Author: M.Cattaneo
Status: Draft 2
Last modified: 29th August 2000

PROPOSED LHCb coding convention: Physical Units

Deadline for comments: 11th September 2000
Presentation for approval: LHCb week in Milano
Implementation: Gaudi release 6

Background

In release 3 of Gaudi, it was decided to adopt the CLHEP convention for physical units, also used by GEANT4, as shown below:

CLHEP system of units	
Length	*millimetre*
Time	*nanosecond*
Energy	*MeV*
Electric charge	*positron charge*
Temperature	*Kelvin*
Amount of substance	*mole*
Plane angles	*radian*
Solid angles	*steradian*

This choice of units may not seem natural in some cases (why MeV rather than GeV?) and indeed leads to some inconsistencies even within Gaudi - for example, in the Gaudi detector store, more familiar units are used for quantities such as density (g/cm**3) and radiation length (cm).

However, we cannot really avoid using CLHEP units if we want our code to be able to use the CLHEP classes transparently.

Proposal

Users should not actually need to know what units are used in the internal representation of the data, as long as they are consistent throughout the Gaudi data stores. What they care about is that they can define and plot quantities with the correct units. In some specialised algorithms they may also wish to renormalise the data to a different set of units, if the default set would lead to numerical precision problems.

We therefore propose the following rules:

1. All dimensioned quantities in the Gaudi data stores shall conform to the CLHEP system of units. Any inconsistencies in the current version of Gaudi will be removed in

the next release.

2. All definitions of dimensioned quantities shall be dimensioned by *multiplying* by the units defined in the `CLHEP/Units/SystemOfUnits.h` header file. For example:

   ```
   const double my_height = 170*cm;
   const double my_weight = 75*kg;
   ```

   Note that I don't care about the value of the numbers my_height and my_weight. Internally these numbers are represented as 1700. and 4.68e+26. respectively, but I do not need to know.

3. All output of dimensioned quantities should be converted to the required units by *dividing* by the units defined in the `CLHEP/Units/SystemOfUnits.h` header file. For example:

   ```
   my_hist = histoSvc()->book( "/stat/diet",  "corpulence (kg/m**2)",
   30, 10., 40. );
   double my_corpulence = my_weight / (my_height*my_height);
   my_hist->fill( my_corpulence/(kg/m2), 1. );
   ```

   which should plot a number between 19. and 25., but which for me is nearer 26...

4. Physical constants should not be defined in LHCb code. They should be taken directly from the `CLHEP/Units/PhysicalConstants.h` header file. For example:

   ```
   float my_rest_energy = my_weight * c_squared;
   ```

5. Sub-detectors may wish to use a different set of units for specific purposes (e.g. when the default units may lead to precision problems). In this case algorithms can renormalise their private copy of the data (as shown in the last line of the rule 3 example) for internal use, but making sure that any data subsequently published to the public data stores is converted back to the CLHEP units.

The proposal is to make rule 1 mandatory with the next release of Gaudi. If the Brunel contacts agree, rules 2, 3 and 4 will be mandatory in Brunel from release v2. Rule 5 should be used only in well-documented situations.

Implications

This proposal may have consequences for the run-time efficiency of the code. It should be checked that multiplication and division by 1. (e.g. 76.*mm) is optimised away by the compiler.

For consistency, the Gaudi jobOptions should be able to parse units in the same format. This is envisaged for Gaudi release 6.