ALICE/SIM 95-44
Internal Note
November 1995

GALICE

The Geant based ALICE detector simulation package

N.J.A.M. van Eijndhoven

Department of Subatomic Physics, Utrecht University/NIKHEF
P.O. Box 80.000, NL-3508 TA Utrecht
The Netherlands

November 19, 1995

Abstract

A description of the main structure of the GALICE detector simulation package is presented together with some simple examples of how to install and run the program.
This report is intended to provide the necessary information for people who want to run the simulation for detector performance studies and to provide the guidelines for those who want to contribute code to the GALICE package in view of detailed detector description.

Contents

1 Introduction

The GALICE detector simulation package is a standalone program based on GEANT3 [1] and is intended to serve detector design- and feasability studies. Both single particles (to enable testbeam simulations) as well as complete physics events generated by any of the event generators from the ALICE pool [2] can be used as input. Using the produced output of detector signals (see hereafter), DAQ design and trigger studies may be performed.

The package is maintained in CMZ [3] format and consists of a top level steering structure from which the various detectors of the ALICE setup are invoked as individual modules. Both the code of the steerings as well as the complete simulation code of the various detectors are contained in the single source code file **galice.cmz** which is available from the central ALICE offline program pool **/afs/cern.ch/alice/offline/pams** at CERN.

2 Program structure

2.1 Coding conventions

To be able to make full use of the exisiting GEANT machinery, the GALICE program has been written in FORTRAN. In order to coordinate the source code contributions from the various detector groups and to prevent clashes in the code of such a complex program, a few conventions have been defined. These coding conventions also ensure proper execution of the GALICE program on all CERN supported platforms. In the following these rules will be described.

- Only standard ANSI Fortran 77 is allowed with the following Fortran 90 compatible extensions :
  - Names of SUBROUTINES and VARIABLES may have a maximum of 20 characters.
  - Names of COMMON blocks may have a maximum of 8 characters.
  - Use of DO WHILE ... ENDDO structures is allowed.
  - The use of IMPLICIT NONE is allowed.
- FORTRAN statements have to be written in CAPITALS.
- Comment lines have to start with a capital C or $*$ in column 1, the rest of the comment line may be written in mixed case.
- It is recommended to use explicit type defintion of the various variables (in combination with IMPLICIT NONE).
  However, the names of the variables should obey the implicit type definition, which means :

- INTEGER ⇒ Variable name starts with I-N
- REAL ⇒ Variable name starts with A-H or O-Z

In this way the detector defined space point data (see hereafter) can be directly entered into the SPC data column wise ntuple (CWN) [4] and also the type of routine/function arguments can be checked directly from the variable names by compiler and/or CMZ utilities.

- In case some printout is made, the printed text should contain the name of the routine in which it was produced.
  All printout has to be produced via the PRINT instruction of Fortran 77.

2.2 Naming conventions

In the following the structure and functionality of the main steering routines are described and the interface between the various detector modules is explained.

SXINIT Initialisation at the beginning of a simulation run, called from the main program GALICE.
This routine calls the **name_INIT** routines of the various detectors, where **name** stands for :

- IP : Interaction point
- ITS : Inner tracking system
- MAG : Magnet barrel
- TPC : Time projection chamber
- TOF : Time of flight system
- SCB : Scintillator barrel
- PHOS : Photon spectrometer
- ZDC : Zero degree calorimeters
- FMC : Forward multiplicity counters
- RICH : Ring imaging cherenkov detector

Example : Initialisation of the PHOS in PHOS_INIT

These **name_INIT** routines are meant to perform some detector specific initialisations (if needed) and also the contents of the detector SPC data block for the SPC data CWN (see hereafter) have to be specified here. As SPC data block name, the detector name (i.e. PHOS, ZDC etc.) should be used.
The names of the variables within the SPC data block have to be unique.
To achieve this, each variable name has to have as the last character the detector specific identification number which is defined as :

1 Interaction point (fake planes from steerings)

2 Inner tracking system (ITS)

3 Magnet barrel (MAG)

4 Time projection chamber (TPC)

5 Time of flight system (TOF)

6 Scintillator barrel (SCB)

7 Photon spectrometer (PHOS)

8 Zero degree calorimeters (ZDC)

9 Forward multiplicity counters (FMC)

10 Ring imaging cherenkov detector (RICH)

These detector identification numbers are available as **ID_name** (e.g. ID_TPC) via the steering common block **/SCXDB/** (see later).
Each **name_INIT** routine has to produce a short printout indicating the routine name, so that from the log file of a certain run it can be directly seen which detectors were invoked and correctly initialised.
The **name_INIT** routines don't have any arguments.

SXMATE Defining the various kinds of media for the whole setup, called from SXINIT.
This routine calls the **name_MATE** routines of the various detectors.
The **name_MATE** routines don't have any arguments.

The media identifiers available for the various detectors in the calls to the Geant routines GSMATE or GSMIXT are :

ID=100*J+N

where : J = detector identification number
(ex. ID_name like ID_PHOS etc...)
$0 \leq N \leq 99$
J = 0 for steerings

SXTMED Defining the tracking parameters for the various media of the whole setup, called from SXINIT.
This routine calls the **name_TMED** routines of the various detectors.
The **name_TMED** routines don't have any arguments.

The media identifiers for the various detectors in the calls to the Geant routine GSTMED follow the same convention as for GSMATE.

SXGEOM Defining the geometry of the complete setup, called from SXINIT.
This routine calls the **name_GEOM** routines of the various detectors.
The **name_GEOM** routines don't have any arguments.

In the steering routine SXGEOM a default volume has been defined namely :

ALIC : Rectangular box filled with air, to contain the various detectors.

All other volumes as defined by the various detectors have to have a name starting with a pre-defined character in order to avoid name clashes for the GEANT volumes.
The pre-defined characters are the detector specific identification characters and are defined as follows :

- X Interaction point (fake planes from steerings)
- I Inner tracking system (ITS)
- M Magnet barrel (MAG)
- T Time projection chamber (TPC)
- F Time of flight system (TOF)
- S Scintillator barrel (SCB)
- P Photon spectrometer (PHOS)
- Z Zero degree calorimeters (ZDC)
- C Forward multiplicity counters (FMC)
- R Ring imaging cherenkov detector (RICH)

The interaction point (IP) is located in (0,0,0) in ALIC. All detectors have to be placed (GSPOS) at the correct location inside the ALIC box w.r.t. the IP.

The coordinate system used is the same as the one of the actual ALICE setup. This means :

1. It's a right handed system
2. The X axis is pointing to the centre of the LHC ring
3. The Y axis is pointing upwards (i.e. towards the ceiling)
4. The Z axis is along the beam direction

In case rotation matrices (GSROTM) are used, the rotation matrix identifiers (IROT) have to follow the same convention as for the media identifiers.

SXSENS Defining the sensitive detector elements of the complete setup, called from SXINIT.
This routine calls the **name_SENS** routines of the various detectors.
The **name_SENS** routines don't have any arguments.

The names used in the **name_SENS** routines to define (sets of) sensitive detector elements have to start with the pre-defined character as specified for the geometry volumes.

In case user identifiers (IDTYPE) are used for various detector elements (via the Geant routine GSDET), these identifiers have to follow the same convention as for the media identifiers.

SXDRAW Drawing of the layout of the various detectors, called from SXINIT.
This routine calls the **name_DRAW** routines of the various detectors.
The **name_DRAW** routines don't have any arguments.

In case view banks are used in the name_.... routines, the identifiers of these view banks have to follow the same convention as for the media identifiers. Actual drawing (GDRAW, GDSHOW etc...) may only take place in the **name_DRAW** routines, and the view banks created have to be deleted (GDELET) at the end of the **name_DRAW** routines.

SXTRKI Initialisation before tracking for each track, called from GUTRAK.
This routine calls the **name_TRKI** routines of the various detectors.
The **name_TRKI** routines have 1 argument (IFLAG) to denote primary (IFLAG=1) and secondary (IFLAG=2) tracks.

Note : The **name_TRKI** routines are THE location to reset the detector specific hit statistics arrays for a certain track.

SXSTEP Recording of possible hits after each step, called from GUSTEP.
This routine calls the **name_STEP** routines of the various detectors.
The **name_STEP** routines don't have any arguments.

SXTRKE End statistics after tracking for each track, called from GUTRAK.
This routine calls the **name_TRKE** routines of the various detectors.
The **name_TRKE** routines have 1 argument (IFLAG) to denote primary (IFLAG=1) and secondary (IFLAG=2) tracks.

SXDIGT Digitising and recording of possible hits after each track, called from SXTRKE **for primary tracks only** after the calls to the **name_TRKE** routines.
This routine calls the **name_DIGT** routines of the various detectors ONLY in case that detector has been selected (see later) for writing out the SPC space point data. In the **name_DIGT** routines the detector specific SPC data arrays are filled and written onto the SPC data CWN by means of a call to the standard HBOOK [4] routine HFNTB with the corresponding block **name**.
The SPC data arrays have to be specified in a sequence **+KEEP,char_SPC**. Here **char** stands for the detector specific identification character as specified above.
The sequence **+KEEP,char_SPC** however should actually contain the common **/SCXSCR/** which serves as a scratch space buffer for all detector SPC data. Note that all SPC data consist of INTEGER values. As an example the PHOS SPC data structure can be looked at in the GALICE cmz file.
The **name_DIGT** routines don't have any arguments.

SXDIGE Digitising and recording of raw data after each event, called from GUDIGI.
This routine calls the **name_DIGE** routines of the various detectors ONLY in case that detector has been selected (see later) for writing out the RAW data.
In the **name_DIGE** routines the detector specific RAW data arrays are filled and written onto the RAW data output stream.
The RAW data arrays have to be specified in a sequence **+KEEP,char_RAW**.
Here **char** stands for the detector specific identification character as specified above.
The sequence **+KEEP,char_RAW** however should actually contain the common **/SCXSCR/** which serves as a scratch space buffer for all detector raw data. Note that all RAW data consist of INTEGER values. However, the RAW format still has to be defined at the moment.
The **name_DIGE** routines don't have any arguments.

SXEVE Termination after each event, called from GUOUT.
This routine calls the **name_EVE** routines of the various detectors.
The **name_EVE** routines don't have any arguments.

SXEND Termination at the end of a simulation run, called from the main program GALICE.
This routine calls the **name_END** routines of the various detectors.
Each **name_END** routine has to produce a short printout indicating the routine name, so that from the log file of a certain run it can be directly seen which detectors were invoked and correctly terminated.
The **name_END** routines don't have any arguments.

The **ROUTINES** of the various detectors are called **name_uuuuuu**

name = Detector name (e.g. RICH, SCB, ...).

uuuuuu = Left free to the user except for INIT, MATE, TMED GEOM, SENS, DRAW, STEP, TRKI, TRKE, DIGT, DIGE, EVE and END which have always to be provided (see above).

An exception on this are the STEERING routines which are called SXuuuu

The **COMMONS** of the various detectors are called **/char_uuuuuu/**

char = Detector identification character (e.g. I for ITS).

uuuuuu = Left free to the user.

An exception on this are the STEERING commons which are called SCXuuuuu

Notes :

1. All **commons** have to be declared/used via **KEEP sequences**.

2. The KEEP sequence MUST have the same name as the (first) COMMON in that sequence and the use of BLOCKDATA must be omitted (all initialisation in name_INIT).

3. Executable statements or DATA statements are NOT allowed in a KEEP sequence.

2.3 Sequences for general use

The steering structure contains various sequences which are intended to be used for general purposes by the various detectors. The data in these sequences may NOT be changed, unless explicitly stated. In the following the data contents of these general purpose sequences is explained in detail.

```
+KEEP,SCXIO.
      COMMON /SCXIO/ LUNIN,LUNZEB,LUNSPC,LUNPAW,LUNDRW,
                     LUNRDB,LUNRAW
```

This common contains the standard I/O unit numbers as selected via the FFREAD [5] run cards (see later).

```
+KEEP,SCXDB.
      PARAMETER (NDBMAX=10)
      PARAMETER (ID_IP=1,ID_ITS=2,ID_MAG=3,ID_TPC=4,ID_TOF=5,
                 ID_SCB=6,ID_PHOS=7,ID_ZDC=8,ID_FMC=9,ID_RICH=10,
                 ID_STEER=NDBMAX+1)
      COMMON /SCXDB/ IDBUGF(NDBMAX+1),JPAWF(NDBMAX+1),
                     JVERF(NDBMAX+1),JOUTF(NDBMAX+1)
```

This sequence contains the detector identification numbers and various flags concerning selected detector settings.

IDBUGF(J) = Debug level (0,1,2) for detector "J"

JPAWF(J) = Paw level (0,1,2) for detector "J"

JVERF(J) = Version number for detector "J"

JOUTF(J) = RAW data output flag (0,1) for detector "J"

Note: "J" should be addressed as ID_name (e.g. JVERF(ID_TPC)).

```
+KEEP,SCXSCR.
      PARAMETER (NSCR=100000)
      COMMON /SCXSCR/ IARR(NSCR)
```

This is the scratch space for the detector SPC and RAW data arrays.
The data contents of this common may be overwritten by each detector.

+KEEP,SCXGEO.

COMMON /SCXGEO/ DALIC(3),RCOIL,ZCOIL,YBLEP

DALIC(3) = Array with the GEANT dimensions of the ALIC volume

RCOIL = Inner radius of the solenoid

ZCOIL = Half the length of the solenoid

YBLEP = Y position of the LEP beam w.r.t. the LHC beam

+KEEP,SCXEVT.

COMMON /SCXEVT/ JSXRUN,JSXEVT,JSXNPA,JSXZB,JSXZT,
RSXIMP,RSXPNU,RSXECM,NSXPIN(48),
IPX,IPY,IPZ

JSXRUN = Current run number

JSXEVT = Current event number

JSXNPA = Number of original input particles for current event

JSXZB = Atomic number (Z) of beam particles for current event

JSXZT = Atomic number (Z) of target particles for current event

RSXIMP = Impact parameter (in fm) for current event

RSXPNU = Momentum per nucleon in GeV for projectile particles

RSXECM = CMS energy for current event

NSXPIN(J) = Number of accepted input particles with GEANT code "J"

IPX = X coordinate (in micron) of the IP position

IPY = Y coordinate (in micron) of the IP position

IPZ = Z coordinate (in micron) of the IP position

2.4 Detector code structure

The whole simulation code of a certain detector has to be contained in 1 patch namely :

+PATCH,name. → Containing all the code of detector 'name'
+DECK,CDES. → Containing all the private KEEP sequences
+DECK,FORMAT. → Format descriptions of the SPC and RAW output
+DECK,name_uuuuu. → Containing the code of routine 'name_uuuuu'

Example :

```
+PATCH,SCB.
+DECK,CDES.
+KEEP,S_START.
      COMMON /S_START/ JCOUNT,MOD_FIRST
C
+KEEP,S_SPC.
      PARAMETER (MAX6=1000)
      COMMON /SCXSCR/ NHIT6,IROW6(MAX6),ICOL6(MAX6),IADC6(MAX6)
C
+DECK,SCB_INIT.
      SUBROUTINE SCB_INIT
C
+CDE,GCONST.
+CDE,SCXDB.
+CDE,S_START.
C
      (user code ...)
      RETURN
      END
+DECK,SCB_DIGT.
      SUBROUTINE SCB_DIGT
C
+CDE,GCTRAK.
+CDE,GCKINE.
+CDE,SCXDB.
+CDE,SCXEVT.
+CDE,S_SPC.
C
      (user code ...)
      RETURN
      END
```

These patches will all be added together to form the **GALICE** cmz file.
Notes :

1. No controls via selection flags are allowed.
   So, IF=name_.... should NOT be used.

2. On the GALICE cmz file there are dummy routines SUINIT, ..., SUEND.
   These entries may be used to test out new detector simulation codes which don't have an entry yet in the steering structure.

Note that these routines are always called, irrespective of the FFREAD cards selections (see hereafter).

2.5 Detector specific histograms

There exists the possibility for each detector to create private histograms and/or ntuples [4] to investigate the performance.
All this HBOOK/PAW activity MUST be put under the control of the corresponding **JPAWF()** flag (see above) as selected via the FFREAD [5] cards.

The only actions to be performed in the detector routines are :

1. Book the histograms/ntuples in the **name_INIT** routines into the **//GALICE** directory and with a unique identifier.
   e.g. **CALL HBOOKN(IDN,'...',...,'//GALICE',...,...)**

   where IDN stands for the HBOOK identifier with the convention :

   IDN=1000*J+N

   where : J = detector identification number
   (ex. ID_name like ID_PHOS etc...)
   $0 \leq N \leq 999$
   J = 0 for steerings

   Notes :

   - The histogram/ntuple IDN=999 may not be used, since this is reserved for the output SPC data CWN.
   - The histogram/ntuple IDN=888 may not be used, since this is reserved for the input event generator data CWN.

2. Fill the histograms/ntuples in the name_STEP, name_TRKE, name_EVE routines or whatever is the most convenient.

The correct PAW file opening, directory setting and writing out of the ntuples etc... is performed by the general steering routines.

3 Program installation and execution

3.1 Installation of the GALICE library

The complete GALICE library is created and installed by means of the automatic installation procedures provided by CMZ [3].
To create the GALICE library, it is sufficient to enter the command

cmz -install galice

By invoking this command, the CMZ machinery is activated and the installation macros provided in the **galice.cmz** file make sure that compilation and library creation is performed.
Note : Since various GEANT sequences are used within the GALICE code, the **geant.cmz** file has to be contained in the same directory as where the galice.cmz file is located.
Once the compilation and library creation steps have been completed, a file **libgalice.a** is created which contains the compiled subroutines of the complete GALICE package. This file **libgalice.a** may then be moved to a directory containing the libraries of the various other ALICE offline packages. At CERN all these libraries of the offline program pool can be found at **/afs/cern.ch/alice/offline/libs**.
Note : To prevent problems with different compiler versions, it is NOT recommended to copy these binary libraries to your home institute. Make sure to always create the libraries from scratch at your home institute by following the procedures outlined above.

3.2 Execution of the GALICE program

Once the GALICE library has been installed at some central location, various users can use that library to run their private GALICE detector simulation jobs.
An example job for a UNIX environment looks as follows :

```
#!/bin/sh
#
# GEANT SIMULATION FOR THE ALICE SETUP
#
# External input file
ln -s $HOME/shaker3.cwn evtgen.cwn
#
# Zebra output file
ln -s $HOME/test.zeb fort.20
#
# SPC data output file
ln -s $HOME/test.spc galice.spcdata
#
# Paw output file
ln -s $HOME/test.paw galice.pawdata
#
# Graphics output
ln -s $HOME/test.plot ALICEPLT
#
# Reference Data Base file
ln -s $HOME/test.rdb galice.rdb
```

```
#
ln -s $ALICE/pams/galice.cmz galice.cmz
cat << *EOR >cradle.kumac
exec $HOME/cmzlogon
set f -lan
file galice -r
seq steer
cd galice
cc galice
rel galice
exit
*EOR
#
cmz -b cradle.kumac
#
f77 *.o -L$HOME/libs -lgalice -lualice -L$CERN/libs -lgeant321 -lgenlib -lgraflib
        -lgrafX11 -lkernlib -lpacklib -lgraflib -L/usr/lib -lX11
a.out << *EOR
C ******* GEANT STEERING CARDS FOR ALICE SIMULATION *******
LIST
DEBU  0 0 1
DCAY  1
PAIR  1
COMP  1
PHOT  1
PFIS  0
DRAY  0
ANNI  1
BREM  1
MUNU  0
C — Select pure GEANH (HADR 1) or GEANH/NUCRIN (HADR 3)
HADR  1
LOSS  2
MULS  1
RAYL  0
C — Select automatic STMIN etc... calc. (AUTO 1) or manual (AUTO 0)
AUTO  1
C — Select optimisation level for GEANT geometry searches (0,1,2)
OPTI  2
CUTS  1.E-3 1.E-3 1.E-3 1.E-3 1.E-3 1.E-3 1.E-3 1.E-3 1.E-3 1.E-3 1.E10
C — PRIN card : PART VOLU MATE TMED VERT KINE SETS HITS DIGI SECS
PRIN  'PART' 'MATE' 'TMED' 'SETS'
```

```
C — Choices for SAVE card : INIT KINE JXYZ HITS DIGI
SAVE
C — Choices for user plots : XSEC LOSS
PLOT
TIME  0. 1. 1000000
C ************ STEERING CARDS FOR ALICE SIMULATION **************
C — Specify event type to be tracked through the ALICE setup
C — All positions are in cm, angles in degrees, and P and E in GeV
C ikine =  1 Single particle, specified vertex taken
C          2 Single particle, interaction point taken as vertex
C         -2 Single particle, interaction point taken as vertex
C          3 Event read-in from external file
C          ikine   x       y       z       theta   phi     pmom    part
C KINE     1       3.0     7.5     0.      10.     0.      10.     3.
C          ikine   thmin   thmax   phimin  phimax  pmin    pmax    part    npart
C KINE     2       0.0     180.    -180.   180.    -999.   999.    9.      5.
C          ikine   thmin   thmax   phimin  phimax  pmean   sigma   part    npart
C KINE     -2      0.0     180.    -180.   180.    10.     0.5     9.      3.
C          ikine   parmin  parmax  thmin   thmax   phimin  phimax  pmin    pmax
C KINE     3       -999.   999.    -999.   999.    -999.   999.    -999.   999.
KINE       2       85.     95.     265.    275.    1.      5.      6.      1.
C — Number of events to be processed —
TRIG  100
C — Select event number to start with
SXEVT  1
C — Detector selections and parameters for this simulation run
```

C	On/Off	Gate	Version	Debug	Track	Geom	Track	RAW/SPC	PAW
C			(-1=def.)	level	print	draw	draw	output	level
C	(1/0)	(ns)	(-1,0,..)	(0,1,2)	(0/1)	(0/1)	(0/1)	(0/1)	(0,1,2)
SXIP	1	200	-1	1	1	0	0	1	0
SXITS	1	200	-1	1	1	0	0	1	0
SXMAG	1	200	-1	0	0	0	0	1	0
SXTPC	1	200	-1	1	1	0	0	1	0
SXTOF	1	200	-1	1	1	0	0	1	0
SXSCB	1	200	-1	1	1	0	0	1	0
SXPHOS	1	200	-1	1	1	0	0	1	0
SXZDC	0	200	-1	0	0	0	0	1	0
SXFMC	0	200	-1	0	0	0	0	1	0
SXRICH	0	200	-1	0	0	0	0	1	0
SXSTEE	1	200	-1	0	1	0	1	1	0

```
C — Select charge of tracks to be drawn (-0+)
SXDCH  111
```

```
C — Hidden line removal (0=off 1=on) for drawings
SXHID  0
C —     INPUT ZEBRA SPC PAW DRAW RDB RAW unit numbers (0=none)
SXLUN  10        0        30    40     50       0      0
C — Workstationtype of graphics output (0=GKS -111=Postscript etc...)
SXWKS  -111
C — Select fake planes in case of PAW level 1 selected for IP
SXPLN  0
C — Select beam : 101=C 102=S 103=Al 104=Au 105=Ag 106=Cu 107=Pb
SXBEA  107
C — Sigma in (X,Y,Z) (cm) on IP position
SXIPX   0. 0. 0.
C — Specify maximum magnetic field in Tesla (neg. ⇒ default field)
SXFLD  -999.
C — Define initial contents of complete setup (0=air 1=vacuum)
SXVAC  0
C — Include the vacuum tube in the setup (1) or not (0)
SXTUB  1
C — Select (0=off 1=on) timing of subroutines for debug/optimisation
SXTIM  0
END
*EOR
```

The example job shown above consists basically of 3 parts :

- Some system commands to attach files to output streams, to invoke CMZ and to perform the loading step.
- A CMZ macro part which contains the CMZ commands to be executed.
- The FFREAD [5] cards to tailor the GEANT and GALICE running conditions.

In case one wants to test some new user code, it is convenient to replace the above CMZ macro part by the following :

```
#
cat << *EOR >run.car
+PATCH,*RUN.
+USE,P=MAIN,T=E.
+PATCH,MAIN.
+DECK,MAIN.
      PROGRAM GALICE
C
C ***     MAIN PROGRAM FOR ALICE SIMULATION ***
```

```
C
C —       Define dynamic storage sizes —
          PARAMETER (IQSIZE=6000000,IPSIZE=100000)
          COMMON /GCBANK/Q(IQSIZE)
          COMMON /PAWC/ BUF(IPSIZE)
C
C —       Initialise the dynamic storage —
          CALL GZEBRA(IQSIZE)
          CALL HLIMIT(-IPSIZE)
C
C —       Initialisation of the GALICE package —
          CALL SXINIT
C
C —       Simulation of all events —
          CALL GRUN
C
C —       Termination of the GALICE package —
          CALL SXEND
C
          STOP
          END
+DECK,xxxxx.
          (User code to be tested)
*EOR
#
ln -s $ALICE/pams/geant.cmz geant.cmz
ln -s $ALICE/pams/galice.cmz galice.cmz
cat << *EOR >cradle.kumac
exec $HOME/cmzlogon
set f -lan
crea run
ytoc run.car
file geant -r
seq gcdes
rel geant
file galice -r
seq steer
rel galice
pilot *run
cc -p
exit
*EOR
```

In this case a copy of the main program from the galice.cmz file has been included to allow user adjustment of the required program size.

3.3 Tailoring the GALICE running conditions

The user can tailor the GALICE running conditions to his/her own needs by means of the FFREAD [5] steering cards.
The GEANT specific steering cards are described in detail in the GEANT manual [1] whereas the functionality of the GALICE specific steering cards (named SX....) is mostly explained by the comment line(s) in front of them. However, a few main functions will be explained hereafter by means of some examples.
Note that it is important to enter the parameters in the correct format (i.e. REAL or INTEGER).

Event type specification
This is performed via the **KINE** data card, where the **ikine** parameter specifies the event type (see the comments in the example job) and the other parameters denote the various selections for the specified event type as follows :

x :	X coordinate of vertex (cm)
y :	Y coordinate of vertex (cm)
z :	Z coordinate of vertex (cm)
theta :	Polar angle of momentum in degrees
phi :	Azimuthal angle of momentum in degrees
pmom :	Particle momentum in GeV/c
part :	Particle type (GEANT id. convention)
thmin :	Minimal polar angle in degrees
thmax :	Maximal polar angle in degrees
phimin :	Minimal azimuthal angle in degrees
phimax :	Maximal azimuthal angle in degrees
pmin :	Minimal momentum in GeV/c
pmax :	Maximal momentum in GeV/c
npart :	Number of particles to be produced
pmean :	Mean of Gaussian momentum distribution
sigma :	Sigma of Gaussian momentum distribution
parmin :	Lowest particle id. to be accepted
parmax :	Highest particle id. to be accepted

Example :

KINE	2	85.	95.	265.	275.	1.	5.	6.	1.

This would simulate 1 μ^- through the ALICE setup, starting at the interaction point with $85 \leq \theta \leq 95$ and $265 \leq \phi \leq 275$ degrees and $1 \leq p \leq 5$ GeV/c.

Specification of the detector setup

This is performed via the **SXname** data cards, where **name** is one of the detector names as specified before. For a detector to be present in the ALICE setup, it should be selected **On** and a certain 'live time' has to be specified with the **Gate** parameter. Only signals between the start of the interaction and the end of the gate time will be recorded.

Note : In case for a certain track the maximum of all specified gate times has been exceeded, GEANT tracking will be stopped for that track in order to save cpu time.

Also several debug and output levels can be specified with these detector cards as well as the version of the detector geometry to be used.

Example :

C	On/Off	Gate	Version	Debug	Track	Geom	Track	RAW/SPC	PAW
C			(-1=def.)	level	print	draw	draw	output	level
C	(1/0)	(ns)	(-1,0,..)	(0,1,2)	(0/1)	(0/1)	(0/1)	(0/1)	(0,1,2)
SXIP	1	200	-1	0	0	0	0	0	0
SXITS	1	200	-1	1	0	0	0	0	2
SXMAG	1	200	-1	0	0	0	0	0	0
SXTPC	1	200	3	0	0	0	1	0	0
SXTOF	0	200	-1	0	0	0	0	0	0
SXSCB	1	200	-1	0	0	1	0	1	0
SXPHOS	1	200	-1	0	1	0	0	1	0
SXZDC	0	200	-1	0	0	0	0	0	0
SXFMC	0	200	-1	0	0	0	0	0	0
SXRICH	0	200	-1	0	0	0	0	0	0
SXSTEE	1	200	-1	0	0	0	1	1	0

In the above example, a GALICE run is performed with the IP region, ITS, MAG, TPC, SCB and PHOS present in the setup. All detectors have their default geometry except for the TPC where version 3 has been explicitly selected. For the ITS debug level 1 printout will be produced, whereas for the PHOS details of all tracking steps will be printed. The geometry specifications of the SCB will be provided and for the TPC and the Steerings (i.e. the space in between detectors) the particle tracks will be drawn. The SPC (i.e. space point) output for the SCB, PHOS and Steerings will be produced whereas for the ITS the private ntuples according to PAW level 2 will be created.

Notes :

1. Output will only be produced in case the corresponding output unit (see the SXLUN data card) has been activated.

2. PAW level 1 selection for IP will enable positioning of cylindrical planes (filled with air) at various pre-defined distances (via the SXPLN data card). Each activated plane will provide entries in an ntuple, recording various parameters of all

particles passing through that plane. This will allow studies of particle fluxes at various distances from the IP.
Example : SXPLN 1 3 7 8
This will enable particle flux studies through the layers 1, 3, 7 and 8.
The exact location of the various layers can be obtained from the routine S1GEOM.
In general, such a fake layer can be positioned in front of each detector element.

The various output streams
As already mentioned before, the various output streams can be (de)activated by means of the **SXLUN** data card. The streams to be specified are :

INPUT This is the input unit from which complete events (i.e. the event generator output [2]) can be read to be processed through the detector simulation.
This way detector performance under realistic conditions can be studied.

ZEBRA In case one wants to store the data of some GEANT structures (as specified on the **SAVE** data card), activation of this unit invokes output in FZ format [6] of the specified banks.

SPC Output for the detector specific space point data, as specified via the **SXname** data cards.
The format of the SPC data will be described hereafter.

PAW Output unit for the detector specific ntuples/histograms, as specified via the **SXname** data cards.

DRAW Output unit for the detector specific drawings, as specified via the **SXname** data cards. The desired output format (e.g. postscript, gks, ...) is specified on the **SXWKS** data card.

RDB In case one wants to store the geometry data as loaded in the GEANT structures, activation of this unit invokes output in RZ format [6] of all the geometry banks.
The produced file may be loaded directly into memory via an interactive GEANT session [1], thus allowing a complete (interactive) ALICE detector simulation without actually running the whole GALICE machinery. This way of running is very convenient to produce nice color plots of the detector setup.
Note : In running only GEANT interactively, the functionality of the SX.... data cards is not available.

RAW Output unit for the detector specific RAW data, as specified via the **SXname** data cards.
Currently the RAW data output is not implemented since the format still has to be defined.

4 The SPC data format

In this section a description will be presented of the structure of all the data written onto the SPC output stream.
Since the contents of the detector specific data blocks are completely defined by the various detector groups, the user is referred to the corresponding FORMAT decks in the GALICE.CMZ file for a detailed description. However, the output data format of the ALICE event generators [2] is identical for all of them and as such a detailed description will be presented here.

- The format adopted is the exchange format of column wise ntuples [4] (CWN). This means that the data files can be transported by binary FTP and reading can be performed interactively by using PAW [7] directly or in batch using the HBOOK [4] facilities.

- The data structure is based on track information, which means that each record (i.e. a row in the ntuple scheme) contains all the information belonging to a certain track.
  This will keep the number of columns needed for all our data well within the HBOOK limit of 50000 and will enable checking of the results of the pattern recognition / reconstruction directly against the original input data.
  The various data blocks for each track are :

  [GENDAT] [STEER] [TPC] [PHOS] [TOF] etc.....

  The contents of these data blocks will be explained hereafter.

- To indicate also the run and event structure, the first track record of an event contains some additional header data blocks, namely :

  [GENHDR] [EVHEAD]

  Also the contents of these data blocks will be explained hereafter.

- As described in the HBOOK and PAW manuals [4, 7], the so produced files are direct access files, which means that only the data requested will be read-in. In case of our large data samples this easily gives large gain factors in reading speed compared to conventional sequential files, whereas also a certain event can be addressed directly by the record number without 'skipping' the events in front.

- The CWN's contain the possibility of bit-wise packing of the data to be stored and since our raw data consists only of integer values, this packing can be performed automatically and very efficiently [4]. This results in a considerable reduction of the data volume.

4.1 Description of the data blocks

Some of the data blocks are meant for general purposes and as such they have a predefined format. The contents of these data blocks will be described in detail here.

[GENHDR] : Event header data from the event generator input.

Data stored : NIHEAD,IHEAD(NIHEAD),NRHEAD,RHEAD(NRHEAD)

In general, the contents will differ according to the event generator package used. However, the words NIHEAD, NRHEAD and IHEAD(1-6) will be universally defined for all event generators we will use for ALICE [2].
A complete header description for each generator can be found in [2] or in the patch SKELETON which is present in every generator cmz file, available in the ALICE offline program pool. Various generators are currently available and the contents of the SHAKER [2] event header is given below.

Layout of the SHAKER event header data

NIHEAD = Number of words stored in IHEAD()
NRHEAD = Number of words stored in RHEAD()

IHEAD(1) = Generator identifier (2=SHAKER)
IHEAD(2) = date
IHEAD(3) = time
IHEAD(4) = run number
IHEAD(5) = event number
IHEAD(6) = number produced particles
IHEAD(7) = dn/dy of charged particles
IHEAD(8) = not used
IHEAD(9) = not used
IHEAD(10) = not used
IHEAD(11) = not used
IHEAD(12) = not used

RHEAD(1) = lowerbound of rapidity window (YMIN)
RHEAD(2) = upperbound of rapidity window (YMAX)
RHEAD(3) = maximum Pt cutoff (GeV/c)
RHEAD(4) = η/π ratio
RHEAD(5) = p/π ratio
RHEAD(6) = $K^{\pm}/\pi^{\pm}$ ratio

[GENDAT] : Track data from the event generator output.

Data stored : IDPART,THETA,PHI,P,E

This format is absolutely identical for all the event generators of the ALICE event generator pool [2].

IDPART = Particle identification code (GEANT3.21/GALICE convention)
THETA = Polar angle in degrees [0,180]
PHI = Azimuthal angle in degrees [0,360]
P = Momentum in GeV/c
E = Total energy in GeV

Notes :

1. The generator identifier codes are described in [2].

2. In case just single particles were generated within GALICE, the SPC data blocks [GENHDR] and [GENDAT] are automatically created and for these cases IHEAD(1) is set to 0.

3. In running one of the ALICE event generators, the [GENHDR] and [GENDAT] data blocks will be created on output. However, because of the possibility in GALICE to perform a selection on the particles to be processed, the number of particles indicated in the header will be updated according to the applied selections. This means that in the GALICE SPC data output it may happen that only a subset of the original event generator data will be present.

[EVHEAD] : Event header data created by GALICE.

Data stored : NHEADX,IHEADX(NHEADX)

This format is absolutely identical for all simulated SPC data.

NHEADX = Number of words stored in IHEADX()

IHEADX(1) = Date of GALICE processing
IHEADX(2) = Time of GALICE processing
IHEADX(3) = Run number
IHEADX(4) = Event number
IHEADX(5) = Number of particles selected on input
IHEADX(6) = X position of the IP in micron
IHEADX(7) = Y position of the IP in micron
IHEADX(8) = Z position of the IP in micron

[STEER] : SPC data from GALICE steerings.

Data stored : JRUNX,JEVTX,JTKX

JRUNX = Run number for current track
JEVTX = Event number for current track
JTKX = Track number within the event

This steerings data block is very convenient to directly investigate in PAW which signals were induced in which detector(s) for a specific track.

These were the data blocks with a general character. The detector specific data blocks obviously have to be defined and described by the detector groups themselves. A detailed data description has always to be implemented in a deck FORMAT in the GALICE cmz file together with the corresponding detector simulation code.

4.2 Example of a data file

To illustrate the structure of an SPC data file containing several events, the following example shows a schematic layout of a data file with 3 events containing 6, 10 and 5 tracks respectively.

Track 1 ...	[GENHDR]	[GENDAT][EVHEAD]	[STEER][ITS][TPC][TOF]
Track 2 ...		[GENDAT]	[STEER][PHOS]
Track 3 ...		[GENDAT]	[STEER][ITS][TPC][TOF][PHOS]
Track 4 ...		[GENDAT]	[STEER][ITS]
Track 5 ...		[GENDAT]	[STEER][TPC][TOF][PHOS]
Track 6 ...		[GENDAT]	[STEER][ITS][TPC][TOF]
Track 1 ...	[GENHDR]	[GENDAT][EVHEAD]	[STEER][ITS][TPC][TOF]
Track 2 ...		[GENDAT]	[STEER][PHOS]
Track 3 ...		[GENDAT]	[STEER][ITS][TPC][TOF][PHOS]
Track 4 ...		[GENDAT]	[STEER][ITS]
Track 5 ...		[GENDAT]	[STEER][TPC][TOF][PHOS]
Track 6 ...		[GENDAT]	[STEER][ITS][TPC][TOF]
Track 7 ...		[GENDAT]	[STEER][ITS][TPC][TOF][PHOS]
Track 8 ...		[GENDAT]	[STEER][ITS][TPC][TOF][PHOS]
Track 9 ...		[GENDAT]	[STEER][ITS]
Track 10 ...		[GENDAT]	[STEER][ITS][TPC][TOF]
Track 1 ...	[GENHDR]	[GENDAT][EVHEAD]	[STEER][ITS][TPC][TOF]
Track 2 ...		[GENDAT]	[STEER][ZDC]
Track 3 ...		[GENDAT]	[STEER][ITS][TPC][TOF][PHOS]
Track 4 ...		[GENDAT]	[STEER][ITS]
Track 5 ...		[GENDAT]	[STEER][PHOS]

Acknowledgments

The author would like to thank everybody who has contributed to the development of the GALICE simulation package.
In particular he would like to thank Boris Batyunya, Rene Brun, Marek Kowalski, Karel Safarik and Alexandre Zinchenko for the many helpful discussions and valuable suggestions.

References

[1] R. Brun, *et al.*, GEANT3, CERN DD/EE/84-1, (1986).

[2] N. van Eijndhoven, *et al.*, Internal Note ALICE/GEN 95-32, (1995).

[3] CodeME S.A.R.L., CMZ a Source Code Management System, User's guide and Reference Manual, 1994.

[4] R. Brun, *et al.*, HBOOK User's Guide, CERN Program Library Y250, (1992).

[5] R. Brun, *et al.*, FFREAD User's Guide, CERN Program Library I302, (1987).

[6] R. Brun, *et al.*, ZEBRA User's Guide, CERN Program Library Q100, (1987).

[7] R. Brun, *et al.*, PAW Reference Manual, CERN Program Library Q121, (1993).