UE-Library Reference Manual

ND-20.004.1 EN

UE-Library Reference Manual

ND-20.004.1 EN

PRINTING RECORD	
PRINTING	NOTES
06/86	Version 1 EN

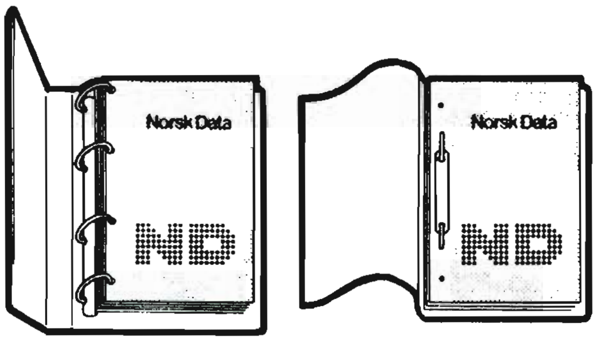
UE–Library Reference Manual
Publ.No. ND–20.004.1 EN

UPDATING

Manuals can be updated in two ways, new versions and revisions. New versions consist of a completely new manual which replaces the old one, and incorporate all revisions since the previous version. Revisions consist of one or more single pages to be merged into the manual by the user, each revised page being listed on the new printing record sent out with the revision. The old printing record should be replaced by the new one.

New versions and revisions are announced in the ND Customer Support Information and can be ordered from the address below.

The reader's comments form at the back of this manual can be used both to report errors in the manual and give an evaluation of the manual. Both detailed and general comments are welcome.



RING BINDER OR PLASTIC COVER

The manual can be placed in a ring binder for greater protection and convenience of use. Ring binders may be ordered at a price of NKr. 45.- per binder.

The manual may also be placed in a plastic cover. This cover is more suitable for manuals of less than 100 pages than for larger manuals.

Please send your order, as well as all types of inquiries and requests for documentation to the local ND office, or (in Norway) to:

Norsk Data A.S
Graphic Center
P.O.Box 25 BOGERUD
N-0621 OSLO 6 - Norway

I would like to order

.... Ring Binders, 40 mm, at NOK 45.- per binder

.... Plastic Covers, at NOK 10.- per cover

Name: ....

Company: ....

Address: ....

THE MANUAL:

This manual describes UE-LIBRARY (ND-210601) and UE-ERRORS (ND-250178) version C. The library may be used to convert standard error message numbers to a standard text, set and fetch the UE/JEC part of the terminal datafield and set and fetch completion codes.

Using this library ensures consistent error messages to the user.

CHANGES TO UE-LIBRARY FROM THE B-VERSION:

- The file names are changed according to the new standards for library file naming. See chapter 4.
- The name of the error file is changed from UE-ERMSG-xx-Byy to UE-ERMSG-xx-Cyy where xx is the language and yy is the the revision of the UE-ERRORS product containing the messages. The default file name is then UE-ERMSG-EN-C.
- The SYSID parameter in the error message calls is changed from a BOOLEAN parameter to a flag word. See chapter 2.2.2.
- The UEERINIT/UEERDINIT/UEIEINI/UEIEDIN calls checks if the error message file is already open before trying to open it if called from background programs.
- The UEEREXIT/UEIEEXI calls may be omitted for background programs.
- The error messages may be tagged by the message number and also contain characters outside the 7 bit ASCII range. The messages are stored in a format suitable for calling VTM for output.

UE-ERRORS

- Due to several complaints about incompatible file and product versions for ND-210586 it has been discontinued after version C (UE-ERMSG-xx-Byy:ERR).
- Instead a new product, ND-250178, has been defined. To keep the product version equal to the version of UE-LIBRARY utilizing the resulting :ERR file, versions A and B have been skipped both for the product and the :ERR files. This also has allowed for the :ERR files to keep the names used on ND-210586.
- The :ERR files on ND-250178Cyy are UE-ERMSG-EN-Cyy:ERR (file number 208210C) and UE-ERMSG-NO-Cyy:ERR (file number 208211C).
- The changes above are administrative only and does not imply any changes to the product contents.

RELATED MANUALS/REPORTS:

ND-60.179 - User Environment Reference Manual
ND-60.261 - User Environment Library Routines

TABLE OF CONTENTS

1 Introduction

UE-LIBRARY contains routines to convert standard error message numbers to standard texts, to print them on the error device, to set and fetch UE/JEC info in the terminal datafield, to set and fetch termination codes and some useful functions which was needed by User Environment, but not previously available.

Several of the routines are available to public users via the library UE-PLIB (see notes below!).

WHEN USING THE ERROR MESSAGE CALLS, THE MESSAGES SHOULD ALWAYS BE TAGGED BY THE ERROR NUMBER. SEE CHAPTER 2.2.2.

CURRENT RESTRICTIONS:

SINTRAN version J or later must be used to achieve full functionality.
SINTRAN version G or earlier can not be used without patching. See chapter 2.1.3.
If intermixing UE-PLIB and UE-LIBRARY, these libraries must have the same version letters.

2 The library routines

This chapter describes the available routines both with PLANC and with FORTRAN declarations.

Routines present in the library, but intended for internal UE/TRUE/JEC use only, are not documented.

2.1 General information

2.1.1 Calling from PLANC

These routines begin with UE. In the parameter descriptions RW means that the parameter is updated by the routine, while W means that the parameter is a value out parameter.

For the error message handling routines ERRETURN is used to report error conditions from the error message handling routines, while value out is used for most of the others.

2.1.2 Calling from FORTRAN/COBOL/Pascal

These routines begin with UEI. In the parameter descriptions RW means that the parameter is updated by the routine, while W means that the parameter is a value out parameter.

All strings are HOLLERITH (INTEGER ARRAYs). CHARACTER may thus not be used!

Errors are reported by setting the status variable to a non zero value when present.

The FORTRAN calls initiate a stack (200 or 600 words depending on which routine is called) and then they generally call the corresponding PLANC routines.

2.1.3 Using SINTRAN version G or earlier

Several of the routines in the library utilizes functions available in later versions of SINTRAN only. Generally the error message routines will work fine for all versions of SINTRAN (except for the reuse of the already open error message file), MN317 will use MON 70 if SINTRAN is older than the I version, UEGFLAGS will give fixed response if SINTRAN is older than the J version and the other routines will return without doing anything except setting all output values to zero if SINTRAN is older than the I version.

To detect which SINTRAN services are available, the library uses two routines, UEMN312 and UESINVER. ***THESE WILL NOT WORK IF SINTRAN IS OLDER THAN VERSION H!*** Instead SINTRAN will abort the program with the error message "ILLEGAL MONITOR CALL"!

To run using SINTRAN G or older, the following dummy PLANC routines must be made and be loaded before loading UE-LIBRARY/UE-PLIB:

UEMN312 This routine checks if a monitor call is available or not using the monitor call 312 (MOINF) available from SINTRAN H on the ND 100 and from SINTRAN I on the ND 500. (When called on the ND 500, the routine will use UESINVER below and return FALSE if the SINTRAN version is older than I).

```
ROUTINE VOID,BOOLEAN(INTEGER) : UEMN312(Moncall_no)
```

The routine must return FALSE for SINTRAN G or older!

UESINVER This routine returns the SINTRAN version letter using the monitor call 262 (CPUST) available from SINTRAN H.

```
ROUTINE VOID,BYTE : USEINVER
```

The routine must return <= #G (107 octal) for SINTRAN G or older!

2.2 Error message handling routines

2.2.1 INIT and EXIT

When using the error message handling routines, the first call must be either of the INIT calls (UEERINIT/UEERDINIT/UEIEINI/UEIEDIN).

The INIT call opens the file containing the error messages and prepares a window area to be used towards the file. If called from a background program it will check if the error file is already open and if so, use the open file number of the opened file. This check is not done if called from RT as the ND 500 monitor will close the error file if opened from an ND 500 program which terminates causing problems for other programs. Besides, ND 500 programs fails to find the open file number of a file not opened by the program itself.

If running from RT, the program must call either of the EXIT routines before terminating (UEEREXIT/UEIEEXI). These calls close the error message file. If omitted the RT-open file table will probably be filled by unused :ERR files!

If running from background the EXIT call may be omitted and, if the file is not closed automatically because of termination of the program, the error message file will remain open and will be used directly by the next program. Files are not closed on the ND 100 if calling programs using MN317.

2.2.2 Flag word used in the message routines

The flag word used in the message calls may take the following values (the least significant bit is bit 0):

Bit UEERSBIT (0) (value = 1)	set:	The error code parameter contains an SSI code and thus the name of this series is returned / displayed.
	reset:	The error code parameter contains an error code and thus the message corresponding to this number is returned / displayed.
Bit UEERNBIT (1) (value = 2)	set:	The error code number should be included in front of the message.
	reset:	Do not include any number tag.
Bit UEERCBIT (2) (value = 4)	set:	Character set information in the messages is OK and thus returned.
	reset:	Only 7-bit ASCII is returned. Any extended character set information is stripped off.

Clear the flag word and then set the bits or add the values given in the parantheses.

Recommended flag values:

If using VTM (directly) for output:
(4: Message without number tag)
5: Name of the system (SYSID)
6: Message tagged with the message number

If not using VTM for output:
(0: Message without number tag)
1: Name of the system (SYSID)
2: Message tagged with the message number

THE MESSAGE TAG SHOULD ALWAYS BE INCLUDED AS IT EASES SUPPORT (THE MESSAGES THEMSELVES ARE TRANSLATED INTO SEVERAL LANGUAGES). WHEN USING VTM (DIRECTLY) FOR OUTPUT (E.G. BY USING VTWRIT), THE ALTERNATIVE CHARACTER SET INFO SHOULD ALSO BE INCLUDED (SEE CHAPTER 2.2.5).

UEERDMSG will ignore the UEERCBIT flag if set.

2.2.3 Dynamic fields

Some programs need messages that contain dynamic text or number fields. UE-LIBRARY offers one text and one number field per message. The format message routines detects the sequences @A, @X, @I, @O and @H in the messages and replace them with texts or numbers. In addition @@ is replaced by a single @.

Conversions done by UE-LIBRARY formatting routines:

@A is replaced by the text parameter
@X is replaced by the text parameter
@I is replaced by the number parameter in decimal format
@O is replaced by the number parameter in octal format
@H is replaced by the number parameter in hexadecimal format
@@ is replaced by a single @

The programs using the UEERMSG call should do the same. See chapter 3.2 for further information.

2.2.4 Standard error message

If the UEERMSG or UEERFMSG call returns zero message length, no message is present in the error file for this number.

To give the user some indication of something being wrong, the UEERFMSG routine may then be called with error code 7707B and with the original error code as the dynamic number. Thus a standard message will be returned:

```
        Error @O (octal). No message available.

Ex:     ERRCODE = 56B
        UEERFMSG(7707B,4,'',ERRCODE,FALSE,0,STRING,LENGTH)
        -> "Error 56 (octal). No message available."
```

2.2.5 Extended character sets

When setting the UEERCBIT in the flag word, possible extended character set information is included in the messages. This means that the messages may contain characters outside the standard ASCII character set range as the french à and â.

The message is stored in VTM format and may thus be written with VTWRIT without any conversion. Note that after displaying the message, the current output character set will be that of the last character in the message outside the ASCII range (if any). Also note that libraries calling VTM for output (e.g. NDP, FOCUS) will generally NOT accept the VTM format!

This also applies when including dynamic text fields containing characters in alternative character sets. The library will by itself, however, switch (back) to any alternative character set needed by the message. The dynamic text should NOT contain any characters from alternative sets, nor contain any VTM commands, if output is done via UEERDMSG or UEEROMSG as these calls don't use VTM!

If this bit is reset, the VTM codes are stripped off and so is the alternative character set bit (bit 7). As the accented characters so far are stored as a single character in character set 3, the letter "à" becomes "!"... Later, however, floating accents may be expected and then the à becomes a.

AS WE WANT TO SELL IN THE WHOLE OF EUROPE, INCLUDING FRANCE, ALL APPLICATIONS SHOULD ACCEPT THE CHARACTER SET INFO (SEE CHAPTER 2.2.2)!

2.2.6 UEERINIT/UEERDINIT/UEIEINI/UEIEDIN - Initialization

These routines opens the error message file holding the messages if the file isn't already open. See chapter 2.2.1 for further details. Either of these routines must be called before the other error message routines may be used.

There are two sets of routines, the general UEERINIT/UEIEINI set and the UEERDINIT/UEIEDIN set which supplies defaults for both the error file name (UE-ERMSG-xx-C) and the window buffers (3*256 bytes).

Format:

```
        ROUTINE VOID,                 & % No invalue
                VOID(                 & % No outvalue
                BYTES) :              & % Language code
                UEERDINIT               % Init call using default values

        SUBROUTINE UEIEDIN(           & % Init call using default values
                   HOLLERITH,         & % Language code (2 chrs)
                   INTEGER W)           % Status

        ROUTINE VOID,                 & % No invalue
                VOID(                 & % No outvalue
                BYTES,                & % Name of the ermsg file
                BYTES POINTER) :      & % Pointer to window buffer
                UEERINIT                % Init call

        SUBROUTINE UEIEINI(           & % Init call
                   HOLLERITH,         & % Name of the ermsg file
                   INTEGER,           & % Length of the name
                   INTEGER ARRAY,     & % Window buffer
                   INTEGER,           & % Length of buffer in BYTES!
                   INTEGER W)           % Status
```

Parameters:

Language code
    The language code in the UEERDINIT/UEIEDIN call is the standard two letter coding (EN, NO, ...).

Name of the ermsg file
    This is normally UE-ERMSG-xx-C, xx being the language code as above. The revision part of the error message file name (ex. 00 in UE-ERMSG-EN-C00) shall not be included in the name parameter! And only one UE-ERMSG-xx-C:ERR file shall be present on any system and then the one having the highest revision number.

Window buffer
    The window buffer must be static data (not located on a stack). Its size must be n*256 bytes where 1 <= n <= 20 and be aligned on a 16 bits word boundary. The window buffer is used by UEERMSG as windows against the error message file.

2.2.7 UEEREXIT/UEIEEXI - Termination

This routine closes the error message file and the error conversion calls may therefore no longer be used. If this call is omitted when an RT (ND 100) program terminates the message file will remain open! See chapter 2.2.1 for further details.

Format:

```
        ROUTINE VOID,                & % No invalue
                VOID :               & % No outvalue
                UEEREXIT               % Termination

        SUBROUTINE UEIEEXI(          & % Termination
                   INTEGER W)          % Status
```

2.2.8 UEERMSG/UEIEMSG - Get message

This routine converts an error code to a message.

Format:

```
        ROUTINE VOID,                 & % No invalue
                VOID(                 & % No outvalue
                INTEGER,              & % Error code
                INTEGER,              & % Flag word
                BYTES W,              & % Error message
                INTEGER W) :          & % Length of the message
                UEERMSG                 % Get message

        SUBROUTINE UEIEMSG(           & % Get message
                   INTEGER,           & % Error code
                   INTEGER,           & % Flag word
                   HOLLERITH W,       & % Error message
                   INTEGER RW,        & % Length of the message
                   INTEGER W)           % Status
```

Parameters:

Error code
    The error code parameter is the error code for which to receive the error message.

Flag word
    See chapter 2.2.2 for explanation of the flag word.

Error message
    The error message parameter will contain the corresponding message upon return.

    In the FORTRAN call the length parameter should contain the length of the string to receive the message as invalue and is updated with the actual length of the message as outvalue.

2.2.9 UEERDMSG/UEIDMSG - Format and display message

This routine formats an error message and displays it on the error device (see section 2.2.12). The message may contain dynamic fields which are filled in before the message is displayed. Optionally the date and time may be displayed in front of the message. This will also result in an extra CRLF in front of the message.

Format:

```
        ROUTINE VOID,                  & % No invalue
                VOID(                  & % No outvalue
                INTEGER,               & % Error code
                INTEGER,               & % Flag word
                BYTES,                 & % Optional dynamic text
                INTEGER4,              & % Optional dynamic number
                BOOLEAN) :             & % Display date flag
                UEERDMSG                 % Display message

        SUBROUTINE UEIDMSG(            & % Display message
                   INTEGER,            & % Error code
                   INTEGER,            & % Flag word
                   HOLLERITH,          & % Optional dynamic text
                   INTEGER,            & % Length of the text
                   INTEGER*4,          & % Optional dynamic number
                   LOGICAL,            & % Display date flag
                   INTEGER W)            % Status
```

Parameters:

Error code
    The error code parameter is the error code for which the corresponding message is to be displayed.

Flag word
    See chapter 2.2.2 for explanation of the flag word. Bit 2 (UEERCBIT) is always reset by this routine.

Optional dynamic text and number
    See chapter 2.2.3 for explanation of the dynamic text and number parameters.

Display date flag
    The current date and time is displayed in front of the message if the display date flag parameter is set to TRUE.

The routine calls UEERFMSG to format the message and UEEROMSG to output it.

2.2.10 UEERFMSG/UEIFMSG - Format message

This routine formats a message as in the previous section but returns the message instead of displaying it.

Format:

```
        ROUTINE VOID,                 & % No invalue
                VOID(                 & % No outvalue
                INTEGER,              & % Error code
                INTEGER,              & % Flag word
                BYTES,                & % Optional dynamic text
                INTEGER4,             & % Optional dynamic number
                BOOLEAN,              & % Include date flag
                INTEGER,              & % Number of lead-in spaces
                BYTES W,              & % Formatted message
                INTEGER W) :          & % Length of the message
                UEERFMSG                % Format message

        SUBROUTINE UEIFMSG(           & % Format message
                   INTEGER,           & % Error code
                   INTEGER,           & % Flag word
                   HOLLERITH,         & % Optional dynamic text
                   INTEGER,           & % Length of the text
                   INTEGER*4,         & % Optional dynamic number
                   LOGICAL,           & % Include date flag
                   INTEGER,           & % Number of lead-in spaces
                   HOLLERITH W,       & % Formatted message
                   INTEGER RW,        & % Length of the message
                   INTEGER W)           % Status
```

Parameters:

Error code
    The error code parameter is the error code for which the corresponding message is to be returned in the formatted message parameter.

Flag word
    See chapter 2.2.2 for explanation of the flag word.

Optional dynamic text and number
    See chapter 2.2.3 for explanation of the dynamic text and number parameters.

Include date flag
    The current time and date is included in front of the message if the include date flag is set to TRUE.

Lead-in spaces
    Number of lead-in spaces to be included in front of the message if the date flag is set to FALSE.

Formatted message

The message corresponding to the error code parameter with the optional text and number parameters filled into any dynamic fields.

In the FORTRAN call the length parameter should contain the length of the string to receive the message as invalue and is updated with the actual length of the message as outvalue.

2.2.11 UEEROMSG/UEIOMSG - Display message

This routine outputs a previously formatted message on the error device (see chapter 2.2.12).

Format:

```
ROUTINE VOID,                   & % No invalue
        VOID(                   & % No outvalue
        BYTES) :                & % The message to output
        UEEROMSG                  % Output message

SUBROUTINE UEIOMSG(             & % Output message
           HOLLERITH,           & % The message to output
           INTEGER,             & % The length of the message
           INTEGER W)             % Status
```

Parameter:

Message to output
: This message is written on the standard output device or on a device set by a previous UEERSDV call.

2.2.12 UEERSDV/UEIESDV - Set output device

This routine sets the device number to output the messages to using UEERDMSG or UEEROMSG.

Format:

```
        ROUTINE VOID,                   & % No invalue
                VOID(                   & % No outvalue
                INTEGER) :              & % Device number
                UEERSDV                   % Set device

        SUBROUTINE UEIESDV(             & % Set device
                   INTEGER)               % Device number
```

Possible device numbers:

```
        -2:      Output to error device always
        -1:      Output to own terminal if background
                 Output to error device if RT program
        0:       No output
        >0:      Output to this device (must be reserved/opened!)
```

The device number is initially -1 (error device for RT and own terminal for background).

The negative numbers are NOT SINTRAN device numbers, but are used as internal signals to UEEROMSG.

Note that output to the error device from ND 500 background programs requires SINTRAN J or later if the error device is device number 1.

2.3 Routines operating on the terminal datafield

UE-LIBRARY contains several routines to operate on termination codes and on UE information stored in the terminal datafield. Below only "public" calls are documented (i.e. calls which may be used by other programs than UE itself).

2.3.1 UEFECCODE/UEIFECC - Fetch completion code

This routine returns the completion code set by a previous UESECCODE call issued either by the current or by a previous program. This call may be used to test if the program terminated normally. See chapter 2.3.3 also.

Format:

```
        ROUTINE VOID,                 & % No invalue
                INTEGER(              & % Status
                INTEGER W,            & % SSI of last system
                INTEGER W) :          & % Termination reason
                UEFECCODE               % Fetch completion code

        SUBROUTINE UEIFECC(           & % Fetch completion code
                   INTEGER W,         & % SSI of last system
                   INTEGER W,         & % Termination reason
                   INTEGER W)           % Status
```

Parameters:

SSI of last system
    The SSI parameter contains the SSI code of the program that set the completion code (e.g. 110 octal for NOTIS-WP).

Termination reason
    The termination reason contains 0 if normal termination or the error code if error termination (e.g. 56 octal - "NO SUCH FILE NAME") of the program.

2.3.2 UESECCODE/UEISECC - Set completion code

This routine sets the completion code for the program. This code is tested by e.g. JEC to see if a job proceeds normally.

ALL PROGRAMS SHOULD SET THE COMPLETION CODE BEFORE TERMINATING. EVEN IF THE OPERATION WENT OK!

Format:

```
        ROUTINE VOID,                 & % No outvalue
                VOID(                 & % Status
                INTEGER,              & % SSI of current system
                INTEGER) :            & % Termination reason
                UESECCODE               % Set completion code

        SUBROUTINE UEISECC(           & % Set completion code
                   INTEGER,           & % SSI of current system
                   INTEGER,           & % Termination reason
                   INTEGER W)           % Status
```

Parameters:

SSI of current system
    The SSI parameter shall contain the SSI code of the program that set the completion code (e.g. 110 octal for NOTIS-WP)

Termination reason
    The termination reason shall contain 0 if normal termination or the error code causing the termination if error termination (e.g. 56 octal - No such file name).

Compilers and loaders should set status different from zero if the compilation or loading contained errors even if they were not fatal. These codes may then be checked by JEC.

The following error codes are specially defined for compilers / loaders:

```
        7720B - Warning during compilation / loading
        7760B - File system error during compilation / loading
        7770B - Error during compilation / loading
        7777B - Fatal error during compilation / loading
                (compiler / loader / hardware error)
```

The SSI code should be set to the code defined for the compiler / loader.

2.3.3 <u>UEGERR/UEIGERR - Get error info</u>

Check if the previous program terminated abnormally (e.g. by a user break) or if the operation of a MON 317 call was successful. See also chapter 2.3.1.

Format:

```
        ROUTINE VOID,                  & % No invalue
                INTEGER(               & % Status
                INTEGER2 ARRAY W) :    & % Information
                UEGERR                   % Get error info

        SUBROUTINE UEIGERR(            & % Get error info
                   INTEGER*2 ARRAY W,  & %Information
                   INTEGER W)            % Status
```

Parameters:

Information
An array containing SINTRAN error information. The contents are described below. The length of the array must be 10 16 bits words.

The contents of the array is as follows:

Word #:	Contents:
0	Error code (for type 1, 2, and 3 below)
1	Type of "error": 1 - Terminated using MON 65 2 - SINTRAN III runtime error (e.g. Protect violation, Privileged instr.) 3 - File system error (Error in MON 317) 4 - @STOP-TERMINAL 5 - User break 6 - Batch processing timeout 7 - @LOGOUT
2	P register when the program terminated
2-9	If type is 1, the words 2-9 contain the register block: P, X, T, A, D, L, STS (bit 0-7), B register in this order.

After the first call, word 0 and 2 are set to 0!

Error type 6 and 7 are trapped by the UE logout handling only. Parameters 2-9 are not to be trusted due to SINTRAN errors.

If this call is to be used to test the status of a MN317 call, then the call must also be made prior to the MN317 call to reset the error code!

2.3.4 UEEDMSG/UEIEDMS - Enable/disable SINTRAN messages

This routine enables and disables SINTRAN messages to the terminal, i.e. turns on and off output of error messages directly from SINTRAN. This is useful if a program uses MN317 to execute SINTRAN commands and doesn't want SINTRAN to display any error messages on the terminal if the command failes. The program may then use UEGERR to check how the operation went and use UEERMSG to translate any error code into an error message to display to the user.

Format:

```
ROUTINE VOID,               & % No invalue
        INTEGER(            & % Status
        BOOLEAN) :          & % Enable/disable
        UEEDMSG               % Enable/disable SINTRAN msg

SUBROUTINE UEIEDMS(         & % Enable/disable SINTRAN msg
           LOGICAL,         & % Enable/disable
           INTEGER W)         % Status
```

Parameter:

Enable/disable
    If the flag is set to TRUE, SINTRAN may write error messages to the terminal. If set to FALSE, output of error messages is inhibited.

2.3.5 UEGFLAGS/UEIGFLA - Get UE flags

This routine returns some UE information which is stored in the terminal datafield.

Format:

```
        ROUTINE VOID,                & % No invalue
                INTEGER(             & % Status
                INTEGER,             & % Flag number
                BOOLEAN W) :         & % Flag value
                UEGFLAGS               % Get UE flags

        SUBROUTINE UEIGFLA(          & % Get UE flags
                   INTEGER,          & % Flag number
                   LOGICAL W,        & % Flag value
                   INTEGER W)          % Status
```

Parameters:

Flag number
    The flag to return the status of:

```
        UEFLUELOGIN  (1): Logged on to UE
        UEFLMUEMENU  (2): Using menus in UE
        UEFLSINOK    (3): SINTRAN commands allowed
        UEFLMAIL     (4): User has mail (NOTIS-ID)
```

The number of the flag is written in parentheses, while the symbolic name present in the UELIB-C:IMPT file is written in front of it.

Flag value
    The status of this flag.

Note that flag 4 is updated by some applications only and NOT by the mail server. Testing this flag in a loop is therefore a waste of CPU time. The UE-PLIB call UEGUP with function UEMAIL should be used to fetch the actual number of nonread messages if this is required by the application.

If using SINTRAN versions older than the J version or if not logged on to UE, the flags 1, 2 and 4 will be set to FALSE and flag 3 will be set to TRUE.

BE ESPECIALLY AWARE OF FLAG 3 - SINTRAN COMMANDS ALLOWED. NO APPLICATIONS SHOULD ALLOW THE USER TO EXECUTE DIRECT SINTRAN COMMANDS IF THIS FLAG IS RESET! IGNORING THIS FLAG RESULTS IN A SERIOUS SECURITY FLAW IN A SYSTEM.

2.4 Miscellaneous SINTRAN utility functions

This section contains a set of SINTRAN utility routines which were needed, but not previously available when UE was developed, but that are not UE functions.

2.4.1 MN317/IMN317 - Execute SINTRAN command

This routine use monitor call 317 (UECOM) instead of 70 (COMND) if present (i.e. if SINTRAN version I or later). MON 317 may be used for all SINTRAN commands (except CC) from the ND 500, while MON 70 is restricted to a few monitor calls only. MON 317 will return to the calling program also in cases of errors (MON 70 will terminate on errors such as "NO SUCH FILE NAME").

Format:

```
        ROUTINE VOID,                & % No invalue
                VOID(                & % No outvalue
                BYTES) :             & % SINTRAN command
                MN317                  % Execute SINTRAN command

        SUBROUTINE IMN317(           & % Execute SINTRAN command
                   HOLLERITH,        & % SINTRAN command
                   INTEGER)            % Length of the command
```

Parameter:

SINTRAN command
    The SINTRAN command to be executed. No apostrophe is needed as string terminator.

The routine will always return OK, even in cases of command failure. To test the operation and handle error reporting properly, the following sequence is recommended:

```
        % Turn off SINTRAN error messages to the terminal
        UEEDMSG(false)
        % Reset the error flags
        UEGERR(info)
        % Execute the command
        MN317(command)
        % Turn on SINTRAN error messages again
        UEEDMSG(true)
        % Test for error and display possible error messages
        UEGERR(info)
        if info(0) >< 0 then
            UEERMSG(info(0),6,message,length)
            display(message(0:length-1))
        endif
```

If a sequence of MN317 calls are executed before testing the status, the status from the last unsuccessful call is reported.

2.4.2 UEWRBIX/UEIWRBI - Write back modified index blocks

This routine forces the file system to flush back modified index pages for a file to the disk.

The index blocks (pointers to the pages of an indexed file) of an open file are kept in memory (if not swapped out due to normal swapping) until the file is closed. In cases of files being open when the computer stops (often the cases for databases and log files), pointers to possible new pages for the file after the last open file call may then be lost. Calling this routine prevents that.

To be completely on the safe side, SINTRAN J revision 4000 or later should be used. This version will also flush back the object entry when the file pointer is updated (e.g. when the file goes from no pages to having pages i.e. becomes an indexed file, and when the file goes from indexed to subindexed). For earlier revisions all contents of a file might be lost if the computer stops with the file open.

Format:

```
        ROUTINE VOID,                 & % No invalue
                VOID(                 & % No outvalue
                INTEGER) :            & % Open file number
                UEWRBIX                 % Write back index blocks

        SUBROUTINE UEIWRBI(           & % Write back index blocks
                   INTEGER,           & % Open file number
                   INTEGER W)           % Status
```

Parameter:

Open file number
    The open file for which to flush the index blocks back to disk.

2.5 Number formatting routines

These routines are used by UEERFMSG and thus by UEERDMSG in order to include numbers and dates in the messages. The UE_OB routine is also used by UEERMSG to include the number tag. These routines are exported in order to be useable also directly from the applications.

Only PLANC interface is available.

2.5.1 UE OB - Convert to octal

Convert a number to octal digits.

Format:

```
        ROUTINE VOID,                 & % No invalue
                INTEGER(              & % First free pos. in the string
                INTEGER4,             & % Number
                BYTES RW,             & % The string
                INTEGER) :            & % Minimum field width
                UE_OB                   % Convert to octal
```

Parameters:

First free position in the string
    The number is stored from the MININDEX of the string. The outvalue is the index to the first byte in the string following the number.

Number
    The number is regarded as a positive number even if the most significant bit is set.

The string
    The number is stored from the start of the string. If the string is too small to contain the complete number, the most significant digits are lost. The rest of the string is spacefilled.

Minimum field width
    This parameter specifies the minimum number of positions the calling routine wants the number to fill (e.g. 6 for the UEERMSG routine). Lead-in zeroes are filled in if the number fits into fewer digits than this width.

2.5.2 UE_DB   - Convert to decimal

Convert a number to decimal digits.

Format:

```
        ROUTINE VOID,                 & % No invalue
                INTEGER(              & % First free pos. in the string
                INTEGER4,             & % Number
                BYTES RW,             & % The string
                INTEGER) :            & % Minimum field width
                UE_DB                   % Convert to decimal
```

Parameters:

First free position in the string
    The number is stored from the MININDEX of the string. The outvalue is the index to the first byte in the string following the number.

Number
    The number is regarded as a signed number.

The string
    The number is stored from the start of the string. If the string is too small to contain the complete number, the most significant digits are lost. The rest of the string is spacefilled.

Minimum field width
    This parameter specifies the minimum number of positions the calling routine wants the number to fill (e.g. 4 for the year when called from UE_DATEB). Lead-in zeroes are filled in if the number fits into fewer digits than this width.

2.5.3 UE_HB - Convert to hexadecimal

Convert a number to hexadecimal digits.

Format:

```
        ROUTINE VOID,                & % No invalue
                INTEGER(             & % First free pos. in the string
                INTEGER4,            & % Number
                BYTES RW,            & % The string
                INTEGER) :           & % Minimum field width
                UE_HB                  % Convert to hexadecimal
```

Parameters:

First free position in the string
    The number is stored from the MININDEX of the string. The outvalue is the index to the first byte in the string following the number.

Number
    The number is regarded as a positive number even if the most significant bit is set.

The string
    The number is stored from the start of the string. If the string is too small to contain the complete number, the most significant digits are lost. The rest of the string is spacefilled.

Minimum field width
    This parameter specifies the minimum number of positions the calling routine wants the number to fill. Lead-in zeroes are filled in if the number fits into fewer digits than this width.

2.5.4 UE_DATEB - Fill in date

This routine fills in the current date and time in ISO format ("YYYY-MM-DD HH:MM:SS "). If a part of the string, e.g. the seconds, is unwanted, the calling program must strip off this part itself after return from the call.

Format:

```
        ROUTINE VOID,                 & % No invalue
                INTEGER(              & % First free pos. in the string
                INTEGER ARRAY,        & % MON CLOCK array
                BYTES RW) :           & % The string
                UE_DATEB                % Convert to standard date
```

Parameters:

First free position in the string
    The first free position in the string contains the index of the first byte in the string following the date, i.e. the MININDEX of the string + 20.

MON CLOCK array
    The clock array is the same as the array returned from MON 113 (CLOCK) (see SINTRAN III Reference Manual)

String
    The string to contain the time and date. One space is filled in after the date/time and the first free position set to point to the position following this space.

3 The error message file

This chapter describes the format of the source files for generation of the error message file, generation of the file, listing of the contents of the file and the use of dynamic fields.

3.1 Format of the source files

The start of the lines are checked for possible commands and comment lines. The SYSID command may be located anywhere in the source files and so may the comments. The SSI command must preceed any messages. There may be several SSI commands in one source file each starting a section of error messages.

The error codes are the SSI code shifted to bits 6 to 15 plus the error number after the ^.

SYSID is a separate range of "messages" containing names of owners of the SSI series.

Detected commands:

%	Comment line
%%	% as the first character (not comment)
^SSI=<no>	SSI code for following messages
^<no>	Error message to error <no> (0:63) follows on the subsequent lines 1) 2)
^SYSID=<no>	Name of the system with SSI <no> follows on the subsequent lines 1)
^^	^ as the first character (not SSI, SYSID or message start)
Other	Add to the preceding message (CR LF between lines is included in the message!)

1) If there is more text on the line containing ^<no> or ^SYSID=<no>, this text is taken as the start of the message. Lead-in spaces are skipped and % may be used to indicate that the text is a comment only.
2) <no> may be greater than 63, thus continuing with the next SSI series.

All numbers are regarded as decimal if not terminated by a B (octal number) or H (hexadecimal number). Hexadecimal numbers must start with a digit.

The maximum message length is 255 bytes.

The source files may be WP files in both 7, 16 and S format.

See chapter 3.5 for an example of a source file.

3.2 Dynamic fields

The messages may contain dynamic fields which are either filled in by the application itself or by the UE-LIBRARY through the use of either the UEERFMSG or the UEERDMSG call.

For the time being two classes of fields are defined, text and number fields. The fields are defined by including @<id> in the message where <id> may be either A, X, O, I, H or @. The latter means that the character @ should be present in the message. The characters must be in upper case!

The meanings:

```
@A - Text
@X - Text (used by WP for special purposes)
@I - Number in decimal format
@O - Number in octal format
@H - Number in hexadecimal format
@@ - The character @
```

Only application messages contain these fields today, but just in case all programs using UEERMSG should detect the fields and output only one @ if @@ is found and e.g. ?? if other fields are found and the program doesn't now what to include in the message.

The UEERFMSG call may be used to get a message with the dynamic fields expanded, whereas the UEERMSG call will return the message with the fields as @O etc.

3.3 Making the error message file

A special purpose program, UEER-CONVERT, has been made to make the error message files. This program takes two to n parameters, the first being the name of the error message file and the rest being message source files to be appended to the error message file. A blank message source file name terminates the job.

Note that messages are always appended to the previous contents of the error message file, but no space is released. Including the same message file n times thus implies that the messages are stored n-1 times without references (but taking space in the file) and one time with references (the latest) and that deleted messages will still be present. Complete message files should therefore always be made using a new file and a backup should be taken before including any additions. Then the backup may be used as the start file if changes to the messages or additions to them must be made.

Generating the error message file:

```
        @DELETE-FILE <Error message file name>
        @UEER-CONVERT-C
        "<Error message file name>"
        <Source file 1>
        <Source file 2>
        .
        .
        .
        <Source file n>
        <Blank line>
```

The standard error error message file name is UE-ERMSG-xx-Cyy

```
where xx is the standard language code: EN
                                        NO
                                        SW
                                        ...
```

yy is the revision number of the UE-ERRORS product containing the messages (ex 00).

WHEN CALLING UEERINIT, THE YY PART OF THE NAME SHALL BE OMITTED.

3.4 Listing the contents of the error message file

The program UEER-LIST may be used to list one or a set of messages from the error message file.

Listing the messages:

```
@UEER-LIST-C
<Error message file name>
<List file name>
<From error code>
<To error code>
```

Parameters:

Error message list file
    The file to list messages from. Default error message file name is UE-ERMSG-EN-C.

List file name
    Default file type for the list file is :LIST. Default list file is the terminal.

    @ may be given as list file. If so, the program loops asking for just one message number for which it displays the corresponding message on the terminal. The loop is terminated by giving CR as response to the "Error code" prompt. See below for the format of this request.

From and to error code
    The range of messages to display/print.

    All numbers are taken octal numbers if not terminated by D (decimal number) or H (hexadecimal number). If an S is appended to the number or the number is biased by 200000, SYSID names are displayed instead of error messages. Default numbers are 0 and 201777 (messages in the range 0 to 177777 and SYSID names in the range 0 to 1777, which covers all messages and SYSIDs).

The parameters may be specified on a single line using comma between the parameters.

Example (user input is underlined):

```
@UEER-LIST-C,,@
Error code: 0S
000000B: SINTRAN
Error code: 56
000056B: No such file name
Error code: 46d
000056B: No such file name
Error code: 2eh
000056B: No such file name
Error code: _
@
```

3.5 Examples for the error message handling

This section contains an example of a message source file and how the UEERMSG, UEERFMSG and UEERDMSG calls and the UEER-LIST program will handle the messages.

The source file:

```
        % This file contains the error messages for my system
        ^sysid=777b My system
        % The SSI code is 777 octal
        ^SSI=777b
        % This message starts on the line after the number
        ^0 % Comment on the message number line
        My system
        % This message starts on the same line as the number
        ^1 This is the first error
        % This message takes two lines
        ^2
        The second error is long,
        so I'll use two lines for it.
        % This message contains dynamic fields
        ^3
        Error no @I in routine @A.
        % Message 63 (77 octal) is normally the fatal error message
        ^63
        Fatal error in my system
        % Automatic continuation in the next SSI series
        ^70 This is message 6 in SSI 1000b
        % SYSID with the name on the following line
        ^sysid=1000b
        My system
        % Those where all the errors
```

Generating the error message file:

```
        @DELETE-FILE UEER-EN-C:ERR
        @UEER-CONVERT-C
        "UE-ERMSG-EN-COO"
        UEER-MYSYS-A
                                           ←— Blank line
        @CC THAT'S IT
```

Some calls to get/display the messages ("" surrounding the messages are included here to show what is returned).

UEERMSG calls:

```
UEERMSG(77701B,0,STRING,LENGTH)
  -> "This is the first error"
UEERMSG(77703B,0,STRING,LENGTH)
  -> "Error no @I in routine @A."
UEERMSG(100006B,2,STRING,LENGTH)
  -> "100006B: This is message 6 in SSI 1000b)
UEERMSG(777B,1,STRING,LENGTH)
  -> "My system"
```

UEERFMSG calls:

```
UEERFMSG(77701B,0,'',0,TRUE,0,STRING,LENGTH)
  -> "1986-05-11 16:05:30 This is the first error"
UEERFMSG(77703B,0,'ABC',5,FALSE,0,STRING,LENGTH)
  -> "Error no 5 in routine ABC."
UEERFMSG(100006B,2,'',0,FALSE,0,STRING,LENGTH)
  -> "100006B: This is message 6 in SSI 1000b)
UEERFMSG(777B,1,'',0,FALSE,3,STRING,LENGTH)
  -> "   My system"
```

UEERDMSG calls (normal use):

```
UEERDMSG(777B,1,'',0,TRUE)
UEERDMSG(77703B,2,'ABC',5,FALSE)
-> "
    1986-05-11 16:05:30 My system"
   "           077703B: Error no 5 in routine ABC."
```

Listing all the messages using UEER-LIST:

```
@UEER-LIST,,,77700,100077
-> 077700B: My system
   077701B: This is the first error
   077702B: The second error is long,
            so I'll use two lines for it.
   077703B: Error no @I in routine @A.
   077777B: Fatal error in my system
   100006B: This is message 6 in SSI 1000b
```

4 The UE-LIBRARY files

The UE-LIBRARY exists for ND 100 (1 and 2 bank) and for ND 500. A special ND 500 version of the library running under SINTRAN IV is available on request. Only the error message routines are yet available in this version.

The library:

1 bank 100:	UELIB-P1-Cxx:BRF	Code
	UELIB-D1-Cxx:BRF	Data
2 bank 100:	UELIB-P2-Cxx:BRF	Code
	UELIB-D2-Cxx:BRF	Data
500:	UELIB-P5-Cxx:NRF	Code
	UELIB-D5-Cxx:NRF	Data
Imports:	UELIB-Cxx:IMPT	

The utilities:

```
UEER-CONVERT-Cxx:PROG
UEER-LIST-Cxx:PROG
```

where xx is the revision level of the files, e.g. 00.

The :PROG files execute on ND 100. The :IMPT file is for PLANC only.

5 Loading - required products

UELIB-Dx should be loaded on one segment only when making multisegment 100 programs!

Any standard ND runtime library may be used (e.g. PLANC, FORTRAN, COBOL or Pascal).

IF LOADING UE-PLIB, UELIB SHOULD HAVE THE SAME VERSION LETTER. ELSE INTERNAL INCONSISTENCIES MAY OCCUR AS PARTS OF UELIB IS INCLUDED IN UE-PLIB.

6 Examples of use

6.1 Fetching error messages

Below are some examples of how to fetch error messages using UE-LIBRARY. Note that some of the code is pseudocode!

Initialize the library and fetch and display messages without dynamic fields:

```
        % Include the routines used
        $include UELIB-C:IMPT
        ...
        % Declaration of data used in UEERINIT
        bytes : errbuffer(0:1777b)
        ...
        % Initialization of UE-LIBRARY
        UEERINIT('UE-ERMSG-EN-C',errbuffer)
or      UEERDINIT('EN')
        ...
        on routineerror do
            % Fetch the message for this error tagged with the errcode
            UEERMSG(errcode,6,string,length)
            display(string(0:length-1))
        endon
        program code...
```

Fetch and display messages with dynamic fields:

```
        on routineerror do
            % Fill in the name of the routine and the line number in
            % the dynamic fields of this message
            UEERFMSG(errcode,6,'ROUTINE-X',line_no,false,0, &
                     string,length)
            display(string(0:length-1))
        endon
        program code...
```

Display messages on the error device:

```
        % Always display on the error device
        UEERSDV(-2)
        ...
        on routineerror do
            % Output the date and the name of the program
            UEERDMSG(my_ssi,1,'',0,true)
            % Output the error code, no dynamic fields
            UEERDMSG(errcode,6,'',0,false)
        endon
        program code...
```

6.2 Executing SINTRAN commands and checking the status

Below is an example of how to execute a SINTRAN command, inhibit messages from being displayed by SINTRAN and then test the result. Before terminating the resulting status is set. Note that some of the code is pseudocode!

```
% Include the routines used
$include UELIB-C:IMPT
...
% Declaration of data
integer2 array : info(0:9)
bytes          : string(0:255)
integer        : length, status
boolean        : sincom_ok
bytes          : user_command(0:79)
...
0 =: status
% Initialization of UE-LIBRARY error part
UEERDINIT('EN')
% Disable output from SINTRAN
UEEDMSG(false)
% If accepting user commands:
% The user may give SINTRAN commands?
UEGFLAGS(ueflsinok,sincom_ok)
...
... Further processing requires entering of a diskett
% Reset error information
UEGERR(info)
% Execute the enter directory command
MN317('ENTER-DIRECTORY ND F--1 0')
% Test the result
UEGERR(info)
if info(0) >< 0 then
    info(0) =: status
    % Fetch the message for this error tagged with the errcode
    UEERMSG(info(0),6,string,length)
    display(string(0:length-1))
endif
...
... May not proceed if status >< 0
if status >< 0 then
    UESECCODE(my_ssi,status)
    mon0
endif
...
... Normal termination
UESECCODE(my_ssi,0)
```

************** SEND US YOUR COMMENTS!!! **************

Are you frustrated because of unclear information in this manual? Do you have trouble finding things? Why don't you join the Reader's Club and send us a note? You will receive a membership card — and an answer to your comments.

Please let us know if you
- find errors
- cannot understand information
- cannot find information
- find needless information

Do you think we could improve the manual by rearranging the contents? You could also tell us if you like the manual!

************ HELP YOURSELF BY HELPING US!! ************

Manual name: UE–Library Reference Manaul

Manual number: ND–20.004.1 EN

What problems do you have? (use extra pages if needed) ______________________

____

____

____

____

Do you have suggestions for improving this manual ? ______________________

____

____

____

____

____

Your name: ______________________ Date: ______________

Company: ______________________ Position: ______________

Address: ______________________________________

____

What are you using this manual for ? ______________________

____

NOTE!
This form is primarily for documentation errors. Software and system errors should be reported on Customer System Reports.

Send to:
Norsk Data A.S
Documentation Department
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

Norsk Data's answer will be found on reverse side

Answer from Norsk Data

Answered by ____________________ Date ________

Norsk Data A.S

Documentation Department
P.O. Box 25, Bogerud
0621 Oslo6, Norway