SIMANA36 User Guide for Basta & CSF under AFS

Version 1.10 (15/04/97)

G.Grosdidier

1 Introduction

Since the use of DELSIM for production and code development is now mainly based on Unix RISC workstations, a user facility called **SIMANA36** has been developed and installed on two of the DELPHI main simulation clusters, Basta (in Lyon), CSF (at CERN) both running under AFS.

This facility, which integrates both DELSIM and the associated DELANA for simulated data processing, allows to produce Monte-Carlo data up to the DST writing in the same batch job. The latter is launched by a single command described in details inside of this note, and must be run preferably in batch.mode. It is nevertheless possible to use the same command in interactive mode when debugging, either for Delsim or for Delana, is required.

It is also very useful to Physics Team people who need to test and produce special samples by their own, since it is even possible to recompile Delsim in the same production job for introducing new user generators, or provide new event filters. In addition, these tests could hopefully be achieved before sending simulation requests to the production centers.

It must be underlined that this facility is heavily used by the standard production: so it is really safe, since most of the problems encountered in running the jobs have been carefully cured.[1]

2 Aim of the SIMANA36 tool

◆ To provide Physicists with an exhaustive tool for producing their own specific channel with their own generator, or filter.

◆ To provide Detector Developers with a complete environment allowing for testing their new tuning or software or parameter values (programs or database).

◆ The environment on CSF is strictly identical to the one available on Basta for installing, building and testing a new version. So, it really offers all the required facilities.

◆ The current setup, installed and updated regularly via FTP mirroring, allows:

- to run on BASTA, CSF, both under AFS, with the same set of commands and tools
- to run on HP-UX 9000, IBM RS 6000 or Alpha OSF1 (D Unix) stations
- to run in Batch or Interactive mode
- to run in debug or standard mode
- to run SIM & ANA, or SIM or ANA separately
- to compile libraries (installators only) or create executables, as well as running them
- to run with different versions of DELSIM or DELANA[2]
- to use a lot of different standard models/generators

---

[1]New items have been flagged by a vertical bar in the left margin.
[2]See Annex 1 for more details

3 The file organization

3.1 Directories

The picture on page 5 displays the current directory structure.

◆ From now on, the whole DELSIM stuff is housed in the cern.ch AFS cell, in directories like:

- `/afs/cern.ch/delphi/`*`platform`*`/delsim`

where *`platform`* is either an OS dependent directory, or `share`, for e.g. source code.

❑ The root directory is machine dependent and shows one subtree for each available **platform**:

HP-UX	`/afs/cern.ch/delphi/hp700_ux90/delsim`
IBM RS	`/afs/cern.ch/delphi/rs_aix32/delsim`
Alpha OSF	`/afs/cern.ch/delphi/alpha_osf1/delsim`

However, the machine independent areas, common to all platforms, are available in the

`/afs/cern.ch/delphi/share/delsim` branch.

In addition, there is now a `/share` tree for the common (machine independent) areas.

❑ Each of these is divided into 3 **project** subdirectories:

`/sim`, `/ana` and `/ddb`.

❑ The next subtree is dedicated to the **version**. Each of the 3 above is divided into:

`/v91_1, /v91_2, /v91_3` (reprocessing),	dropped
`/v92_1, /v92_2,`	on Basta, AFS & CSF
`/v93_s` (scan),	dropped
`/v93_1, /v93_2` (reprocessing),	on Basta, AFS & CSF
`/v94_1, /v94_3` & `/v94_4` (reprocessings),	on Basta, AFS & CSF
`/v95_1, /v95_3` & `/v95_4` (reprocessings),	on Basta, AFS & CSF
`/v96_0,`	dropped
`/v96_2, /v96_3` (reprocessing),	on Basta, AFS & CSF
`/v97_1,`	on Basta, AFS & CSF

In all this setup, in order to save disk space, software links often points to other ares, but one can always address a given version using its full standard name as indicated above.

❑ Finally, each version subdirectory contains the following **working** areas
(this is the deepest level of the tree):

- `/car, /cra, /data, /usr.` machine independent areas

(these areas also exist in the machine dependent trees, but they are only software links to the above)

- `/src, /exe, /lib, /bin, /log` machine dependent stuff

What are exactly their contents ?

`/car`	card PAMs in ASCII format	(extension .car)
`/cra`	cradles for creating libraries and executables	(extension .cra)
`/data`	title files and ASCII input data	
`/usr`	binary files for the DDAPP database (`/ddb` tree only)	
`/lib`	libraries	(extension .a)
`/exe`	executables	
`/src`	subdirectories containing the Fortran source files	(extension .f)

(this directory exists only when in development phase, and removed otherwise to save disk space)

`/bin` unused (temporary area only)

`/log` log files for each step of library or executable creation. For the `/ddb` tree, they also contain the log of the job which created the "dtbzeb" file (preloaded snapshot database file created by DELSIM in Zebra format).

Ex: the latest HP-UX version of the Delsim executable is in

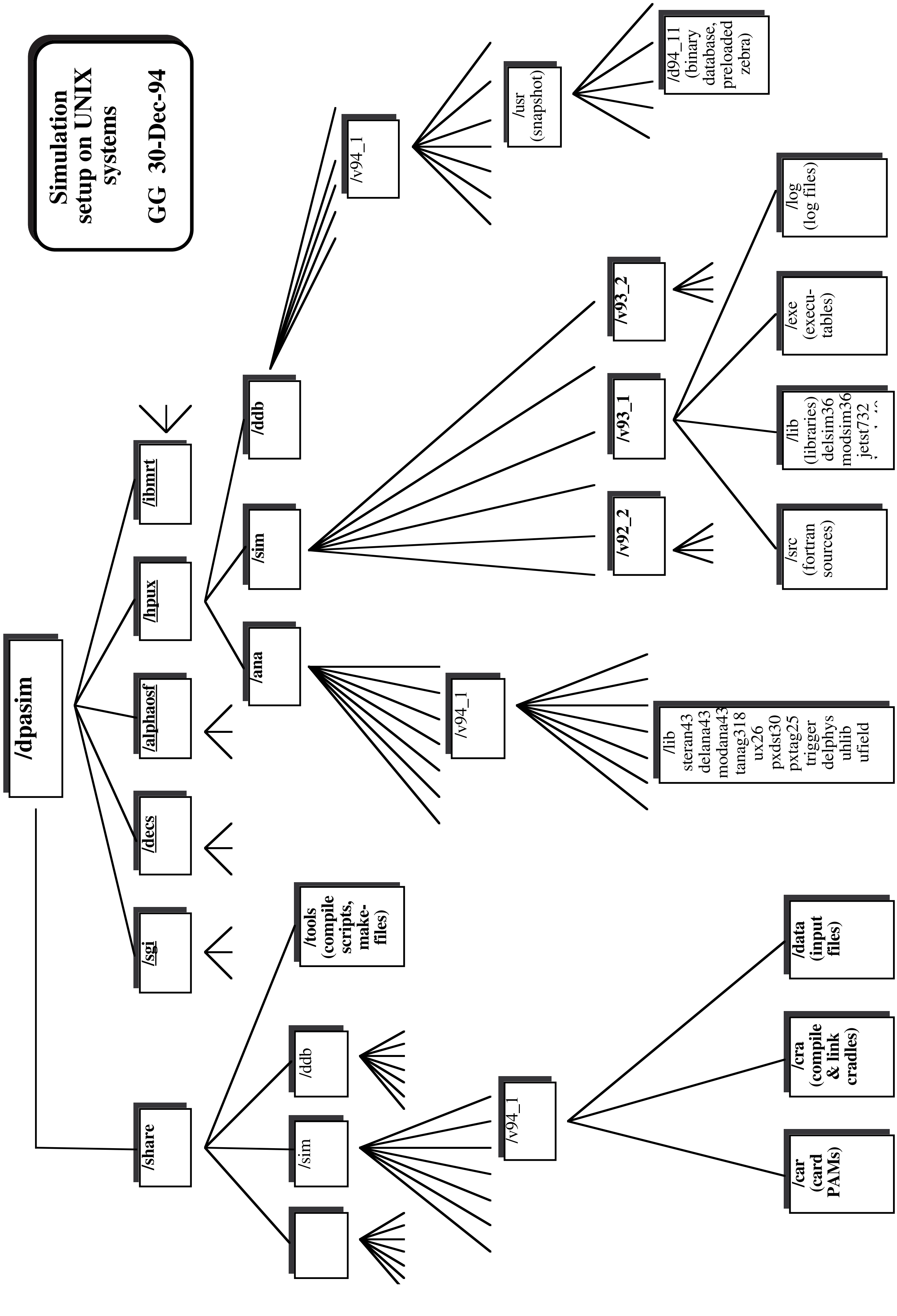
`/afs/cern.ch/delphi/hp700_ux90/delsim/sim/v97_1/exe/simrun36`

❑ An important area is located in ($DELTOL):

`/afs/cern.ch/delphi/share/delsim/share/tools` machine independent utilities

It contains shell scripts like:

- `zfortlib` for patchy preprocessing and compilation of libraries
- `zformake` for patchy preprocessing, compilation and link of executables
- `makefile` which contains the library declarations for every link sequence
- `gom` for navigation between directories (cf below)
- `simana36` for running the production


Simulation setup on UNIX systems
GG 30-Dec-94
/dpasim
/share
/sgi
/decs
/alphaosf
/hpux
/ibmrt
/ana
/sim
/ddb
/v94_1
/usr (snapshot)
/d94_11 (binary database, preloaded zebra)
/v92_2
/v93_1
/v93_2
/src (fortran sources)
/lib (libraries) delsim36 modsim36 jetst732
/exe (execu-tables)
/log (log files)
/v94_1
/lib steran43 delana43 modana43 tanag318 ux26 pxdst30 pxtag25 trigger delphys uhlib ufield
/tools (compile scripts, make-files)
/ddb
/sim
/v94_1
/car (card PAMs)
/cra (compile & link cradles)
/data (input files)

3.2 **Navigation** (across directories)

◆ There are 2 methods:

➪ the standard way is available, of course, via the Unix "`cd`" command:

e.g. `cd /afs/cern.ch/delphi/rs_aix32/delsim/sim/v94_4/lib`

➪ However, I strongly recommend to use the "`gom`" tool provided in the `$DELTOL` directory:

- first install it in your `.login` file as explained in chapter 5 of this note
- then if you type `gom`
- and answer something like `sim lib v94_4 rs`

you will access the same subdirectory as above.

➪ The following keywords can be used for each of these 4 arguments:

project	**working**	**version**	**machine**
sim	src	...	hp
ana	bin	v93_1	rs
ddb	lib	v93_2	al
	car	v94_4	osf
	cra	v95_3	(sgi)
	exe	v96_3	(sun)
	data	v97_1	...
	usr	...	
	log		

◆ Remarks:

– the fourth argument is optional. It defaults to “`hp`” on HP-UX machines, to “`rs`” on IBM RS stations, and to “`al`” or “`osf`” on Alpha OSF1 (D Unix) platforms.

– the meaning of the other three args are easy is quite straigthforward

– the third one defaults to the current value when not provided

– the second one defaults to log when not provided

◆ Last but not least, this tool sets up a lot of environment variables:[1]

- $DELSHAR, pointing to the root of the `share` area
- $DELMACH, pointing to the root of the machine dependent area
- $DELTOL = `$DELSHAR/tools`
- $SIMCOM, which houses the library of filters (cf option `-FILT`)
- $SIMCAR, $SIMEXE, $SIMDAT, and so on,
- $ANACAR, $ANAEXE, $ANADAT, ...,
- $DDBCAR, $DDBEXE, $DDBUSR, ...,

accordingly to the version you have selected (v95_4, v97_1, ...).

◆ But, in fact, all of these is required only to inspect or fetch something inside of the setup, not to run the `simana36` facility itself. In most cases, the use of the gom tool is only required, for a normal user, to set up the value of the DELTOL variable.

[1]See Annex 2 for an exhaustive list.

4 Description of the SIMANA36 command

4.1 Executing SIMANA36

The “**simana36**” tool is located in the $DELTOL subdir.

There are 2 ways for using it, after having run the "`gom`" command to set up the DELTOL variable:

4.1.1 ◆ Interactively (not recommended), typing the command:

```
“$DELTOL/simana36 \
          nrun nevmax rawtape model quark –VERS setup [other_args]”
```

will start execution in the current directory. Since many files and pointers will be created, it is safest to do it in an empty subdirectory.

Ex: “`$DELTOL/simana36 \`

```
  5836 20 ex5678 mtrx 4 –V941 gas –DST ex1234 –FILT sxcmif.f –SCRA stan”
```

Apart from “`other_args`”, the order of arguments is **mandatory** (these are positional arguments).

4.1.2 ◆ In **batch**, typing:

```
“$DELTOL/subloop \
 queue taprefix first last model nevmax quark –VERS setup [other_args]”
```

will submit one or more jobs in the CSF queues.

The log file(s), together with other files you will ask for through the options, will be written in your current directory (submit directory) at the end of the job. It can easily be inspected during execution using the `qcat` command. The log file name will be of the form "`job_name.ojob_id`", job_id being a 4-5 decimal digit request number.

Ex: “`$DELTOL/subloop M ex 5678 5684 show 360 5 –V953 liq –DST ex1234`”

will submit

- in the M (Middle) queue
- 7 jobs named ex5678 up to ex5684
- the VID of the Raw Data cartridge will be the same as the job name
- the run numbers will range from 5678 up to 5684
- each job will produce 360 PS $b\bar{b}$ evts
- they will be simulated with the v95_1 simulation version with liquid setup, and processed with DELANA version 95C
- the 7 DST files will be appended to the same ex1234 cartridge.

4.2 Arguments

4.2.1 ◆ The meaning of the **positional arguments**:

- *nrun* — run number, positive integer
- *nevmax* — amount of evts to simulate
  Ex: 360 (for a $q\bar{q}$ run)
- *rawtape* — VID of the Raw Data type to be written (always on first file)
  Ex: ex9876
- *model* — generator keyword, explained below
  Ex: show (for PS $q\bar{q}$)
- *quark* — either 0 for all quarks, or 1/2/3/4/5 for u/d/s/c/b in Lund
  Ex: 0 will produce mixed $q\bar{q}$ evts
  4 will produce only $c\bar{c}$ evts
- *– VERS setup* — the version flag together with one of liq/gas/sca for ***liq***uid/***gas***/***sca***n setups
  Ex: –V953 liq

Warning: The setup value is restricted to "liq" from '94 ownwards.
Warning: The following versions of Delsim + Delana are currently available through this facility:

- *v92_2* allowed through `-V922 gas` or `-V922 liq`
- *v93_2* allowed through `-V932 gas` or `-V932 liq`
- *v94_4* allowed through `-V944 liq` only
- *v95_4* allowed through `-V953 liq` only (soon)
- *v96_3* allowed through `-V963 liq` only
- *v97_1* allowed through `-V971 liq` only

- *queue* — one of **S/M/L** for small/middle/large batch queues on **CSF**
  one of **A/G/I** for small/middle/large batch queues on **Basta**
- *taprefix* — the 2 (resp. 4) first characters of the raw data tape VID on CSF (resp. Basta)
  Ex: ez (on CSF), ex56 (on Basta)
- *first* — the last 4 (resp. 2) numeric digits of the VID
  of the first raw data tape of the serie on CSF (resp. Basta)
  Ex: 9876 (on CSF), 78 (on Basta)
- *last* — the same for the last tape of the serie
  Ex: 9882 (on CSF), 84 (on Basta)

NB: there will be one job per run, one run per tape, and as much jobs as tapes in the range [first, last]

- *other_args* optional arguments, will be detailed in subsection **4.2.3**

They are all of the form "`–KEYWORD values`", with:

- "–KEYWORD" in uppercase
- "values" one or several lowercase values separated by spaces

Ex: "`–V944 liq`" as above for the setup definition
"`–SCRA $cwd/my_simrun49.cra`"

4.2.2 Details about the generators

◆ List of available generators prewired in the SIMANA36 script

(cf above the `model` positional argument)

- show — standard PS $q\bar{q}$ (IGENER=15)
- mtrx — matrix element $q\bar{q}$ (probably obsolete)
- baba — barrel Bhabhas "à la Mustraal" (IGENER=14)
- bafo — forward Bhabhas (cf above)
- nunu — $\nu\bar{\nu}$ "à la Mustraal" (ν_e)
- mumu — $\mu^+\mu^-$ "à la Mustraal"
- toto — $\tau^+\tau^-$ "à la Mustraal"
- hphm — Higgs generation with IGENER=3 (cf `–XMH`)
- higgsbb — Higgs generation with IGENER=2 (cf `–XMH`)
- kora — $\tau^+\tau^-$ "à la KORALZ", with external generator
- dymu — $\mu^+\mu^-$ "à la DYMU3", with external generator
- h2h3 — special external Higgs generator
- gext — use of IGENER=0, with external Lund file (provided through the `–GEXT` option)
- fext — use of IGENER=0, with external Fortran, data & library files (provided through the `–FEXT, –DEXT & –LEXT` options)
- user — use of IGENER=99, with internal user generator (provided through the Delsim cradle, see `–SCRA`)

Other models are available, of less general use. They will not be explained here (on request only).

4.2.3 The optional arguments

◆ What are the optional arguments ? Several types are available.

They are all of the form: "`–KEYWORD values`", with:

✜ "–KEYWORD"	in uppercase
✜ "values"	one or several lowercase values separated by spaces

They can be provided in any order.

4.2.3.1 ➪ The **technical** options are:

- `-WDBZ`

  writes the Zebra preloaded database file in your home directory, instead of reading the standard one in the official areas (useful with –VIRT option, cf below).

- `-DST tapedst`

  the DST output file is written at the end of the existing dataset on tape which VID is "tapedst". This requires special initialization of the tape if first file (CSF only).

  Ex: –DST ex1234

- `-NOWSIM`

  the Raw data file is not written down on tape (default: writes it).
  To be used only with –DST above.

- `-WANA`

  write DST data even if some incident prevents DELANA processing to be achieved on the whole DELSIM file (default: not written if less than 90% of the evts are processed).

- `-RNDM`

  the random seeds are not any more connected to the run number (default: connected).

- `-SEED numevt numlog`

  expert use only (allows to reproduce the evt # "numevt" of a previous execution logged in file "numlog" in your home directory).

  Ex: –SEED 325 ex9875.o3428

- `-REPT jflrep`

  expert use only (allows to cycle twice among a serie of "jflrep" evts to check the reproducibility of simulation).

  Ex: –REPT 5

4.2.3.2 ➪ Options dedicated to **setting of physical parameters** in the simulation step:

- `-EBEA ebeam`
  beam energy value (floating, defaults to 45.625, in GeV).

  Ex: –EBEA 92.0

- `-XMH xmh`
  Higgs mass value for several models (defaults to 10.0).

  Ex: –XMH 12.5

- `-FILT filter`
  this one allows to provide in the “filter” file the Fortran code to select special evts (using COMIS). You can find examples of these filters used for production in the “`/afs/cern.ch/delphi/share/delsim/com`” directory ($SIMCOM).

  Ex: –FILT $YOUR_HOME/gen/my_own_generator.f

Warning: the routine name (SXCMIS), number and type of arguments (only one integer argument), and name of accessed common (LUJETS) are **mandatory**. If you really want to modify one of these, you will need to also recompile (cf –SCRA below) the routine P=UTILITY, D=SXGMIS in Pam DELSIM36.

- `-GEXT gextfile`

  absolute pathname of the Lund file containing the pregenerated evts in case of IGENER=0 (gext model).

  Ex: –GEXT $home/lund_64.dat

To run with the `fext` model (external generator provided through a Fortran file), the following 3 options are available (first 2 are mandatory):

- `-FEXT fextfile`

  absolute pathname of the Fortran file containing to be compiled, linked and ran as an external generator. It must read Input data through Fortran unit 19 and write Event data in Lund format on unit 19 using the SXWRLU routine available in the DELSIM package.

  Ex: –FEXT $this_home/twogam.f

- `-DEXT dextfile`

  absolute pathname of the Data file used at run time to initialize the external generator above (to be linked to Fortran unit 19 in your Fortran).

  Ex: –DEXT $this_home/twogam.tit

- `-LEXT lextfile`

  absolute pathname of the Library file containing the utility routines to be linked together with the Fortran file `fextfile` described above when running the SIMANA36 script.
  When not provided, this library defaults to the Delphi private `jetset` library.
  In all cases, the program is always linked against a subset of the CERN libraries (`mathlib`, `kernlib`).
  This option is expected to be repeatable: when it is required to provide several such libraries, you must provide them through several "-LEXT lextfile" bunches as shown in the 2nd example below.

  Ex: –LEXT $AFS_HOME/public/tbdklib.a
  –LEXT $YOUR_HOME/libtwogam.a -LEXT $OTHER_HOME/libjetset.a

Warning: when using -LEXT, you must take care that such libraries are machine dependent. This option, in this regard, is not yet very well implemented.

4.2.3.3 ➪ Of standard use in case of **testing** :

- `-SIM`

  the Delsim step only is achieved.

- `-STST`

  the Delsim output file is not even created (to use with `-SIM` above).

- `-ATST`

  the Delana output file is not even created (null file), and the Delsim one is not written on tape.

4.2.3.4 ➪ Dedicated to **DELANA processing**:

- `-LANA tapein`

  only the Delana step is achieved, and the Raw input file is fetched from tape with VID "tapein".

  Ex: –LANA ex1836

- `-ANA anafile`

  same as above, but the input evts are read in the file "anafile" in your home directory.

  Ex: –ANA short_raw_file.dat

- `-RANA`

  same as -LANA, but the input tape number is taken from the Raw Data tape one (used in reprocessing mode).

- `-FSEQ fileseq`

  when the Raw Data are written on a multiple file tape (high density tapes), this option allows to provide the file sequence of the target file on the input tape when doing reprocessing.

  The syntax of fileseq can be:

`-FSEQ 7`	for a single file
`-FSEQ 7,8,10,12`	for isolated files
`-FSEQ 7-15`	for a string of contiguous files

  but mixing of the 3 above is not allowed.

4.2.3.5 ➪ About the **Title files**:

- `–STITL titfile`
  you can provide in "titfile" your own complete Delsim command file. Examples can be found in the $SIMDAT directory, or using the "`–PUT stit`" option (cf below).

  Ex: –STITL $cwd/frich_liq.special_titles

- `–ATITL titfile`
  same facility for Delana.

- `–STITA titfile`
  you can provide an additional title file for running Delsim. It will be appended to the standard one, and will allow you to provide additional keywords, or to overwrite the standard values of already existing ones.

- `–ATITA titfile`
  additional title file facility for Delana.

- `–SSTRG additional_string`
  you can provide an additional title string (a single line) for running Delsim. It will be appended to the standard one, and will allow you to provide additionnal keywords, or to overwrite the standard values of already existing ones. Warning: one MUST provide the whole command as a single word, the spaces being replaced with underscores. This is a repeatable option.

  Ex: –SSTRG JFLPRM_TRUE

- `–ASTRG additional_string`
  additional title string facility for Delana.

  Ex: –ASTRG NEVMAX_200

4.2.3.6 ➪ When you will have to **compile or test a new version**:

- `–SCRA sim_cra`

  a temporary Delsim executable is created using the cradle “sim_cra”, and used to achieve the run. The compilation is achieved in debug mode (standard –g option on any Unix machine), allowing for further interactive debugging. If “sim_cra” is set to “`stan`”, the standard cradle of the version is used, but the resulting executable is compiled in debug mode, on the contrary to the standard one.

  Ex: –SCRA $cwd/simrich.cra
  –SCRA stan

- `–ACRA ana_cra`

  same thing as above, but for the Delana executable. they can be used simultaneously.

  Ex: –ACRA anatest.cra

- `–VIRT virtfile`

  you can provide in “virtfile” an additional ASCII virtual database file to be added to the standard snapshot. To be used in conjunction with “–TITL” option above in case you need to modify the dates, and also “–WDBZ”, may be. Here again, there is no such facility for Delana.

  Ex: –VIRT $MY_HOME/hactest/hacsim_calb.ascii

- `–MAP`

  when this option is used with one (or both) of the 2 above, a load map of the corresponding executable is written down in your home directory, with the name “sim_cra.something” or “ana_cra.something”. To be used in interactive mode only.

4.2.3.7 ➪ Dedicated to **debugging**:

- `-PUT put_list`

  used together with –ACRA or –SCRA, this option allows you to get several files written down into your submit directory during execution. The "put_list" is a space-separated list of any number of the following keywords (in lowercase):
  `scra, sfor, sexe, stit, sdat, & acra, afor, aexe, atit, adat`
  where:
  - the s/a prefix stands for sim/ana
  - cra: the cradle you have been using for compilation
  - for: the resulting Fortran file
  - exe: the executable used or created
  - tit: the title file used for executing the above
  - dat: the evt output file

  Ex: –PUT scra sfor

- `-STOPS`

  requires the execution of "`simana36`" to stop just after the Delsim executable is created, and the environment is set up. This allows to start interactive debugging on the executable program.

- `-STOPA`

  same thing as above, but for Delana

4.2.3.8 ➪ Other options (like `-VSIM, -VANA, -VDDB`) which are useful for running production and for debugging, but of very little use for standard customers, will not be detailed here.

5 Additional comments

◆ About **filenames**, a general recommendation could be: whenever you have to provide a filename, better to give the absolute pathname, like "$cwd/yourfile", or "/afs/cern.ch/user/y/yourname/public/myfile". Please read carefully the examples.

◆ About the SIMANA36 **C-shell script**:

➪ You will probably need to understand in deeper details what is done in "simana36". Please don't hesitate to print the script file and read it prior to ask any question!

➪ If you really need to modify it to add some new feature(s) or models for example, you can obviously do it as well on your private copy, but do not modify or patch other parts of this script file: this would make it very uneasy to debug afterwards, and I would probably be unable to help you in case of troubles. Instead, if you discover some bug, please mail it to me (not too often !).

◆ **Warnings**

➪ The scripts and files provided in the /delphi area are obviously subject to change without notice. But : the functionalities explained above will be supported whenever possible !

◆ **Helps**

➪ A postscript version of this document will be kept and updated as often as possible in the $DELTOL directory, in the file simana36_help.ps. It is now also available via W3, under the Delphi Off-Line page.

➪ The particle list (Delsim & Lund) is available in the $SIMDAT area for a given version under filename particle.list_v941 for example. It includes the branching ratios and decays, as obtained through the JFLPRM flag of Delsim.

◆ Some tricks

➪ Try to put somewhere in your `.login` file (if you use C-shell or tcsh) the few following lines:

- at CCIN2P3

```
if ( $SERVICE == 'BASTA' || $SERVICE == 'BAHIA' ) then
  setenv DELTOL '/afs/cern.ch/delphi/share/delsim/tools'
  set path = ( $path $DELTOL )
  alias gom "source $DELTOL/gom"
endif
```

- at CERN

```
if ( $SERVICE == 'hpplus' || $SERVICE == 'cernsp' || \
             $SERVICE == 'dxplus' || $SERVICE == 'csf' ) then
      setenv DELTOL '/afs/cern.ch/delphi/share/delsim/tools'
      set path = ( $path $DELTOL )
      alias gom "source $DELTOL/gom"
endif
```

Annex 1

Available Simulation Samples.

Simulates	Delsim version	Started	First Delana	Reprocessed Delana	Raw Data
'91 data	v91_1	Jul 91	91_D	Discarded	Scratched
'91 data	v91_2	May 92	91_E	91_F (Apr 93)	Scratched
'92 data	v92_1	Dec 92	92_C	Discarded	Scratched
'92 data	v92_2	Jun 93	92_D (May 93)		TO BE KEPT
'93 scan	v93_s	Aug 93	93_B (Aug 93)		??
'93 data	v93_1	Jan 94	93_B (Nov 93)	93_C (Jul 94)	TO BE KEPT
'94 data	v94_1	Dec 94	94_B (Nov 94)	94_D (Dec 96)	TO BE KEPT
'95 data (P1+P2)	v95_1	Sep 95	95_A (Jul 95)	95_D (May 97 ?)	TO BE KEPT
'95 data (P3)	v96_0	May 95	96_Z (May 95)	Discarded	
'96 data	v96_2	Apr 96	96_A (Apr 96)	96_E (Jul 97)	TO BE KEPT
'97 data	v97_1	Mar 97	97_A (Mar 97)		TO BE KEPT

Annex 2

Environment variables as set up by "`gom`" on CSF, under AFS (HP nodes).

```
SIMCOM=/afs/cern.ch/delphi/share/delsim/sim/com
DELSHAR=/afs/cern.ch/delphi/share/delsim
DELMACH=/afs/cern.ch/delphi/hp700_ux90/delsim
DELSIM=/afs/cern.ch/delphi/hp700_ux90/delsim/sim/v97_1
SIMUSR=/afs/cern.ch/delphi/hp700_ux90/delsim/sim/v97_1/usr
SIMLOG=/afs/cern.ch/delphi/hp700_ux90/delsim/sim/v97_1/log
SIMCRA=/afs/cern.ch/delphi/hp700_ux90/delsim/sim/v97_1/cra
SIMDAT=/afs/cern.ch/delphi/hp700_ux90/delsim/sim/v97_1/data
SIMEXE=/afs/cern.ch/delphi/hp700_ux90/delsim/sim/v97_1/exe
SIMOUT=/afs/cern.ch/delphi/hp700_ux90/delsim/sim/v97_1/dataout
SIMMGR=/afs/cern.ch/delphi/hp700_ux90/delsim/sim/v97_1/mgr
SIMSRC=/afs/cern.ch/delphi/hp700_ux90/delsim/sim/v97_1/src
SIMLIB=/afs/cern.ch/delphi/hp700_ux90/delsim/sim/v97_1/lib
SIMCAR=/afs/cern.ch/delphi/hp700_ux90/delsim/sim/v97_1/car
DELANA=/afs/cern.ch/delphi/hp700_ux90/delsim/ana/v97_1
ANACRA=/afs/cern.ch/delphi/hp700_ux90/delsim/ana/v97_1/cra
ANAUSR=/afs/cern.ch/delphi/hp700_ux90/delsim/ana/v97_1/usr
ANALOG=/afs/cern.ch/delphi/hp700_ux90/delsim/ana/v97_1/log
ANAEXE=/afs/cern.ch/delphi/hp700_ux90/delsim/ana/v97_1/exe
ANADAT=/afs/cern.ch/delphi/hp700_ux90/delsim/ana/v97_1/data
ANAMGR=/afs/cern.ch/delphi/hp700_ux90/delsim/ana/v97_1/mgr
ANASRC=/afs/cern.ch/delphi/hp700_ux90/delsim/ana/v97_1/src
ANALIB=/afs/cern.ch/delphi/hp700_ux90/delsim/ana/v97_1/lib
ANACAR=/afs/cern.ch/delphi/hp700_ux90/delsim/ana/v97_1/car
DDAPP=/afs/cern.ch/delphi/hp700_ux90/delsim/ddapp/rdb
DELDDB=/afs/cern.ch/delphi/hp700_ux90/delsim/ddb/v97_1
DDBLOG=/afs/cern.ch/delphi/hp700_ux90/delsim/ddb/v97_1/log
DDBCAR=/afs/cern.ch/delphi/hp700_ux90/delsim/ddb/v97_1/car
DDBEXE=/afs/cern.ch/delphi/hp700_ux90/delsim/ddb/v97_1/exe
DDBDAT=/afs/cern.ch/delphi/hp700_ux90/delsim/ddb/v97_1/data
DDBCRA=/afs/cern.ch/delphi/hp700_ux90/delsim/ddb/v97_1/cra
DDBMGR=/afs/cern.ch/delphi/hp700_ux90/delsim/ddb/v97_1/mgr
DDBSRC=/afs/cern.ch/delphi/hp700_ux90/delsim/ddb/v97_1/src
DDBLIB=/afs/cern.ch/delphi/hp700_ux90/delsim/ddb/v97_1/lib
DDBUSR=/afs/cern.ch/delphi/hp700_ux90/delsim/ddb/v97_1/usr
SUBPRO=v97_1
DELROT=sim
PROJET=/afs/cern.ch/delphi/hp700_ux90/delsim/sim/v97_1
DELTOL=/afs/cern.ch/delphi/share/delsim/tools
```

Annex 3

Example of a Fortran Filter routine

```
CKEEP, SXGSELBODY, IF=TEAM11.                                 rou 03/12/90

      SUBROUTINE SXCMIS (IUSEIT)

      COMMON/LUJETS/N,K(4000,5),P(4000,5),V(4000,5)

***********************************************************************
*  INPUT :
*  -----
*   ITYPLP = LUND KF CODE OF THE DESIRED LEPTON (13=MU  11=E )
*   ITYPSL = TYPE OF SELECTION ( 1=DIRECT LEPT FROM B
*                                2=DIRECT LEPT FROM C IN CCBAR EVTS
*                                3=DIRECT LEPTON FROM C IN BBAR EVTS )
*
*  OUTPUT:
*  ------
*   IUSEIT=0    EVT REJECTED
*         =1    EVT ACCEPTED
***********************************************************************
*   DO NOT FORGET TO CHANGE THE CARD FFREAD : NGMAX 10000
*  IF NOT THE PROGRAM WILL STOP AT THE 1ST EVT(NGMAX IS THE MAX NB OF
*  EVTS WHICH CAN BE REJECTED BEFORE YOU KEEP A GOOD ONE )
***********************************************************************
      IUSEIT=0
***********************************************************************
*   LUND KF CODE FOR THE SELECTED LEPTON ( 13=MU 11=E )
      ITYPLP= 13
*   EVTS SELECTION CATEGORY (1=DIRECT L. FROM B , 2=DIRECT L.FROM C
*                            3=DIRECT L. FROM C IN BBAR EVTS)
      ITYPSL=  1
*********************************************************************
      IF(N .LE. 3) RETURN
*  SEARCH FOR A MUON
      DO 1000 IMU=1,N
      IF(IABS(K(IMU,2)) .NE. ITYPLP ) GO TO 1000
      IPAREN=K(IMU,3)
      IF(IPAREN .LE. 0) GO TO 1000
      ITYPAR=IABS(K(IPAREN,2))
*  DIRECT LEPT FROM BBAR EVTS
      IF(ITYPSL .EQ. 1) THEN
*
*  IS THE PARENT A PSI PARTICLE ?
        IF(ITYPAR .EQ. 443) THEN
          IPAR2=K(IPAREN,3)
          IF(IPAR2 .LE. 0) GO TO 1000
          ITYPA2=IABS(K(IPAR2,2))
*    IS THE PARENT A B MESON ?
          IF(ITYPA2.EQ.511 .OR. ITYPA2.EQ.521 .OR. ITYPA2.EQ.531) THEN
            IUSEIT=1
            GO TO 1001
          ELSE IF(ITYPA2.EQ.5122 .OR. ITYPA2.EQ.5132 .OR. ITYPA2.EQ.5232
     +   .OR. ITYPA2.EQ.5332) THEN
```

```
            IUSEIT=1
            GO TO 1001
          ENDIF
          GO TO 1000
*  IS THE PARENT A STABLE B PARTICLE?
        ELSEIF(ITYPAR.EQ.511 .OR. ITYPAR.EQ.521 .OR. ITYPAR.EQ.531) THEN
          IUSEIT=1
          GO TO 1001
        ELSE IF(ITYPAR.EQ.5122.OR.ITYPAR.EQ.5132 .OR. ITYPAR.EQ.5232
     +    .OR. ITYPAR.EQ.5332) THEN
          IUSEIT=1
          GO TO 1001
        ENDIF
*  DIRECT LEPTON FROM CHARM (INCLUDING LEPTONS FROM TAU )
      ELSE IF (ITYPSL .GE. 2) THEN
        ICHARM=0
        IF(ITYPAR .EQ. 15) ICHARM=1
        IF(ITYPAR.EQ.411 .OR. ITYPAR.EQ.421 .OR. ITYPAR.EQ.431) ICHARM=1
        IF(ITYPAR.EQ.4122.OR. ITYPAR.EQ.4132.OR. ITYPAR.EQ.4232
     + .OR. ITYPAR.EQ.4332) ICHARM=1
        IF( ICHARM .EQ. 0) GO TO 1000
*  THE LEPTON COMES FROM CHARM ; LOOK IF IT COMES ALSO FROM B
        IBEAUT=0
1003    IPAR2=K(IPAREN,3)
        IF(IPAR2 .LE. 0) GO TO 1002
        ITYPA2=IABS(K(IPAR2,2))
*    IS THE PARENT A B MESON ?
        IF(ITYPA2.EQ.511 .OR. ITYPA2.EQ.521 .OR. ITYPA2.EQ.531) IBEAUT=1
        IF(ITYPA2.EQ.5122 .OR. ITYPA2.EQ.5132 .OR. ITYPA2.EQ.5232
     +    .OR. ITYPA2.EQ.5332)IBEAUT=1
        IF(IBEAUT .EQ. 1) GO TO 1002
        IPAREN=IPAR2
        GO TO 1003
1002    IF(ITYPSL .EQ. 3) THEN
*  SELECT LEPTONS FROM D FROM B
          IF(IBEAUT .EQ. 1) THEN
            IUSEIT=1
            GO TO 1001
          ELSE
            GO TO 1000
          ENDIF
        ELSE IF (ITYPSL .EQ. 2) THEN
*  SELECT LEPTONS FROM D FROM C
          IF(IBEAUT .EQ. 1) THEN
            GO TO 1000
          ELSE
            IUSEIT=1
            GO TO 1001
          ENDIF
        ENDIF
      ENDIF
1000  CONTINUE
1001  CONTINUE
***********************************************************************

      END
```

Index

Table of contents