

RootMaker

1.0

Generated by Doxygen 1.7.4

Fri Jun 15 2012 21:54:52

Contents

1	Main Page	1
1.1	Introduction	1
1.2	Installation	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	Event Struct Reference	7
4.1.1	Detailed Description	7
4.2	EventHit Struct Reference	7
4.2.1	Detailed Description	8
5	File Documentation	9
5.1	dumproot.cc File Reference	9
5.1.1	Detailed Description	10
5.1.2	Function Documentation	10
5.1.2.1	main	10

Chapter 1

Main Page

1.1 Introduction

The RootMaker software can be used to convert standardized (40 scintillators, can easily be changed) output from the PIBIDS simulation directly to a root file in order to make a preliminary data analysis using ROOT.

1.2 Installation

For installation, download the RootMaker software from [here](#), unpack it and run "make". Note that a correctly configured installation of [ROOT](#) is required in order for the Root-Maker to work correctly.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Event (Constitues an event (3 event hits + event number and A-Z information))	7
EventHit (Constitutes an event hit)	7

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

[dumproot.cc](#) (File containing the RootMaker software source code) 9

Chapter 4

Class Documentation

4.1 Event Struct Reference

Constitutes an event (3 event hits + event number and A-Z information).

Public Attributes

- int [eventno](#)
Event number.
- int [A](#)
A of the atom.
- int [Z](#)
Z of the atom.
- [EventHit hit](#) [3]

4.1.1 Detailed Description

Constitutes an event (3 event hits + event number and A-Z information).

Definition at line 33 of file `dumproot.cc`.

The documentation for this struct was generated from the following file:

- [dumproot.cc](#)

4.2 EventHit Struct Reference

Constitutes an event hit.

Public Attributes

- double [time](#)
Time of event hit.
- float [UFSP](#)
Energy deposited in UFSP.
- float [LFSP](#)
Energy deposited in LFSP.
- float [BSP](#)
Energy deposited in BSP.
- float [Ge](#)
Energy deposited in Ge detector.
- float **DET** [NUMBER_OF_TUBES]

4.2.1 Detailed Description

Constitutes an event hit.

Definition at line 19 of file dumproot.cc.

The documentation for this struct was generated from the following file:

- [dumproot.cc](#)

Chapter 5

File Documentation

5.1 dumproot.cc File Reference

File containing the RootMaker software source code.

```
#include <stdlib.h>
#include <stdio.h>
#include "TFile.h"
#include "TTree.h"
```

Classes

- struct [EventHit](#)
Constitutes an event hit.
- struct [Event](#)
Constitutes an event (3 event hits + event number and A-Z information).

Defines

- #define **xstr**(s) str(s)
- #define **str**(s) #s
- #define **NUMBER_OF_TUBES** 40

Functions

- int [main](#) (int argc, char **argv)

5.1.1 Detailed Description

File containing the RootMaker software source code.

Author

Rikard Lundmark, Håkan Johansson

Definition in file [dumproot.cc](#).

5.1.2 Function Documentation

5.1.2.1 `int main (int argc, char ** argv)`

Parameters

<i>argc</i>	Numer of arguments
<i>argv</i>	ArgumentsMain function

Definition at line 42 of file dumproot.cc.

References `Event::eventno`, and `EventHit::time`.

```
{
    if (argc < 3)
    {
        fprintf (stderr, "Usage: %s <infile.txt> <outfile.root>\n", argv[0]);
        exit(1);
    }

    TFile hfile(argv[2], "RECREATE");

    // Create a ROOT Tree

    TTree *tree = new TTree("T", "Simulation result.");

    Event event;

    // Create branches.

    tree->Branch("event", &event.eventno,
                "eventno/I:A/I:Z/I");

    //hopefully implantation.
    tree->Branch("hit1", &event.hit[0].time,
                "h1T/D:h1UFSP/F:h1LFSP/F:h1BSP/F:h1Ge/F:h1DET[" xstr(NUMBER_OF_TUB
                ES) "]/F");
    //hopefully decay.
    tree->Branch("hit2", &event.hit[1].time,
                "h2T/D:h2UFSP/F:h2LFSP/F:h2BSP/F:h2Ge/F:h2DET[" xstr(NUMBER_OF_TUB
                ES) "]/F");
    //some other decay?
    tree->Branch("hit2", &event.hit[2].time,
                "h3T/D:h3UFSP/F:h3LFSP/F:h3BSP/F:h3Ge/F:h3DET[" xstr(NUMBER_OF_TUB
                ES) "]/F");

    FILE * inputFile = fopen(argv[1], "r");
```

```
if(inputFile==NULL)
{
    fprintf(stderr,"Could not open input file, exiting.");
    exit(1);
}

int numberOfFoundEvents = 0;
int currentHitPtr=0;
while(!feof(inputFile))
{
    int tmpEventNo, tmpA, tmpZ;
    double tmpTime;
    float tmpUFSP, tmpLFSP, tmpBSP, tmpGe;
    float tmpTube[NUMBER_OF_TUBES];

    //event# Z A time upper lower back Ge Tubes[0-39]
    fscanf(inputFile,"%d %d %d",&tmpEventNo, &tmpA, &tmpZ);

    fscanf(inputFile, "%le", &tmpTime);

    fscanf(inputFile,"%e %e %e %e", &tmpUFSP, &tmpLFSP, &tmpBSP, &tmpGe);

    for(int i = 0; i<NUMBER_OF_TUBES; i++)
    {
        fscanf(inputFile,"%e",&tmpTube[i]);
    }

    if(event.eventno!=tmpEventNo) //Create new event.
    {
        if(numberOfFoundEvents>0) //don't fill the tree if there is nothing to
        fill it with.
            tree->Fill();
        currentHitPtr=0;
        ++numberOfFoundEvents;
    }
    if(currentHitPtr==0)
    {
        memset(&event,0,sizeof(event));
    }
    if(currentHitPtr<2)
    {
        event.eventno=tmpEventNo;
        event.A = tmpA;
        event.Z = tmpZ;
        event.hit[currentHitPtr].time=tmpTime;
        event.hit[currentHitPtr].UFSP=tmpUFSP;
        event.hit[currentHitPtr].LFSP=tmpLFSP;
        event.hit[currentHitPtr].BSP=tmpBSP;
        event.hit[currentHitPtr].Ge=tmpGe;
        for(int i = 0; i<NUMBER_OF_TUBES; i++)
        {
            event.hit[currentHitPtr].DET[i]=tmpTube[i];
        }
    }
    ++currentHitPtr;
}
++numberOfFoundEvents;
tree->Fill();

fclose(inputFile);
hfile.Write();
hfile.Close();
```

```
    printf("A total of %d events was processed.\n",numberOfFoundEvents);  
    return 0;  
}
```