$112.50

# TSC 6800 Sort/Merge Package

AP68-10


TECHNICAL SYSTEMS CONSULTANTS

# TSC 6800 Sort/Merge Package

COPYRIGHT © 1978 BY
Technical Systems Consultants, Inc.
P.O. Box 2574
West Lafayette, Indiana 47906
All Rights Reserved

## 6) Sorting Random Access Files with SORT/MERGE

We have received a number of calls about sorting FLEX random access files with our Sort/Merge package. It's really not difficult to do as we'll explain here. To make things easiest, you should obtain version #2 of sort/merge if you don't already have it. To determine what version you have, run a quick test sort and see what version number is printed in the header sent to the screen. Version #2 has a few minor bugs fixed up and has one added feature: the ability to create a random file on output. Armed with that version you are ready to sort random files.

There are two things different about specifying how the input file is to be read. First you will probably be wanting to sort on some fixed record length rather than having an end-of-record character. For example, if you have records that are 252 characters in length (one record = one full data sector), you would probably want to specify that your input records were fixed length records of length 252. The second thing you need to inform sort/merge about is the fact that the input file is binary and not textual. This is done during the additional input parameter prompting from the SORT command. Sort/merge assumes textual input files unless you specifically tell it that the input is binary. We want to sort the input file as binary so that no space expansion goes on thus screwing up record lengths.

There are three things to tell sort/merge about the output file being created. First you must declare the output file as binary instead of textual so that no space compression goes on which would mess up record lengths in the output file. Second you need to declare the output file as a random access file. This is done in the additional prompting section of the SORT utility (only in version #2 or higher). If you do not specify the output file as random, it will be output as a sequential file. Third, you will most likely want to specify no end-of-record character for the output file. By default, sort/merge outputs a carriage return at the end of each output record. Assuming we are sorting fixed length records, this would add a character to each output record and thus mess up the record lengths. In the additional prompting section of SORT you are asked for an "EOR CHARACTER FOR OUTPUT RECORDS (DEFAULT=$0D)?". Responding with an 'N' tells sort/merge to not append any character on the end of the output records.

That's all there is to it. By making sure that you properly specify these five points, sorting random files should present no problem.

TABLE of CONTENTS



USE FOR FOLLOWING PROMPTS ON TERMINAL

# 1.0 Introduction to the TSC SORT/MERGE PACKAGE

The TSC SORT/MERGE PACKAGE is a very powerful and quite complex program. It was, however, designed with the operator in mind and is relatively simple to use. This manual was written with the non-computerist in mind and will lead you gradually into use of the sort/merge package. The user is advised to experiment with the use of the sort/merge on the sample data file supplied on the disk or with a non-critical file of his own. The best way to learn the operation of the software is from hands-on experience.

As the name implies, the sort/merge package has two major functions, sorting and merging. The most used function will be that of sorting and will be the first discussed in this manual. Much of the description of the sort, however, will also apply to the merge.

In order to use the TSC SORT/MERGE PACKAGE, you must have a disk system with the FLEX operating system. A minimum of 8K of user memory must be available starting at address 0000. The input to the sort/merge program is one or more disk files and output can be to terminal, to printer, or to a named disk file.

## 2.0 SORTING CONCEPTS

The TSC SORT/MERGE PACKAGE can be used to sort almost anything which is contained in a disk file. Generally it is used to sort some type of textual data such as a file of names and addresses. The "items" or pieces of information which are sorted into order are called "records". If, for example, you had a file containing a list of 10 peoples last names, each name would be called a record. If the list contained the last name followed by the first name, then one record would include the last name and the first name. The sort routine may be used to rearrange these records into some defined order. For example you could sort the list of last names into alphabetical order or perhaps into reverse alphabetical order (Z's at the top of the list). If sorting the file described above which had first and last names, it would be possible to sort according to the last name only so that the finished output would have last names in alphabetical order with the first names following along. Alternatively, we could sort according to first name, so that the first names would be in order with the last names following along. In short, it is possible to sort according to any portion of the input record. This portion which is used as a reference for sorting is called an "input key". Thus we could specify to the sort package that we wanted the last name as the key or that we wanted the first name. We could go a step further and specify more than one key. That is, we might sort first on the last name and then on the first name. This would mean that anytime there were several records with identical last names, they would end up being sorted together with the corresponding first names also put into order. Or we could specify the first name as the first input key and the last name as the second input key. This would sort the records into order by first name and where there were multiple records with identical first names, the corresponding last names would also then be put into order. If the file we wished to sort had more than just a first and last name (such as address, phone number, etc.), we could sort on several different keys. It is possible to specify as many as 20 input keys to the sort routine. Each key may be individually specified as ascending or descending (ie. alphabetical or reverse alphabetical), right or left justified (trailing or leading blanks omitted), and may be specified as any portion of the input record by specifying column numbers. For example, we might want to sort on columns 1 through 10 (inclusive) which we know contains a last name in each record. Or perhaps on columns 21 through 30 (inclusive) which is known to contain a phone number in each record.

In the simplest case, the output file is a rearranged version of the records of the input file. Note that it is not necessary to actually produce a disk file of the output. If desired, the output can be routed to the terminal or a printer. When a disk output file is produced, it is entirely independent of the input file. In other words, the input file is not altered in any way. The sort simply reads through the input file and produces a new file with the output. If desired, the output records (one "record" of information is sent to the output for each record which is input) do not have to be a carbon copy of the input. You may specify that only certain portions of the input record be sent to the output record. These "portions" are called output keys much like

case, as many as possible are merged into another single run or work file which may then be merged with the remaining runs in a second pass. In extreme cases (caused by large files and little memory available) several of these merge passes may be required.

It should be obvious that the sort/merge package requires a good deal of information or parameters specified in order to know just how to carry out the sort. The TSC SORT/MERGE PACKAGE actually consists of six disk files. One of those files, called "SRTMRG.SYS", is responsible for the actual sorting and merging processes. The other five are concerned only with supplying all the necessary parameters to the SRTMRG.SYS program. A specific area of memory is set aside to hold all the parameters. When the SRTMRG.SYS program is called, it expects to find all the necessary parameters in that area. Each of the five programs sets up the parameters in that area, loads SRTMRG.SYS, and starts execution. Each one has a different way of setting up the parameters as explained below.

The first is called "SORT" and is used to set up parameters for a sort operation. It does so by prompting the user for all the necessary information. When all the parameters are set, there is an option for saving them in a disk file so that the very same parameters may be used again (as explained later) without having to re-enter them.

The second is called "PSORT" which stands for Parameter file Sort and is also used for a sort operation. This command allows the user to specify a parameter file (which was produced by the SORT command) by name so that previously setup parameters may be easily recalled.

The third is called "CSORT" which stands for Command line Sort and is also used to setup parameters for a sort. Instead of being prompted for the parameters, the user must supply them in a somewhat condensed form on a single command line. This is quicker and often more convenient for the user who fully understands the operation and use of the sort/merge.

The fourth is called "MERGE" and is used to setup paramters for a merge-only operation. The program prompts the user for all the information much as in the SORT command and then proceeds with the merge. There is also a provision for saving the parameters in a disk file for later use.

The fifth and final command is "PMERGE" standing for Parameter file Merge and is also used to prepare for a merge-only operation. Much like PSORT, it allows the user to specify a parameter file which has been prepared by either SORT or MERGE.

This modularity gives the user a great variety of capabilities and allows convenient operation of the sort/merge package under almost all circumstances.

## 3.0 SORT TUTORIAL

The SORT command is probably the easiest method of using the sort/merge package, especially for the novice. It prompts the user for all the necessary information thus obviating the need to memorize what parameters must be supplied. Most of the information asked for will default to some preset default value if the operator simply hits a carriage return in response to a particular prompt. The prompts usually include what choices the operator has for a response. The choice which will be used as a default is generally designated by a following asterisk ('*'). For example, the first prompt you will receive will be

OUTPUT TO DISK (Y OR N*)?

The two choices you have as a response are 'Y' and 'N' for yes and no. You could, however, simply hit a carriage return and the SORT program will default to no disk file output since the 'N' response is flagged with an asterisk.

It should be noted that there are two types of responses that may be given to a prompt. If the response being typed by the user is to be only a single character (such as the 'Y' or 'N' above), when the character is hit the SORT program will accept it and immediately continue without requiring a carriage return. Some responses, however, require an entire line of information (such as a prompt for the output file name). In these cases, nothing is done by SORT until the entire line has been entered and a carriage return typed. When entering this type of response, there are three control characters which may be used as an added convenience. Control characters are entered by typing the 'CTRL' key at the same time as some letter key is hit. The possible control keys are CTRL H, CTRL X, and CTRL Z. The CTRL H causes a single backspace in the line being typed. If your keyboard has a backspace key, it will accomplish the same function. The CTRL X is a cancel function which will cancel the line presently being typed and issue a prompt ('??') for a replacement line. The CTRL Z is a restart function meaning that anytime a CTRL Z is hit, the SORT program will start over with the first prompt. The CTRL Z may be used either on a line response or on a single character response.

To help clarify the following description of use of the SORT command, references will be made to a sample data file named, "NAMES.TXT" as found on the disk on which the sort/merge package is distributed. It is listed here as a reference.

sorting is so short, there will be enough room in memory for the entire sort, so no work files will be produced. If desired, you may enter a one digit drive number anyway. You may also simply hit a carriage return which will cause the work drive to be set to the system drive which has been established in the disk operating system. If no system drive was setup, the first available drive would be used.

Next we see the prompt

FIXED OR VARIABLE LENGTH RECORDS (F OR V*)?

We are sorting variable length records (even though they are all the same length in this case) so type a 'V' or a carriage return. More information on input record type is given later.

The computer will then ask

EOR CHARACTER OR FIELD COUNT (DEFAULT IS EOR=$0D)?

where EOR stands for End Of Record. Our file has an EOR character which is a carriage return (a hex 0D) so we can either hit a carriage return allowing the default to be set or type a "$0D" to set the EOR to a carriage return.

Now we are asked

FIELD SEPARATOR CHARACTER?

We do not have "fields" (these will be described later), so simply type a carriage return which defaults to a null field separator character.

Next the computer asks

OUTPUT FROM KEY, INPUT, OR OTHER (K, I*, OR O)?

This prompt gives us the option of having the data in the output records come from the actual input records, from the sort key itself, or from some other source as will be described later. In most cases you will probably want the output to come from the input records. That is the case here since we simply want a rearranged version of the input records. Your response should therefore be an 'I' or use the default by typing a carriage return.

Now we get to the real meat of the sort parameters, the input keys. We are prompted with

ENTER INPUT KEYS. DEFAULTS TO "A(1)1-10".
?

As mentioned before, the input keys are specified by the starting and ending column numbers of the portion of the input record that is to be used as a sorting reference. Therefore, a part of EVERY input key specification must be of the form "sss-eee", where "sss" represents the starting column and "eee" represents the ending column inclusive. Now in our sample we want to sort on the last names. We can see that they

```
=== TSC SORT/MERGE V1.3 ===

SORT RUN 01 - 10 RECORDS
ABBOTT    JIM       4572091
CASWELL   AMY       9258411
FREEMAN   LINDA     4229105
LYON      WILLIAM   8538227
PERRY     ARNOLD    4226008
SMITH     JERRY     4226002
SMITH     HAROLD    4227264
SOUTH     MARGARET  4221267
TYLER     HENRY     4575573
WILSON    JIM       4578339

10 RECORDS SORTED
```

As you see, the records have been put into ascending alphabetical order by last name. Now if we had wanted, we could have specified that the sorted output be sent to disk. SORT would have prompted for a file name, and would have written the sorted data out to that file.

## 3.2 SORTING WITH INPUT KEYS

If you will examine just what prompts you were required to fully answer in the preceding example and which ones you were able to use the default value on, you will notice that all responses could be defaulted. Often most responses can be defaulted except for the input keys. Generally your specific sort needs will require you to enter some key other than "1-10". At this point you might try a couple of more sorts on the file NAMES.TXT. Instead of "1-10" for an input key, try sorting by first name only. This could be done by responding in the same manner as the example above to all the prompts except the one for entering input keys. To this you should respond with "11-20" since the first names all start in column 11 and can extend up to column 20. Then try sorting by phone number in the same manner. This would require an input key of "21-27".

If you will examine the output of our first example above, you will see that there are two Mr. Smiths. Their last names are sorted into order as requested, but their first names (which were simply "carried along" during the sort) are not. If we wanted the file sorted first by last name and then by first, there are a couple of ways we could go about it. The simplest and most logical method in this particular case would be to specify a single input key that would include both names since they are placed in the record with last name first. The key would simply be "1-20". However, for instructional purposes, we will perform the sort with multiple input keys. In other words, we will first specify a key for the last name and then another for the first name. The prompt and response would look like this

```
ENTER INPUT KEYS.  DEFAULTS TO "A(1)1-10".
? 1-10,11-20
?
```

specification. This should now be as shown.

```
ENTER INPUT KEYS.  DEFAULTS TO "A(1)1-10".
? D1-20
?
```

Performing the sort with these parameters should give you

```
=== TSC SORT/MERGE V1.3 ===

SORT RUN 01 - 10 RECORDS
WILSON    JIM       4578339
TYLER     HENRY     4575573
SOUTH     MARGARET  4221267
SMITH     JERRY     4226002
SMITH     HAROLD    4227264
PERRY     ARNOLD    4226008
LYON      WILLIAM   8538227
FREEMAN   LINDA     4229105
CASWELL   AMY       9258411
ABBOTT    JIM       4572091

10 RECORDS SORTED
```

It is also possible to have multiple input keys with some keys specified in ascending order and some in descending order. If for some reason we wanted the last names sorted in descending order but the first names in ascending order, we could specify input keys of "D1-10,A11-20" or taking advantage of the default sort order, "D1-10,11-20".

## 3.3 SORTING WITH OUTPUT KEYS

We now turn our attention to output key specification. In all the above examples, we output the entire input record by default on the prompt to enter output keys. It is possible to do a certain amount of output formatting by use of output keys. These output keys are similar to input keys in that they allow you to send selected portions of the input record to the sorted output records. As with the input key, the simplest form of output key would be "sss-eee" where 'sss' represents the starting column and 'eee' represents the ending column.

Let's assume we wish to sort the file NAMES.TXT according to last name, but want the output to be only the last names. You would begin exactly as before with an input key of "1-10", but when prompted with

```
ENTER OUTPUT KEYS.  DEFAULTS TO ENTIRE RECORD.
?
```

do not use the default. Instead, type in the output key necessary to send only the last name to the output record. In this case it would be columns 1 through 10, so enter "1-10". Performing the sort with these

```
=== TSC SORT/MERGE V1.3 ===

SORT RUN 01 - 10 RECORDS
       AMY
    ARNOLD
    HAROLD
     HENRY
     JERRY
       JIM
       JIM
     LINDA
  MARGARET
   WILLIAM

10 RECORDS SORTED
```

Notice that the first names are all right justified. This is a convenient feature for producing readable output lists from the sort/merge package. Right or left justify can also be used with input keys as will be described later.

There are three special types of output keys that may be used. One is a simple tab function for tabbing to a specified column and the other two allow the insertion of some string of characters or words that are not in the input record.

The tab function is called by typing an at-sign ('@') followed by the column number (largest allowed = 255) to which you wish to tab. For example, "@25" is a valid output key which will cause a tab to column 25 before any further outputting. The tab is performed by simply inserting the required number of spaces in the output record to reach the specified column.

Another of the three special output key types allows you to specify some particular chracter to be inserted into the output key. The character is specified by a hexadecimal ASCII value (see Appendix A) preceded by a dollar-sign ('$'). For example to send a period to the output record we could use a key of "$2E". This also allows the insertion of control characters into the output record. A control character is a non-printing character whose ASCII value is less than hex 20.

The third special type of output key permits whole strings of characters to be inserted. The string of characters is preceded and followed by an apostrophe or single-quote. For example, a key of 'SAMPLE' (apostrophes included) would insert the word SAMPLE into each output record. Note that an apostrophe cannot be included in the string of characters as it would appear as the string terminator. If an apostrophe is desired in the output key, it must be specified as a hex value character as shown above.

Let's put some of these special output keys to work in another example with our NAMES.TXT file. Answer all prompts as before until you reach the input key prompt. Here use an input key of "1-20" for an alphabetical listing of last and first names. Respond to the output key prompt as shown.

```
WILSON,JIM,4578339
SMITH,JERRY,4226002
ABBOTT,JIM,4572091
CASWELL,AMY,9258411
PERRY,ARNOLD,4226008
SOUTH,MARGARET,4221267
SMITH,HAROLD,4227264
LYON,WILLIAM,8538227
TYLER,HENRY,4575573
FREEMAN,LINDA,4229105
```

Notice that the list is exactly as before except that instead of finding the names and number in fixed columns they are packed together with only a comma separating them. These can be considered fields since they do have a character separating them. We will need to specify to the SRTMRG.SYS program what the field separator character is. We will also need to specify from which field the input or output keys are to come. This is done by placing the field number in parenthesis before the starting and ending key columns. For example, to specify the phone number in NAMES2.TXT we would type "(3)1-7". This says to use columns one through seven of field three. The column numbers no longer refer to the entire input record but rather to the particular field specified. If no field separator character is specified (defaults to a null as in the examples above), the SRTMRG.SYS program assumes that the input record is all one field. Note that the field number specification in a key defaults to one if there is not a parenthesis enclosed number. Thus in all the example sorts above, we were defaulting to field one since there was only one field.

Let's try sorting NAMES2.TXT. Suppose we wish to sort on the phone numbers and output the number followed by last name, comma, first name. We would begin by typing

```
SORT NAMES2.TXT  or just  SORT NAMES2
```

since the extension defaults to TXT. The first several prompts and responses should be exactly as before and are shown here as they would appear on your screen.

```
=== TSC SORT PARAMETER EDITOR ===

OUTPUT TO DISK (Y OR N*)? N
INTERMEDIATE WORK FILE DRIVE?

FIXED OR VARIABLE LENGTH RECORDS (F OR V*)? V
EOR CHARACTER OR FIELD COUNT (DEFAULT IS EOR=$0D)?
```

At this point we receive a prompt for a field separator character. We wish to specify a comma and there are two ways to do that. You may specify the character as a hex value by prefacing the value with a dollar-sign ('$') or as an ASCII character by prefacing the character with an apostrophe or single quote. For a comma the prompt and response would be

```
FIELD SEPARATOR CHARACTER? $2C
```

column one through the end of the field but that is only allowed on output keys - not input keys. All input keys must be of the same length, so it is required that you specify that length by giving an ending column number. You should in fact specify an ending column that will fit the largest number one field found in any of the input records. If there are less actual columns in the field than you specify, the sort/merge program will "pad" the key with spaces to fill out the number of columns specified. In our case, we know that field number one is never longer than 10 characters, so we might specify "(1)1-10" or using the default field, simply "1-10". Note that you can always specify a key that is longer than necessary if you are unsure of the largest field. The only problem with this is that the larger the input keys, the more memory is required and the longer the sort will take. Enter the input key of "(1)1-10" and an output key of your choice (or default to outputting the entire record). The sort key that is built up of the input key for the first record would have 10 characters in it, the letters "WILSON" followed by four spaces. Executing the sort with the entire input records being output will yield the following.

```
=== TSC SORT/MERGE V1.3 ===

SORT RUN 01 - 10 RECORDS
ABBOTT,JIM,4572091
CASWELL,AMY,9258411
FREEMAN,LINDA,4229105
LYON,WILLIAM,8538227
PERRY,ARNOLD,4226008
SMITH,HAROLD,4227264
SMITH,JERRY,4226002
SOUTH,MARGARET,4221267
TYLER,HENRY,4575573
WILSON,JIM,4578339

10 RECORDS SORTED
KEY PADDING WAS REQUIRED
```

You will notice a message to the fact that key padding was required. This is not an error message, just a report of the fact that padding was required.

The same type of padding will be done on output keys if necessary. There will, however, be no report of such as with the input key padding. For instance, an output key in the preceding example of "(2)1-15" would print the first name and then spaces until 15 columns had been printed.

# 4.0 GENERAL USER SPECIFICATIONS

There are several things which were touched on in the preceding tutorial which may be specified to SRTMRG.SYS such as input and output keys. These specifications are the same for all five parameter supplying commands and are therefore elaborated on in this section. This section should be read prior to the sections which follow on the indivdual commands.

## 4.1 SPECIFYING INPUT RECORDS

As seen before, each input file is made up of "records" of information. These records are to be sorted into some logical order. Since these input records can vary in type and size, we must have some way of specifying to the sort/merge program where one record ends and another begins. There are two basic types of input records, "fixed length" and "variable length". Fixed length records are as the name implies records which will all be the same length. It is not necessary to have some character to mark the end of the records, but rather simply to specify how long the record is in number of characters. Variable length records do not have to all be of the same length. They are specified in one of two ways. The first is to specify some particular character which signals the end of the record. This character is called an "End Of Record" character. The second method is to specify some field count. In other words, the user would specify a count which would be the number of fields included in each record. These fields are signaled by an End of Field character as described before. Thus you may have a file which is made up of a large number of fields and split it into records by giving a field count.

Most files will probably be sorted as variable length files with an end of record (EOR) character. All five sort/merge commands default to this type with a carriage return (hex 0D) as the EOR. All our examples in the preceding section were done with this type of record. Note that the EOR character is never included in the input record that the sort/merge sees. It is in effect "swallowed up" as the records are read.

If we wanted, we could have sorted the NAMES.TXT file as a fixed-length record file. The only problem would be that the carriage return would then NOT be swallowed up by sort/merge. The carriage return would have to be part of the record. If we wanted to do this, we would simply specify the decimal number of characters to be put into each record. In the case of NAMES.TXT that number would be 27. We could have answered the prompts associated with record specification as follows.

```
FIXED OR VARIABLE LENGTH RECORDS (F OR V*)? F
```

include the following:

A ... Ascending sort order
D ... Descending sort order
L ... Left justify the key
R ... Right justify the key

At most there should only be two, 'A' or 'D' and 'L' or 'R'. If conflicting options are specified (both ascending and descending or both left and right justify) the last one specified will take precedence. These option letters may be specified in any order, so long as they come before the field number and column specs. It is not required to give any options. If this is the case, the key defaults to ascending with neither right or left justify.

Up to 20 input keys may be specified and may come in any order. That is to say, the first key might come from the end of the record while the second key came from the start. Keys may also overlap each other's position in the input record.

## 4.3 SPECIFYING OUTPUT SOURCE

The sort does not actually sort the entire input records. It only sorts the input keys which were specified. These input keys have pointers to their parent record's location on the disk. When it comes time to output the input records in order, the sort program looks at which key is highest, finds out where it came from, and then re-reads that input record, writing it to the output as specified by the output keys. This means the "source" for the output is the input file. It is possible, however, to specify that the output data come from another location, namely the key itself. In the SORT command, for example, there is a prompt,

OUTPUT FROM KEY, INPUT, OR OTHER (K, I*, OR O)?

In all the previous examples, we selected the input as our output source. By simply typing a 'K', we can cause the output to come from the sort key itself. It is still possible to default to outputting the entire record (in this case the entire key) or to specify output keys with the columns now referring to the columns in the key rather than the input record. Note that there will rarely be fields to specify if outputting from the key. The advantage to outputting from the key is that it is considerably faster than outputting the input record since it is not necessary to re-read the input file as described above. There are two disadvantages to outputting from the key. First is that all the data from the input may not be in the key. Obviously when outputting from the key, only those portions of the input record which have been used as key data may be output. The second disadvantage is that if the upper case equal to lower case option is selected, all letters in the key will have been converted to upper case. This may or may not be acceptable depending on the situation.

2) Entire Field Spec... This type of key is much like the previous type except that the field MUST be specified and no starting or ending columns are specified. The entire field is sent to the output record. For example, to output all of field 3 we could enter "(3)" as an output key which would be equivalent to "(3)1-E". This type of key simply saves typing over the first type.

3) Literal String... This type of output key allows the insertion of some constant string of characters (a literal) into the output record. The same string will be sent to each record output. The string of characters is specified by enclosing the characters in single quotes. Any printable ASCII characters may be included.

4) Hexadecimal Value... Any single, 2 digit hexadecimal value may be inserted in the output record by specifying it with a preceding dollar-sign. For example, to insert a carriage return in the output simply type "$0D". This allows any ASCII character to be output, printable or non-printable. If more than two hex digits are typed, only the last two are used with the others being ignored.

5) Horizontal Tab... Another type of output key is the horizontal tab which allows you to tab over to some particular column number in the output record with spaces used for padding. This key is specified as an at-sign followed by the decimal column number. For example, to tab over to column 25, enter "@25". If you are already past the column specified in a tab key, the tab key specification will be ignored.

The output keys may call data from the input record or sort key in any order. That is to say, data from field 4 may precede data from field 2. The only limit on the number of output keys is the amount of space reserved for the input and output keys. This should always be sufficient unless several very large literal string keys are specified.

# 5.0 THE SORT COMMAND

The SORT command is perhaps the easiest method of performing a sort operation as it prompts you for all the necessary parameters. This obviates the need to memorize what parameters must be supplied and how to supply them. The disadvantage is that for simple sorts which need few parameters you must still answer all prompts. This is simplified by accepting defaults for most prompts. Operation is as follows. Upon issuing a SORT command, the SORT.CMD module is loaded. This module interacts with the user, prompting for the necessary parameters. It is thus called a "parameter editor". When all parameters have been obtained, the user is allowed to save these parameters as a disk file if desired. Then he may exit the SORT module (back to DOS) or may continue with the sort. If elected to continue, the SORT module will attempt to load the SRTMRG.SYS module from the same disk as the SORT.CMD module. If successful, the sort will then be performed.

## 5.1 GENERAL USE OF SORT

To initiate SORT, simply type a command of the general form:

```
SORT <file>
   or
SORT <file1>,<file2>,<file3>,...
```

Where "<file>" is a standard file specification for the file to be sorted. The default extension is TXT. Note that more than one file may be sorted. When one file has been completely read it is closed and the next file is immediately opened for reading. All the files must, however, reside on the same disk. The output file will consist of all the records of the specified input files. It is possible to simply type "SORT" with no file specifications. This is useful only when the user wishes to edit a parameter file and not proceed with a sort operation.

The computer should respond to the SORT command by typing

```
=== TSC SORT PARAMETER EDITOR ===
```

This shows that the parameter editor has been entered and the prompts for sort parameters will follow. These prompts and the responses they require are described here. Note that the prompting may be restarted at any time by typing a 'CTRL Z'.

FIELD SEPARATOR CHARACTER?

At this point a field separator character may be specified as a hex value preceded by a dollar-sign or as a printable ASCII character preceded by a single quote. If no field separator is desired, simply type a carriage return as the default is a null field separator character. Note that if a field count was specified as the input record terminator a field separator character will be required.

OUTPUT FROM KEY, INPUT, OR OTHER (K, I*, OR O)?

This allows the user to specify the source for output data. The default is the input record. Typing a 'K' will cause the output data to come from the sort key itself. Typing an 'O' will cause another prompt to be issued as follows.

OUTPUT INDEXED OR TAG FILE (I OR T*)?

This allows the basis for an indexed file or a tag file to be output. These type files are elaborated on later in this manual. The user will probably never need to specify an indexed file, it is merely added for completeness. The tag file is specified with a 'T' or simply a carriage return.

p.57

ENTER INPUT KEYS. DEFAULTS TO "A(1)1-10".
?

The input keys may be entered all on one line or on several lines. To put more than one key on a line, separate them with a comma or a space. When all desired keys have been entered on a line, hit a carriage return. The computer will respond with another question mark which is prompting for more keys. If you do not wish to enter more keys, the key prompt mode may be terminated by hitting a carriage return. See section 4.2 for details of input key specifications.

p.22

SYS-DIR. = 1-4

ENTER OUTPUT KEYS. DEFAULTS TO ENTIRE RECORD.
?

Output keys are entered exactly like input keys with the capability to put all keys on one line or on multiple lines. See section 4.4 for details of output key specification.

p.24

FURTHER OPTIONS REQUIRED (Y OR N*)?

At this point, the basic parameters necessary for a sort operation have been specified. There are, however, several other parameters or options which may be specified. If you need to set further options, type a 'Y'. You will then be prompted for them as described in section 5.2. If no further options are required, type an 'N' or simply default with a carriage return.

p.30

At this point we see the message

=== PARAMETERS ARE NOW SET ===

This informs us that all the necessary parameters are set in the computer. We now have the opportunity to save these parameters as a file and then to exit or continue with the sort operation. The prompts for these functions follow.

ALTERNATE COLLATING SEQUENCE (Y OR N*)?

The sort/merge package normally does all sorting according to the ASCII collating sequence (see Appendix A). If the ASCII sequence is suitable, type an 'N' or simply a carriage return. If it is necessary to sort according to a different collating sequence, type a 'Y'. You will then be prompted for the file name of the desired collating sequence file. For a decription of the format of an alternate collating sequence file see section 12.0.

C/R

TREAT LOWER CASE EQUIVALENT TO UPPER (Y OR N*)?

In the ASCII coding scheme there is a different code for upper and lower case letters. This implies that an upper case 'E' would not sort equivalently to a lower case 'e'. In fact, the upper case characters are all lower in value than the lower. Thus a 'Z' would be sorted before an 'a' if a normal ascending sort was performed. Typing a 'Y' in response to this prompt will cause the sort program to treat upper case letters equivalent to lower case. In actuality, the sort keys are all converted to upper case. For this reason, if the output records come from the key instead of the input records, all letters would be upper case. Note that this feature is functional only if the ASCII character set is being used. If upper case SHOULD be sorted different from lower case, simply type an 'N' or a carriage return.

Y

DELETE RECORDS WITH BLANK SORT KEYS (Y* OR N)?

Depending on how the key is specified and what the data contains, it is possible to end up with sort keys which are all spaces or blanks. Generally these keys are of no value since they contain no information. They simply take up space in memory and thus slow down the sort. These keys may be deleted from the sort operation by typing a 'Y' or a carriage return. In effect, the input record is hereby deleted. It still remains a part of the original data file, but no information from it is included in the output. This fact is aluded to at the end of a sort operation when a message is printed stating the number of records deleted. In some cases, a blank key may be meaningfull and should not be deleted. If this is the case, type an 'N' in response to the prompt and the blank keys will be included.

N

SELECT/EXCLUDE OPTION (Y OR N*)?

P.32

The sort/merge package has the ability to select or exclude certain records from the sort depending on their contents. If this option is not required, type an 'N' or a carriage return. If it is required, type a 'Y'. Several related prompts will then be issued. The specifications required for this option are quite involved and thus a separate section of this manual has been devoted to describing them. It follows shortly as section 5.3.

(1.4, 47.)

N

IS OUTPUT FILE TEXT OR BINARY (T* OR B)?

As with the input records, the output records may contain text or binary data. In general the type of the output file (text or binary) should coincide with that of the input file. Type a 'T' or carriage return for text or a 'B' for binary. This effectively sets or clears the space compression flag in FLEX before any output is performed.

B

This is a prompt for a SINGLE input key specification which tells what portion of the input record to which you wish to compare the select/exclude key. We want to compare to the exchange number, so you should respond with a "21-23" which are the columns containing the exchange. Now we receive the prompt:

```
SELECT OR EXCLUDE (S* OR E)?
```

This allows us to either select matching records or exclude them. We want to select all records with "422" as the exchange so type an 'S' or simply a carriage return. We now see the prompt:

```
ON KEY '<', '=', OR '>' (DEFAULT IS '=')?
```

Here we can specify that we want the records to be selected (or excluded had we so chosen) if the key is less than the specified portion of the input record ('<'), equal to it ('='), or greater than ('>'). We want to select if "422" is equal to columns "21-23" of the input record so type an equals sign ('=') or default with a carriage return. The final select/exclude prompt will now be issued:

```
KEY STRING?
```

This is the prompt for the actual data which you want to select or exclude on. We want to select on the exchange equal to "422" so simply type "422" followed by a carriage return.

At this point the prompts will continue as described in section 5.2. When the sort operation is complete, a run-time message will be printed informing you of the number of input records which were excluded from the output. If you were selecting records (as in our example) this would effectively be the number of records which were not selected.

The output from our example would look something like this (depending on what output key specifications you chose):

```
=== TSC SORT/MERGE V1.3 ===

SORT RUN 01 - 5 RECORDS
FREEMAN   LINDA     4229105
PERRY     ARNOLD    4226008
SMITH     HAROLD    4227264
SMITH     JERRY     4226002
SOUTH     MARGARET  4221267

5 RECORDS SORTED
5 RECORDS EXCLUDED
```

There are limitless possibilities to what can be accomplished with the Select/Exclude option. We could have easily excluded all records with an exchange of 422. We could have selected all records which had an exchange higher than 399 (i.e. 400 through 999). By doing multiple sort operations, we can even be more selective. Say for example we wish to sort all names which have a phone number with an exchange between 200 and 600 inclusive. First perform a sort selecting all

## 6.0 THE PSORT COMMAND

The PSORT command allows a user to supply all the necessary sort parameters in a named disk file. This file may be created through the use of the parameter editor (the SORT command described in section 5.0). The PSORT command simply loads this file into the parameter area between $00C0 and $027F, loads the SRTMRG.SYS module, and then proceeds with the sort. This type of sort operation is especially convenient where one type of sort must be repeated on several files or repeated several times on one often changed file.

There are two basic forms of this command:

```
PSORT <parameter file>,<input file>
                or                          e.g. PSORT.2,
PSORT <parameter file>,(<output file>),<input file>
```

The default extension for the parameter file is 'BIN' while the input and output files default to a 'TXT' extension. The first form shown loads the parameter file specified, loads SRTMRG.SYS from the same disk as PSORT.CMD and proceeds to sort the input file specified. The second form allows one of the parameters in the parameter file to be altered, that being the output file specification. The first form simply uses whatever was specified in the parameter file as the output file spec but the second form allows the user to specify some different file as the output file. Note that the parentheses are required to differentiate an output file spec from an input file spec. It is possible to specify an output file even though none was specified in the original parameter file. It is also possible to specify that no output file be produced (even if one is specified in the parameter file) by placing the parentheses in the command with no output file specified inside them. In other words entering "()" will disable the production of an output file. Note that in any case the parameter file remains unchanged.

Multiple input files may be specified with either form as shown:

```
PSORT <parmeter file>,<file1>,<file2>,<file3>,...
                or
PSORT <parameter file>,(<output file>),<file1>,<file2>,...
```

The input files will be sorted together to produce one output file containing all the records of all the input files.

Several checks are made to ensure that the specified parameter file is actually a properly formatted parameter file. First, it must be only two sectors in length (four in mini FLEX version) which is all that is required to contain all the parameters. Second, it must be a binary type file as opposed to text. Third, it must load at location $00C0. And finally, the parameter file must be for a sort operation as opposed to a merge-only operation. Any parameter file produced by the SORT

## 7.0 THE CSORT COMMAND

The CSORT command allows the user to specify the necessary sort parameters on the command line (CSORT stands for Command line Sort). This is very convenient for users who are proficient with the sort operation and don't need the prompting of the SORT command or for those users who have an aversion for profuse prompting. The user simply types a single command line containing any desired parameters as shown below and hits a carriage return. The CSORT module fills in the required parameter area, loads SRTMRG.SYS, and immediately begins execution of the sort.

The general form of a CSORT command is as follows:

CSORT <infile>,<outfile>,+<input specs>,+<output specs>

There are several things to note about this line. First, both <infile> and <outfile> are standard FLEX file specifications and default to a 'TXT' extension. Note also that only one input file may be specified to CSORT and is required. Everything past the input file name is optional. However, if there are any output specs, both plus signs must be included (even though there may be no input specs). Several samples later in this section will help clarify the syntax of this command line.

Once a plus sign has been hit in the command line, CSORT begins looking for input specifications. These include input sort keys, record specifiers, etc. They may come in any desired order on the command line and are separated by either a space or a comma. The input specs may be any of the following:

W=<decimal>
: This is the work drive specification or the drive number onto which any temporary work files are written. <decimal> is the desired drive number from 0 to 3. If this parameter is not specified, the work file drive will be the assigned system drive of FLEX or the first available drive if unassigned.

L=<decimal>
: This is the input record length specification. <decimal> is the decimal length of the fixed length input records. It may be any number from 1 to 64K.

E=<hex or ASCII>
: This is the End of Record character specifier for input records. <hex or ASCII> represents a hex character value preceded by a dollar-sign or an ASCII character preceded by a single quote.

If CSORT sees another plus sign while looking for input specifications, it will immediately terminate its search for input specifications and begin looking for output specifications. These output specifications are entered much as the input ones. That is to say they may come in any order and should be separated by a space or comma. Possible output specifications are as follows:

O=<I, K, or T>
: This is the output record source specification (see section 4.3). The output may come from the input records by entering 'O=I', it may come from the sort key by typing 'O=K', or a tag file can be produced by typing 'O=T'. If none of these are entered, CSORT defaults to 'O=I'.

E=<hex, ASCII, or N>
: This is the End of Output Record specification. <hex, ASCII, or N> represents a hex character value preceded by a dollar-sign, an ASCII character preceded by a single quote, or typing 'E=N' instructs sort/merge that the end of output record character is to be null, that is there will be NO character output at the end of output records. If this specification is left out, CSORT defaults to a carriage return as the end of output record character.

T or B
: These are the output file type specifiers. By simply typing a 'T' as an output specification, the user can indicate that the output file is to be a text type file and thus will have space compression. Typing a 'B' implies a binary type output file which means space compression will be turned off before writing to the output file. If neither of these characters are entered on the command line, the output file is assumed to be a text type file and will therefore be space compressed.

M
: This is the message level specifier. Entering this character as an output specification will suppress all run time messages during the sort/merge operation.

STANDARD OUTPUT KEYS
: Any standard output key may be included in the output specifications. For details on a standard output key specification see section 4.4. Note that if no standard output keys are given, CSORT outputs the entire input record to the output record.

As before, the output specifications may appear in any order but the order of any output keys is significant. If there are conflicting specifications, the last one given is used with the first ones ignored.

Note that in order to make CSORT convenient to use, some of the features of sort/merge were not implemented. In particular, only one input file may be specified, an alternate collating sequence is not allowed, the select/exclude option is not allowed, and it is not possible to set an upper memory limit. If the user needs these features, he has no alternative but to use the SORT and PSORT commands.

# 8.0 THE MERGE COMMAND

To this point, most descriptions have centered around the sort operation. We now turn our attention to the merge-only type operation. Note that merging may take place in a sort but it is transparent to the operator. A "merge-only" operation means that no actual sorting is to be done. Instead, two or more files that are assumed to be already individually sorted are merged into one, ordered file. This is done by looking at the top record of each input file and selecting the lowest record (or highest depending on the order of the merge) to be sent to the output. The next record in that input file is then brought to the top for the next compare. Note that if the input files are not in sorted order the output file will not be in order. The MERGE command is much like the SORT command in that it is essentially a "parameter editor" to setup the parameters for the SRTMRG.SYS module.

To initiate a merge-only operation enter a command of the form:

MERGE <file1>,<file2>,<file3>,...

The file specifications are standard FLEX file specs with a default extension of 'TXT'. Any number of input files may be specified so long as they fit on the command line. These input files MUST all reside on the same disk. It is also possible to simply type "MERGE" with no input file specifications. This is useful only when the user wishes to edit a parameter file and not proceed with the merge operation.

The computer should respond to the MERGE command line with:

=== TSC MERGE PARAMETER EDITOR ===

This shows that the parameter editor has been entered and the prompts for sort parameters will follow. The prompts which are issued are identical to those of the SORT command of section 5.0 and require the same responses with two exceptions. The first is that the Select/Exclude option is not allowed in a merge-only operation. Thus there is no prompt for such in MERGE. The second difference is that the final prompt is now "EXIT OR PROCEED WITH MERGE (E OR M)?". This prompt may not be defaulted but must be answered with an 'E' or 'M'.

The remainder of the prompts are identical to those in SORT and the reader is directed to section 5.1 for further descriptions. As in the SORT command, a 'CTRL Z' will restart the prompts from the top.

# 9.0 THE PMERGE COMMAND

The PMERGE command allows a user to supply all the necessary merge-only parameters in a named disk file. This file may be created through the use of a parameter editor (the SORT command described in section 5.0 or the MERGE command in section 8.0). The PMERGE command simply loads this file into the parameter area between $00C0 and $027F, loads the SRTMRG.SYS module, and then proceeds with the merge. This type of merge operation is especially convenient where one type of merge operation must be often repeated. It is also valuable for performing sorts on several files using a parameter file sort and then merging these files with PMERGE using the same parameter file.

There are two basic forms of this command:

```
PMERGE <parameter file>,<file1>,<file2>,...
                or
PMERGE <parameter file>,(<output file>),<file1>,<file2>,...
```

The default extension for the parameter file is 'BIN' while the input and output files default to a 'TXT' extension. The first form shown loads the parameter file specified, loads SRTMRG.SYS from the same disk as PMERGE.CMD and proceeds to merge the input files specified. The second form allows one of the parameters in the parameter file to be altered, that being the output file specification. The first form simply uses whatever was specified in the parameter file as the output file spec but the second form allows the user to specify some different file as the output file. Note that the parentheses are required to differentiate an output file spec from an input file spec. It is possible to specify an output file even though none was specified in the original parameter file. It is also possible to specify that no output file be produced (even if one is specified in the parameter file) by placing the parentheses in the command with no output file specified inside them. In other words entering "()" will disable the production of an output file. Note that in any case the specified parameter file remains unchanged.

Several checks are made to ensure that the specified parameter file is actually a properly formatted parameter file. First, it must be only two sectors in length (four in mini FLEX version) which is all that is required to contain all the parameters. Second, it must be a binary type file as opposed to text. And finally, it must load at location $00C0. If any of these conditions are not met, an error message will be issued and the merge operation will be aborted.

It should be noted that PMERGE can make use of a parameter file which was prepared as a merge-only parameter file OR one prepared as a sort parameter file.

## 10.0 RUN-TIME MESSAGES

There are several run-time messages which may be printed out during the operation of the sort/merge package. These are not to be confused with error messages as they report no error. They simply report on the status of the sort/merge since the operation can require quite a lengthy period of time. These messages are printed by the SRTMRG.SYS module which performs the actual sorting and merging. One message is simply a header at the outset of a sort/merge operation, one is printed for each sort run, one for each merge pass, and up to four messages are printed upon completion of the operation to summarize just what occurred. Recall that a sort run is one memory buffer full of data which is sorted individually and saved as a temporary file while other runs are sorted. These messages are printed on the operator console only. They are not sent to an output file (if one is specified) and are not sent to a line printer (if one is selected by use of the 'P' command in FLEX). Thus the operator may monitor the sort operation at the console while output is being sent to a disk file or to a printer.

These run-time messages are optional. They may be turned off if desired as explained in the separate parameter supplying commands, SORT, CSORT, and MERGE. If so, absolutely no run-time messages will be issued.

The messages are listed and explained here:

1) === TSC SORT/MERGE Vx.y === V2
   Where "x.y" is the version number of the SRTMRG.SYS module in use. This is simply a header message to let the operator know that the sort/merge operation has successfully begun. It is the very first function that SRTMRG.SYS performs.

2) SORT RUN xx - yy RECORDS
   This message is printed for every sort run required. "xx" represents the sequential number of the run in progress while "yy" represents the number of records which are contained in that run. This line is actually printed in two parts. The sort run number is printed before any work has been initiated on the particular run. The number of records in the run is printed after the run has been read into the computer's memory. Thus you will see the sort run message, a pause while the records are being read from disk into memory, the number of records message, and then another pause while the run is sorted and written out to a temporary file.

3) MERGE PASS xx - yy RUNS
   This message is printed for every merge pass required. This will generally be only one pass. "xx" represents the sequential number of the merge pass in progress while "yy" represents the number of runs being merged in that pass. This line is all printed at once before the merge pass has been initiated.

## 11.0 ERROR MESSAGES

There are several error messages associated with SRTMRG.SYS and the five parameter supplying routines. Many of them are common between routines and will therefore be grouped together in this section to avoid redundancy. We will place the messages in four groups, those from SORT and MERGE, from PSORT and PMERGE, from CSORT, and from SRTMRG.SYS.

There is one type of error message issued by PSORT, PMERGE, and SRTMRG.SYS which should be brought out here. It is a two line message with the first line being:

ERROR WITH FILE '<filename>'...

where <filename> is the name of the file with which the error occurred. The second line of the message will follow immediately and will be a normal FLEX error message. The user should refer to the FLEX User's Guide for descriptions of these messages.

### 11.1 ERROR MESSAGES FROM SORT AND MERGE

In general, the error messages associated with these two modules are of an interactive, non-fatal nature. In other words, they inform you of an error and allow you to re-enter the corrected data. They do not cause the program to abort. The messages are as follows:

1) ILLEGAL FILE SPECIFICATION
   An illegal file specification has been given in response to a prompt for such. The prompt will be re-issued and a valid file specification should be entered.

2) NON-ZERO FIELD COUNT REQUIRES A FIELD SEPARATOR
   This message is issued if the input records were specified by a field count and then no field separator character is specified. In order to count fields, there obviously must be some way of separating the input record into fields. The prompt will be re-issued.

3) ILLEGAL KEY SPECIFICATION
   An error was found in one of the keys in the line just entered. Any keys already entered are discarded and a new prompt is issued.

## 11.2 ERROR MESSAGES FROM PSORT AND PMERGE

These errors are generally fatal in that they cause the execution of sort/merge to be terminated and control returned to FLEX.

1) ERROR WITH FILE '<filename>'...
   The standard FLEX error message which follows this message describes an error which occurred with the file named.

2) ILLEGAL PARAMETER FILE SPECIFICATION
   The file specification given for the parameter file is invalid.

3) ILLEGAL OUTPUT FILE SPECIFICATION
   The file specification given for an output file is invalid.

4) PARAMETER FILE NOT BINARY
   The specified parameter file is not a binary type file. Check to be sure you have entered the correct name and extension or assumed the correct default extension.

5) ILLEGAL PARAMETER FILE
   There are two possible reasons for this message. First is that the specified parameter file does not load into the correct parameter area for SRTMRG.SYS. The second is that the specified parameter file is not of the correct length. PSORT and PMERGE require that the parameter file be exactly two sectors in length (four in mini FLEX) to prevent invalid files from being loaded. Check to be sure you have specified the correct file.

6) 'SRTMRG.SYS' NOT PRESENT ON DISK
   Issued if the file 'SRTMRG.SYS' cannot be found on the same disk from which PSORT or PMERGE was loaded.

There is one error message in PSORT which is not found in PMERGE:

7) PARAMETER FILE IS MERGE-ONLY
   This message is issued if the parameter file specified to PSORT was prepared using MERGE. This arrangement is not permissable.

There are two error messages in PMERGE which are not found in PSORT. They come about due to the fact that PMERGE scans the input file specifications while PSORT leaves that task to the SRTMRG.SYS module.

8) ILLEGAL INPUT FILE SPECIFICATION
   One or more of the input file specifications is invalid.

9) INPUT FILES NOT ON SAME DISK
   The merge operation requires that all input files reside on the same disk. This message is issued if the different disks are specified.

10) ILLEGAL FIELD COUNT
The field count specified is invalid. It must be a decimal number between 1 and 255 inclusive.

11) ONLY ONE OF L, E, OR C MAY BE SPECIFIED
This message is issued if more than one of the three record defining specs was found. It is only possible to have one of the three in a single command.

12) ILLEGAL FIELD SEPARATOR
The field separator character was specified incorrectly. It must be either a hex value preceded by a dollar-sign or an ASCII character preceded by a single quote.

13) NON-ZERO FIELD COUNT REQUIRES A FIELD SEPARATOR
This message is issued if a field count has been specified but no field separator. In order to have a field count, there must be a field separator character.

14) OUTPUT KEY ERROR
An error was found in one of the output keys specified.

15) ILLEGAL OUTPUT SOURCE SPECIFIED
The output source may be specified only as 'I', 'K', or 'T' for input, key, or tag.

16) ILLEGAL END OF OUTPUT RECORD CHARACTER
An invalid end of output record character was specified. It must be a hex value preceded by a dollar-sign, an ASCII character preceded by a single quote, or the letter 'N' which signifies a null end of output record character (no EOOR character appended to record).

17) TOO MANY KEYS SPECIFIED
Sort/Merge reserves 256 bytes for input and output key specifications. The user is limited to 20 input keys and as many output keys as will fit in the remainder of the 256 bytes not used by input keys. This message is issued if more than 20 input keys were specified or if the input and output keys overflowed this 256 byte limit. The only recourse is to lower the number of specified keys.

18) EQUALS SIGN REQUIRED IN SPEC
As seen in section 7.0, many of the input and output specs are a single letter, followed by an equals sign, followed by the actual parameter. If the equals sign is omitted from one of these specs, this error message will result.

19) SYNTAX ERROR
This message is issued if there is a syntactical error in the command line such as more than two plus signs.

7) CANNOT RENAME A .BAK OUTPUT FILE

This message is given when an output file with a "BAK" extension was specified and a file by that name and extension already exists. This is an illegal situation.

8) ERROR IN RENAMING OUTPUT FILE. RE-TRY.

If the output file specified already exists on the disk, sort/merge attempts to rename the one on the disk to a backup and then write the new file. If this message occurs, there was probably some problem with the hardware and you should try repeating the entire sort or merge process.

9) SEQUENCE FILE NOT BINARY

The alternate collating sequence file specified was not a binary type file.

10) ILLEGAL SEQUENCE FILE

This indicates that the alternate collating sequence file specified is invalid for one of two reasons. The first is that the specified file is too large (an alternate collating sequence file should only be 2 sectors long in FLEX or 3 sectors in mini FLEX) and the second is that the file did not have a valid load address (the alternate collating sequence file should load at $0600.

11) SORT KEY TOO LONG

The total length of all the input keys specified must not exceed 250 bytes. If it does, this error message is issued and execution is terminated.

12) TOO MANY WORK FILES

Sort/Merge allows a maximum of 99 temporary work files. This should never be a limitation, but if the number is exceeded, this error message is issued. The only solution is to increase the amount of user memory available to sort/merge.

## 12.0 ALTERNATE COLLATING SEQUENCE FILE

Sort/Merge generally performs all sorting and merging according to the ASCII sequence of characters (see appendix A). Thus for example, a '3' is higher in the sequence than a 'P' which is higher than a 'g'. It is possible to sort according to some other sequence or to use a different code than ASCII. This is done thru the use of an "alternate collating sequence file" which is simply a named disk file which contains all the characters listed in the desired order (each character is represented by a single byte). Thus we could use the same ASCII code (ie. '3' = 33 hex, 'P' = 50 hex, and 'g' = 67 hex) but place the characters in a different sequence in the alternate collating sequence file so that the collating order of these three characters might be 'P', 'g', and '3' or possibly 'g', 'P', and '3'. It is also possible to specify a totally different character set such as EBCDIC by simply specifying the proper character representations in the desired order.

An alternate collating sequence file must adhere to the following specifications:

1) Must be a binary type file.
2) Must load at 0600 hex.
3) Must be 2 sectors long on FLEX 1.0 & 2.0 versions
4) Must be 3 sectors long on mini FLEX version

The most logical way to prepare an alternate collating sequence file is with an assembler. A sample program is listed here which when properly assembled will produce a valid alternate sequence file. This file uses the ASCII code, but sorts spaces first, followed by upper case, followed by lower case, followed by numbers, followed by the graphic characters and control characters.

```
* ALTERNATE COLLATING SEQUENCE FILE
 ORG $0600
 FCC / ABCDEFGHIJKLMNOPQRSTUVWXYZ/
 FCC /abcdefghijklmnopqrstuvwxyz/
 FCC /0123456789/
 FCC /!"#$%&'()*+/
 FCB $2C,$2D,$2E,$2F
 FCC /:;<=>?@/
 FCC /[\]^_/
 FCB $60,$7B,$7C,$7D,$7E,$7F
 FCB 0,1,2,3,4,5,6,7,8,9,10,11,12
 FCB 13,14,15,16,17,18,19,20,21,22
 FCB 23,24,25,26,27,28,29,30,31
 END
```

## 13.0 TAG AND INDEXED FILE OUTPUT

There are two special types of output from the sort/merge package that warrant our attention. They are tag files and indexed files. The indexed file output is not a complete indexed file, but is rather the skeleton needed by the TSC Indexed File package. As such, there is probably never a need for the user to utilize this type of output file. It is merely mentioned here for completeness.

The "tag file" output is a special output file which contains only the sorted pointers back to the original file. Thus each output record has no data from the input record, but has three pointer bytes to the absolute disk address of the start of the corresponding input record. These pointer bytes are the track address, sector address, and offset into the sector, respectively. The tag file is always a binary type file but does not contain record start markers and load addresses. It is simply a string of the three byte pointers.

The purpose of the tag file is to provide a very succinct form of storing information needed to produce a sorted file. This might be necessary where the file being sorted is very large and there is little disk space remaining or where it is desired to maintain several sorted versions of one database. Keeping several of these tag files would be much more efficient than the wasteful and redundant method of keeping several sorted copies of the database itself.

Tag files alone are quite useless. There must be some user written program to make use of the tag files and the original database as desired to print out a sorted list or perform some other function. The Sort/Merge package has no provisions for using the tag files produced. That is left up to the user for his own particular application.

## 14.0 PARAMETER FILE DESCRIPTION

The actual sorting and merging program, SRTMRG.SYS, expects to find all the necessary parameters in memory between hex 00C0 and 027F inclusive. It is the responsibility of the commands SORT, PSORT, CSORT, MERGE, and PMERGE to set up the necessary parameters in this area. Following is a map of this area and then further descriptions of each parameter.

PARAMETER FILE AREA MAP

| Location | Name | Description |
|---|---|---|
| 00C0 | OUTDRV | Output file drive number |
| 00C1-00CB | OUTNAM | Output file name |
| 00CC | ALTDRV | Alternate sequence file drive number |
| 00CD-00D7 | ALTSEQ | Alternate sequence file name |
| 00D8 | RUNDRV | Run or temporary work file drive number |
| 00D9-00DA | MEMEND | Memory end address |
| 00DB-00DC | OSPEC | Output key specs pointer |
| 00DD | EOR | End Of Record character for inpu |
| 00DE | EOF | End Of Field character |
| 00DF | EOOR | End Of Output Record character |
| 00E0 | GROUP | Field count or group count |
| 00E1-00E2 | RLNGTH | Fixed length record length |
| 00E3 | LOWUP | Treat lower case equal to upper flag |
| 00E4 | FROMKY | Output from key designator |
| 00E5 | ISPCOF | Turn off input space compression flag |
| 00E6 | OSPCOF | Turn off output space compression flag |
| 00E7 | IDXTAG | Indexed or Tag file output flag |
| 00E8 | OUTALL | Output entire input record flag |
| 00E9 | MSGLVL | Run-time message level |
| 00EA | DELNUL | Delete records w/ null keys flag |
| 00EB | SLEXCL | Select/Exclude information |
| 00EC-00F0 | SESPEC | Select/Exclude key spec |
| 00F0-00FD | | Reserved for future use |
| 00FE | MGONLY | Merge-only flag |
| 00FF | RTSFLG | Return flag |
| 0100-01FF | ISPEC | Input and output specs |
| 0200-027F | SEDATA | Select/Exclude string data |

GROUP This byte holds the group count or field count. It is the number of fields required to make up one input record. If input records are specified by an EOR or a fixed record length, this byte should be cleared to zero.

RLNGTH This is a two byte value containing the length of fixed length input records in binary. If input records are specified by an EOR or by a field count, these two bytes should be cleared to zero.

LOWUP This flag may be set non-zero to cause lower case characters to be sorted equivalent to upper case characters. Note that this is only valid if using the ASCII character set.

FROMKY This flag should be set non-zero to cause the data for the output records to come from the sort keys themselves.

ISPCOF This is the "input space compression off" flag. It may be set non-zero to turn off space compression when reading input records. This implies a binary type file.

OSPCOF This is the "output space compression off" flag. It may be set non-zero to turn off space compression when outputting records. This implies a binary type output file.

IDXTAG This byte signals the output file to be an indexed file or a tag file. If a tag file, the high order bit (bit 7) should be set. For indexed files, any other bit(s) should be set. If neither tag or indexed, this byte should be zero.

OUTALL This byte may be set non-zero to indicate that the entire input record should be sent to the output record. If this is the case, sort/merge will ignore any output key specs and the output file will be a re-arranged copy of the input file.

MSGLVL This byte may be set non-zero to suppress the printing of all run-time messages.

DELNUL This byte may be set non-zero to cause any records with null sort keys to be automatically deleted from the sort operation.

SLEXCL A non-zero value in this byte signals a select/exclude operation. The high order bit (bit 7) denotes select if cleared or exclude if set. The low order bits are set to 1, 2, or 3 to denote equal, greater than, or less than respectively.

SESPEC These four bytes hold the select/exclude key specification. It is a single, standard input key spec as described below under the ISPEC description.

just as before.

The next type of output key to describe is a string literal or hexadecimal value. This key must contain the string or a byte containing the hex value. Since the string can be any length, this type of key must also be of varying length. It is in fact of length N+2 where 'N' represents the number of characters in the string or is equal to one if a hexadecimal value. The format of this type key is as follows.

BYTE 1: Always equal to FD hex.
BYTE 2: Number of characters in string (1 if a hex value)
BYTES 3 thru (N+2): Actual string or hexadecimal value

The final type of output key spec is a tab. It is always two bytes in length and is as follows.

BYTE 1: Always equal to FC hex.
BYTE 2: Column number to which to tab (1 to 255).

SEDATA This space holds the actual string data for select/exclude comparison if that option is in use. The string may be up to 128 bytes in length and must start at location 0200 hex. If the string is not 128 bytes in length, the remainder of this area (up to 027F hex) should be filled with space characters or 20 hex.

One other necessary piece of data if the user is preparing his own parameter editor is the start address of SRTMRG.SYS to know where to begin execution once loaded. That start address is 0700 hex.

# 15.0 APPENDIX A - ASCII CHARACTER SET

| HEX | CHARACTER | NAME |
|---|---|---|
| 00 | CTRL @ | NUL |
| 01 | CTRL A | SOH |
| 02 | CTRL B | STX |
| 03 | CTRL C | ETX |
| 04 | CTRL D | EOT |
| 05 | CTRL E | ENQ |
| 06 | CTRL F | ACK |
| 07 | CTRL G | BEL,Bell |
| 08 | CTRL H | BS,Backspace |
| 09 | CTRL I | HT,Horizontal Tab |
| 0A | CTRL J | LF,Line Feed |
| 0B | CTRL K | VT,Vertical Tab |
| 0C | CTRL L | FF,Form Feed |
| 0D | CTRL M | CR, Carriage Return |
| 0E | CTRL N | SO |
| 0F | CTRL O | SI |
| 10 | CTRL P | DLE |
| 11 | CTRL Q | DC1 |
| 12 | CTRL R | DC2 |
| 13 | CTRL S | DC3 |
| 14 | CTRL T | DC4 |
| 15 | CTRL U | NAK |
| 16 | CTRL V | SYN |
| 17 | CTRL W | ETB |
| 18 | CTRL X | CAN, Cancel |
| 19 | CTRL Y | EM |
| 1A | CTRL Z | SUB |
| 1B | CTRL [ | ESC, Escape |
| 1C | CTRL \ | FS |
| 1D | CTRL ] | GS |
| 1E | CTRL ^ | RS |
| 1F | CTRL _ | US |
| 20 | | Space, Blank |
| 21 | ! | |
| 22 | " | |
| 23 | # | |
| 24 | $ | |
| 25 | % | |
| 26 | & | |
| 27 | ' | Apostrophe, Single Quote |
| 28 | ( | |
| 29 | ) | |
| 2A | * | |
| 2B | + | |
| 2C | , | Comma |
| 2D | - | Minus |
| 2E | . | Period |
| 2F | / | |

| HEX | CHARACTER | NAME |
|---|---|---|
| 60 | ` | Accent Grave |
| 61 | a | |
| 62 | b | |
| 63 | c | |
| 64 | d | |
| 65 | e | |
| 66 | f | |
| 67 | g | |
| 68 | h | |
| 69 | i | |
| 6A | j | |
| 6B | k | |
| 6C | l | |
| 6D | m | |
| 6E | n | |
| 6F | o | |
| 70 | p | |
| 71 | q | |
| 72 | r | |
| 73 | s | |
| 74 | t | |
| 75 | u | |
| 76 | v | |
| 77 | w | |
| 78 | x | |
| 79 | y | |
| 7A | z | |
| 7B | { | |
| 7C | \| | Vertical Slash |
| 7D | } | Alt Mode |
| 7E | ~ | (Alt Mode) |
| 7F | | DEL, Delete, Rubout |