BRUNEL

LHCb Reconstruction Program

User Guide

Version: 1
Issue: 1
Edition: 0
Status:
ID: [Document ID]
Date: 28 July 2000

European Laboratory for Particle Physics
Laboratoire Européen pour la Physique des Particules
CH-1211 Genève 23 - Suisse

Document Control Sheet

Table 1 Document Control Sheet

Document	**Title:**	BRUNEL User Guide		
	Version:	1		
	Issue:	1		
	Edition:	0		
	ID:	[Document ID]		
	Status:			
	Created:	25 May 2000		
	Date:	28 July 2000		
	Access: :			
	Keywords:			
Tools	**DTP System:**	Adobe FrameMaker	**Version:**	5.5
	Layout Template:	Software Documentation Layout Templates	**Version:**	V1 - 15 January 1999
	Content Template:	--	**Version:**	--
Authorship	**Coordinator:**	M.Cattaneo		
	Written by:	M.Cattaneo		

Document Status Sheet

Table 2 Document Status Sheet

Title:	BRUNEL User Guide		
ID:	[Document ID]		
Version	**Issue**	**Date**	**Reason for change**
1	0	26 May 2000	First draft version
1	1	28 July 2000	Minor changes for Brunel v1r2

Table of Contents

Chapter 1
Introduction

1.1 Purpose of this document

This document is a user guide and reference manual for the LHCb reconstruction program, Brunel. It should be useful both to users wishing to run the program, and to programmers wishing to add functionality.

This document does not describe the physics algorithms or the data model.

1.2 What does "Brunel" mean?

All LHCb data processing applications are based on a framework which enforces the GAUDI architecture. Antoni Gaudi [1] was a Catalan architect who greatly influenced the development of Barcelona around the beginning of the nineteenth century. For the reconstruction program we decided to use the name of an engineer. Isambard Kingdom Brunel [2] was a British engineer who greatly contributed to the industrial revolution in the first half of the eighteenth century.

1.3 Editor’s note

This document is a snapshot of the Brunel software at the time of the release of version v1r2. We have made every effort to ensure that the information it contains is correct, but in the event of any discrepancies between this document and information published on the Web, the latter should be regarded as correct, since it is maintained between releases and, in the case of code documentation, it is automatically generated from the code.

We encourage our readers to provide feedback about the structure, contents and correctness of this document and of other Gaudi documentation. Please send your comments to the editor, *Marco.Cattaneo@cern.ch*

Chapter 2
Structure of Brunel

2.1 Brunel Phases

The LHCb reconstruction program, Brunel, is composed of a number of Gaudi algorithms: `BrunelInitialisation`, `BrunelFinalisation` and a number of `BrunelPhase`s.

`BrunelInitialisation` is where all initialisations which are independent of BrunelPhase are performed. These can be global program initialisations (in the `initialise()` method), or event by event initialisations (in the `execute()` method). Note that initialisations specific to a given BrunelPhase should not be performed here.

`BrunelFinalisation` is where all finalisations which are independent of BrunelPhase are performed. These can be global program finalisations (in the `finalise()` method), or event by event finalisations (in the `execute()` method). Note that finalisations specific to a given BrunelPhase should not be performed here.

`BrunelPhase` is where the meat of the reconstruction program lies. BrunelPhase is a base class from which actual phases are derived. Each BrunelPhase should be independent of other BrunelPhases: it should be possible to run only one phase, providing of course event input data in the appropriate format exists[1]. All initialisations and finalisations specific to the phase should be performed inside the phase. The following BrunelPhases are currently implemented:

- **`BrunelDigi`** is where simulated RAW Hits are converted into DIGItisings. The output of this phase has the same format as real RAW data coming from the detector[2]. Obviously this phase would not be present when reconstructing real data, and could be moved to the simulation program when reconstructing simulated data. Note that this implies some discipline when designing the DIGItised data model, in particular for what concerns links to Monte Carlo truth information.

1. This is not entirely true in the current version of the reconstruction program, due to the underlying calls to SICBDST routines which do not have this structure.

- **`BrunelTrigger`** is where the LHCb trigger decision is applied. The input event data are DIGItisings. The output are also DIGItisings, with the addition of the trigger decision information.
- **`BrunelReco`** is where the first pass reconstruction is carried out. By first pass we mean that the reconstruction algorithms in this phase rely only on DIGItisings and do not require input from the reconstruction of other subdetectors. This restriction can be somewhat relaxed by ensuring that subdetectors are reconstructed in a specific order: those that only require input from the DIGItisings are processed first, those that require input from the reconstruction of other sub-detectors are processed after those sub-detectors.
- **`BrunelFinalFit`** is the second pass reconstruction, to allow for processing which requires input from the reconstruction of several subdetectors.

Note that additional phases could easily be implemented if further reconstruction passes are required.

2.1.1 Instantiating Brunel Phases

Brunel Phases are Gaudi top Algorithms. They are therefore instantiated using the standard Gaudi job option `ApplicationMgr.TopAlg` [3]. Listing 1 shows the value of this option for the current implementation. Note the different phases in lines 2 to 5, which are different instances of the `class BrunelPhase` and will be executed in the order shown

Listing 1 Brunel Top Algorithms

```
1:  ApplicationMgr.TopAlg = { "BrunelInitialisation/BrunelInit",
2:                            "BrunelPhase/BrunelDigi",
3:                            "BrunelPhase/BrunelTrigger",
4:                            "BrunelPhase/BrunelReco",
5:                            "BrunelPhase/BrunelFinalFit",
6:                            "BrunelFinalisation/BrunelFinish" };
```

2.2 Brunel sub-detector code

It is expected that sub-detector specific code will be executed inside one or more Brunel Phases. Each Brunel Phase instantiates a sub-algorithm for each detector participating in that phase. The class name of the sub-algorithm has to follow a specific convention: it is composed of the Phase name (e.g. `BrunelDigi`) followed by the abbreviated sub-detector name (e.g. `MUON`). These sub-algorithms are intended to be the phase specific steering algorithms of the sub-detectors.

The advantage of this system is that it is easily extendable and modifiable. To add a new phase, or a new sub-detector, or to change the name of a phase or subdetector, it is sufficient to

2. This is not entirely true in the current version of the reconstruction program, due to the underlying use of the SICB event data model, which does not have this structure.

provide an appropriately named algorithm and to make the necessary changes to the job options to instantiate this new algorithm. No changes are necessary to the Brunel steering code.

The list of sub-detector names and sub-algorithm classes currently implemented is shown in Table 3

Table 3 Sub-detector algorithms currently implemented in Brunel

Sub-detector	Abbreviation	Algorithms implemented
Electromagnetic Calorimeter	ECAL	`BrunelDigiECAL` `BrunelRecoECAL`
Hadron Calorimeter	HCAL	`BrunelDigiHCAL` `BrunelRecoHCAL`
Muon Detector	MUON	`BrunelDigiMUON`
Ring Imaging Cherenkov	RICH	`BrunelDigiRICH` `BrunelRecoRICH`

Tracking Detectors	TRAC	`BrunelDigiTRAC`
		`BrunelRecoTRAC`
		`BrunelFinalFitTRAC`

Trigger System	TRIGGER	`BrunelTriggerTRIGGER`
Vertex Locator	VELO	`BrunelDigiVELO`

2.2.1 Instantiating sub-detector algorithms

The reason for the naming convention described above is to provide a simple method for selecting which sub-detectors to reconstruct and in which order. It is sufficient to provide a `DetectorList` job option for each phase, containing the list of sub-detectors to be processed in that phase, as shown in Listing 2.

Listing 2 Processing order of sub-detector algorithms in Brunel.

```
1:  BrunelDigi.DetectorList = { "VELO","TRAC","RICH","ECAL","HCAL","MUON" };
2:  BrunelTrigger.DetectorList = { "TRIGGER" };
3:  BrunelReco.DetectorList = { "TRAC" , "RICH" , "ECAL" , "HCAL" };
4:  BrunelFinalFit.DetectorList = { "TRAC" };
```

2.3 Accessing Gaudi services and data from Brunel

Brunel sub-detector algorithms are instances of Gaudi algorithms. As such they have access to all the services currently implemented in Gaudi, and to all data in the Gaudi data stores. Please refer to the Gaudi user guide [3] for details.

2.4 Adding user code

User code can be added to Brunel in several ways:

1. By providing a Gaudi Algorithm that can be run as a top algorithm outside of a Brunel phase. This would typically be a monitoring algorithm that would analyse the progress of the recostruction. It can be inserted into the application by declaring it as an additional `ApplicationMgr.TopAlg` in the job option shown in Listing 1.
2. By providing a new sub-detector algorithm to be called within a given Brunel Phase. The new algorithm should have a class name following the convention described in section 2.2 . It is instantiated by adding the appropriate sub-detector name to the `DetectorList` job option shown in Listing 2. Please look at an existing sub-detector algorithm for details of the structure of such an algorithm.
3. By replacing an existing sub-detector algorithm.
4. In the current implementation, it is also possible to add a Fortran analysis routine, using the SICB user routined SUINIT, SUANAL, SULAST. SUANAL is called at the end of all event processing.

Chapter 3
Current Implementation

The current version of Brunel implements the "wrapping" of SICBDST Fortran code: Brunel is simply a skeleton within Gaudi which calls the full set of SICBDST Fortran algorithms. There are no C++ algorithms in this version.

3.1 Wrapped SICBDST

The SICBDST code has been wrapped into Brunel Phases and Brunel sub-detector algorithms. Each algorithm calls the corresponding FORTRAN steering routine, as summarised in Table 4

Table 4 List of wrapped SICBDST sub-detector steering routines

Brunel Algorithm	SICBDST steering routine
BrunelInitialisation	DETINIT (calls ECINIT, ECDINIT, HCINIT, HCDINIT, MUGINIT, MPINIT)
BrunelDigiVELO	VSDIGI
BrunelDigiTRAC	WDDIGI
BrunelDigiRICH	RIDIGI
BrunelDigiECAL	ECDIGI
BrunelDigiHCAL	HCDIGI
BrunelDigiMuon	MUDIGI
BrunelTriggerTRIGGER	TRIGGER
BrunelRecoTRAC	AXTFIT
BrunelRecoRICH	RIRECO

Table 4 List of wrapped SICBDST sub-detector steering routines

Brunel Algorithm	SICBDST steering routine
BrunelRecoECAL	ECRECO
BrunelRecoHCAL	HCRECO
BrunelFinalFitTRAC	AXRECO
BrunelFinalisation	SUANAL RECEVOUT

Communication between the various FORTRAN algorithms is done, as in SICBDST, via COMMON blocks, in particular the ZEBRA common blocks. The FORTRAN algorithms are controlled via the SICB data cards file. All data cards recognised by SICBDST are valid, with the exception of cards dealing with input event data (TRIGGERS card, IOPA 'GETX', 'GETY', 'GETZ' cards) and selection of processing steps (SKIP data card). Please refer to the SICB documentation [4] for details

3.2 Input/Output definition

The current version of Brunel uses the Gaudi `SicbEventSelector` to read in event data from a SICBMC RAWH file. The relevant job options are shown in Listing 3

Listing 3 Job Options for event input definition

```
// Input file name (all on one line!)
ApplicationMgr.EvtSel = "JOBID 19612";
// Number of events to be processed (default is 10)
EventSelector.EvtMax = 100;
// Print event number at each event
EventSelector.PrintFreq = 1;

// Enable next card if you wish to skip some events
// EventSelector.FirstEvent = 3;
```

Event pileup is implemented by the Gaudi framework. In order to switch it on in Brunel, you have to add the job options shown in Listing 4

Listing 4 Job Options for pileup

```
// Declare the Pileup event Selector
ApplicationMgr.ExtSvc += { "SicbEventSelector/PileUpSelector" };
// Define the Pileup mode
PileUpAlg.PileUpMode = "LUMI";
// Define the file containing the pileup events
PileUpSelector.JobInput = "JOBID 19065";
```


In addition, you have to declare the `PileUpAlg` algorithm as the first top algorithm. The `ApplicationMgr.TopAlg` job option that was shown in Listing 1 becomes as shown in Listing 5

Listing 5 Brunel Top Algorithms when doing pileup

```
1:  ApplicationMgr.TopAlg = { "PileUpAlg",
2:                            "BrunelInitialisation/BrunelInit",
3:                            "BrunelPhase/BrunelDigi",
4:                            "BrunelPhase/BrunelTrigger",
5:                            "BrunelPhase/BrunelReco",
6:                            "BrunelPhase/BrunelFinalFit",
7:                            "BrunelFinalisation/BrunelFinish" };
```

The Gaudi framework does not provide a facility for writing out event data to ZEBRA files. For this reason, Brunel calls the SICB routine RECEVOUT to write out the SICB DST file. The output stream is defined using an IOPA 'SAVX' data card as for SICBDST [4].

In addition, it is possible to write out an object-oriented DST to a ROOT file, using the facilities provided by Gaudi. Please refer to the Gaudi manual [3] for details

3.3 Histograms

In the current version of Brunel, the only predefined histograms are those created by the subdetector code inside SICBDST, control of filling and of output of these histograms is via the SICB data cards in the usual way.

The standard SICBDST checking histograms are also linked into Brunel by default. This is done by line 2 in the code fragment below, part of the Brunel requirements file.

```
1:  application Brunel ../Brunel/*.cpp \
2:  #/afs/cern.ch/lhcb/software/LHCbCMT/SICBDST/v233r2/dst/*.F \
3:   ../Brunel/*.F
```

Fill of these histograms is enabled with the following SICB data card:

```
IOPA
    'CHCK' 'HO' '$WORKDIR/Brunel.hbook!'
```

In addition, it is possible for users to define their own histograms inside Gaudi algorithms, using the facilities provided by Gaudi. Such histograms are output by the Gaudi histogram service, to a file defined by the following card

```
HistogramPersistencySvc.OutputFile  = "histo.hbook";
```


3.4 Debug printout

In the current version of Brunel, control of debug printout from the SICBDST Fortran algorithm is via the SICB data cards in the usual way.

In addition, it is possible to define the level of debug printout available from the Gaudi Services and from the Brunel control framework via the standard Gaudi MessageSvc job options:

```
// Global output level
MessageSvc.OutputLevel     = 3;

// Over-ride global level for some algorithms
BrunelInit.OutputLevel     = 2;
BrunelDigiVELO.OutputLevel = 2;
BrunelRecoVELO.OutputLevel = 2;
```


Chapter 4
Running Brunel

Brunel is implemented as a CMT [5] package, with the following subdirectory structure:

- `Brunel` C++ and Fortran source code
- `doc` release notes
- `job` example jobs and steering data cards
- `mgr` CMT `requirements` file
- `Visual` Visual Studio Workspace

The `job` subdirectory contains two example jobs for running Brunel either interactively or in batch on Linux, and examples of a Gaudi job options file (`BrunelOptions.txt`) and a SICB data file (`Brunel.cards`). You should customise these two files according to your needs.

The files `BrunelOptions.txt` and `Brunel.cards` are picked up by default when you run one of the example jobs on Linux, or inside Visual Studio on NT. To pick up different files, you should modify the following two lines in the `requirements` file

```
# Set the paths for Brunel and SICBDST data cards.
set JOBOPTPATH ${BRUNELROOT}/job/BrunelOptions.txt
set SICBCARDS ${BRUNELROOT}/job/Brunel.cards
```


Appendix A
References

1 See for example *http://www.gaudiclub.com/ingles/i_vida/i_menu.html* for more information about Antoni Gaudi

2 See for example *http://www.spartacus.schoolnet.co.uk/RAbrunel.htm* for more information about Isambard Kingdom Brunel

3 The GAUDI users guide is available at: *http://lhcb.cern.ch/computing/Components/Gaudi_v5/gug.pdf*

4 The SICB documentation is available at: *http://lhcb.cern.ch/computing/SICB/*

5 CMT documentation is available at *http://lhcb.cern.ch/computing/Support/html/cmt.htm*