1 ParticleID headers

1.1 ParticleID.hh

namespace HepPDT

Free functions:

double spinitod(int js);
int spindtoi(double spin);

Public members:

enum location { nj=1, nq3, nq2, nq1, nl, nr, n, n8, n9, n10 };
struct Quarks { short nq1; short nq2; short nq3; };

CLASS ParticleID

Public Methods:

ParticleID(int pid = 0);
The constructor.

ParticleID(const ParticleID & orig);
The copy constructor.

ParticleID & operator=(const ParticleID &);
The assignment constructor.

void swap(ParticleID & other);
The swap constructor.

bool operator < (ParticleID const & other) const;
Comparison operator.

bool operator == (ParticleID const & other) const;
Equality operator.

int pid( ) const;
Returns the PID.

int abspid( ) const;
Returns the absolute value of the PID.

bool isValid( ) const;
Returns true if this integer obeys the numbering scheme rules.

bool isMeson( ) const;

Returns true if this integer obeys the meson portion of the numbering scheme rules

bool isBaryon( ) const;
Returns true if this integer obeys the baryon portion of the numbering scheme rules.

bool isDiQuark( ) const;
Returns true if this integer obeys the diquark portion of the numbering scheme rules.

bool isHadron( ) const;
Returns true if either isBaryon or isMeson is true.

bool isLepton( ) const;
Returns true if the fundamentalID is 11-18.

bool isNucleus( ) const;
Returns true if this integer obeys the ion numbering scheme rules.

bool isPentaquark( ) const;
Returns true if this integer obeys the pentaquark numbering scheme rules.

bool hasUp( ) const;
Returns true if this is a valid PID and it has an up quark.

bool hasDown( ) const;
Returns true if this is a valid PID and it has a down quark.

bool hasStrange( ) const;
Returns true if this is a valid PID and it has a strange quark.

bool hasCharm( ) const;
Returns true if this is a valid PID and it has a charm quark.

bool hasBottom( ) const;
Returns true if this is a valid PID and it has a bottom quark.

bool hasTop( ) const;
Returns true if this is a valid PID and it has a top quark.

In practice, it is better to query the ParticleData class.

int jSpin( ) const;
jSpin returns 2J+1, where J is the total spin

int sSpin( ) const;
sSpin returns 2S+1, where S is the spin

int lSpin( ) const;
lSpin returns 2L+1, where L is the orbital angular momentum

int fundamentalID( ) const;
Returns the first 2 digits if this is a valid PID and it is neither neither a meson, a baryon, nor a diquark. If this is a meson, baryon, or diquark, fundamentalID returns zero.

int extraBits( ) const;
Returns any digits beyond the 7th digit (e.g. outside the numbering scheme).

Quarks quarks( ) const;
Returns a struct with the 3 quarks.

int threeCharge( ) const;
Returns 3 times the charge, as inferred from the quark content.
If the fundamentalID is non-zero, then a lookup table is used.

int A( ) const;
If this is an ion, returns A.

int Z( ) const;
If this is an ion, returns Z.

unsigned short digit(location) const;
digit returns the base 10 digit at a named location in the PID

const std::string PDTname() const;
Returns the HepPDT standard name.

Private Members:

int itsPID;

1.2 ParticleIDTranslations.hh

namespace HepPDT

Free functions:

```
int translatePythiatoPDT( const int pythiaID );
int translatePDTtoPythia( const int pid );

int translateIsajettoPDT( const int isajetID );
int translatePDTtoIsajet( const int pid );

int translateHerwigtoPDT( const int herwigID);
int translatePDTtoHerwig( const int pid );

int translateQQtoPDT( const int qqID);
int translatePDTtoQQ( const int pid );

int translateGeanttoPDT( const int geantID);
int translatePDTtoGeant( const int pid );

int translatePDGtabletoPDT( const int pdgID);
int translatePDTtoPDGtable( const int pid );

int translateEvtGentoPDT( const int evtGenID );
int translatePDTtoEvtGen( const int pid );
```

1.3 ParticleName.hh

namespace HepPDT

Free functions:

std::string particleName(const int);
Returns the HepPDT standard name.

void listHepPDTParticleNames(std::ostream & os);
List all defined names.

bool validParticleName(const int);
Verify that this particle ID has a valid name.

typedef std::map< int, std::string > ParticleNameMap;
ParticleNameMap const & getParticleNameMap();
Access ParticleNameMap for other purposes.

Only getParticleNameMap is allowed to access ParticleNameMap. ParticleNameMap is initalized by the first call to getParticleNameMap. Because the map is static, this initialization only happens once. We use a data table so that compile time is not impacted.