Contents

1 What is Beta? 3

2 Where to get information? 4

3 SLAC UNIX account 4

4 What you need to change in your account set-up at UCSB 5

5 Setting up a test release 5

6 A few words about Objectivity 6

7 Compiling, linking, and running 7

8 Creating your own module 11

8.1 A brief overview of module structure 11

8.2 A simple way to create your own module 12

8.3 Handling lists 14

9 What are the Beast tapes? 18

10 Using EvtGen 19

11 Running BBSIM 20

```
21 (1)   B z
 5 (1)   mass of selected K0s
17 (1)   di-neutral spectrum
 8 (1)   mass of correct J/Psis
```

Let us now see where these histograms were declared and how they were created.

8 Creating your own module

8.1 A brief overview of module structure

There is some documentation about the BaBar application framework on the Web at the following URL:

http://www.slac.stanford.edu/BFROOT/doc/Computing/NewUser/htmlbug/bug.html

and in particular Chapter 7 talks about modules.

You can also go directly to

/babar2/BFROOT/dist/releases/6.0.2/Framework/Framework

and then look at the corresponding header files, such as APPFramework.hh and APPModule.hh. This is tedious, but it might give you more detailed information about the module structure, as it is implemented in BaBar software. Finally, some useful module examples are located in

/babar2/BFROOT/dist/releases/6.0.2/FrameExample

Very briefly, the Framework package of BaBar allows one to handle basic event-processing tasks. The basic unit of the Framework is a *module*. All modules inherit from the base class APPModule in Framework, so, just like APPModule, they all have methods for initialization, event processing, and ending. The main user interface used by the Framework is the Tcl interpreter. It allows one to enable and disable modules, combine them into sequences and paths, set the number of events to process, etc..

Let us now consider an example. Go to /babar2/your_directory/my6.0.2/BetaUser. There is a great sample module there, SampleBetaAnalysis, written by Bob Jacobsen. Let's see what its main parts are. First, look at the header file, SampleBeta-Analysis.hh. You see in the class declaration (after the comment "class interface") that this class inherits from AppModule:

```
class SampleBetaAnalysis : public AppModule {
```

Then, there are five methods declared after the constructor and destructor:

```
virtual void          begin( AppJob* );
virtual void          begin( AppRun* );
virtual AbsEvent* event( AbsEvent* anEvent, int mode = 0 );
virtual void          end  ( AppRun* );
virtual void          end  ( AppJob* );
```

Here, "begin" is the initialization method, "event" is the main event processing method, and "end" is the termination method.

In the "protected" part of the program, there are a number of objects of the type "AbsParmIfdStrKey" (this class is also in Framework). These objects refer to various lists of BtaCandidates - e.g., _btaTrackList refers to a list of all charged tracks, _btaCalorList refers to a list of all calorimeter objects, etc. Finally, there are several pointers to the class HepHistogram, which (just as one would expect) allow one to create histograms by using the so-called *global* histogram manager declared in the .cc file. In section "Handling lists", we will add our own histogram to the existing ones.

8.2 A simple way to create your own module

Let us now try to create our own module. In my view, the simplest thing to do is to take an existing module and modify it. So, let's copy SampleBetaAnalysis.cc and SampleBetaAnalysis.hh to MySampleBetaAnalysis.cc and MySampleBetaAnalysis.hh, respectively. Now, let's first edit MySampleBetaAnalysis.hh, the header file. Let us make sure all the titles are right. Find the two lines at the beginning of the file which say:

```
#ifndef SAMPLEBETAANALYSIS_HH
#define SAMPLEBETAANALYSIS_HH
```

and change it to

```
#ifndef MYSAMPLEBETAANALYSIS_HH
#define MYSAMPLEBETAANALYSIS_HH
```

Now, after the comment "Class Interface", change the class declaration to