ISTITUTO NAZIONALE DI FISICA NUCLEARE

Sezione di Trieste

INFN/TC-91/04

27 Marzo 1991

Claudio Strizzolo

SEV (VERSION 1.00) USER GUIDE

Servizio Documentazione
dei Laboratori Nazionali di Frascati

SEV (Version 1.00) User's Guide

(SEV is a tool to select events)

Claudio Strizzolo
I.N.F.N. Sezione di Trieste, Italy

Abstract

SEV is a program (written using Vax Fortran) for Vax/VMS, to create an ".EPIO", ".NATIVE" or ".EDIR" file from a list of run-number, event-number, relative to ALEPH data. The program extracts the events from the data files containing them (looking sequentially into EPIO or Vax NATIVE format files, or performing direct access by means of EDIR files, if available) allowing to build a single file (in EPIO or NATIVE format) enclosing all of them, and the ".EDIR" ("event directory") file referring to the newly created data file, or just the ".EDIR" file, pointing to the original input files, too. An ".EDIR" file allows direct reference to the events contained into ALEPH data files (either EPIO or NATIVE, depending on which format is used into the data file relative to each ".EDIR" file).

Even if it has been studied to work on CERN data structure, it can be easily adapted to different sites, or to personal needs.

Contents

1 Introduction

SEV has been written to allow an easy selection and extraction of a group of events from the ALEPH data files containing them. The general structure of the program has been derived from the one used for HEM tool, but these two tools are rather different, anyway: HEM allows to create an EDIR file relative to a list of run-number, event-number, getting as input files only the original EDIR files; it is generally faster than SEV when you need to do only this kind of operation (HEM doesn't run any ALPHA job, while SEV runs an ALPHA job to perform the selection), and it is simpler to use, while SEV allows a wider variety of possibilities to select and also extract events to an EPIO or NATIVE output file.

In fact, SEV allows to build easily an EPIO or Vax NATIVE file containing the events you need, and it automatically produces the EDIR file relative to the data file that it has generated; you can also create just an output EDIR file relative to the input data files. An internal option allows to choose if you want to perform the selection of events by running a batch job or if you prefer to run it in interactive mode. Submitting a batch job is useful to save time, especially if you are considering a great number of events.

Adapting SEV to different data organizations is very easy to do, even if it has been thought to be used on the collection of data that they keep at CERN; in fact it reads parameters about data organization from an external file, that is very easy to modify.

SEV has been built to work very easily on the data files containing runs; it can be used also on Monte-Carlo files, even if it requires some extra operations when working on this kind of files. So, if you want to select events from Monte-Carlo files, read the chapters regarding running SEV on normal run files, and then see chapter 5 for more information about running on Monte-Carlo files.

2 Installing and getting started

Installing SEV is very easy to perform; normally, after you have copied all the files relative to SEV into a directory on your site, you just need to create a SEVDFT.LOCAL file as explained in chapter 3. The complete list of the files composing SEV package is available in table 1.

If you realize that the version of ALPHA used to link the ALPHA job run by SEV is not the one you currently use on your site, you probably will have to re-compile and re-link the ALPHA source program (SEV_MAKEIT.FOR). You could do it by yourself, or contact the author for more information.

SEV needs some logical names and symbols to be defined to work correctly. A command procedure named SEV_SETUP.COM performs some of these definitions. Please execute it at login time or before running SEV, only once per login

Filename		*Purpose*
SEV.EXE		SEV image file
SEV_SETUP.COM		Settings for SEV
SEV_MAKEIT.EXE		ALPHA image run by SEV
SEV.FOR	*(optional)*	SEV program source file
SEV_MAKEIT.FOR	*(optional)*	Source file of ALPHA job run by SEV
SEV.HLP	*(optional)*	Help file
SEV.README	*(optional)*	List of files composing SEV package
SEV.SOURCE	*(optional)*	Short description of functions and variables used in SEV.FOR
SEV.TEX	*(optional)*	This SEV User's Guide in LaTeX
SEVDFT.CERN	*(optional)*	Parameters-setting file for CERN site (Example of SEVDFT.LOCAL file, see section 3.14)
SEVDFT.TRIESTE	*(optional)*	Parameters-setting file once used in Trieste (Example of SEVDFT.LOCAL file, see section 3.14)

Table 1: List of SEV files

session. In particular, this procedure defines the logical name SEV_DIR pointing to the directory where SEV files are stored, and the symbol SEV as "RUN SEV_DIR:SEV.EXE" to run SEV from anywhere. To execute SEV_SETUP please put the following line into your login file or type it directly when needed

```
$ @path_where_sev_is:SEV_SETUP
```

Some logical names should be defined anyway, as they are not assigned by SEV_SETUP.COM procedure. They refer to the location of ALEPH data files on your site: AL$DATA refers to the place where EPIO or NATIVE files are stored, AL$EDIR refers to the site where EDIR files (relative to EPIO or NATIVE file) are stored. These two logical names are used on ALWS and also on several sites where ALEPH data are used, so they have been taken as a standard rule.

Optionally, to complete your installation, you could add the help file named SEV.HLP to your help libraries.

3 Working on different sites

SEV has been studied to respond to different requests in data organization, as there are different ways to collect ALEPH's data depending on the different sites in which they are used. For instance, at CERN there is a ".EPIO" or ".NATIVE" file for each run, while, in other sites (such as, for instance, we used to do on our cluster, in Trieste, some time ago), they prefer to group data in larger files containing several runs each. The ways to reach a desired event are different.

To adapt SEV to the characteristics of your site, you have to create a parameters-setting file named SEV_DIR:SEVDFT.LOCAL following some rules, to assign correct values to a group of parameters. These parameters vary depending on your organization rules; in particular, it is important to know if you keep runs unpacked (that is, each file contains a single run, such as at CERN) or if you keep them grouped (packed) in larger files containing several runs each.

The valid parameters you can set into the parameters-setting file are explained in the following sections.

3.1 GENFIL parameter

This parameter defines the name of a generic file containing *a single run.* It will be used in particular if you don't use any packed file or if, even using packed files, you have some unpacked runs. The names of the files containing a single run are directly relative to the run numbers, as each of them contains the number of the run into its name. CERN files containing runs 5517, 5654 and 5660, for instance, have been named D0005517.EPIO, D0005654.EPIO, D0005660.EPIO. Our files in Trieste containing the same runs have been named T0005517.NATIVE, T0005654.NATIVE and T0005660.NATIVE (the format of data files, EPIO or NATIVE, is not fundamental now; it can be specified in a different way: now only the names of the files should be considered). So we can establish a general rule: CERN DST file names are in the form

D%%%%%%%

in which the "%" stay for the run number, a "%" for each digit of the number. Our filenames in Trieste are in the form

T%%%%%%%

and so on for every site. The way to specify this situation to SEV by using the GENFIL parameter is to put a line such as the following into your parameters-setting file (example is about CERN organization)

```
GENFIL D%%%%%%%
```

You might want to establish a "search list", so that SEV will look for each run into a group of categories of files, sequentially, instead of a single category; for instance, at CERN there are different kinds of data files, such as DST, POT, and so on. You could specify that, if a run is not available in a DST file, SEV must look for the corresponding POT file instead of abandoning the research. The way you can do it is to write several GENFIL assignments into the parameters-setting file, one for each category of files. You must write them in the order you want them to be scanned by SEV. For instance, as POT filenames at CERN are in the form P%%%%%%%.EPIO, you may specify

```
GENFIL D%%%%%%%
GENFIL P%%%%%%%
```

so that SEV will look for a DST file named as the run you selected, and, if it doesn't exist, it will look for a POT file instead of skipping it immediately. Only if the POT file doesn't exist too, the run will be ignored.

Remember that:

- Multiple definitions on the same line are not allowed. So the following assignment would be ignored

  ```
  GENFIL D%%%%%%%, P%%%%%%%
  ```

- You can write up to 20 GENFIL definitions in a parameters-setting file (search list).
- The extension of the file will be set to the format that you specify, on keyboard, when you are asked to supply the input data format (read chapter 7). So if you specify a format into GENFIL parameter lines, such as, for instance, typing

  ```
  GENFIL D%%%%%%%.EPIO
  ```

  the format specification ".EPIO" will be ignored, and only the general form of the filename (D%%%%%%%) will be considered.

- You must set at least one GENFIL value and/or a value for the TABLE parameter (if you set the TABLE parameter, you must also set the relative RUNCOL, RUNLEN, FILCOL, FILLEN parameters, see section 3.2), depending on your data organization; SEV will quit if both GENFIL and TABLE parameters are undefined.

3.2 TABLE parameter

This parameter defines the name of a table file containing relationships between the run numbers and the names of the files containing them.

You should assign a value to this parameter only if you use packed files or if there is not a direct relationship between the numbers of the runs and the names of the files containing them. This parameter is generally set when you use Monte-Carlo files too, see chapter 5 for more information.

The table file must have at least two columns, the first containing the run numbers, the second the name of the file containing each run. If there are other

```
I col.      XI col.
↓           ↓

Run      *   File    *    Date     *  Ecms
---------------------------------------------
0006822 * GT000035 * 29-JUN-1990 * 91.220
0006824 * GT000035 * 29-JUN-1990 * 91.220
0006930 * Unpacked *             * 91.220
0006931 * GT000001 * 21-JUN-1990 * 91.220

                  (...)
```

Figure 1: Example of *run numbers - filenames* relationship table

columns, they will be ignored. For instance, the table file we used in Trieste when we used to keep grouped runs was structured as showed in figure 1.

If you write the word "UNPACKED" (either in upper or lower case) instead of the file name, the run will be looked for into a file whose name is formatted as you specified when defining the GENFIL parameter of the parameters-setting file (if any, else it will be ignored). If several masks have been set for GENFIL (see section 3.1), all the search list will be scanned until an existent file is found, otherwise the events belonging to that run will be ignored. For instance, using in Trieste the table showed in figure 1, unpacked run 6930 would be looked for into a file named T0006930, while packed run 6931 would be looked for into a file named GT000001.

Every line of the table which does not contain a valid run number will be ignored. Every input run which is not included into the table will be treated as if it was "UNPACKED", so the same rules as for unpacked files will be followed (see above).

To specify a table like this as a valid parameter, write a line such as

TABLE *name_of_your_table_file*

into the parameters-setting file. For instance

```
TABLE MY$ROOT:MY_TABLE.OFRUNS
```

Remember that:

- Only one TABLE specification is allowed into a parameters-setting file; if more are specified, only the first one will be considered as a good value, while the other ones will be ignored.

- The table must be sorted by ascending run number.

- If you set a definition for the TABLE parameter, you must set also a value for the following four parameters: RUNCOL, FILCOL, RUNLEN, FILLEN, otherwise an error will occur and SEV will quit.

- You must set at least one GENFIL value and/or a value for the TABLE parameter, depending on your data organization; SEV will quit if both GENFIL and TABLE parameters are undefined.

3.3 RUNCOL parameter

This parameter defines the column of the table file, specified as TABLE parameter, where you can find the number of the run. For instance, in the example table showed in figure 1, this parameter must be set to 1, because run numbers begin at the first column of the table (that is, at the first character from the left border of the table), so you should specify something like

```
RUNCOL 1
```

Remember that:

- RUNCOL value will be used only if TABLE parameter is set.

- Only one RUNCOL specification is allowed into a parameters-setting file; if more are specified, only the first one will be considered as a good value, while the other ones will be ignored.

3.4 RUNLEN parameter

This parameter defines the number of characters to be considered to obtain a complete run number (starting from the column specified as RUNCOL parameter). For instance, in the example table showed in figure 1, this parameter should be set to 7, specifying something like

```
RUNLEN 7
```

Remember that:

- RUNLEN value will be used only if TABLE parameter is set.

- Only one RUNLEN specification is allowed into a parameters-setting file; if more are specified, only the first one will be considered as a good value, while the other ones will be ignored.

3.5 FILCOL parameter

This parameter defines the column of the table file, specified as TABLE parameter, where you can find the name of the packed file containing a specified run. For instance, in the example table showed in figure 1, this parameter should be set to 11, because the names of the files begin at the 11th character from the left border of the table, so you should specify something like

```
FILCOL 11
```

Remember that:

- FILCOL value will be used only if TABLE parameter is set.
- Only one FILCOL specification is allowed into a parameters-setting file; if more are specified, only the first one will be considered as a good value, while the other ones will be ignored.

3.6 FILLEN parameter

This parameter defines the number of characters to be considered to obtain a complete packed file name (starting from the column specified as FILCOL parameter). For instance, in the example table showed in figure 1, this parameter should be set to 8, specifying something like

```
FILLEN 8
```

Remember that:

- FILLEN value will be used only if TABLE parameter is set.
- Only one FILLEN specification is allowed into a parameters-setting file; if more are specified, only the first one will be considered as a good value, while the other ones will be ignored.

3.7 ALDATA parameter

This parameter allows to create a "search list" in which input data files will be looked for. A search list is a logical name having multiple equivalence definitions; it often is used when you want to search for a file into a group of directories. SEV will use the specifications you set as ALDATA values only if you supply EPIO or NATIVE as the format of input files. This allows to include a personal set of data directories where data files will be looked for, and is useful especially when you are writing a personal MY_SEVDFT.LOCAL file (see chapter 4). The search list will be named ALDATA.

The basic default search list is defined as AL$DATA, the standard ALEPH data location on ALWS and several sites using ALEPH data. If you don't specify any ALDATA parameter, your data files will be searched only into AL$DATA; if you define some ALDATA assignments, AL$DATA will be used anyway as the last element of your search list, so it's of no use specifying it into ALDATA definitions, except if you want it to be used before using other definitions of the ALDATA parameter.

Wildcards "*", "%", and ellipsis "..." are not allowed into ALDATA specifications.

Some examples of ALDATA specifications follow: the first and the second are valid declarations, the last two are invalid ones

```
ALDATA MY_ROOT:[MYDATA.EPIO]
ALDATA YOUR_DATA
ALDATA YOUR_DATA,HIS_DATA:[NATIVE]
ALDATA MY$ROOT:[DATA...]
```

In fact, the third declaration includes two specifications on the same line, and the fourth includes a wildcard.

The first ALDATA definition will produce a search list composed as follows

```
"ALDATA" = "MY_ROOT:[MYDATA.EPIO]"
         = "AL$DATA"
```

It is the same definition that you could obtain by giving a DCL command such as

```
$ DEFINE ALDATA MY_ROOT:[MYDATA.EPIO],AL$DATA
```

It means that every data file will be looked for into the directory named MY_ROOT:[MYDATA.EPIO] and, if SEV can't find it there, into AL$DATA. If the two valid definitions showed above were used in the same parameters-setting file, in the order they are showed above, the search list would be

```
"ALDATA" = "MY_ROOT:[MYDATA.EPIO]"
         = "YOUR_DATA"
         = "AL$DATA"
```

Remember that:

- The order in which you specify the ALDATA values into the default parameters setting file will be followed while composing the search list; AL$DATA will be always the last element of the search list.
- You may specify up to 20 ALDATA lines, each one containing only one directory specification.

3.8 ALEDIR parameter

This parameter is very similar to the ALDATA parameter (see section 3.7); it allows to create a "search list" in which input data files will be looked for, but SEV will use the specifications you set as ALEDIR values only if you specify EDIR as the format of the input files. The search list will be named ALEDIR.

The basic default search list is defined as AL$EDIR, the standard ALEPH location of EDIR files on ALWS and several sites using ALEPH data. If you don't specify any ALEDIR parameter, your input ".EDIR" files will be searched only into AL$EDIR; if you specify some ALEDIR assignments, AL$EDIR will be used anyway as the last element of your search list, so it's of no use specifying it into ALEDIR definitions, except if you want it to be used before using other definitions of the ALEDIR parameter.

Wildcards "*", "%", and ellipsis "..." are not allowed into ALEDIR specifications.

The valid and invalid examples that we could do about ALEDIR, are similar to the ones that you can see into section 3.7 about ALDATA; the only difference is that the general form of the assignment is, of course,

```
ALEDIR edir_site_specification
```

Remember that:

- The order in which you specify the ALEDIR values into the default parameters setting file will be followed while composing the search list; AL$EDIR will be always the last element of the search list.

- You may specify up to 20 ALEDIR lines, each one containing only one directory specification.

3.9 DFTIN parameter

This is an optional parameter, defining the default format of the input files. This value will be used when you will be asked to enter the format of the input files from keyboard: if you will press the [RETURN] key without selecting a format, this default value will be used. If, on your site, data files have normally just one data format (EPIO or NATIVE) or if you generally use EDIR files as input, specifying this parameter allows to input data very fastly. To set it, specify something like

```
DFTIN EPIO      or
DFTIN NATIVE    or
DFTIN EDIR
```

Remember that:

- If you don't specify this parameter, the default input data format will be set to EPIO.

- Only one DFTIN specification is allowed into a parameters-setting file; if more are specified, only the first one will be considered as a good value, while the other ones will be ignored.

3.10 DFTOUT parameter

This is an optional parameter, defining the default format of the output file. This value will be used when you will be asked to supply the format of the output file from keyboard: if you will press the [RETURN] key without selecting a format, this default value will be used. If, on your site, data files have normally only one data format (EPIO or NATIVE), specifying this parameter allows to input data very fastly. To set it, specify something like one of the following lines, depending on the format you want to set

```
DFTOUT EPIO      or
DFTOUT NATIVE    or
DFTOUT EDIR
```

Remember that:

- If you don't define a value for this parameter, the default output data format will be set to EPIO.
- Only one DFTOUT specification is allowed into a parameters-setting file; if more are specified, only the first one will be considered as a good value, while the other ones will be ignored.

3.11 DFTMOD parameter

It is an optional parameter, defining the default mode of execution. This value will be used when you will be asked to enter the running mode from keyboard: if you will press the [RETURN] key without selecting a mode, this default value will be used. Available modes are INTERACTIVE and BATCH. This parameter is useful if you generally run SEV always in one of the two modes; so, if you specify BATCH as the value for DFTMOD, for instance, you will have it as your default way of running, instead of INTERACTIVE. To set it, specify something like

```
DFTMOD BATCH          or
DFTMOD INTERACTIVE
```

Remember that:

- If you don't specify this parameter, the default running mode will be set to INTERACTIVE.
- Only one DFTMOD specification is allowed into a parameters-setting file; if more are specified, only the first one will be considered as a good value, while the other ones will be ignored.

3.12 DFTQUE parameter

It is an optional parameter, defining the default batch queue used for the execution of the ALPHA batch jobs. This value will be used when you will be asked to enter the batch queue from keyboard, after selecting batch running mode: if you will press the [RETURN] key without selecting a queue, this default value will be used. The general form of this specification is

`DFTQUE` *queue-name*

For instance, you could specify

```
DFTQUE MYQUEUE$BATCH
```

Remember that:

- If you don't specify a value for this parameter, the default queue will be set to SYS$BATCH, that is the default queue for the DCL command SUBMIT.
- Only one DFTQUE specification is allowed into a parameters-setting file; if more are specified, only the first one will be considered as a good value, while the other ones will be ignored.

3.13 Writing comments

Comment lines can be written into your parameters-setting file. You have to put the symbol "*" as the first character of every comment line. For instance, you could write

```
* This is a comment, it is not an executable line
```

The lines that you define as comments won't be interpreted: they can be useful for you to write some information about the structure and the purposes of the file you are creating.

3.14 Examples

Listings of SEVDFT.CERN file and SEVDFT.TRIESTE file are available in figure 2. The first one could be used on CERN site, the second one was used on our site in Trieste, and it is a good example of a SEVDFT.LOCAL file to be created when you use packed runs. These two files should be available in every complete SEV installation.

Remember that:

- The order in which you specify the parameters into a parameters-setting file is not important, except GENFIL, ALDATA and ALEDIR lines, of course.
- The maximum length of every line of the parameters-setting file is 80 characters. Extra characters will not be considered.

```
* This is SEVDFT.CERN
GENFIL D%%%%%%%
GENFIL P%%%%%%%
DFTIN  EDIR
DFTOUT EPIO
DFTMOD INTERACTIVE
```

```
* This is SEVDFT.TRIESTE
* Now obsolete
GENFIL T%%%%%%%
TABLE  SEV_DIR:OURDATA.TABLE
RUNCOL 1
RUNLEN 7
FILCOL 11
FILLEN 8
DFTIN  NATIVE
DFTOUT NATIVE
DFTMOD INTERACTIVE
```

Figure 2: Examples of parameters-setting files

4 Personal settings

SEV can be easily adapted to the needs of every single user on a site. In fact, every user can create a personal parameters-setting file, containing his/her own parameters settings, that SEV will read instead of the general-user's parameters-setting file created for a particular site.

To adapt SEV to your needs, just create a file named MY_SEVDFT.LOCAL and put it in the directory SYS$LOGIN, that is the one in which you are placed any time you log in. When you run SEV, it first looks for a file named MY_SEVDFT.LOCAL contained into your SYS$LOGIN directory, while searching for the parameters-setting file. Only if this file doesn't exist, it will look for a file named SEV_DIR:SEVDFT.LOCAL, containing the general rules for your site, as described into chapter 3.

The rules to follow when writing a personal MY_SEVDFT.LOCAL file are the same explained into chapter 3. Contact the responsible for data files on your site, when creating your personal parameters-setting file, to make it correct according to the data organization on your site. The better thing to do, if you want to personalize it, is to copy SEV_DIR:SEVDFT.LOCAL file to the directory where you log in, naming it MY_SEVDFT.LOCAL, and then to modify this file according to your needs, avoiding to change the fundamental parameters for your site.

This operation can be useful, in particular, if you have a personal group of data files, and you want to extract events from these files instead of public data files available on your site, or if you want to extract events from both private and public data files.

5 Working on Monte-Carlo files

SEV has been built to work very easily on the normal files containing runs, either containing a single run or several runs each. Working on Monte-Carlo files can't be performed in the same way as for working on runs; in fact nowadays there isn't a direct relationship between the name of a Monte-Carlo file and the number of the run which identifies all the events contained into it, because there isn't a standard way of fixing these names. Consequently, different Monte-Carlo files containing events having the same run number exist. This situation is ambiguous for SEV, as its only significant key to search the events is the run number.

A particular way of working allows to skip this problem: you should create a table file, structured as described in section 3.2. For instance you could use the table showed in figure 3.

```
I col.      XI col.
↓           ↓

  Run     * MC-File  *
--------------------
101       * AC0969   *
511       * AF7213   *

                          (...)
```

Figure 3: Example of relationship table for Monte-Carlo files

The rules followed to build a table like this are the same described in section 3.2, about building a table file for common run files. The only difference is that the column of the table in which filenames are written, contains the names of the Monte-Carlo files. Only one Monte-Carlo file containing events belonging to a run can be included into the table (that is: you can't write two lines having the same run number into the table); if you want to extract events from two Monte-Carlo files containing events having the same run number, you must include the first Monte-Carlo file in the table, run SEV, modify the table replacing the first Monte-Carlo file in which you are interested with the second Monte-Carlo file name and finally run SEV again: you will obtain two output files, the first including the events extracted from the first Monte-Carlo file, the second the ones extracted from the other Monte-Carlo file.

Before running SEV, you must create (or change, if one exists already) your personal parameters-setting file MY_SEVDFT.LOCAL (see chapter 4 for more information about building this file). You must include a definition for the TABLE, RUNCOL, RUNLEN, FILCOL, FILLEN parameters, so that your new table containing the Monte-Carlo relationships will be used to find the desired files.

Extracting events belonging to Monte-Carlo files having different run numbers is allowed in a single SEV running, by including all the specifications of the Monte-Carlo files in which you are interested into the table.

6 Input and output files

SEV works on several files. To run it correctly you must have READ access to the following files:

- SEV_DIR:SEVDFT.LOCAL (see chapter 3) if it exists (or SYS$LOGIN:-MY_SEVDFT.LOCAL, if you set a personal parameters-setting file, see chapter 4);
- all the data files containing the runs that you want to examine, either EDIR, NATIVE or EPIO format files. They are normally contained into AL$EDIR and AL$DATA areas (see chapter 2), but they could be contained into different areas if you specify ALDATA or ALEDIR parameters into the parameters-setting file (see section 3.7 about ALDATA or 3.8 about ALEDIR). If a data file is not readable, the runs included into it will be ignored;
- the table file containing relationships between run numbers and file names for your site (if you specified it as TABLE parameter into the default parameters-setting file, see section 3.2);

You must have READ EXECUTE access to the following files:

- executable files contained into SEV_DIR directory; in particular: SEV_SETUP.COM (setting logical names and symbols), SEV.EXE (SEV executable file) and SEV_MAKEIT.EXE (ALPHA job run by SEV.EXE to select events);

You must have WRITE DELETE access to the current directory (no matter if you are running in batch or interactive mode), because some temporary files will be created and then deleted by SEV, and all the output files will be put into your current directory. Be sure to have enough room for all the output and temporary files into your current directory!

The output files will have the same name as the input list; for instance, if your list is named MYSCAN.DAT, your output file will be named MYSCAN.NATIVE or MYSCAN.EPIO (depending on the kind of output you selected), while the event directory will be MYSCAN.EDIR. The temporary CARDS file will be named MYSCAN.CARDS. To avoid ambiguous situations, you should not give a filetype of EPIO, NATIVE, EDIR or CARDS to your input list.

If running in batch mode, some extra temporary file will be created, see section 7.6 for more information.

7 Running SEV

7.1 Starting

To execute SEV from anywhere, you can use a symbol that is defined by SEV_SETUP.COM procedure (see chapter 2); so just SET DEFAULT to the directory where you want your output files to be created and type

```
    $ SEV
```

Please notice that if logical name SEV_DIR is undefined or if it has been badly assigned (it means you didn't run SEV_SETUP procedure), SEV can't run correctly, so the program quits immediately.

If you need to quit SEV, you can use the [CTRL-Y] key sequence.

7.2 Setting default values

When you run SEV, it tries to load the values for default parameters; as explained in chapter 4, it first looks for a personal parameters-setting file, placed into the directory where you log in, named SYS$LOGIN:MY_SEVDFT.LOCAL; if this file exists, the parameters will be loaded from it, otherwise SEV looks for the site specific parameters-setting file, named SEV_DIR:SEVDFT.LOCAL. If this file exists, SEV starts to load default parameters values from it. You will be notified about every external setting (that is, every definition read from the parameters-setting file) by a message saying

... Setting value of *parameter name* = *parameter value*

if there are some invalid definitions, you will be notified about them. Some parameters will be set internally by SET, if you don't specify them into the parameters-setting file. When this happens, you will be notified by messages saying

... Internal value of *parameter name* = *parameter value*

The values that will be assigned to each parameter if this situation occurs, are described into chapter 3 when explaining about parameters. Maybe some values you set as external are not good ones (for instance, if you give as DFTMOD parameter value a different value from "INTERACTIVE" or "BATCH"); in this case, you will be notified about the invalid value and it will be replaced by the internal default value.

If no parameters-setting file is found (both personal or general), SEV will ask you about using CERN default values, that are the ones showed in the listing of SEVDFT.CERN file in figure 2. If you answer "N" to this question, no values are available to SEV, so the program quits.

7.3 Typing input

When the values of the parameters are available to SEV, the input section starts; in this section of your running, you will be asked to input a number of data from keyboard. Most of the questions include a specification in square brackets: if you hit the [RETURN] key without supplying an input value, and such a specification exists, the value in square brackets will be taken as input.

The first datum you are asked about is the name of the file containing the list of run-number, event-number you want to be processed. Type the complete path of your file; press the [RETURN] key instead of typing the name if you want to exit from SEV. The list of events can be produced by using a program such as PAW, or directly by using an editor: it must be in the form of a table, composed at least by two columns, one containing the run number and the other the event number of every event to be processed. If there are other columns, containing other information, they will be ignored.

For instance, we might use the table in figure 4 (it is a part of a table created by using PAW)

```
I col.       XIII col.     XXVII col.
↓            ↓             ↓

*****************************************************
* ENTRY *   RUN      *   EVENT     *   MM2         *
*****************************************************
!  0036 !   7711.0   !   16.000    !   2103.3      !
!  0063 !   7711.0   !   2353.0    !   3914.3      !
!  0498 !   7396.0   !   2632.0    !   507.79      !
!  0582 !   7397.0   !   2644.0    !   5849.7      !
!  0741 !   6972.0   !   5083.0    !   4051.7      !
!  1243 !   7142.0   !   2082.0    !   5739.5      !

                       (...)
```

Figure 4: Example of input list

The list might contain up to 10000 events.

You should not give a filetype of EPIO, NATIVE, EDIR or CARDS to the list, as this could be ambiguous when producing the output files. See chapter 6 for more explaination.

SEV needs to know some more information about the list; in particular it needs the column where the run numbers and the event numbers are written in your table; so it asks you to input the column of the table (that is, the number of characters from the left border of the table) where run numbers are available; in the table showed in figure 4, you could input the number 13. SEV needs to know also the number of characters to be taken when getting the run numbers, so this value is asked to you. In the table showed in figure 4, you could type 7: it means that the run numbers will be taken considering the 7 characters of the table from the 13th.

The same data are needed about the event numbers. You could input 27 as the initial column, and also 7 as the number of characters to be taken. (Notice that the blank column at the left of the table is counted as a column of the table.)

The specified range of columns may include leading and/or trailing blanks.

As SEV allows to use input files having a filetype of NATIVE, EPIO or EDIR and to produce outputs having one of the same formats, you must supply the input and the output format; please notice that all the input data files must have the same format. The two messages asking this information suggest, in brackets, that you must press "E" key to select EPIO format, "N" for NATIVE and "D" for EDIR. A default format is available in square brackets: it is the one specified as DFTIN (for input format) or DFTOUT (for output format) parameter in SEVDFT.LOCAL (or MY_SEVDFT.LOCAL) file, or "EPIO" if no format was specified; if you hit [RETURN] key instead of selecting a format, the one in brackets will be used.

If you select EDIR as input format, the EDIR files will be used to find needed data into EPIO and NATIVE files faster; of course, either EDIR and EPIO (or NATIVE) files must exist. When selecting EPIO or NATIVE format as output, an EDIR file will be created too, relative to the EPIO or NATIVE output file; if you select EDIR output format, only an EDIR file pointing to the original input data files will be created.

The last information you must input is the mode of running the ALPHA job that will select events. You can run it in interactive or batch mode. The requesting message suggests, in brackets, that you must press "I" for interactive, and "B" for batch mode. A default value is available in square brackets: it is the one specified as DFTMOD parameter into the parameters-setting file, or "INTERACTIVE" if no mode was specified. If you select batch mode, you will be asked for the starting time and the batch queue where the ALPHA job will be submitted. The default value of the starting time (used if you press [RETURN] key instead of giving a valid time specification) is "NOW": it means that the job will start immediately; you can specify the starting time as either an absolute time or as a combination of absolute and delta time, exactly as you can do when specifying a value for the /AFTER qualifier of the DCL command SUBMIT. The default queue is the one you specified as DFTQUE parameter into the parameters-setting file; if no queue was specified, SYS$BATCH is taken as default. You can submit the ALPHA job into any queue you want, specifying it when SEV will ask it to you.

7.4 Selecting events

Before running the ALPHA job, SEV needs to create a CARDS file; a temporary CARDS file is a file containing the specifications about input and output used by an ALPHA job and several other information needed by the ALPHA job.

SEV scans the list of events, ignoring every line containing invalid data (you will be notified about any invalid line found), loads the list into memory and sorts it, using the run number as primary key and the event number as secondary key. These operations might require some time, so please be patient until SEV will have performed them. The input file may contain up to 10000 valid events specifications. They should be enough, but if you want to process more than 10000 events, please modify the source program and recompile it or contact the author for help.

When the list is sorted, SEV starts to write the CARDS file, naming it with the same filename as your input file, but changing the filetype to ".CARDS". For instance, the CARDS file corresponding to an input file named MYSCAN.DAT will be named MYSCAN.CARDS. SEV looks for the data files containing the runs you need, following the rules defined when specifying the GENFIL and/or TABLE parameter into the parameters-setting file. If it doesn't succeed in finding the files it need, SEV will notify you: the events belonging to the runs which SEV isn't able to find will be ignored. If you specified a search list by defining GENFIL parameters into the parameters-setting file, you will be notified about any unsuccessfull attempt to look for a valid file.

The structure of a typical ".CARDS" file written by SEV is showed in figure 5.

```
******* Cards for CARDS-file-name file
DEBU 0
NOPH
FILI ' first-input-file-name '
SEVT list-of-events-to-select-from-first-input-file
FILI ' second-input-file-name '
SEVT list-of-events-to-select-from-second-input-file

                         (...)

FILO ' output-file-name '
ENDQ
```

Figure 5: Structure of a file of CARDS written by SEV

Read chapter 6 for more explaination about output files naming.

When the CARDS file is complete, an ALPHA job will run to select files: different operations are needed depending on the mode of running the ALPHA job: interactive or batch.

7.5 Running ALPHA job in interactive mode

If you selected to run the ALPHA job in interactive mode, it will start immediately after the creation of the CARDS file. The ALPHA image is named SEV_DIR:-SEV_MAKEIT.EXE and it will be spawned as a subprocess of the process running SEV. The output of the ALPHA job will be showed on your screen, so you will be able to control the situation. When the ALPHA job will end, a short statistic will be showed about the number of input and output events, and the number of events skipped while searching for the corresponding data files. The name of the output files will be showed too.

The CARDS file will be automatically deleted by SEV at the end of the running.

7.6 Running ALPHA job in batch mode

Running in batch mode allows to save time, especially when the list of events to be selected is rather long; in fact SEV can submit the ALPHA job as a batch job, finishing your SEV running very soon, while the ALPHA job continues to run. The output files will be ready only when the batch process will be ended.

When running in batch mode, SEV creates a series of files: some of them are temporary and will be deleted before ending SEV execution, other ones won't be automatically erased, as they are needed by the batch process, so you should not delete them before the batch process ends. These files have the same name as the input file, but different filetypes, in particular:

- *filename*.TMP_1 and *filename*.TMP_2 are temporary files created by SEV and deleted before the execution ends, if everything runs correctly;
- *filename*.JOB is the command file that will be submitted as a batch job; it contains the needed commands to run the batch job correctly; it will be automatically deleted at the end of the batch running;
- *filename*.LOG will be created by *filename*.JOB when this command procedure will be run as a batch job; it is the log file for the ALPHA job, containing all the information relative to the ALPHA job itself. Reading it when an ALPHA job is ended allows to know what happened during ALPHA execution; this is the reason why this file is not deleted after the end of the job.

The job will be submitted using the following parameters: /KEEP (the LOG file won't be erased after the job ends), /NOPRINT (the LOG file won't be automatically printed after the job ends), /NOTIFY (a message will be sent to the user who ran SEV to notify him/her when the ALPHA job ends), /DELETE (the command file will be erased after execution). The batch job will be named using the same name of the input file, sumbitted on the queue that you selected while giving input on keyboard, starting at the time you specified.

All of the temporary files will be created into your current directory: they are not very big, but you should be sure to have enough free room to contain them and the output files too.

Some statistics will be showed before ending SEV execution, about the number of input and output events, and the number of events skipped while searching for the corresponding data files. The name of the output files will be showed too, including the name of the batch command procedure and the one of the log file. The starting time and the batch queue will be showed too.

8 Error messages

There are two kinds of error messages, corresponding to two levels of severity.

First-level errors are the ones which are not fatal for program execution: for instance, the error messages showed when you type an out-of-range data on keyboard, or messages about a badly formatted data line in the input file, are first-level errors. First-level error messages have "???" as first characters. If an error of this kind appears, read it with care, even if it is not a fatal error, because maybe something will not work in the exact way you expect it. For instance, maybe some events you want to select will be ignored because of a missing data file or because of a badly formatted line in your input list.

Second-level errors are fatal errors; the execution is aborted when a second-level error occurs. For instance, if something bad occurs while opening the input list, a second-level error occurs. Second-level messages have "###" as first characters.