NORD-100
Sort/Merge System

NORSK DATA A.S

NORD-100
Sort/Merge System

NOTICE

PRINTING RECORD

Printing	Notes
02/81	Version 01

NORD-100 SORT/MERGE SYSTEM
Publ.No. ND-60.146.01

NORSK DATA A.S
P.O. Box 4, Lindeberg gård
Oslo 10, Norway

Manuals can be updated in two ways, new versions and revisions. New versions consist of a complete new manual which replaces the old manual. New versions 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 Bulletin and can be ordered as described below.

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

These forms, together with all types of inquiry and requests for documentation should be sent to the local ND office or (in Norway) to:

Documentation Department
Norsk Data A.S
P.O. Box 4, Lindeberg gård
Oslo 10

PREFACE

THE PRODUCT

This manual describes the use of the NORD-100 Sort/Merge System, Version 1 of October 1980, a general system for sorting data contained on mass storage. The system is available both as an interactive SINTRAN III subsystem, and as a subroutine package.

This sort/merge system is an extension of the the MSD SORT System and is to be considered a new version of that system, although the name has been changed to the NORD—100 Sort/Merge System.

THE MANUAL

This manual describes the command and parameter input for the interactive subsystem, and the calling structure for the subroutines. It contains all information needed for the use of both forms.

THE READER

The manual should be read by users who want to use the interactive subsystem, and programmers who want to call the subroutines from their own programs.

TABLE OF CONTENTS

+ \+ \+

1 INTRODUCTION

The NORD–100 SORT/MERGE System is a program package enabling the user to sort mass storage files (magnetic tapes included) containing records of fixed or variable length. It is also possible to merge separately sorted files together. The package is available in the two following forms:

- SINTRAN III interactive subsystem
- Subroutines callable from a user program

The program is written in PLANC and occupies 9k of the main memory as subroutines. In addition, a sort buffer area is used. To minimize the sort time, the buffer area is set as large as possible. For the SINTRAN III subsystem, the buffer area is set to 59k, for the subroutines, the user must specify his own user area.

The SORT/MERGE System uses a scratch file. The size of the scratch file depends on input on the file, and will be either the same size or twice the size of the file to be sorted (depending on the buffer size). The user may specify his own scratch file or use the default scratch file.

In this version it is possible to sort on alphanumeric keys, numeric keys with different forms of sign representation, on binary integer keys, and on BCD keys. Ascending and descending sort sequence may be specified. It is also possible to build up an alternative sort sequence.

2 CAPACITY OF NORD–100 SORT/MERGE SYSTEM

The file to be sorted is divided into partitions, which are sorted independently of each other. The sorted partitions are stored temporarily on the scratch file. When all partitions are sorted, they are merged and written into the output file. If the file is small, the entire file will be sorted as one partition.

If not all sorted partitions can be merged in one pass (due to lack of available memory buffer area), they will be merged into greater partitions and stored temporarily back on the scratch file. The process will be repeated until the number of partitions is less than the maximum number that can be merged in one pass.

The maximum input file size the SORT/MERGE System is able to sort is approximately:

((30.000 * A) – 60.000.000) bytes

where A is the buffer area size in bytes.

This gives as the maximum size of the input file:

Buffer area size in k words	Maximum input file size in mega-bytes *
2	60 (30)
4	180 (90)
8	420 (210)
16	900 (450)
32	1860 (930)
58	3400 (1700)

TABLE 2.1

*If the record length is an odd number of bytes, the maximum size is half (See Section 4.2)

3 USING THE SYSTEM

3.1 *DESCRIPTION OF PARAMETERS FOR THE SUBSYSTEM*

The SORT/MERGE System is implemented as a subsystem of SINTRAN III, and may be called from the terminal as follows:

@ SORT—MERGE

and the following commands will be available:

HELP
 Lists the available commands

EXIT
 Exits to the operating system

SCRATCH-FILE <file-name >

<file-name >	=	A file name or, if previously opened for random read/ write, an octal file number. Default file type is DATA.

This command may be omitted and the scratch file 100 will be used.

RECORD-DESCRIPTION <min-reclength [:max-length]> <no. of keys> <rec-mode>

<min-reclength [:max-reclength]>	=	The minimum and maximum length of the records on the file. If recording mode is 'FIXED' (see below), the <:max-reclength> is to be omitted. If <:max-reclength> is omitted, and <rec-mode> is specified different from 'FIXED', the value given for <min-reclength> is assumed to be the maximum value and minimum is set to zero.
<no.of-keys>	=	The number of different key fields in the record
<rec-mode>	=	Describes the way the records on the file are separated. It may be specified as:

FIXED	All the records are of equal length
TEXT	The records are separated by 'CR' and 'LF'.
VARYING	Each record starts with two bytes containing the length of the record as a binary integer number.

KEY–DESCRIPTION <key-pos> <key-length> <sequence> <key-type>
[<key-pos> <key-length> <sequence> <key-type> ....]

<key-pos>	= The position in the record of the first byte in the key. If the key is located at the start of the record this parameter should be equal to 1.
<key-length>	= The length (in bytes) of the key.
<sequence>	= The sequence the key is to be sorted in. This may be:
ASCENDING	The records will be sorted in ascending sequence of key values.
DESCENDING	Descending sequence
<key-type>	= The type of key to be sorted. It may be specified as:
ASCII	The key will be sorted according to the ASCII character value.
ALTERNATIVE–ASCII	The key will be sorted according to the alternative collating sequence. (See the ALTERNATIVE–COLLATING–SEQUENCE command)
NUMERIC–UNSIGNED	The key must be integers. There is no sign permitted, and any characters other than decimal digitsmay give 'garbage'.
NUMERIC–LEADING SEPARATE	They key must be integers. If the first byte in the key is the sign '–' (minus), the key is taken as a negative number.
NUMERIC–TRAILING SEPARATE	The key must be integers. The last byte of the key is reserved for sign representation. If this byte contains the character '–', the key will be taken as a negative number.
NUMERIC–LEADING EMBEDDED	The key must be integers. The fist byte of the key contains either a '–' or a '+' in a multipunch representation. If the character present in the first byte is a member of the set -0:-9 (ASCII 112B:122B), the key will be taken as a negative number.
NUMERIC–TRAILING EMBEDDED	This representation is equal to the LEADING–EMBEDDED, except that the last byte position of the key is used for the sign representation.

INTEGER	The key must be a <key-length > bytes representation of a binary integer. If the first bit in the first byte in the key is set, the number will be considered negative.
BCD	The key must be a string of 4-bit BCD bytes. The <key-length > parameter always specifies bytes, so the key will be of <key-length >*2 BCD positions. The last BCD position contains sign specification, where the values 13B and 15B represent negative sign.

Note: The TRAILING—EMBEDDED representation is the default sign representation in ANS COBOL systems. The embedded sign representation of former NORD—COBOL systems (older than the 1980 version) is not completely compatible with this standard and may, in certain cases, cause unpredictable results.

BLOCK—FACTOR—INPUT <number> <unit>

<number>	= Number of <units> on each block on the input file.
<unit>	= May be specified as:
CHARACTERS	<number> is number of bytes in each block
RECORDS	For FIXED recording mode, each block must contain an integer number of records. <number>* <record-length> is number of bytes in each block.

If <unit> is omitted, it is assumed to be characters.

This gives the block-factor for the input file (magnetic-tape files only).

BLOCK—FACTOR—OUTPUT <number> <unit>

<number>	= Number of <units> on each block of the input file.
<unit>	= As for BLOCK—FACTOR—INPUT

This gives the block-factor for the output file (magnetic-tapes files only)

ALTERNATIVE-COLLATING-SEQUENCE <file-name>

<file-name>	= Name or octal number of file where the ascending sequence of the alternative sort sequence is specified.

The format of the contents of the file is:

```
<character>  <space>        [<cr> <lf> ]
             <comma> . . .
```

The characters not specified, will be appended to the ascending sequence, according to the ASCII value. Default file type is DATA.

Example of a file where the alternative collating sequence is specified. The ascending sequence is to be:

- space
- characters A—Z
- figures 0—9
- the remaining part of the ASCII character set.

```
 ,A,B,C,D,E,F,G,H,I,J,K,L,M,N
O,P,Q,R,S,T,U,V,W,X,Y,Z,0,1,2
3,4,5,6,7,8,9
```

SORT <input-file> <output-file>

<input-file>	= Name or octal number of input file. If number, the file must have been opened for random-read.
<output-file>	= Name or octal number of output file. If number, the file must have been opened for random-write.

Input file and output file may be the same file. Default file type is DATA.

MERGE <no.of-files> <input-file> <input-file> [....] <output-file>

<no.of-files>	= The number of files (previously sorted) to be merged.
<input-file>.....	= As many names or octal numbers of input files as specified in <no-of-files>
<output-file>	= Name or octal number of output file.

The output file can not be the same as any of the input files.

The SORT and MERGE commands must be entered after all other commands. RECORD–DESCRIPTION and KEY–DESCRIPTION are required commands; the others are optional.

The SORT/MERGE System will ask for all missing parameters, so if you don't remember which parameters to give, just terminate each parameter by 'CR'.

The parameters may be separated by commas, or by one or several spaces.

3.2 *DESCRIPTION OF PARAMETERS FOR THE SUBROUTINES*

The SORT/MERGE System is implemented as two subroutines callable from user programs. They may be called as follows:

CALL SORT (input, output, scratch, minlen, maxlen, rmode, n-key, key, buff-size, buff, bl-inp, bl-outp, coll-file, status)

input	= Name of input file (FORTRAN character format) For the specification of FORTRAN character format see Appendix B. Default file type is DATA.
output	= Name of output file (FORTRAN character format). If no output file 0, (integer) is specified. Default file type is DATA.
scratch	= Name of scratch file (FORTRAN character format). If default scratch file is to be used, this parameter is set to 0 (integer). Default file type is DATA.
minlen	= The minimum length of the record on the file. If the length is fixed, this parameter gives the length.
maxlen	= The maximum legth of the records on the file. If the length is fixed, this parameter is ignored.
rmode	= The way the records are separated. It may be specified as:
0	– FIXED
1	– TEXT (Delimited by 'CR' and 'LF')
2	– VARYING (Two bytes in front indicating length)
n-key	= Number of key fields in the records.
key	= Integer array containing as many <key-lists> as specified in <n-key>. A <key-list> consists of:<key-pos>, <key-length>, <sequence>, <key-type>
key-pos	– The position within the record of the first byte of the key. If the key is located at the start of the record, this parameter should be specified to 1.
key-length	– The length, in bytes, of the key.

sequence	— Can be be specifed as: 0 - ascending or 1 - descending
key-type	— Gives the key representation, and may be: 0 - ASCII 1 - ALTERNATIVE—ASCII 2 - NUMERIC—UNSIGNED 3 - NUMERIC—LEADING—SEPARATE 4 - NUMERIC—TRAILING—SEPARATE 5 - NUMERIC—LEADING—EMBEDDED 6 - NUMERIC—TRAILING—EMBEDDED 7 - INTEGER 8 - BCD
buff-size	= Sort buffer area size in words (integer). The buffer size must be greater than 1k words (1024).
buff	= The sort buffer area (integer array)
bl-inp	= Number of characters on each block for input file (only for magnetic-tape files). If specified as 0, default block size is used.
bl-outp	= Number of characters on each block for output file. If 0, default block size is used.
coll-file	= Name of the file where the ascending sequence for the alternative sort sequence is specified (FORTRAN character format). If no alternative sort sequence is used, this parameter should be specified as 0 (integer). For the specification of the contents of the file, see ALTERNATIVE-COLLATING-SEQUENCE used as SINTRAN III Subsystem. Default file type is DATA.
Status	= Status returned from the SORT/MERGE System.

STATUS = 0: The sorting has finished and no error has occurred

0 <STATUS < 400B: I/O system error, and STATUS contains the SINTRAN III file system error code. Consult the SINTRAN III refrence manual.

STATUS = 402B :	NO SUCH COLLATING SEQUENÇE
STATUS = 404B :	SORT FILE TOO BIG FOR SPECIFIED BUFFER SIZE
STATUS = 406B :	TOO LONG TOTAL KEY
= 410B :	ERROR IN SPECIFYING ALTERNATIVE COLLATING SEQUENCE
STATUS = 412B :	NO SUCH RECORD TYPE
STATUS = 415B :	RECORD GREATER THAN SPECIFIED MAX SIZE
STATUS = 416B :	EOF MET WITHIN RECORD
STATUS = 417B :	MISMATCH OF RECORD LENGTH AND FILE SIZE
STATUS = 422B :	RECORD SMALLER THAN SPECIFIED MINIMUM SIZE
STATUS = 423B :	NO SUCH KEY TYPE

These messages are explained in Section 3.5 Error Messages.

To merge the records of two previously separately sorted files, the MERGE subroutine call may be used as follows:

CALL MERGE (n-files, input, output, scratch, minlen, maxlen, rmode, n-key, key, buff-size, buff, bl-inp, bl-outp, coll-file, status).

n-files	= Number of files to be marged
input	= The names of the input file in a FORTRAN character array. (See Appendix B).

The rest of the parameters are to be specified as for the SORT subroutine call.

An additional error code may be returned:

STATUS = 421B : TOO MANY INPUT FILES.

3.3 *EXAMPLES OF USING THE SYSTEM*

(User entries underlined).

EXAMPLE 1:

```
@SORT-MERGE
*RECORD-DESCRIPTION 80,2,FIXED;
*KEY-DESCRIPTION 1,12,ASCENDING,ASCII,25,5,DESCENDING,ASCII
*SORT INN-DATA, OUT-DATA
*EXIT
```

EXAMPLE 2:

```
@SORT-MERGE
*RECORD-DESCRIPTION
 REC-LENGTH[:REC-LENGTH] : 80
 NO.OF-KEYS : 2
 REC-TYPE : FIXED
*KEY-DESCRIPTION
 KEY-POS : 1
 KEY-LENGTH : 12
 SEQUENCE : ASCENDING
 TYPE : ASCII
 KEY-POS : 25
 KEY-LENGTH : 5
 SEQUENCE : DESCENDING, ASCII
*SORT
 INPUT-FILE : INN-DATA  OUT-DATA
*EXIT
```

EXAMPLE 3:

```
@SORT-MERGE
*RECORD-DESCRIPTION 80
 NO.OF-KEYS : 2 FIXED
*KEY-DESCRIPTION 1,12,ASCENDING,ASCII
 KEY-POS : 25,5,DESCENDING,ASCII
*SORT INN-DATA
 OUTPUT-FILE : OUT-DATA
*EXIT
```

All three of the above examples have the following meaning:

— Sort the file INN—DATA and place the results on file OUT—DATA

— The data on INN—DATA consists of records of fixed length 80 bytes (characters).

3.4 MESSAGES FROM THE SYSTEM

While the sorting is being done, and after it is finished, the system will print out some information, such as:

```
MERGE STARTED
<number > RECORDS SORTED
```

These messages are all self-explanatory.

3.5 ERROR MESSAGES

The following error messages may be printed by the Sort program.

SORT FILE TOO BIG FOR SPECIFIED BUFFER SIZE
: This message is printed if the input file is greater than it is possible to sort with the present buffer area (See Table 2.1).

TOO MANY KEYS
: Maximum 7 keys permitted, only for SINTRAN III subsystem.

TOO LONG TOTAL KEY
: Maximum length (sum of individual key lengths) of total key is 255 bytes. Also total key length can not be greater than the maximum key length.

ERROR IN DECIMAL NUMBER
: Input not decimal number.

ERROR IN OCTAL NUMBER
: Input not octal number

NO SUCH COLLATING SEQUENCE
: Specified sequence does not exist.

NO SUCH KEY TYPE
: Specified key type does not exist.

NO SUCH RECORD TYPE
: Specified record type (recording mode) does not exist.

NO VALUE GIVEN FOR PARAMETER
: A parameter with no default value has been left unspecified.

IMPOSSIBLE COMBINATION OF PARAMETER VALUES
: The combination of parameter values is illogical.

ILLEGAL VALUE FOR PARAMETER
A parameter has been specified as an illegal value.

TO MANY INPUT FILES
To many input files have been specified for MERGE (maximum 4)

MISMATCH OF RECORD LENGTH AND FILE SIZE
The size of the file does not fit an integer number of fixed records. (FIXED recording mode only).

RECORD GREATER THAN MAX SIZE
A record has been found greater than the specified maximum size.

RECORD SMALLER THAN THE MINIMUM SIZE
A record has been found smaller than the specified minimum size (VARYING recording mode only).

EOF FOUND WITHIN RECORD
The length field of a VARYING type record points beyond the end of the file.

ILLEGAL COMMAND SEQUENCE (MISSING INFO)
Either the SORT or the MERGE command has been entered before both RECORD–DESCRIPTION and KEY–DESCRITPION have been propely specified.

3.6 *HINTS AND RESTRICTIONS FOR THIS SORT SYSTEM*

– It is possible to sort on alphanumeric keys, and on numeric integer keys, binary integer keys and BCD fields.

– Restrictions on input file size (See Table 2.1).

– Maximum number of keys is 7 (only for SINTRAN III subsystem). Used as a subroutine there is no limit on number of keys.

– Total key length is maximum 255.

– For magnetic tape input, always specify block factor input. The number must be equal to the block factor on the tape.

– Using alternative sort sequence is as fast as normal sort sequence. Normal and alternative sequence can be used in the same run.

– Specify your own scratch file and use a continuous file. Using continuous file is faster than using indexed file.

4 METHOD USED

4.1 *SORTING*

A most-significant-digit-first radix sort algorithm is used to sort the partitions (MSD-radix).

The number of records sorted in each partition is determined as the integral number buffer size/record length. The sorting is performed from the most significant byte towards the least significant byte. Records with identical k first bytes in their keys are chained together. The sorting of their k + 1'th key position will generally split the chain into several sub-chains. When a chain contains a single record, its position can be determined and this record is not involved in any further processing. The sequence of sorted records is built up in an array and each record will be moved once (at most). The terminal sort condition is reached when:

$$\frac{n}{C^k} = 1, k <= k_{max}$$

where:

n	number of records in the partition
C	number of different characters used in the key alphabet
k	average number of key characters to be processed
k_{max}	total number of key characters in a record

This means that:

$$k = \ln n / \ln C$$

If we roughly assume the sorting time (exclusive I/O, which is proportional to the record length) to be proportional to the number of characters processed (all records in main memory) the algorithm is always better than normal radix sort where all key positions are processed (in reversed order) $(k = k_{max})$. When either the key-alphabet-set or the key-length are reduced, the improvements of MSD-radix are rather poor. However, in practical cases the improvements are significant. With a record length of 80 characters (all key characters randomly distributed), key length of 20, C=26 (all letters) and n=1000, the MSD-radix is 9 times faster. If the key is extended to cover all 80 characters, the difference will increase to about 36 times faster because it is independent of key length.

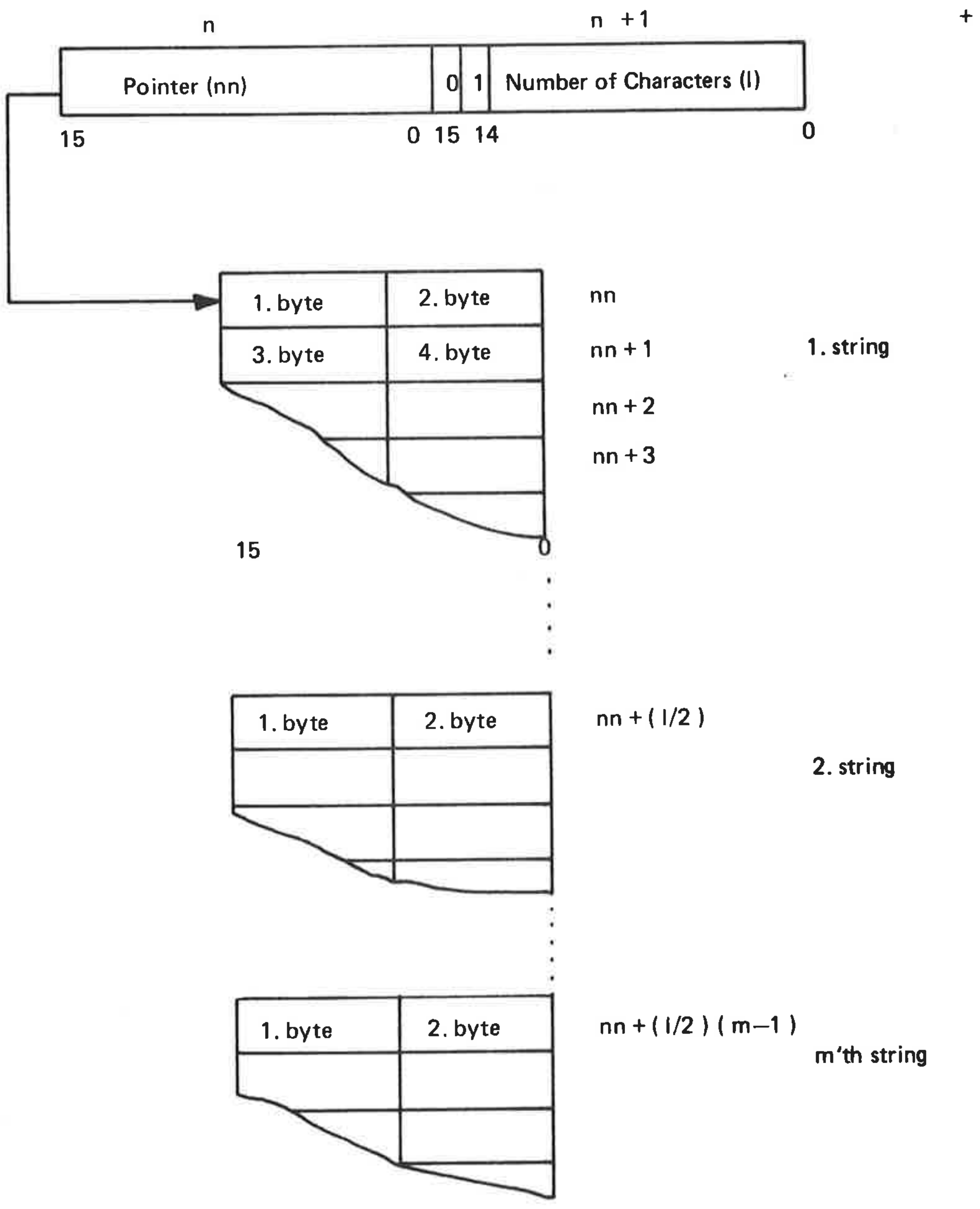
Graphic:	Octal Value:	Decimal Value:	ASCII Abbreviation:	Comments:
)	51	41	)	Closing parenthesis
*	52	42	*	Asterisk
+	53	43	+	Plus
,	54	44	,	Comma
—	55	45	—	Hyphen (Minus)
.	56	46	.	Period (Decimal)
/	57	47	/	Slant
0	60	48	0	Zero
1	61	49	1	One
2	62	50	2	Two
3	63	51	3	Three
4	64	52	4	Four
5	65	53	5	Five
6	66	54	6	Six
7	67	55	7	Seven
8	70	56	8	Eight
9	71	57	9	Nine
:	72	58	:	Colon
;	73	59	;	Semi-colon
<	74	60	<	Less than
=	75	61	=	Equals
>	76	62	>	Greater than
?	77	63	?	Question mark
@	100	64	@	Commercial at
A	101	65	A	Uppercase A
B	102	66	B	Uppercase B
C	103	67	C	Uppercase D
D	104	68	D	Uppercase E
E	105	69	E	Uppercase E
F	106	70	F	Uppercase F
G	107	71	G	Uppercase G
H	110	72	H	Uppercase H
I	111	73	I	Uppercase I
J	112	74	J	Uppercase J
K	113	75	K	Uppercase K
L	114	76	L	Uppercase L
M	115	77	M	Uppercase M
N	116	78	N	Uppercase N
O	117	79	O	Uppercase O
P	120	80	P	Uppercase P
Q	121	81	Q	Uppercase Q
R	122	82	R	Uppercase R
S	123	83	T	Uppercase S
T	124	84	T	Uppercase T
U	125	85	U	Uppercase U
V	126	86	V	Uppercase V

Graphic:	Octal Value:	Decimal Value	ASCII Abbreviation:	Comments:
W	127	87	W	Uppercase W
X	130	88	X	Uppercase X
Y	131	89	Y	Uppercase Y
Z	132	90	Z	Uppercase Z
[	133	91	[	Opening bracket
\	134	92	\	Reversing slant
]	135	93	]	Closing bracket
^ or ↑	136	94	↑	Circumflex, up-arrow
_ or ←	137	95	_, UND, BKR	Underscore, back arrow
`	140	96	`, GRA	Grave accent
a	141	97	a, LCA	Lowercase a
b	142	98	b, LCB	Lowercase b
c	143	99	c, LCC	Lowercase c
d	144	100	d, LCD	Lowercase d
e	145	101	e, LCE	Lowercase e
f	146	102	f, LCF	Lowercase f
g	147	103	g, LCG	Lowercase g
h	150	104	h, LCH	Lowercase h
i	151	105	i, LCI	Lowercase i
j	152	106	j, LCJ	Lowercase j
k	153	107	k, LCK	Lowercase k
l	154	108	l, LCL	Lowercase l
m	155	109	m, LCM	Lowercase m
n	156	110	n, LCN	Lowercase n
o	157	111	o, LCO	Lowercase o
p	160	112	p, LCP	Lowercase p
q	161	113	q, LCQ	Lowercase q
r	162	114	r, LCR	Lowercase r
s	163	115	s, LCS	Lowercase s
t	164	116	t, LCT	Lowercase t
u	165	117	u, LCU	Lowercase u
v	166	118	v, LCV	Lowercase v
w	167	119	w, LCW	Lowercase w
x	170	120	x, LCX	Lowercase x
y	171	121	y, LCY	Lowercase y
z	172	122	z, LCZ	Lowercase z
{	173	123	{, LBR	Opening (left) brace
|	174	124	|, VLN	Vertical line
}	175	125	}, RBR	Closing (right) brace
~	176	126	~, TIL	Tilde
	177	127	DEL	Delete, rubout

APPENDIX B

FORTRAN CHARACTER STRINGS

The data format of strings consists of a two-word object which contains a pointer to the memory location of the string and the number of characters in each element of the string array. Bit 15 of the second word indicates odd (right) first byte. If the character variable is undimensioned (contains only one string), the string starts at this location. If the string is a character array, each element occupies as many bytes as the length indicates. The strings themselves consist of the characters packed two by two into each word. The words are stored in consecutive order. The parity bit (bit7) is always set to 0.



NORSK DATA A.S
P.O. Box 4, Lindeberg gård
Oslo 10, Norway

COMMENT AND EVALUATION SHEET

Nord-100 Sort/Merge System ND-60.146.01

February 1981

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 preaddressed form and mail it. Please be specific wherever possible.

FROM ....

....

....