A working prototype of FOT filing program.

L.Chiappetti -IFCTR

Issue 1.0 - Mon, 16 May 1994
DAWG-REP.3.0/94

1 Introduction

This note describes the design assumptions made for the development of the prototype SAX FOT filing program. It provides also a description of the prototype program from the user's point of view.

The program has been tested using the "sample FOT" distributed on DAT by Telespazio and circulated to the DAWG by Daniele Dal Fiume (ITESRE Fax SGS/06.05.84/1) of 6 May 94. Tests have been performed on Ultrix, SunOS and VMS and are reported in section 4.

Open points which may need to be changed in a final version of the program are indicated by a bar on the margins (like this paragraph).

I regard the tests as successful. The feasibility of the architectural approach has been proven, and I would be glad to pass the further development of the prototype to some skilled programmer.

2 User information

2.1 Program calling sequence

The program (which is actually a multi-pass program, see below section 3) is called as :

```
xasset rootdir someabsolutepath
xasset fotdir somename
fotfile [tape experiment inslewobs targetobs outslewobs datatypes]
```

where the first two commands establish the directory (*someabsolutepath/somename*) where the data will be filed, and the last invokes the first pass of the FOT filing procedure according to the usual XAS user interface (i.e. parameters which are not passed on the run string are prompted to the terminal - or read from a command file if one is defined). The meaning and syntax of the various parameters is :

`tape` is so far a system dependent tape device name (e.g. `/dev/nrmt0l`, `MUA0:`). Currently no check is made that it is syntactically correct for the particular operating system. Also he tape must be local to the machine from which `fotfile` is run.

Note that a delay will occur after the request of the `tape` argument, while the tape directory is bein filed to disk. All further questions are asked by pass 2 of the filing procedure.

`experiment` is an instrument code, specifying which files are actually copied to disk. It can be one of the following :

`XX` requests to file "any" instrument. This is not exactly the same as the "`**`" code appearing in the tape directory (which incidentally cannot be used since it interferes with the Unix shell wildcards). `XX` means copy all files flagged as "`**`" and all files of the first instrument appearing on tape. If more instrument datasets appear on tape, the following ones must be requested explicitly. Note that the file naming is such that it is not advised to file data from separate instruments in the same `fotdir` directory.
If one instrument only is requested (codes below) the files flagged as "`**`" are always filed unless they are excluded by datatype or class.

`ME`	for data files of the MECS (inclusive of `M1`,`M2`,`M3`; it has not been tested)
`Mn`	for data files flagged as (`n`=1-3) of MECS unit `n`, plus `ME` data files (also not tested).
`LE`	for data files of the LECS
`HP`	for data files of the HP-GSPC
`PD`	for data files of the PDS (inclusive of `P1-P4`; also not tested)
`Pn`	for data files of PDS unit `n` (`n`=1-4), plus `PD` data files (also not tested)
`Wn`	for data files of WFC `n (n`=1 or `2)`

`inslewobs` is an "extended range" of observations of the initial (ingoing) slew for which filing is desired. Extended ranges are currently defined in the format `n+m+p-q` (i.e. a range `p-q` indicates all observations from `p` to `q`, a single number `n` indicates observation `n`, and a plus sign performs an "and" of the ranges). E.g. specifiying `1+3+5-7` will file observations 1,3,5,6 and 7.
At the moment no blanks can be imbedded between observation numbers and the plus (`+`) and dash (`-`) sign.
A range of `0` indicates no data to be filed. A range of `1-99` indicates all data to be filed.

`targetobs` is an "extended range" of observations of the pointed part of the observing period ("normal" in Telespazio parliance), specified with the same syntax as `inslewobs.`

`outslewobs` is an "extended range" of observations of the final (outgoing) slew, specified with the same syntax as `inslewobs`.

`datatypes` is intended to specify which types of files have to be filed. The proposed syntax could be of the form `class+class+type+NOclass+NOtype`, where `class` is a code designating a group of file types, `type` is a single filetype as appearing in the FOT tape directory (but case insensitive), while the `NO` prefix excludes a given type or class from filing (e.g. `A+C` files class A and C data, `ALL+NOB` files all data but class B ...).
At the moment the following classes are defined :

`A`	the `tapedir` file
`B`	the `ephemeris`, `attitude` and `obt_utc` "common" files
`C`	all spacecraft HK "common" files
`D`	all "instrument" directories (`instdir`, `obsdir`, `expconf`)
`E`	all instrument HK (both VC1 and science VC) files
`F`	all instrument science data
`ALL`	`A+B+C+D+E+F`
`DEF`	`A+B+D+E+F` (default)

In a future the definition of classes could be changed (e.g. B could be split to avoid filing orbit data if unnecessary; F could be split in "primary" science (direct or indirect energy spectra) and "secondary" science (indirect timing, non-energy spectra, science HK).

Currently the selection of individual datatypes is <u>not supported</u> at any level, the `NO` specifier is not supported, and the class specifiers are <u>ignored</u>. In all cases the `DEF`ault class is assumed, <u>irrespective</u> of user input.

2.2 The procedure

The following procedure is recommended for orderly filing :

`fotfile tape XX 0 0 0 A` to file the tape directory only
look at the tapedir file (which is a plain ASCII file)
reissue `fotfile` for the wished experiment and subset of files
analyse one instrument
change `fotdir`
reissue `fotfile` for another experiment

The current version of the program does not support multi-volume tapes.

`fotfile` is a multipass procedure which will display a number of messages. Some of them are issued by the program passes (e.g. the list of selected and rejected files, or the final verification messages), some other by the system commands used for actual tape copy. The last pass after the tape copy attempts to verify that filing was correct. As a result the last part of each line in the tapedir is replaced by a flag field with values `OK`, `NO` or `**`, meaning respectively filed correctly, not filed at all, filed with errors.
An example session is given below in appendix.

2.3 File naming convention

The various files present on the FOT are filed on disk (in the fotdir directory) with a *name* and a *type*. The file type is taken *verbatim* (but neglecting case) from the corresponding field in the tape directory (e.g. `tapedir`, `ephemeris`, `instdir`, `w1hkd000`, `w1dir001` etc.), while the names are generated as follows :

`saxfot`	For the "common" files not depending on instrument (orbit, attitude, etc. appearing at the front of the tape)
`exp`	The `instdir` will be filed with the relevant instrument name, e.g. `w1.instdir`.
opexpobs	All other data files (numbered by observation) will be filed with a code composed by 1-character observing period type `op` (=`i`,`n`,`f`), a 2-character `exp` code, and a three-digit observation number `obs`. E.g. the `expconf` file of ingoing slew observation 1 will receive a name like `iw1001.expconf`.

3 Program design

3.1 architecture

According to DAWG-REP.18/92 the program has been designed as a multipass procedure which makes use of system dependent facilities to access tapes.

The complete procedure includes three passes as follows :

Pass 1	the program fotfile retrieves the tape drive name, and creates a file temp.com in which it writes the system dependent commands to file the tapedir from tape to disk, and the command to start pass 2. It is care of this pass to retrieve all other arguments and pass them unchanged (along with the tape name) to pass 2. It then terminates via a Z_RUN call which overlays itself with the execution of the temp.com file.
temp.com	rewinds the tape, copies the tapedir to disk, rewinds the tape again and starts pass 2.
Pass 2	the program fotfile_2 retrieves all other arguments, reads the tapedir from disk record by record, and for each tape file checks whether it is requested to file it or not. It inserts accordingly into a new version of temp.com the system dependent commands necessary to copy the files or to position the tape, and the command to start pass 3. It then terminates via a Z_RUN call which overlays itself with the execution of the temp.com file.
temp.com	rewinds the tape, copies all requested files and starts pass 3.
Pass 3	the program fotfile_3 reads the tapedir from disk, and record by record checks whether the tape file has been filed to disk, and whether it has the correct size (this is the only check which makes sense both in Unix and VMS; in VMS could also check the record length ...). It prints a message to the screen and updates the tapedir with a flag OK NO or ** (filed correctly, not filed, wrong size). Finally it deletes any leftover temp.com file.

3.2 pass 1

The current source code of fotfile is somewhat redundant of commented code, since originally I thought to have all the dialogue and retrieval of arguments in this pass. (later I moved most of this to pass 2).
At the moment there is no check on the tape drive name (e.g. presence of /dev in Unix, presence of trailing colon in VMS, existence of the device on the machine).
The handling of the tape commands is done as specified below in 3.5.
The tapedir is assumed to have constantly a record length of 132 and a blocksize of 32736 (so that this information can be hardcoded in the program).
The command used to execute (Z_RUN) pass 2 is system dependent, and is csh temp.com in Unix and @temp.com in VMS.

3.3 pass 2

Also the source code of fotfile_2 may contain redundant code since it has been derived from fotfile via a copy (e.g. error handling may be incorrect).
A routine parseobs takes care to parse the "extended observation ranges" of the form n+m-p+q, with the rather crude mechanism of setting into a 99-position integer array a non-zero value for observations to be filed.
At the moment there is no parsing of the "filetype classes". It is foreseen to have a family of logical flags (A to F) corresponding to the classes defined in 2.1, but at the moment their values are hardcoded (for the DEFault class, i.e. all are true and C is false). Rejection of classes via the NO prefix, and selection on individual file types are not supported (and possibly they'd never need to be).
The handling of the tape commands is done as specified below in 3.5.
The first record of the tapedir (descriptive string) is currently ignored, also because its syntax is still under redefinition (date fields are at the moment in an illegal format).
There is no handling of the volume fields (hence no handling of multivolume tapes; I suggest to defer this until it will be clear that multi-volume tapes will be actually produced).

The program then scans the tapedir and for each record checks if filing is requested for the particular file :

It first checks the instrument code.
The first record with an instrument field not equal to "`**`" is compared with the requested instrument, and, if the requested instrument is `XX`, is assumed as "default" (otherwise the default is the requested instrument). Only data files flagged as "`**`" or as the default instrument are taken.
In addition there is a special handling for ME and Mn, and PD and Pn. The desirable behaviour has to be verified, and the relevant code to be tested.
Files flagged as "`**`" are always accepted at this stage.

It then checks the observation number.
Files belonging to observations outside the requested ranges are rejected. Note that files flagged as "observation 0" are always accepted at this stage.

It finally checks the datatype.
At the moment only checking by class is implemented (no individual datatype rejection).
While some classes (A,B,D) are defined by a list of datatypes, other are defined by a pattern match on the filetype field in the tapedir :
Class C (s/c HK) is defined by a filetype `xxHKDnnn` with `xx` not equal to the fotdir instrument code;
Class E (p/l HK) is defined by a filetype `xxHKDnnn` or `xxENGnnn` with `xx` equal to the instrument
Class F (science) is defined by a filetype `xxyyynnn` with `xx` equal to the instrument code, and `yyy` not equal to `HKD` or `ENG`.

The filenames are constructed according to the naming convention defined in 2.3.

A rewind tape command is always inserted at the beginning and end of the filing procedure, to leave the tape in a known position. Tape skip commands are inserted when appropriate (but not after the last requested file). Messages telling the user what is happening are always inserted before each tape command.

3.4 pass 3

Also the source code of `fotfile_3` may contain redundant code, on the other hand it is also possible that some code (e.g. building file names) might be shared with pass 2, and should be moved to a library subroutine.
This pass is to be regarded as highly provisional (it verifies consistency between tape directory and status of disk files, but at the moment it handles all files present on disk, and does not verify whether those requested (and only those) have been filed.

This pass scans the tapedir, and for each file it checks whether it has been filed or not. In the case it is filed it also checks the file size is as expected (number or records * record length). It does not check the record length, since this is not possible in Unix. In addition in the case of Unix for ASCII files (i.e. all directories, and the orbit, attitude and OBT files), the record length for the check is 1-character longer than the one in the tapedir (to account for the newline inserted by Unix).

Ideally this pass should open the tapedir, and update each record by cutting out the last part of the comment field and replacing it with a quoted 2-character flag (in the forms `'OK' 'NO'` or `'**'`). At the moment this is not done accessing the file as direct access (this is not possible on Ultrix because the Unix record length is 133 - not multiple of 4 - because of the newline, and `Z_OPEN` does not handle this). Therefore the content of the original tapedir file is copied record to record to a temporary file, and at the end is copied back into the tapedir, while the temporary file is deleted. A more efficient way of doing it, would be to avoid the final copy-back, and just delete the original tapedir, and renaming the temporary file as the tapedir. This would require introducing in the VOS a `Z_RENAME_FILE` function.

3.5 Tape handling

A number of library routine write into the temporary file temp.com the necessary commands for :

tape initialization: (VMS only : the appropriate MOUNT command)
tape rewind
tape skip (forward file skip)

unblocking of a tape file with known logical record length and blocksize in either ASCII or binary form
final commands (invocation of next pass with appropriate run string)

Most of them (not the initial and final commands which are hardcoded in the program) write out an *edited template* command. The templates are read once at beginning from a tape capability file `tape.cmds`, which shall be present in `$XASTOP/local`. There should be a different file for each operating system. It contains the template for all necessary tape commands, plus for the `MESSAGE` command (ncessary to type an informative message from a script). Its syntax is defined in the following example :

```
# This file contains the template for tape commands
# Entries are in the following format
#
#    anything starting with # in col. 1 is a comment
#    each entry is 5 lines long
#    first line contains the tape name (so far only the DEFAULT tape is supported) terminated by semicolon
#    the other entries are in the form function: command
#    where the four functions REWIND, SKIP, BINCOPY, ASCCOPY must appear as in the example
#    followed by the MESSAGE function
#    In the template commands "tokens" of the form $x indicate :
#
#       $t is the tape device name
#       $f is a file name or a string
#       $n is a numeric repeat count
#       $b is a blocksize
#       $r is a record length
#
DEFAULT:
REWIND : mt -f $t rewind
SKIP   : mt -f $t fsf $n
BINCOPY: dd if=$t of=$f ibs=$b
ASCCOPY: dd if=$t of=$f ibs=$b cbs=$r conv=unblock
MESSAGE: echo $f
#
#-------- end of file -------------------------------------------------------------------------
```

At the moment this file contains only a `DEFAULT` family of entries, which applies irrespectively of the tape devices requested. In the future one might implement more families of entries to cater for the case different tape drives require different commands (unlikely, the system commands should cater for this), or to support symbolic names for tapes (this might be useful in an Unix environment to support a case of remote tape drives to be addressed via rsh).

In a future version either Pass2 or Pass 3 should also execute (on non-Digital Unix systems) a (packetcap driven) `swapdata` program to prepare binary data in the "normal byteswapping order" (big endian) required by such system.

3.6 Unix dependent issues

In Unix the tape commands to be used are `dd` and `mt. echo` is used for displaying messages. Separate `dd` options are used for the case of binary data filing (copy tape stream unchanged), and for the case of ASCII data filing (used for all directories and orbit/attitude/OBT files; in this case it is necessary to place a newline after each record to allow typing at the terminal). The relevant commands are presented in 3.5 above in the template `tape.cmds` file.

The `temp.com` file is executed passing it as argument to `csh` (it cannot be executed as a shell script, since it cannot be created with execute permission set - the current VOS has no function to set such permission).

3.7 VMS dependent issues

There are a number of VMS-specific tape commands to be inserted in `temp.com` (executed as a standard DCL `.com` file), namely a `MOUNT/FOREIGN` at the beginning (with a blocksize of 32768 bytes), and a `DISMOUNT/NOUNLOAD` at the end.

In pass 3 there is a problem testing the size of the tapedir (Z_INQUIRE returns zero for an opened file), which is cured testing its size before opening it.
Also testing the sizes of the other ASCII files depends on their format (fixed, variable or STREAM_LF) therefore ultimately on the unblocking command used (see section tests in 4.4)

3.8 Software installation and distribution

There are three main program source files for the three program passes (`fotfile.f`, `fotfile_2.f` and `fotfile_3.f`), a number of subroutines with a provisional arrangement into `saxfotlib.inc` (which will be moved to the `fotlib` library contextually with the XAS 1.1 release), an include file `saxfot.inc`, and a local customization file `tape.cmds` to be installed in `$XASTOP/local`.

All these files are currently installed in my own `$XASTOP` (`/poseidon/lucio/xas` or `DUA0:[LUCIO.XAS]`).

I would *not recommend* retrieval and re-installation of software from the source files, until it will be integrated in the XAS 1.1 release.
Those desiring to test the software, should retrieve *only* the executables of `fotfile`, `fotfile_2` and `fotfile_3` (from `$XASTOP/bin`) and the tape customization file `tape.cmds` to be installed in `$XASTOP/local`.

4 Tests

4.1 Ultrix local drives

The tests necessary to qualify the basic functionality of the program have been done on an Ultrix DECstation (`kronos.ifctr.mi.cnr.it`) with a locally attached DAT drive, using the DAT cassette supplied by D.Dal Fiume. An example of the test can be seen in the sample session in Appendix.

It has to be noted that the content of all files on the Telespazio DAT, but the tapedir, are meaningless (and I would say that their content is also misleading, being it represented by a stream of binary values 01 02... also in the case of files which should be ASCII and contain printable data.

An error has been detected in three files (`ephemeris`, `attitude` and `obt_utc`) in which the 32-nd record on the tape is 2-byte shorter than expected. This is shown as an "incorrectly filed" message in Pass 3 of the sample session shown in Appendix. The reason for this is that dd in unblock mode trims trailing blanks (and the 32-nd record of the Telespazio meaningless file ends with two blanks). Besides the verification error, there are no side effects (i.e. the files can be accessed without problems).

A further test has been done on the same Ultrix DECstation with a locally attached half-inch tape. The tape has been created (at 6250 bpi) with the procedure `diskfot` provided by D.Dal Fiume using the data provided in the associated tar file from ITESRE. This gives exactly the same results as above (inclusive of the errors).

4.2 Sun local drives

A further test with the same tape has been done on a Sun, using an identical `tape.cmds` file. This allowed to discover a minor bug in the routine `add_end`. This also gives exactly the same results as above.

4.3 Ultrix remote drives

A further test has been done on an Ultrix DECstation (`poseidon.ifctr.mi.cnr.it`) with a remote DAT drive on `kronos`, using the following `tape.cmds` file :

```
REWIND : rsh kronos mt -f $t rewind
SKIP   : rsh kronos mt -f $t fsf $n
BINCOPY: rsh kronos dd if=$t of=$f ibs=$b
ASCCOPY: rsh kronos dd if=$t of=$f ibs=$b cbs=$r conv=unblock
MESSAGE: echo $f
```

This allows to read data from a remote tape exactly as in all previous cases, only slightly slowly.
Attention ! The syntax used (2B in `DAWG-REP.18/92`) will work only if the output directory is NFS-mounted on kronos. The alternate syntax (2A in `DAWG-REP.18/92` with the `| cat >$f` instead of the `of=$f`) will not work with the current version of the program, since the routine (`edit_cmd`) which edits the template at the moment requires that the edits are applied left-to-right in the same order in which the tokens appear.

4.4 VAX local drives

A further test with the half-inch tape has been done on a Vax (`IFCTR`) with a preliminary `tape.cmds` like the following :

```
REWIND : $ SET MAGTAPE $t /REWIND
SKIP   : $ SET MAGTAPE $t /SKIP=FILES:$n
BINCOPY: $ UNBLOCK $t $f $b $r
ASCCOPY: $ UNBLOCK $t $f $b $r !may not work
MESSAGE: $ WRITE SYS$OUTPUT "$f
```

The `UNBLOCK` program is described in `DAWG-REP.18/92` and does not handle perfectly ASCII files (it dumps them as VMS `FIXED` files, they can be accessed programmatically also sequentially without problems, only typing them at the terminal may not look nice). `UNBLOCK` has been recompiled with a maximum block size of

32768,

This reads in all files and gives some errors in the verify-size phase: the tapedir (filed correctly) gives a zero-size if its size is tested while it is open (this is cured anticipating the test, see 2.7 above), and the obsdir files give an error (this is due to the fact that they have an illegal record length of 50 bytes (not multiple of 4) on the original tape and therefore are filed with a record length of 52). All other files have correct size and record length.

In addition there is a known bug with argument passage between passes. If all arguments are specified on the runstring for `fotfile` everything runs OK. If some are omitted, pass 2 seems unable to retrieve them from the run string.

Ideally one should use a program faster than `UNBLOCK`, which should be provided by the `TD` program written by B.Martino and G.Di Persio (see note by IAS). However at the moment I am unable to understand what is the correct syntax to use it. I have *inferred* the following which I have used to set up a `tape.cmds :`

```
REWIND : $ MT REW $t
SKIP   : $ MT SKI $t $n
BINCOPY: $ TD IF=$t OF=$f OBS=$b CBS=$r CONV=BLOCK
ASCCOPY: $ TD IF=$t OF=$f OBS=$b CBS=$r CONV=UNBLOCK
MESSAGE: $ WRITE SYS$OUTPUT "$f"
```

Of course foreign tasks shall be defined for `MT` and `TD` (or equivalently my `PATH` procedure can be used to set up them as known).

I have encountered several problems with `TD`. First of all the syntax is not exactly like to `dd`, I find "experimentally" that I have to put an `obs` argument where I expect an `ibs` ...
Then when converting to ASCII files it would be preferred to create `STREAM_LF` files and not plain sequential files (incidentally this causes the size-verify check to file, because the file appears 2 bytes/record longer because of the length words).
When dumping the tapedir using a blocksize of 32736 (which is larger than the actual size on tape, but is the *only one which can be assumed a priori*) the tapedir is created correctly, but a number of null records (records full of binary zeros) up to 32736 bytes are inserted, while `TD` should truncate the file at its actual length, as dd does. Since those null records cause an error in pass 2 (when reading the tapedir) I have preliminarily cured this truncating the file manually in edit, and running pass2 manually.
Also `MT` gives "Fatal error fault detected in module MTLIB" and physically dismount the tape. I presume this is an argument syntax problem (tape drive and number of skips inverted w.r.t. the documentation by IAS). This occurs on the `SKIP` syntax, therefore I have inverted it manually (I cannot change it in the program).
Also `TD` gives "output statement overflows record" on the ephemeris file (in ASCII mode as it should be; no error is given in binary mode), possibly it cannot handle very long records (this has recl=184).
Finally `TD` in `BLOCK` mode (is this the correct way to handle binary data ?) produces in output less records than expected. I have the impression that the do-loop in the program does not advance the record pointer (quantiy `Rec=count` in the `WRITE` statement).

Notwithstanding the above problems (which do not affect immediately the fotfile program) I consider that the test has demonstrated the viability of the overall architecture solution.

Appendix : sample session

The following is a sample `fotfile` session on `kronos`, reading the Telespazio template DAT supplied by D.Dal Fiume. The following conventions are used :

`Courier bold` indicates the system prompt.
`Courier` indicates `fotfile` program output (prompts and messages)
`Courier underlined` indicates program output echoing user input (XAS user interface)
`Courier italics` indicates output from the temporary script temp.com
Helvetica indicates user input.
Times Roman indicates explanatory comments.

```
temp 12> xasset datadir $cwd                                  set necessary environment
   Variable name (or FROM)  datadir
   Variable value (or ?)  /poseidon/lucio/temp
temp 13> xasset fotdir Fotx
   Variable name (or FROM)  fotdir
   Variable value (or ?)  Fotx
temp 14> fotfile /dev/nrmt0l XX 0 1-2 1-99 def                            start pass 1
   Tape drive :  /dev/nrmt0l
Rewinding tape...                                             temp.com files the tapedir
Processing ./Fotx/saxfot.tapedir
0+1 records in
11+1 records out
Rewinding tape...

  PASS 2 starting ...                                                     start pass 2
   Tape drive :  /dev/nrmt0l                                     retrieved from pass 1
   Experiment :  XX                                          retrieved from run string
  Select observation ranges to be filed as follows :
     1-99 or return to file all observations
     0              to file none of them
     1+3+7          to file obs. 1,3 and 7
     2-5            to file obs. from 2 to 5
     1+3-6+9        to combine ranges and single obs.
  NB : separate selection for pointing and slews !
   Initial SLEW observations to be filed :  0
   TARGET       observations to be filed :  1-2
   Final   SLEW observations to be filed :  1-99
  Select data types to be filed as
     ALL      =A+B+C+D+E+F
     DEFault  =A+B+D+E+F
     A        tapedir only
     B        orbit/attitude/obt+utc
     C        spacecraft HK
     D        instrument directories
     E        instrument HK
     F        instrument science data
  or a combination of the form C+E+F
  or a negative form like NOB
  or a combination like DEFAULT+NOB+C
   Data types to be filed :  def

  File            1 tapedir   rejected :           1
  File            2 ephemeris taken
  File            3 attitude  taken
  File            4 obt_utc   taken
  File            5 ithkd000  rejected :           1
  File            6 tchkd000  rejected :           2
  File            7 instdir   taken
  File            8 obsdir    rejected :           1
  File            9 expconf   rejected :           2
  File           10 w1hkd000  rejected :           3
  File           11 w1eng000  rejected :           4
  File           12 w1dir001  rejected :           5
  File           13 obsdir    taken
  File           14 expconf   taken
  File           15 w1hkd000  taken
  ....                                                              omissis
  File           27 w1dir001  taken
  File           28 instdir   rejected :           1
  File           29 obsdir    rejected :           2
  ....                                                              omissis
  File           43 w2dir002  rejected :          16

Rewinding tape...                              temp.com filing all requested files
Skipping 0001 files...
Processing ./Fotx/saxfot.ephemeris
1+0 records in
46+1 records out
Processing ./Fotx/saxfot.attitude
1+0 records in
31+1 records out
Processing ./Fotx/saxfot.obt_utc
1+0 records in
8+1 records out
Skipping 0002 files...
Processing ./Fotx/w1.instdir
1+0 records in
1+1 records out
Skipping 0005 files...
Processing ./Fotx/nw1001.obsdir
  ....                                                              omissis
Processing ./Fotx/fw1001.obsdir
  ....                                                              omissis
Processing ./Fotx/fw1001.w1dir001
2+1 records in
128+1 records out
Rewinding tape...

  PASS 3 starting ...                                               start pass 3
  File saxfot.tapedir   verified
  File saxfot.ephemeris was filed INCORRECTLY !
  DEBUG nrec recl exp siz         130         184       24050       24048
  File saxfot.attitude  was filed INCORRECTLY !
  DEBUG nrec recl exp siz          90         176       15930       15928
  File saxfot.obt_utc   was filed INCORRECTLY !
  DEBUG nrec recl exp siz         100          40        4100        4098
  File saxfot.ithkd000  was not filed
  File saxfot.tchkd000  was not filed
  File w1.instdir   verified
  File iw1001.obsdir    was not filed
  ....                                                              omissis
  File nw1001.obsdir    verified
  ....                                                              omissis
  File fw1001.w1dir001  verified
  File w2.instdir   was not filed
  ....                                                              omissis
  File fw2001.w2dir002  was not filed
temp 15 >
```