Sort/Merge User's Manual

TABLE OF CONTENTS

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 the input keys described before. Besides allowing you to output portions of the input file, they allow tabbing to some specific column number on the line being output and allow you to insert some comment or characters into each output record.

Another option which is supported is that of an "alternate collating sequence file". Sorting is normally done according to the ASCII sequence of characters. ASCII stands for American Standard Code for Information interchange which is simply a widely used code or sequence of characters. See Appendix A for a chart of the ASCII sequence. If it is desirable in some instance to sort according to some other sequence of characters, that sequence may be supplied to the sort package in the form of a disk file containing the characters in the desired order. This alternate collating sequence file will then be used instead of ASCII for sorting purposes.

Other features of the sort/merge package include the ability to treat lower case characters equivalent to upper case characters for sorting purposes. This means that "Jones", "JONES", and "jones" would all be considered equal. It is possible to have the sort program automatically delete or ignore any records which have an entirely blank (all spaces) sort key, where "sort key" refers to all the input keys put together. There is also a "select/exclude" option which allows input records to be selected or excluded from the sort depending on their contents.

The sort/merge package is so called for two reasons. One is that it is capable of merging two, or more input files which have been previously sorted or are known to he in order. This is called a "merge-only" operation since no actual sorting is done and still requires most of the specifications which the sort requires such as input and output keys. The second reason for being called a sort/merge package is in the way sorting is sometimes done. The actual sorting is always done entirely within the computer's memory. The input records themselves are not sorted, but rather only the input keys. Each key has a pointer to where the input record it came from resides so that when the keys have been sorted, the input records can now be read back in the same order as the keys. This usually saves memory in that the sort key is almost always shorter than the input record. The program reads an input record into a temporary buffer and produces the sort key from the input keys specified. This sort key is then written into the available memory space. This process is repeated until enough sort keys have been written to fill the available memory space or until all records of the input file have been read. If the sort keys for the entire input file all fit in memory, the sort is carried out and the output file is written out directly. If there is not enough memory for the entire input file, however, the program sorts what did fit and then writes that sorted portion of the file out to a temporary work file on disk or "run" as we will often refer to it. Then the input and sort process is repeated on the next section of the input file which will fit in memory and another run is written out. This continues until the entire input file has been read. We now have several temporary work files each of which contains a sorted portion of the original file. The sort/merge package will now merge those work files or runs into one sorted output file and delete all the temporary files on completion. if there are a large number of runs, it may not be possible to merge them all at once. if this is the 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 he 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 parameters 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.

```
       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
```

There are several things which should be noted about this file. First of all, this is a list of names and phone numbers. Each line of the file should be considered one input record. Note that all records in this particular file are of the same length, namely 27 characters. Each last name begins in column 1 and nay run through column 10. Each first name begins in column 11 and may run through column 20. Each phone number is in columns 21 through 27. Most important, however, is the fact that this file is in no particular order!

3.1 SAMPLE SORT

Suppose we wished to sort the file NAMES.TXT into order according to last names. Let us go through an example of such a sort. At this point, it is not crucial to completely understand all the prompts and responses. Simply go through the steps as instructed and watch the results. More details will be given later. The first step is to insert the disk containing the file NAMES and the sort/merge package into a disk drive and type the command

```
      SORT NAMES
```

This tells the computer to load the SORT command and use it to sort the input file named NAMES.TXT. Note that all the sort/merge commands use TXT as a default extension for the input file. The computer should respond with

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

     OUTPUT TO DISK (Y OR N*)?
```

If we wanted the output to be put on a disk file, we would type a 'Y'. For this sample, type either an 'N' or simply hit a carriage return which will default to 'N'. This will cause the output to be sent to the terminal.

Next the computer will prompt

```
      INTERMEDIATE WORK FILE DRIVE?
```

This gives the user the ability to select the drive on which any necessary work files or "runs" will be placed. Since the file we are 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 all start in column one and end with column 10. Therefore, our input key specification would simply be "1-10". Type those 4 characters after the question mark. Note that we could use the default input key by typing a carriage return since the SORT routine will default to "A(1)1-10". The "A(1)" portion of this default specification is some added information about the input key which will be covered later.

When you have entered the "1-10" and hit a carriage return, another question mark ('?') prompt will be issued. This is to allow another input key to be entered if desired. It Is not required In our sample since we are sorting only on the last name, so type a carriage return to exit the input key prompting. The computer will respond with

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

At this point, you could specify that you only wanted certain portions of the input record sent to the output record. We, however, want the entire input record sent as is to the output record. There are several ways to specify this, but the easiest is to use the default by typing a carriage return.

Now we see the prompt

```
      FURTHER OPTIONS REQUIRED (Y OR N*)?
```

If a 'Y' is typed, SORT will begin prompting for several other possible input parameters. We will not be needing any of these somewhat specialized parameters, so type a 'N' or carriage return.

At this point, SORT will type

```
      === PARAMETERS ARE NOW SET ===

      SAVE PARAMETER FILE ON DISK (Y OR N*)?
```

This tells us that we have completed the setting up of the sort parameters. If desired, all these parameters may be saved in a disk file so that we could later repeat the sort without needing to retype them. For now, just type an 'N' or a carriage return since the parameters we are using are quite simple to enter.

Finally we see the prompt

```
      EXIT OR PROCEED WITH SORT (E OR S)?
```

This allows us to exit back to the disk operating system or to proceed with the sort as specified. Since we did not save our parameters in a file, we want to continue with the sort. Type an 'S' to do so. Note that there is no default value in this case. It is necessary to actually type an 'E' or an 'S'.

At this point, the program SRTMRG.SYS is loaded and the actual sorting begins. You should see the following information on your terminal.

```
=== 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
?
```

Note that both keys were specified on one line, separated by a comma. In all the examples given in this manual, commas will be used as delimiters, but if desired a space may always be used instead. It would also be possible to put the keys on more than one line as shown

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

When you have entered all the keys you wish, you can exit the key input mode (question mark prompts) by hitting a carriage return in response to the question mark.

If a sort is performed with these input keys and all other responses as before, you should see the following.

```
=== TSC SORT/MERGE ===

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
```

Notice that the Smiths are now in order by last AND first name. Experiment at this time with other combinations of multiple input key sorts. For example try sorting on phone number, then last name, then first name. Or perhaps on first name, then last name, then phone number.

Multiple input keys can be entered in any order. That is to say, the first key might come from 21-27 while the second comes from 1-10. It is also possible to use the same portion of the input record more than once. This does not often make sense, but you could, for example, specify a key like "1-8,30-40,1-10".

To this point, all our sorts have been in ascending order. That is 'A' to 'Z' or '0' to '9'. Sometimes you may wish to sort in a descending order. This is very easily accomplished through the manner in which the input keys are specified. Any input key may be prefaced with the letter 'A' for ascending or 'D' for descending. If neither is specified, the key is sorted in ascending order or in other words, 'A' is the default sorting order. That is what is meant by the 'A' in the input key prompt default. As an example, let's sort NAMES.TXT on the last and first name, but in DESCENDING order. All responses to the SORT command's prompts are as before (may be defaulted) except for the input key 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 parameters would yield the following output.

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

     SORT RUN 01 - 10 RECORDS
     ABBOTT
     CASWELL
     FREEMAN
     LYON
     PERRY
     SMITH
     SMITH
     SOUTH
     TYLER
     WILSON

     10 RECORDS SORTED
```

Now try sorting the NAMES file on any input keys you desire, but only output the first name or phone number. You can also specify more than one output key. You might try outputting the first name, last name, and then phone number. This would require a list of output keys like "11-20,1-10,21-27".

There is a special symbol which may be used in place of the ending column number in an output key. That is an asterisk ('*') and it represents the end of the input record. In other words, an output key of "11-*" would start outputting from column 11 and continue to the end of the record. For example, "11-20,1-10,21-*" would output first name, last name, and then phone number exactly as before. A "1-*" would mean to output the entire input record. This would have the same effect as defaulting to the entire input record.

It does not make sense to use an 'A' or 'D' (ascending or descending) as a preface to an output key, but there is another type of preface that can be used. That is an 'R' or 'L' which stand for Right or Left justify respectively. Any output key which is prefaced by one of these characters will be either right or left justified. This simply means that in the case of a right-hand justify any blanks or spaces at the end of the key will be moved to the front or in the case of a left-hand justify any leading blanks will be moved to the end of the key. An example will help clarify. Perform a sort on NAMES.TXT with an input key of "11-20" and respond to the output key prompt as shown

```
     ENTER OUTPUT KEYS.  DEFAULTS TO ENTIRE RECORD.
     ? R11-20
     ?
```

When the sort is initiated, you should see the following output.

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

    SORT RUN 01 - 10 RECORDS
            AMY
         ARNOLD
         HAROLD
          HENRY
          JERRY
            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 character 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.

```
     ENTER OUTPUT KEYS.  DEFAULTS TO ENTIRE RECORD.
     ? R11-20,$20,1-10,'    phone no. ',21-23,'-',24-*
```

This looks rather complicated, but if you examine each key one at a time it is really quite simple. We are printing the first name right justified, followed by a space (a hex 20), followed by the last name. Next we print a string which spaces over a few columns, prints "phone no.", and then spaces one more column. Next we output the first three characters of the phone number, a dash, and the rest of the phone number. The result is as follows.

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

     SORT RUN 01 - 10 RECORDS SORTED
            JIM ABBOTT        phone  no. 457-2091
            AMY CASWELL       phone  no. 925-8411
          LINDA FREEMAN       phone  no. 422-9105
        WILLIAM LYON          phone  no. 853-8227
         ARNOLD PERRY         phone  no. 422-6008
         HAROLD SMITH         phone  no. 422-7264
          JERRY SMITH         phone  no. 422-6002
       MARGARET SOUTH         phone  no. 422-1267
          HENRY TYLER         phone  no. 457-5573
            JIM WILSON        phone  no. 457-8339

     10 RECORDS SORTED
```

This should give you some idea of the endless possibilities for output records.

3.4 SORTING WITH FIELDS

Now let us examine another feature of the sort/merge package, "fields". Often it is desirable to divide a single input record into a number of distinct portions. We call these portions "fields". In our example with the file NAMES.TXT we had a last name, first name, and phone number but they were actually not separate fields since each item was separated from the others only by its columnar position in the record. To have fields, there must be some sort of field separator character to split the input record into distinct fields. There is another file on the disk called 'NAMES2.TXT', which does have fields. It looks like this.

```
        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=$OD)?
```

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
     or alternatively,

     FIELD SEPARATOR CHARACTER?
```

Enter one of these responses and continue as seen here

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

     ENTER INPUT KEYS.  DEFAULTS TO "A(1)1-10".
     ? (3)1-7
     ?

     ENTER OUTPUT KEYS.  DEFAULTS TO ENTIRE RECORD.
     ? (3)1-3,'-',(3)4-E,' ',(1),$2C,$20,(2)
     ?
```

There are a couple of new concepts presented in these output keys. First is the key "(3)4-E". The 'E' in place of the ending column number represents the end of the field much as an asterisk represents the end of the input record. Thus this output key says to start outputting from column 4 of field 3 and continue outputting until a field separator is found (without outputting the field separator). The second new concept is the way in which fields one and two have been specified. You will note that the field number is present in parenthesis but that no starting or ending columns are specified. This is a special case of an output key which means to output the entire field. Thus an output key of "(2)" would be equivalent to "(2)1-E". This shorthand notation often saves much typing.

The rest of the prompts may be defaulted as before. When the sort is initiated, you should receive the following output.

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

     SORT RUN 01 - 10 RECORDS
     422-1267  SOUTH, MARGARET
     422-6002  SMITH, JERRY
     422-6008  PERRY, ARNOLD
     422-7264  SMITH, HAROLD
     422-9105  FREEMAN, LINDA
     457-2091  ABBOTT, JIM
     457-5573  TYLER, HENRY
     457-8339  WILSON, JIM
     853-8227  LYON, WILLIAM
     925-8411  CASWELL, AMY

     10 RECORDS SORTED
```

To make another point, let's try sorting NAMES2 with respect to the last name. All prompts would be answered as before (set the field separator to a comma) until the input keys are requested. Now we know we want field number one since that is where the last name is, and we know we want to start the key in column one of that field, but what should be the ending column. It might seem natural to specify "(1)1-E" to use 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
PERRYIARNOLD,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.

3.5 SORTING WITH RIGHT OR LEFT JUSTIFICATION

Let us now reiterate somewhat on right and left justification. It should be obvious what right and left justification can be used for on output keys. It allows you to align columns of words or digits for a nice looking printout or to do simple formatting of an output file. However, the use of justification of input keys is not quite so obvious. Left justify on input keys will probably be rarely used. It would allow you to alphabetize some portion of a record which was not already justified, but this is probably a rare case. There is a good use for right justification of input keys. It relates to sorting numbers. Suppose we have a short file of dollar values that looks like this:

```
     8.75
     1.25
     62.00
     225.65
     1.00
```

If we sorted this file with an input key of "1-10", the output would be ordered as follows.

```
     1.00
     1.25
     225.65
     62.65
     8.75
```

This is due to the fact that SRTMRG.SYS compares keys one column at a time from the left. If you examine the first column only, you will find that they are in order. If, however, we sorted the file with the input key right justified such as "R1-10", the result would be the proper numerical order. The keys which SRTMRG.SYS would have built up would look like:

```
       8.75
       1.25
      62.00
     225.65
       1.00
```

You can see that if these keys are sorted column at a time from the left, the correct order will result since the spaces are considered lower in value than any of the digits.

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 individual 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
```

Answering thusly with an 'F' yields the prompt for the record length which should be answered as shown

```
      RECORD LENGTH? 27
```

The next prompt would be for the field separator character and would continue as before. The one problem with trying to sort the file as a fixed length file is that the carriage return is now part of the record. If you instruct the sort program to output the entire input record, either by defaulting or with an output key that includes all, the carriage return in the record will be output along with the carriage return which is always output at the end of an output record. This may not be the desired effect. There is a way around this problem, however. The carriage return character which is added to the end of every output record may be changed to any desired character or simply turned off. This is done in the special options section of SORT as described later. Thus we could turn off the carriage return added to the end of our output records and only have the one that is part of the input record.

One point should be noted about input records. Any input record which is a null record (ie. a variable length record containing only the end of record character) is ALWAYS automatically deleted from the sort/merge process. There will be no indication of any such deletions to the user.

4.2 SPECIFYING INPUT KEYS

The sort tutorial section discusses the use and specification of input keys. This section is meant to elaborate somewhat on that discussion, stating the general format and limitations of input in more detail. A general syntax of a single input key specification would appear like:

```
     <options>(<field #>)sss-eee
```

The "sss" represents the starting column number of the key within the field in use. If no field separator character has been defined, the entire input record is considered to be one field of number one. The "eee" represents the ending column of the key. These values must be between 1 and 250 inclusive. Note that "eee" may be smaller than "sss", in which case the key would be backward. In other words, the comparison of keys would take place one column at a time, beginning with column "eee" and continuing as necessary through column "sss". It is also permissible to have "sss" equal to "eee" in which case the input key would be only one character long. The starting and ending columns are ALWAYS required.

The <field #>, which must be enclosed in parenthesis as shown, is the number of the field from which the key should cone. Fields are numbered starting with one and there may be up to 64 fields in each record. This portion of the key specification is optional and will default to field number one if omitted.

The <options> specification represents a list of options which can 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.

The "OTHER" in the above prompt for output source refers to tag or indexed file output. A "tag" file is an output disk file which contains only the pointers to each key's parent input record. Each pointer is only three bytes, so we can maintain the necessary pointers for a sorted list in a very small space. This might be useful in the case where you wanted to keep several different ordered lists of a large file, but had limited disk space. These tag files do, however, require some external means of re-reading the input file in the order specified. More information on tag files is given in a later section.

Another possibility for output is an "indexed" file. This is a special type output which includes the entire sort key and the tag (pointer to input record). It is made use of by the TSC Indexed File Access package. There should be no need for the user to specify this type of output, but the capability is included for completeness.

4.4 SPECIFYING OUTPUT KEYS

Output key specification was covered in the preceding sections, but is repeated here in somewhat more detail. There are five types of output keys, each of which simply specifies some set of data to be sent to the output record.

1) The Full Key Spec ... This is the most commonly used output key and is very similar to the input key discussed earlier. It has the general form:

   ```
   <justify>(<field #>)sss-eee
   ```

   The "sss" represents the starting column of the key within the field in use. It is a number from 1 to 250 inclusive. If no field separator character has been defined, the entire input record (or entire sort key depending on what was selected as output source) is considered to be field number one. The "eee" represents the ending column of the key. It can be a number between 1 and 250 inclusive or may be one of two special characters. The first is an asterisk ('*') which represents the ending column of the current input record (or of the sort key). Thus "1-*" as an output key would output the entire record. The second special character is the letter "E". it represents the end of the current field. Thus an output key of "1-E" would output all of field number one (the default field). Unlike input keys, "sss" may not be larger than "eee". They may, however, be equal. The <field #>, which must be enclosed in parenthesis as shown, is the number of the field from which the key should come. This portion of the output key specification is optional and will default to field number one if omitted. The <justify> portion of the output key specification is a one character option to either right justify the particular key (specified by the letter "R") or left justify (specified by the letter "L"). This portion is optional and defaults to neither left or right justified.

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.

4.5 SPECIFYING OUTPUT DESTINATION

Sort/Merge routes the output to the CRT or terminal device by default. Output may, however, be routed to a disk file or to a line printer.

To output to a disk file requires specifying an output file name to one of the parameter specifying modules. This causes all actual output from the sort/merge to be written to the named file (excluding the run-time messages which still are still printed on the terminal). In this case there will be no echo of the output data on the terminal.

To output to the line printer, one must utilize the "P" command found in the FLEX operating system itself. Simply precede the command line you would have normally typed with the command "P". This will cause all actual output from the sort/merge to be printed on the device setup in the "P" command (again excluding the run-time messages which are still printed on the terminal).

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 then. 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 commend 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' .

OUTPUT TO DISK (Y OR N*)?

Normally, output of the sort package is sent to the terminal or to the line printer via the 'P' command in FLEX. This prompt allows routing the output to a disk file instead. If you want the output sent to the terminal or printer, type an 'N' or simply a carriage return. If you want a disk file produced, type a 'Y'.

If output is to be routed to the disk, the next prompt will be

FILENAME?

Type in a file specification for the file to be produced. The extension will default to TXT. It is possible to place the output on any disk drive by simply specifying the drive number in the normal FLEX manner. If no drive number is specified, the output file will be placed on the first available drive.

After specifying the output file name or if no output disk file is to be produced, we see the prompt

INTERMEDIATE WORK FILE DRIVE?

This gives the user the ability to select the drive on which any necessary work files or "runs" will be placed. Simply type a single digit drive number. If a carriage return is hit in response to this prompt instead of an actual drive number, the work files will be placed on the assigned system drive (see the FLEX user's manual). If no system drive is assigned, the first available drive will be used.

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

Here the user may specify variable length input records by typing a 'V' or by defaulting with a carriage return. Alternatively he may specify fixed length records by typing an 'F'.

If fixed length records are chosen, the next prompt will be

RECORD LENGTH?

The record length in characters should be entered as a decimal number greater than zero.

If variable length records are chosen, the following prompt would be issued instead

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

The response to this prompt tells sort how to terminate a variable length input record. There are two possible responses. The first is to specify an End-Of-Record character as a hex value preceded by a dollar-sign or as a printable ASCII character preceded by a single quote. Notice that the default is a carriage return or OD hex. The second possible response is a decimal field count. This tells sort to terminate an input record when the specified number of fields have been read.

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.

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 nay be terminated by hitting a carriage return. See section 4.2 for details of input key specifications.

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.

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.

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.

SAVE PARAMETER FILE ON DISK (Y OR N*)?
Typing a 'Y' will allow the parameters just specified to be saved as a disk file. A prompt will be issued for the filename to be used. The default extension is BIN. Any drive may be specified in the normal FLEX manner. The parameters are written to the file but also remain in memory. If no parameter file is needed, type an 'N' or a carriage return. In this case the parameters remain in memory only.

EXIT OR PROCEED WITH SORT (E OR S)?
At this point, the user has the option of exiting back to the operating system or continuing with the sort. Type an 'E' to exit or an 'S' to continue with the sort. Note that there is no default. Hitting a carriage return will result in the prompt being re-issued.

If continuing with the sort, the SRTMRG.SYS module will now be loaded and executed. The parameters needed by SRTMRG.SYS are resident in memory in the parameter file area. Upon completion of the sort, control is transferred back to FLEX.

5.2 ADDITIONAL PARAMETER OPTIONS

There are several options in the sort which are less frequently needed. Instead of always prompting for then, the sort parameter editor keeps them as a separate group of prompts which will only be issued if desired. They are accessed by typing a 'Y' in response to the prompt "FURTHER OPTIONS REQUIRED (Y OR N*)?" as described above. When the last of these additional prompts has been answered, the message "=== PARAMETERS ARE NOW SET ===" will be issued and control will continue as described before.

There is one of these additional options that requires further elaboration. It is the Select/Exclude option and is described in section 5.3. The other prompts and required responses are as follows.

MEMORY END IN HEX?

This prompt is issued in the 6800 version only. The 6809 version ALWAYS obtains the end of memory address from FLEX's MEMEND location. The 6800 version will use the FLEX MEMEND address unless the user responds to this prompt with an address. To allow sort/merge to use all of the available memory, simply hit a carriage return in response to this prompt. If for some reason a part of the upper end of the user memory should be saved, the user may enter the highest address which sort/merge should use. It should be entered as a four digit hexadecimal address. In the 6809 version (or the 6800 version) the user may limit the memory that sort/merge uses by altering the value of MEMEND before starting execution.

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

The TSC Sort/Merge Package is capable of sorting both text and binary files. In most cases the file to be sorted will be text and this may be specified by a 'T' or a carriage return. If the input file(s) is binary, type a 'B'. This essentially turns off space compression to allow the input file to be read as is.

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 description of the format of an alternate collating sequence file see section 12.0.

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 cone 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.

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 alluded 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 meaningful 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.

SELECT/EXCLUDE OPTION (Y OR N*)?

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.

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.

EOR CHARACTER FOR OUTPUT RECORDS (DEFAULT=$0D)?

The output key specifications allow the user to build up output records of any desired format. A built in function allows some end-of-record character to be appended to the data specified by the output keys. By default, this character is a carriage return (hex 0D). If a different character is desired, it may be specified here by typing a single quote followed by the ASCII character or by typing a dollar-sign followed by a two digit hex value for the character. If no end-of-record character is desired, type an 'N' for null (do not precede it with a single quote).

PRINT RUN-TIME MESSAGES (Y- OR N)?

The sort/merge package has several messages which it prints on the terminal during a sort operation to let the operator know just what is going on. These messages are explained in section 10.0. If the output is routed to a printer or to a disk file, these messages are not routed but rather are still printed on the terminal. To suppress these messages, type an 'N'.

At this point, the message "=== PARAMETERS ARE NOW SET ===" and control will continue from this point as described in section 5.1.

5.3 SELECT/EXCLUDE OPTION

It is possible to have the sort package select only certain input records for the sort or to exclude certain input records. This is done by specifying a "select/exclude key". If this key matches the correct portion of the input record then that record will be selected or excluded. It is also possible to require that the key be greater than or less than the correct portion of the input record in order to be selected or excluded. This may sound a little confusing but can be cleared up by a couple of examples.

Suppose we wish to sort our file NAMES.TXT according to last name and then first name but only want those people whose phone numbers have "422" as the exchange number (first 3 digits) to be in the output. In other words, we want to "select" the input records when the exchange is equal to 422. The sort parameters should be set up exactly as before with an input key of "1-20" to cause the sort to be done on last and first names. When the prompt "FURTHER OPTIONS REQUIRED (Y OR N*)?" is received, type a 'Y' so that we will receive the select/exclude option prompts. The prompts received may all be answered as desired or simply answered with a carriage return until you see the prompt:

```
     SELECT/EXCLUDE OPTION (Y OR N-)?
```

You should respond with a 'Y' which will cause the following prompt to be issued:

```
     SELECT/EXCLUDE KEY SPEC?
```

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 records with an exchange of greater than 199 or excluding those which are less than 200. Then sort the output of this first sort selecting all records less than 601 or excluding all records greater than 600. We could even go a step further and exclude from this list all records which have a last name of "SMITH".

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 he 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, 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
      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 <parameter 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 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 for the 6800 version or $0000 for the 6809 version. 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 command is for a sort operation, while a parameter file produced by a MERGE command is for a merge-only operation. If any of these conditions are not met, an error message will be issued and the sort operation will be aborted.

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 bits 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.

C=<decimal>

This is the field count specifier for use when an input record is to be specified by a fixed number of fields. <decimal> represents the decimal number of fields desired.

F=<hex or ASCII>

This is the field separator character specification. <hex or ASCII> represents a hex character value preceded by a dollar-sign or an ASCII character preceded by a single quote. If this specification is left out, the field separator character will be a null, effectively disabling fielding.

T or B

These are the input file type specifiers. By simply typing a 'T' as an input specification, the user can indicate that the input file is a text type file and thus it is taken to have space compression. Typing a 'B' implies a binary type input file and space compression is turned off before attempting to read from the file. If neither of these characters are entered on the command line, the input file is assumed to be a text type file.

U

If this character is entered on the command line, sort/merge will treat lower case characters equivalent to upper case characters so long as the ASCII character set is in use.

K

This character instructs the sort/merge package to "Keep" all records which have blank sort keys. Recall that normally these type records are deleted from the sort process. This specification turns off that deletion process.

STANDARD INPUT KEYS

Any standard input key may be included in the input specifications. For details on a standard input key specification see section 4.2.

A couple of comments on these input specifications are in order. First, the user may only specify one of 'L', 'E', or 'C' on a single command line. These specify when to terminate an input record and there can be only one method in use. If none of these three are given, CSORT will default to a carriage return (OD hex) as an End of Record character. Also note that if a field count is specified (C=<decima]>), then a field separator character must also be specified. If not an error message will result.

The input specifications may appear in any order but remember that the order of the input keys is significant. If there are conflicting specifications (ie. a 'T' followed later by a 'B' or a 'W=1' followed later by a 'W=O'), the last one on the command line is used and the first is ignored.

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=1'.

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, and the select/exclude option is not allowed. If the user needs these features, he has no alternative but to use the SORT and PSORT commands.

A couple of sample CSORT command lines should help clarify its use.

1) The very first example of section 3.1 could be duplicated with the following command:

```
      CSORT NAMES
```

This uses defaults for all parameters much as was done by hitting carriage returns in section 3.1. Input keys default to "A(1)1-10" and the output defaults to outputting the entire input record as is.

2) The last example of section 3.3 could be duplicated with the command:

```
      CSORT NAMES +1-20+R11-20,$20,1-10,'   phone no. ',21-23,'-',24-*
```

Here we have specified an input key of "1-20" and several output keys as discussed in section 3.3.

3) The first example of section 3.4 could be duplicated by:

```
      CSORT NAMES2 +F=$2C,(3)1-7+(3)1-3,'-',(3)4-E,' ',(1),$2C,$20,(2)
```

Here, we set the field separator to hex 2C (a comma) before giving the input key spec.

4) Following are some valid CSORT command lines for some fictional input files. They are simply to show how to place the input and output specs and output file name on the command line.

```
      CSORT JUNK,JUNK.OUT,+E=$02,F=$0D,10-15
      CSORT DATA ++10-20,1-9
      CSORT TEST.DAT,TEST,+100-115+O=K,M,1-10
      CSORT PHONE+U,K,31-40,1-10,21-30,+M
```

One point that should be noticed from the above samples is that the plus signs which separate input and output specs may or may not be preceded by a delimiter (space or comma) as desired by the user.

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, 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 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 for the 6800 version or $0000 for the 6809 version. 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 ===
   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.

4) xx RECORDS SORTED
   This message is printed at the completion of a sort operation. "xx" represents the total number of records that have been sorted and sent to the output. Note that this may not be equal to the number of records in the input file due to deleted records and excluded records.

5) xx RECORDS MERGED
   This message is printed at the completion of a merge-only operation. "xx" represents the total number of records that have been merged to form the output file. Note that this may not be equal to the total number of records in the input files due to deleted records.

6) xx RECORDS DELETED
   This message is printed at the completion of a sort or merge-only operation. "xx" represents the number of records which were deleted from the process due to a blank sort key. This message is only printed if there were records deleted.

7) xx RECORDS EXCLUDED
   This message is printed at the completion of a sort operation if any records were excluded (or not selected) through use of the Select/Exclude option. If the Exclude option was set, "xx" represents the number of records which were excluded during the sort. If the Select option was set, "xx" represents the number of records which were NOT SELECTED or effectively excluded.

8) KEY PADDING REQUIRED
   This message is printed at the completion of a sort or merge-only operation if key padding was required. This does not denote an error condition but simply informs the user that the padding was required. If this should not have occurred, it is up to the user to remedy the situation.

There is one run-time message which is in a different class than those described above. It is as follows:

```
 DELETE OUTPUT BACKUP FILE (Y OR N)?
```

This prompt is issued if an output file was specified and there already existed one file by that name and extension and another by the same name with a 'BAK' extension. The prompt is for permission to delete the 'BAK' extensioned file so that the existing file with the same name and extension can be converted to a backup (BAK) file, thus allowing the output file to be written. If necessary, this message will ALWAYS be printed regardless of whether the option to list run-time messages is on or off. As in the above messages, it will always be sent to the console and not to an output file or line printer.

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.

4) TOO MANY KEYS. RE-ENTER.
   There is a 256 byte buffer reserved for key information. Each input key requires 4 bytes and except for literal strings each output key requires 3 bytes. The number of input keys is limited to 20 while the number of output keys is limited by the amount of space left in the buffer. Literal strings as output keys are stored in the key buffer and thus long strings can rapidly eat up the available space. The 256 bytes should be sufficient, however, for most any application. If this message is issued, any existing keys will be discarded and a new prompt issued.

5) FILE EXISTS. DELETE ORIGINAL (Y OR N*)?
   This is not exactly an error message, but is listed here for the lack of a better place. It is issued when the user attempts to save a parameter file with a filename that is already in use. If the existing file should be deleted and replaced by the new one, type a 'Y'. If the existing file should not be deleted, type an 'N' or a carriage return. A prompt will be issued for a new filename under which to save the parameter file.

6) 'SRTMRG.SYS' NOT PRESENT ON DISK
   Once the parameters have been finalized, SORT or MERGE attempts to load the actual sort/merge program called 'SRTMRG.SYS' from the same disk which SORT or MERGE was obtained. If not found, this message is issued and the program is aborted, returning control to FLEX.

There are two additional messages which are only found in MERGE. They come about due to the fact that MERGE reads through the input file list while SORT does not (SORT leaves that checking up to SRTMRG.SYS). These messages are as follows:

7) ILLEGAL INPUT FILE SPECIFICATION
   One (or more) of the input files specifications is in error. This error will cause the program to abort, returning control to FLEX.

8) INPUT FILES NOT ON SAME DISK
   All input files must reside on the same disk. If files from multiple disks were specified, this message is issued and the program is aborted, returning control back to FLEX.

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 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 permissible.

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.

11.3 ERROR MESSAGES FROM CSORT

Errors in CSORT are generally of a fatal nature in that they cause execution of the sort to be terminated and return control to FLEX.

1) ILLEGAL INPUT FILE SPECIFICATION
   The input file specification is invalid. Refer to the FLEX manual.

2) ILLEGAL OUTPUT FILE SPECIFICATION
   The output file specification is invalid. Refer to the FLEX manual.

3) SPECS MUST START WITH PLUS SIGN
   This message is issued if CSORT finds something on the command line after the file specs which is not preceded by a plus sign. The plus sign is required as seen in section 7.0.

4) UNRECOGNIZABLE INPUT SPEC
   CSORT has found an unrecognizable input specification. In other words, CSORT was looking for an input specification such as an input key, the work drive spec, or a record terminator spec, but instead found some unrecognizable item. Note that a recognizable input spec which is in error will not produce this message but rather a more specific message as seen later.

5) UNRECOGNIZABLE OUTPUT SPEC
   CSORT has found an unrecognizable output specification. In other words, CSORT was looking for an output specification such as an output key, the output source, or an end of output record character spec, but instead found some unrecognizable item. Note that a recognizable output spec which is in error will not produce this message but rather a more specific message as seen later.

6) INPUT KEY ERROR
   An error was found in one of the input keys specified.

7) ILLEGAL WORK DRIVE SPECIFIED
   The work drive specified was not in the range of 0 to 3.

8) ILLEGAL RECORD LENGTH
   An invalid number was given as the record length. The value must be decimal and must not be zero.

9) ILLEGAL END OF RECORD CHARACTER
   An invalid end of record character was specified. The EOR character must be specified as a hex value preceded by a dollar-sign or an ASCII character preceded by a single quote.

10) ILLEGAL FIELD COUNT
    The field count specified is invalid, it must be a decimal number between I 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.

20) 'SRTMRG.SYS' NOT PRESENT ON DISK
CSORT attempts to load the file SRTMRG.SYS from the same disk from which CSORT.CMD was obtained. This message is issued if not found.

11.4 ERROR MESSAGES FROM SRTMRG.SYS

These errors are generally fatal causing execution 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) ERROR WITH INPUT FILE ...
This is a two line message much like the previous one, but describes a standard FLEX error which occurred with one of the input files.

3) INSUFFICIENT MEMORY
The TSC Sort/Merge Package requires a minimum of about 8K of user memory. If this message is issued, first check to insure the memory end limit set in SORT or MERGE was not inadvertently specified as lower than 8K. If this is not the case, you will simply need more memory in your system.

4) TOO MANY FILES OR INSUFFICIENT MEMORY
This message can only be received during a merge-only operation. It can be caused by two things. The first is insufficient memory exactly as in the preceding error message. The second is that an attempt has been made to merge too many files. Note that in a merge-only operation, all merging must be done in one pass. The merge operation must have an open FCB (see the FLEX manual) for each file being merged. Thus if many files are being merged and there is not much user memory available, there may not be enough room in memory for all the FCB'S. There are two solutions to this problem, either add more memory to the system or instead of a single large merge, merge the files in several smaller groups which can then be merged.

5) ILLEGAL FILE SPECIFICATION
This message is issued if an invalid input file specification is found.

6) INPUT FILES NOT ON SAME DISK
Sort/Merge requires that all input files come from the same disk. This message is issued if multiple files are specified from different disks.

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) and the second is that the file did not have a valid load address. The alternate collating sequence file should load at $0600 for the 6800 version or $0000 for the 6809 version.

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 in 6800 or $0000 in 6809.
3) Must be 2 sectors long.

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                              (Use $0000 for 6809!!)
 FCC / ABCDEFGHIJKLMNOPQRSTUVWXYZ/
 FCC /abcdefghijklmnopqrstuvwxyz/
 FCC /0123456789/
 FCC /!"#$%&'()*+/
 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

In the 6800 version the actual sorting and merging program, SRTMRG.SYS, expects to find all the necessary parameters in memory between hex 00C0 and 027F inclusive. The 6809 version is relocatable, so it places all the parameters on the user stack (relative to the "U" stack pointer). As such, there is no absolute location for the parameters. Instead, the parameters are stored beginning at an offset of 14 from the U register which should be equal to MEMEND-$640.

It is the responsibility of the commands SORT, PSORT, CSORT, MERGE, and PMERGE to set up the necessary parameters in these areas. Following are maps of these areas and then further descriptions of each parameter.

6800 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 input
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
00F1-00FD		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

The following map is for the 6809 version of sort/merge. The addresses given in the map are NOT absolute addresses, but rather offsets into the parameter file area no matter where it may be located. Thus we can see that the End-Of-Record byte for input (EOR) is the 13th byte in the parameter area no matter where the parameter file is placed.

The parameter file should always be placed at a fixed offset of +14 (0E hex) from the value in the "U" register. Furthermore, the "U" register should be assigned a value of 640 hex bytes less than the memory end address (U=MEMEND-$640). When using any of the supplied programs (SORT, PSORT, CSORT, MERGE, PMERGE, or SRTMRG.SYS), the placement of the parameter file at this location relative to "U" is done automatically.

6809 PARAMETER FILE AREA MAP

Location	Name	Description
0000	OUTDRV	Output file drive number
0001-000B	OUTNAM	Output file name
000C	ALTDRV	Alternate sequence file drive number
000D-0017	ALTSEQ	Alternate sequence file name
0018	RUNDRV	Run or temporary work file drive number
0019-001A	MEMEND	Not used in 6809 version
001B-001C	OSPEC	Output key specs pointer
001D	EOR	End Of Record character for input
001E	EOF	End Of Field character
001F	EOOR	End Of Output Record character
0020	GROUP	Field count or group count
0021-0022	RLNGTH	Fixed length record length
0023	LOWUP	Treat lower case equal to upper flag
0024	FROMKY	Output from key designator
0025	ISPCOF	Turn off input space compression flag
0026	OSPCOF	Turn off output space compression flag
0027	IDXTAG	Indexed or Tag file output flag
0028	OUTALL	Output entire input record flag
0029	MSGLVL	Run-time message level
002A	DELNUL	Delete records w/ null keys flag
002B	SLEXCL	Select/Exclude information
002C-0030	SESPEC	Select/Exclude key spec
0031-003D		Reserved for future use
003E	MGONLY	Merge-only flag
003F	RTSFLG	Return flag
0040-013F	ISPEC	Input and output specs
0140-01BF	SEDATA	Select/Exclude string data

The following descriptions give the necessary details on what each parameter represents and what values it may take on.

OUTDRV	This specifies the drive onto which the output file will be placed (if one is called for). It should be a valid drive number from 0 to 3.
OUTNAM	These 11 bytes hold the output file name and extension. The name should be in the first 8 bytes and the extension in the last 3 bytes. If the name or extension does not fill out its respective field, zeros should be used to fill in. If there is to be no output file, the first byte of this parameter must be equal to zero.
ALTDRV	This parameter holds the alternate collating sequence file drive number. It may take on the same values, as the output drive number, namely 0 to 3.
ALTSEQ	These 11 bytes are for the alternate collating sequence file name and extension. The format should be exactly as that of the output file name above. If there is to be no alternate collating sequence, the first byte must be equal to zero.
RUNDRV	This byte holds the drive number onto which the temporary work files or runs will be written. It may be set to a number from 0 to 3 or to hex FF in which case the assigned system drive will be used or if unassigned, the first ready drive.
MEMEND	If desired in the 6800 version an upper memory limit may be specified in these two bytes. No memory above this address will be touched. If these two bytes are set to zero, the upper memory limit of FLEX will be used. The 6809 version ignores these two bytes and ALWAYS uses FLEX's MEMEND value.
OSPEC	In the 6800 version these 2 bytes hold the absolute address of the first byte of the output key specifications. In the 6809 version they hold an offset value from the beginning of the input key specifications to the first byte of the output key specifications. The output key specifications should be placed directly after the input specs.
EOR	This is the End of Record character for input records. If input records are to be fixed length or if a field count is specified, this byte should be set to zero. Note that hex 00 may not be used as an actual EOR character.
EOF	This is the End of Field character used to separate input records into fields. If no fields are desired, this byte should be cleared to zero. This implies that 00 cannot be a true field character. This byte must be non-zero if a field or group count is set.

EOOR	This is the End of Output Record character which is added onto the end of every output record. One exception is when this byte is zero, no EOOR character is added to the output records.
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 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.

MGONLY This flag may be set non-zero to signal a merge-only operation. If set, the entire sort operation will be skipped and the input files named will be merged.

RTSFLG This RTS flag allows sort/merge to be called as a subroutine. If set to some non-zero value, sort/merge will save the stack pointer on startup and upon completion will restore this stack pointer and execute a return (RTS). If RTSFLG is cleared, sort/merge will jump to FLEX warm start upon completion.

ISPEC This is a 256 byte area in which all input and output key specifications are stored. The input keys are stored first followed by the output keys. The OSPEC value above points to the address in this area where the output keys start. Since the format of the input and output key specs are different, we will describe them separately.

INPUT KEY SPEC FORMAT

Each input key requires exactly four bytes of storage. The keys are stored adjacent in memory starting at 0100 hex. The input keys MUST be terminated by a zero byte immediately following the last input key. This appears to sort/merge like a key with zero length thereby terminating the fetching of input keys. The four bytes contain the following information.

```
BYTE 1: Length of key in bytes  (1-250)
BYTE 2: High order bit (bit 7) =  0 if forward
                               =  1 if backward
        Low order bit (bit 0)  =  0 if ascending
                               =  1 if descending
BYTE 3: High order bit (bit 7) =  1 if left justified
        Next high bit  (bit 6) =  1 if right justified
        Lower bits (bits 5-0)  =  (field number)-1
BYTE 4: Starting column number of key
```

OUTPUT KEY SPEC FORMAT

Output keys can be of variable lengths. The most common is three bytes in length and is essentially like bytes 1, 3, and 4 of an input key spec. The output keys immediately follow the zero byte after the input keys. OSPEC should be set to point to the first byte of the output key specs. As with the input specs, the output key specs are terminated by a following zero byte which looks to sort/merge like a key length of zero. A normal, three byte output key spec is setup as follows.

```
BYTE 1: Number of columns in key (1-250 decimal)
BYTE 2: High order bit (bit 7) = 1 if left justified
        Next high bit  (bit 6) = 1 if right justified
        Lower bits (bits 5-0) = (field number)-1
BYTE 3: Starting column number of key
```

The next type of key is very much like the one just shown but specifies outputting to the end of the field or to the end of the record. This type key is always 3 bytes and is signaled by the first byte being FF hex for outputting to end of record or FE hex for outputting to end of field. Bytes 2 and 3 are 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. If the string is not 128 bytes in length, the remainder of this area 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. In the 6800 version that address is 700 hex. In the 6809 version, execution should be started at whatever location SRTMRG.SYS is loaded into. As distributed, this will default to location 0000.

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
IE	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
30	0	Number zero
31	1	Number one
32	2	
33	3	
34	4	
35	5	
36	6	
37	7	
38	8	
39	9	
3A	:	
3B	;	
3C	<	Less Than
3D	=	
3E	>	Greater Than
3F	?	
40	@	At Sign
41	A	
42	B	
43	C	
44	D	
45	E	
46	F	
47	G	
48	H	
49	I	Letter I
4A	J	
4B	K	
4C	L	
4D	M	
4E	N	
4F	O	Letter O
50	p	
51	Q	
52	R	
53	S	
54	T	
55	U	
56	v	
57	W	
58	X	
59	y	
5A	Z	
5B	[	
5C	\	
5D	]	
5E	^	Up Arrow
5F	_	Underscore

HEX	CHARACTER	NAME
60	`	Accent Grave
61	a	
62	b	
63	c	
64	d	
65	e	
66	f	
67	9	
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