Standard Interfaces between Modules of Event Generators using dynamical Common Structures

F. Carminati, O. Di Rosa*, B. van Eijk*,**, I. Zacharov
CERN, Geneva, Switzerland

D. Hatzifotiadou
World Laboratory / HED Project, Lausanne, Switzerland

Abstract

Modularity in High Energy Physics Monte Carlo event simulation programs allows to combine physics models involved in different programs in one. To interface parts of the programs to each other it is necessary to standardise the data formats. Based on the identification of the information needed in the different phases of the event generation chain, we propose a set of simple commons allowing different program units to have access to the same data in a standard way.

---

* LAA Project, CERN, Geneva, Switzerland

** Also supported by the 'Netherlands Organization for Scientific Research (NWO)', The Netherlands

1. Introduction

At recent meetings of the ECFA working group on physics and detector simulation for future colliders, LHC in particular, the necessity has been stressed to introduce a standard recording of event history, particle properties data and particle decay tables. This to ease the exchange of information between different program units (modules) of different event generators in High Energy Physics (HEP). A step towards standardisation for Monte Carlo (M.C.) programs has been the proposal of the M.C. particle numbering scheme by the Particle Data Group (P.D.G.) [1, 2]. It constitutes a coding convention for particle identification essentially based on their "elementary particles" content. Further proposals have been presented notably by T. Sjöstrand et al. [3] and B. van Eijk [4]. T. Sjöstrand et al. have mainly been concerned with the standardisation of event generator output formats. In particular, they indicated a simple solution to store the main features of simulated events via a FORTRAN data structure - a Common - to allow future analysis programs to input in an unique way output data from various generators. This paper describes a more general approach to standardisation of HEP data. Our implementation of Event, Particles and Decay records is also based on simple FORTRAN commons. The choice of the variables therein is based on the quantities which the physicist as a user of a Monte Carlo event generator needs to know or store, and the relationship between them. It is based on the design of a modular M.C. system and aimed to meet the expectations of the M.C. user community in and outside CERN.

2. Contents of the proposal

In a run of an event generator a varying number of events may be produced, each providing many "entries" (i.e. a particle, a cluster, or any other system you may want to describe). The description of an event is the description of the entries thereby produced.

Let us first briefly recall which information the already established event commons [3] *HEPEVT* and *HEPSPN* foresee before describing the proposed commons. *HEPEVT* records for each event: its particle content, the event history, momenta, and the production vertices coordinates:

```
PARAMETER  (NMXHEP  =  2000)
COMMON  /  HEPEVT  /  NEVHEP,  NHEP,  ISTHEP(NMXHEP),
IDHEP(NMXHEP),  JMOHEP(2,NMXHEP),  JDAHEP(2,NMXHEP),
PHEP(5,NMXHEP),  VHEP(4,NMXHEP)
```

Here the different parameters, variables and arrays have the following meaning:

NMXHEP	maximum number of entries per event.
NEVHEP	the event number.
NHEP	number of entries (particles) in current event.
ISTHEP	status code for current entry.
IDHEP	particle identity (standard).
JMOHEP	pointer to mother.
JDAHEP	pointer to daughter.
PHEP	energy-momentum.
VHEP	production vertex position.

The second existing standard common *HEPSPN* contains the spin four-vector from which polarization information may be calculated. The detailed explanation of the variables of both commons is given in [3]. Some of the authors of major M.C. programs have already adopted these commons in their programs [5,6,7,8]. We notice that in the original proposal [3] only the event information was covered.

Our current proposal offers the possibility to handle particle properties, particle decays and naming in addition to the event information. The color connection information of the particles produced is now provided within the event common, and the spin has been included in the same commons rather than being in a separate one. The main improvement of the present proposal is that memory layout is continuous without unnecessary empty spaces. This has two advantages: a) memory access is optimized, since all information related to one particle is kept in adjacent memory locations; b) the length of the commons can be changed locally without the need of recompiling all the other modules referencing the common (dynamical -).

It is noted that this introduces a somewhat less transparent way to access the information. Therefore we propose and provide a set of routines to handle data structures within the commons. Alternatively, the data relative to one particle can be accessed via symbolic *offsets* (index). The index to a given particle can be found via a simple arithmetic calculation or via a statement function. We propose the following set of commons to be filled at different levels in the event simulation chain:

a) *HEPEVE*, HEP EVEnt, to record any phase of an event during the simulation chain.

b) *HEPPPR*, HEP Particle PRoperties, used for general Particle Properties.

c) *HEPPDE*, HEP Particle DEcay, used to describe the particle decays.

d) *HEPPNA*, HEP Particle NAme, used to store the particle character names.

The structure and the usage of the common is discussed in the following chapters. Appendix A shows a symbolic chart of a modular event generator illustrating the use of these commons.

3. The Event common

The layout of the new event common is shown in Fig. 1.

```
C  HEPEVT definitions...
   INTEGER IEVMAX, IEVSIZ, IEVDIM
   PARAMETER (IEVMAX = 2000, IEVSIZ = 23)
   PARAMETER (IEVDIM = IEVMAX*IEVSIZ)
   INTEGER IEVRUN, IEVNUM, IEVTOT, IEVSTA, IEVMCC, IEVMT1,
   IEVMT2, IEVDT1, IEVDT2, IEVCOM, IEVACM, IEVCOD, IEVACD, IHEP(IEVDIM)
   REAL EVMOMX, EVMOMY, EVMOMZ, EVENER, EVMASS, EVXVER, EVYVER, EVZVER,
   EVTVER, EVSPI1, EVSPI2, EVSPI3, EVSPI4, QHEP(IEVDIM)
   COMMON / HEPEVT / IEVNUM, IEVRUN, IEVTOT, IEVSTA, IEVMCC, IEVMT1,
   IEVMT2, IEVDT1, IEVDT2, IEVCOM, IEVACM, IEVCOD, IEVACD, EVMOMX,
   EVMOMY, EVMOMZ, EVENER, EVMASS, EVXVER, EVYVER, EVZVER, EVTVER,
   EVSPI1, EVSPI2, EVSPI3, EVSPI4, IHEP
   EQUIVALENCE (IHEP(1), QHEP(1))
   LHEP(NPHIS) = (NPHIS-1)*IEVSIZ
```

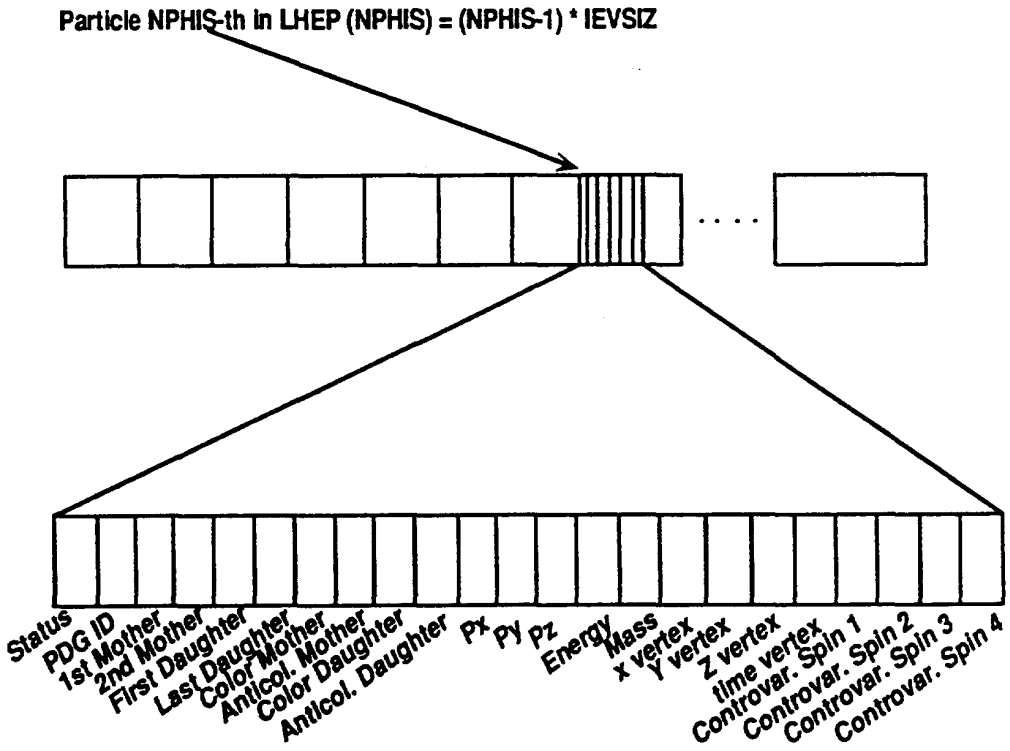


Fig. 1 The dynamical event common

Here the parameters are:

IEVMAX	maximum number of entries per event.
IEVSIZ	number of parameters characterizing one entry in the common.

IEVRUN is the user defined run number, IEVNUM is the current event number. IEVTOT is the total number of entries in the current event. The particles are counted from 1 to IEVTOT and introduced into the common in that sequence. The subsequent (23) variables, from IEVSTA to EVSPI4, hold the record of the last produced or accessed entry. Entries with their attributes are stored in the array IHEP contiguously (starting from position 27 in the common) in *particle records*. The particle record contains information on one particle as illustrated in Fig. 1. The array index J of one specified record is found from the count NPHIS of the particle ("n^{th} particle in the history") inside the event via a simple arithmetic operation or via the statement function LHEP: J = LHEP(NPHIS). The structure is the following:

IHEP(J+1)	status code for the particle;
IHEP(J+2)	identification P.D.G. code of the particle;
IHEP(J+3)	number of the first mother of the particle;
IHEP(J+4)	number of the second mother of the particle;
IHEP(J+5)	number of the first daughter of the particle;
IHEP(J+6)	number of the last daughter of the particle;
IHEP(J+7)	color mother of the particle;
IHEP(J+8)	anticolor mother of the particle;
IHEP(J+9)	color daughter of the particle;
IHEP(J+10)	anticolor daughter of the particle;
QHEP(J+11)	X-component of the 4-momentum of the particle;
QHEP(J+12)	Y-component of the 4-momentum of the particle;
QHEP(J+13)	Z-component of the 4-momentum of the particle;
QHEP(J+14)	energy of the particle;
QHEP(J+15)	mass of the particle;
QHEP(J+16)	X-coordinate of the production vertex;
QHEP(J+17)	Y-coordinate of the production vertex;
QHEP(J+18)	Z-coordinate of the production vertex;
QHEP(J+19)	time of the production vertex;
QHEP(J+20)	first spin component of the particle;
QHEP(J+21)	second spin component of the particle;
QHEP(J+22)	third spin component of the particle;
QHEP(J+23)	fourth spin component of the particle.

A more complete description of these variables is given in appendix B.

The following fragment of code shows how to calculate the combined mass of the decay products of particle number NPHIS and to check that it remains below the limit of the mass of the particle NPHIS:

```
   HEPEVE definitions
.
.
J=LHEP(NPHIS)
EVMASS  =  QHEP(J+15)
IEVDT1=IHEP(J+5)
IEVDT2=IHEP(J+6)
PXTOT=0.0
PYTOT=0.0
PZTOT=0.0
ENTOT=0.0
DO  10  JDAUG=IEVDT1,  IEVDT2
      J=LHEP(JDAUG)
      PXTOT=PXTOT+QHEP(J+11)
      PYTOT=PYTOT+QHEP(J+12)
      PZTOT=PZTOT+QHEP(J+13)
      ENTOT=ENTOT+QHEP(J+14)
10 CONTINUE
HMCOM=SQRT(ENTOT**2-PXTOT**2-PYTOT**2-PZTOT**2)
IF  (HMCOM.GT.EVMASS)  THEN
WRITE(6,*),"CHECK  FAILED"
STOP
ENDIF
```

Using the *HEPEVE* definition "box" as above we can handle up to 2000 entries / event. To extend the allowed number of entries we change the parameter **IEVMAX** only in the main program and there is no need to modify and recompile the other routines referencing the *HEPEVE* common. An "automated" way to fetch (or store) information about a particle **NPHIS** is provided with the subroutine **HEPHIS(NPHIS, CHOPT)**, which copies the particle record from (to) **IHEP** array to (and from) the first 23 words in front of it in the common. The "direction" of copy is specified by the **CHOPT** parameter:

CALL HEPHIS (NPHIS, CHOPT)

NPHIS	number of the particle in the history (input INTEGER)
CHOPT	character describing to the routine the action to take (input CHARACTER):

'G' GET, retrieve the parameters of particle number NPHIS;
'S' SET, set the parameters of particle number NPHIS. If NPHIS < IEVTOT and NPHIS≥ 0, a new entry is added and IEVTOT is increased by 1.

The following fragment of code shows how to calculate the combined mass of the products of particle number NPHIS calling HEPHIS (under the assumption that daughters of a particle are all recorded sequentially in the IHEP vector):

```
   HEPEVE definitions
.
.
CALL  HEPHIS  (NPHIS,  'G')
PXTOT=0.0
PYTOT=0.0
PZTOT=0.0
ENTOT=0.0
JDAUG1  =  IEVDT1
JDAUG2  =  IEVDT2
DO  10  JDAUG  =  IDAUG1,  JDAUG2
      CALL  HEPHIS  (JDAUG,  'G')
      PXTOT=PXTOT+EVMOMX
      PYTOT=PYTOT+EVMOMY
      PZTOT=PZTOT+EVMOMZ
      ENTOT=ENTOT+EVENER
10 CONTINUE
   HMCOM=SQRT(ENTOT**2-PXTOT**2-PYTOT**2-PZTOT**2)
   IF  (HMCOM.GT.EVMASS)  THEN
   WRITE(6,*),"CHECK  FAILED"
   STOP
   ENDIF
.
.
```

4. The Particle Properties Common

The Review of Particle Properties [2] which is published every two years in Physics Letters

by the Particle Data Group lists all the experimental results on the properties of particles. These properties include masses, widths or lifetimes, decay modes and so on. Where feasible, it provides a suggested "best value" for each parameter, based on their judgment using the best available data. It also provides an extensive summary of searches for hypothesized particles in the form of mass limits under specified assumptions. There is a collaboration of the Monte Carlo Simulation Laboratory group and the P.D.G. group aiming to provide a file containing a summary of the particle information needed by the M.C. programs. This file will be available at CERN and will be updated more frequently than the Review of Particle Properties itself, i.e. every year. In the meanwhile, the various event generators use values for the particle properties either coded in the program (e.g. in block data statements) or read from some file, provided by the M.C. author. When the value of one of these quantities is not experimentally well established, it is common practice by M.C. authors to adopt some theoretical estimate for it.

```
C  HEPPPR definitions...
   INTEGER IPPMAX, IPPSIZ, IPDIME
   PARAMETER (IPPMAX = 2000, IPPSIZ = 11)
   PARAMETER (IPDIME = IPPMAX*IPPSIZ)
   INTEGER NPARPS, IPPCOD, IPPPAP, IPPDEC, IPPFDP, IPPFDA,
   IPART(IPDIME)
   REAL PPCHAR, PPMASS, PPLMAS, PPUMAS, PPLIFE, PPFWHM,
   QPART(IPDIME)
   COMMON / HEPPPR / NPARPS, IPPCOD, PPCHAR, IPPPAP, PPMASS,
   PPLMAS, PPUMAS, PPLIFE, PPFWHM, IPPDEC, IPPFDP, IPPFDA,
   IPART
   EQUIVALENCE (IPART(1), QPART(1))
   LPART(IPAPSD) = (IPAPSD-1)*IPPSIZ
```

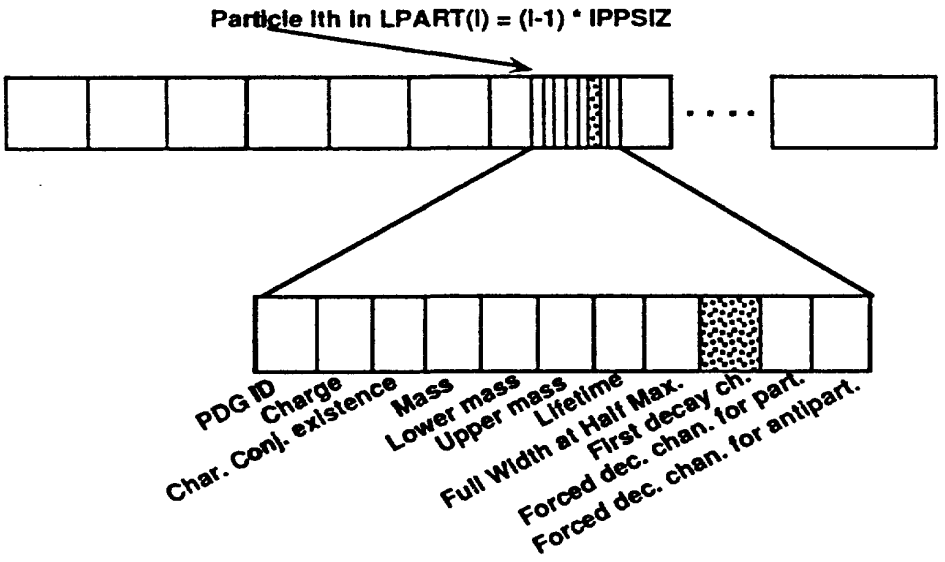


Fig. 2. The structure of the particle properties common

Different guess values for various particle properties are also used in M.C. simulation by different users. We propose to use standard commons for the particle properties, the decay channels and the particle names. At initialization, M.C. programs will read particle data (e.g. from a file) and will fill these commons (vectors IPART and IDEC) as proposed below, so that the data can be used by all program modules of the simulation chain.

The structure of the Particle Properties standard common is shown in Fig.2.

IPPMAX	maximum number of particles at initialization.
IPPSIZ	number of parameters characterizing one particle in the proposed common.

NPARPS is the total number of particles described in the common. The subsequent (11) variables, from IPPCOD to IPPFDA, hold the record of the last updated or accessed particle. Particle properties are stored in the array IPART contiguously (starting from position 13 in the common) in records of particle properties. The array index of one specified record J is found from the particle number IPAPSD inside the common by the statement function LPART. If J is the index to the record describing particle IPAPSD, then J = LPART(IPAPSD) and the structure of the record is the following:

IPART(J+1)	Monte Carlo Standard code, according to P.D.G.
QPART(J+2)	charge
IPART(J+3)	flag indicating the existence of charge conjugate; 1 if it exists
QPART(J+4)	mass
QPART(J+5)	lower mass limit
QPART(J+6)	upper mass limit
QPART(J+7)	life time
QPART(J+8)	full width at half maximum in the mass distribution
IPART(J+9)	pointer to the first decay channel (0 for stable particles)
IPART(J+10)	pointer to the forced decay channel for particle
IPART(J+11)	pointer to the forced decay channel for antiparticle

It is sufficient to give the standard identification code of a given particle to store or to fetch the particle data. The particle number IPAPSD, is derived from its standard identification code, IDPDG, by a translation routine (function LOCPNS) based on the original approach by G. Lynch (P.D.G., Berkeley). If we have a particle with code IDPDG and we want to find the mass and charge, the following sequence of code could be used:

```
HEPPPR definitions
```

```
.
.
JPART=LPART(LOCPNS(IDPDG))
.
.
```

Alternatively, there is a routine to fetch (and store) the particle record from (to) the first 11 words of the common:

```
CALL HEPPAR (IDPDG, CHOPT)
```

IDPDG — the standard particle code in the M.C. coding scheme (input INTEGER)

CHOPT — character describing to the routine the action to take (input CHARACTER):

- 'G' GET: retrieve the parameters for a particle given its P.D.G. code
- 'S' SET: set the parameters for a particle given its P.D.G. code

The following fragment of code shows how to increase by 10% the mass of a particle with P.D.G. code IDPDG:

```
HEPPPR definitions
```

```
.
.
CALL HEPPAR (IDPDG,'G')
PPMASS=PPMASS*1.10
CALL HEPPAR (IDPDG,'S')
```

5. The Decay table common

One of the "properties" of particles are their decays. The third common proposed is the one containing the decay table for particles. Normally particles have more than one modes of decay with different probabilities. The Particle Properties common described in the previous

```
C  HEPPDE Definitions...
   INTEGER IPDMAX
   PARAMETER (IPDMAX = 20000)
   INTEGER IPDCOD, IPDMXE, IPDMLT, IPDNEX, IPDPRO(10), IDEC(IPDMAX)
   REAL PDBRAT, QDEC(IPDMAX)
   COMMON / HEPPDE / IPDCOD, IPDMXE, PDBRAT, IPDMLT, IPDNEX, IPDPRO, IDEC
   EQUIVALENCE (IDEC(1), QDEC(1))
C  The value of the pointer to the next decay channel is:
C  IDEC(J) = J+5 + IDEC(J-1), if that is not the last channel
C  IDEC(J) = 0, if this is the last decay channel
C  Notice that the pointer to the first channel is known from the
C  common / HEPPPR /
```

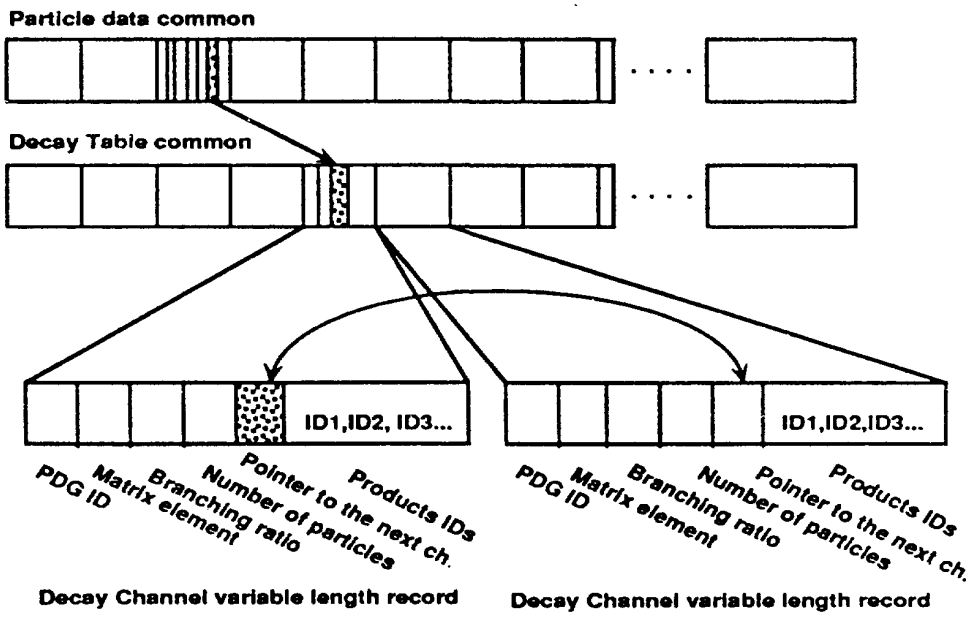


Fig. 3 The Decay Table Common and its structure linked to the particle properties common.

chapter contains a pointer to the "first" decay channel. This pointer is just an array index of the record of this decay channel in the Decay Table common. Therefore, the two commons are linked, as shown in Fig. 3. The variable **IPPDEC** in the **HEPPPR** common points to the variable **IPDNEX** in the *HEPPDE* common. The **IPDNEX** variable contains the index of the next decay channel, which in turn points to the next, and so on (see fig. 3). When the last decay channel of the particle is reached the pointer is set to 0.

The tables describing the decay of the particles are stored contiguously in the array **IDEC**. Analogously to the previous cases 5 variables and one array at the beginning of the common are filled with the information referring to the current channel. If **J** is the pointer to a decay channel, then the structure of the related information in the common is the following:

`IDEC(J-4)`	P.D.G. code of the parent particle
`IDEC(J-3)`	matrix element for the decay
`QDEC(J-2)`	branching ratio for the current channel
`IDEC(J-1)`	number of products (`IPDMLT`) in this decay channel
`IDEC(J)`	pointer to the next decay channel for the current particle; last decay channel if 0
`IDEC(J+1)`	P.D.G. code of the first product for the current decay channel
.	
.	
`IDEC(J+IPDMLT)`	P.D.G. code of the last product for the current decay channel

The following fragment of code shows how to choose a decay channel for a particle with P.D.G. code **IDPDG**:

```
   HEPPPR definitions
```

```
  HEPPDE definitions
```

```
.
.
CALL  HEPPAR  (IDPDG,'G')
RAN=RNDM(Q)
CBRAT=0.0
JDECCH=IPPDEC
10 CONTINUE
CBRAT=CBRAT+QDEC(JDECCH-2)
```

```
JDECCH=IDEC(JDECCH)
IF(CBRAT.LT.RAN.AND.JDECCH.NE.0)  GO  TO  10
.
.
```

A routine to fetch or to store the information in the particle decay table common is proposed:

`CALL HEPDEC (IDPDG, J, CHOPT)`, with

`IDPDG` P.D.G. code of the parent (input INTEGER)

`J` on input, either pointer to a decay channel or `-1`, on output either pointer to the next decay channel (zero if none) or to the current one, according to the value of `CHOPT` (input/output INTEGER)

`CHOPT` character describing to the routine the action to take (input CHARACTER):

- `' '` blank, retrieve the decay channel pointed to by `J`. If `J=-1` the first decay channel is retrieved. On output `J` points to the next decay channel, or is zero if the last one has been retrieved;
- `'G'` GET, retrieve the decay channel pointed to by `J`. If `J=-1` the first decay channel is retrieved. On output `J` points to the current decay channel;
- `'A'` ADD, add the given channel to the decay table for particle P.D.G.. On output `J` points to the new decay channel. If `J=-1` there is no more space in the decay table to store a new decay channel. The branching ratios are renormalized automatically;
- `'D'` DELETE, delete the decay channel pointed to by `J`. If `J=-1` the first decay channel is deleted. On output `J` points to the next decay channel;
- `'S'` SET, set the decay channel pointed to by `J`. If `J=-1` the first decay channel is set. On output `J` points to the current decay channel.

The following fragment of code shows how to choose a decay channel for a particle with P.D.G. code `IDPDG` using `HEPDEC`:

```
HEPPDE definitions
.
.
RAN=RNDM(Q)
CBRAT=0.0
JDECCH  =  -1
10 CALL  HEPDEC  (IDPDG,  JDECCH,  '  ')
CBRAT=CBRAT+PDBRAT
IF   (CBRAT.LT.RAN.AND.JDECCH.NE.0)  GOTO  10
.
.
```

6. The Particle Names Common

The use of the P.D.G. particle numbering scheme makes it possible to refer to a particle unambiguously. Nevertheless, it may be practical sometimes to name a particle. We therefore introduce one more common to record the particle names. Sixteen characters are foreseen for the full name and four for the abbreviated name, which is preferred in some applications. The proposed structure is included with the *HEPPPR* common:

```
CHARACTER*20  CPNAME
CHARACTER*16  FPNAME
CHARACTER*4   APNAME
COMMON  /  HEPPNA/  FPNAME,  APNAME,  CPNAME(IPPMAX)
```

In this scheme, **FPNAME** and **APNAME** are the full and the abbreviated name of the current particle. **CPNAME** is the array containing name and short name of all the particles. The index J of the current particle is still found from the IDPDG code via the usual function LOCPNS and the content of **CPNAME(J)** is the following:

```
CPNAME(J)  (1:16) full name
CPNAME(J)  (17:20)  abbreviation for that name.
```

In order to fetch and store particle names given the particle ID, we use **HEPPAR** as illustrated in chapter 6.

7. Summary and outlook

The aim of the paper is to define the minimum information needed to allow a User to construct his own simulation chain based on different modules from different Monte Carlo event generators. The current proposal is optimized from the point of view of memory space allocation and use and, more in general, as far as software engineering is concerned (within the framework of possibilities of FORTRAN77; tailoring to FORTRAN90 will be fully supported). It is also complete in its physics contents (as far as event, particle properties and decays are concerned). It gives flexibility, maintainability and it is still very easy to use. The set of dynamic commons and related handling routines are proposed as a standard to achieve modularity in

the various steps of Monte Carlo simulation. Working software has been developed and is available in C.E.R.N. libraries.

We have clearly profited from earlier work by T. Sjöstrand.

The facility for interfacing different program modules which has been described so far will be used in the COSMOS project [4, 9]. This project provides the "support environment" for the organization and the interface of different simulation packages and tracking programs, together with a user interface. A file containing the P.D.G. particle properties is available at C.E.R.N.. This file can be used to fill the *HEPPPR*, *HEPPDE* and *HEPPNA* commons.

References

[1] G.R. Lynch and T.G. Trippe, "Particle I.D. Numbers, Decay Tables, and Other Possible Contributions of the Particle Data Group to Monte Carlo Standards", LBL-24287, Proc. of the Workshop on Detector Simulation for the SSC, Argonne, Illinois (USA), August 1987.

[2] M. Aguilar Benitez et al., Particle Data Group, Phys. Lett. B 204 (1988) 1.

[3] G. Altarelli, R. Kleiss and C. Verzegnassi, "Z Physics at LEP 1", Vol. 3, CERN 89-08, 21 September 1989.

[4] B. van Eijk, Proposal for the C. & C. Huygens Research Fund, 'Netherlands Organization for Scientific Research (NWO)', The Netherlands (1988),
B. van Eijk et al., COSMOS, 'COmprehensive Super MOntecarlo System', in preparation.

[5] F. Anselmo and B. van Eijk, EUROJET version 3.1, in preparation,
A.Ali and B.van Eijk, EURODEC version 2.5, CERN Pool Program W5048, March 12, 1990.

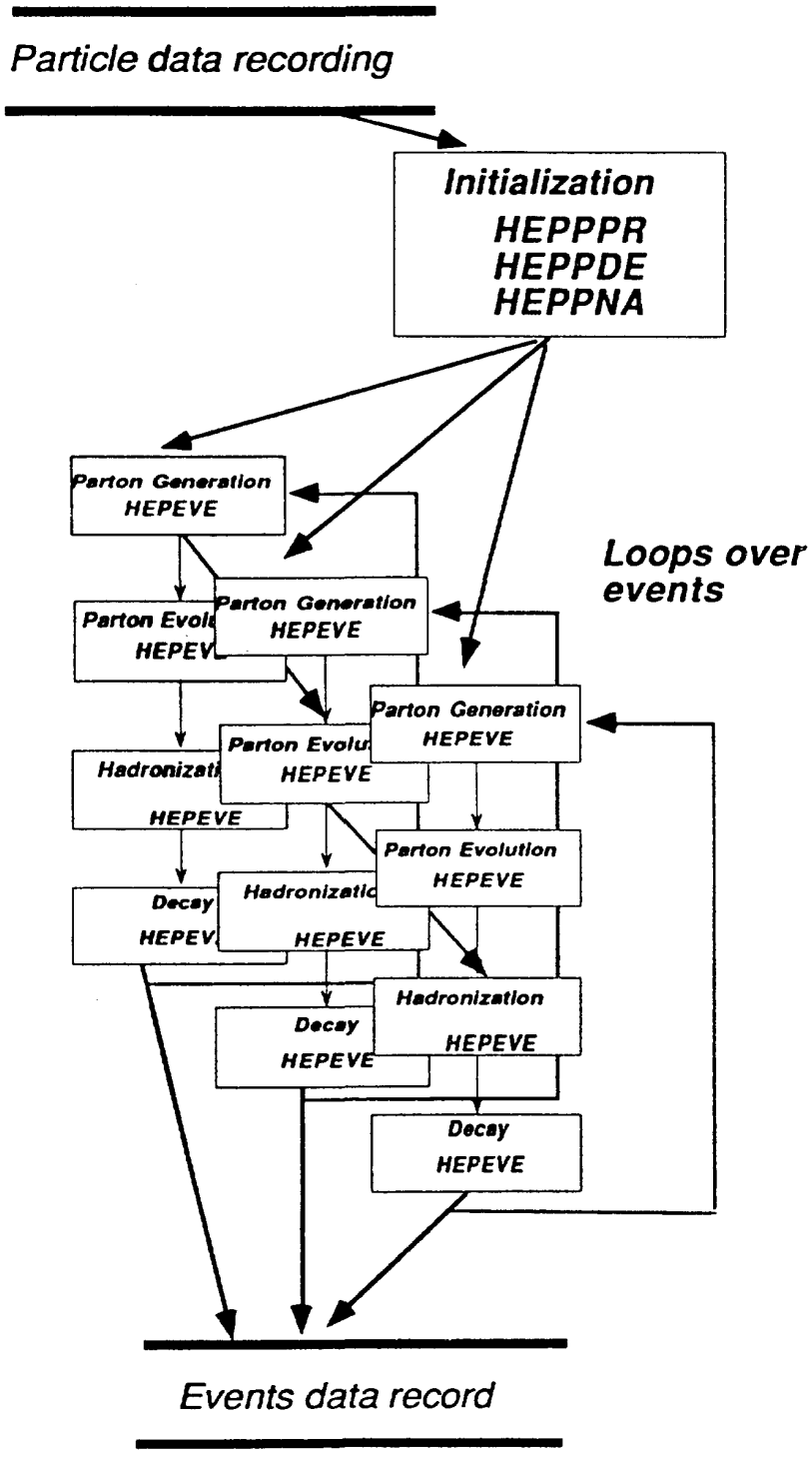
[6] The Lund Monte Carlo Programs, CERN Pool Program W5035/W5046/W5047/ W5048 long write-up, 3 April 1987, and references therein,
H.U. Bengtsson and T. Sjöstrand, Comput. Phys. Commun. **46** (1987) 43.

[7] G. Marchesini and B.R. Webber, HERWIG version 4.2, CERN Pool Program W5037, February 1990.

[8] R. Odorico, Comput. Phys. Comm. **59** (1990) 527.

[9] J.M. Alberty et al. , "An integration of Physics Packages, Data Modelling and Human Interface", to appear in the Proceedings of the International Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics, Lyon / Villeurbanne (France), March 1990.

Appendix A
Modularity in Event Generators with the Standard HEP common blocks



Appendix B

Detailed description of the variables used in the *HEPEVE* common.

IEVRUN	the run number as user-defined (input/output INTEGER)
IEVNUM	the event number as user-defined (input/output INTEGER)
IEVTOT	the actual number of entries produced and stored in current event. These are found in the first **NPART** positions of the history vector **IHEP** in the common. Index **NPHIS**, 1 ≤ **NPHIS** ≤ **NPART**, is used to denote a given entry (output INTEGER)
IEVSTA	status code with following meanings (input/output INTEGER): 0 null entry; 1 an existing entry, which has not decayed or fragmented. This is the main class of entries which represents the *final state* given by the generator; 2 an entry which has decayed or fragmented and therefore is not appearing in the final state, but is retained for event history information; 3 a documentation line, defined separately from the event history. This could include the two incoming reacting particles, etc; 4-10 undefined, but reserved for future standards; 11-20 at the disposal of each model builder for constructs specific to his program, but equivalent to a null line in the context of any other program. One example is the cone defining vector of HERWIG, another cluster or event axes of the JETSET analysis routines; 21- at the disposal of users, in particular for event tracking in the detector;
IEVMCC	particle identity, according to the Particle P.D.G. Group standard (input/output INTEGER)
IEVMT1	particle number of the first mother. The value is 0 for initial entries (input/output INTEGER)
IEVMT2	particle number of the second mother. The value is 0 for initial entries (input/output INTEGER). Normally only one mother exists, in which case the value 0 is used. In cluster fragmentation models, the two mothers would correspond to the q and qbar which join to form a cluster. In string fragmentation, the two mothers of a particle produced in the fragmentation would be the two end-points of the string (with the range in between implied); (input/output INTEGER)
IEVDT1	particle number of the first daughter. If an entry has not decayed, this is 0; (input/output INTEGER)
IEVDT2	particle number of the last daughter. If an entry has not decayed, this is 0. This pointers are specially useful under the assumption that the daughters of a particle (or cluster or string) are stored sequentially, so that the whole range **IEVDT1** to **IEVDT2** contains daughters. Even in cases where only one daughter is defined (e.g. K°->K°S) both values should be defined, to make for a uniform approach in terms of loop constructions (input/output INTEGER)
IEVCOM	color mother of the particle
IEVACM	anticolor mother of the particle
IEVCOD	color daughter of the particle
IEVACD	anticolor daughter of the particle
EVMOMX	momentum in the X direction, in GeV/c (input/output REAL)

EVMOMY	momentum in the Y direction, in GeV/c (input/output REAL)
EVMOMZ	momentum in the Z direction, in GeV/c (input/output REAL)
EVENER	energy, in GeV (input/output REAL)
EVMASS	mass, in GeV/c**2. For space-like partons, it is allowed to use a negative mass, according to **AMASS=-SQRT(-m^2)** (input/output REAL);
EVXVER	production vertex x position, in mm (input/output REAL)
EVYVER	production vertex y position, in mm (input/output REAL)
EVZVER	production vertex z position, in mm (input/output REAL)
EVTVER	production time, in mm/c (= $3.33*10^{-12}$ s) (input/output REAL)
EVSPI1	spin first component of the particle
EVSPI2	spin second component of the particle
EVSPI3	spin third component of the particle
EVSPI4	spin fourth component of the particle.