EPOS 1 User's Guide

Klaus WERNER [1], **Tanguy PIEROG** [2], **Sarah PORTEBOEF** [1]

[1] *SUBATECH, Université de Nantes – IN2P3/CNRS– EMN, Nantes, France*
werner@subatech.in2p3.fr

[2] *Forschungszentrum Karlsruhe, Karlsruhe, Germany*
pierog@ik.fzk.de

Contents

Chapter 1

Introductory remarks

1.1 References

- **CORE-CORONA SEPARATION IN ULTRA-RELATIVISTIC HEAVY ION COLLISIONS,** By Klaus WERNER, Phys. Rev. Lett. 98, 152301 (2007), arXiv: 0704.1270.

- **PARTON LADDER SPLITTING AND THE RAPIDITY DEPENDENCE OF TRANSVERSE MOMENTUM SPECTRA IN DEUTERON-GOLD COLLISIONS AT RHIC**. By Klaus Werner, Fu-Ming Liu, Tanguy Pierog, Phys. Rev. C 74, 044902 (2006), arXiv:hep-ph/0506232.

- **PARTON-BASED GRIBOV-REGGE THEORY**, H. J. Drescher, M. Hladik, S. Ostapchenko, T.Pierog, and K. Werner, Phys. Rept. 350 (2001) 93-289 .

1.2 Copyright Notice

Copyright reserved in all countries of the world. The program may not be reproduced by any method without prior consent of the authors.

1.3 Installation and File Structure

The EPOS package consists of several source code programs (epos...f), include files (epos.inc...), pre-calculated tables (epos.ini...), the epos shell script epos (or epos- for special applications), and the Makefile. We refer to all this as source code files.

Input is read from some file referred to as optns-file. Many optns-files may be created referring to different reactions, optns, etc. The corresponding directory is called optns-directory.

There are two ways to obtain output from a simulation. Some online analysis options (specified in the optns-file) allow to obtain standard histograms. The corresponding histograms will be written into files called histo-files. This is more meant to be used by the developers, since users usually have their own analysis tools. In this case a so-called data-file should be created, storing particle properties, which allow later analysis of the simulation data. The choice of the output has to be defined in the optns-file.

Finally a check-file is created containing warning messages or run information if requested (in the optns-file).

All these files may reside in the same directory, but in particular when many optns and data-files are created, it may be more useful to put the different types of files into different directories. This statement also holds for the object and executable files.

So the very first task is the choice of the directories, where the different types of files will be found. Since this is essentially a question of taste, the user is completely free to make his choice. For this purpose on needs to put the configuration file “.epos” into the home directory, and edit the file (self-explaining). Here we define environment variables, which are then used in the makefile and the epos script. After editing the file, one has to type “source .epos” to make the definitions effective. Make sure that these variables are always defined when you compile and run the program.

Then one has to go to the epos-directory, as defined in .epos, unpack the epos-package (containing the source code files), if not yet done so. The program is compiled and linked via “make”.

1.4 Running EPOS

Make sure that the environment variables defining directories are active, for example with a “source .epos” in the shell login file.

Per default, the random number generator starts always with the same seed. If you prefer random seed, you may edit the epos script, and uncomment two lines (well explained in the file).

Go to the optns-directory, create an optns file, with the name <name>.optns, where <optns> is some arbitrary name, and run the program via “epos <name>” .

As an example, create a file called “test.optns”, with the following content (deuteron-gold simulation):

```
application hadron
set nevent 1000     !number of events
set modsho 200     !printout after modsho
events
set ecms 200          !energy per nucleon-nucleon
in cms
set laproj 1          !proj Z
set maproj 2          !proj A
set latarg 79          !targ Z
set matarg 197      !targ A
switch splitting on     !nuclear effects due to
                        !parton splitting and
fusion,
                        !leads to screening
(saturation)
                        !and more
nodecay 20       !suppress Kshort decay
frame nucleon-nucleon

!define output format:
output epos
record event nevt nptl b endrecord
record particle id p1 p2 p3 p4 endrecord
```

Text following the exclamation mark “!” is considered to be a comment.

Chapter 2

EPOS Input Commands

Specification of the reaction, options, parameter modifications, output definition, ..., all this will be defined via a number of input commands in the optns-file.

Actually the optns-file is read from the epos script, there are some "predefined" commands already executed before reading the optns-file, but these are definitions the user does not really want to worry about, like the names of certain input tables and so on.

Attention: all commands are case sensitive, so the command `Set` is not the same as `set`.

The commands are executed in the order of their appearance, in the case of simple definitions as in the example in the preceding chapter, the order is irrelevant.

In the following, we discuss some elementary commands.

2.1 The Command "application"

For all hadronic interactions, like hadron-hadron (proton-proton, pion-proton, ...), hadron-nucleus, nucleus-nucleus, one needs

```
application hadron
```

Many other choices exist, which allow the simulation of electron-positron annihilation, or deep inelastic scattering, or microcanonical cluster decay, ..., but we do not discuss these options here.

2.2 The Command "set"

The command set allows to define variables, via

```
set <variable> <value>
```

where `<variable>` is the name of the variable which should be defined, and `<value>` the corresponding value. Depending on the variable, the value may be a real or an integer number, however, the type conversion is automatic, the user does not have to worry about this. If for example an integer is expected, a value 5.0 will be interpreted as 5, even 5.1 or 4.9 are taken as 5.

In the following we give a list of the most essential variables, which allow to specify the reaction:

`nevent` .... number of events
`ecms` or `elab` ... energy (use one of the two !)
`laproj` .... number of projectile protons)
`maproj` .... number of projectile nucleons
`latrag` .... number of target protons
`matarg` .... number of target nucleons

An arbitrary projectile hadron (say with id <id>, see list below) may be considered via

```
set laproj -1
set maproj 1
set idproj <id>
```

Per default, the impact parameter, (b,phi) in polar coordinates, is chosen randomly. There is an option to choose b and phi randomly within a user defined range, defined via the variables

`bminim` .... minimum value of b
`bmaxim` .... maximum value of b
`phimin` .... minimum value of phi
`phimax` .... maximum value of phi

The initial seed for the random number generator can be changed via

`seedj` .... initial seed (some big number)

which sets the seed before the first event.

2.3 The Command "switch"

The command switch allows to turn on or off certain features, via

```
switch <feature> on
```

or

```
switch <feature> off
```

where `<feature>` is one of the following (default is "on"):

`splitting` ... consideration of parton splitting and fusion
`decay` .... consider resonance decay

2.4 The Command "nodecay"

Per default (unless decay is switched off) all particles having an entry in the decay table will decay. To prevent the decays of certain particles, the command nodecay may be used. The syntax is:

```
nodecay <id>
```

where `<id>` is the particle id code (see list below). The nodecay command may be used several times, to suppress several decays. So, for example, to suppress the decay of K_s and Λ, one needs

```
nodecay 20
nodecay 2130
```

2.5 The Command "input"

To include commands from another file, the command input may be used. Syntax:

```
input <filename>
```

with `<filename>` being the name of the input file. The command input may appear several times to read in input from different files.

2.6 The Command "frame"

In case of nucleon-nucleon, nucleon-nucleus, or nucleus-nucleus, the natural reference frame is the nucleon–nucleon center-of-mass frame. All internal calculations are performed in the natural frame, and therefore any output into the check-file during the run refers to this frame. The natural frame is also referred to as working frame.

There is the possibility to present the final results (written into the data file) in a different frame. This has to be specified by the command frame, using the following syntax

```
frame <frame>
```

where `<frame>` specifies the frame. The following frames are recognized, with obvious meanings

```
nucleon-nucleon
target
```

Setting the variable `elab` implies automatically the target frame, whereas setting `ecms` implies the definition of the nucleon-nucleon cms frame. If in addition the frame is defined via the frame command, the latter one counts, even if the frame command appears earlier than the set command.

2.7 The Command "fname"

There are several input and output files, like the optns-file, the data-file, input tables, and so on. File names are specified via

```
fname <ftype> <name>
```

with `<ftype>` being the file type (`histo` and `data` being the most important) and `<name>` being the filename. All these file name definitions are done in the epos script.

There are in particular the file name definition of pre-calculated tables. The program checks, whether these files exist (and are calculated with the correct parameters). Usually this is the case, and the user has not to worry about this. The tables are taken from these files, and not calculated again (which is very time consuming). If the program complains about some missing table, and starts a long table calculation, maybe simply a path is wrongly defined (check .epos).

2.8 The Command "output"

The "output" command affects the way output is produced during a EPOS run. The syntax is

```
output <output>
```

where `<output>` may be any of the following keywords:

`epos`	Standard EPOS output
`osc1997a` ...	OSCAR 1 convention
`osc1999a` ...	OSCAR 2 convention
`ustore`	User defined output (modify subroutine ustore)

In case of the EPOS option, one has to specify which information one wants to print into the data file.

In general, the structure of the data file is the following: for each event, first an "event record" gives information concerning the event, then follow a number of "particle records", giving information about the particles, one record per particle.

The content of the records is defined via the "record" command, to be discussed in the following.

2.9 The Command “record”

Using the EPOS output option (see preceding section), one need to define the event and particle records, which is done via

```
record event evar1 evar2 ...  endrecord
```

for the event record structure, and

```
record particle pvar1 pvar2 ...  endrecord
```

for the particle record structure. `evar1 evar2 ...` is a list of an arbitrary number of event variables (characterizing an event) and `pvar1 pvar2 ...` is a list of an arbitrary number of particle variables (characterizing a particle).

A complete list of event and particle variables can be found in the tables of the following chapter.

2.10 The Command “idchoice”

The particle id is a common choice among the elements of the particle record. In case of oscar output, the “pdf” codes are used, otherwise the standard epos id code (= isajet code = venus code = nexus code). This can be changed by using

```
idchoice pdf
```

which provides the pdf code even for epos output.

2.11 The Command “print”

During the execution of EPOS, it is possible to print out information (into the check-file). Each print statement has a level assigned, an integer between 0 and 9. The higher the level, the more detailed is the printed information. One may control the printout by using the print command, via

```
print <subroutine> <level>
```

where `<subroutine>` is a name of a EPOS subroutine and `<level>` an integer between 0 and 9. This command causes printout of levels 0 to `<level>` to be active in subroutine `<subroutine>`. We adopt the following conventions:

level 0 ... no output (default)
level 1 ... warnings
level 2 ... minimum output to reconstruct event
level 3 ... minimum output to reconstruct program flow
level 4 ... more detailed output (for the user)
level 5 ... very detailed output (for the user)
level 6 ... infos for the developers
level 7 ... more detailed infos for the developers
level 8 ... output in utility routines
level 9 ... more detailed output in utility routines

The choice `*` for `<subroutine>` affects all subroutines, this means

```
print * <level>
```

activates printout up to level `<level>` in all routines.

The print command is not defined for all subroutines. In case this possibility is not foreseen for a given subroutine, the print command is simply ignored.

Output usually written into the check-file may be redirected to the screen via the input command

```
printcheck screen
```

The print command may be used as a debugging tool. In case of errors, one may locate the problem, and provide sufficiently output, to allow the program developers to make a diagnose without having to rerun the code.

Chapter 3

Tables

3.1 Event Variables

variable	meaning	*Fortran*
`nevt`	event number	*nrevt*
`nptl`	number of particles per event	*nptevt*
`b`	absolute value of impact parameter	*bimevt*
`phi`	angle of impact parameter	*phievt*
`ncol`	number of collisions	*kolevt*
`pmx`	reference momentum	*pmxevt*
`egy`	pp cm energy	*egyevt*
`npj`	number of primary proj participants	*npjevt*
`ntg`	number of primary targ participants	*ntgevt*

3.2 Particle Variables

variable	meaning	*Fortran*
`i`	particle number (consecutive numbering)	*i*
`id`	particle id	*idptl(i)*
`p1`	x-component of particle momentum	*pptl(1,i)*
`p2`	y-component of particle momentum	*pptl(2,i)*
`p3`	z-component of particle momentum	*pptl(3,i)*
`p4`	particle energy	*pptl(4,i)*
`p5`	particle mass	*pptl(5,i)*
`o1`	particle number of first origin	*iorptl(i)*
`o2`	particle number of last origin	*jorptl(i)*
`c1`	particle number of first child	*ifrptl(1,i)*
`c2`	particle number of last child	*ifrptl(2,i)*
`st`	particle status	*istptl(i)*
`ty`	type of ptl source	*ityptl(i)*
`x1`	x-component of formation point	*xorptl(1,i)*
`x2`	y-component of formation point	*xorptl(2,i)*
`x3`	z-component of formation point	*xorptl(3,i)*
`x4`	formation time	*xorptl(4,i)*
`ido1`	id of father	*idiptl(i)*
`ido2`	id of mother	*idjptl(i)*

Some of the variables in the above lists require some more explication, which will be provided in the following.

The Particle Status

The particle status is a two digit number, which contains information about the particle type (first digit = particle type) and about the fact whether or not it is a last generation particle (second digit = generation flag). We consider 5 particle types, listed in the following table:

Particles	particle type
Hadrons, leptons, photon	0
Droplets	1
Partons	2
Pomerons	3
Remnants	4

During the generation procedure, whenever a particle is created, it gets the generation flag 0 (= last generation), whenever a particles interacts or decays, it gets the generation flag 1 (= not last generation). This applies to all particle types. So the variable `st` may assume the following values:

	last generation	not last generation
Hadrons, leptons, photons	00	01
Droplets	10	11
Partons	20	21
Pomerons	30	31
Remnants	40	41

Special case strings :

	`st`
strings	29

There is another special case concerning the generation flag: the value 2 means that this particle is obsolete (being finally discarded, for example in case of Pomerons having not enough mass, etc).

The command

```
set istmax <istmax>
```

with <istmax> being some integer number, allows to restrict the output particles to those with status less or equal than `<istmax>`. For `<istmax>` = 0 , for example, only the last generation hadrons, leptons, photons are written, for `<istmax>` = 1, in additions all parents.

Type of particle sources

Particles may originate from strings, remnants, etc., referred to as particle source. The variable `ty` specifies the type of the particle source, for each particle, the following valuse are possible

type	`ty` range
Soft Pomeron	20-29
Semihard Pomeron	30-39
Projectile remnants	40-49
Target remnants	50-59

Children, origin

Considering resonance, droplet, or string decay, the decay products are considered to be the children with respect to the decaying object, the latter one is the origin with respect to the decay products. The variable `o1` refers here to the particle number of the origin (decaying object), `c1` and `c2` refer to the particle numbers of the first and the last child (decay product). In other words, since the children are stored consecutively, their numbers go from c1 to c2.

The notion of children and origin is also useful for Pomerons, remnants, and partons. A Pomeron is always associated to a projectile and a target nucleon, these two being considered as origins (o1 and o2) of the Pomeron. A remnant is associated to either a projectile or a target nucleon, which is considered to be its origin (o1).

On the other hand, a Pomerons or a remnant is realized as a chain of partons, these partons being coinsidered to be the children (c1 to c2) of the Pomeron or remnant, the latter one is the origin (o1) of the partons.

Several partons constitute a string, so a string is considered child (c1) of the corresponding partons, the partons are the origins (o1 to o2) with respect to the string.

Special case: the participating projectile and target nucleons have origin o1 equal to -1.

For <istmax> = 0, only the last generation particles are stored. In this case, the particle numbers of the origins provide useless information, because the particles with these numbers are not stored. One may nevertheless get information about the origins by considering the variables ido1 (and ido2), the particle ids of the first and last origin.

For better understanding the use of children - origin see the following exemple.

Exemple

We show a realistic example for a particle list, representing an event of a pp scattering at 200 GeV in the cms. Each line represents a particle (in a general sense, including Pomerons, partons, ...). We only print variables related to status - children - origin. The third column represents the particle numbering i, the first two columns the origins o1 o2 of particle i, columns 4 and 5 the children c1 c2 of particle i, column 6 finally represents the status of particle i.

o1	o2	i	c1	c2	st
-1	0	1	0	0	1
-1	0	2	0	0	1
1	2	3	0	0	32
1	2	4	5	14	31
4	0	5	21	0	21
4	0	6	21	0	21
4	0	7	26	0	21
4	0	8	26	0	21
4	0	9	26	0	21
4	0	10	26	0	21
4	0	11	26	0	21
4	0	12	26	0	21
4	0	13	26	0	21
4	0	14	26	0	21
1	0	15	17	18	41
2	0	16	19	20	41
15	0	17	37	0	21
15	0	18	37	0	21
16	0	19	45	0	21
16	0	20	45	0	21
5	6	21	22	25	29
21	0	22	0	0	0
21	0	23	0	0	0
21	0	24	0	0	0
21	0	25	0	0	0
7	14	26	27	36	29
26	0	27	51	53	1
26	0	28	54	55	1
26	0	29	0	0	0
26	0	30	0	0	0
26	0	31	0	0	0
26	0	32	0	0	0

```
26   0  33  56 58   1
26   0  34  59 61   1
26   0  35   0  0   0
26   0  36  62 64   1
17  18  37  38 44  29
37   0  38   0  0   0
37   0  39  65 66   1
37   0  40   0  0   0
37   0  41  67 69   1
37   0  42  70 72   1
37   0  43   0  0   0
37   0  44   0  0   0
19  20  45  46 50  29
45   0  46   0  0   0
45   0  47  73 75   1
45   0  48   0  0   0
45   0  49  76 77   1
45   0  50   0  0   0
27   0  51  78 79   1
27   0  52   0  0   0
27   0  53   0  0   0
28   0  54   0  0   0
28   0  55   0  0   0
33   0  56   0  0   0
33   0  57   0  0   0
33   0  58   0  0   0
34   0  59   0  0   0
34   0  60   0  0   0
34   0  61   0  0   0
36   0  62   0  0   0
36   0  63   0  0   0
36   0  64   0  0   0
39   0  65   0  0   0
39   0  66   0  0   0
41   0  67   0  0   0
41   0  68   0  0   0
41   0  69   0  0   0
42   0  70   0  0   0
42   0  71   0  0   0
42   0  72   0  0   0
47   0  73   0  0   0
47   0  74   0  0   0
47   0  75   0  0   0
49   0  76  80 81   1
49   0  77   0  0   0
51   0  78   0  0   0
51   0  79   0  0   0
76   0  80   0  0   0
76   0  81   0  0   0
```

3.3 ID Codes of Quarks, Leptons, Photon

We are using the ISAJET/VENUS particle code. Quarks and leptons are numbered in order of mass:

u quark	1	electron neutrino	11
d quark	2	electron	12
s quark	3	muon neutrino	13
c quark	4	muon	14
b quark	5	tau neutrino	15
t quark	6	tau	16

with a negative sign for antiparticles. The code for a diquark is +-IJ00, where J≤K refers to the quarks Arbitrary conventions are:

gluon	9
photon	10
KS	20
KL	-20
W+	80
Z0	90

3.4 ID Codes of Basic Hadrons

The code for a meson is a compound integer +-JKL, where J$\leq$K are the quarks and L is the spin. The sign is for the J quark. Flavor singlet mesons are ordered by mass, PI0 = 110 ETA = 220 ETAP = 330 ETAC = 440 which is natural for the heavy quarks. Similarly, the code for a baryon is a compound integer +-IJKL formed from the three quarks I,J,K and a spin label L=0,1.
The complete list of basic id codes is provided in the following. The antiparticles of the baryons are not explicitly shown.

Mesons:

id	label	mass	charge
110	PI0	.13496E+00	0.00
120	PI+	.13957E+00	1.00
-120	PI-	.13957E+00	-1.00
220	ETA	.54745E+00	0.00
130	K+	.49367E+00	1.00
-130	K-	.49367E+00	-1.00
230	K0	.49767E+00	0.00
-230	AK0	.49767E+00	0.00
330	ETAP	.95760E+00	0.00
140	AD0	.18645E+01	0.00
-140	D0	.18645E+01	0.00
240	D-	.18693E+01	-1.00
-240	D+	.18693E+01	1.00
340	F-	.19688E+01	-1.00
-340	F+	.19688E+01	1.00
440	ETAC	.29788E+01	0.00
150	B+	0.5279E+01	1.00
-150	B-	0.5279E+01	-1.00
250	B0	0.5279E+01	0.00
-250	AB0	0.5279E+01	0.00
350	B0S	0.5369E+01	0.00
-350	AB0S	0.5369E+01	0.00
111	RHO0	.76810E+00	0.00
121	RHO+	.76810E+00	1.00
-121	RHO-	.76810E+00	-1.00
221	OMEG	.78195E+00	0.00
131	K*+	.89159E+00	1.00
-131	K*-	.89159E+00	-1.00
231	K*0	.89610E+00	0.00
-231	AK*0	.89610E+00	0.00
331	PHI	.10194E+01	0.00
141	AD*0	.20071E+01	0.00
-141	D*0	.20071E+01	0.00
241	D*-	.20101E+01	-1.00
-241	D*+	.20101E+01	1.00
341	F*-	.21103E+01	-1.00
-341	F*+	.21103E+01	1.00
441	JPSI	.30969E+01	0.00
151	B+*	0.5325E+01	1.00
-151	B-*	0.5325E+01	-1.00
251	B0*	0.5325E+01	0.00
-251	AB0*	0.5325E+01	0.00

Baryons:

id	label	mass	charge

1120	P	.93828E+00	1.00
1220	N	.93957E+00	0.00
1130	S+	.11894E+01	1.00
1230	S0	.11925E+01	0.00
2130	L	.11156E+01	0.00
2230	S-	.11974E+01	-1.00
1330	XI0	.13149E+01	0.00
2330	XI-	.13213E+01	-1.00
1140	SC++	.24527E+01	2.00
1240	SC+	.24529E+01	1.00
2140	LC+	.22849E+01	1.00
2240	SC0	.24525E+01	0.00
1340	USC.	.25000E+01	1.00
3140	SUC.	.24000E+01	1.00
2340	DSC.	.25000E+01	0.00
3240	SDC.	.24000E+01	0.00
3340	SSC.	.26000E+01	0.00
1440	UCC.	.35500E+01	2.00
2440	DCC.	.35500E+01	1.00
3440	SCC.	.37000E+01	1.00
1111	DL++	.12320E+01	2.00
1121	DL+	.12320E+01	1.00
1221	DL0	.12320E+01	0.00
2221	DL-	.12320E+01	-1.00
1131	S*+	.13823E+01	1.00
1231	S*0	.13820E+01	0.00
2231	S*-	.13875E+01	-1.00
1331	XI*0	.15318E+01	0.00
2331	XI*-	.15350E+01	-1.00
3331	OM-	.16722E+01	-1.00
1141	UUC*	.26300E+01	2.00
1241	UDC*	.26300E+01	1.00
2241	DDC*	.26300E+01	0.00
1341	USC*	.27000E+01	1.00
2341	DSC*	.27000E+01	0.00
3341	SSC*	.28000E+01	0.00
1441	UCC*	.37500E+01	2.00
2441	DCC*	.37500E+01	1.00
3441	SCC*	.39000E+01	1.00
4441	CCC*	.48000E+01	2.00

3.5 ID Codes of Higher Resonances

In the following, we list id codes of higher resonances. In some cases, one id code represents the “average” of several resonances, the mass is correspondingly the average mass. Antiparticles are not listed.

id	label
332	F0(975)
112	A0(980)
122	A+(980)
1112	DL++(1620)
1113	DL++(1700)
1114	DL++(1925)
2222	DL-(1620)
2223	DL-(1700)
2224	DL-(1925)
1122	N*+(1440)
1123	N*+(1530)
1124	DL+(1620)
1125	N*+(1665)
1126	DL+(1700)
1127	N*+(1710)
1128	DL+(1925)
1222	N*0(1440)
1223	N*0(1530)

1224	DL0(1620)
1225	N*0(1665)
1226	DL0(1700)
1227	N*0(1710)
1228	DL0(1925)
1233	L(1405)
1234	L(1520)
1235	L(1645)
1236	S0(1665)
1237	S0(1776)
1238	L(1845)
1239	S0(1930)
1132	S+(1665)
1133	S+(1776)
1134	S+(1930)
2232	S-(1665)
2233	S-(1776)
2234	S-(1930)

3.6 ID Codes of Exotic Particles

We also defined id codes of exotic particles, referring to objects outside the resonance table (strings or droplets). The id for these objects is only meant to characterize the quark content, and not, as for the above particles, also the mass (or spin).

The general code consists of a 9 digit integer, the first one being the number '8'. the next 4 digits represent the numbers of u quarks, d quarks, s quarks, and c quarks. the following 4 digits represent the corresponding numbers of antiquarks. For example, 833000000 represents an object with 3 u quarks and 3 d quarks; 800303000 represents an object with 3 s quarks and 3 u antiquarks.

There is clearly a problem, when for (at least) one flavor there are more than 9 (anti)quarks. In this case one uses a code, again consisting of 9 digits, the first one now being the number '7'. The following 4 digits have to be considered as a four digit number representing the number of quarks, and the last four digits represent correspondingly the number of antiquarks. For example, 700270000 represents an object with 27 quarks; 701210013 represents an object with 121 quarks and 13 antiquarks.