string when calling from FORTRAN, or an alpha-numeric variable in DISPLAY format defined on the 01 or 77 levels when calling from COBOL.

4.2.17 *Status*

This is mainly a return parameter, an integer variable provided in all calls. It can have the following values:

2: an error has occurred

1: illegal picture number

0: ok

−n: field number/indicator, picture name/number n in error

For 2, the error code indicator word ITERM (7) will always be set with a value indicating the error. For 1 or −n it may also be set. The various error codes are given in Appendix C. This status variable is also used as input to the RFLDS *(See Section 4.3.6).*

4.2.18 *File-Number*

This is a FORTRAN compatible file-number. COBOL programs can obtain such a file number by calling OPENFC. Files opened in this way can be closed by calling CLOSEC. Thess subroutines are described in Appendix H.

4.3 *THE SUBROUTINE CALLS*

Each of the subroutine calls will be described in detail. Parameters in italics specify where the result of the operation is to be stored.

4.3.1 *GTPIC (Get Picture) - Entry Point from COBOL — GTPICC*

CALL GTPIC (picture-file-name, number-of-pictures, picture-name-string, *picture-number-array, status*)

The purpose of this subroutine is to read pictures into the current ITERM (5) picture buffer and to provide picture numbers which can then be used to refer to the pictures in other subroutine calls. Nothing is written on the display.

4.3.2 *RMPIC (Remove-Picture)*

CALL RMPIC (number-of-pictures, picture-number-array, *status*)

The purpose of this subroutine call is to remove pictures from the PRIVATE picture buffer.

If the number of pictures is zero, the PRIVATE buffer will be cleared.

4.3.3 *WRPTD (Write Picture To Display)*

CALL WRPTD (picture-number, *status*)

The purpose of this subroutine is to write a picture on the display showing all leading texts but with all fields "blanked" out and full stops for each character position.

Only one picture per call can be written out, but it is possible to have any number of pictures visible on the screen at one time. However, only pictures which do not overlap one another are displayed simultaneously.

When a line of a picture is written out by WRPTD, leading spaces are replaced by cursor rights, trailing spaces (i.e. spaces on a line after the first field or visible character and to the carriage return) are always printed.

4.3.4 *GTFDN (Get Field Numbers)*

CALL GTFDN (picture-number, number-of-field-indicators, field-indicator-array, *field-number-array, number-of-fields, status*)

The purpose of this subroutine is to translate a field indicator array into a field number array which is then used to refer to fields in other subroutine calls.

Field indicators within the field indicator array may be:

0;	all fields in a picture; number-of-field indicators must then be 1.
line number * 256	all fields on a line
line number * 256 + position (i.e. first, second) on line:	an individual field on the specified line
1 - 255:	absolute field number i.e. an individual field

GTFDN checks that the field indicators given are acceptable, and then generates the corresponding field number array.

4.3.5 *WFLDS (Write Fields to VDU)*

CALL WFLDS (code, picture-number, number-of-fields, field-number-array, record, data-element-index-array, *status*)

The purpose of this subroutine call is to write a specified number of fields to the VDU. *Code* is an input parameter which selects a display mode for all the fields to be written. It can have the values:

0	normal
1	blink
2	underline
3	low intensity
4	inverse video
5	invisible
6-11	same as 1-5 but with bell (pip)

Note that these codes will have no effect if the display modes do not exist on the terminal type being used.

Only unlocked fields will be written. Single or double integer Storage Code data elements which:

1. are defined with sign suppress and have negative values,

   or

2. have been given values containing more decimal digits than the field allows

will be written out with stars instead of digits.

Byte Storage Code data elements that contain bytes with values less than 040 octal will be replaced by stars; inconsistent data elements for both Storage Codes 4 and 5 will also be represented by stars.

4.3.6 *RFLDS (Read Fields)*

CALL RFLDS (code, picture-number, number-of-fields, *field-number-array, record, data-element-index-array, number-of-fields-read, terminating-character, status)*

The main purpose of this subroutine call is to read fields from the keyboard to the VDU screen, and to convert the characters read to their respective data element value. The data element value is then placed in the record. RFLDS has four functions, and each function is indicated by the value of the input parameter *"code"*.

code:	*meaning:*
0	normal read
1	control read, or verify
2	check record
3	read password(s)

Each of these functions will be described in detail, but first some information is required about RFLDS in general.

RFLDS is the "heart" of the NSL library. In addition to managing the input to an individual field, it:

* determines whether a value read in is acceptable according to the defined control functions
* executes accumulation and equivalent control functions, writing the new accumulated or equivalent values to the appropriate field
* manages all tabbing functions
* provides comprehensive editing funtions
* ensures that fields having the must-read bit set cannot be bypassed without receiving a value
* ensures that locked fields are always bypassed

When a field is to be read, RFLDS places the cursor at the first position of the field on the VDU screen. You can then type in characters. If they are legal characters for the field type, they will appear on the screen, if not, an audible signal will be given. Such is the case when alpha characters are typed in numeric fields. A signal will also be given if you attempt to type more characters in the field than what there is space for. *(See Section 3.4.3 on Read-Strategy).* An exception in this case occurs when reading a password (code = 3), if:

(a) no characters appear on the screen

(b) no indication of illegal characters is given.

Input to a field is terminated either automatically when the last character is accepted (depending on the read strategy) or explicitly by typing a terminating character (carriage return or user defined). When the field has been terminated, the characters read in may be rewritten to the field in their edited form and/or with a display mode other than normal, depending of course on your particular type of terminal.

Normal Read (Code = 0)

Copy Field

It is often useful to be able to correct a record, or parts of a record, which has been read in previously. The record is written to the VDU screen using WRPTD and WFLDS or simply WRPTF. Fields which you do not wish to change are then locked. Fields which you may wish to change/edit are set read, i.e. the read bit in the appropriate field numbers is set to 1, prior to calling RFLDS. This tells RFLDS that the "old value" characters are to be made available for editing/copying when the field is read. Characters may also be copied from "identical" fields immediately above the field being read *(See Section 5.2).*

When reading in a new record, this feature also applies to all fields already read. That is, if you tab back to a field that you have already typed in, its "old" characters can be copied/edited.

Setting the Read-bit (to 1)

This occurs when a field is given a value. Read bits set on entry to RFLDS (to indicate old values) will be set to zero on exit from RFLDS if no value has been read into the field.

Empty Data-Elements/Fields

For Byte Storage Code numeric fields, an empty data element (containing all spaces) is regarded as different from a data element containing just one zero + spaces. (Or spaces + one zero).

It is possible to force an "empty value" into a field/data-element during the execution of RFLDS by typing Q^C followed by J^C. The corresponding data-element will receive the empty value via the clear-buffer function, and will then be written out. For Non-byte storage codes the result will be the same as typing in a zero. For Byte storage codes the data element is filled with spaces.

Note that typing Q^C J^C is regarded as giving the field a value, and in a normal RFLDS call it will be set-read. In a control read it will be checked against the old data-element value, and if different but to be accepted, it will also be set read.

If the start field indicated is in error, start field will be set automatically to the first field number.

Control

If control functions are defined for a field, the control will normally be carried out after termination of the reading of the field *(See Section 2.4.5).*

Multiple Zeroes

You are not permitted to type leading zeroes into a numeric field. To indicate the value zero, a single zero may be typed. It is illegal to type a zero in a numeric field immediately after a minus. The exception to this is after a decimal point/comma has been given to a decimal field. If you must type leading zeroes, use edit code K when defining the field.

Trailing/Leading Zeroes, Spaces

When the character string for a data-element is generated for control-read or to make the old characters available for edit purposes, leading/trailing fill characters are always stripped from right/left justified fields. Leading zeroes are always stripped from numeric fields, trailing zeroes are always stripped from left justified numeric fields. For example:

10000 in left justified numeric is just one character, namely 1.

11100 in left justified numeric is three characters, 111.

10000 in right justified numeric is 5 characters.

This has been done to reduce to a minimum the amount of typing required.

Field Handling

Carriage return or a user defined terminating character as the first character typed to a numeric field causes a zero value to be placed in the corresponding data element. For fields with edit codes 0 – 9, typing carriage return or the terminating character causes the decimal point/comma to be inserted after the last typed digit (if it has not been bypassed). All digits in the field are right justified and field reading is terminated.

The edit codes 0 – 9 specify from 0 – 9 digit positions after the decimal point/ comma, and here (,), (.), the terminating character, and space have special meanings for these edit codes. The (,) and the (.), if typed, will cause the decimal point/ comma to be inserted visibly on the VDU screen. If there are not sufficient digit positions remaining, i.e., the decimal point/comma position has been passed, a "pip" will be given. Note that the use of comma and full stop is different in England and Norway. You can choose what you want by altering the contents of a special word *(See Appendix A).*

Number-of-Fields-Read (Parameter)

This will be equal to the number of fields to which characters have been typed and a new value accepted. Note that typing the same value again is equivalent to a new value!

Control Read (Code = 1)

Using control read, the value read in is compared with the data-element value already in the record. The comparison occurs both as characters are typed and at the termination of each individual field reading. It is possible to have a *non-match* at the character level and a match at the data-element level.

A character not matching the old value's character string causes an audible signal and the cursor will not move. The next character will be accepted independently of matching, provided that it is legal for the field. If it does match, then the matching process described above is executed for each new character. If the character does not match, a signal will be given for each non-matching character subsequently typed but the characters *will be accepted* without further protest.

At field termination, if the values are different, a "pip" will be given and the cursor will be repositioned at the beginning of the field. The old value will be displayed on the last line of the VDU screen. The terminal operator can type "forward cursor" (→) in which case the new value will be accepted. Otherwise, the field can be typed in again.

Editing

Only the editing characters control A, and or ← are legal. Control Q resets everything the same as when you first began typing into the field.

Number of Fields Read

For a RFLDS call with code 1, a control read, the *number-of-fields-read* will, on return, be equal to the number of fields which have received a new value. On entry all fields not locked will automatically receive "old value", and Must-Read is effectively set for all fields.

Control Read (Verify) and Normal Read

We have attempted to make the record verifying operations at the keyboard identical to those used when you orignally entered the record via a normal RFLDS call. For example (it is assumed that the fields were originally read into a record where clear-buffer had been executed prior to typing in data)

Original Read	*Control Read*
bypass using →	bypass using → illegal if value in old field; → has same function as before if different value given.
$Q^C J^C$	$Q^C J^C$ giving empty field; accepted if old field was empty. In which case → would have the same function. Otherwise, the field will be blanked out and the old value will be written on the last line.
CR	always accepted as zero value if old value was 0. (i.e. accepted as if 0 + CR had been typed).
LF	bypass to first field on next line

If after terminating a field the old and new values are found to be different and the old value was an empty Byte field, then the message:

"field was empty"

will be written on the last line. To type in an empty field one must type Q^C J^C !!

Termination of Control Read, Verify

The verify function is rather strict. You cannot wander about from field to field; you are forced to follow a predetermined order from top to bottom and left to right in the picture. Previously, the only way to avoid this was either to verify all fields, or to type your NSL escape character. When using the latter possibility with the NORD Data Entry system, there is the disadvantage that the whole call is regarded as aborted, and any changes to the record are ignored.

It is now possible to terminate a control read by typing control Q twice followed by control S. However, you then have no way of knowing which fields, after the last one corrected, have been verified.

Check Record (Code = 3)

The execution of this function has no bearing on the VDU screen itself. This feature enables you to use the control functions which have been defined for a record, on records which have the same format, but have been originated independently of input via a normal RFLDS call.

If code = 2, check record will be executed when read fields is called.

Upon return, all non-locked fields found to be correct, will have the read bit set to 1, i.e. they will be negative.

All unlocked fields found to be incorrect will have the Must-Read bit set to 1.

If control functions are not defined for an unlocked field, then no status bit will be set. *Number-of-fields read* will be the number of data-items controlled and found to be correct.

If any items are found to be incorrect, the *status* upon return from read fields will be 2, and ITERM (7) will be 40.

Read Password (s) (Code = 3)

If code = 3, then RFLDS will be executed normally except that none of the characters typed in will be echoed.

The cursor will move from field to field, but no "pip" will be given for incorrect characters. If the field has had any control functions defined, they will be executed. All control of passwords in an application program should take place externally to read fields.

4.3.7 *WRPTF (Write-Picture-To-File)*

CALL WRPTF (file-number, flag, code, picture-number, number-of-fields, field-number-array, record, data-element-index-array, *status*)

The purpose of this subroutine is to write a picture with leading texts and fields (except locked fields) to a file. The file can be the display, so that WRPTF can be used as WRPTD with field values.

The parameter *flag* is used to distinguish between normal sequential media and the VDU screen, and can have the values:

1: do not write display mode characters, use space and line feed.

0: write display mode characters.

The parameter *code* can have values 0 - 5 corresponding to the first 6 values of the same parameter in WFLDS. Note that the file number must be FORTRAN compatible.

4.3.8 *CFLDS (Clear Fields)*

CALL CFLDS (picture-number, number-of-fields, field-number-array, *status*)

The purpose of this subroutine call is to clear a given number of fields in a picture (i.e., put (.)'s in all character positions). Locked fields are not cleared.

The Data record is not affected.

4.3.9 *CLBUF (Clear Buffer)*

CALL CLBUF (picture-number, number-of-fields, field-number-array, record, data-element-index-array, *status*)

The purpose of this subroutine call is to zero out data elements in the record. For Storage Code 3, the data element is filled with spaces; for other storage codes the data element is filled with binary zeroes.

4.3.10 *ZREAD/RREAD, ZLOCK/RLOCK, ZMUST/RMUST*

CALL ZXXXX or RXXXX (picture-number, number-of-fields, start-index, field-number-array, *status*)

The purpose of these subroutine calls is to set (to 1) or remove (set to 0) the three status bits in a field number. On one call, one or a number of *successive* field numbers can get a particular bit set or reset.

On entry to a RFLDS call, the READ bit indicates whether or not there is a value for the field in the record (1 or 0), and on return from a RFLDS call it indicates whether or not a value has been read to the record (1 or 0). Note that even if the READ bit is 1 on entry to RFLDS it will be zero on return if no new value has been read in.

The LOCK bit is used to lock a field. It can then neither be written to or read into, and in fact is treated as a leading text. Note that locked fields will not appear if WRPTF is used.

The MUST bit is used to set a field so that once entered, the field must be given a value. Note that carriage return only (giving zero value) is not accepted. A space or a zero must be typed explicitly. (This bit has meaning only for RFLDS).

Start-index indicates which field number in the field number array one is to start at.

4.3.11 *ZVERI/RVERI, ZMALL/RMALL*

(set/remove XXXX)

CALL ZXXXX or RXXXX (picture-number, number-of-fields, field-number-array, status)

These calls act on the relevant bits as described in section 4.3.10. However, these commands work only for those fields defined with MUST or VERIFY during picture definition *(See Sections 2.2.5 - 2.2.8).*

4.3.12 *CLSCR (Clear Screen)*

CALL CLSCR (code, first-line, last-line, start-or-end-position, *status*)

The purpose of this subroutine call is to clear all or part of the VDU screen; that is, lines from position 1 to start-or-end-position, or all characters from start-or-end position to 79.

Possible values for code are:

0: clear whole screen
1: clear whole lines
2: clear from and including start-position to end of lines
3: clear from and including position 1 to and including end-position

4.3.13 *WMSGE (Write Message) - Entry Point WMSGEC from COBOL*

CALL WMSGE (Text)

The purpose of this subroutine is to write a message on the last line of the VDU screen.

Text must be a variable containing up to 79 characters terminated with an apostophe or a *. All non-printable characters will be replaced by *. Text must be a character string when calling from FORTRAN, and an alpha-numeric variable in DISPLAY format defined at the 01 or 77 levels when from calling COBOL.

4.3.14 *CLMSG (Clear Message)*

Clear message on screen.

4.3.15 *ZCURS (Set Cursor)*

CALL ZCURS (<Picture Number>, <Field Indicator>, <Status>).

Sets the cursor to the start of the wanted field.

4.3.16 *ZBELL (Ring Bell)*

CALL ZBELL

Gives a "pip" on the terminal.

5 USING SCREEN PICTURES

The previous chapters have described the definition and programming aspects of screen pictures. In this section, the use of screen pictures is described from the terminal operator's point of view.

5.1 *EDITING*

The editing of fields follow the same strategy as that used in picture definition, except that editing is oriented towards fields, not lines, and using control P or R and ↑, only indentical fields lying directly above can be copied. Only significant (not editing) characters are copied.

The following control characters can be used for editing:

CONTROL A	Delete previous character
CONTROL BX	Copy identical field above characters up to and including X
CONTROL C	Copy one character from old value
CONTROL D	Copy old value to end of field without terminating the field
CONTROL E	Insert characters until the next control E is typed
CONTROL F	At the beginning of a field, write out version number
CONTROL G	Multipunch. Hexadecimal code for character given on last line
CONTROL NX	Copy field above up to X
CONTROL OX	Copy old characters in this field up to but not including X
CONTROL P	Copy one character from identical field above
CONTROL Q	Delete all characters in this field
CONTROL R	Copy rest of field above without terminating the field
CONTROL ZX	Copy old characters in this field up to and including X
CONTROL KX	Same as control ZX (for Vista and Infoton 200 only).
CONTROL Q CONTROL J }	Give field a zero value
LINE FEED	Go to first field on next line that has fields.

In addition, when in a field, the cursor control characters can be used as follows:

Curser Right	Equivalent to Control C
Curser Down	Equivalent to Control D but terminates field
Curser Left	Equivalent to Control A
Curser Up	Equivalent to Control R but terminates field

The insert function is initiated and terminated via control E. Unlike QED, the insert mode is not visibly indicated. Every character is echoed with a "pip" in insert mode. Even after the insert mode has been left, and control A or ← is used to remove characters, a pip will be given each time an inserted character is removed.

The multipunch function enables the user to insert any character value (0 -377B) in the data element for a string field. The value for the character is represented on the screen as a star. Multipunch characters with values > 177B or which represent characters illegal for the field type cannot be copied via control C, D, etc. They must be entered again.

5.2 *AUTOMATIC PICTURE RECOVERY WITHIN READ-FIELDS*

During the execution of a read-fields call, the picture on the VDU screen can disappear for various reasons. For example:

*noise, static electricity

*terminal power failure

*switch off/accident etc.

A system for automatic recovery has been implemented. It clears the screen, writes out the picture again with all filled out and set read fields displayed, and positions the curser at the start of the current field.It is triggered by typing control Q three times in succession when in "input to field" modus.

APPENDIX A

EDIT CODES AND FORMATS

When defining a field, the user may specify an editing option. NSHS contains a table of 29 standard formats, stored in the labelled common area with the name FORMAT. A format consists of a series of words in the area FORMAT. When these words are read in ascending order (i.e., IFORMAT (N), IFORMAT (N + 1), IFORMAT (N + 2) ...) they describe the format of the characters to be printed from *RIGHT → LEFT*. The contents of these words are interpreted as follows:

Value:	*Meaning:*
1	Print the sign (if any), space or —
2	Print next character
3	Print thousands separator for decimal fields
4	Print decimal position for decimal fields
>31	Print the ASCII character represented by value. That is, a format character.
<0	Loop in the format

An example will illustrate how it works. Consider edit code 2. *(See Section 2.4.4.2)*. This edit code will print a negative 10 significant digit number as follows:

12.345.678,90—

and the format string for this is:

1,2,2,4,2,2,2,3,—4,

3 and 4 are the codes for the thousands separator character and the decimal position character, respectively.

The two characters to be used for formatting decimal fields are stored in the 41st word of a labelled common area called PICDEF. The word normally contains the ASCII code for (, .). In England this should be patched to (. ,).

Suppose, for example, you wished to print a character string read into a byte field as follows:

C*CCCC*CC/ /CCCC*

Here, C represents a character read in and the * and/represent format characters. The format string for this would be:

45,2,2,2,2,47,47,2,2,45,2,2,—5

The correct position in the format array is determined by the edit code, and a pointer array called FPOINT. This pointer array contains one word for each edit code. Each word contains:

Left byte: 0 or terminating character for this edit code

Right byte: index to start of format string in FORMAT

You may change the contents of both FPOINT and FORMAT, but be aware of the following restrictions.

1. Edit code 0 - 9 must have at least one format character.

2. Format characters cannot be defined to appear left-most in a field (include them in the leading text!).

3. The contents of FPOINT and FORMAT MUST BE IDENTICAL when defining pictures and using the same pictures.

4. FORMAT should not be greater than 256 words.

Both FPOINT and FORMAT for the standard NSHS system are shown below.

```
*)9BEG;)9ENT FPOIN

% FPOINT is the format pointers array with one word for each edit code.
% The contents of each word are as follows:

% Left byte:    terminating character for fields with this edit code (no parity bit
                set), or zero

% Right byte:   index to start of format string in FORMAT array

SUBR FPOINT
@ICR
FPOINT:; INTEGER ARRAY PPPP: = (
   0 \ 20,0 \ 22,0 \ 24,0 \ 26,0 \ 30,0 \ 32,0 \ 34,0 \ 35,
   0 \ 40,0 \ 74,0 \ 42,0 \ 45,0 \ 60,0\ 42,0 \ 42,0 \ 42,
   0 \ 42,0\ 42,0 \ 42,0\ 42,0\ 65,0 \ 65,0 \ 65,0 \ 67,
   0 \ 67,0 \ 67,0 \ 65,0 \ 65,0 \ 65) ; @CR
RBUS
*)9END
   COMMON /FORMAT/ IFORMAT(80)
   DATA IFORMAT/
- 2,2,2,2,2,2,2,2,2,4,2,2,2,3,-4,
- 1,-8,1,-11,1,-14,1,-17,1,-20,1,-23,1,-26,1,-29,1,-32,
- 1,2,-1,2,2,2,2,2,56B,2,2,56B,2,-1,1,2,2,56B,-3,
- 2,-1,2,2,2,57B,-4,1,-51,19*0/
```

Symbolic versions of the above are supplied as part of the NSH system.

APPENDIX B

CONTROL CHARACTER HANDLING

NSHS is designed for asynchronus ASCII oriented video terminals with cursor control. Differences between terminal types occur at the control character level, i.e. characters with octal values between 1 and 37 inclusive.

In NSHS a "standard" control character set similar to the TDV 2000 control character set is used. For each terminal type, control characters must be translated to the NSHS set on input, and from the NSHS set on output.

The internal control character set is defined below: (I) indicates that the character value going into NSHS defines the function; (O) indicates the character value sent out by NSHS initiates the function.

1	Control A	Delete previous character (I)
2	Control B (X)	Defines blink display mode (NSD). Copies identical field above up to and including X (I) (NSL)
3	Control C	Copy one character from old line/field (I)
4	Control D	Copy all characters from old line/field without terminating (I)
5	Control E	Define field (NSD), insert characters in field (NSL) (I)
6	Control F	Abort picture creation/editing (I) (NSD).
7	Control G	Remove one line (NSD) (I), bell, i.e., audio signal (O)
10	Control H	Cursor left (I/O), ← or delete previous character if in a field or line (I)
11	Control I	Cursor right, → , (I/O) or copy old character if in field or line (I)
12	Control J	Insert one line (NSD) (I), skip to first field on next line in NSL and zero or blank out field if immediately after Control Q. (NSL)
13	Control K	Cursor down, ↓ , (I/O), or copy all characters old line or field and terminate (I)
14	Control L	Defines low intensity display mode (I)
15	Control M (CR)	Field and line terminator (I)
16	Control N (X)	Defines normal display mode (NSD) (I). Copies identical field above up to but not including X (NSL) (I)
17	Control O (X)	Defines invisible ("off") display mode. Copies old field up to but not including X (NSL) (I)

20	Control P	Copies one character or field from previous line (NSD), or one character from identical field above (NSL) (I). Cursor addressing character (O), sets terminal in cursor address mode
21	Control Q	Deletes line or field (I), if given 3 times recreates picture.
22	Control R	Copies previous line (NSD) or identical field above (NSL) up to and including the last character without terminating (I). Represents normal display mode internally in NSHS (O)
23	Control S	Represents blink display mode internally in NSHS (O)
24	Control T	Represents low intensity display mode internally in NSHS (O)
25	Control U	Defines underline display mode (NSD). Represents underline display mode internally in NSHS (O)
26	Control V	Defines inverse video display mode (NSD) (I). Represents inverse video display mode internally in NSHS (O)
27	Control W	Represents invisible display mode internally in NSHS (O), used to terminate picture editing (I) in NSD.
30	Control X	Not used
31	Control Y	Erase screen (O)
32	Control Z (X)	Copy old field up to and including X (I)
33	Control Æ	Erase line [for output use only! This is ESCAPE!] (O)
34	Control Ø	Cursor up, ↑, (I/O) or copy all characters above line (NSD), or identical field (NSL) and terminate (I)
36	Control Å	Home, ↖ (I/O), and when in a line, copy previous field (NSD) (I)
36		Alternative to Control W for terminating picture editing (I).
37		Clear whole line (0), (NSL).

NSHS can accommodate up to 8 different types of terminals simultaneously. New VDU types can also be implemented, however experience has shown that this is not always easy; It could take 2 days or 2 weeks effort by an expert! If you intend to use new VDU types you should contact your local ND support organisation for advice in this matter.

APPENDIX C

ERROR MESSAGES AND CODES

The following error messages can occur in the NORD Screen Handling system:

- "no such picture"
- "no pictures defined on file"

In addition, the following can occur with or without NORD file system messages.

- "file not successfully opened"
- "error in reading first block of file"
- "error in writing first block of file"
- "error in closing scratched file"
- "first block of file damaged"
- "block size on file in error"
- "error in writing block back to file"
- "array not big enough for picture!!"
- "picture name table on file full"
- "entry number greater than number of pictures (internal error!)"
- "error in reading block from file"
- "first block of picture is not first block in chain"
- "not enough blocks available on file"

These error messages are written out with some tracing information;

```
<["possible file system error text"]
"error position identifiers are:  99999, 99999  source/object file"
"error message">
```

The tracing information can be particularly helpful in locating a bug in NSD.

In the NORD Screen Library system, error messages are printed on the last line of the terminal and only during the execution of an RFLDS call. These messages are in English and can be changed through editing by your local ND support organisation.

For all subroutine calls the return status parameter "IST" can have the values:

0:	call successfully executed
1:	unknown picture name or number
–N:	field number/indicator or picture number/name data-element-index number N is in error
2:	other errors

For other errors and at times for 1 or –N, an additional error code is placed in ITERM(7). These error codes are as follows:

Value:	*Meaning*
1	terminal common array in error
2	illegal picture number
3	no picture with this number in buffer

4	first field indicator is zero, but number of field indicators is not 1
5	number of field indicators <0 or > 255
6	line number in a field indicator is too large
7	no fields defined on this line
8	no fields defined on this picture
9	not so many fields on line
10	field number is zero
11	absolute field number > number of fields in picture
12	field numbers in wrong order
13	picture buffer full??
14	too many pictures requested, or buffer full
15	number of pictures in error
16	picture buffer damaged
17	too many field numbers given
18 + 100*EN	error in reading or opening picture file, here EN is the NORD file system error number. For example 66 18 is file system error 66 (decimal), 102 octal
19	no pictures defined on this picture file
20	not enough space in buffer for this picture
21	data element index array in error
22	picture file block damaged
23	picture description must not start at address zero
24	parameter "code" in error
25	number of field numbers < or = 0
26	all fields locked on an RFLDS or WFLDS call
27	line number in error
28	position number in error
29	area mode does not allow reading/removal of pictures (IPUBL(1) or IPRIV(1) ≠ 0)
30	no pictures in private buffer
31	incorrect program mode
32	no such picture file
33	user control illegal
50	area mode in error

APPENDIX D

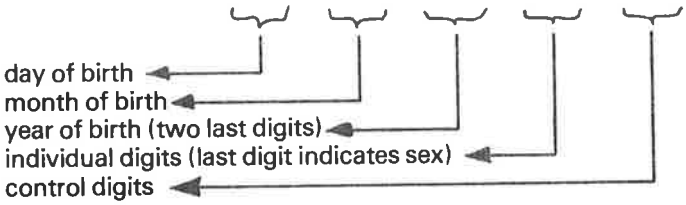
SYSTEM CONTROL

There are 6 system control functions implemented:

1 Person Number
2 Post Office Account Number - modus 10
3 Post Office/Bank Account Number - modus 11
4 Date (Year, month, day)
5 Date (Day, month, year)
6 Date (Month, day, year)

1. Person Number

Person Number must consist of 11 digits following the standard:



2. Post Office Account - Modus 10

The Post Office Account Number consists of up to 20 digits where the last digit is the control digit.

3. Post Office/Bank Account - Modus 11

The Post Office Account Number consists of up to 20 digits where the last digit is the control digit. The Bank Account Number must consist of 11 digits where the last digit is the control digit.

Note that the vector digits of points 1, 2 and 3 follow the Norwegian standard.

The default weight digits for modus 11 are 2,3,4,5,6,7,2,3,4 ...counted from right to left. This means that the last digit should be multiplied by 2; the one before by 3, etc. You may replace the existing digits in modus 11 by including your own weight digits. To define your own weight digits, a routine of one common block must be declared and loaded *after* the NSL.

The layout of the routine must be as follows:

```
BLOCK DATA
COMMON/PICDEF/IUSVEC (10)
DATA IUSVEC/1003B, 2005B, 3007B, 1003B, 2005B, 3007B
—,1003B, 2005B, 3007B, 11003B/
END
```

Note that the weight digits are stored in octal with two digits in each word, and are therefore best declared octally.

Example:

If you wish to have the following weight digits 1,2,3,4,5,6,7,1,2,3... in your system check on modus 11,
the "PICDEF" common will be:

```
BLOCK DATA
COMMON/PICDEF/IUSVEC (10)
DATA IUSVEC/402B, 1404B, 2406B, 3401B, 1003B, 2005B, 3007B
—, 402B, 1404B, 2406B/
END
```

4,5,
6. Date

If the year is within the range 1900 - 1999, the last two digits are suffcient, otherwise, all four digits (i.e., 1876) must be used.

Six additional date control numbers are used for the date system control to check the legality of the date input:

1 = date must equal today (i.e., equal to what SINTRAN III believes the date to be)

2 = date must be equal to or after today

3 = date must be after today

4 = date must be before or equal to today

5 = date must be before today

6 = no check of date

APPENDIX E

TROUBLE SHOOTING

1. If the pictures came out incorrectly on the VDU screen:

   a) Have you set the correct terminal type and the required picture displacement?

   b) If TDV 2000, is it in page or roll mode and if so, have you set bit 15 of the terminal type?

   c) Have you set break strategy to 0? If not, do you have a SINTRAN III system with a version date later than 1977?

   d) If INFOTON 200, is the terminal in Page Mode?

Otherwise

2. Have you checked the status value returned by all the picture system routines? If this status is 2, have you checked the value of the error code in ITERM(7)?

   A common error code is 7618, which means that the user calling GTPIC does not have a read and common access to the picture file.

3. Have you set up the terminal buffer, the private buffer and the public buffer correctly?

4. If your program has inexplicable or mysterious errors, are you explicitly using any of the "internal" routines in the NSL? If so, are these "assembly" routines and have you declared them as such in all subroutines which use them?

APPENDIX F

USER CONTROL, AN EXAMPLE

When the system encounters a field that has been assigned Field Control Function 9 (User Defined Control Algorithm), the subroutine UCONT is called *automatically* by RFLDS both before the field is entered and after a value has been given. When UCONT is activated, its status parameter, IST, on entry will be −1. After a value has been read successfully to the field, IST will be +1 and Read-fields will proceed to the next field. The value −1 returned after a field has been read indicates that UCONT has rejected the value typed in and at that point Read-fields will reposition the cursor to the beginning of the field so that a new value can be given.

Your control algorithm can be designed so as to give IST different values, but take particular notice of the following:

When Read-fields automatically activates UCONT *both* before and after:

1) If IST is greater than 99, Read-fields will be aborted giving the same value for status (i.e. > 99) and the terminating character in Read-fields will be −1, just as if the escape character had been typed.

2) If IST is equal to zero, the value of the relevant data element will be accepted and written out in the field.

3) If the error code in ITERM(7) is not equal to zero on return from the main control routine, an NSL error condition exists. ITERM(7) will most likely be 33 indicating that a User Control has malfunctioned.

The following internal NSL routines may be useful in UCONT.

ICHEK

ICHEK is a FORTAN integer function which checks the field-number-array and data-element-index-array and returns the address of the picture description table. The function returns 0 if everything is OK.
ICHEK is called as follows:

IST = ICHEK (0,IPN,N,IFNARAY,IDIARAY,IADDR)

WHERE:

IPN	is the picture number.
N	is the number of fields.
IFNARAY	is the field-number-array.
IDIARAY	is the data-element-index-array.
IADDR	is where the address of the picture description table is placed.
IST	is 0 on return if everything is OK.

Note that the first parameter must be 0 due to another internal use of this function.

ITBIT

ITBIT is an assembly function to check a bit in an array.
ITBIT must be declared asssembly.
ITBIT is called as follows:

```
ASSEMBLY ITBIT
    ----
    ----
IST = ITBIT(IARRAY,IBIT)
```

WHERE:

IARRAY	is the actual array.
IBIT	is the actual bit.
IST	returns the content (0 or 1).

G F I N F

GFINF is used to get all information about a field to a field information array.
GFINF is an assembly routine and must be declared as assembly in the call.
GFINF is called as follows:

```
DIMENSION IFINFO(22)
ASSEMBLY GFINF
        --
        --
CALL GFINF (IFINFO,IFN,IADDR)
```

WHERE:

IFINFO	is the 22 cells field information array in which the field information is placed.
IFN	is the absolute field number.
IADDR	is a return parameter from the ICHEK function and isthe address of the picture description table.

FIELD INFORMATION ARRAY (IFINFO):

IFINFO (1) - number of characters read.
IFINFO (2) - fill code (space or 0).
IFINFO (3) - sign code (—1 if not defined or space/0 or 55B (—)).
IFINFO (4) - edit code (1—29).
IFINFO (5) - number of significant characters (not including edit codes etc.).
IFINFO (6) - number of positions (including everything in a field).
IFINFO (7) - storage code (1—5).
IFINFO (8) - logical device number for terminal.
IFINFO (9) - code for read terminate.
IFINFO (10) - not used.
IFINFO (11) - line number of field.
IFINFO (12) - column number of field.
IFINFO (13) - field number of identical field above (if any).
IFINFO (14) - control read flag (0 or 1).
IFINFO (15) - no display flag, if 1 then display modes are dropped.
IFINFO (16) - length of data element in words.
IFINFO (17) - control type.
IFINFO (18) - number of words of control information.
IFINFO (19) - address of control information.
IFINFO (20) - code for internal escape function.
IFINFO (21) - return strategy (code = 0—23).
IFINFO (22) - break strategy.

Example of user control where control number 1 checks that the product of the input integer * 2 is not greater than 10000. Control number 2 checks if the BCD input consists of 4 digits and is an even number. Control number 3 converts the input integer to the same number expressed in letters and writes it out in the next field

SUBROUTINE UCONT (ICOD, IPN, IFINFO, N, IFNARAY, IREC, IDIARAY, INDEX, IREDBITS, IST)

WHERE:

ICOD	is the user control number.
IPN	is the picture number.
IFINFO	is the field information array.
N	is the number of fields.
IFNARAY	is the field number array.
IREC	is the data record.
IDIARAY	is the data element index array.
INDEX	is index to this field number.
IREDBITS	is the bit array where information 'if field read' is found.
IST	is −1 on entry if nothing read and returns the status after the call.

```
Dimension IFNARAY (50),IREC(50),IDIARAY(50),IREDBITS(20)
Character CHAR(10)*6
Dimension IQST(1)
Dimension IFINFB(22)
Equivalence (IQST(1),CHAR(1) )
Assembly GFINF,ITBIT

DATA CHAR(1)/'ONE   '/
DATA CHAR(2)/'TWO   '/
DATA CHAR(3)/'THREE '/
DATA CHAR(4)/'FOUR  '/
DATA CHAR(5)/'FIVE  '/
DATA CHAR(6)/'SIX   '/
DATA CHAR(7)/'SEVEN '/
DATA CHAR(8)/'EIGHT '/
DATA CHAR(9)/'NINE  '/
DATA CHAR(10)/'TEN  '/

C     GET ADDRESS OF PICTURE DESCRIPTION TABLE:
      If (IST.EQ.−1) go to 72
      If (ICHEK(0,IPN,N,IFNARAY,IDIARAY,IADDR).NE.0) go to 71

C     FIND INDEX IN RECORD ARRAY:
      IACT = IDIARAY(INDEX)

C     GO TO ACTUAL CONTROL:
      22  Go to (1,2,3,70) ICOD

C     CHECK IF INTEGER:
      1   If (IFINFO(7).NE.1) go to 72

C     DO THE ACTUAL CHECK:
      If (IREC(IACT)*2.GT.10000) go to 72
```

```
C     CHECK IF BCD:
      2    If (IFINFO(7).NE.5) go to 72

C     CHECK IF 4 CHAR. READ:
      If (IFINFO(1).NE.4) go to 72

C     CHECK IF EVEN:
      If (IREC(IACT).and.18).NE.0) go to 72
      go to 70

C     CHECK IF INTEGER:
      3    If (IFINFO(7).NE.1) go to 72

C     CHECK INPUT
      If (IREC(IACT).LT.1.or.IREC(IACT).GT.10) go to 72
      J = IREC(IACT)

C     GET INDEX TO NEXT FIELD:
      IACT = IDIARAY(INDEX + 1)

C     GET ABSOLUTE FIELD NO. TO NEXT FIELD:
      IF N = IFNARAY(INDEX + 1).and.17777B

C     GET FIELD INFORMATION TABLE OF NEXT FIELD:
      Call GFINF(IFINFB,IFN,IADDR)

C     CHECK IF IT IS ENOUGH ROOM IN THIS FIELD:
      If (IFINFB(5).LT.6) go to 71

C     PLACE STRING IN DATA RECORD:
           J = J*3–2
           DO 31 I = 1,3
      31   IREC(IACT–1 + 1) = IQST(J–1 + 1)
      32   Call WFLDS(1,IPN,N,IFNARAY,IREC,IDIARAY,IST)
           If (IST.NE.0) Pause 4

C     OK RETURN:
      70   IST = 0
           Return

C     SKIP TO NEXT FIELD:
      71   IST = 1
           Return

C     CONTROL WRONG – TRY AGAIN:
      72   IST = –1
           Return

           End
```

APPENDIX G

STANSAAB USED BY NSHS

According to the STANSAAB terminology there are only three types of fields:

1. protected fields
2. unprotected fields
3. input fields

NSHS uses only protected and input fields. Everything on the screen that is not an input field will be a protected field.

The comments and some of the NSHS calls specified below apply only to STANSAAB, and are not necessarily described elsewhere in this manual.

RFLDS (read fields)

None of the control code features have any meaning for STANSAAB and are omitted. The following edit code features have no meaning for STANSAAB and are omitted:

Edit codes: 0,1,2,3,4,5,6,7,8,9,
B,C
N,O,P,Q,R,S

The various function keys on the STANSAAB have been implemented to give the following terminating character values:

XXX	= regenerate picture
AVBRYT	= –2
MEDDEL VÄNTAR	= –1
P1,P2,P3,P4,P5	= 1,2,3,4,5
Shift P1,P2,P3,P4,P5	= 6,7,8,9,10
PF1,PF2	= 11,12
PF10,PF11,PF12–PF36	= 20,21,22–46

PF1–PF36 return in addition any typed-in fields. If an error occurs in one or more input fields, information about this is returned in the IST parameter.

This parameter contains 100+ the field number of the first incorrect field on a picture. In addition, all incorrect fields will have the "must" bit in the field number set.

WFLDS (write fields)

Only three display codes may be used for fields:

0 = normal
3 = high intensity
5 = invisible

This call does not write out fields where the lock bit is set.

WRPTD (write picture to display)

Only three display codes may be used for texts:

0 = normal
3 = high intensity
5 = invisible

WRPTF (write picture to file)

If this call is used with the STANSAAB as output file, the flag parameter must be equal to 0 to ensure that the STANSAAB codes for protected or input fields are output.

If you wish to have an output on a line printer this parameter must be equal to 1 to prevent these special codes from being printed.

The WRPTF call writes "....." to fields where the lock bit is set.

CLSCR (clear screen)

The whole screen is always cleared.

WMSGE (write message)

No "pip" is given.

ZLOCK/RLOCK

Sets/resets the lock bit in the field number(s)

ZMUST/RMUST

Sets/resets the MUST-BIT in the field number(s)

The respective field numbers' field control character on the screen must be set correspondingly. The user must do this himself using the subroutine SETDM (set-display-mode)

CALL SETDM (<code>, <must>, <picture-number>, <number-of-fields>, <field-indicator-array>, <status>).

WHERE:

code = display mode
1 = normal intensity
3 = high intensity
5 = invisible
6 = operator protected field, normal intensity
9 = operator protected field, high intensity
11 = operator protected field, invisible

must = 1 the field is always sent to the computer.

must = 0 the field is sent only if the operator has entered characters into it.

Fields in which the Lock bit has been set to 1 must have a display mode corresponding to a protected field.

The user is warned that the Must-bit in field numbers is also used to indicated which fields contained illegal characters after a read-fields call. Thus he must reset Must bits before the next SETDM call.

ZREAD/RREAD

No meaning for STANSAAB

ZBELL (set bell)

Sends bell which lights the "MEDDEL VÄNTAR" button and the terminal gives its characteristic tone.

CLFRM (clear from)

CALL CLFRM (<picture number>, <field indicator>, <status>)
Clears the input fields from a given position to the end of the screen.

SETDM (set-display-mode)

CALL SETDM (code, picture number, field indicator, status)
Set display mode character in front of a field.

APPENDIX H

NSL AND COBOL

COMMON AREAS

If an application program is written in COBOL, a FORTRAN BLOCK DATA subroutine must be loaded prior to loading NSL and normally this happens after the COBOL program has been loaded.

The BLOCK DATA subroutine dimensions and initiates the NSL Private and Public common areas.

Example:

```
BLOCK DATA
COMMON/PRIVATE/ITERM(128), IPRIV(1920)
COMMON/PUBLIC/IPUBL(6)
DATA ITERM/1,3,7,0,0,0,0,0,0,1920,118 * 0/
DATA IPRIV/0,1920,3,0,0, 1915 * 0/
END
```

Once again, the lengths and contents of IPRIV along with the contents of the *fisrt* 10 words of ITERM, are used here solely as an example. The contents of ITERM and IPRIV can be set from a COBOL program by calling NSL subroutine COSCR1.

```
                SUBROUTINE COSCR1(IARR)

C     PARAMETERS:
C     INTEGER ARRAY IARR(8),
C     IARR(1) = DEVICE NR. FOR TERMINAL
C     IARR(2) = TERMINAL TYPE AND PICTURE DISPLACEMENT
C     IARR(3) = ESCAPE CHARACTER
C     IARR(4) = READ STRATEGY
C     IARR(5) = PROGRAM MODE
C     IARR(6) = BREAK AND ECHO STRATEGY
C     IARR(7) = OPTION WORD
C     IARR(8) = MAX NR OF PICTURES

                INTEGER IARR(1)
                COMMON/PRIVATE/ITERM(128),IPRIV(30)
                COMMON/PUBLIC/IPUBL(6)
                ITERM(1) = IARR(1)
                ITERM(2) = IARR(2)
                ITERM(3) = IARR(3)
                ITERM(4) = IARR(4)
                ITERM(5) = IARR(5)
                ITERM(6) = 0
                ITERM(7) = 0
                ITERM(8) = IARR(6)
                ITERM(9) = 1915
                IPRIV(1) = 0
                IPRIV(2)-1915
                IPRIV(3) = IARR(8)
                IPRIV(4) = 0
                IPRIV(5) = 0
                RETURN
                END
```

Values for ITERM(6) and ITERM(7) are returned from the subroutine ERRORC.

```
      SUBROUTINE ERRORC(ITERM7,ITERM6)
      COMMON/PRIVATE/ITERM(128)
C     ITERM7:  NSL ERROR-CODE ITERM(7)
C     ITERM6:  CURSOR POSITION ITERM(6)

      ITERM7 = ITERM(7)
      ITERM6 = ITERM(6)
      RETURN
      END
```

COBOL-FORTRAN INTERFACE

In COBOL there is no data type that corresponds to the FORTRAN data type "CHARACTER". Consequently, new entry points to some of the FORTRAN subroutines have been designed especially to overcome this language incompatability. These routines can be distinguished by their suffix, namely, the letter C (ERRORC, OPENFC, CLOSEC, GTPICC, WMSGEC).

A file is opened in a COBOL program by calling the subroutine OPENFC:

```
      SUBROUTINE OPENFC(IPNAS,IACC,IFTNR,IST)
      INTEGER IP NAS(1),IFNAM(20)
      CHARACTER NAME*40
      CHARACTER ACS*2
      EQUIVALENCE (IFNAM(1),NAME)

C     IPNAS: FILE NAME STRING.
C     IACC  : ACCESS
C             0 = SEQUENTIAL WRITE
C             1 = SEQUENTIAL WRITE APPEND.
C     IFTNR: FORTRAN FILE NUMBER.
C     IST    : RETURN STATUS
C             0 = OK.
C             –1 = ERROR IN PARAMETER IACC.
C             OTHER = FORTRAN ERRCODE.


      IF (IACC .NE. 0 .AND. IACC .NE. 1) GOTO 91
      ACS = 'W'
      IF (IACC .EQ. 1) ACS = 'WA'
      DO 10 I = 1,20
10    IFNAM(I) = IPNAS(I)
      OPEN (IFTNR,FILE = NAME, ACCESS = ACS,
      STATUS = 'UNKNOWN',ERR = 92)
      IST = 0
      GOTO 999
91    IST = –1
      GOTO 999
92    IST = ERRCODE
999   RETURN
      END
```

To close the file, use subroutine CLOSEC:

```
                SUBROUTINE CLOSEC(IFTNR)

C     IFTNR: FORTRAN FILE NUMBER.
C            –1 = CLOSE ALL OPENED FILES.


                CLOSE (IFTNR)
                RETURN
                END
```

Example:

As you read through this example of a COBOL program you will notice that whenever string parameters are used in Fortran, COBOL demands PIC variables.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.
  Y-001
DATA DIVISION.
WORKING-STORAGE SECTION                          COMMENTS
01      ITERMC    COMP.                          (Terminating character)
01      NOFR      COMP.                          (No. of fields read)
01      ISTATUS   COMP.
01      NOF       COMP.                          (No. of fields)
01      W-0       COMP       VALUE 0.
01      W-1       COMP       VALUE 1.
01      W-2       COMP       VALUE 2.
01      W-3       COMP       VALUE 3.
01      W-4       COMP       VALUE 4.
01      W-5       COMP       VALUE 5.
01      W-6       COMP       VALUE 6.
01      W-7       COMP       VALUE 7.
01      W-8       COMP       VALUE 8.
01      IFTNR     COMP       VALUE 20.  (Fortran file No.)
01      IDEIA     PIC X(40).            (Data element index array)
01      IFNA      PIC X(40).            (Field No. array)
01      IPNA      PIC X(16).            (Picture No. array)
01      IREC      PIC X(400).           (Record)
01      T-INITA.
        03 INITA   COMP       OCCURS 8
01      IPNAS     PIC X(64).            (Picture name "string")
01      IPFNS     PIC X(81).            (Picture file name "string")
01      IMSG1     COMP.                 (NSL-error code)
01      IMSG2     COMP.                 (Cursor Position)
```

```
PROCEDURE DIVISION.
A000.
        MOVE1                   TO INITA (1).
        MOVE 5                  TO INITA (2).
        MOVE -32768             TO INITA (3).
        MOVE 0                  TO INITA (4).
        MOVE 0                  TO INITA (5).
        MOVE 0                  TO INITA (6).
        MOVE 0                  TO INITA (7).
        MOVE 8                  TO INITA (8).
        MOVE "(SI-KAL)TOR:OBJ"      TO IPFNS.
        MOVE "P1   P2   P3"   TO IPNAS.
        CALL "COSCRI" USING T-INITA.
        DISPLAY W-0.
        CALL "GTPICC" USING IPFNS  W-3  IPANS  IPNA  ISTATUS.
        DISPLAY W-1.
        IF ISTATUS NOT = 0 GO TO ERRSTA.
        CALL "CLSCR" USING  W-0  W-0  W-0  W-0  ISTATUS.
        DISPLAY W-2.
        IF ISTATUS NOT = 0 GO TO ERRSTA.
        CALL "GTFDN" USING  W-2  W-1  W-0  IFNA  NOF  ISTATUS.
        DISPLAY W-3.
        IF ISTATUS NOT = 0 GO TO ERRSTA.
        CALL "WRPTD" USING  W-2  ISTATUS.
        DISPLAY W-4.
        IF ISTATUS NOT = 0 GO TO ERRSTA.
        CALL "CLBUF" USING  W-2  NOF  IFNA  IREC  IDEIA
        ISTATUS.
        DISPLAY W-5
        IF ISTATUS NOT = 0 GO TO ERRSTA.
        CALL "RFLDS" USING  W-0  W-2  NOF  IFNA  IREC  IDEIA
            NOFR  ITERMC  ISTATUS.
        DISPLAY W-6.
        IF ISTATUS NOT = 0 GO TO ERRSTA.
        MOVE  "L-P"             TO IPFNS.
        CALL  "OPENFC"  USING IPFNS W-0
        IFINR ISTATUS                       (Open for sequential write)
        DISPLAY W-7.
        IF ISTATUS NOT = 0 GO TO ERRSTA.
        CALL  "WRPTF"  USING  IFTNR  W-1  W-0  W-2  NOF  IFNA  IREC  IDEIA
        ISTATUS
        DISPLAY W-8
        IF ISTATUS NOT = 0 GO TO ERRSTA.
        CALL "CLOSEC" USING IFTNR.
        MOVE "YOU MADE IT !!!!!" TO IPFNS.
        CALL "WMSGEC" USING IPFNS.
        GO TO 10.
ERRSTA.
          CALL "ERRORC" USING  IMSG1  IMSG2
          DISPLAY IMSG1 IMSG2.
10.
        STOP RUN.
```

Unused Byte Positions

Cobol programmers should note that unlike COBOL where a field is byte-oriented, NSL is *word-oriented*. In other words, in NSL, fields can contain unused bytes. This situation will occur for example with Storage Code 3, if the number of significant characters in addition to a possible sign renders an uneven number of bytes. When programming in COBOL therefore, it is wise to define byte fields such that they use an *even* number of positions. The unused position is filled with the pad character.

Leading Bytes in Numeric Fields

In NSL, numeric fields stored as bytes will always have the unused leading positions filled with spaces (ASCII Code 40B). Cobol uses zeroes (ASCII Code 60B). Therefore in some instances it may be necessary to convert COBOL to NSL on entry and NSL back to COBOL on exit.

A conversion routine has been designed to accomplish this. On entry to NSL, that is to subroutines using a record as a parameter, leading zeroes in numeric fields are converted to spaces and conversely on exit, leading spaces are converted to zeroes. This routine also converts on entry a trailing + sign to space and on exit a trailing space to a + sign.

To activate this conversion function you must set the Cobol flag, bit 15, in ITERM (3). This is accomplished by setting IARR(3) to −32768 plus the escape character, before calling COSCRI.

Note that in NSHS because the sign byte is always trailing, COBOL numeric variables should be defined with 'SIGN TRAILING SEPARATE' clause.

APPENDIX I

SINTRAN FEATURES

Break and Echo Strategy

SINTRAN-78 contains newly implemented features which, among other things, make character input/output to NSHS more efficient. These enhancements in effect, allow you to choose between two break and echo strategies. By setting the strategy in ITERM(6) to 1, the system upon entering a field, will select a strategy table that, even though it causes the terminal driver to echo legal characters, *will not generate* a break until the last character (maximum for the field) has been echoed. Should an illegal character be entrered, it will not be echoed. Instead, a break will be generated and NSL will switch back to the normal strategy, that is, to option 0, whereby for the rest of the field break and echo occurs on every character.

If you do not have SINTRAN-78, the option described above is not applicable; you have no choice but to enter 0 in ITERM(6).

Both options utilise strategy tables. The tables are generated automatically and are presented here solely for background information.

When ITERM(6) = 0, the following table is operative:

```
BRK0: = (−1, −1, −1, −1, −1, −1, −1, −1)
ECH0: = (0, 0, 0, 0, 0, 0, 0, 1)
```

When ITERM(6) = 1, depending on the edit code of the field being read, table 3 or 4 or 5 is used.

```
BRK3: = (−1, −1, −1, 100077, −1, −1, −1, −1) % No break on 1 or 9
BRK4: = (−1, −1, 77777, −1, 100000, 37, 10000, 37) % No break on letters
BRK5: = (−1, −1, 0, 0, 0, 0, 0, 1) % No break on alphanumeric

ECH3: = (−1, −1, −1, 10077, −1, −1, −1, −1) % echo 1−9
ECH4: = (−1, −1, 77777, −1, 100000, 37, 100000, 37) % echo letters
ECH5: = (−1, −1, 0, 0, 0, 0, 0, 1) % echo alphanumeric
```

Outstring (OUTST) and 8 character outbyte (M8OUT)

These enhancements to SINTRAN-78 affect NSL in the sense that output is more efficient. They are activated by setting either bit 3 or bit 4 in the Option Word ITERM(9). Choose one - but not both. If you do not have Sintran -78 these options are not applicable.

NORSK DATA A.S
Postboks 4, Lindeberg gård
Oslo 10, Norway

COMMENT AND EVALUATION SHEET

NORD SCREEN HANDLING SYSTEM
APRIL 1979

ND.60.088.02

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 _____

_____

_____