# Satellite Simulation on RaspberryPI

v2.0

Generated by Doxygen 1.8.6

Fri Mar 7 2014 13:00:15

# Contents

# 1 Hierarchical Index

## 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

# 2 Class Index

## 2.1 Class List

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

# 3 File Index

## 3.1 File List

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

# 4 Class Documentation

## 4.1 action Struct Reference

Contents an instruction.

```
#include <Plan.h>
```

#### Public Attributes

- int numero_action
- string nature
- struct tm date_action
- long int numero_image
- double duree_action
- double angle_prise_vue [3]
- int nbre_images_a_envoyer
- int id_images_a_envoyer [10]

### 4.1.1 Detailed Description

Contents an instruction.

### 4.1.2 Member Data Documentation

#### 4.1.2.1 double action::angle_prise_vue[3]

Attitude angle of the satellite

#### 4.1.2.2 struct tm action::date_action

Date for the realisation of the instruction

#### 4.1.2.3 double action::duree_action

Duration of the action

#### 4.1.2.4 int action::id_images_a_envoyer[10]

Image Id to be transferred

#### 4.1.2.5 string action::nature

Type of the command (IMG, TSF, PLA)

#### 4.1.2.6 int action::nbre_images_a_envoyer

Number of images to be transferred

#### 4.1.2.7 int action::numero_action

Index of the instruction in the plan

#### 4.1.2.8 long int action::numero_image

ID of the image

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

- /home/william/ARINC653v2/sources/Chaire_SE_Student/Plan.h

## 4.2 Analyseur Class Reference

Analyses and compress the images from the camera.

```
#include <Analyseur.h>
```

**Public Member Functions**

- Analyseur ()

  *Constructor.*
- int traiter_image (char ∗)

  *Image analysis.*
- int compresser_image (char ∗)

  *Compress an image.*

### 4.2.1 Detailed Description

Analyses and compress the images from the camera.

It analyses the image and compares the analysis to a minimal value of quality (written in the plan). It also compress the file.

### 4.2.2 Member Function Documentation

#### 4.2.2.1 int Analyseur::compresser_image ( char ∗ *image_traitee* )

Compress an image.

Creates a .zip from an image file.

**Parameters**

| | |
|---|---|
| *char*∗ | : name of the file. |

**Returns**

0 for an error, 1 if completed.

#### 4.2.2.2 int Analyseur::traiter_image ( char ∗ *image_traitee* )

Image analysis.

Analyses an image regarding the cloud cover, (10% of chance to be rejected for simulate this phenomenon).

**Parameters**

| | |
|---|---|
| *char*∗ | : name of the file. |

**Returns**

0 for an error, 1 if completed.

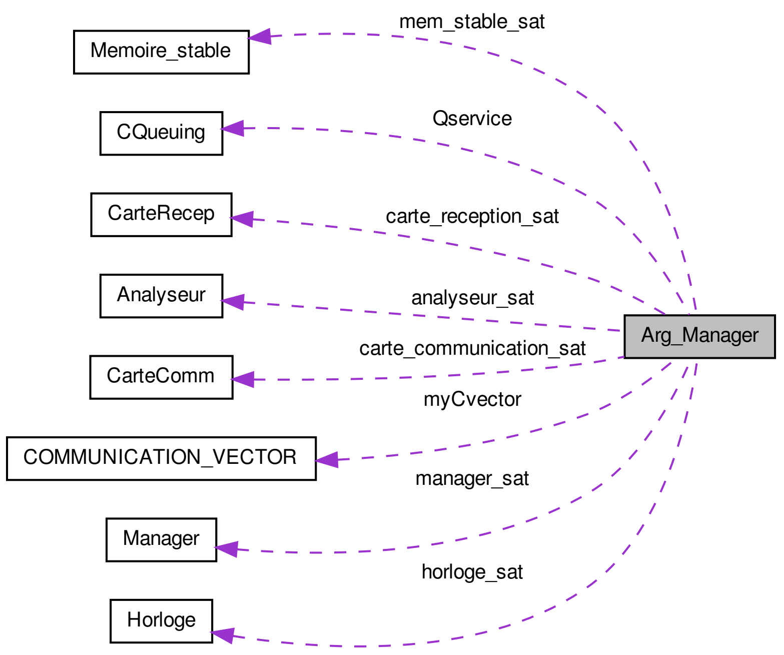
The documentation for this class was generated from the following files:

- /home/william/ARINC653v2/sources/Chaire_SE_Student/Analyseur.h
- /home/william/ARINC653v2/sources/Chaire_SE_Student/Analyseur.cpp

## 4.3 Arg_Manager Struct Reference

Content all the objects necessary to the main manager thread.

Collaboration diagram for Arg_Manager:



#### Public Attributes

- Analyseur ∗ **analyseur_sat**
- Memoire_stable ∗ **mem_stable_sat**
- Horloge ∗ **horloge_sat**
- Manager ∗ **manager_sat**
- COMMUNICATION_VECTOR **myCvector**
- CQueuing **Qservice**
- CarteComm ∗ **carte_communication_sat**
- CarteRecep ∗ **carte_reception_sat**
- int **myArgumentc**
- string **mode_fonctionnement**
- char ∗ **myArgumentv**

### 4.3.1 Detailed Description

Content all the objects necessary to the main manager thread.

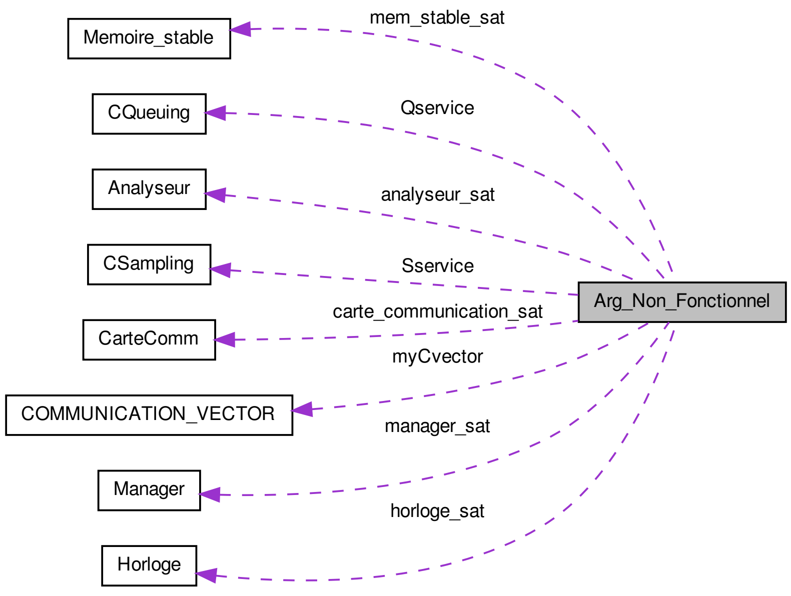
The documentation for this struct was generated from the following files:

- /home/william/ARINC653v2/sources/Chaire_SE_Student/Master/Master.cpp
- /home/william/ARINC653v2/sources/Chaire_SE_Student/Slave/Slave.cpp

## 4.4 Arg_Non_Fonctionnel Struct Reference

Content all the objects necessary to the non functionnal thread (including Watchdog).

Collaboration diagram for Arg_Non_Fonctionnel:



#### Public Attributes

- Analyseur ∗ **analyseur_sat**
- Memoire_stable ∗ **mem_stable_sat**
- Horloge ∗ **horloge_sat**
- Manager ∗ **manager_sat**
- COMMUNICATION_VECTOR **myCvector**
- CSampling **Sservice**
- CQueuing **Qservice**
- CarteComm ∗ **carte_communication_sat**
- string **mode_fonctionnement**
- int **myArgumentc**
- char ∗ **myArgumentv**

### 4.4.1 Detailed Description

Content all the objects necessary to the non functionnal thread (including Watchdog).

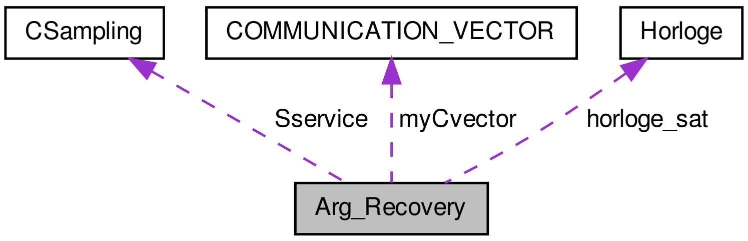
The documentation for this struct was generated from the following files:

- /home/william/ARINC653v2/sources/Chaire_SE_Student/Master/Master.cpp
- /home/william/ARINC653v2/sources/Chaire_SE_Student/Slave/Slave.cpp

## 4.5 Arg_Recovery Struct Reference

Content all the objects necessary to the recovery thread.

Collaboration diagram for Arg_Recovery:



#### Public Attributes

- CSampling **Sservice**
- COMMUNICATION_VECTOR **myCvector**
- Horloge ∗ **horloge_sat**

### 4.5.1 Detailed Description

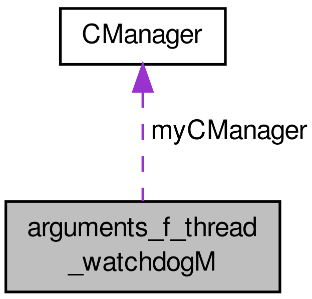
Content all the objects necessary to the recovery thread.

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

- /home/william/ARINC653v2/sources/Chaire_SE_Student/Master/Master.cpp
- /home/william/ARINC653v2/sources/Chaire_SE_Student/Slave/Slave.cpp

## 4.6 arguments_f_thread_watchdogM Struct Reference

Collaboration diagram for arguments_f_thread_watchdogM:



#### Public Attributes

- pid_t ∗ **pid_to_watch**
- CManager ∗ **myCManager**
- pid_t ∗ **pid_result**

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

- /home/william/ARINC653v2/sources/include/CManager.h

## 4.7 CArgument Class Reference

#### Public Member Functions

- std::vector< int > **split_arg** (std::string Arg)

The documentation for this class was generated from the following files:

- /home/william/ARINC653v2/sources/include/CArgument.h
- /home/william/ARINC653v2/sources/common/CArgument.cpp

## 4.8 CarteComm Class Reference

Emulates a TM controller.

`#include <CarteComm.h>`

#### Public Member Functions

- bool activer_carte ()

  *Activation protocol.*
- bool envoyer (char ∗, char ∗)

  *Transmission protocol.*
- bool desactiver_carte ()

  *Desactivation protocol.*
- bool get_etat ()

  *Accessor: active.*

#### Protected Attributes

- bool active

### 4.8.1 Detailed Description

Emulates a TM controller.

Emulates a TC controller.

This device is used to transmit images from satellite to earth using high speed communications.

This device is used to recieve mission plan from earth using low speed communications.

#### 4.8.2 Member Function Documentation

##### 4.8.2.1 bool CarteComm::activer_carte ( )

Activation protocol.

**Returns**

Status of the device (activated or not).

##### 4.8.2.2 bool CarteComm::desactiver_carte ( )

Desactivation protocol.

**Returns**

Status of the device (activated or not).

##### 4.8.2.3 bool CarteComm::envoyer ( char ∗ *id_image_a_envoyer,* char ∗ *machine_name* )

Transmission protocol.

This function uses scp protocol and rsa cryptage to send images to a distant machine.

**Parameters**

| | |
|---|---|
| *char*∗ | Image's id |
| *char*∗ | Name of the machine which will recieve the data. |

**Returns**

true in case of succes, else false.

##### 4.8.2.4 bool CarteComm::get_etat ( )

Accessor: active.

**Returns**

Status of the device

#### 4.8.3 Member Data Documentation

##### 4.8.3.1 bool CarteComm::active [protected]

Status of the device. True = Activated

The documentation for this class was generated from the following files:

- /home/william/ARINC653v2/sources/Chaire_SE_Student/CarteComm.h
- /home/william/ARINC653v2/sources/Chaire_SE_Student/CarteComm.cpp

### 4.9 CarteRecep Class Reference

**Public Member Functions**

- bool activer_carte ()

  *Activation protocol.*

- int recevoir_plan ()

    *Plan reception protocol.*
- bool desactiver_carte ()

    *Desactivation protocol.*
- bool get_etat ()

    *Accessor: active.*

**Protected Attributes**

- bool etat

### 4.9.1 Member Function Documentation

#### 4.9.1.1 bool CarteRecep::activer_carte ( )

Activation protocol.

**Returns**

Status of the device (activated or not).

#### 4.9.1.2 bool CarteRecep::desactiver_carte ( )

Desactivation protocol.

**Returns**

Status of the device (activated or not).

#### 4.9.1.3 bool CarteRecep::get_etat ( )

Accessor: active.

**Returns**

Status of the device

#### 4.9.1.4 int CarteRecep::recevoir_plan ( )

Plan reception protocol.

This function uses scp protocol and rsa cryptage to recieve plans from a distant machine.

**Returns**

0 for an error, else 1.

### 4.9.2 Member Data Documentation

#### 4.9.2.1 bool CarteRecep::etat [protected]

Status of the device. True = Activated

The documentation for this class was generated from the following files:

- /home/william/ARINC653v2/sources/Chaire_SE_Student/CarteRecep.h
- /home/william/ARINC653v2/sources/Chaire_SE_Student/CarteRecep.cpp

## 4.10 CConfig Class Reference

### Public Member Functions

- void **read_process** (std::vector< CPartition > &vpart, std::string filepath)
- void **read_communication** (std::vector< CPartition > &vpart, std::string filepath)

The documentation for this class was generated from the following files:

- /home/william/ARINC653v2/sources/include/CConfig.h
- /home/william/ARINC653v2/sources/simulateur/CConfig.cpp

## 4.11 Checkpoint Class Reference

Creation and management of the checkpoints.

`#include <Checkpoint.h>`

### Public Member Functions

- bool get_etat (int)

  *State accessor.*
- int get_id_image_sauvegarde ()

  *Accessor id_image_sauvegarde.*
- void set_checkpoint (int, int)

  *id_action & id_image_sauvegarde mutator.*
- int get_id_action ()

  *Accessor id_action.*
- void set_etat (int, bool)

  *Single state mutator.*
- void set_etat (int, int, bool)

  *Single state and image id mutator.*
- void set_tous_etats (Checkpoint)

  *All-states mutator.*
- void set_tous_etats (bool[5])

  *State mutator.*
- void set_taille_image (int)

  *Image weight mutator.*
- void set_taille_image (Checkpoint)

  *Image weight mutator.*
- void set_pointeur_mem (int)

  *Memory pointer mutator.*
- void set_pointeur_mem (Checkpoint)

  *Memory pointer mutator.*
- int get_taille_image ()

  *taille_image assessor*
- int get_pointeur_mem ()

  *pointeur_mem assessor*

#### Protected Attributes

- int id_image_sauvegarde
- int id_action
- bool etat [5]
- int taille_image
- int pointeur_mem

### 4.11.1 Detailed Description

Creation and management of the checkpoints.

### 4.11.2 Member Function Documentation

#### 4.11.2.1 bool Checkpoint::get_etat ( int *i* )

State accessor.

Returns the i-th state of an image.

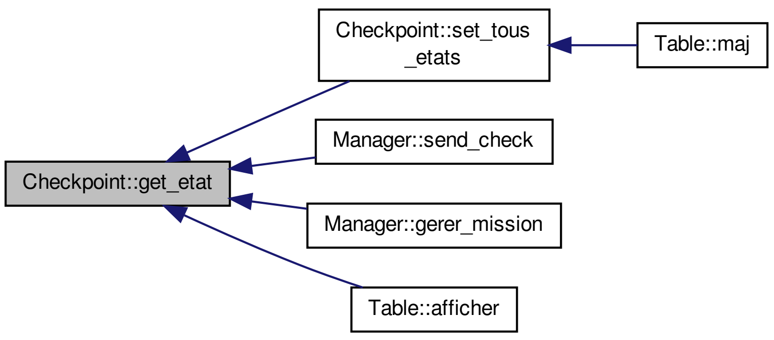
**Parameters**

| | |
|---|---|
| *int* | State index (from 0 to 4). |

**Returns**

Image's state.

Here is the caller graph for this function:



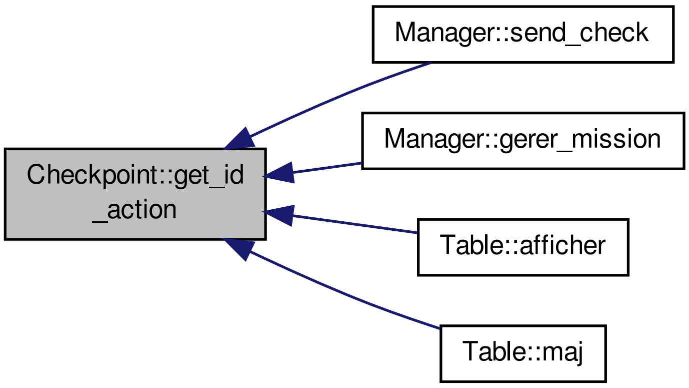
#### 4.11.2.2 int Checkpoint::get_id_action ( )

Accessor id_action.

**Returns**

id_action.

Here is the caller graph for this function:



#### 4.11.2.3 int Checkpoint::get_id_image_sauvegarde ( )

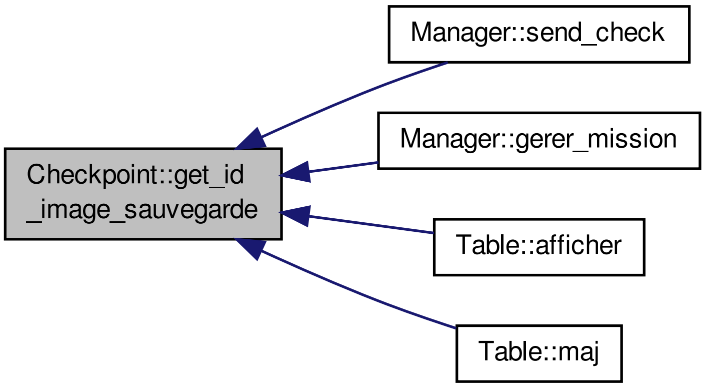
Accessor id_image_sauvegarde.

Returns the image's id stored in the checkpoint.

**Returns**

Image id.

Here is the caller graph for this function:



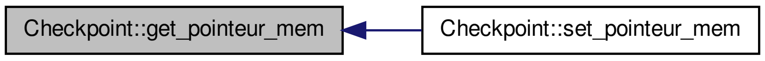
#### 4.11.2.4 int Checkpoint::get_pointeur_mem ( )

pointeur_mem assessor

**Returns**

pointeur_mem.

Here is the caller graph for this function:



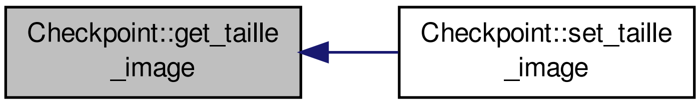
#### 4.11.2.5 int Checkpoint::get_taille_image ( )

taille_image assessor

**Returns**

taille_image.

Here is the caller graph for this function:



#### 4.11.2.6 void Checkpoint::set_checkpoint ( int *id_action_prog,* int *id_image* )

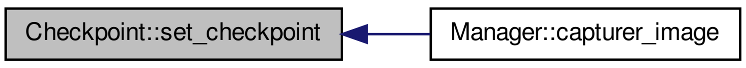
id_action & id_image_sauvegarde mutator.

**Parameters**

| | |
|---|---|
| *int* | New id_image_sauvegarde. |
| *int* | New id_action. |

<connaitre l'id de la sauvegarde de l'image

Here is the caller graph for this function:



#### 4.11.2.7 void Checkpoint::set_etat ( int *numero_etat,* bool *v_etat* )

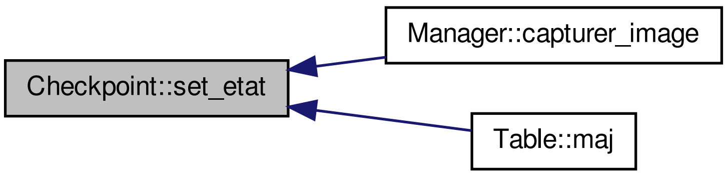
Single state mutator.

Method to change a specific state in the checkpoint.

**Parameters**

| | |
|---|---|
| *int* | State index (from 0 to 4). |
| *bool* | New value of the state. |

Here is the caller graph for this function:



#### 4.11.2.8 void Checkpoint::set_etat ( int *id_image,* int *numero_etat,* bool *v_etat* )

Single state and image id mutator.

Changes the image id and a state.

**Parameters**

| | |
|---|---|
| *int* | New image id. |
| *int* | State index (from 0 to 4). |
| *bool* | New value of the state. |

#### 4.11.2.9 void Checkpoint::set_pointeur_mem ( int *pointeur* )

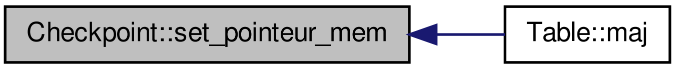
Memory pointer mutator.

Changes the values of pointeur_mem.

**Parameters**

| | |
|---|---|
| *int* | New value of pointeur_mem. |

Here is the caller graph for this function:



#### 4.11.2.10 void Checkpoint::set_pointeur_mem ( Checkpoint *check* )

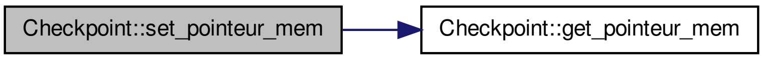
Memory pointer mutator.

Changes the values of pointeur_mem with the one of an existing checkpoint.

**Parameters**

| | |
|---|---|
| *Checkpoint* | Checkpoint contening the new value of pointeur_mem. |

Here is the call graph for this function:



#### 4.11.2.11 void Checkpoint::set_taille_image ( int *taille* )

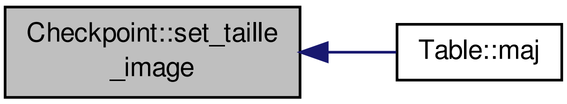
Image weight mutator.

Changes the values of taille_image.

**Parameters**

| | |
|---|---|
| *int* | New value of taille_image. |

Here is the caller graph for this function:



#### 4.11.2.12 void Checkpoint::set_taille_image ( **Checkpoint** *check* )

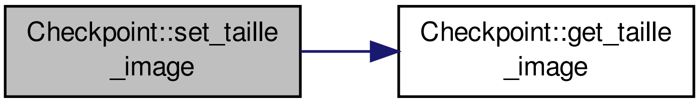
Image weight mutator.

Changes the values of taille_image with the one of an existing checkpoint.

**Parameters**

| | |
|---|---|
| *Checkpoint* | Checkpoint contening the new value of taille_image. |

Here is the call graph for this function:



#### 4.11.2.13 void Checkpoint::set_tous_etats ( **Checkpoint** *new_checkpoint* )

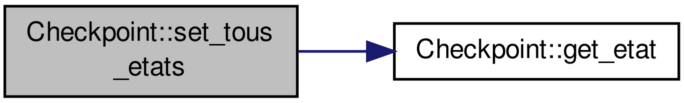
All-states mutator.

Changes all the values of a checkpoint with the values of another one.

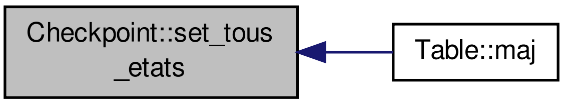
**Parameters**

| | |
|---|---|
| *Checkpoint* | Checkpoint from wich the values are taken. |

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.11.2.14 void Checkpoint::set_tous_etats ( bool *new_table_etats[5]* )

State mutator.

Changes all the states with the values of another one.

**Parameters**

| | |
|---|---|
| *bool∗* | New states. |

### 4.11.3 Member Data Documentation

#### 4.11.3.1 bool Checkpoint::etat[5] [protected]

States of the image: took, analysed, validated, stored, transmitted

#### 4.11.3.2 int Checkpoint::id_action [protected]

Index in the plan of the corresponding action

#### 4.11.3.3 int Checkpoint::id_image_sauvegarde [protected]

Image id

#### 4.11.3.4 int Checkpoint::pointeur_mem [protected]

Pointer in the memory

#### 4.11.3.5 int Checkpoint::taille_image [protected]

Weight of the image

The documentation for this class was generated from the following files:

- /home/william/ARINC653v2/sources/Chaire_SE_Student/Checkpoint.h
- /home/william/ARINC653v2/sources/Chaire_SE_Student/Checkpoint.cpp

## 4.12 CManager Class Reference

The documentation for this class was generated from the following files:

- /home/william/ARINC653v2/sources/include/CManager.h
- /home/william/ARINC653v2/sources/simulateur/CManager.cpp

## 4.13 communication_obj Class Reference

**Public Member Functions**

- **communication_obj** (int nbarg, char ∗argument[])
- char **get_emetteur** ()
- int **get_vsamp_socket** ()
- int **get_vqueuing_socket** ()
- int **get_vsamp_port** ()
- int **get_vqueuing_port** ()

The documentation for this class was generated from the following files:

- /home/william/ARINC653v2/sources/include/communication_obj.h
- /home/william/ARINC653v2/sources/common/communication_obj.cpp

## 4.14 COMMUNICATION_VECTOR Struct Reference

### Public Attributes

- std::string **emetteur**
- std::vector< int > **vsamp_socket**
- std::vector< int > **vqueuing_socket**
- std::vector< int > **vsamp_port**
- std::vector< int > **vqueuing_port**

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

- /home/william/ARINC653v2/sources/include/CCommunication.h

## 4.15 CPartition Class Reference

### Public Member Functions

- **CPartition** (std::string nameProcess, std::string pathProcess, int time)
- void **Display** ()
- std::string **nameProcess** ()
- std::string **pathProcess** ()
- std::vector< int > **get_wSport** ()
- std::vector< int > **get_wQport** ()
- std::vector< int > **get_rSport** ()
- std::vector< int > **get_rQport** ()
- std::vector< int > **get_vSsock** ()
- std::vector< int > **get_vQsock** ()
- int **time** ()
- int **wSport_add** (int wport)
- int **wQport_add** (int wport)
- int **rSport_add** (int rport)
- int **rQport_add** (int rport)
- void **vSsock_add** (int sock)
- void **vQsock_add** (int sock)

The documentation for this class was generated from the following files:

- /home/william/ARINC653v2/sources/include/CPartition.h
- /home/william/ARINC653v2/sources/simulateur/CPartition.cpp

## 4.16 Cport_service Class Reference

### Public Member Functions

- int **CREATE_SAMPLING_PORT** (int portID, int portName, int maxMessage_size, bool portDirection, int refreshPeriod)
- int **CREATE_QUEUING_PORT** (int portID, int portName, int maxMessage_size, bool portDirection, int refreshPeriod)

The documentation for this class was generated from the following files:

- /home/william/ARINC653v2/sources/include/Cport_service.h
- /home/william/ARINC653v2/sources/simulateur/Cport_service.cpp

## 4.17 CQueuing Class Reference

### Public Member Functions

- **CQueuing** (std::string name, int portID, int numSocket)
- int **WRITE_QUEUING_MESSAGE** (char ∗name, int portId, int sock, std::string emetteur, std::string addr_-message)
- int **READ_QUEUING_MESSAGE** (int sock)
- void **Display_Message** ()
- void **Trace_Message** (Type_Message ∗msg)
- Type_Message **get_Message** ()

The documentation for this class was generated from the following files:

- /home/william/ARINC653v2/sources/include/CQueuing.h
- /home/william/ARINC653v2/sources/common/CQueuing.cpp

## 4.18 CSampling Class Reference

### Public Member Functions

- **CSampling** (std::string name, int portID, int numSocket)
- int **WRITE_SAMPLING_MESSAGE** (char ∗name, int portId, int sock, std::string emetteur, std::string addr_-message)
- int **READ_SAMPLING_MESSAGE** (int sock)
- void **Display_Message** ()
- Type_Message **get_Message** ()

The documentation for this class was generated from the following files:

- /home/william/ARINC653v2/sources/include/CSampling.h
- /home/william/ARINC653v2/sources/common/CSampling.cpp

## 4.19 geolocation Struct Reference

Structure containing the geolocation of a picture.

### Public Attributes

- char **LatitudeRef** [1]
- int **Latitude** [6]
- char **LongitudeRef** [1]
- int **Longitude** [6]

### 4.19.1 Detailed Description

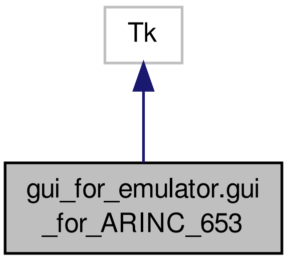
Structure containing the geolocation of a picture.

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

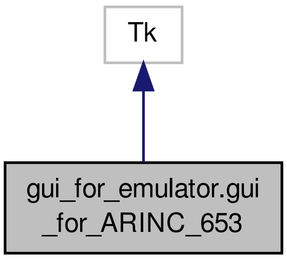
- /home/william/ARINC653v2/sources/Chaire_SE_Student/Leica/Leica.cpp
- /home/william/ARINC653v2/sources/Chaire_SE_Student/Scao/Scao.cpp

## 4.20 gui_for_emulator.gui_for_ARINC_653 Class Reference

Inheritance diagram for gui_for_emulator.gui_for_ARINC_653:



Collaboration diagram for gui_for_emulator.gui_for_ARINC_653:



### Public Member Functions

- def **__init__**
- def **initialisation**
- def **affiche_commande**
- def **click_python**
- def **readfunction**

### Public Attributes

- **parent**
- labelVariable

  *Creation d'une etiquette inseree dans 'menu_bottom' etiquette = Label(menu_bottom,text = 'Commande : ') etiquette.grid(row=1,column=1,sticky='w')*
- **backcolorlabel**
- **frontcolorlabel**

#### Static Public Attributes

- tuple **sortieTube** = os.open(input_pipe,os.O_RDONLY)
- tuple **entreeTube** = os.open(output_pipe,os.O_WRONLY)
- tuple **output_pipe** = tkFileDialog.askopenfilename(title='Choose the output pipe to open', defaultextension ='.fifo')

### 4.20.1 Member Data Documentation

#### 4.20.1.1 gui_for_emulator.gui_for_ARINC_653.labelVariable

Creation d'une etiquette inseree dans 'menu_bottom' etiquette = Label(menu_bottom,text = 'Commande : ') etiquette.grid(row=1,column=1,sticky='w')

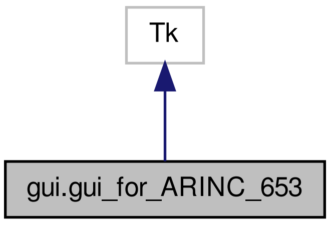
Creation d'un boutton dans 'menu_bottom' b = Button(menu_bottom,text='execute',command=self.affiche_-commande) b.grid(row=1,column=3,sticky='e') Creation d'un boite d'edition dans 'nenu_top' entre = Entry(menu_-bottom,width=20,relief='sunken') entre.insert(END,'Texte a afficher') entre.grid(row=1,column=2,sticky='ew')

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

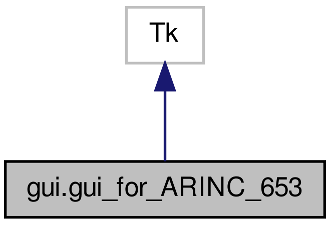
- /home/william/ARINC653v2/sources/GUI/gui_for_emulator.py

## 4.21 gui.gui_for_ARINC_653 Class Reference

Inheritance diagram for gui.gui_for_ARINC_653:



Collaboration diagram for gui.gui_for_ARINC_653:



#### Public Member Functions

- def **__init__**
- def **initialisation**
- def **readfunction**

#### Public Attributes

- **parent**

#### Static Public Attributes

- tuple **sortieTube** = os.open(input_pipe,os.O_RDONLY)
- tuple **entreeTube** = os.open(output_pipe,os.O_WRONLY)
- tuple **output_pipe** = tkFileDialog.askopenfilename(title='Choose the output pipe to open', defaultextension ='.fifo')
- tuple **ScreenSizeX** = self.winfo_screenwidth()
- tuple **ScreenSizeY** = self.winfo_screenheight()
- int **CorrectionX** = 30
- int **CorrectionY** = 30
- tuple **FrameSizeX** = int(ScreenSizeX ∗ 0.5)
- tuple **FrameSizeY** = int(ScreenSizeY ∗ 0.5)
- int **FramePosX** = 0
- int **FramePosY** = 0

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

- /home/william/ARINC653v2/sources/GUI/gui.py

## 4.22 Horloge Class Reference

Emulates the clock used by a partition.

```
#include <Horloge.h>
```

#### Public Member Functions

- double get_temps ()
  *Returns the time since the big_bang value in seconds.*
- double date2seconds (struct tm p_temps)
  *Transforms a tm structure to a date in seconds.*

#### Protected Attributes

- time_t big_bang

### 4.22.1 Detailed Description

Emulates the clock used by a partition.

### 4.22.2 Member Function Documentation

#### 4.22.2.1 double Horloge::date2seconds ( struct tm *p_temps* )

Transforms a tm structure to a date in seconds.

**Parameters**

| | |
|---|---|
| *tm* | timer containing the date. |

**Returns**

Relative date in seconds.

#### 4.22.2.2 double Horloge::get_temps ( )

Returns the time since the big_bang value in seconds.

**Returns**

Relative date in second.

### 4.22.3 Member Data Documentation

#### 4.22.3.1 time_t Horloge::big_bang `[protected]`

Origin of time for the simulator

The documentation for this class was generated from the following files:

- /home/william/ARINC653v2/sources/Chaire_SE_Student/Horloge.h
- /home/william/ARINC653v2/sources/Chaire_SE_Student/Horloge.cpp

## 4.23 Manager Class Reference

Used in Master and Slave as the mission manager object. It regroups all the functionnalities and the objects necessary to manage the mission.

`#include <Manager.h>`

### Public Member Functions

- void set_mode (string)

  *Mutator on the mode.*
- void set_partition (string)

  *Mutator on partition name.*
- void send_check (Checkpoint)

  *Send a checkpoint to the backup partition.*
- void actualiser_table (int, int, bool)

  *Refresh checkpoint table.*
- void actualiser_table (Checkpoint)

  *Refresh checkpoint table.*
- int table_id_image_to_position (int)

  *Find the index in the checkpoint table from an image id.*
- int table_position_to_id_image (int)

  *Find the image from its index in the checkpoint table.*
- Checkpoint table_get_check (int)

  *Get a checkpoint from its index in the table.*
- Checkpoint recuperer_dernier_check ()

  *Get the last checkpoint updated.*

- int gerer_mission (string, string, Horloge ∗, Analyseur ∗, Memoire_stable ∗, CarteComm ∗, CarteRecep ∗, COMMUNICATION_VECTOR)

  *Mission manager function.*
- void init_plan (Plan)

  *Replace the current plan by a new one.*
- int trouver_action (int)

  *Find an action in the current plan.*
- int order_transfer (bool ∗, char ∗, double, double, double)

  *Send a transmission order to the Leica partition.*
- int receive_transfer_report (bool ∗, char ∗, double, double, double)

  *Get the report of the Leica partition.*
- void Creer_Plan (int, int, string)

  *Create a new Plan.*
- int capturer_image (char ∗, int, int, int, double, double, bool)

  *Send order to Leica to take a picture.*
- int orienter_satellite (float, float, float, double, double, bool)

  *Send positionning orders to SCAO.*

### 4.23.1 Detailed Description

Used in Master and Slave as the mission manager object. It regroups all the functionnalities and the objects necessary to manage the mission.

### 4.23.2 Member Function Documentation

#### 4.23.2.1 void Manager::actualiser_table ( int *id_image,* int *num_etat,* bool *val_etat* )

Refresh checkpoint table.

**Parameters**

| | |
|---|---|
| *int* | Image id of the checkpoint |
| *int* | Status index |
| *bool* | New state value |

#### 4.23.2.2 void Manager::actualiser_table ( Checkpoint *nouveau_checkpoint* )

Refresh checkpoint table.

**Parameters**

| | |
|---|---|
| *Checkpoint* | Checkpoint to be updated |

#### 4.23.2.3 int Manager::capturer_image ( char ∗ *name,* int *largeur,* int *duree_image,* int *qualite_jpg,* double *date_action,* double *timeout,* bool *recovery* )

Send order to Leica to take a picture.

**Parameters**

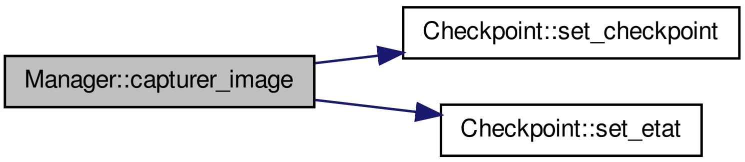
| | |
|---|---|
| *char∗* | name of the image |
| *int* | Weight of the image (px) |

| | |
|---:|---|
| *int* | Duration of the action (corresponding to the image length in px) |
| *int* | Jpg compression rate |
| *double* | Action deadline |
| *double* | Maximum execution time |
| *bool* | recovery mode (true or false) |

**Returns**

Error code for picture

Here is the call graph for this function:



#### 4.23.2.4 void Manager::Creer_Plan ( int *date_debut,* int *nb_instr,* string *fichier* )

Create a new Plan.

This is a test function to create a random plan.

**Parameters**

| | |
|---:|---|
| *int* | First action deadline delay |
| *int* | Number of instruction |
| *string* | Name of file |

Ouverture en écriture Plan.txt

#### 4.23.2.5 int Manager::gerer_mission ( string , string , Horloge ∗ , Analyseur ∗ , Memoire_stable ∗ , CarteComm ∗ , CarteRecep ∗ , COMMUNICATION_VECTOR )

Mission manager function.

This function uses all the methods to execute the mission. Contains functionnal and recovery functionalities

**Parameters**

| | |
|---:|---|
| *string* | : Partition name |
| *string* | : Partition mode |
| *Horloge∗* | Clock |
| *Analyseur∗* | Image analyser |
| *Memoire_-stable∗* | Image memory |
| *CarteComm∗* | Image transmission device |
| *CarteRecep∗* | Plan transmission device |

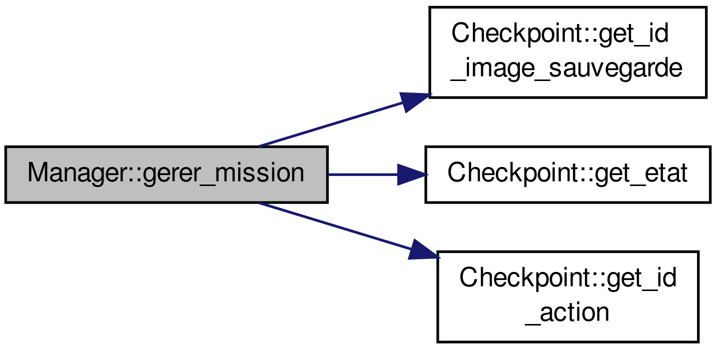
| | |
|---|---|
| *COMMUNICATI-ON_VECTOR* | Communication information vector |

**Returns**

0 in case of error.

Lecture de la prochaine action du plan

Here is the call graph for this function:


Checkpoint::get_id _image_sauvegarde
Manager::gerer_mission
Checkpoint::get_etat
Checkpoint::get_id _action

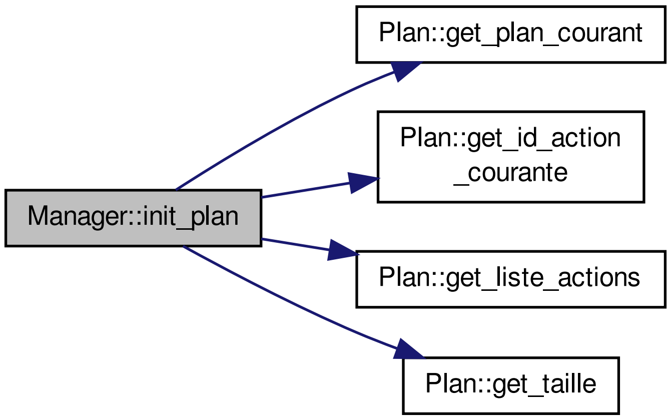
#### 4.23.2.6 void Manager::init_plan ( Plan *pl* )

Replace the current plan by a new one.

**Parameters**

| | |
|---|---|
| *Plan* | New plan to be treated |

Here is the call graph for this function:



#### 4.23.2.7 int Manager::order_transfer ( bool ∗ *order_status,* char ∗ *name,* double *date_action,* double *duration,* double *timeout* )

Send a transmission order to the Leica partition.

**Parameters**

| | |
|---:|---|
| *bool∗* | Order status (realised or not) |
| *char∗* | Image name |
| *double* | Action deadline |
| *double* | Action duration |
| *double* | Maximum execution time. |

**Returns**

Error code for image transmission.

#### 4.23.2.8 int Manager::orienter_satellite ( float *command_pitch,* float *command_roll,* float *command_yaw,* double *date_action,* double *timeout,* bool *recovery* )

Send positionning orders to SCAO.

**Parameters**

| | |
|---:|---|
| *float* | : Pitch value |
| *float* | : Roll value |
| *float* | : Yaw value |
| *double* | Action deadline |
| *double* | Maximum execution time |
| *bool* | recovery mode (true or false) |

**Returns**

Error code for positionning

#### 4.23.2.9 int Manager::receive_transfer_report ( bool ∗ *order_status,* char ∗ *name,* double *date_action,* double *duration,* double *timeout* )

Get the report of the Leica partition.

**Parameters**

| | |
|---:|---|
| *bool∗* | Order status (realised or not) |
| *char∗* | Image name |
| *double* | Action deadline |
| *double* | Action duration |
| *double* | Maximum execution time. |

**Returns**

Error code for image transmission.

#### 4.23.2.10 Checkpoint Manager::recuperer_dernier_check ( )

Get the last checkpoint updated.

**Returns**

The last checkpoint updated

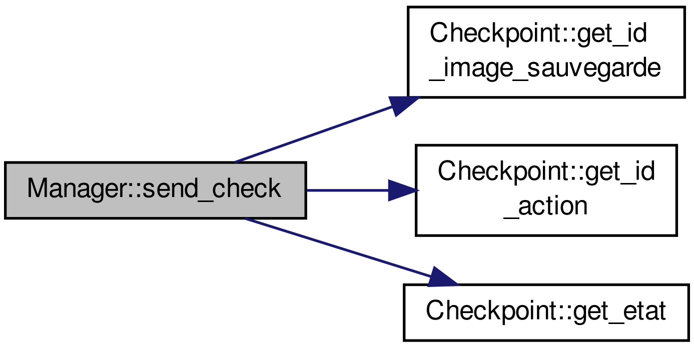
#### 4.23.2.11 void Manager::send_check ( Checkpoint *check* )

Send a checkpoint to the backup partition.

**Parameters**

| | |
|---:|---|
| *Checkpoint* | Checkpoint to be send. |

Here is the call graph for this function:


Checkpoint::get_id _image_sauvegarde
Manager::send_check
Checkpoint::get_id _action
Checkpoint::get_etat

#### 4.23.2.12 void Manager::set_mode ( string *smode* )

Mutator on the mode.

**Parameters**

| | |
|---|---|
| *string* | New mode value. |

#### 4.23.2.13 void Manager::set_partition ( string *myPart* )

Mutator on partition name.

**Parameters**

| | |
|---|---|
| *string* | New partition name |

#### 4.23.2.14 Checkpoint Manager::table_get_check ( int *i* )

Get a checkpoint from its index in the table.

**Parameters**

| | |
|---|---|
| *int* | Table index |

**Returns**

Checkpoint at the indexed position

#### 4.23.2.15 int Manager::table_id_image_to_position ( int *i* )

Find the index in the checkpoint table from an image id.

**Parameters**

| | |
|---|---|
| *int* | Image id |

**Returns**

Index in the table

#### 4.23.2.16 int Manager::table_position_to_id_image ( int *i* )

Find the image from its index in the checkpoint table.

**Parameters**

| | |
|---|---|
| *int* | Index in the table |

**Returns**

Image Id

#### 4.23.2.17 int Manager::trouver_action ( int *i* )

Find an action in the current plan.

**Parameters**

| | |
|---|---|
| *int* | action index |

**Returns**

-1 in case of error,else current action id

The documentation for this class was generated from the following files:

- /home/william/ARINC653v2/sources/Chaire_SE_Student/Manager.h
- /home/william/ARINC653v2/sources/Chaire_SE_Student/Manager.cpp

## 4.24 Memoire_stable Class Reference

Image memory.

`#include <Memoire_stable.h>`

#### Public Member Functions

- void stocker (image)

  *Copy the images from the temporary memory to the stable memory.*
- void vider (image)

  *Deletes an image in the memory.*
- void purger ()

  *Deletes all the images stored.*

### 4.24.1 Detailed Description

Image memory.

This class emulates the hardware memory used to store the image took by the camera. It could be considered as a descriptor table for the storage of the images. In the simulator this class uses bash command "cp" to send the images to an other station (connected by ethernet or wifi).

### 4.24.2 Member Function Documentation

#### 4.24.2.1 int Memoire_stable::stocker ( image )

Copy the images from the temporary memory to the stable memory.

**Parameters**

| | |
|---|---|
| *image* | Image to store. |

#### 4.24.2.2 void Memoire_stable::vider ( image *suppr* )

Deletes an image in the memory.

**Parameters**

| | |
|---|---|
| | Image to be deleted. |

The documentation for this class was generated from the following files:

- /home/william/ARINC653v2/sources/Chaire_SE_Student/Memoire_stable.h
- /home/william/ARINC653v2/sources/Chaire_SE_Student/Memoire_stable.cpp

## 4.25 order Struct Reference

Structure of the order from managers partition.

#### Public Attributes

- char **nature** [ORDER_NATURE_LENGTH]
- char **id_image** [IMAGE_ID_LENGTH]
- double **order_date**
- int **larg_px**

- double **duration**
- int **qualite**

#### 4.25.1 Detailed Description

Structure of the order from managers partition.

Structure of geolocation informations.

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

- /home/william/ARINC653v2/sources/Chaire_SE_Student/Leica/Leica.cpp

## 4.26 orientation Struct Reference

Structure containing the attitude of the satellite.

#### Public Attributes

- float **yaw**
- float **pitch**
- float **roll**

#### 4.26.1 Detailed Description

Structure containing the attitude of the satellite.

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

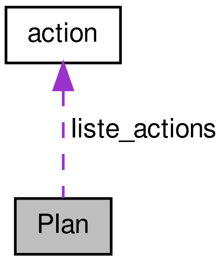
- /home/william/ARINC653v2/sources/Chaire_SE_Student/Scao/Scao.cpp

## 4.27 Plan Class Reference

Imports and stocks the instructions.

`#include <Plan.h>`

Collaboration diagram for Plan:



#### Public Member Functions

- int set_plan ()

    *Updates the plan.*
- int set_plan_recouvrement ()

    *Updates the plan in case of recovery.*
- action next_action ()

    *Updates the next action.*
- int get_id_action_courante ()

    *Accessor on the current action id.*
- action get_action (int)

    *Action assessor.*
- void set_id_action_courante (int)

    *Mutator id_action_suivante.*
- int get_taille ()

    *Accessor taille_plan.*
- void set_taille (int)

    *Mutator taille_plan.*
- void set_plan_courant (char ∗)

    *Mutatro plan_courant.*
- char ∗ get_plan_courant ()

    *Assessor plan_courant.*
- void set_action (action, int)

    *Modify an action in the list.*
- void set_liste_actions (action ∗)

    *Mutator liste_actions.*
- action ∗ get_liste_actions ()

    *Assessor liste_actions.*
- void afficher_action (action)

    *Display function.*

#### Protected Attributes

- char plan_courant [30]
- int id_action_courante
- action liste_actions [50]
- int taille_plan

### 4.27.1 Detailed Description

Imports and stocks the instructions.

It manages the importation and the use of the instructions from the .txt file send by the on-earth station.

### 4.27.2 Member Function Documentation

#### 4.27.2.1 void Plan::afficher_action ( action *action_a_afficher* )

Display function.

Display all the members of an action

**Parameters**

| | |
|---|---|
| *action* | the action to be displayed |

#### 4.27.2.2 action Plan::get_action ( int *numero* )

Action assessor.

Returns the action corresponding to the id sent in parameter.

**Parameters**

| | |
|---|---|
| *int* | Action id |

**Returns**

Action matching the id.

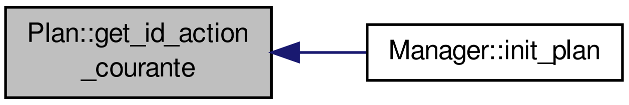
#### 4.27.2.3 int Plan::get_id_action_courante ( )

Accessor on the current action id.

**Returns**

Current action id.

Here is the caller graph for this function:



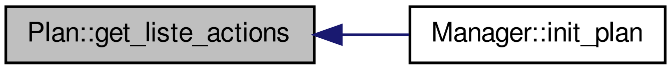
#### 4.27.2.4 action ∗ Plan::get_liste_actions ( )

Assessor liste_actions.

**Returns**

A pointer to a list of action

Here is the caller graph for this function:



#### 4.27.2.5 char ∗ Plan::get_plan_courant ( )

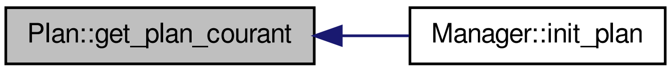
Assessor plan_courant.

Returns the name of the current plan

**Returns**

name of the plan

Here is the caller graph for this function:



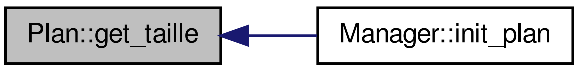
#### 4.27.2.6 int Plan::get_taille ( )

Accessor taille_plan.

**Returns**

taille_plan

Here is the caller graph for this function:



#### 4.27.2.7 action Plan::next_action ( )

Updates the next action.

Returns the next action and update the index in the plan.

**Returns**

Next action to be done.

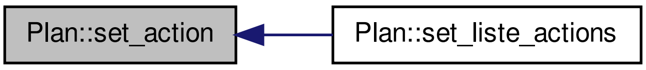
#### 4.27.2.8 void Plan::set_action ( action *nouvelle_action,* int *rang* )

Modify an action in the list.

**Parameters**

| | |
|---|---|
| *action* | new action |
| *int* | index in the plan |

Here is the caller graph for this function:



#### 4.27.2.9 void Plan::set_id_action_courante ( int *numero* )

Mutator id_action_suivante.

Changes the value of id_action_suivante.

**Parameters**

| | |
|---|---|
| *int* | new value of id_action_suivante |

#### 4.27.2.10 void Plan::set_liste_actions ( action ∗ *nouvelle_liste* )

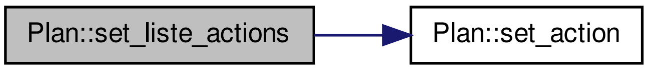
Mutator liste_actions.

Modify the whole list of action

**Parameters**

| | |
|---|---|
| *action∗* | pointer to the new list |

Here is the call graph for this function:



#### 4.27.2.11 int Plan::set_plan ( )

Updates the plan.

Updates the plan from an existing .txt file. If there's no new plan it keeps the old one, if the old one is finished then it switchs on the default plan. The new plan shall named NewPlan.txt and will be renamed OldPlan.txt.

**Returns**

-1: uploads the default plan. 0: uploads the new one. 1: keeps the old one. 2: keeps the old one in case of a recovery. 3: in case of error.

#### 4.27.2.12 void Plan::set_plan_courant ( char ∗ *nouveau_fichier* )

Mutatro plan_courant.

Changes the name of the plan currently used.

**Parameters**

| | |
|---|---|
| *char∗* | new name of plan |

#### 4.27.2.13 int Plan::set_plan_recouvrement ( )

Updates the plan in case of recovery.

**Returns**

-1 for error, 1 for success

#### 4.27.2.14 void Plan::set_taille ( int *nouvelle_taille* )

Mutator taille_plan.

**Parameters**

| | |
|---|---|
| *int* | new value of taille_plan |

### 4.27.3 Member Data Documentation

#### 4.27.3.1 int Plan::id_action_courante `[protected]`

Index of the action in the plan

#### 4.27.3.2 action Plan::liste_actions[50] `[protected]`

List of the actions to be done

#### 4.27.3.3 char Plan::plan_courant[30] `[protected]`

Name of the current plan

#### 4.27.3.4 int Plan::taille_plan `[protected]`

Length of the plan

The documentation for this class was generated from the following files:

- /home/william/ARINC653v2/sources/Chaire_SE_Student/Plan.h
- /home/william/ARINC653v2/sources/Chaire_SE_Student/Plan.cpp

## 4.28 RebootFlag Class Reference

**Public Member Functions**

- void inc ()

  *Increases the flag value.*
- void dec ()

  *Decreases flag_value.*
- int get_flag_value ()

  *Accessor: flag_value.*

- void set_flag_limit (int)

  *Mutator flag_limit.*
- void set_partition_pid (int)

  *Mutator partition_pid.*
- bool flag ()

  *Compares flag_value to flag_limit.*
- void reboot ()

  *Reboots the partition.*

#### Protected Attributes

- int **flag_value**
- int **flag_limit**
- int **partition_pid**

### 4.28.1 Member Function Documentation

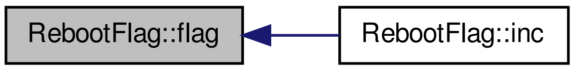
#### 4.28.1.1 bool RebootFlag::flag ( )

Compares flag_value to flag_limit.

**Returns**

The result of flag_value $<$ flag_limit

Here is the caller graph for this function:



#### 4.28.1.2 int RebootFlag::get_flag_value ( )

Accessor: flag_value.

**Returns**

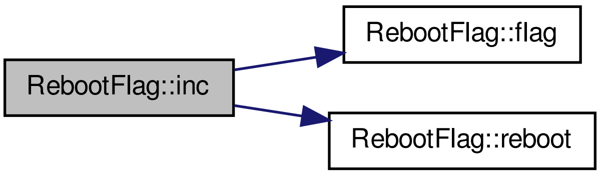
flag_value

#### 4.28.1.3 void RebootFlag::inc ( )

Increases the flag value.

Each time this function is called, the flag_value is increased and controlled, if it's over the flag_limit (check by calling flag()) it calls the reboot() function to kill the partition process.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- /home/william/ARINC653v2/sources/Chaire_SE_Student/RebootFlag.h
- /home/william/ARINC653v2/sources/Chaire_SE_Student/RebootFlag.cpp

## 4.29 SAMPLING_PORT_STATUS_TYPE Struct Reference

### Public Attributes

- int **MAX_MESSAGE_SIZE**
- bool **PORT_DIRECTION**
- int **REFRESH_PERIOD**
- int **LAST_MSG_VALIDITY**

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

- /home/william/ARINC653v2/sources/include/CCommunication.h

## 4.30 Table Class Reference

Buffer of checkpoints.

```
#include <Table.h>
```

### Public Member Functions

- void afficher ()

  *Prints the checkpoint table.*
- int maj (Checkpoint)

  *Add a checkpoint to the list.*
- int maj (int, int, bool)

  *Modify a checkpoint with the image id and the state to be modified.*
- int get_num_check ()

  *Accessor : id of the last cherckpoint.*
- Checkpoint get_dernier_check ()

  *Accessor : last checkpoint stored.*
- int id_image_to_position (int)

  *Get the index of an image in the checkpoint table.*

- int position_to_id_image (int)

    *Get the id of an image from its position in the table.*
- Checkpoint get_check (int)

    *Get the checkpoint of an image from its id.*

#### 4.30.1 Detailed Description

Buffer of checkpoints.

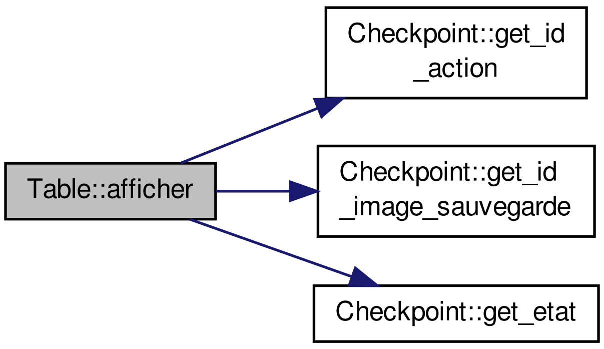
#### 4.30.2 Member Function Documentation

##### 4.30.2.1 void Table::afficher ( )

Prints the checkpoint table.

This method is used to print the checkpoint list

Here is the call graph for this function:



##### 4.30.2.2 Checkpoint Table::get_check ( int *position* )

Get the checkpoint of an image from its id.

**Parameters**

| | |
|---|---|
| *int* | image id |

**Returns**

checkpoint of the image

##### 4.30.2.3 Checkpoint Table::get_dernier_check ( )

Accessor : last checkpoint stored.

**Returns**

last checkpoint stored

#### 4.30.2.4 int Table::get_num_check ( )

Accessor : id of the last cherckpoint.

**Returns**

id of the last checkpoint

#### 4.30.2.5 int Table::id_image_to_position ( int *id_image* )

Get the index of an image in the checkpoint table.

**Parameters**

| | |
|---|---|
| *int* | image id |

**Returns**

index of the image in the table

#### 4.30.2.6 int Table::maj ( Checkpoint *checkpoint* )

Add a checkpoint to the list.

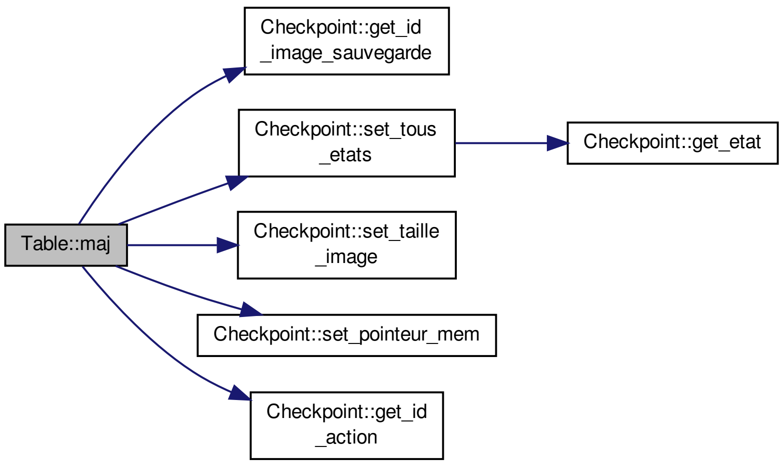
**Parameters**

| | |
|---|---|
| *checkpoint* | checkpoint to be added |

**Returns**

1 if operation was successful, 0 if not

Here is the call graph for this function:


Checkpoint::get_id _image_sauvegarde
Checkpoint::set_tous _etats
Checkpoint::get_etat
Table::maj
Checkpoint::set_taille _image
Checkpoint::set_pointeur_mem
Checkpoint::get_id _action

#### 4.30.2.7 int Table::maj ( int *id_image,* int *num_etat,* bool *val_etat* )

Modify a checkpoint with the image id and the state to be modified.

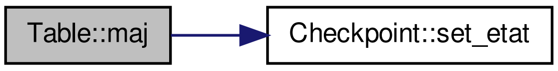
**Parameters**

| | |
|---|---|
| *int* | id of the image |
| *int* | id of state of the image to be modified |
| *bool* | new state |

**Returns**

1 if operation was successful 0 if not

Here is the call graph for this function:



#### 4.30.2.8 int Table::position_to_id_image ( int *position* )

Get the id of an image from its position in the table.

**Parameters**

| | |
|---|---|
| *int* | index |

**Returns**

image id

The documentation for this class was generated from the following files:

- /home/william/ARINC653v2/sources/Chaire_SE_Student/Table.h
- /home/william/ARINC653v2/sources/Chaire_SE_Student/Table.cpp

## 4.31 Type_Message Struct Reference

**Public Attributes**

- char **m_sender** [MSG_LENGTH]
- int **m_length**
- char **m_message** [MSG_LENGTH]

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

- /home/william/ARINC653v2/sources/include/CCommunication.h

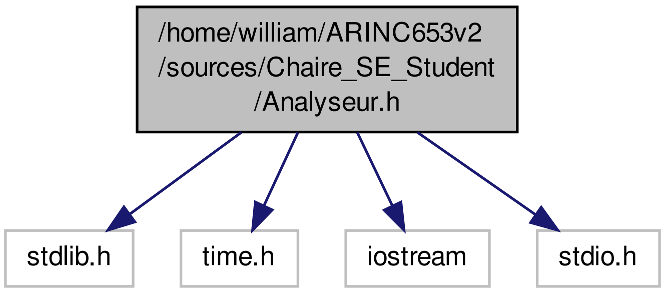
# 5 File Documentation

## 5.1 /home/william/ARINC653v2/sources/Chaire_SE_Student/Analyseur.h File Reference

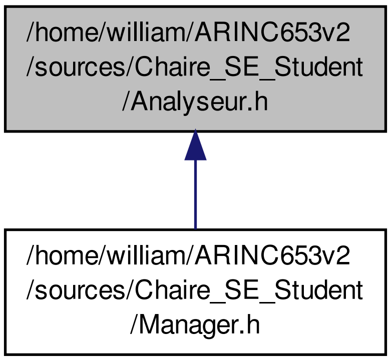
Analyse and compress the images from the camera.

```
#include <stdlib.h>
#include <time.h>
#include <iostream>
#include <stdio.h>
```

Include dependency graph for Analyseur.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class Analyseur

    *Analyses and compress the images from the camera.*

### 5.1.1 Detailed Description

Analyse and compress the images from the camera.

**Version**

2.0

**Author**

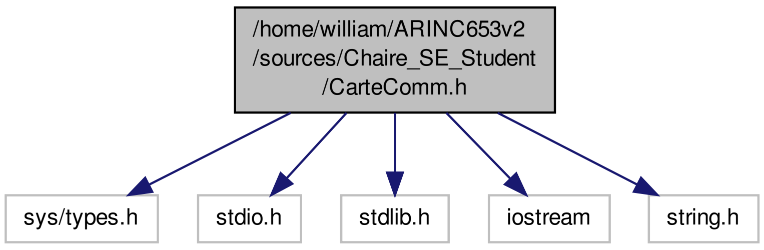
Lucie BEAUSSART Thomas BETOUS Abdelkader BOUARFA William EXCOFFON

## 5.2 /home/william/ARINC653v2/sources/Chaire_SE_Student/CarteComm.h File Reference

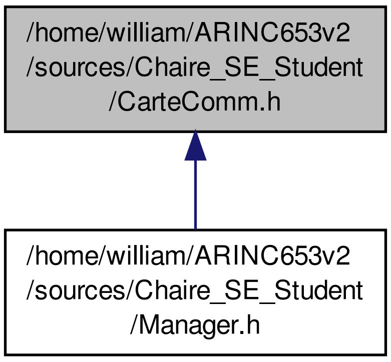
Image transmission device.

```
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <string.h>
```

Include dependency graph for CarteComm.h:



This graph shows which files directly or indirectly include this file:



**Classes**

- class CarteComm

*Emulates a TM controller.*

### 5.2.1 Detailed Description

Image transmission device.

**Version**

2.0

**Author**

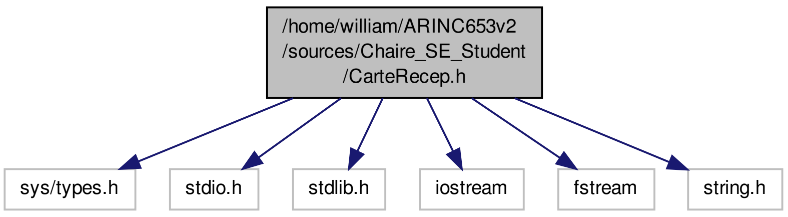
Lucie BEAUSSART Thomas BETOUS Abdelkader BOUARFA William EXCOFFON

## 5.3 /home/william/ARINC653v2/sources/Chaire_SE_Student/CarteRecep.h File Reference

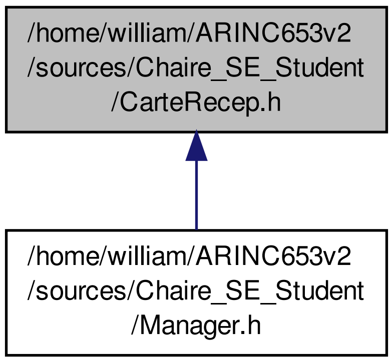
Plan reception device.

```
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <string.h>
```

Include dependency graph for CarteRecep.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class CarteRecep

### 5.3.1 Detailed Description

Plan reception device.

**Version**

2.0

**Author**

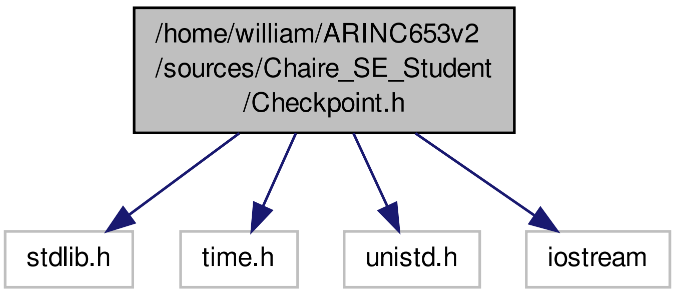
Lucie BEAUSSART Thomas BETOUS Abdelkader BOUARFA William EXCOFFON

## 5.4 /home/william/ARINC653v2/sources/Chaire_SE_Student/Checkpoint.h File Reference

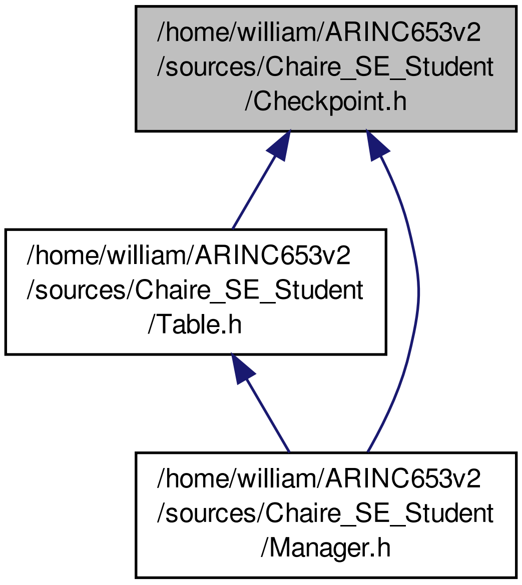
Backup structure to save the states of each images.

```
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
#include <iostream>
```

Include dependency graph for Checkpoint.h:



This graph shows which files directly or indirectly include this file:


/home/william/ARINC653v2 /sources/Chaire_SE_Student /Checkpoint.h
/home/william/ARINC653v2 /sources/Chaire_SE_Student /Table.h
/home/william/ARINC653v2 /sources/Chaire_SE_Student /Manager.h

**Classes**

- class Checkpoint

  *Creation and management of the checkpoints.*

### 5.4.1 Detailed Description

Backup structure to save the states of each images.

**Version**

2.0

**Author**

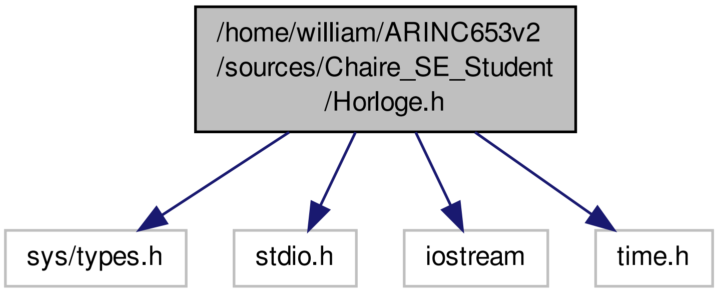
Lucie BEAUSSART Thomas BETOUS Abdelkader BOUARFA William EXCOFFON

## 5.5 /home/william/ARINC653v2/sources/Chaire_SE_Student/Horloge.h File Reference

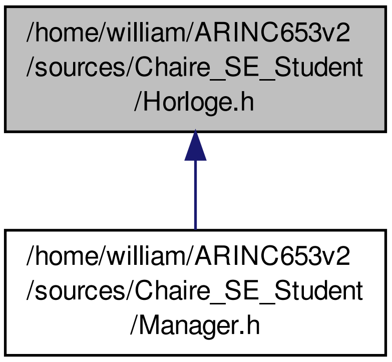
Emulate the clock used by a partition.

```
#include <sys/types.h>
#include <stdio.h>
#include <iostream>
#include <time.h>
```

Include dependency graph for Horloge.h:



This graph shows which files directly or indirectly include this file:



**Classes**

- class Horloge

  *Emulates the clock used by a partition.*

### 5.5.1 Detailed Description

Emulate the clock used by a partition.

**Version**

2.0

**Author**

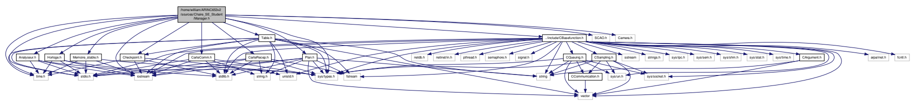
Lucie BEAUSSART Thomas BETOUS Abdelkader BOUARFA William EXCOFFON

## 5.6 /home/william/ARINC653v2/sources/Chaire_SE_Student/Manager.h File Reference

Backup structure to save the states of each images.

```
#include "../include/CBasefunction.h"
#include "Analyseur.h"
#include "SCAO.h"
#include "Horloge.h"
#include "Plan.h"
#include "Camera.h"
#include "Memoire_stable.h"
#include "Table.h"
#include "Checkpoint.h"
#include "CarteComm.h"
#include "CarteRecep.h"
#include <sys/types.h>
#include <stdio.h>
#include <iostream>
#include <time.h>
```

Include dependency graph for Manager.h:



### Classes

- class Manager

  *Used in Master and Slave as the mission manager object. It regroups all the functionnalities and the objects necessary to manage the mission.*

### Macros

- #define TIMEOUT_IMG 10

  *Timeout limit for image acquisition (in s).*
- #define TIMEOUT_TSF 10

  *Timeout limit for image transmission (in s).*
- #define TIMEOUT_SCAO 10

  *Timeout limit for positionning (in s).*

### 5.6.1 Detailed Description

Backup structure to save the states of each images.

**Version**

2.0

**Author**

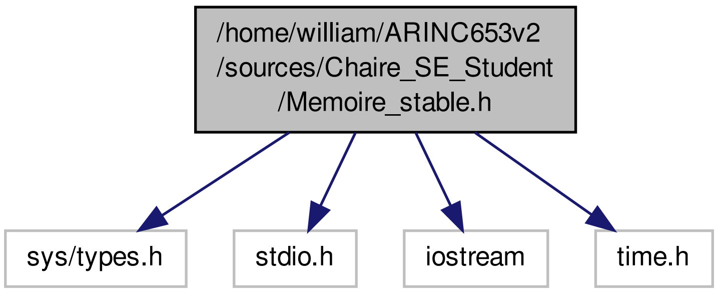
Lucie BEAUSSART Thomas BETOUS Abdelkader BOUARFA William EXCOFFON

## 5.7 /home/william/ARINC653v2/sources/Chaire_SE_Student/Memoire_stable.h File Reference

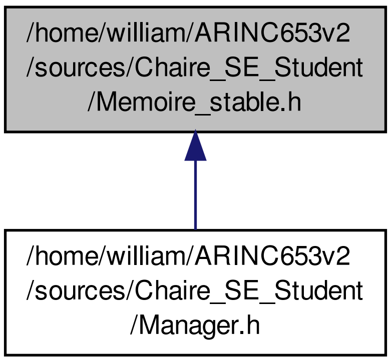
Emulate the memory used for image storage.

```
#include <sys/types.h>
#include <stdio.h>
#include <iostream>
#include <time.h>
```

Include dependency graph for Memoire_stable.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class Memoire_stable

  *Image memory.*

### 5.7.1 Detailed Description

Emulate the memory used for image storage.

**Version**

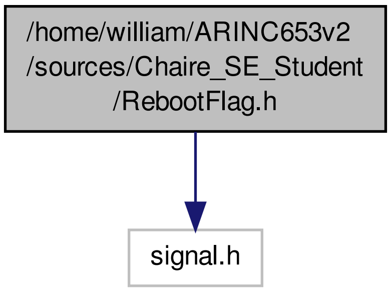
2.0

**Author**

Lucie BEAUSSART Thomas BETOUS Abdelkader BOUARFA William EXCOFFON

## 5.8 /home/william/ARINC653v2/sources/Chaire_SE_Student/RebootFlag.h File Reference

`#include "signal.h"`
Include dependency graph for RebootFlag.h:



**Classes**

- class RebootFlag

### 5.8.1 Detailed Description

**Version**

2.0

**Author**

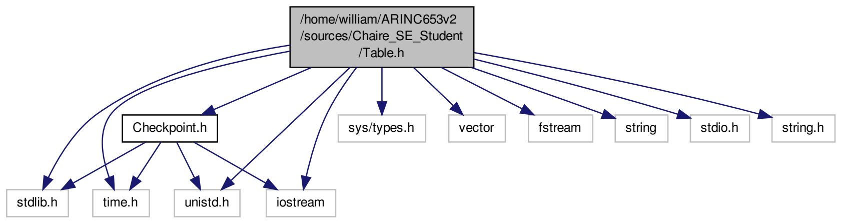
Lucie BEAUSSART Thomas BETOUS Abdelkader BOUARFA William EXCOFFON

### 5.9 /home/william/ARINC653v2/sources/Chaire_SE_Student/Table.h File Reference

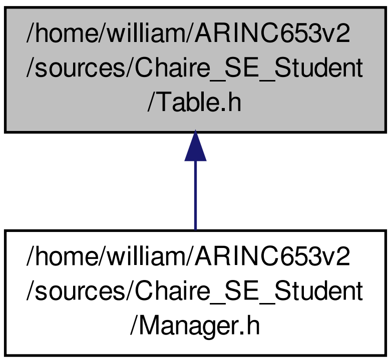
Storage object for instruction.

```
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
#include <sys/types.h>
#include <iostream>
#include <vector>
#include <fstream>
#include <string>
#include <stdio.h>
#include <string.h>
#include "Checkpoint.h"
```

Include dependency graph for Table.h:



This graph shows which files directly or indirectly include this file:



**Classes**

- class Table

*Buffer of checkpoints.*

### Macros

- #define TAILLE_TABLE 75

  *Size of the Checkpoint buffer.*

### 5.9.1 Detailed Description

Storage object for instruction. Backup structure saving the checkpoints.

**Version**

2.0

**Author**

Lucie BEAUSSART Thomas BETOUS Abdelkader BOUARFA William EXCOFFON

# Index